MC56F825x/4x Reference Manual

Supports the MC56F8257, MC56F8256, MC56F8255, MC56F8247, MC56F8246, and MC56F8245

Document Number: MC56F825XRM
Rev. 2, 10/2010

Preliminary

freescale semiconductor

Contents

Section Number **Title** **Page**

Chapter 1
Device Overview

Chapter 2
Analog-to-Digital Converter (ADC)

Chapter 3
High Speed Comparator (HSCMP)

Chapter 4
5-Bit Voltage Reference Digital-to-Analog Converter (VREF_DAC)

Chapter 5
12-Bit Digital-to-Analog Converter (DAC)

Chapter 6
Quad Timer (TMR)

Chapter 7
Enhanced Flex Pulse Width Modulator (eFlexPWM)

Chapter 8
Crossbar Switch (XBAR)

Chapter 9
General-Purpose Input/Output (GPIO)

Chapter 10
Inter-Integrated Circuit (I2C)

Chapter 11
Queued Serial Communications Interface (QSCI)

Chapter 12
Queued Serial Peripheral Interface (QSPI)

Chapter 13
Freescale's Scalable Controller Area Network (MSCAN)

Chapter 14
Interrupt Controller (INTC)

Chapter 15
On-Chip Clock Synthesis (OCCS)

Chapter 16
System Integration Module (SIM)

Chapter 17
Power Supervisor (PS)

Chapter 18
Computer Operating Properly (COP)

Chapter 19
Cyclic Redundancy Check Generator (CRC)

Chapter 20
Flash Memory (HFM)

Chapter 21
Joint Test Action Group (JTAG) Port

Chapter 1
Device Overview

1.1 Introduction

The MC56F825x/MC56F824x is a member of the 56800E core-based family of Digital Signal Controllers (DSCs). It combines, on a single chip, the processing power of a DSP and the functionality of a microcontroller with a flexible set of peripherals to create a cost-effective solution. Because of its low cost, configuration flexibility, and compact program code, the MC56F825x/MC56F824x is well-suited for many consumer and industrial applications.

The 56800E core is based on a dual Harvard-style architecture consisting of three execution units operating in parallel, allowing as many as six operations per instruction cycle. The MCU-style programming model and optimized instruction set allow straightforward generation of efficient, compact DSP and control code. The instruction set is also highly efficient for C compilers to enable rapid development of optimized control applications.

1.2 Features

The MC56F825x/MC56F824x includes many peripherals that are especially useful for industrial control, motion control, home appliances, general purpose inverters, smart sensors, fire and security systems, switched-mode power supply, power management, and medical monitoring applications.

The MC56F825x/MC56F824x Digital Signal Controller includes up to 64 KB of program flash and up to 8 KB of unified data/program RAM. The page size of on-chip flash is 1024 words. It can be independently bulk erased or erased in pages.

A comprehensive set of programmable peripherals—including a PWM, dual high-speed ADCs, queued SCIs, a queued SPI, I2Cs, an MSCAN, an inter-module Crossbar Switch, Quad Timers, a CRC block, DACs, analog comparators, and on-chip and off-chip clock

sources—supports various applications. Each peripheral can be independently shut down to save the power. Any pin in these peripherals can also be used as General Purpose Input/Outputs (GPIOs).

Feature highlights follow:

- Highly efficient hybrid DSP core
  - 60 MHz operation frequency
  - Efficient 56800E Digital Signal Processor (DSP) engine with dual Harvard architecture
    - Three internal address buses
    - Four internal data buses
  - 155 basic instructions in conjunction with up to 20 address modes
  - 32-bit internal primary data buses supporting 8-bit, 16-bit, and 32-bit data movement, addition, subtraction, and logical operation
  - Single-cycle 16 x 16-bit parallel Multiplier-Accumulator (MAC)
  - Four 36-bit accumulators, including extension bits
  - 32-bit arithmetic and logic multi-bit shifter
  - Parallel instruction set with unique DSP addressing modes
  - Hardware DO and REP loops
  - Instruction set that supports both DSP and controller functions
  - Controller-style addressing modes and instructions for compact code
  - Efficient C compiler and local variable support
  - Software subroutine and interrupt stack with depth limited only by memory
  - JTAG/Enhanced On-Chip Emulation (EOnCE) for unobtrusive, processor-speed-independent, real-time debugging
- On-chip memory
  - Dual Harvard architecture that permits as many as three simultaneous accesses to program and data memory
  - Data RAM that supports byte, word, and double-word address modes
  - EEPROM emulation capability using flash
  - Flash security and protection that prevent unauthorized users from gaining access to the internal flash
- System and power management
  - Dual clock sources
    - External crystal, resonator, external clock source
    - Internal 8 MHz/400 kHz relaxation oscillator
  - On-Chip Clock Synthesis (OCCS) module
  - On-chip Power Reset and Brown-Out Reset
  - On-chip low voltage detection

- Computer Operating Properly (COP) watchdog timer with multi-clock source selection
- Wait and stop modes with fast wakeup
- Single supply

- Interrupt Controller (INTC)
  - Five interrupt priority levels
    - One non-maskable interrupt for core-related interrupts
    - Three user-programmable priority levels for each interrupt source: levels 0, 1, and 2
    - Lowest-priority software interrupt: LP
  - Support for hardware nested interrupts
  - Support for two programmable fast interrupts that can be assigned to any interrupt source

- Peripherals
  - One Enhanced Flex Pulse Width Modulator (eFlexPWM)
    - 6-channel with with NanoEdge™ placement; 520 ps PWM resolution
    - 3-channel with enhanced capture functionality
  - Two Quad Timers with a total of eight 16-bit timers and up to 120 MHz operation clock
  - Two 12-bit ADCs with up to 16 channels
  - One 12-bit DAC
  - Three analog High Speed Comparators (HSCMPs) with integrated digital output filtering and windowing features
  - Three 5-bit reference DACs (32-tap voltage reference) for HSCMPs
  - One Queued Serial Peripheral Interface (QSPI) module with 4-word FIFO
  - Two high speed Queued Serial Communication Interface (QSCI) modules with 4-byte FIFO and up to 120 MHz operation clock
  - Two Inter-Integrated Circuit (I2C) ports with SMBus support
  - One Scalable Controller Area Network (MSCAN) module
  - Cyclic Redundancy Check (CRC) Generator
  - Inter-module Crossbar Switch (XBAR) that provides the programmable internal module connections between and among the PWM, ADCs, Quad Timers, 12-bit DAC, HSCMPs, and package pins
  - Up to 54 general purpose I/O (GPIO)
    - 5 V tolerance
    - Configurable push-pull and open-drain output

- Ability to generate interrupt with programmable rising or falling edge and software interrupt
- Configurable drive strength: 4 mA/8 mA

1.3 Core Block Diagram

The DSC core is composed of several independent functional units. The program controller, address generation unit (AGU) and data arithmetic logic unit (ALU) contain their own register sets and control logic, allowing them to operate independently and in parallel, increasing throughput. There is also an independent bit manipulation unit, which provides for efficient bit-manipulation operations. Each functional unit interfaces with the other units, memory, and the memory-mapped peripherals over the core's internal address and data buses, and through the IP bus bridge (off core).

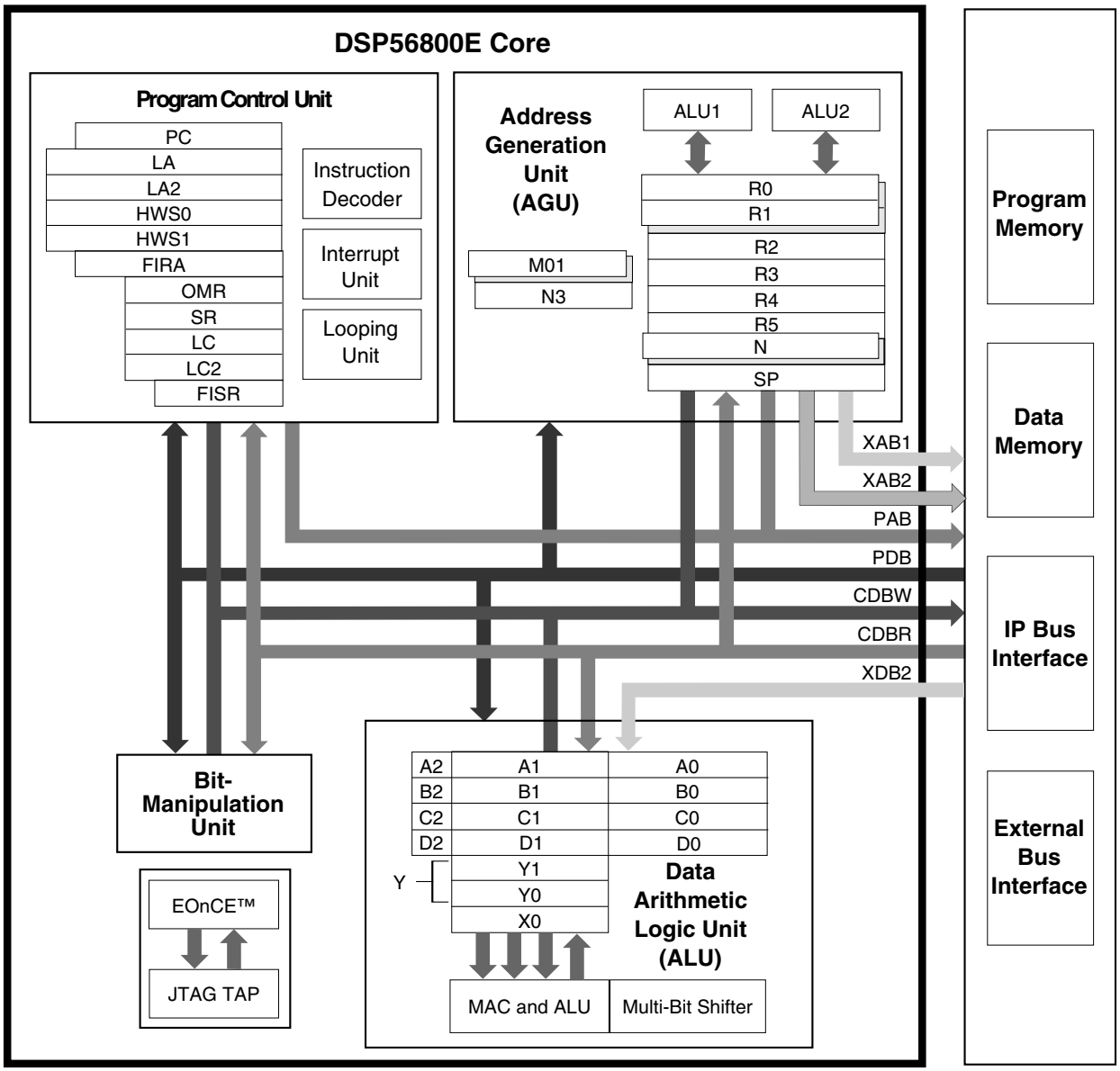


Figure 1-1. DSP56800E Core Block Diagram

1.4 Peripheral Subsystem

All functional pins on this device are multiplexed with one of the GPIO ports. To use one of the pins as a peripheral pin, enable the peripheral function by programming the corresponding bit in the GPIO port's peripheral enable register. If a pin is multiplexed with different peripheral functions, control the selection of the peripheral function by using one of the GPIO peripheral select registers in the System Integration Module (SIM).

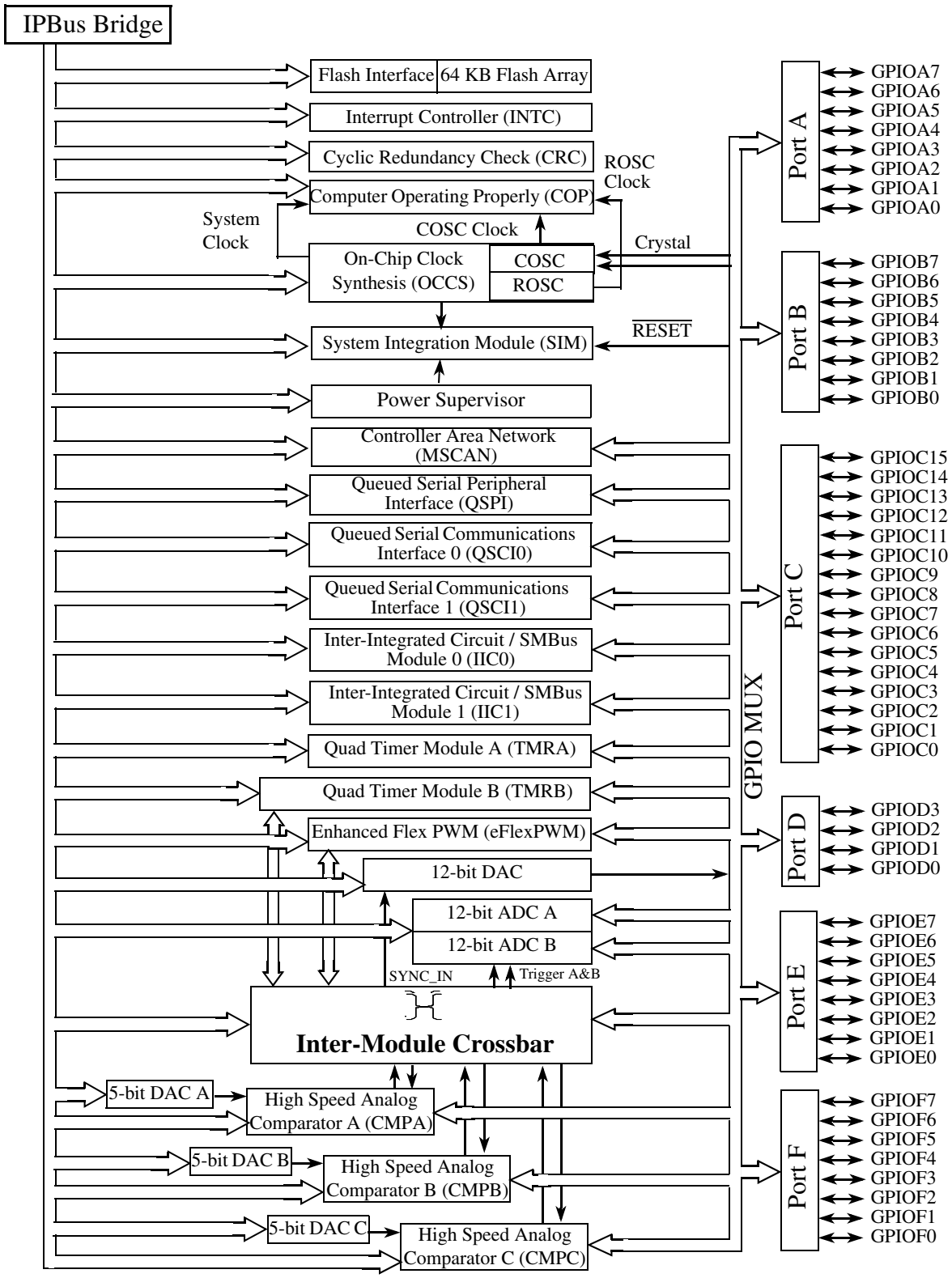


Figure 1-2. Peripheral Subsystem

1.5 Clock Generation and Distribution

The MC56F825x/MC56F824x's CPU and most peripherals can be run at clock rates up to (and including) 60 MHz. Additionally, the Serial Communication Interface (SCI) modules and quad timers can optionally be run at 120 MHz.

The device is equipped with an internal relaxation oscillator, with modes for 400 kHz and 8 MHz (default at reset). In addition, the crystal oscillator module supports external crystals and resonators ranging from 4 MHz to 16 MHz in operation.

NOTE

A crystal or resonator is recommended when using the MSCAN module.

The following diagram presents the system clock distribution. Additional clock information appears in descriptions of the On-Chip Clock Synthesis module (OCCS) and System Integration Module (SIM).

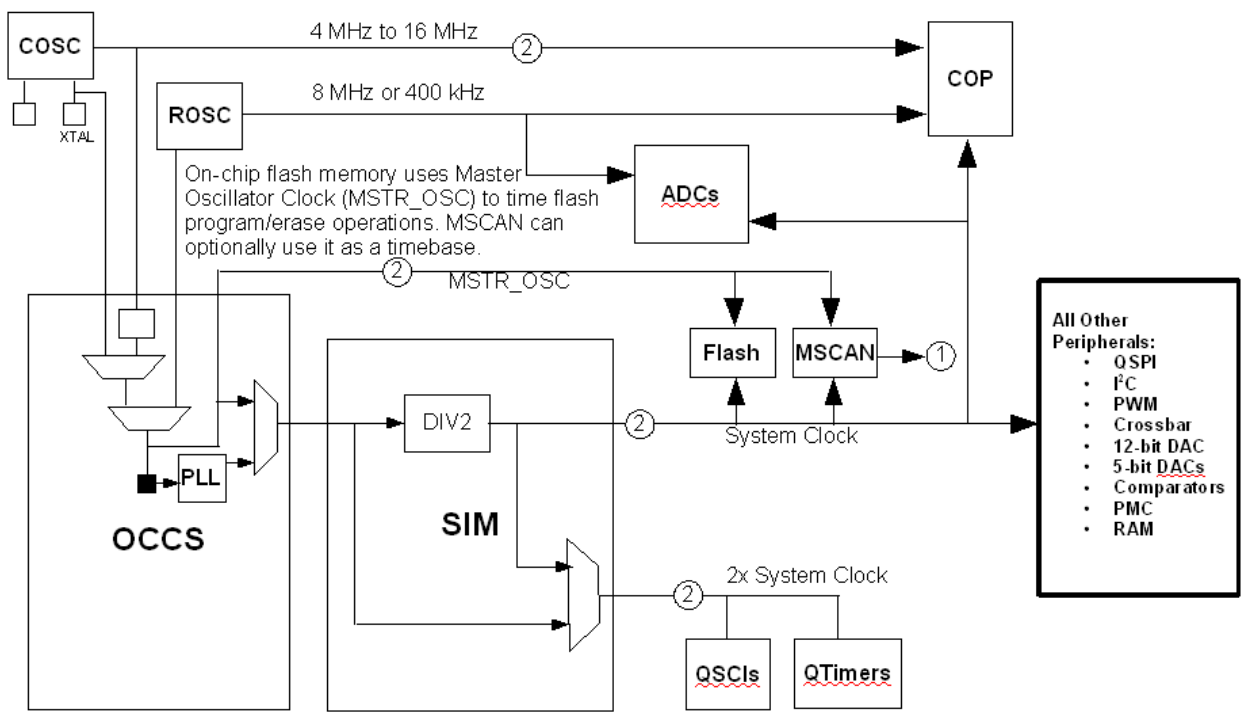


Figure 1-3. System Clock Distribution

When the external clock is chosen as the master clock source, its frequency should not exceed 60 MHz for the following cases:

1. The operation being performed is a flash programming/erase operation.
2. The CLKSRC bit of the MSCAN CANCTL1 register is set to 0.

Chapter 2
Analog-to-Digital Converter (ADC)

2.1 Introduction

2.1.1 Overview

The analog-to-digital converter (ADC) is one dual 12-bit ADC in which each ADC converter has a separate voltage reference and control block.

2.1.2 Features

The analog-to-digital (ADC) converter function consists of two separate analog-to-digital converters, each with eight analog inputs and its own sample and hold circuit. A common digital control module configures and controls the functioning of the converters. ADC features include:

- 12-bit resolution
- Maximum ADC clock frequency of 15 MHz with 100 ns period
- Sampling rate up to 3.33 million samples per second[1]
- Single conversion time of 8.5 ADC clock cycles (8.5 × 100 ns = 850 ns)
- Additional conversion time of 6 ADC clock cycles (6 × 100 ns = 600 ns)
- Eight conversions in 26.5 ADC clock cycles (26.5 × 100 ns = 2.65 µs) using parallel mode
- Can be synchronized to the PWM through the SYNC0/1 input signal if the integration permits the PWM to trigger a timer channel connected to that input

1. In loop mode, the time between each conversion is 6 ADC clock cycles (600 ns). Using simultaneous conversion, two samples can be obtained in 600 ns. Samples per second is calculated according to 600 ns per two samples or 3,333,333 samples per second.

- Sequentially scans and stores up to sixteen measurements
- Scans and stores up to eight measurements each on two ADC converters operating simultaneously and in parallel
- Scans and stores up to eight measurements each on two ADC converters operating asynchronously to each other in parallel
- A scan can pause and await new SYNC input prior to continuing
- Gains the input signal by x1, x2, or x4
- Optional interrupts at end of scan if an out-of-range limit is exceeded or there is a zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single-ended or differential inputs
- PWM outputs with hysteresis for three of the analog inputs

2.1.3 Block Diagram

The following figure illustrates the dual ADC configuration.

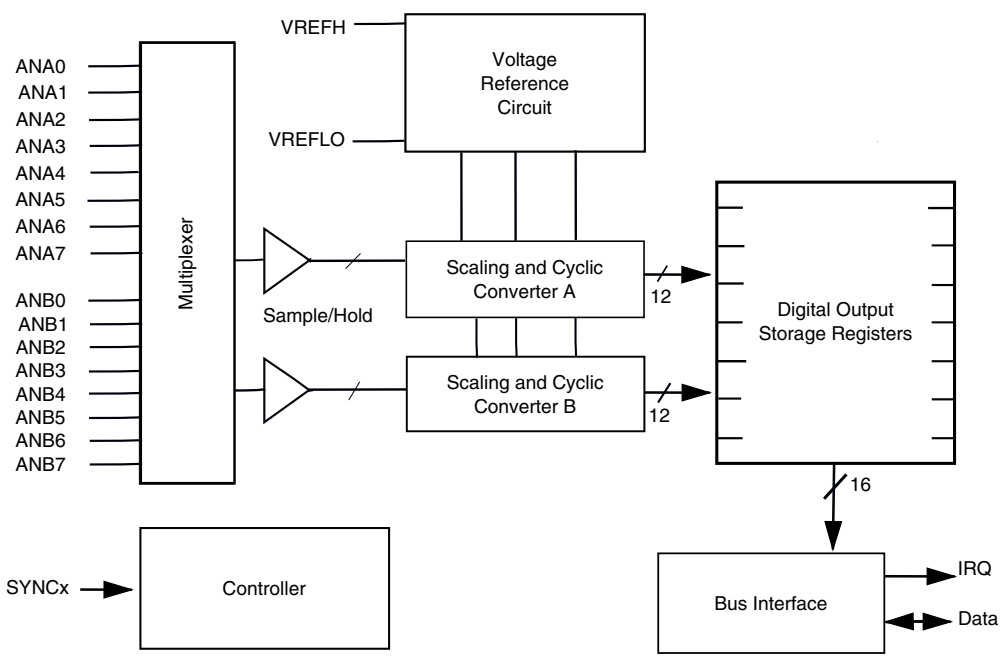


Figure 2-1. Dual ADC Block Diagram

2.2 Signal Descriptions

2.2.1 Overview

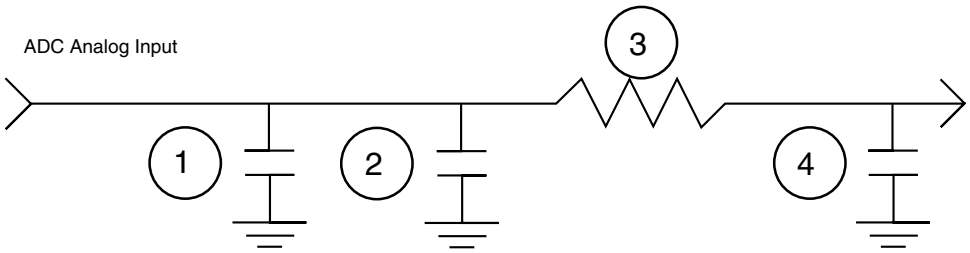
Table 2-1. External Signal Properties

Name	I/O Type	Function	Reset State	Notes
ANA0–ANB7	Input	Analog Input Pins	n/a	—
VREFH	Input	Voltage Reference Pin	n/a	Selectable between VDDA and ANA2
VREFLO	Input	Voltage Reference Pin	n/a	Selectable between VSSA and ANA3
VREFHI	Input	Voltage Reference Pin	n/a	Selectable between VDDA and ANB2
VREFLO	Input	Voltage Reference Pin	n/a	Selectable between VSSA and ANB3
VDDA	Supply	ADC Power	n/a	—
VSSA	Supply	ADC Ground	n/a	—

2.2.2 External Signal Descriptions

2.2.2.1 Analog Input Pins (ANA[0:7] and ANB[0:7])

Each ADC module has sixteen analog input pins that are subdivided into two sets of eight (ANA[0:7] and ANB[0:7]), each with their own S/H unit and converter. This configuration allows simultaneous sampling of two selected channels, one from each subgroup. Sequential scans have access to all sixteen analog inputs. During parallel scans, each ADC converter has access to its eight analog inputs. An equivalent circuit for an analog input is shown below:



1. Parasitic capacitance due to package, pin-to-pin, and pin-to-package base coupling. 1.8pf
2. Parasitic capacitance due to the chip bond pad, ESD protection devices, and signal routing. 2.04pf
3. Equivalent resistance for the ESD isolation resistor and the channel select multiplexer. 500Ω
4. Sampling capacitor at the sample and hold circuit. 1pf

Figure 2-2. Equivalent Analog Input Circuit

2.2.2.2 Voltage Reference Pins (VREFH and VREFLO)

The voltage difference between V_{REFH} and V_{REFLO} provides the reference voltage against which all analog inputs are measured. V_{REFH} is nominally set to V_{DDA}. V_{REFLO} is nominally set to 0V. Any external reference voltage should come from a low noise filtered source. The external reference source should provide up to 1mA of reference current. illustrates the internal workings of the ADC voltage reference circuit. V_{REFH} must be noise filtered. A minimum configuration is shown in the figure.

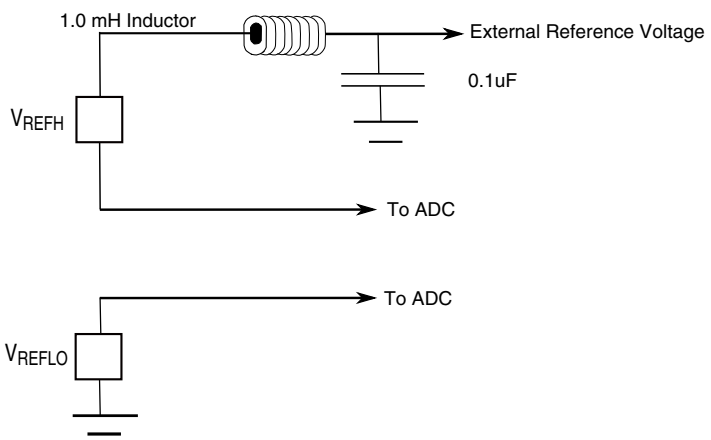


Figure 2-3. ADC Voltage Reference Circuit

When V_{DDA} is used as V_{REFH}, measurements are made with respect to the amplitude of V_{DDA}. Special precautions must be taken to assure that the voltage applied to V_{REFH} is as noise free as possible. Any noise residing on the V_{REFH} voltage is directly transferred to the digital result.

Dedicated power supply pins, V_{DDA} and V_{SSA}, are provided to reduce noise coupling and to improve accuracy. The power to these pins should come from a low noise filtered source. For details, refer to the chapter on the Analog-to-Digital module. Uncoupling capacitors should be connected between V_{DDA} and V_{SSA}.

V_{SS} is shared between both analog and digital circuitry.

There are dedicated analog power pins on Anguilla Black, both V_{DDA} and V_{SSA}.

2.3 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ADC_CTRL1	R	0	STOP0		SYNC0	EOSIE0	ZCIE	LLMTIE	HLMTIE	CHNCFG_L				0	SMODE		
		W			STARTO													
1	ADC_CTRL2	R	0	STOP1	0	SYNC1	EOSIE1	0	CHNCFG_H				SIMULT	DIV				
		W			STARTI													
2	ADC_ZXCTRL	R	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
		W																
3	ADC_CLIST1	R	SAMPLE3				SAMPLE2				SAMPLE1				SAMPLE0			
		W																
4	ADC_CLIST2	R	SAMPLE7				SAMPLE6				SAMPLE5				SAMPLE4			
		W																
5	ADC_CLIST3	R	SAMPLE11				SAMPLE10				SAMPLE9				SAMPLE8			
		W																
6	ADC_CLIST4	R	SAMPLE15				SAMPLE14				SAMPLE13				SAMPLE12			
		W																
7	ADC_SDIS	R	DS															
		W																
8	ADC_STAT	R	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMTI	UNDEFINED							
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
9	ADC_RDY	R	RDY[15:0]															
		W																
A	ADC_LIMSTAT	R	HLS[7:0]								LLS[7:0]							
		W																
B	ADC_ZXSTAT	R	0								ZCS[7:0]							
		W																
C	ADC_RSLT0	R	SEXT	RSLT												0		
		W																
D	ADC_RSLT1	R	SEXT	RSLT												0		
		W																
E	ADC_RSLT2	R	SEXT	RSLT												0		
		W																
F	ADC_RSLT3	R	SEXT	RSLT												0		
		W																
10	ADC_RSLT4	R	SEXT	RSLT												0		
		W																
11	ADC_RSLT5	R	SEXT	RSLT												0		
		W																
12	ADC_RSLT6	R	SEXT	RSLT												0		
		W																
13	ADC_RSLT7	R	SEXT	RSLT												0		
		W																
14	ADC_RSLT8	R	0	RSLT												0		
		W																
15	ADC_RSLT9	R	0	RSLT												0		
		W																
16	ADC_RSLT10	R	0	RSLT												0		
		W																
17	ADC_RSLT11	R	0	RSLT												0		
		W																
18	ADC_RSLT12	R	0	RSLT												0		
		W																

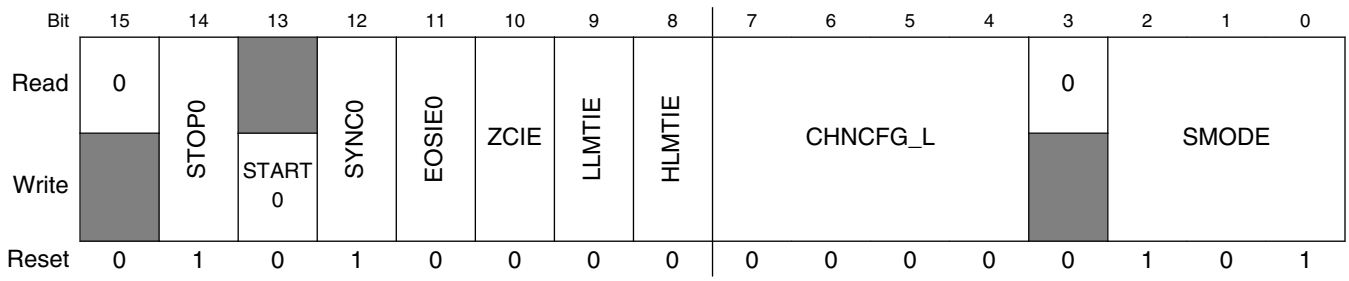
Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
19	ADC_RSLT13	R	0	RSLT													0	
		W																
1A	ADC_RSLT14	R	0	RSLT													0	
		W																
1B	ADC_RSLT15	R	0	RSLT													0	
		W																
1C	ADC_LOLIM0	R	0	LLMT													0	
		W																
1D	ADC_LOLIM1	R	0	LLMT													0	
		W																
1E	ADC_LOLIM2	R	0	LLMT													0	
		W																
1F	ADC_LOLIM3	R	0	LLMT													0	
		W																
20	ADC_LOLIM4	R	0	LLMT													0	
		W																
21	ADC_LOLIM5	R	0	LLMT													0	
		W																
22	ADC_LOLIM6	R	0	LLMT													0	
		W																
23	ADC_LOLIM7	R	0	LLMT													0	
		W																
24	ADC_HILIM0	R	0	HLMT													0	
		W																
25	ADC_HILIM1	R	0	HLMT													0	
		W																
26	ADC_HILIM2	R	0	HLMT													0	
		W																
27	ADC_HILIM3	R	0	HLMT													0	
		W																
28	ADC_HILIM4	R	0	HLMT													0	
		W																
29	ADC_HILIM5	R	0	HLMT													0	
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2A	ADC_HILIM6	R	0	HLMT												0		
		W																
2B	ADC_HILIM7	R	0	HLMT												0		
		W																
2C	ADC_OFFST0	R	0	OFFSET												0	0	0
		W																
2D	ADC_OFFST1	R	0	OFFSET												0	0	0
		W																
2E	ADC_OFFST2	R	0	OFFSET												0	0	0
		W																
2F	ADC_OFFST3	R	0	OFFSET												0	0	0
		W																
30	ADC_OFFST4	R	0	OFFSET												0	0	0
		W																
31	ADC_OFFST5	R	0	OFFSET												0	0	0
		W																
32	ADC_OFFST6	R	0	OFFSET												0	0	0
		W																
33	ADC_OFFST7	R	0	OFFSET												0	0	0
		W																
34	ADC_PWR	R	ASB	0		1	PSTS 1	PSTS 0	PUDELAY						APD	1	PD1	PD0
		W																
35	ADC_CAL	R	SEL_ VREFH_B	SEL_ VREFLO_B	SEL_ VREFH_A	SEL_ VREFLO_A	0								0		SEL_DAC_B	SEL_DAC_A
		W																
36	ADC_GC1	R	GAIN7		GAIN6		GAIN5		GAIN4		GAIN3		GAIN2		GAIN1		GAIN0	
		W																
37	ADC_GC2	R	GAIN15		GAIN14		GAIN13		GAIN12		GAIN11		GAIN10		GAIN9		GAIN8	
		W																
38	ADC_SCTRL	R	SC[15:0]															
		W																
39	ADC_PWR2	R	0			DIV1[4:0]					0				SPEEDB		SPEEDA	
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

2.3.1 ADC Control Register 1 (ADC_CTRL1)

Bits 14, 13, 12, and 11 in CTRL1 control all types of scans except parallel scans in the B converter when CTRL2[SIMULT]=0. Non-simultaneous parallel scan modes allow independent parallel scanning in the A and B converter. Bits 14, 13, 12, and 11 in CTRL2 are used to control B converter scans in non-simultaneous parallel scan modes.

Address: ADC_CTRL1 – F080h base + 0h offset = F080h



ADC_CTRL1 field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 STOP0	Stop When this bit is asserted, the current scan is stopped and no further scans can start. Any further SYNC0 input pulses (see CTRL1[SYNC0] bit) or writes to the CTRL1[START0] bit are ignored until this bit has been cleared. After the ADC is in stop mode, the results registers can be modified by the processor. Any changes to the result registers in stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur when authorized. **This is not the same as DSP STOP mode.** 0 Normal operation 1 Stop mode
13 START0	START0 Conversion A scan is started by writing 1 to this bit. This is a write only bit. Writing 1 to it again while the scan remains in process, is ignored. The ADC must be in a stable power configuration prior to writing the start bit. Refer to the functional description of power modes for further details. 0 No action 1 Start command is issued
12 SYNC0	SYNC0 Enable A conversion may be initiated by asserting a positive edge on the SYNC0 input. Any subsequent SYNC0 input pulses while the scan remains in process are ignored unless the scan is awaiting furhter SYNC inputs due to the SCTRL[SCn] bits. CTRL1[SYNC0] is cleared in ONCE mode, CTRL1[SMODE=000 or 001], when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed.

Table continues on the next page...

ADC_CTRL1 field descriptions (continued)

Field	Description
	The ADC must be in a stable power mode prior to SYNC0 input assertion. Refer to the functional description of power modes for further details. In "once" scan modes, only a first SYNC0 input pulse is honored. CTRL1[SYNC0] is cleared in this mode when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The CTRL1[SYNC0] bit can be set again at any time including while the scan remains in process 0 Scan is initiated by a write to CTRL1[START0] only 1 Use a SYNC0 input pulse or CTRL1[START0] to initiate a scan
11 EOSIE0	End Of Scan Interrupt Enable This bit enables an EOSI0 interrupt to be generated upon completion of the scan. For looping scan modes, the interrupt will trigger after the completion of each iteration of the loop. 0 **Interrupt disabled** 1 **Interrupt enabled**
10 ZCIE	Zero Crossing Interrupt Enable This bit enables the zero crossing interrupt if the current result value has a sign change from the previous result as configured by the ZXCTRL register. 0 **Interrupt disabled** 1 **Interrupt enabled**
9 LLMTIE	Low Limit Interrupt Enable This bit enables the Low Limit exceeded interrupt when the current result value is less than the low limit register value. The raw result value is compared to LOLIM[LLMT] before the offset register value is subtracted. 0 **Interrupt disabled** 1 **Interrupt enabled**
8 HLMTIE	High Limit Interrupt Enable This bit enables the High Limit exceeded interrupt if the current result value is greater than the high limit register value. The raw result value is compared to HILIM[HLMT] before the offset register value is subtracted. 0 **Interrupt disabled** 1 **Interrupt enabled**
7–4 CHNCFG_L	CHCNF (Channel Configure Low) bits The bits configure the analog inputs for either single ended or differential conversions. Differential measurements return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input is at V_{REFLO} and the - input is at V_{REFH}, and scale linearly between based on the voltage difference between the two signals. Single ended measurements return the max value when the input is at V_{REFH}, return 0 when the input is at V_{REFLO}, and scale linearly between based on the amount by which the input exceeds V_{REFLO}. xxx1 **Inputs = ANA0-ANA1** — Configured as differential pair (ANA0 is + and ANA1 is --) xxx0 **Inputs = ANA0-ANA1** — Both configured as single ended inputs xx1x **Inputs = ANA2-ANA3** — Configured as differential pair (ANA2 is + and ANA3 is --)

Table continues on the next page...

ADC_CTRL1 field descriptions (continued)

Field	Description
	xx0x **Inputs = ANA2-ANA3** — Both configured as single ended inputs x1xx **Inputs = ANB0-ANB1** — Configured as differential pair (ANB0 is + and ANB1 is --) x0xx **Inputs = ANB0-ANB1** — Both configured as single ended inputs 1xxx **Inputs = ANB2-ANB3** — Configured as differential pair (ANB2 is + and ANB3 is --) 0xxx **Inputs = ANB2-ANB3** — Both configured as single ended inputs
3 Reserved	This read-only bit is reserved and always has the value zero.
2–0 SMODE	ADC Scan Mode Control This field controls the ADC module's scan mode. All scan modes use 16 sample slots defined by the CLIST1-4 registers. A scan is the process of stepping through a subset of these sample slots, converting the input indicated by a slot, and storing the result. Unused slots may be disabled using the SDIS register. Input pairs ANA0-1, ANA2-3, ANA4-5, ANA6-7, ANB0-1, ANB2-3, ANB4-5, and ANB6-7 may be configured as differential pairs using the CHNCFG fields. When a slot refers to either member of a differential pair, a differential measurement on that pair is made; otherwise, a single ended measurement is taken on that input. The CTRL*[CHNCFG] fields' descriptions detail differential and single ended measurement. The SMODE field determines whether the slots are used to perform one long sequential scan or two shorter parallel scans, each performed by one of the two converters. SMODE controls how these scans are initiated and terminated. It also controls whether the scans are performed once or repetitively. For details, refer to Sequential Versus Parallel Sampling and Scan Sequencing. Parallel scans may be simultaneous (CTRL2[SIMULT] is 1) or non-simultaneous. Simultaneous parallel scans perform the A and B converter scan in lock step using one set of shared controls. Non-simultaneous parallel scans operate the A and B converters independently, with each converter using its own set of controls. Refer to the CTRL2[SIMULT] bit's description for details. Setting any sequential mode overrides the setting of CTRL2[SIMULT]. 000 **Once (single) sequential** — Upon start or an enabled sync signal, samples are taken one at a time starting with CLIST1[SAMPLE0], until the first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after CLIST4[SAMPLE15]. If the scan is initiated by a SYNC signal, only one scan is completed because the CTRL*[SYNC*] bit is cleared automatically by the initial SYNC detection. CTRL*[SYNC*] can be set again at any time during the scan. 001 **Once parallel** — Upon start or an armed and enabled sync signal: In parallel, converter A converts SAMPLEs 0-3, 8-11, and converter B converts SAMPLEs 4-7, 12-15. When CTRL2[SIMULT] is 1 (default), scanning stops when either converter encounters a disabled sample or both converters complete all 8 samples. When CTRL2[SIMULT] is 0, a converter stops scanning when it encounters a disabled sample or completes all 8 samples. If the scan is initiated by a SYNC signal, only one scan is completed because the CTRL*[SYNC*] bit is cleared automatically by the initial SYNC detection. CTRL*[SYNC*] can be set again at any time during the scan. If CTRL2[SIMULT] is 0, the B converter must be rearmed by writing the CTRL2[SYNC1] bit. 010 **Loop sequential** — Upon an initial start or enabled sync pulse, up to 16 samples in the order SAMPLEs 0-15 are taken one at a time until a disabled sample is encountered. The process repeats perpetually until the CTRL1[STOP0] bit is set. While a loop mode is running, any additional start commands or sync pulses are ignored unless the scan is paused using the SCTRL[SC*] bits. If PWR[ASB] or PWR[APD] is the selected power mode control, PWR[PUDELAY] is applied only on the first conversion. 011 **Loop parallel** — Upon an initial start or enabled sync pulse, converter A converts SAMPLEs 0-3, 8-11, and converter B converts SAMPLEs 4-7, 12-15. Each time a converter completes its current scan, it immediately restarts its scan sequence. This process continues until the CTRL*[STOP*] bit is asserted. While a loop is running, any additional start commands or sync pulses are ignored unless the scan is paused using the SCTRL[SC*] bits. When CTRL2[SIMULT] is 1 (default), scanning restarts when either converter encounters a disabled sample. When CTRL2[SIMULT] is

Table continues on the next page...

ADC_CTRL1 field descriptions (continued)

Field	Description
	0, a converter restarts scanning when it encounters a disabled sample. If PWR[ASB] or PWR[APD] is the selected power mode control, PWR[PUDELAY] is applied only on the first conversion. 100 **Triggered sequential** — Upon start or an enabled sync signal, samples are taken one at a time starting with CLIST1[SAMPLE0], until the first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after CLIST4[SAMPLE15]. If external sync is enabled, new scans start for each SYNC pulse that does not overlap with a current scan in progress. 101 **Triggered parallel (default)** — Upon start or an enabled sync signal: In parallel, converter A converts SAMPLEs 0-3, 8-11, and converter B converts SAMPLEs 4-7, 12-15. When CTRL2[SIMULT] is 1 (default), scanning stops when either converter encounters a disabled sample. When CTRL2[SIMULT] is 0, a converter stops scanning when it encounters a disabled sample. If external sync is enabled, new scans start for each SYNC pulse that does not overlap with a current scan in progress. 11x **Reserved**

2.3.2 ADC Control Register 2 (ADC_CTRL2)

Address: ADC_CTRL2 – F080h base + 1h offset = F081h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	STOP1	0	SYNC1	EOSIE1	0	CHNCFG_H				SIMULT	DIV				
Write			START1													
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	0

ADC_CTRL2 field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 STOP1	Stop During parallel scan modes when SIMULT = 0, this bit enables stop control of a B converter parallel scan. When this bit is asserted, the current scan is stopped and no further scans can start. Any further SYNC1 input pulses (see CTRL2[SYNC1] bit) or writes to the CTRL2[START1] bit are ignored until this bit has been cleared. After the ADC is in stop mode, the results registers can be modified by the processor. Any changes to the result registers in stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur when authorized. **This is not the same as DSP STOP mode.** 0 Normal operation 1 Stop mode
13 START1	START1 Conversion

Table continues on the next page...

ADC_CTRL2 field descriptions (continued)

Field	Description
	During parallel scan modes when SIMULT = 0, this bit enables start control of a B converter parallel scan. A scan is started by writing 1 to this bit. This is a write only bit. Writing 1 to it again while the scan remains in process, is ignored. The ADC must be in a stable power configuration prior to writing the start bit. Refer to the functional description of power modes for further details. 0 No action 1 Start command is issued
12 SYNC1	SYNC1 Enable During parallel scan modes when CTRL2[SIMULT]=0, setting this bit to 1 permits a B converter parallel scan to be initiated by asserting the SYNC1 input for at least one ADC clock cycle. CTRL2[SYNC1] is cleared in ONCE mode, CTRL1[SMODE=000 or 001], when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The ADC must be in a stable power mode prior to SYNC1 input assertion. Refer to the functional description of power modes for further details. In "once" scan modes, only a first SYNC1 input pulse is honored. CTRL2[SYNC1] is cleared in this mode when the first SYNC1 input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The CTRL2[SYNC1] bit can be set again at any time including while the scan remains in process. 0 B converter parallel scan is initiated by a write to CTRL2[START1] bit only 1 Use a SYNC1 input pulse or CTRL2[START1] bit to initiate a B converter parallel scan
11 EOSIE1	End Of Scan Interrupt Enable During parallel scan modes when SIMULT = 0, this bit enables interrupt control for a B converter parallel scan. This bit enables an EOSI1 interrupt to be generated upon completion of the scan. For looping scan modes, the interrupt will trigger after the completion of each iteration of the loop. 0 **Interrupt disabled** 1 **Interrupt enabled**
10 Reserved	This read-only bit is reserved and always has the value zero.
9–6 CHNCFG_H	CHCNF (Channel Configure High) bits The bits configure the analog inputs for either single ended or differential conversions. Differential measurements return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input is at V_{REFLO} and the - input is at V_{REFH}, and scale linearly between based on the voltage difference between the two signals. Single ended measurements return the max value when the input is at V_{REFH}, return 0 when the input is at V_{REFLO}, and scale linearly between based on the amount by which the input exceeds V_{REFLO}. xxx1 **Inputs = ANA4-ANA5** — Configured as differential pair (ANA4 is + and ANA5 is --) xxx0 **Inputs = ANA4-ANA5** — Both configured as single ended inputs xx1x **Inputs = ANA6-ANA7** — Configured as differential pair (ANA6 is + and ANA7 is --) xx0x **Inputs = ANA6-ANA7** — Both configured as single ended inputs x1xx **Inputs = ANB4-ANB5** — Configured as differential pair (ANB4 is + and ANB5 is --) x0xx **Inputs = ANB4-ANB5** — Both configured as single ended inputs

Table continues on the next page...

ADC_CTRL2 field descriptions (continued)

Field	Description
	1xxx **Inputs = ANB6-ANB7** — Configured as differential pair (ANB6 is + and ANB7 is --) 0xxx **Inputs = ANB6-ANB7** — Both configured as single ended inputs
5 SIMULT	Simultaneous mode This bit only affects parallel scan modes. By default (CTRL2[SIMULT]=1) parallel scans operate in simultaneous mode. The scans in the A and B converter operate simultaneously and always result in pairs of simultaneous conversions in the A and B converter. CTRL1[STOP0, SYNC0, and START0] control bits and the SYNC0 input are used to start and stop scans in both converters simultaneously. A scan ends in both converters when either converter encounters a disabled sample slot. When the parallel scan completes, the STAT[EOSI0] triggers if CTRL1[EOSIE0] is set. The STAT[CIP0] status bit indicates that a parallel scan is in process. When CTRL2[SIMULT]=0, parallel scans in the A and B converters operate independently. The B converter has its own independent set of the above controls (with a 1 suffix) which control its operation and report its status. Each converter's scan continues until its sample list is exhausted (8 samples) or a disabled sample IN ITS LIST is encountered. For looping parallel scan mode, each converter starts its next iteration as soon as the previous iteration in that converter is complete and continues until the CTRL*[STOP*] bit for that converter is asserted. 0 Parallel scans done independently 1 Parallel scans done simultaneously (default)
4–0 DIV	Clock Divisor Select The divider circuit generates the ADC clock by dividing the system clock by 2*(DIV0[4:0]+1). A DIV0 value must be chosen so the ADC clock does not exceed the maximum frequency. The following table shows ADC clock frequency based on the value of CTRL2[DIV0] for these various OCCS configurations.

Table 2-5. ADC Clock Frequency for Various Conversion Clock Sources

DIV0	Divisor	ROSC Standby 400kHz	ROSC Normal 8MHz	PLL 60 MHz	External CLK
0_0000	2	100K	2.00M	30.0M	CLK/4
0_0001	4	100K	1.00M	15.00M	CLK/8
0_0010	6	100K	667K	10.00M	CLK/12
0_0011	8	100K	500K	7.50M	CLK/16
0_0100	10	100K	400K	6.00M	CLK/20
0_0101	12	100K	333K	5.00M	CLK/24
-	-		-	-	
-	-		-	-	
1_1111	64	100K	62.5K	937K	CLK/128

This clock is used by ADCA during all scans and is used by ADCB during sequential scan modes and during parallel simultaneous scan modes.

2.3.3 ADC Zero Crossing Control Register (ADC_ZXCTRL)

Address: ADC_ZXCTRL – F080h base + 2h offset = F082h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_ZXCTRL field descriptions

Field	Description
15–14 ZCE7	Zero crossing enable 7 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
13–12 ZCE6	Zero crossing enable 6 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
11–10 ZCE5	Zero crossing enable 5 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
9–8 ZCE4	Zero crossing enable 4 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
7–6 ZCE3	Zero crossing enable 3 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
5–4 ZCE2	Zero crossing enable 2 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change**

Table continues on the next page...

ADC_ZXCTRL field descriptions (continued)

Field	Description
	10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
3–2 ZCE1	Zero crossing enable 1 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**
1–0 ZCE0	Zero crossing enable 0 00 **Zero Crossing disabled** 01 **Zero Crossing enabled for positive to negative sign change** 10 **Zero Crossing enabled for negative to positive sign change** 11 **Zero Crossing enabled for any sign change**

2.3.4 ADC Channel List Register 1 (ADC_CLIST1)

Address: ADC_CLIST1 – F080h base + 3h offset = F083h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE3				SAMPLE2				SAMPLE1				SAMPLE0			
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

ADC_CLIST1 field descriptions

Field	Description
15–12 SAMPLE3	Sample Field 3 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-**

Table continues on the next page...

ADC_CLIST1 field descriptions (continued)

Field	Description
	1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE2	Sample Field 2 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
7–4 SAMPLE1	Sample Field 1 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
3–0 SAMPLE0	Sample Field 0 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-**

Table continues on the next page...

ADC_CLIST1 field descriptions (continued)

Field	Description
	0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

2.3.5 ADC Channel List Register 2 (ADC_CLIST2)

Address: ADC_CLIST2 – F080h base + 4h offset = F084h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE7				SAMPLE6				SAMPLE5				SAMPLE4			
Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

ADC_CLIST2 field descriptions

Field	Description
15–12 SAMPLE7	Sample Field 7 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE6	Sample Field 6 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-**

Table continues on the next page...

ADC_CLIST2 field descriptions (continued)

Field	Description
	0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
7–4 SAMPLE5	Sample Field 5 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
3–0 SAMPLE4	Sample Field 4 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-**

Table continues on the next page...

ADC_CLIST2 field descriptions (continued)

Field	Description
	1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

2.3.6 ADC Channel List Register 3 (ADC_CLIST3)

Address: ADC_CLIST3 – F080h base + 5h offset = F085h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE11				SAMPLE10				SAMPLE9				SAMPLE8			
Reset	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0

ADC_CLIST3 field descriptions

Field	Description
15–12 SAMPLE11	Sample Field 11 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE10	Sample Field 10 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-**

Table continues on the next page...

ADC_CLIST3 field descriptions (continued)

Field	Description
	0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
7–4 SAMPLE9	Sample Field 9 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
3–0 SAMPLE8	Sample Field 8 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

2.3.7 ADC Channel List Register 4 (ADC_CLIST4)

Address: ADC_CLIST4 – F080h base + 6h offset = F086h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLE15				SAMPLE14				SAMPLE13				SAMPLE12			
Reset	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0

ADC_CLIST4 field descriptions

Field	Description
15–12 SAMPLE15	Sample Field 15 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
11–8 SAMPLE14	Sample Field 14 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-**

Table continues on the next page...

ADC_CLIST4 field descriptions (continued)

Field	Description
	1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
7–4 SAMPLE13	Sample Field 13 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**
3–0 SAMPLE12	Sample Field 12 0000 **Single Ended: ANA0, Differential: ANA0+, ANA1-** 0001 **Single Ended: ANA1, Differential: ANA0+, ANA1-** 0010 **Single Ended: ANA2, Differential: ANA2+, ANA3-** 0011 **Single Ended: ANA3, Differential: ANA2+, ANA3-** 0100 **Single Ended: ANB0, Differential: ANB0+, ANB1-** 0101 **Single Ended: ANB1, Differential: ANB0+, ANB1-** 0110 **Single Ended: ANB2, Differential: ANB2+, ANB3-** 0111 **Single Ended: ANB3, Differential: ANB2+, ANB3-** 1000 **Single Ended: ANA4, Differential: ANA4+, ANA5-** 1001 **Single Ended: ANA5, Differential: ANA4+, ANA5-** 1010 **Single Ended: ANA6, Differential: ANA6+, ANA7-** 1011 **Single Ended: ANA7, Differential: ANA6+, ANA7-** 1100 **Single Ended: ANB4, Differential: ANB4+, ANB5-** 1101 **Single Ended: ANB5, Differential: ANB4+, ANB5-** 1110 **Single Ended: ANB6, Differential: ANB6+, ANB7-** 1111 **Single Ended: ANB7, Differential: ANB6+, ANB7-**

2.3.8 ADC Sample Disable Register (ADC_SDIS)

Address: ADC_SDIS – F080h base + 7h offset = F087h

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

ADC_SDIS field descriptions

Field	Description
15–0 DS	Disable Sample Bits 0 Enable CLIST*[SAMPLEx]. 1 Disable CLIST*[SAMPLEx] and all subsequent samples. Which samples are actually disabled will depend on the conversion mode, sequential/parallel, and the value of CTRL2[SIMULT].

2.3.9 ADC Status Register (ADC_STAT)

This register provides the current status of the ADC module. STAT[HLMTI and LLMTI] bits are cleared by writing 1s to all asserted bits in the limit status register, LIMSTAT. Likewise, the STAT[ZCI] bit, is cleared by writing 1s to all asserted bits in the zero crossing status register, ZXSTAT. The STAT[EOSIx] bits are cleared by writing a one to them.

Except for STAT[CIP0 and CIP1] this register's bits are sticky. Once set to a one state, they require some specific action to clear them. They are not cleared automatically on the next scan sequence.

Address: ADC_STAT – F080h base + 8h offset = F088h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CIP0	CIP1	0	EOSI 1	EOSI 0	ZCI	LLMT I	HLMT I	UNDEFINED							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STAT field descriptions

Field	Description
15 CIP0	Conversion in Progress

Table continues on the next page...

ADC_STAT field descriptions (continued)

Field	Description
	This bit indicates whether a scan is in progress. This refers to any scan except a B converter scan in non-simultaneous parallel scan modes. 0 Idle state 1 A scan cycle is in progress. The ADC will ignore all sync pulses or start commands
14 CIP1	Conversion in Progress This bit indicates whether a scan is in progress. This refers only to a B converter scan in non-simultaneous parallel scan modes. 0 Idle state 1 A scan cycle is in progress. The ADC will ignore all sync pulses or start commands
13 Reserved	This read-only bit is reserved and always has the value zero.
12 EOSI1	End of Scan Interrupt This bit indicates whether a scan of analog inputs have been completed since the last read of the status register or since a reset. This bit is cleared by writing a one to it. This bit cannot be set by software. In looping scan modes, this interrupt is triggered at the completion of each iteration of the loop. This interrupt is triggered only by the completion of a B converter scan in non-simultaneous parallel scan modes. 0 A scan cycle has not been completed, no end of scan IRQ pending 1 A scan cycle has been completed, end of scan IRQ pending
11 EOSI0	End of Scan Interrupt This bit indicates whether a scan of analog inputs have been completed since the last read of the status register or since a reset. This bit is cleared by writing a one to it. This bit cannot be set by software. STAT[EOSI0] is the preferred bit to poll for scan completion if interrupts are not enabled. In looping scan modes, this interrupt is triggered at the completion of each iteration of the loop. This interrupt is triggered upon the completion of any scan except for the completion of a B converter scan in non-simultaneous parallel scan modes. 0 A scan cycle has not been completed, no end of scan IRQ pending 1 A scan cycle has been completed, end of scan IRQ pending
10 ZCI	Zero Crossing Interrupt If the respective offset register is configured by having a value greater than 0000h, zero crossing checking is enabled. If the offset register is programmed with 7FF8h, the result will always be less than or equal to zero. On the other hand, if 0000h is programmed into the offset register, the result will always be greater than or equal to zero, and no zero crossing can occur because the sign of the result will not change. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active ZXSTAT[ZCS] bits. 0 No zero crossing interrupt request 1 Zero crossing encountered, IRQ pending if CTRL1[ZCIE] is set
9 LLMTI	Low Limit Interrupt

Table continues on the next page...

ADC_STAT field descriptions (continued)

Field	Description
	If the respective low limit register is enabled by having a value other than 0000h, low limit checking is enabled. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active LIMSTAT[LLS] bits. 0 No low limit interrupt request 1 Low limit exceeded, IRQ pending if CTRL1[LLMTIE] is set
8 HLMTI	High Limit Interrupt If the respective high limit register is enabled by having a value other than 7FF8h, high limit checking is enabled. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active LIMSTAT[HLS] bits. 0 No high limit interrupt request 1 High limit exceeded, IRQ pending if CTRL1[HLMTIE] is set
7–0 UNDEFINED	This read-only bitfield is undefined and will always contain random data.

2.3.10 ADC Ready Register (ADC_RDY)

This register provides the current status of the ADC conversions. RDY[RDYx] bits are cleared by reading their corresponding result registers (RSLTx).

Address: ADC_RDY – F080h base + 9h offset = F089h

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

ADC_RDY field descriptions

Field	Description
15–0 RDY[15:0]	Ready Sample These bits indicate samples fifteen through zero are ready to be read. These bits are cleared after a read from the respective results register. The RDY[RDYn] bits are set as the individual channel conversions are completed. Polling the RDY[RDYn] bits can determine if a particular sample is ready to be read. 0 Sample not ready or has been read 1 Sample ready to be read

2.3.11 ADC Limit Status Register (ADC_LIMSTAT)

The limit status register latches in the result of the comparison between the result of the sample and the respective limit register, HILIM0-7and LOLIM0-7. Here is an example: If the result for the channel programmed in CLIST1[SAMPLE0] is greater than the value programmed into the High Limit Register zero, then the LIMSTAT[HLS0] bit is set to one. An interrupt is generated if the CTRL1[HLMTIE] bit is set. A bit may only be cleared by writing a value of one to that specific bit. These bits are sticky. Once set, the bits require a specific modification to clear them. They are not cleared automatically by subsequent conversions.

Address: ADC_LIMSTAT – F080h base + Ah offset = F08Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HLS[7:0]								LLS[7:0]							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_LIMSTAT field descriptions

Field	Description
15–8 HLS[7:0]	High Limit Status Bits
7–0 LLS[7:0]	Low Limit Status Bits

2.3.12 ADC Zero Crossing Status Register (ADC_ZXSTAT)

Address: ADC_ZXSTAT – F080h base + Bh offset = F08Bh

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

ADC_ZXSTAT field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.

Table continues on the next page...

ADC_ZXSTAT field descriptions (continued)

Field	Description
7–0 ZCS[7:0]	Zero Crossing Status The zero crossing condition is determined by examining the ADC value after it has been adjusted by the offset for the result register. Each bit of the register is cleared by writing a one to that register bit. 0 Either: •A sign change did not occur in a comparison between the current channelx result and the previous channelx result, or•Zero crossing control is disabled for channelx in the zero crossing control register, ZXCTRL 1 In a comparison between the current channelx result and the previous channelx result, a sign change condition occurred as defined in the zero crossing control register (ZXCTRL)

2.3.13 ADC Result Registers with sign extension (ADC_RSLT*n*)

The result registers contain the converted results from a scan. The CLIST1[SAMPLE0] result is loaded into RSLT0, CLIST1[SAMPLE1] result in RSLT1, and so on. In a parallel scan mode, the first channel pair designated by CLIST1[SAMPLE0] and CLIST2[SAMPLE4] are stored in RSLT0 and RSLT4, respectively.

Note

When writing to this register, only the RSLT portion of the value written is used. This value is modified and the result of the subtraction is stored. The SEXT bit is only set as a result of this subtraction and is not directly determined by the value written.

Addresses: ADC_RSLT0 – F080h base + Ch offset = F08Ch
ADC_RSLT1 – F080h base + Dh offset = F08Dh
ADC_RSLT2 – F080h base + Eh offset = F08Eh
ADC_RSLT3 – F080h base + Fh offset = F08Fh
ADC_RSLT4 – F080h base + 10h offset = F090h
ADC_RSLT5 – F080h base + 11h offset = F091h
ADC_RSLT6 – F080h base + 12h offset = F092h
ADC_RSLT7 – F080h base + 13h offset = F093h

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

ADC_RSLT*n* field descriptions

Field	Description
15 SEXT	Sign Extend This is the sign-extend bit of the result. RSLT*[SEXT] set to one implies a negative result. RSLT*[SEXT] set to zero implies a positive result. If positive results are required, then the respective offset register must be set to a value of zero.
14–3 RSLT	Digital Result of the Conversion RSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the RSLT can be used directly. As a signed integer, it is an option to right shift with sign extend (ASR) three places and interpret the number, or accept the number as presented, knowing there are missing codes. The lower three bits are always going to be zero. Negative results, RSLT*[SEXT] = 1, are always presented in two's complement format. If it is a requirement of your application that the result registers always be positive, the offset registers must always be set to zero. The interpretation of the numbers programmed into the limit and offset registers, LOLIM, HILIM, and OFFST should match your interpretation of the result register.
2–0 Reserved	This read-only bitfield is reserved and always has the value zero.

2.3.14 ADC Result Registers (ADC_RSLT*n*)

The result registers contain the converted results from a scan. The CLIST3[SAMPLE8] result is loaded into RSLT8, CLIST3[SAMPLE9] result in RSLT9, and so on. In a parallel scan mode, the first channel pair designated by CLIST3[SAMPLE8] and CLIST4[SAMPLE12] are stored in RSLT8 and RSLT12, respectively.

Note

When writing to this register, only the RSLT portion of the value written is used.

Addresses: ADC_RSLT8 – F080h base + 14h offset = F094h
ADC_RSLT9 – F080h base + 15h offset = F095h
ADC_RSLT10 – F080h base + 16h offset = F096h
ADC_RSLT11 – F080h base + 17h offset = F097h
ADC_RSLT12 – F080h base + 18h offset = F098h
ADC_RSLT13 – F080h base + 19h offset = F099h
ADC_RSLT14 – F080h base + 1Ah offset = F09Ah
ADC_RSLT15 – F080h base + 1Bh offset = F09Bh

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

ADC_RSLT*n* field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14–3 RSLT	Digital Result of the Conversion RSLT consists of the raw conversion results; no offset calculation has been applied.
2–0 Reserved	This read-only bitfield is reserved and always has the value zero.

2.3.15 ADC Low Limit Registers (ADC_LOLIM*n*)

Each ADC sample is compared against the values in the limit registers. The comparison is based upon the raw conversion value with no offset correction applied. The limit register used corresponds to the result register the value will be written to. The high limit register is used for the comparison of Result > High Limit. The low limit register is used for the comparison of Result < Low Limit. The limit checking can be disabled by programming the respective limit register with 7FF8h for the high limit and 0000h for the low limit. At reset, limit checking is disabled.

Addresses: ADC_LOLIM0 – F080h base + 1Ch offset = F09Ch
ADC_LOLIM1 – F080h base + 1Dh offset = F09Dh
ADC_LOLIM2 – F080h base + 1Eh offset = F09Eh
ADC_LOLIM3 – F080h base + 1Fh offset = F09Fh
ADC_LOLIM4 – F080h base + 20h offset = F0A0h
ADC_LOLIM5 – F080h base + 21h offset = F0A1h
ADC_LOLIM6 – F080h base + 22h offset = F0A2h
ADC_LOLIM7 – F080h base + 23h offset = F0A3h

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

ADC_LOLIM*n* field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14–3 LLMT	Low Limit Bits
2–0 Reserved	This read-only bitfield is reserved and always has the value zero.

2.3.16 ADC High Limit Registers (ADC_HILIM*n*)

Each ADC sample is compared against the values in the limit registers. The comparison is based upon the raw conversion value with no offset correction applied. The limit register used corresponds to the result register the value will be written to. The high limit register is used for the comparison of Result > High Limit. The low limit register is used for the comparison of Result < Low Limit. The limit checking can be disabled by programming the respective limit register with 7FF8h for the high limit and 0000h for the low limit. At reset, limit checking is disabled.

Addresses: ADC_HILIM0 – F080h base + 24h offset = F0A4h
ADC_HILIM1 – F080h base + 25h offset = F0A5h
ADC_HILIM2 – F080h base + 26h offset = F0A6h
ADC_HILIM3 – F080h base + 27h offset = F0A7h
ADC_HILIM4 – F080h base + 28h offset = F0A8h
ADC_HILIM5 – F080h base + 29h offset = F0A9h
ADC_HILIM6 – F080h base + 2Ah offset = F0AAh
ADC_HILIM7 – F080h base + 2Bh offset = F0ABh

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

ADC_HILIM*n* field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14–3 HLMT	High Limit Bits
2–0 Reserved	This read-only bitfield is reserved and always has the value zero.

2.3.17 ADC Offset Registers (ADC_OFFST*n*)

The value of the offset register is used to correct the ADC result before it is stored in the RSLT registers.

The offset value is subtracted from the ADC result. To obtain unsigned results, program the respective offset register with a value of $0000, thus giving a result range of $0000 to $7FF8.

Addresses: ADC_OFFST0 – F080h base + 2Ch offset = F0ACh
ADC_OFFST1 – F080h base + 2Dh offset = F0ADh
ADC_OFFST2 – F080h base + 2Eh offset = F0AEh
ADC_OFFST3 – F080h base + 2Fh offset = F0AFh
ADC_OFFST4 – F080h base + 30h offset = F0B0h
ADC_OFFST5 – F080h base + 31h offset = F0B1h
ADC_OFFST6 – F080h base + 32h offset = F0B2h
ADC_OFFST7 – F080h base + 33h offset = F0B3h

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

ADC_OFFST*n* field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14–3 OFFSET	ADC Offset Bits
2 Reserved	This read-only bit is reserved and always has the value zero.
1 Reserved	This read-only bit is reserved and always has the value zero.
0 Reserved	This read-only bit is reserved and always has the value zero.

2.3.18 ADC Power Control Register (ADC_PWR)

This register controls the power management features of the ADC module. There are individual manual power down controls for the two ADC converters and the voltage reference generators. There are also five distinct power modes. The following terms are used to describe power modes and their related controls.

Power down state	Each converter and voltage reference generator can individually be put into a power down state. When powered down, the unit consumes no power. Results of scans referencing a powered down converter are undefined. At least one converter must be powered up to use the ADC module.
Manual power down controls	Each converter and voltage reference generator have a manual power control bit capable of putting that component into the power down state. Converters have other mechanisms that can automatically put them into the power down state.

Idle state	The ADC module is idle when neither of the two converters has a scan in process.
Active state	The ADC module is active when at least one of the two converters has a scan in process.
Current Mode	Both converters share a common current mode. Normal current mode is used to power the converters at clock rates above 600kHz. Current mode does not affect the number of ADC clock cycles required to do a conversion or the accuracy of a conversion. The ADC module may change the current mode when idle as part of the power saving strategy.
Startup delay	Auto-powerdown and auto-standby power modes cause a startup delay when the ADC module goes between the idle and active states to allow time to switch clocks or power configurations.

Address: ADC_PWR – F080h base + 34h offset = F0B4h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ASB	0		1	PSTS 1	PSTS 0	PUDELAY						APD	1	PD1	PD0
Write																
Reset	0	0	0	1	1	1	0	0	1	1	0	1	0	1	1	1

ADC_PWR field descriptions

Field	Description
15 ASB	Auto Standby This bit selects auto-standby mode. PWR[ASB] is ignored if PWR[APD] is 1. When the ADC is idle, auto-standby mode selects the standby clock as the ADC clock source and puts the converters into standby current mode. At the start of any scan, the conversion clock is selected as the ADC clock and then a delay of PWR[PUDELAY] ADC clock cycles is imposed for current levels to stabilize. After this delay, the ADC will initiate the scan. When the ADC returns to the idle state, the standby clock is again selected and the converters revert to the standby current state. **NOTE:** This mode is not recommended for conversion clock rates at or below 100 kHz. Instead, set PWR[ASB and APD]=0 and use standby power mode (normal mode with a sufficiently slow conversion clock so that standby current mode automatically engages). This provides the advantages of standby current mode while avoiding the clock switching and the PWR[PUDELAY]. **NOTE:** Set PWR[ASB] prior to clearing PWR[PD1/0]. 0 Auto standby mode disabled 1 Auto standby mode enabled
14–13 Reserved	This read-only bitfield is reserved and always has the value zero.
12 Reserved	This read-only bit is reserved and always has the value one.
11 PSTS1	ADC Converter B Power Status This bit is asserted immediately following a write of "1" to PWR[PD1]. It is de-asserted PWR[PUDELAY] ADC clock cycles after a write of "0" to PWR[PD1] if PWR[APD] is "0". This bit can be read as a status bit

Table continues on the next page...

ADC_PWR field descriptions (continued)

Field	Description
	to determine when the ADC is ready for operation. During auto-powerdown mode, this bit indicates the current powered state of converter B. 0 ADC Converter B is currently powered up 1 ADC Converter B is currently powered down
10 PSTS0	ADC Converter A Power Status This bit is asserted immediately following a write of "1" to PWR[PD0]. It is de-asserted PWR[PUDELAY] ADC clock cycles after a write of "0" to PWR[PD0] if PWR[APD] is "0". This bit can be read as a status bit to determine when the ADC is ready for operation. During auto-powerdown mode, this bit indicates the current powered state of converter A. 0 ADC Converter A is currently powered up 1 ADC Converter A is currently powered down
9–4 PUDELAY	Power Up Delay This 6-bit field determines the number of ADC clocks provided to power up an ADC converter (after setting PWR[PD0 or PD1] to 0) before allowing a scan to start. It also determines the number of ADC clocks of delay provided in auto-powerdown (APD) and auto-standby (ASB) modes between when the ADC goes from the idle to active state and when the scan is allowed to start. The default value is 13 ADC clocks. Accuracy of the initial conversions in a scan will be degraded if PWR[PUDELAY] is set to too small a value. **NOTE:** PWR[PUDELAY] defaults to a value that is typically sufficient for any power mode. The latency of a scan can be reduced by reducing PWR[PUDELAY] to the lowest value for which accuracy is not degraded. Refer to the data sheet for further details.
3 APD	Auto Powerdown Auto-powerdown mode powers down converters when not in use for a scan. PWR[APD] takes precedence over PWR[ASB]. When a scan is started in PWR[APD] mode, a delay of PWR[PUDELAY] ADC clock cycles is imposed during which the needed converter(s), if idle, are powered up. The ADC will then initiate a scan equivalent to that done when PWR[APD] is not active. When the scan is completed, the converter(s) are powered down again. **NOTE:** If PWR[ASB or APD] is asserted while a scan is in progress, that scan is unaffected and the ADC will wait to enter its low power state until after all conversions are complete and both ADC's are idle. PWR[ASB and APD] are not useful in looping modes. The continuous nature of scanning means that the low power state can never be entered. 0 Auto Powerdown Mode is not active 1 Auto Powerdown Mode is active
2 Reserved	This read-only bit is reserved and always has the value one.
1 PD1	Manual Power Down for Converter B This bit forces ADC converter B to power down. Asserting this bit powers down converter B immediately. The results of a scan using converter B will be invalid while PWR[PD1] is asserted. When PWR[PD1] is cleared, converter B is either continuously powered up (PWR[APD] = 0) or automatically powered up when needed (PWR[APD]=1).

Table continues on the next page...

ADC_PWR field descriptions (continued)

Field	Description
	When clearing this bit in any power mode except auto-powerdown (PWR[APD]=1), wait PWR[PUDELAY] ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PWR[PSTS1] bit can be polled to determine when the PWR[PUDELAY] time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples. 0 Power Up ADC converter B 1 Power Down ADC converter B
0 PD0	Manual Power Down for Converter A This bit forces ADC converter A to power down. Asserting this bit powers down converter A immediately. The results of a scan using converter A will be invalid while PWR[PD0] is asserted. When PWR[PD0] is cleared, converter A is either continuously powered up (PWR[APD] = 0) or automatically powered up when needed (PWR[APD]=1). When clearing this bit in any power mode except auto-powerdown (PWR[APD]=1), wait PWR[PUDELAY] ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PWR[PSTS0] bit can be polled to determine when the PWR[PUDELAY] time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples. 0 Power Up ADC converter A 1 Power Down ADC converter A

2.3.19 ADC Calibration Register (ADC_CAL)

The ADC provides for off-chip references that can be used for ADC conversions.

Address: ADC_CAL – F080h base + 35h offset = F0B5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SEL_VREFH_B	SEL_VREFLO_B	SEL_VREFH_A	SEL_VREFLO_A	0								0		SEL_DAC_B	SEL_DAC_A
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CAL field descriptions

Field	Description
15 SEL_VREFH_B	Select V_{REFH} Source This bit selects the source of the V_{REFH} reference for all conversions in converter 1. 0 Internal VDDA 1 ANB2
14 SEL_VREFLO_B	Select V_{REFLO} Source This bit selects the source of the V_{REFLO} reference for all conversions in converter 1.

Table continues on the next page...

ADC_CAL field descriptions (continued)

Field	Description
	0 Internal VSSA 1 ANB3
13 SEL_VREFH_A	Select V_{REFH} Source This bit selects the source of the V_{REFH} reference for all conversions in converter 0. 0 Internal VDDA 1 ANA2
12 SEL_VREFLO_A	Select V_{REFLO} Source This bit selects the source of the V_{REFLO} reference for all conversions in converter 0. 0 Internal VSSA 1 ANA3
11–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3–2 Reserved	This read-only bitfield is reserved and always has the value zero.
1 SEL_DAC_B	Select DAC Alternate Source This bit selects the source of the ADCB7 input as being either the input pin ADCB7 or DAC output. 0 Normal operation (ADCB7). 1 ANB7 input is replaced with DAC output.
0 SEL_DAC_A	Select DAC Alternate Source This bit selects the source of the ADCA7 input as being either the input pin ADCA7 or DAC output. 0 Normal operation (ADCA7). 1 ANA7 input is replaced with DAC output.

2.3.20 Gain Control 1 Register (ADC_GC1)

The gain control registers are used to control amplification of each of the 16 input channels. GAIN0-GAIN7 control the amplification of inputs ANA0-ANA7 while GAIN8-GAIN15 control the amplification of inputs ANB0-ANB7.

Address: ADC_GC1 – F080h base + 36h offset = F0B6h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	GAIN7		GAIN6		GAIN5		GAIN4		GAIN3		GAIN2		GAIN1		GAIN0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_GC1 field descriptions

Field	Description
15–14 GAIN7	Gain Control Bit 7 GAIN 7 controls ANA7 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
13–12 GAIN6	Gain Control Bit 6 GAIN 6 controls ANA6 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
11–10 GAIN5	Gain Control Bit 5 GAIN 5 controls ANA5 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
9–8 GAIN4	Gain Control Bit 4 GAIN 4 controls ANA4 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
7–6 GAIN3	Gain Control Bit 3 GAIN 3 controls ANA3 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
5–4 GAIN2	Gain Control Bit 2 GAIN 2 controls ANA2 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved

Table continues on the next page...

ADC_GC1 field descriptions (continued)

Field	Description
3–2 GAIN1	Gain Control Bit 1 GAIN 1 controls ANA1 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
1–0 GAIN0	Gain Control Bit 0 GAIN 0 controls ANA0 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved

2.3.21 Gain Control 2 Register (ADC_GC2)

The gain control registers are used to control amplification of each of the 16 input channels. GAIN0-GAIN7 control the amplification of inputs ANA0-ANA7 while GAIN8-GAIN15 control the amplification of inputs ANB0-ANB7.

Address: ADC_GC2 – F080h base + 37h offset = F0B7h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	GAIN15		GAIN14		GAIN13		GAIN12		GAIN11		GAIN10		GAIN9		GAIN8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_GC2 field descriptions

Field	Description
15–14 GAIN15	Gain Control Bit 15 GAIN 15 controls ANA15 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
13–12 GAIN14	Gain Control Bit 14 GAIN 14 controls ANA14

Table continues on the next page...

ADC_GC2 field descriptions (continued)

Field	Description
	00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
11–10 GAIN13	Gain Control Bit 13 GAIN 13 controls ANA13 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
9–8 GAIN12	Gain Control Bit 12 GAIN 12 controls ANA12 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
7–6 GAIN11	Gain Control Bit 11 GAIN 11 controls ANA11 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
5–4 GAIN10	Gain Control Bit 10 GAIN 10 controls ANA10 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
3–2 GAIN9	Gain Control Bit 9 GAIN 9 controls ANA9 00 x1 amplification 01 x2 amplification 10 x4 amplification 11 reserved
1–0 GAIN8	Gain Control Bit 8 GAIN 8 controls ANA8 00 x1 amplification

Table continues on the next page...

ADC_GC2 field descriptions (continued)

Field	Description
	01 x2 amplification 10 x4 amplification 11 reserved

2.3.22 ADC Scan Control Register (ADC_SCTRL)

This register is an extension to the CLIST1-4 registers, providing the ability to pause and await a new sync while processing samples programmed in the CLIST*[SAMPLE0–SAMPLE15] fields.

These 16 control bits are used to determine whether a sample in a scan occurs immediately or if the sample waits for an enabled sync input to occur. The sync input must occur after the conversion of the current sample completes. During sequential mode scans, the SCTRL[SC] bits are used in order from SC0 to SC15. During simultaneous parallel scan modes, the bits are used in order from SC0 to SC7. In non-simultaneous parallel scans, ADCA uses the bits in order from SC0 to SC3 followed by SC8 to SC11. ADCB will use bits SC4 to SC7 followed by SC12 to SC15 in non-simultaneous parallel scans.

When setting SCTRL[SC0], don't set CTRL1[START0] or CTRL2[START1]. Just clear CTRL1[STOP0] or CTRL2[STOP1] and the first enabled sync input will start the scan.

Setting SC0 delays sample 0 until a sync pulse occurs. Setting SC1 delays sample 1 until a sync pulse occurs after completing sample 0.

Address: ADC_SCTRL – F080h base + 38h offset = F0B8h

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

ADC_SCTRL field descriptions

Field	Description
15–0 SC[15:0]	Scan Control Bits 0 Perform sample immediately after the completion of the current sample. 1 Delay sample until a new sync input occurs.

2.3.23 ADC Power Control Register (ADC_PWR2)

Address: ADC_PWR2 – F080h base + 39h offset = F0B9h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			DIV1[4:0]					0				SPEEDB		SPEEDA	
Write																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

ADC_PWR2 field descriptions

Field	Description
15–13 Reserved	This read-only bitfield is reserved and always has the value zero.
12–8 DIV1[4:0]	Clock Divisor Select The divider circuit operates in the same manner as the CTRL2[DIV0] field but is used to generate the clock used by ADCB during parallel non-simultaneous scan modes.
7–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3–2 SPEEDB	ADCB Speed Control Bits These bits configure the clock speed at which the ADCB can operate. Faster conversion speeds require greater current consumption. 00 Conversion clock frequency ≤ 5 MHz 01 Conversion clock frequency ≤ 12 MHz 10 Conversion clock frequency ≤ 15 MHz 11 Reserved. Setting SPEEDB to this value will lead to unexpected current consumption by the converters and should be avoided.
1–0 SPEEDA	ADCA Speed Control Bits These bits configure the clock speed at which the ADCA can operate. Faster conversion speeds require greater current consumption. 00 Conversion clock frequency ≤ 5 MHz 01 Conversion clock frequency ≤ 12 MHz 10 Conversion clock frequency ≤ 15 MHz 11 Reserved. Setting SPEEDA to this value will lead to unexpected current consumption by the converters and should be avoided.

2.4 Functional Description

The ADC consists of two eight-channel input select functions, which are two independent sample and hold (S/H) circuits feeding two separate 12-bit ADCs. The two separate converters store their results in an accessible buffer, awaiting further processing.

The conversion process is initiated either by a SYNC signal from one of the on-device timer channels or by writing a 1 to a START bit.

Starting a single conversion actually begins a sequence of conversions, or a scan. The ADC operates in either sequential scan mode or parallel scan mode. In sequential scan mode, scan sequence is determined by defining sixteen sample slots that are processed in order, SAMPLE[0:15]. In parallel scan mode, converter A processes SAMPLE[0:3] and SAMPLE[8:11] in order, and converter B processes SAMPLE[4:7] and SAMPLE[12:15] in order. SAMPLE slots can be disabled using the ADC_SDIS control register to terminate a scan early.

In sequential scan mode, a scan takes up to sixteen single-ended or differential samples, one at a time. See the following figure.

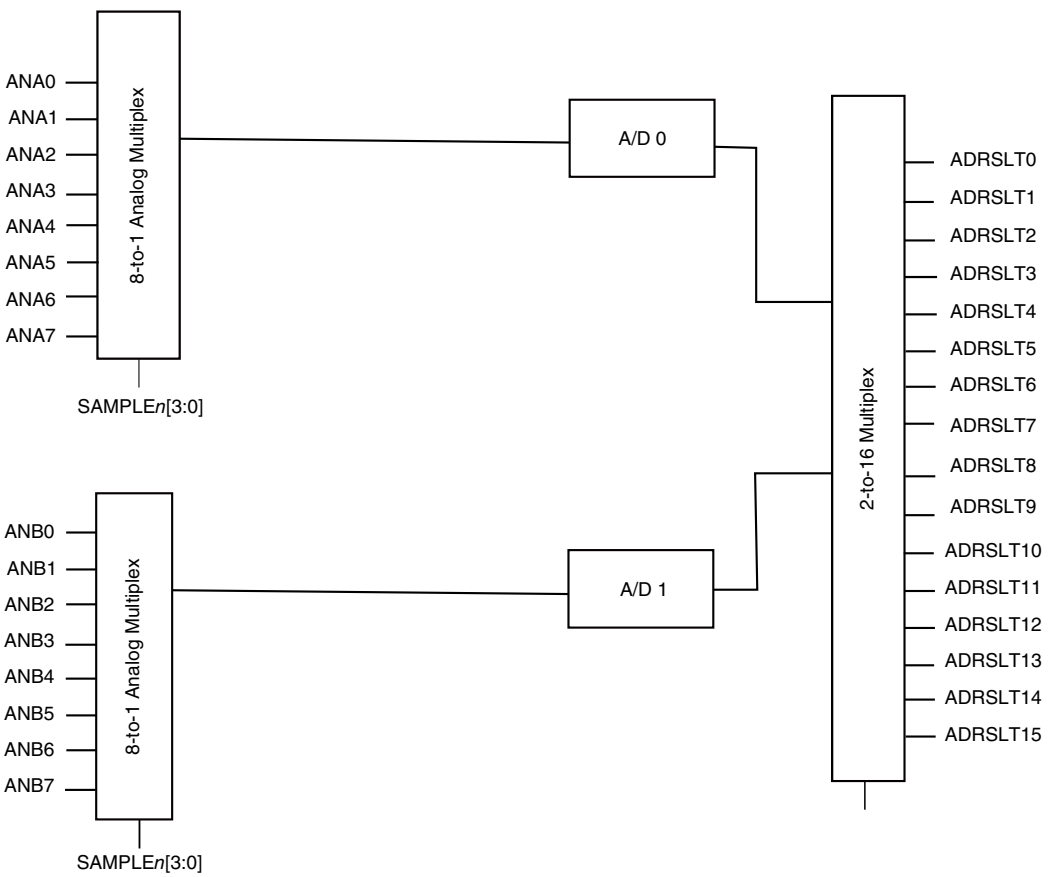


Figure 2-67. ADC Sequential Scan Mode

In parallel scan mode, eight of the sixteen samples are allocated to converter A and eight are allocated to converter B. Two converters operate in parallel, and each can take at most eight samples. Converter A can sample only analog inputs ANA[0:7], and converter B can sample only analog inputs ANB[0:7].

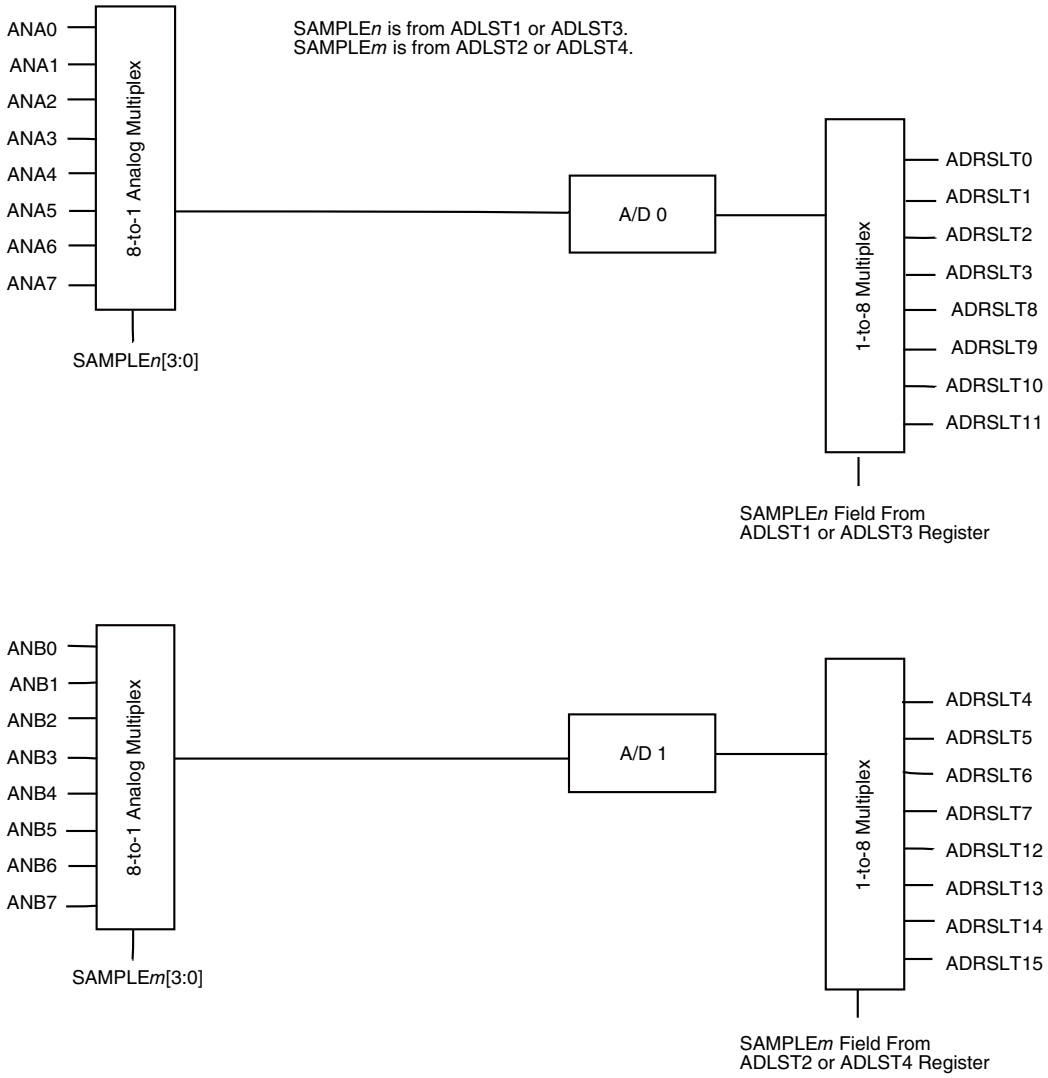


Figure 2-68. ADC Parallel Scan Mode

Each of the following pairs of analog inputs can be configured as a differential pair:

- ANA[0:1], ANA[2:3], ANA[4:5], ANA[6:7]
- ANB[0:1], ANB[2:3], ANB[4:5], ANB[6:7]

When they are so configured, a reference to either member of the differential pair by a sample slot results in a differential measurement using that differential pair.

Parallel scan mode can be simultaneous or non-simultaneous. In simultaneous scan mode, the parallel scans in the two converters occur simultaneously and result in simultaneous pairs of conversions, one by converter A and one by converter B. The two converters share the same start, stop, sync, end-of-scan interrupt enable control, and interrupts. Scanning in both converters terminates when either converter encounters a disabled sample. In non-simultaneous scan mode, the parallel scans in the two converters occur

independently. Each converter has its own start, stop, sync, end-of-scan interrupt enable controls, and interrupts. Scanning in either converter terminates only when that converter encounters a disabled sample.

The ADC can be configured to perform a single scan and halt, perform a scan whenever triggered, or perform the scan sequence repeatedly until manually stopped. The single scan (once mode) differs from the triggered mode only in that SYNC input signals must be re-armed after each use and subsequent SYNC inputs are ignored until the SYNC input is re-armed. This arming can occur any time after the SYNC pulse, including while the scan is still in process.

Optional interrupts can be generated at the end of a scan sequence. Interrupts are available simply to indicate that a scan has ended, that a sample is out of range, or several different zero crossing conditions. Range is determined by the high and low limit registers.

2.4.1 Input Multiplex Function

The following figure shows the input multiplex function. The ChannelSelect and SingleEndedDifferential switches are indirectly controlled by settings within the following registers:

- CLIST1, CLIST2, CLIST3, CLIST4, and SDIS registers
- CTRL1[CHNCFG_L]
- CTRL2[CHNCFG_H]

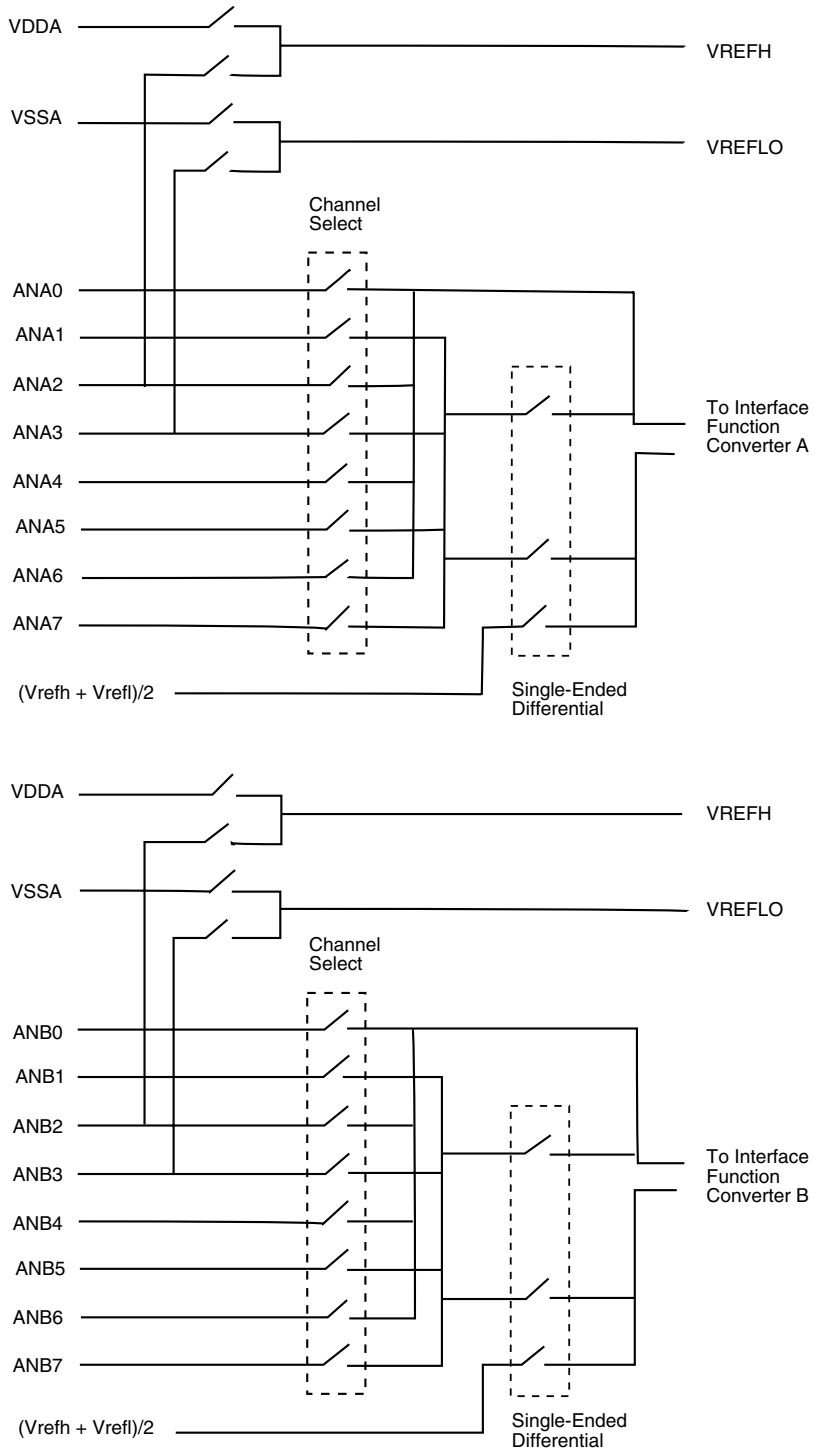


Figure 2-69. Input Select Multiplex

The multiplexing for conversions in different operating modes is as follows:

- Sequential, single-ended mode conversions — During each conversion cycle (sample), any one input of the two four input groups can be directed to its corresponding output.
- Sequential, differential mode conversions — During any conversion cycle (sample), either member of a differential pair may be referenced, resulting in a differential measurement on that pair.
- Parallel, single-ended mode conversions — During any conversion cycle (sample), any of ANA[0:7] can be directed to the converter A output and any of ANB[0:7] can be directed to the converter B output.
- Parallel, differential mode conversions — During any conversion cycle (sample), either member of any possible differential pair—ANA0/1, ANA2/3, ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, and ANB6/7—can be referenced, resulting in a differential measurement of that pair at the converter A output (for ANA pairs) or converter B output (for ANB pairs).

The details of single-ended and differential measurement are described under the CHNCFG field. Internally, all measurements are performed differentially.

Table 2-67. Analog Multiplexing Controls for Each Conversion Mode

Conversion Mode	Channel Select Switches	Single-Ended Differential Switches
General		The two lower switches within the dashed box are controlled so that one switch is always closed and the other open.
Sequential, single-ended	The two 1-of-8 select multiplexes can be set for the appropriate input line.	The lower switch is closed, providing (V_{REFH}+V_{REFLO})/2 to the differential input of the A/D. The upper switch is always closed so that any of the four inputs can get to the A/D input.
Sequential, differential	The channel select switches are turned on in pairs, providing a dual 1-of-4 select function so that either of the two differential channels can be routed to the A/D input.	The upper and lower switches are open and the middle switch is closed, providing the differential channel to the differential input of the A/D.
Parallel, single-ended	The two 1-of-8 select multiplexes can be set for the appropriate input line.	The lower switch is closed, providing V_{REF}/2 to the differential input of the A/D. The upper switch is always closed so that any of the four inputs can get to the A/D input.

Table continues on the next page...

Table 2-67. Analog Multiplexing Controls for Each Conversion Mode (continued)

Conversion Mode	Channel Select Switches	Single-Ended Differential Switches
Parallel, differential	The channel select switches are turned on in pairs, providing a dual 1-of-4 select function so that either of the two differential channels can be routed to the A/D input.	The upper and lower switches are open and the middle switch is closed, providing the differential channel to the differential input of the A/D.

2.4.2 ADC Sample Conversion Operating Modes

The ADC consists of a cyclic, algorithmic architecture using two recursive sub-ranging sections (RSD 1 and RSD 2) as shown in the following figure. Each sub-ranging section resolves a single bit for each conversion clock, resulting in an overall conversion rate of 2 bits per clock cycle. Each sub-ranging section runs at a maximum clock speed of 10 MHz, so a complete 12-bit conversion can be accommodated in 600 ns, not including sample or post-processing time.

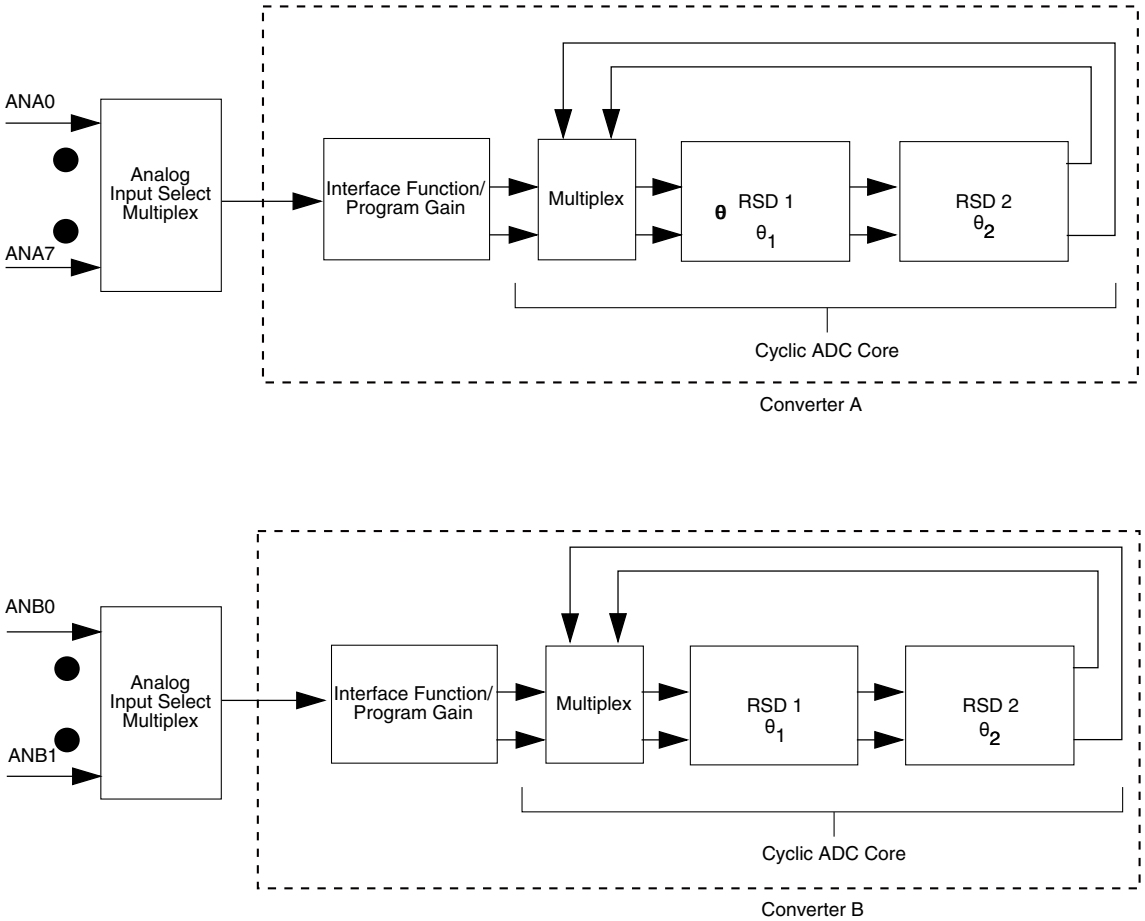


Figure 2-70. Top-Level Diagram of Cyclic ADC

2.4.2.1 Normal Mode Operation

The ADC has two normal operating modes: single-ended mode and differential mode. For a given sample, the mode of operation is determined by the CTRL1[CHNCFG] field:

- Single-ended mode (CHNCFG bit=0). The input multiplex of the ADC selects one of the 8 analog inputs and directs it to the plus terminal of the A/D core. The minus terminal of the A/D core is connected to the V_{REFLO} reference. The ADC measures the voltage of the selected analog input and compares it against the (V_{REFH} - V_{REFLO}) reference voltage range.
- Differential mode (CHNCFG bit=1). The ADC measures the voltage difference between two analog inputs and compares that value against the (V_{REFH}–V_{REFLO}) voltage range. The input is selected as an input pair: ANA0/1, ANA2/3, ANA4/5,

ANA6/7, ANB0/1, ANB2/3, ANB4/5, or ANB6/7. The plus terminal of the A/D core is connected to the even analog input, and the minus terminal is connected to the odd analog input.

A mix and match combination of single-ended and differential configurations may exist. For example:

- ANA[0:1] differential; ANA[2:3] single-ended
- ANA[4:5] differential; ANA[6:7] single-ended
- ANB[0:1] differential; ANB[2:3] single-ended
- ANB[4:5] differential; ANB[6:7] single-ended

2.4.2.1.1 Single-Ended Samples

The ADC module performs a ratio metric conversion. For single-ended measurements, the digital result is proportional to the ratio of the analog input to the reference voltage:

$SingleEndedValue = round(((V_{IN} - V_{REFLO})/(V_{REFH} - V_{REFLO})) \times 4096) \times 8$

V_{IN} is the applied voltage at the input pin.

V_{REFH} and V_{REFLO} are the voltages at the external reference pins on the device (typically V_{REFH}=V_{DDA} and V_{REFLO}=V_{SSA}).

Note: The 12-bit result is rounded to the nearest LSB.
Note: The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted by 3 bits on the 16-bit data bus. As a result, the magnitude of this function, as read from the data bus, is now 32760.

2.4.2.1.2 Differential Samples

For differential measurements, the digital result is proportional to the ratio of the difference in the inputs to the difference in the reference voltages (V_{REFH} and V_{REFLO}).

When differential measurements are converted, the following formula is useful:

$$DifferentialValue = round\left(\left(\frac{VIN1-VIN2}{VREFH-VREFLO} \times 2048\right) + 2048\right) \times 8$$

V_{IN}=Applied voltage at the input pin

V_{REFH} and V_{REFLO}=Voltage at the external reference pins on the device (typically V_{REFH}=V_{DDA} and V_{REFLO}=V_{SSA})

Note: The 12-bit result is rounded to the nearest LSB.
Note: The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted three bits on the 16-bit data bus, so the magnitude of this function, as read from the data bus, is now 32760.

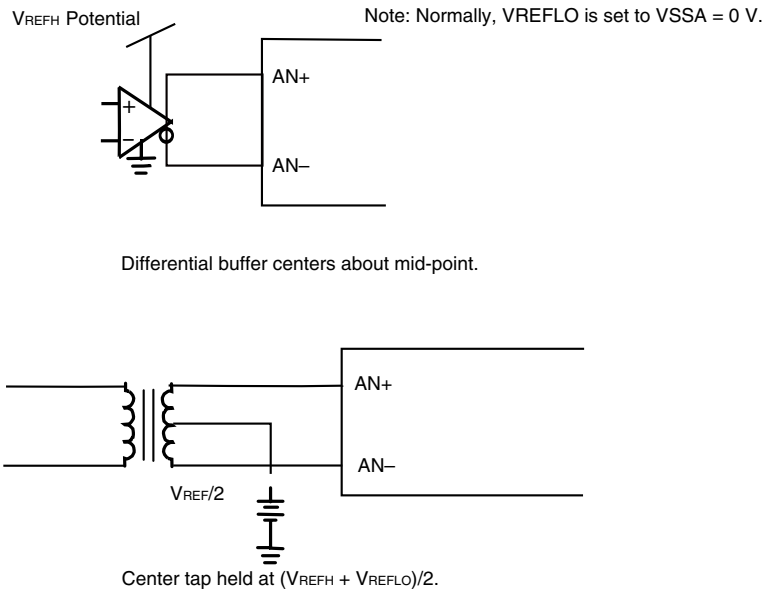


Figure 2-71. Typical Connections for Differential Measurements

2.4.3 ADC Data Processing

The result of an ADC conversion process is normally sent to an adder for offset correction, as the following figure shows. The adder subtracts the ADC_OFFST register value from each sample, and the resultant value is stored in the result register (RSLT). The raw ADC value and the RSLT values are checked for limit violations and zero-crossing, as shown. Appropriate interrupts are asserted, if enabled.

The result value sign is determined from the ADC unsigned result minus the respective offset register value. If the offset register is programmed with a value of zero, the result register value is unsigned and equals the cyclic converter unsigned result. The range of the result (RSLT) is 0000H–7FF8H, assuming that the offset register (OFFST) is cleared to all zeros. This is equal to the raw value of the ADC core.

The processor can write the result registers used for the results of a scan when the STOP bit for that scan is asserted. This write operation is treated as if it comes from the ADC analog core, so the limit checking, zero crossing, and the offset registers function as if in normal mode. For example, if the STOP bit is set to one and the processor writes to RSLT5, the data written to the RSLT5 is multiplexed to the ADC digital logic inputs, processed, and stored into RSLT5 as if the analog core had provided the data. This test data must be justified, as illustrated by the RSLT register definition and does not include the sign bit.

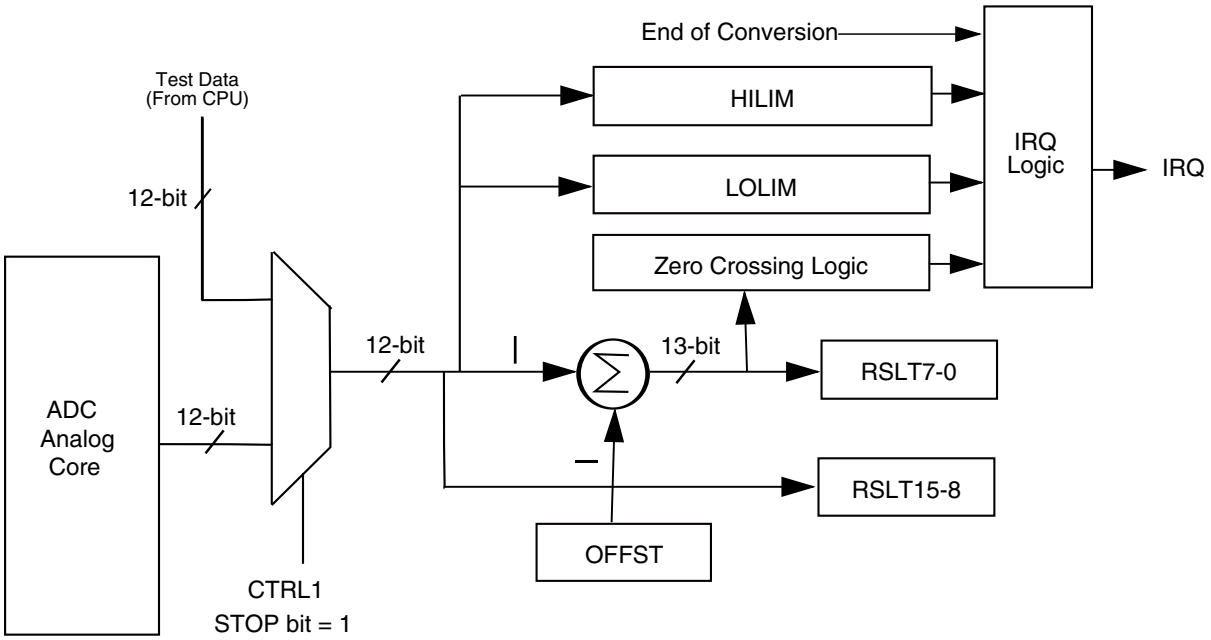


Figure 2-72. Result Register Data Manipulation

2.4.4 Sequential Versus Parallel Sampling

All scan modes use the sixteen sample slots in the CLIST1–4 registers. The slots are used to define which input or differential pair to measure at each step in a scan sequence. The SDIS register defines which sample slots are enabled. Input pairs ANA0/1, ANA2/3, ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, and ANB6/7 can be set to be measured differentially using the CHNCFG field. If a sample refers to an input that is not configured as a member of a differential pair, a single-ended measurement is made. If a sample refers to either member of a differential pair, a differential measurement is made.

Scan are either sequential or parallel. In sequential scans, up to sixteen sample slots are sampled one at a time in order, SAMPLE [0:15]. Each sample refers to any of the sixteen analog inputs ANA0–ANB7, so the same input can be referenced by more than one sample slot. Only SAMPLE[0:7] have the full functionality of offset subtraction and high/low limit compare. SAMPLE[8:15] only store the raw conversion results. Scanning is initiated when the CTRL1 [START0] bit is written with a 1 or when the CTRL1[SYNC0] bit is set and the SYNC0 input goes high. A scan ends when the first disabled sample slot is encountered per the SDIS register. Completion of the scan triggers the STAT[EOSI0] interrupt if the CTRL1[EOSIEN0] interrupt enable is set. The CTRL1[START0] bit and SYNC0 input are ignored while a scan is in process. Scanning stops and cannot be initiated when the CTRL1 [STOP0] bit is set.

Parallel scans differ in that converter A performs up to eight samples (SAMPLE[0:3], SAMPLE[8:11]) in parallel with converter B (SAMPLE[4:7], SAMPLE[12:15]). Constraints are as follows:

- SAMPLEs[0:3] and SAMPLEs[8:11] can reference only the ANA[0:7] inputs.
- SAMPLEs[4:7] and SAMPLEs12:15] can reference only the ANB[0:7] inputs.

Within these constraints, any sample can reference any pin, and more than one sample slot can reference the same sample and the same input. Only SAMPLE[0:7] have the full functionality of offset subtraction and high/low limit compare. By default (when CTRL2[SIMULT]=1), the scans in both converters are initiated when the CTRL1[START0] bit is written with a 1 or when the CTRL1[SYNC0] bit has a value of 1 and the SYNC0 input goes high. The scan in both converters terminates when either converter encounters a disabled sample slot. Completion of a scan triggers the STAT[EOSI0] interrupt if the CTRL1[EOSIEN0] interrupt enable is set. Samples are always taken simultaneously in both the A and B converters. Setting the CTRL1 [STOP0] bit stops and prevents the initiation of scanning in both converters.

Setting CTRL2[SIMULT]=0 (non-simultaneous mode) causes parallel scanning to operate independently in the A and B converter. Each converter has its own set of START, STOP, SYNC, and EOSIEN control bits, SYNC input, EOSI interrupt, and CIP status indicators (suffix 0 for converter A and suffix 1 for converter B). Though still operating in parallel, the scans in the A and B converter start and stop independently according to their own controls and can be simultaneous, phase shifted, or asynchronous depending on when scans are initiated on the respective converters. The A and B converters can be of different length (still up to a maximum of 8) and each converter's scan completes when a disabled sample is encountered in that converters sample list only. CTRL1[STOP0] stops the A converter only and CTRL2[STOP1] stops the B converter only. Looping scan modes iterate independently. Each converter independently restarts its scan after completing its list or encountering a disabled sample slot.

2.4.5 Scan Sequencing

The sequential and parallel scan modes fall into three types based on how they repeat:

- Once scan. A once scan executes a sequential or parallel scan only once each time it is started. It differs from a triggered scan in that sync inputs must be re-armed after each use.
- Triggered scan. Identical to the corresponding once scan modes except that resetting CTRL*[SYNC*] bits is not necessary.
- Looping scan. Automatically restarts a scan, either parallel or sequential, as soon as the previous scan completes. In parallel looping scan modes, the A converter scan restarts as soon as the A converter scan completes and the B converter scan restarts

as soon as the B converter scan completes. All subsequent start and sync pulses are ignored after the scan begins unless the scan is paused by the SCTRL[SC] bits. Scanning can only be terminated by setting the STOP bit.

All scan modes ignore sync pulses while a scan is in process unless the scan is paused by the SCTRL[SC] bits. Once scan modes continue to ignore sync pulses even after the scan completes until the CTRL*[SYNC*] bit is set again. However, a reset can occur any time including during the scan. The SYNC0 input is re-armed by setting the CTRL1[SYNC0] bit, and the SYNC1 input is reset by setting the CTRL2[SYNC1] bit. A reset can be performed any time after a scan starts.

2.4.6 Power Management

The five supported power modes are are discussed in order from highest to lowest power usage at the expense of increased conversion latency and/or startup delay. Changes to the SIM and OCCS that affect the power modes should be made while the PWR[PD0] and PWR[PD1] bits are both asserted. See the Clocks section for details on the various clocks referenced here.

2.4.6.1 Low Power Modes

In the following table, the low-power modes are discussed in order from from highest to lowest power usage.

Mode	Description
Normal power	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), the PWR[APD and ASB] bits are both 0, and the SIM_PCE[ADC] bit is 1. The ADC uses the conversion clock as the ADC clock source in either active or idle. The conversion clock should be configured at or near 15 MHz to minimize conversion latency although PWR2[SPEED*n*] can be used for reduced power consumption when lower conversion frequencies are acceptable . No startup delay (PWR[PUDELAY]) is imposed.
Auto-standby	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), PWR[APD] is 0, PWR[ASB] is 1, and the SIM_PCE[ADC] bit is 1. The OCCS MSTR_OSC signal must operate at 8 MHz. MSTR_OSC is divided by 80 to generate a 100 kHz standby clock either using an 8 MHz external clock source or using the ROSC as the clock source in its normal mode of operation (ROPD=ROSB=0). The ADC uses the conversion clock when active and the 100 kHz standby clock when idle. The standby (low current) state automatically engages when the ADC is idle. The conversion clock should be configured at or near 15 MHz to minimize conversion latency when active although PWR2[SPEEDn] can be used for reduced power consumption when lower conversion frequencies are acceptable. At the start of all scans, there is a startup delay of PWR[PUDELAY] ADC clocks to engage the conversion clock and revert from standby to normal current mode. Auto-standby is a compromise between normal and auto-powerdown modes. This mode offers moderate power savings at the cost of a moderate latency when leaving the idle state to start a new scan.

Table continues on the next page...

Mode	Description
Auto-Powerdown	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), PWR[APD] is 1, and the SIM_PCE [ADC] bit is 1. The conversion clock should be configured at or near 15 MHz to minimize conversion latency when active although PWR2[SPEEDn] can be used for reduced power consumption when lower conversion frequencies are acceptable. The ADC uses the conversion clock when active. For maximum power savings, it gates off the conversion clock and powers down the converters when idle. At the start of all scans, there is a startup delay of PWR[PUDELAY] ADC clocks to stabilize normal current mode from a completely powered off condition. This mode saves more power than auto-standby but requires more startup latency when leaving the idle state to start a scan (higher PWR[PUDELAY] value).
Standby	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), the PWR[ASB] bit is 0, the SIM_PCE[ADC] bit is 1, the PLL is bypassed, and MSTR_OSC is driven from the ROSC in standby mode (PRECS=0, ROSB = 1 and ROPD = 0) at 400 kHz. The ADC clock operates continuously at 100 kHz and standby current mode is enabled continuously without loss of conversion accuracy. Though no startup delay (PWR[PUDELAY]) is imposed, the latency of a scan is affected by the low frequency of the ADC clock. Standby mode is not available when an external clock source is used because there is no way to determine that MSTR_OSC is driven at a low enough frequency to support this mode. To use auto-powerdown mode and standby mode together, set PWR[APD]. This hybrid mode converts at an ADC clock rate of 100 kHz using standby current mode when active, and it gates off the ADC clock and powers down the converters when idle. At the start of all scans, there is a startup delay of PWR[PUDELAY] ADC clock cycles to engage the conversion clock and revert power-up the converters and stabilize them in the standby current mode. This is the slowest and lowest power operational configuration of the ADC.
Powerdown	Both ADC converters and voltage references are powered down (PWR[PD0 and PD1] are both 1) and the SIM_PCE[ADC] bit is 0. In this configuration, the clock trees to the ADC and all of its analog components are shut down and power utilization is eliminated.

2.4.6.2 Startup in Different Power Modes

The ADC voltage reference and converters are powered down (PWR[PDn]=1) on reset. Individual converters and voltage references can be manually powered down when not in use (PWR[PD0]=1 or PWR[PD1]=1). When the ADC reference is powered down, the output reference voltages are set to Low (VSSA) and the ADC data output is driven low.

A delay of PWR[PUDELAY] ADC clock cycles is imposed when PWR[PD0 or PD1] are cleared to power up a regulator and also to transition from an idle state in which neither converter has a scan in process to an active state in which at least one converter has a scan in process. ADC data sheets recommend the use of two PWR[PUDELAY] values: a large value for full powerup and a moderate value for transitioning from standby current levels to full powerup.

To start up in normal mode or standby power mode, perform the following steps:

1. Set PWR[PUDELAY] to the large power -up value.
2. Clear PWR[ASB and APD].
3. Clear the PWR[PD0 and/or PD1] bits to power up the required converters.
4. Poll the status bits until all required converters are powered up.
5. Start scan operations. This will provide a full power-up delay before scans begin.

Normal mode does not use PWR[PUDELAY] at start of scan, so no further delay is imposed.

To start up in auto-standby, use the normal mode startup procedure first. Before starting scan operations, set PWR[PUDELAY] to the moderate standby recovery value, and set PWR[ASB]. Auto- standby mode automatically reduces current levels until active and then imposes the PWR[PUDELAY] to allow current levels to rise from standby to full power levels.

To start up in auto powerdown mode, perform the following steps:

1. Set PWR[PUDELAY] to the large power-up value.
2. Clear PWR[ASB] and set PWR[APD].
3. Clear the PWR[PD0 and/or PD1] bits for the required converters.

Converters remain powered off until scanning goes active. Before a scan starts, there is a large PWR [PUDELAY] to go from the powered down to the fully powered state.

To avoid ambiguity and ensure that the proper delays are applied when powering up or starting scans, both regulators should be powered off (PWR[PD0]=PWR[PD1]=1) when the clock or power controls are configured.

Attempts to start a scan during the PWR[PUDELAY] are ignored until the appropriate PWR[PSTS*n*] bits are cleared.

Any attempt to use a converter when it is powered down or with the voltage references disabled will result in invalid results. It IS possible to read ADC result registers after converter power down for results calculated before power-down. A new scan sequence must be started with a SYNC pulse or a write to the START bit before new valid results are available.

In auto-powerdown mode, when the ADC goes from idle to active, a converter is powered up only if it is required for the scan as determined by the CLIST1-4 and SDIS registers.

2.4.6.3 Stop Mode of Operation

Any conversion sequence can be stopped by setting the relevant STOP bit. Any further sync pulses or writes to the start bit are ignored until the STOP bit is cleared. In stop mode, the results registers can be modified by writes from the processor. Any write to the result register in the ADC-STOP mode is treated as if the analog core supplied the data, so limit checking and zero crossing and associated interrupts can occur if enabled.

2.5 Reset

At reset, all the registers return to the reset state. The source of the single $\overline{\text{RST}}$ signal is the SIM.

2.6 Clocks

The ADC has two external clock inputs to drive two clock domains within the ADC module.

Table 2-69. Clock Summary

Clock input	Source	Characteristics
IP Clock	SIM	Maximum rate is 60 MHz. When the PLL is on and selected, it is PLL output divided by 4. When PLL is not selected, it is MSTR_OSC/2. When the device is in low-power mode, ROSB=1, the rate is 200 kHz.
adc_8_clk	ROSC clock	ROSC provides 8 MHz for auto-standby power saving mode.

The IP_CLK is enabled only when the SIM_PCE[ADC] bit is set. This clock enable bit must be set before the ADC can be used.

The conversion clock is the primary source for the ADC clock and is always selected as the ADC clock when conversions are in process. The clock source controls in the OCCS (PRECS, ROPD, ROSB), and CTRL2[DIV0] and PWR2[DIV1] should be configured so that conversion clock frequency falls between 100 kHz and 15 MHz. Operating the ADC at out-of-spec conversion clock frequencies or reconfiguring the parameters that affect clock rates or power modes while the regulators are powered up (PWR[PD0]=0 or PWR[PD1]=0) negatively affects conversion accuracy.

The conversion clock that the ADC uses for sampling is calculated using the IP bus clock and the clock divisor bits within the ADC Control Register 2. The ADC clock is active 100 percent of the time in looping modes or in normal power mode. It is also active during all ADC powerup sequences for a period of time determined by the PWR[PUDELAY] field. If a conversion is initiated in power savings mode, then the ADC clock continues until the conversion sequence completes.

The following diagram shows the structure of the clocking system.

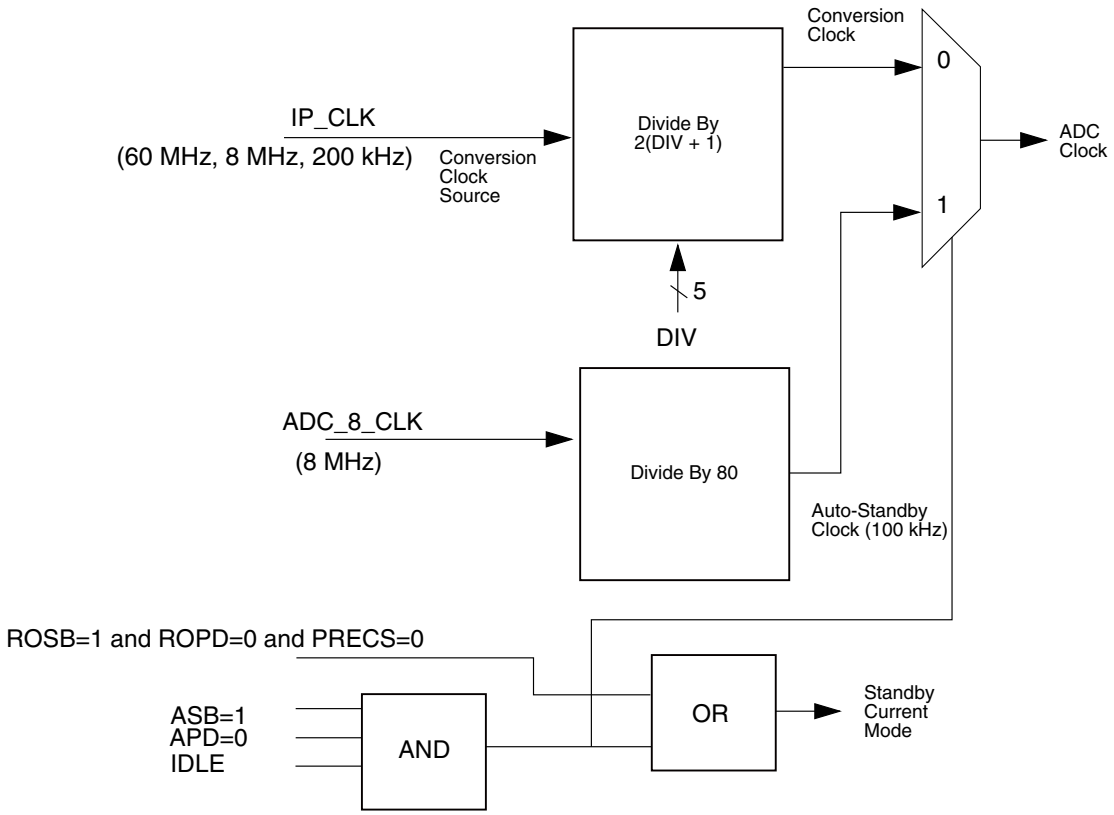


Figure 2-73. ADC Clock Generation

ADC_8_CLK clocks a divider to generate the auto-standby clock, which is selected as the ADC clock only during auto-standby power mode and only when both converters are idle. Auto-standby power mode requires the ADC_8_CLK to be 8 MHz using either the ROSC in normal mode or an 8 MHz external clock. The logic that selects the standby clock as the ADC clock also asserts standby current mode, which is available only at an ADC clock rate of less than 667 kHz. This mode provides substantially power savings yet requires less latency when switching back to the conversion clock at the start of a scan than auto-powerdown mode, which uses only the conversion clock as the ADC clock source but fully powers down the converters when idle.

The standby current mode is also engaged when IP_CLK is driven from the ROSC (PRECS=0) and the ROSC is in standby mode (ROSB=1 and ROPD=0). This ensures a 100 kHz ADC conversion clock rate while a conversion is in process. This configuration, referred to as standby power mode, provides accurate conversion without startup latency at the reduced power levels of standby current mode but requires a 100 kHz conversion clock. Standby power mode is not available with external clocking.

The ADC_CLK is an output of the gasket used to operate the two converters during scan operations. It is derived by multiplexing the conversion clock (divided version of the conversion clock source) and the standby clock (divided version of MSTR_OSC). This clock can be selected in the SIM for external output for debug and failure analysis.

2.7 Interrupts

The following table summarizes the ADC interrupts.

Table 2-70. Interrupt Summary

Interrupt	Source	Description
ADC_ERR_INT_B	STAT[[ZCI], STAT[LLMTI], STAT[HLMTI]	Zero Crossing, low Limit, and high limit interrupt
ADC_CC0_INT_B	STAT[EOSI0]	Conversion Complete Interrupt for any scan type except converter B scan in non-simultaneous parallel scan mode (see EOSI0)
ADC_CC1_INT_B	STAT[EOSI1]	Conversion Complete Interrupt for converter B scan in non-simultaneous parallel scan mode (see EOSI1)

ADC interrupts fall into two categories:

- Threshold interrupts, which are caused by three different events. All of these interrupts are optional and enabled through control register CTRL1:
  - Zero crossing — occurs if the current result value has a sign change from the previous result as configured by the ZXCTRL register.
  - Low limit exceeded error — occurs when the current result value is less than the low limit register value. The raw result value is compared to LOLIM[LLMT] before the offset register value is subtracted.
  - High limit exceeded error — is asserted if the current result value is greater than the high limit register value. The raw result value is compared to HILIM[HLMT] before the offset register value is subtracted.
- Conversion complete interrupts, which are generated upon completion of any scan and convert sequence when CTRL1[EOSIE0]=1. Additional bits may need to be set in the Interrupt Control Module to enable the CPU to receive the interrupt signal.

2.8 Timing Specifications

The following figure shows a timing diagram for the ADC module. The ADC is assumed to be in Once or Triggered mode, so the ADC clock is shown in the OFF state prior to the SYNC pulse or START bit write. The ADC clock restarts (switching high) within 1 to 2 IP bus clocks of that event. ADC_CLK is derived from the ROSC or PLL output. The frequency relationship is programmable. Conversions are pipelined. The second start command is ignored because the ADC is busy with the previous start request. The third

start command is recognized and is synchronized to the positive edge of the ADC clock when the conversion process is restarted. The ADC has two possible interrupts that are latched in the ADSTAT register:

- Conversion complete interrupt (End of Scan interrupt, EOSIx)
- Zero crossing or limit error interrupt (ZCI, LLMTI, and HLMTI)

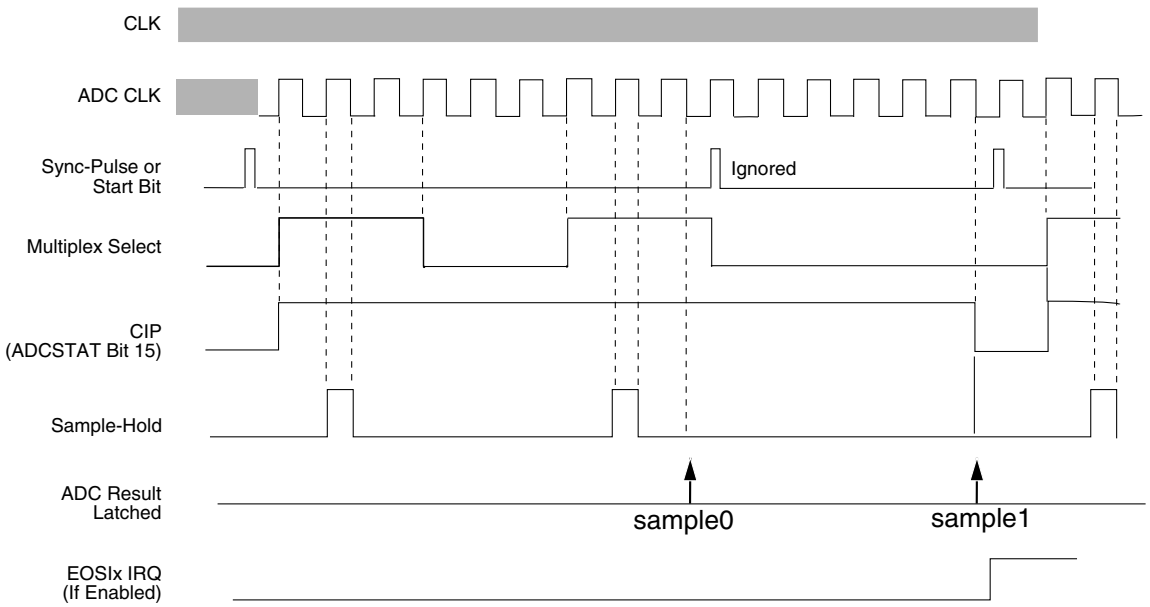


Figure 2-74. ADC Timing

As the figure shows, a conversion is initiated by a sync pulse originating from the timer module or by a write to a start bit. In APD or ASB mode, a delay of PWR [PUDELAY] ADC clock cycles is imposed. The conversion is initiated in the next clock cycle. The ADC clock period is determined by the CTRL2[DIV0] or PWR2[DIV1] value and the OCCS clock configuration.

The first conversion takes 8.5 ADC clocks to be valid. Then, each additional sample takes only six ADC clocks. The start conversion command is latched and the real conversion process is synchronized to the positive edge of the ADC clock.

Because the conversion is a pipeline process, after the last sample is in the S/H, the ADC cannot be restarted until the pipeline is emptied. However, the conversion cycle can be aborted by issuing a STOP command.

The figure shown here illustrates the case in which PWR[APD and ASB] are not in use. When the PWR[APD or ASB] bit is set, the sync pulse or start powers up the ADC, waits for a number of ADC clocks (determined by the PWR[PUDELAY] bits) for the ADC circuitry to stabilize, and only then begins the conversion sequence.

Chapter 3
High Speed Comparator (HSCMP)

3.1 Introduction

The High Speed Comparator module (HSCMP) 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).

3.2 Features

The HSCMP has the following features:

- Operates over the entire supply range
- Inputs may range from rail to rail.
- Less than 40 mV of input offset.
- Less than 15 mV of hysteresis.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output.
- Selectable inversion of comparator output.
- Comparator output may be:
    - Sampled
    - Windowed (ideal for certain PWM zero-crossing-detection applications)
    - Digitally filtered:
        - Filter can be bypassed.
        - May be clocked through the external SAMPLE signal or a scaled peripheral clock.

- External hysteresis can be used while the output filter is used for internal functions.
- The plus and minus inputs of the comparator are both driven from 4-to-1 multiplexers, providing additional flexibility in assigning I/O as comparator inputs during PCB design.
- Two software-selectable performance levels:
  - High power, with shorter propagation delay. This mode can be used only when the VDDA rail is above the low voltage interrupt trip point.
  - Low power, with longer propagation delay.

3.3 Block Diagram

The block diagram for the High Speed Comparator module is shown in the following block diagram.

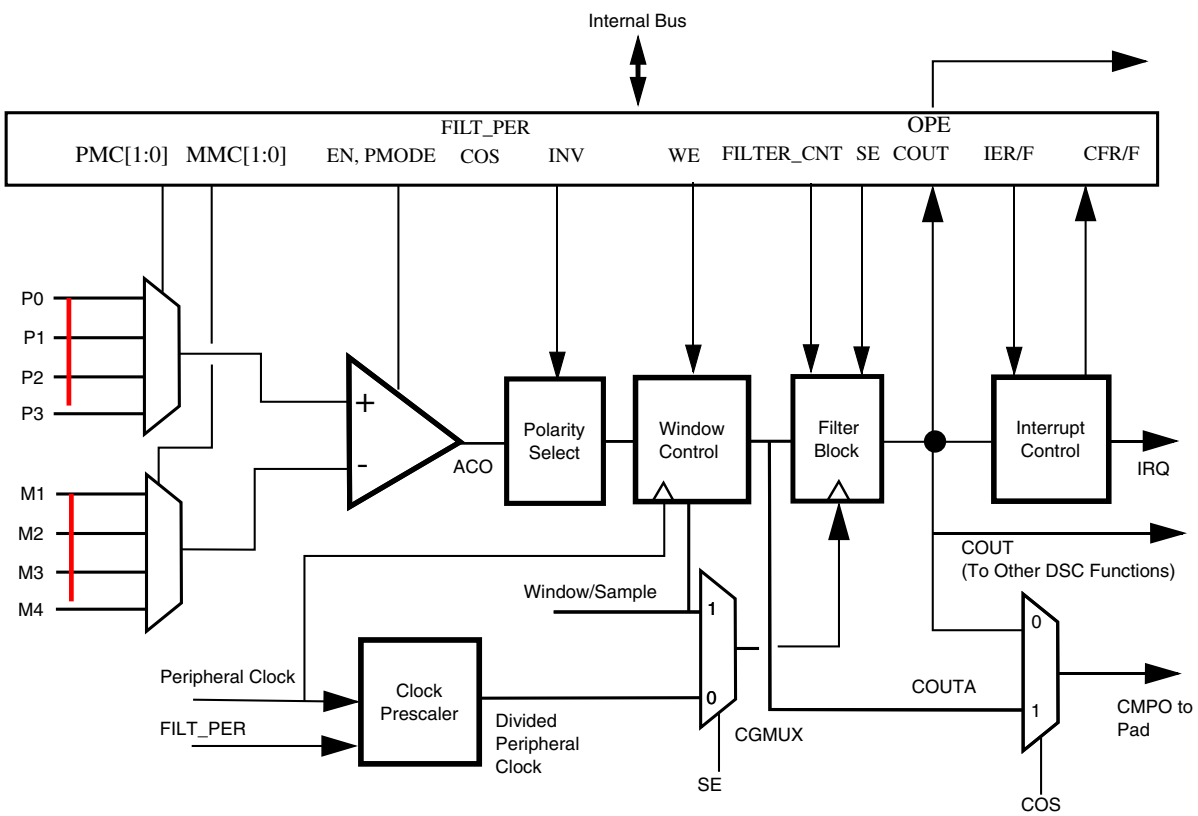


Figure 3-1. High-Speed Comparator Module Block Diagram

In the block diagram:

- The Window Control block is completely bypassed when WE = 0.
- If WE = 1, the comparator output is sampled on every peripheral clock when WINDOW = 1 to generate COUTA. Sampling does *not* occur when WINDOW = 0.
- The Filter block is bypassed when not in use.

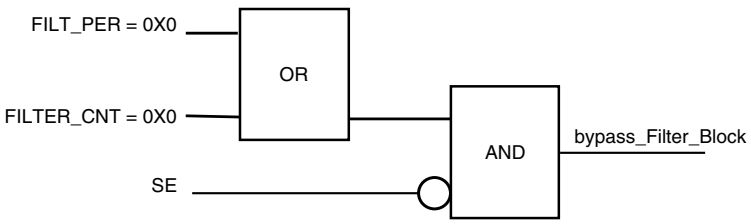


Figure 3-2. Filter Block Bypass Logic

- The Filter block acts as a simple sampler if bypass_Filter_Block && FILTER_CNT = 0x1.
- The Filter block filters based on multiple samples when bypass_Filter_Block && FILTER_CNT > 0x1:
  - If SE = 1, the external SAMPLE input is used as sampling clock.
  - IF SE = 0, the divided peripheral clock is used as sampling clock.
- If enabled, the Filter block incurs up to 1 IP Bus additional latency penalty on COUT because COUT (which is crossing clock domain boundaries) must be resynchronized to the peripheral clock.
- WE and SE are mutually exclusive.

3.4 Pin Descriptions

3.4.1 External Pins

Each HSCMP has up to eight external analog input pins and one external output pin. Each input pin can accept an input voltage that varies across the full operating voltage range of the DSC.

Consult the data sheet to determine what functions are shared with analog inputs. As shown in the block diagram, the Pn pins are connected to the comparator non-inverting input. Mn pins are connected to the inverting input of the comparator.

The user can select either filtered or unfiltered comparator outputs for use on an external I/O pin.

Operation of the CR1[OPE] register simplified examples is shown in the following figure.

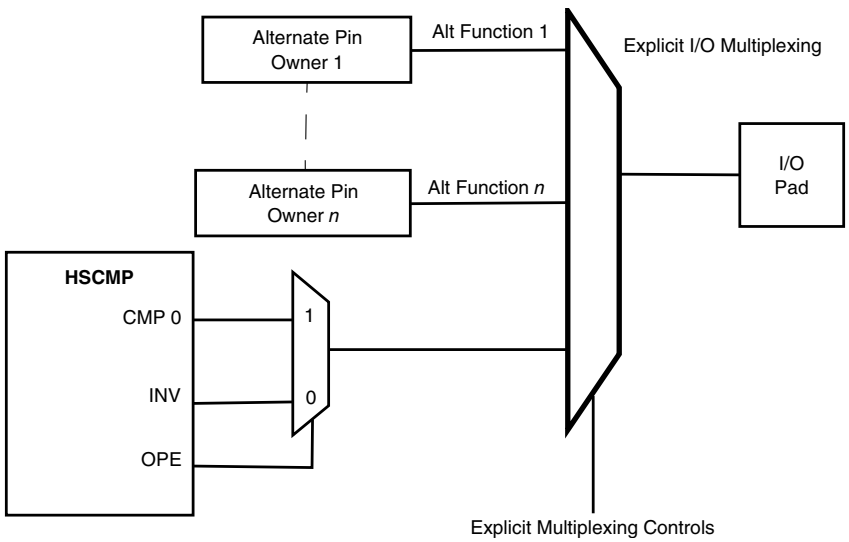


Figure 3-3. OPE Operation

3.5 Memory Map and Register Definitions

Address offsets are in terms of 16-bit words. The 8-bit register descriptions shown are zero-extended to 16 bits.

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CMPA_CR0	R	0								FILTER_CNT				PMC		MMC	
		W																
1	CMPA_CR1	R	0								SE	WE	0	PMODE	INV	COS	OPE	EN
		W																
2	CMPA_FPR	R	0								FILT_PER							
		W																
3	CMPA_SCR	R	0								HYST_SEL		SMELB	IER	IEF	CFR	CFF	COUT
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CMPB_CR0	R	0								FILTER_CNT				PMC		MMC	
		W																
1	CMPB_CR1	R	0								SE	WE	0	PMODE	INV	COS	OPE	EN
		W																
2	CMPB_FPR	R	0								FILT_PER							
		W																
3	CMPB_SCR	R	0								HYST_SEL		SMELB	IER	IEF	CFR	CFF	COUT
		W																
0	CMPC_CR0	R	0								FILTER_CNT				PMC		MMC	
		W																
1	CMPC_CR1	R	0								SE	WE	0	PMODE	INV	COS	OPE	EN
		W																
2	CMPC_FPR	R	0								FILT_PER							
		W																
3	CMPC_SCR	R	0								HYST_SEL		SMELB	IER	IEF	CFR	CFF	COUT
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

3.5.1 Control Register 0 (CMPx_CR0)

Addresses: CMPA_CR0 – F1B0h base + 0h offset = F1B0h
CMPB_CR0 – F1C0h base + 0h offset = F1C0h
CMPC_CR0 – F1D0h base + 0h offset = F1D0h

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

CMPx_CR0 field descriptions

Field	Description
15–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6–4 FILTER_CNT	Filter Sample Count

Table continues on the next page...

CMPx_CR0 field descriptions (continued)

Field	Description
	These bits represent the number of consecutive samples that must agree prior to the comparator output filter accepting a new output state. 000 Filter is disabled. If SE = 1, Then COUT will be a logic zero, and is not recommended). If SE = 0, COUT = COUTA. 001 1 consecutive samples must agree (comparator output is simply sampled) 010 2 consecutive samples must agree 011 3 consecutive samples must agree 100 4 consecutive samples must agree 101 5 consecutive samples must agree 110 6 consecutive samples must agree 111 7 consecutive samples must agree
3–2 PMC	Positive Input Mux Control Determines which input is selected for the positive input of the comparator. 00 P0 01 P1 10 P2 11 P3
1–0 MMC	Minus Input Mux Control Determines which input is selected for the minus input of the comparator. 00 M0 01 M1 10 M2 11 M3

3.5.2 Control Register 1 (CMPx_CR1)

Addresses: CMPA_CR1 – F1B0h base + 1h offset = F1B1h
CMPB_CR1 – F1C0h base + 1h offset = F1C1h
CMPC_CR1 – F1D0h base + 1h offset = F1D1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SE	WE	0	PMODE	INV	COS	OPE	EN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMPx_CR1 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.

Table continues on the next page...

CMPx_CR1 field descriptions (continued)

Field	Description
7 SE	Sample Enable At most, one of SE and WE can be set at a time. If a write to this register attempts to set both, then SE will be set and WE will be cleared. However, writing ones to both bit locations should be avoided, as the “11” case is reserved and may change in future implementations. 0 Sampling Mode NOT Selected 1 Sampling Mode Selected
6 WE	Windowing Enable At most, one of SE and WE can be set at a time. If a write to this register attempts to set both, then SE will be set and WE will be cleared. However, writing ones to both bit locations should be avoided, as the “11” case is reserved and may change in future implementations. 0 Windowing Mode NOT Selected 1 Windowing Mode Selected
5 Reserved	This read-only bit is reserved and always has the value zero.
4 PMODE	Power Mode Select 0 Power Savings Mode Selected 1 High Speed Comparison Mode Selected
3 INV	Comparator INVERT This bit allows you to select the polarity of the comparator function. It is also driven to the COUT output (on both the device pin and as SCR[COUT]) when OPE=0. 0 Do NOT invert the comparator output 1 Invert the output of the analog comparator
2 COS	Comparator Output Select 0 Set CMPO to equal COUT (filtered comparator output) 1 Set CMPO to equal COUTA (unfiltered comparator output)
1 OPE	Comparator Output Pin Enable OPE is used to enable the comparator output to be placed onto the external pin, CMPO. Operation of the OPE bit varies, depending on how it is encapsulated within the SoC. There are two cases. For devices which utilize explicit mux control to I/O pin functions (DSC, and non-Flexis ColdFire devices), we have: 0 The comparator output (CMPO) is not available on associated CMPO output pin. Instead, the INV bit is driven IF the comparator owns the pin in question (usually a result of properly setting pin mux controls at the SoC level). If the comparator does not own the pin, this bit has no effect. 1 The comparator output (CMPO) is driven out on associated CMPO output pin if the comparator owns the pin. Again, if the comparator does not own the pin, this bit has no effect.
0 EN	Comparator Module Enable The EN bit enables the Analog Comparator Module. When the module is not enabled, it remains in the off state, and consumes no power.

Table continues on the next page...

CMPx_CR1 field descriptions (continued)

Field	Description
	0 Analog Comparator disabled 1 Analog Comparator enabled

3.5.3 Filter Period Register (CMPx_FPR)

Addresses: CMPA_FPR – F1B0h base + 2h offset = F1B2h
CMPB_FPR – F1C0h base + 2h offset = F1C2h
CMPC_FPR – F1D0h base + 2h offset = F1D2h

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

CMPx_FPR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 FILT_PER	Filter Sample Period When CR1[SE] is equal to zero, this field specifies the sampling period, in peripheral clock cycles, of the comparator output filter. Setting FILT_PER to 0x0 disables the filter. This field has no affect when CR1[SE] is equal to one. In that case, the external SAMPLE signal is used to determine the sampling period.

3.5.4 Status and Control Register (CMPx_SCR)

NOTE

Interrupts must be disabled during power up, recovery from partial-power-down and module enable sequences. The comparator output is not guaranteed to be stable until the module has had time to reach its stable operating point. See the tONEN, tONPOR, and tONPPD electrical specifications in the data sheet.

Addresses: CMPA_SCR – F1B0h base + 3h offset = F1B3h
CMPB_SCR – F1C0h base + 3h offset = F1C3h
CMPC_SCR – F1D0h base + 3h offset = F1D3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								HYST_SEL		SMELB	IER	IEF	CFR	CFF	COUT
Write																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

CMPx_SCR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–6 HYST_SEL	Hysteresis Select These control bits are used to select the amount of hysteresis on the analog comparator. Values range from 0 (least hysteresis) to 3(most hysteresis) with the reset value of 1being equivalent to the hysteresis found in past comparators in this family.
5 SMELB	Stop Mode Edge / Level Interrupt Control This bit controls whether the CFR and CFF bits are edge sensitive or level sensitive in STOP mode. 0 CFR/CFF are Level Sensitive in STOP mode. CFR will be asserted when COUT is high. CFF will be asserted when COUT is LOW 1 CFR/CFF are Edge Sensitive in STOP mode. An active low-to-high transition must be seen on COUT to assert CFR, and an active high-to-low transition must be seen on COUT to assert CFF.
4 IER	Comparator Interrupt Enable Rising The IER bit enables the CFR interrupt from the ACM. When this bit is set, an interrupt will be asserted when the CFR bit is set. 0 Interrupt disabled 1 Interrupt enabled
3 IEF	Comparator Interrupt Enable Falling The IEF bit enables the CFF interrupt from the ACM. When this bit is set, an interrupt will be asserted when the CFF bit is set. 0 Interrupt disabled 1 Interrupt enabled
2 CFR	Analog Comparator Flag Rising During normal operation, the CFR bit is set when a rising edge on COUT has been detected. The CFR bit is cleared by writing a logic one to the bit. During STOP mode, CFR can be programmed as either edge or level sensitive via the SMELB bit. **NOTE:** Edge detection during STOP is only supported on platforms which allow peripherals to be clocked during STOP modes. If the CFF and CFR flags are to be active during STOP, then SMELB must be set to “0” for cases where the it is not receiving a clock during STOP. 0 Rising edge on COMPO has not been detected. 1 Rising edge on COUT has occurred.

Table continues on the next page...

CMPx_SCR field descriptions (continued)

Field	Description
1 CFF	Analog Comparator Flag Falling During normal operation, the CFF bit is set when a falling edge on COUT has been detected. The CFF bit is cleared by writing a logic one to the bit. During STOP mode, CFF can be programmed as either edge or level sensitive via the SMELB bit. **NOTE:** Edge detection during STOP is only supported on platforms which allow peripherals to be clocked during STOP modes. If the CFF and CFR flags are to be active during STOP, then SMELB must be set to “0” for cases where the it is not receiving a clock during STOP. 0 A Falling edge on COUT has not been detected. 1 Falling edge on COUT has occurred.
0 COUT	Analog Comparator Output Reading the COUT bit will return the current value of the analog comparator output. The register bit is reset to zero and will read as CR1[INV] when the Analog Comparator module is disabled (CR1[EN] = 0). Writes to this bit are ignored.

3.6 Functional Description

The high-speed comparator can be used to compare two analog input voltages applied to Pn and Mn. The analog comparator output (ACO) is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting the CR1[INV] bit to 1.

The SCR[IER] and SCR[IEF] bits select the condition that causes the comparator module to assert an interrupt to the processor. The SCR[CFR] bit is set on a rising edge of the comparator output. The SCR[CFF] bit is set on a falling edge of the comparator output. The (optionally filtered) comparator output can be read directly through the SCR[COUT] bit.

3.6.1 HSCMP Functional Modes

There are three main sub-blocks to the comparator module: the comparator itself, the window function, and the filter function. The filter can be clocked from an internally generated clock or from an external sample input. Additionally, the filter is programmable with respect to how many samples must agree before a change on the output is registered. In the simplest case, only one sample must agree, and the filter acts as a simple sampler.

The external sample input is enabled using the CR1[SE] bit. When this bit is set, the output of the comparator is sampled only on rising edges of the sample input.

The windowing mode is enabled by setting the CR1[WE] bit. When this bit is set, the comparator output is sampled only when the WINDOW input signal equals one. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This feature is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed in the sections that follow.

Table 3-21. Comparator Sample/Filter Controls

Mode	EN	WE	SE	FILTER_CNT	FILT_PER	Operation
1	0	X	X	X	X	**Disabled**
2A	1	0	0	0x0	X	**Continuous Mode**
2B	1	0	0	X	0x00	
3A	1	0	1	0x1	X	**Sampled, Non-Filtered mode**
3B	1	0	0	0x1	> 0x0	
4A	1	0	1	> 0x1	X	**Sampled, Filtered mode**
4B	1	0	0	> 0x1	> 0x0	
5A	1	1	0	0x0	X	**Windowed mode** Comparator output is sampled on every rising peripheral clock edge when SAMPLE=1 to generate COUTA.
5B	1	1	0	X	0x00	
6	1	1	0	0x1	0x01 - 0xFF	**Windowed/Resampled mode** Comparator output is sampled on every rising peripheral clock edge when SAMPLE=1 to generate COUTA, which is then resampled on an interval determined by FILT_PER to generate COUT.
7	1	1	0	> 0x1	0x01 - 0xFF	**Windowed/Filtered mode** Comparator output is sampled on every rising peripheral clock edge when SAMPLE=1 to generate COUTA, which is then resampled and filtered to generate COUT.
All other combinations of EN, WE, SE, FILTER_CNT, and FILT_PER are illegal.						

For cases where a comparator is used to drive a fault input of the PWM, it should generally be configured to operate in continuous mode so that an external fault can immediately pass through the comparator to the PWM fault circuitry.

CAUTION

Filtering and sampling settings should be changed only after setting SE to 0 and FILTER_CNT to 0x0. These values have the effect of resetting the filter to a known state.

3.6.1.1 Disabled Mode (1)

In disabled mode, the analog comparator is non-functional and consumes no power. The output of the analog comparator block (ACO) is zero in this mode. It is possible to further reduce power consumed by disabling the peripheral clock to the comparator logic. This is a function of the system integration module, or SIM.

3.6.1.2 Continuous Mode (2A and 2B)

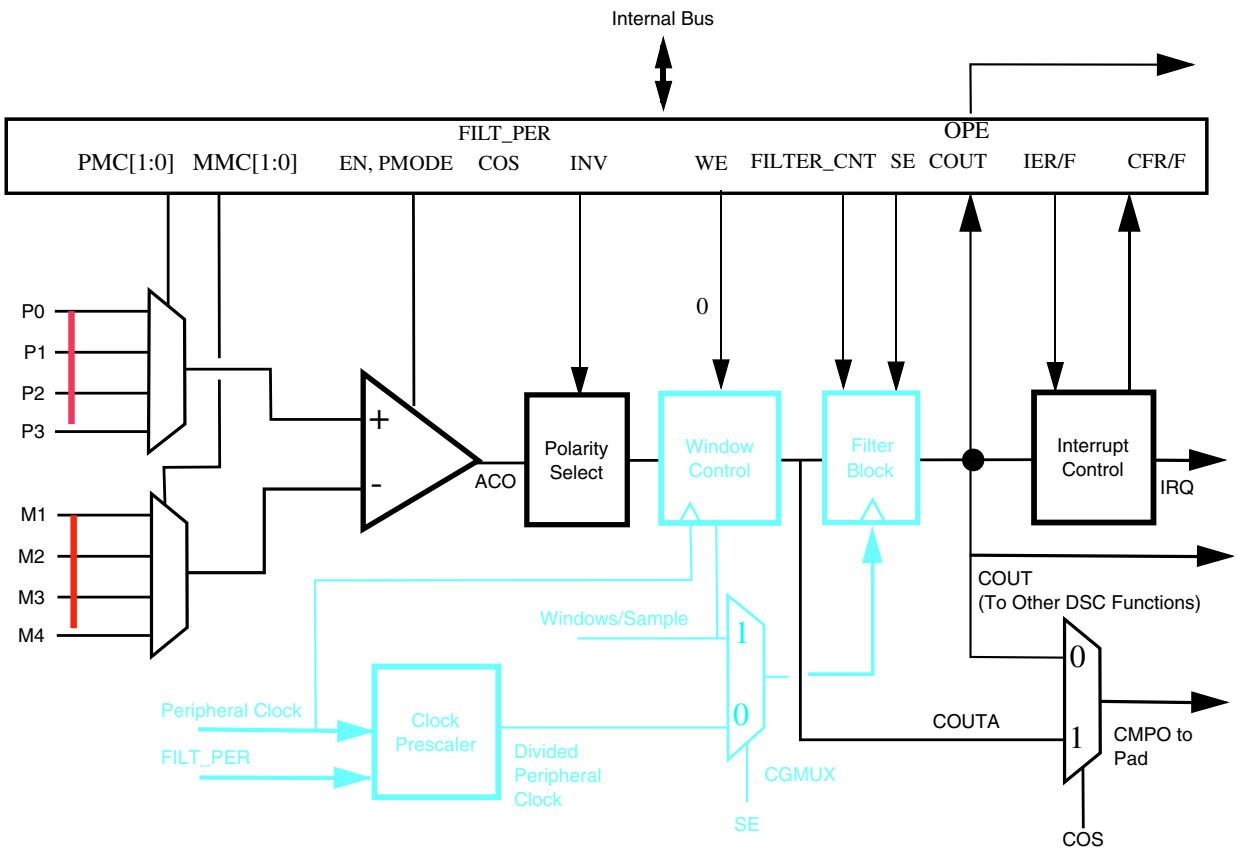


Figure 3-20. Comparator Operation in Continuous Mode

The analog comparator block is powered and active. ACO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. The SCR[COUT] bit is updated continuously. The path from comparator input pins to output pin is operating in combinational (unclocked) mode. COUT and COUTA are identical.

3.6.1.3 Sampled, Non-Filtered Mode (3A and 3B)

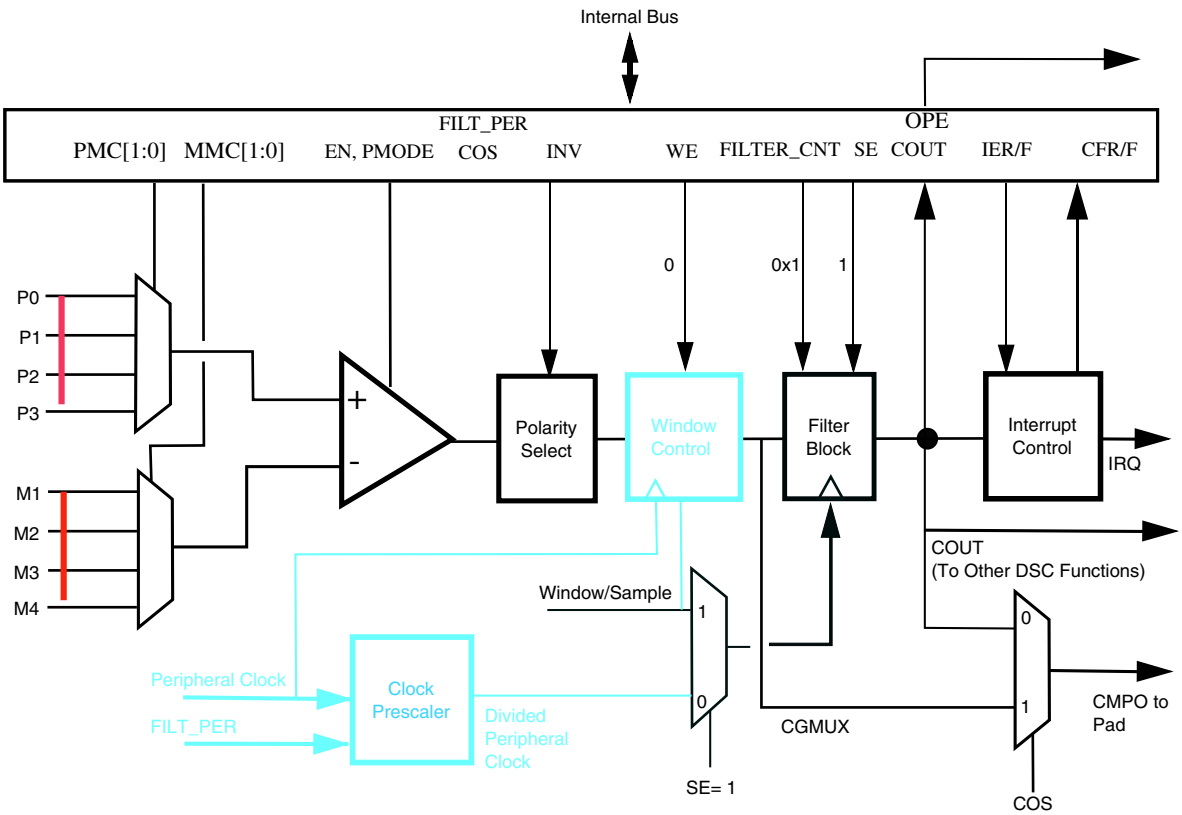


Figure 3-21. Sampled, Non-Filtered (3A): Sampling Point Externally Driven

In sampled, non-filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational (unclocked). Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (3A) and Sampled, Non-Filtered (3B) is in how the clock to the filter block is derived.

The comparator filter has no other function than sample/hold of the comparator output in this mode.

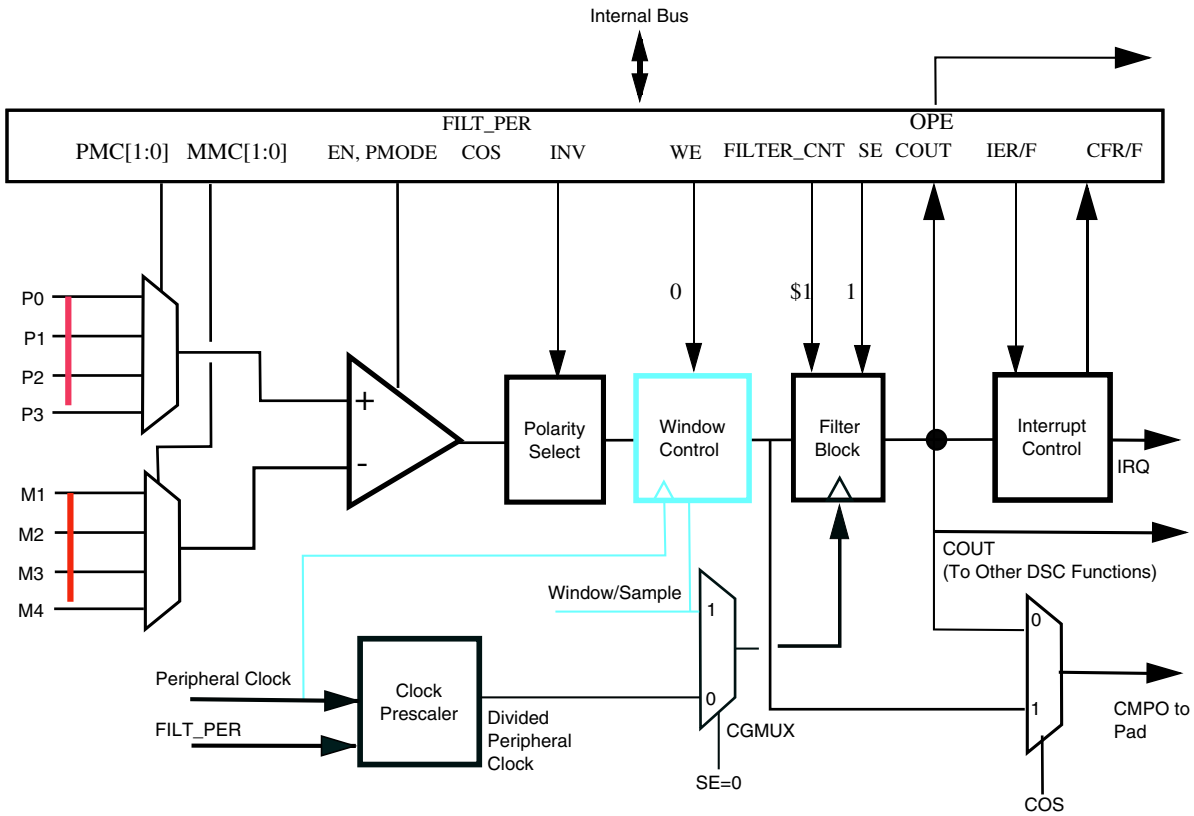


Figure 3-22. Sampled, Non-Filtered (# 3B): Sampling interval internally derived

3.6.1.4 Sampled, Filtered Mode (4A and 4B)

In sampled, filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational (unclocked). Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (3A) and Sampled, Filtered (4A) is that FILTER_CNT is now greater than 1, which activates filter operation.

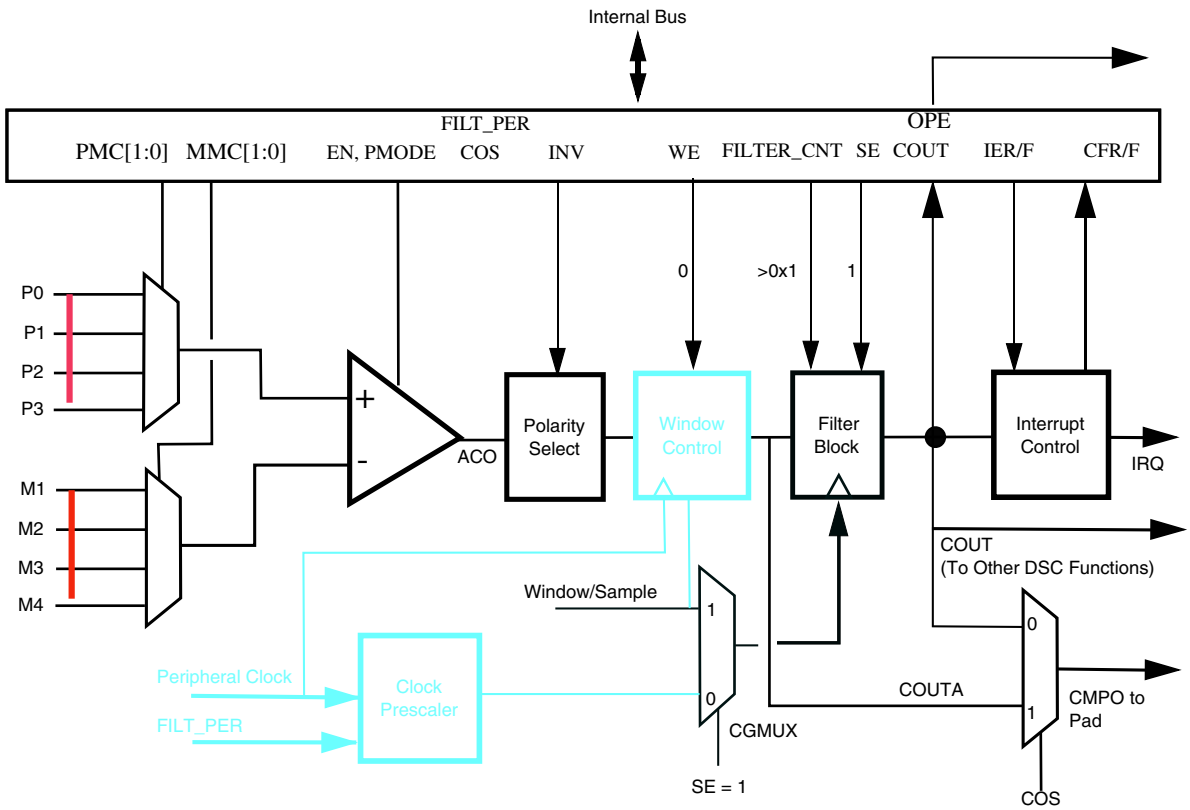


Figure 3-23. Sampled, Filtered (4A): Sampling Point Externally Driven

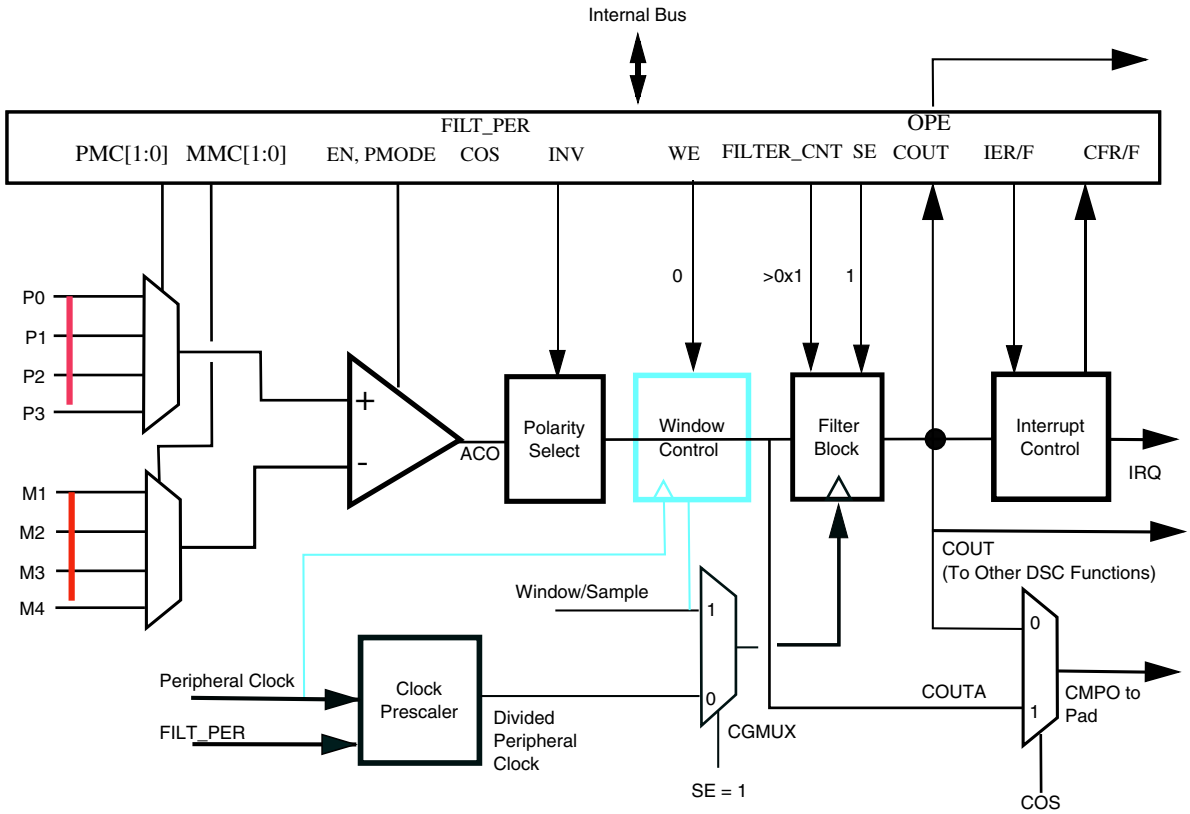


Figure 3-24. Sampled, Filtered (4B): Sampling Point Internally Derived

The only difference in operation between Sampled, Non-Filtered (3B) and Sampled, Filtered (4B) is that FILTER_CNT is now greater than 1, which activates filter operation.

3.6.1.5 Windowed Mode (5A and 5B)

The following image illustrates comparator operation in the windowed mode, ignoring latency of the analog comparator, polarity select and window control block. It also assumes that the polarity select is set to non-inverting. Note that the analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one peripheral clock cycle plus the combinational path delay through the comparator and polarity select logic.

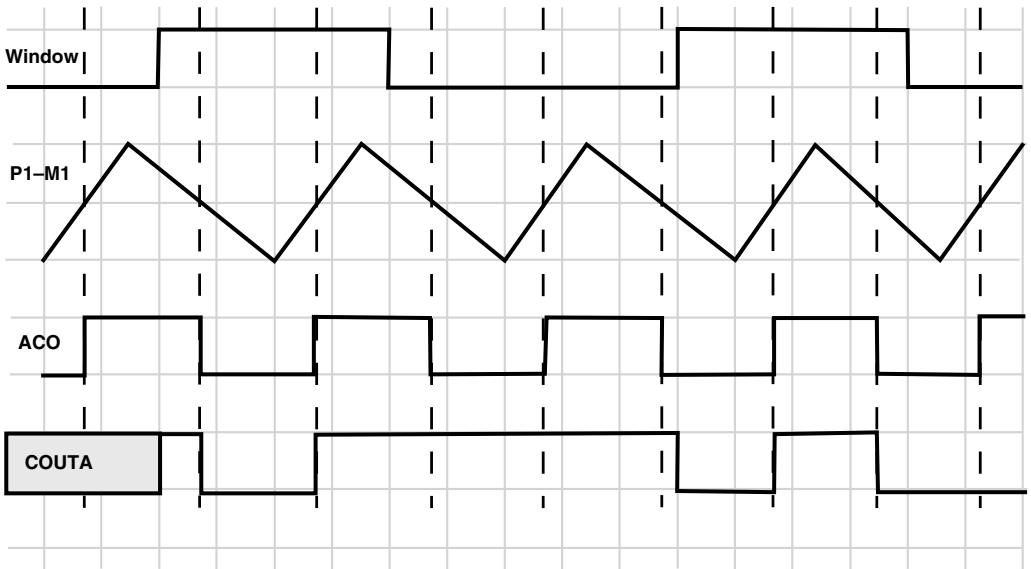


Figure 3-25. Windowed Mode Operation

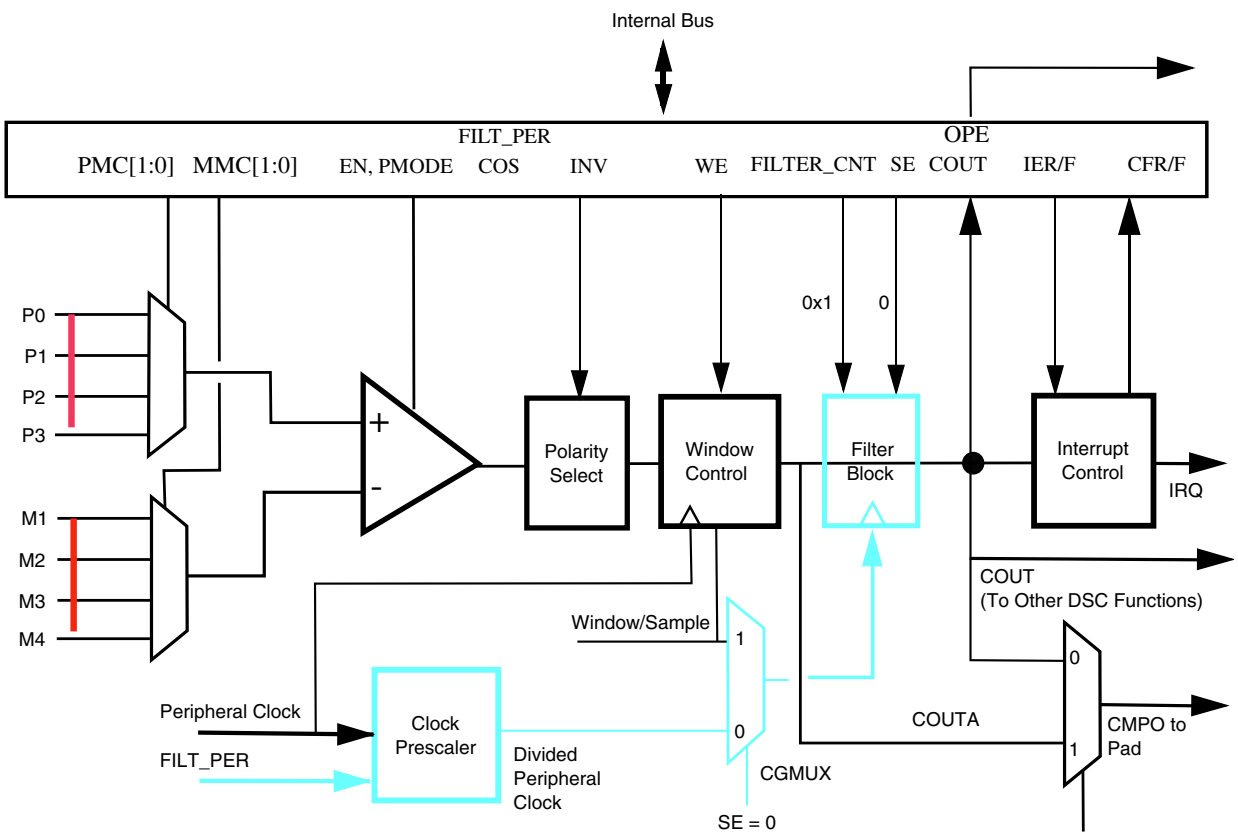


Figure 3-26. Windowed Mode

When any windowed mode is active, COUTA is clocked by the peripheral clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

3.6.1.6 Windowed/Resampled Mode (6)

This example uses the same input stimulus shown in Figure 4-9, and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows. Again, prop delays and latency are ignored for clarity's sake. This example was generated solely to demonstrate operation of the comparator in windowing / resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

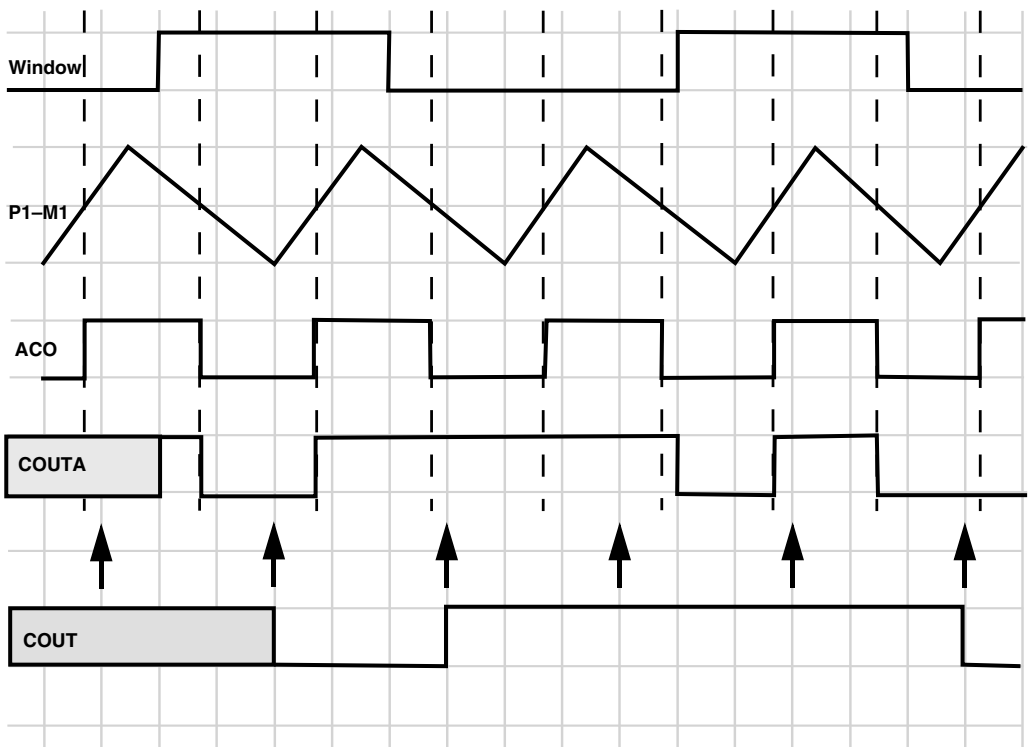


Figure 3-27. Windowed / Resampled Mode Operation

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by FILT_PER and the peripheral clock rate. Configuration for this mode is virtually identical to that for the windowed/filtered mode shown in the next section. The only difference is that the value of FILTER_CNT must be exactly one.

3.6.1.7 Windowed/Filtered Mode (7)

Windowed/filtered mode is the most complex mode of operation for the comparator block because it uses both windowing and filtering features. It also has the highest latency of any mode. This can be approximated: up to 1 peripheral clock synchronization in the window function + ((FILTER_CNT X FILT_PER) + 1) X peripheral clock for the filter function.

When any windowed mode is active, COUTA is clocked by the peripheral clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

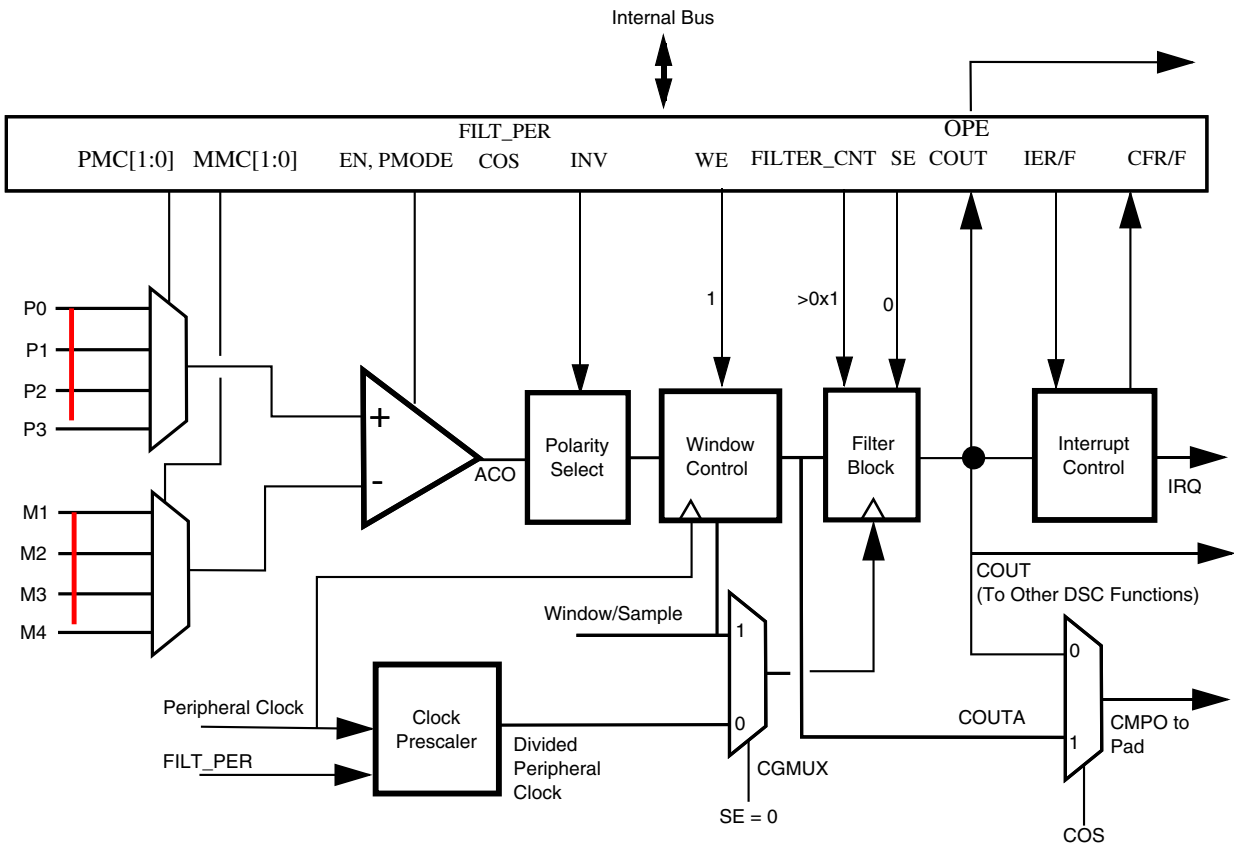


Figure 3-28. Windowed/Filtered Mode

3.6.2 Power Modes

The following table summarizes the terms that apply for each power mode.

Table 3-22. Freescale Power Modes

Power Mode	Comments
Run	Normal operating mode

Table continues on the next page...

Table 3-22. Freescale Power Modes (continued)

Power Mode	Comments
Wait	Processor halted, peripherals continue to run.
Stop	Processor and peripheral clocks halted. Regulator is fully engaged.

3.6.2.1 Wait Mode Operation

During wait mode the HSCMP, if enabled, continues to operate normally. Also, if enabled, the interrupt can wake the DSC core.

3.6.2.2 Stop Mode Operation

The DSC core is brought out of stop when a compare event occurs and corresponding interrupt is enabled. Similarly, if the CR1[OPE] bit is set to 1, the comparator output operates as in the normal operating mode and the comparator output is placed onto the external pin.

If stop is exited with a reset, all comparator registers are put into their reset state.

Windowed, sampled, and filtered modes of operation continue to operate in stop mode if the clock to the peripheral is enabled for stop. Edge detection for compare events also requires a peripheral clock. None of these features function correctly unless the clock to the HSCMP is enabled for stop (via the SIM stop disable registers).

3.6.2.3 Debug Mode Operation

When the DSC is in debug mode, the HSCMP continues to operate normally.

3.6.3 Hysteresis

The following figure illustrates implementation of an external hysteresis resistor bridge between the asynchronous comparator output and the positive (+) input of the comparator. Because positive feedback is required, INV must be set to 0 when the hysteresis resistor bridge is added to the positive (+) input of the comparator. INV must be set to 1 when the hysteresis resistor bridge is added to the negative (–) input of the comparator.

The option of adding an external resistor bridge for the purpose of adding hysteresis to the comparator and the amount of hysteresis depends on the user's individual requirements. Hysteresis can be important in some system designs. In the absence of hysteresis, the continuous comparison of nearly identical analog inputs may add noise and waste power by generating high-frequency oscillations at CMPO.

If external hysteresis is added to the comparator, the bridge must be designed to consider other issues than how much hysteresis is needed. The resistor values must be sufficiently high so that they do not cause the drive strength of the digital output driver in the CMPO IO circuitry to be exceeded. Also, if any digital function other than CMPO must operate on the CMPO pad in the presence of such a bridge, the resistor values must be sufficiently high so that the IO circuitry can function appropriately in its digital role.

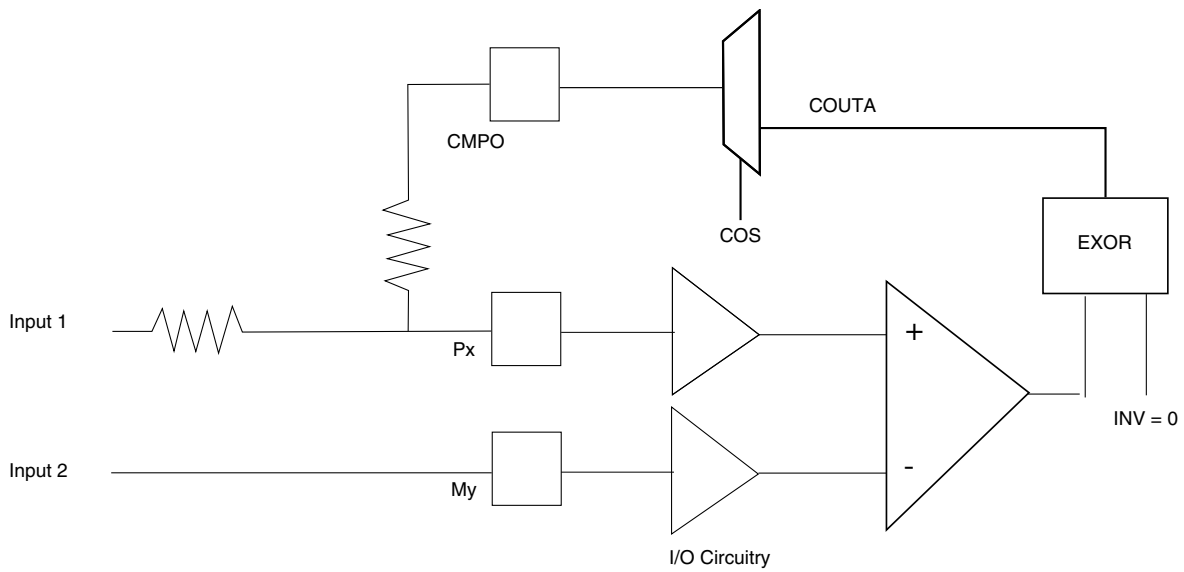


Figure 3-29. External Hysteresis Circuit

3.6.4 Startup and Operation

In a typical startup sequence, the time required to stabilize COUT is the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function, and filter. The windowing function has a maximum of 1 peripheral bus clock period delay. Filter delay is specified in the Low-Pass Filter section.

Interrupts should be disabled during power up, recovery from partial-power-down, and while re-enabling the module. Failure to do so could result in spurious interrupts.

During operation, the propagation delay of the selected data paths must always be considered. It can take many peripheral bus clock cycles for COUT and the CFR/CFF status bits to reflect an input change or a configuration change to one of the components involved in the data path.

When programmed for filtering modes, COUT is initially equal to zero until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a one.

3.6.5 Low Pass Filter

3.6.5.1 Introduction

The low-pass filter operates on the unfiltered, unsynchronized, and optionally-inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by the FPR[FILT_PER] field or by using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high frequency oscillations can be generated at COUTA when the selected N and P input voltages differ by less than the offset voltage of the differential comparator.

3.6.5.2 Enabling Filter Modes

Filter modes are enabled by setting the CR0[FILTER_CNT] field to be greater than 0x1 and setting the FPR[FILT_PER] to a non-zero value *or* setting SE to 1. If the divided peripheral clock is driving the filter, it takes samples of COUTA every FILT_PER peripheral bus cycles.

The filter output is at zero when first initialized and subsequently changes when the FILTER_CNT field's consecutive samples all agree that the output value has changed. Said another way, COUT is zero for some initial period, even when COUTA is at one.

Setting both SE and FILT_PER to 0 disables the filter and eliminates switching current associated with the filtering process. Always switch to this setting before changing any filter parameters. It resets the filter to a known state. Switching the FILTER_CNT field on the fly without this intermediate step can result in unexpected behavior.

If SE is 1, the filter takes samples of COUTA on each positive transition of the SAMPLE input. The output state of the filter changes when the FILTER_CNT field's consecutive samples all agree that the output value has changed.

3.6.5.3 Latency Issues

The FILT_PER field's value (or SAMPLE period) should be set such that the sampling period is just larger the period of the expected noise. This way a noise spike corrupts only one sample. The FILTER_CNT field's value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of the FILTER_CNT field's value.

The following table summarizes maximum latency values for the various modes of operation in the absence of noise. Filtering latency is restarted each time an actual output transition is masked by noise.

The values of the FILT_PER field (or SAMPLE period) and the FILTER_CNT field must also be balanced against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of the FILTER_CNT field's value.

Table 3-23. Comparator Sample/Filter Maximum Latencies

Mode	EN	WE	SE	FILTER_CNT	FILT_PER	Operation	Maximum Latency[1]
1	0	X	X	X	X	**Disabled**	N/A
2A	1	0	0	0x0	X	**Continuous Mode**	T_{PD}
2B	1	0	0	X	0x00		
3A	1	0	1	0x1	X	**Sampled, Non-Filtered mode**	$T_{PD} + T_{SAMPLE} + T_{per}$
3B	1	0	0	0x1	> 0x0		$T_{PD} + (FILT\_PER \times T_{per}) + T_{per}$
4A	1	0	1	> 0x1	X	**Sampled, Filtered mode**	$T_{PD} + (FILTER\_CNT \times T_{SAMPLE}) + T_{per}$
4B	1	0	0	> 0x1	> 0x0		$T_{PD} + (FILTER\_CNT \times FILT\_PER \times T_{per}) + T_{per}$
5A	1	1	0	0x0	X	**Windowed mode**	$T_{PD} + T_{per}$
5B	1	1	0	X	0x00		$T_{PD} + T_{per}$

Table continues on the next page...

Table 3-23. Comparator Sample/Filter Maximum Latencies (continued)

Mode	EN	WE	SE	FILTER_CNT	FILT_PER	Operation	Maximum Latency[1]
6	1	1	0	0x1	0x01 - 0xFF	**Windowed / Re-sampled mode**	T_{PD} + (FILT_PER X T_{per}) + $2T_{per}$
7	1	1	0	> 0x1	0x01 - 0xFF	**Windowed / Filtered mode**	T_{PD} + (FILTER_CNT X FILT_PER X T_{per}) + $2T_{per}$

1. T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{SAMPLE} is the clock period of the external sample clock. T_{per} is the period of the peripheral bus clock.

3.7 Interrupts

The HSCMP module can generate an interrupt on either the rising or falling edge of the comparator output (or on both edges). The interrupt request is asserted when both the SCR[IER] bit and the SCR[CFR] bit are set. It is also asserted when both the SCR[IEF] bit and the SCR[CFF] bit are set. The interrupt is deasserted by clearing either the SCR[IER] or SCR[CFR] bit for a rising-edge interrupt, or by clearing the SCR[IEF] and SCR[CFF] bits for a falling-edge interrupt.

Chapter 4
5-Bit Voltage Reference Digital-to-Analog Converter (VREF_DAC)

4.1 Introduction

4.1.1 Overview

The 5-bit VREF_DAC is 32-tap resistor ladder network that provides a selectable voltage reference for applications that need a voltage reference. The 32-tap resistor ladder network divides the supply reference V_{DDA} into 32 voltage levels. A 5-bit digital signal input selects the output voltage level, which varies from V_{DDA} to $V_{DDA}/32$.

4.1.2 Features

Features of the 5-bit VREF_DAC include:

- 5-bit resolution
- Power down mode to conserve power when it is not being used
- Output that is internally routed to internal analog comparator positive input 0
  - 5bDACAO is internally routed to CMPA_P0
  - 5bDACBO is internally routed to CMPB_P0
  - 5bDACCO is internally routed to CMPC_P0
- Ability to remain operating in STOP mode

4.1.3 Block Diagram

The following figure is a block diagram of the 5-bit VREF_DAC module. It contains a 32-tap resistor ladder network and a 32-to-1 multiplexer, which selects an output voltage from 1 of 32 distinct levels that outputs from DACnO. The module can be disabled and powered down when it is not used. When the VREF_DACn is disabled, DACnO is connected to V_{SSA}.

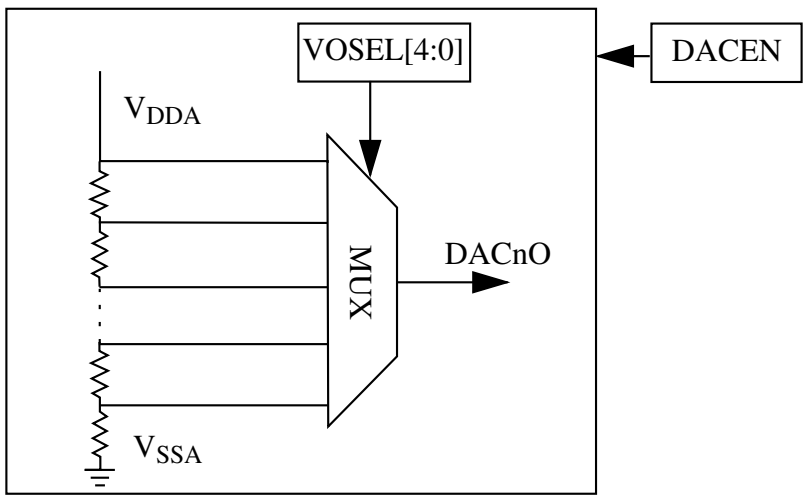


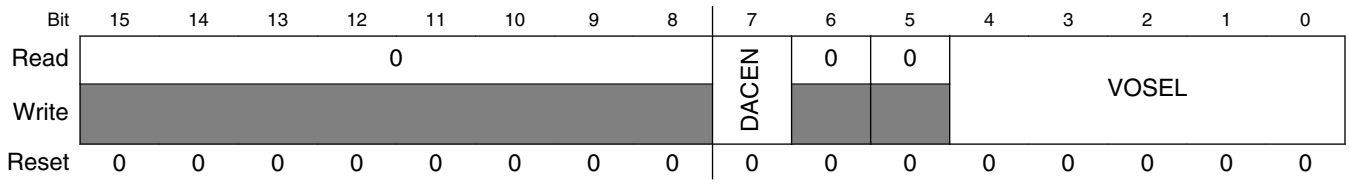
Figure 4-1. DAC Block Diagram

4.2 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	REFA_ DACCTRL	R	0								DACEN	0	0	VOSEL				
		W																
0	REFB_ DACCTRL	R	0								DACEN	0	0	VOSEL				
		W																
0	REFC_ DACCTRL	R	0								DACEN	0	0	VOSEL				
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

4.2.1 Voltage Reference DAC Control Register (REFx_DACCTRL)

Addresses: REFA_DACCTRL – F240h base + 0h offset = F240h
REFB_DACCTRL – F250h base + 0h offset = F250h
REFC_DACCTRL – F260h base + 0h offset = F260h



REFx_DACCTRL field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 DACEN	DAC Enable This bit is used to enable the module. When the module is disabled, the module is powered down to conserve power. 0 DAC is disabled (powered down) and output is Vssa 1 DAC is operational
6 Reserved	This read-only bit is reserved and always has the value zero.
5 Reserved	This read-only bit is reserved and always has the value zero.
4–0 VOSEL	DAC Output Voltage Select This bitfield selects an output voltage from one of 32 distinct levels: DACO = (V_{DDA}/32)×(VOSEL[4:0] +1), where the DACO range is from V_{DDA}/32 to V_{DDA}.

4.3 Functional Description

4.3.1 Operation

A 32-to-1 multiplexer selects an output voltage from 1 of 32 distinct levels that are generated from a 32-tap resistor ladder network. The Reference DAC control register (DACCTRL) enables the selection of the output voltage level and of the resistor ladder network's supply reference source. When the module is disabled, its output, DACO, should be equal to (is connected to) V_{SSA}. The Reference DAC output is internally routed to the positive input 0 of one of the comparators.

4.4 Resets

This module has a single reset input, which corresponds to the device-wide peripheral reset.

4.5 Clocks

This module has a single clock input: the bus peripheral clock.

4.6 Interrupts

This module has no interrupts.

Chapter 5
12-Bit Digital-to-Analog Converter (DAC)

5.1 Introduction

5.1.1 Overview

The 12-bit voltage Digital-to-Analog converter (DAC) provides a reference to on-chip comparators or an output to a package pin. It can also be used as a waveform generator to generate square, triangle, and sawtooth waveforms. The output range of this DAC is between V_{SSA} and V_{DDA}. This DAC can be put in power down mode if needed. Data presented to this DAC is buffered so that the DAC conversion is controllable by other module outputs.

5.1.2 Features

DAC features include:

- 12-bit resolution
- Powerdown mode
- Output can be routed to the internal comparator or off the device
- Choice of asynchronous or synchronous updates

  (Sync input can be connected to internal crossbar)
- Automatic mode to generate square, triangle, and sawtooth output waveforms
- Automatic mode to allow programmable period, update rate, and range
- Support of two digital formats
- Glitch filter to suppress output glitch during data conversion

5.1.3 Block Diagram

The following figure illustrates the DAC configuration.

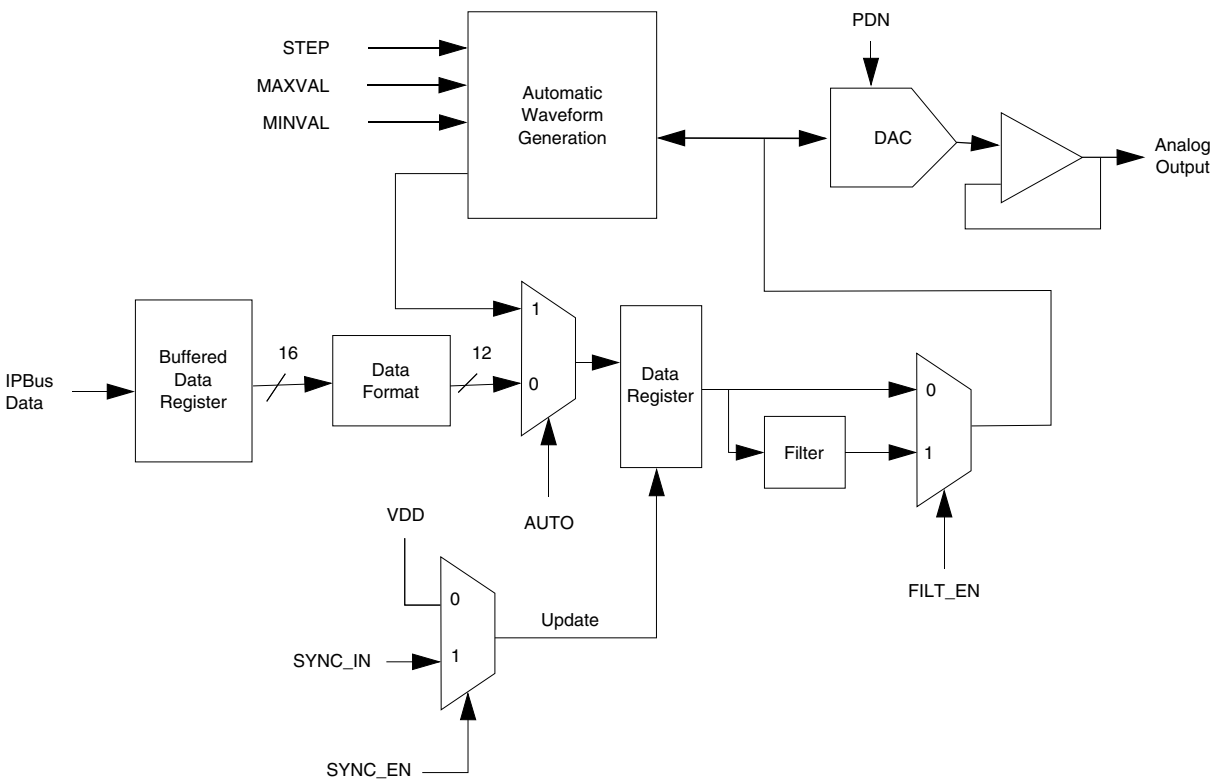


Figure 5-1. DAC Block Diagram

5.2 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DAC_CTRL	R	FILT_CNT			FILT_EN	0						UP	DOWN	SYNC_EN	AUTO	FORMAT	PDN
		W																
1	DAC_DATA [FORMAT=0]	R	0				DATA											
		W																
1	DAC_DATA [FORMAT=1]	R	DATA												0			
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2	DAC_STEP [FORMAT=0]	R	0				STEP											
		W																
2	DAC_STEP [FORMAT=1]	R	STEP												0			
		W																
3	DAC_MINVAL [FORMAT=0]	R	0				MINVAL											
		W																
3	DAC_MINVAL [FORMAT=1]	R	MINVAL												0			
		W																
4	DAC_MAXVAL [FORMAT=0]	R	0				MAXVAL											
		W																
4	DAC_MAXVAL [FORMAT=1]	R	MAXVAL												0			
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

5.2.1 Control Register (DAC_CTRL)

Address: DAC_CTRL – F1A0h base + 0h offset = F1A0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FILT_CNT			FILT_EN	0						UP	DOWN	SYNC_EN	AUTO	FORMAT	PDN
Write																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1

DAC_CTRL field descriptions

Field	Description
15–13 FILT_CNT	Glitch Filter Count Bits On chips whose maximum IPBus clock frequency is 32MHz or less, this field represents the number of IPBus clock cycles that the DAC output is held unchanged after new data is presented to the analog DAC's inputs. On chips that have IPBus clock frequencies higher than 32MHz, then the number of IPBus clock cycles that are delayed is 2*CTRL[FILT_CNT]+1. Approximately 240nsec is needed for worst case settling of the DAC output, therefore, a value of 7 should be used for both 32MHZ operation (7 IPBus clock cycles of delay) and for 60MHz operation (15 IPBus clock cycles of delay). **NOTE:** When using the glitch filter be sure that the filter count is less than the update count otherwise the DAC output will never be updated.
12 FILT_EN	Glitch Filter Enable This bit enables the glitch suppression filter. This introduces a latency based on CTRL[FILT_CNT] for DAC updates.

Table continues on the next page...

DAC_CTRL field descriptions (continued)

Field	Description
	NOTE: Do not set CTRL[FILT_EN] when CTRL[AUTO] is 1 and CTRL[SYNC_EN] is 0. When CTRL[AUTO] is set and CTRL[SYNC_EN] is clear, it indicates that the data to the DAC should be updated every clock cycle. Enabling the filter in this situation would cause the glitch filter to filter the changes in the data being presented to the DAC and result in no change to the DAC output. 0 Filter disabled. 1 Filter enabled.
11–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5 UP	Enable Up Counting This bit enables counting up in automatic mode. See the functional description of automatic mode to understand how this bit affects automatic waveform generation. 0 Up counting disabled. 1 Up counting enabled.
4 DOWN	Enable Down Counting This bit enables counting down in automatic mode. See the functional description of automatic mode to understand how this bit affects automatic waveform generation. 0 Down counting disabled. 1 Down counting enabled.
2 SYNC_EN	Sync Enable This bit enables the SYNC_IN input to be used to trigger an update of the buffered data being presented to the analog DAC input. If CTRL[SYNC_EN] is clear, then asynchronous mode is indicated and data written to the Buffered Data Register (DATA) will be presented to the analog DAC input in the following IPBus clock cycle. 0 Asynchronous mode. Data written to the DATA register is presented to DAC inputs on the next clock cycle. 1 Synchronous mode. A rising edge of SYNC_IN updates the data in the DATA register presented to the DAC input.
2 AUTO	Automatic Mode This bit enables automatic waveform generation mode. In automatic mode an external source (typically a timer module) driving SYNC_IN determines the data update rate while the STEP, MINVAL, and MAXVAL registers and CTRL[UP] and CTRL[DOWN] are used to shape the waveform. If CTRL[SYNC_EN] is not set when using this mode, then the data for the analog DAC will be updated every clock cycle, but the DAC output may be unable to keep up with this update rate. 0 Normal mode. Automatic waveform generation disabled. 1 Automatic waveform generation enabled.
1 FORMAT	Data Format Two data formats can be used for the DAC. When this bit is clear, the 12 bits of data are right justified within the 16 bit DATA register. When this bit is set, the 12 bits of data are left justified. In either case the 4 unused bits are ignored.

Table continues on the next page...

DAC_CTRL field descriptions (continued)

Field	Description
	0 Data words are right justified. (default) 1 Data words are left justified.
0 PDN	Power Down This bit is used to power down the analog portion of the DAC (resulting in its output being pulled low) when not in use. This bit does not reset the registers and upon clearing of CTRL[PDN] the analog DAC will output the value currently presented to its inputs. The analog block requires 10 usec to recover from the power down state before proper operation is guaranteed. 0 DAC is operational. 1 DAC is powered down. (default)

5.2.2 Buffered Data Register (DAC_DATA [FORMAT=0])

Address: DAC_DATA [FORMAT=0] – F1A0h base + 1h offset = F1A1h

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

DAC_DATA [FORMAT=0] field descriptions

Field	Description
15–12 Reserved	This read-only bitfield is reserved and always has the value zero.
11–0 DATA	DAC data (right justified) Data written to this register is held in a buffer. The digital data contained in this buffer will be presented to the analog DAC upon the rising edge of the SYNC_IN signal (or at the next clock cycle if CTRL[SYNC_EN] is clear) and converted to analog and output by the DAC. Reading this register returns the value being presented to the analog DAC which may differ from the data value being held in the write buffer. The data in this buffer can be updated at any rate, but the DAC output load impedance may affect the updating rate.

5.2.3 Buffered Data Register (DAC_DATA [FORMAT=1])

Address: DAC_DATA [FORMAT=1] – F1A0h base + 1h offset = F1A1h

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

DAC_DATA [FORMAT=1] field descriptions

Field	Description
15–4 DATA	DAC data (left justified) Data written to this register is held in a buffer. The digital data contained in this buffer will be presented to the analog DAC upon the rising edge of the SYNC_IN signal (or at the next clock cycle if CTRL[SYNC_EN] is clear) and converted to analog and output by the DAC. Reading this register returns the value being presented to the analog DAC which may differ from the data value being held in the write buffer. The data in this buffer can be updated at any rate, but the DAC output load impedance may affect the updating rate.
3–0 Reserved	This read-only bitfield is reserved and always has the value zero.

5.2.4 Step Size Register (DAC_STEP [FORMAT=0])

Address: DAC_STEP [FORMAT=0] – F1A0h base + 2h offset = F1A2h

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

DAC_STEP [FORMAT=0] field descriptions

Field	Description
15–12 Reserved	This read-only bitfield is reserved and always has the value zero.
11–0 STEP	STEP size (right justified) When the DAC is in automatic mode (CTRL[AUTO] = 1), the step size contained in this register will be added to or subtracted from the current value to create the next value presented to the DAC inputs. This register is not used during normal mode operation, but can still be written to and read from.

5.2.5 Step Size Register (DAC_STEP [FORMAT=1])

Address: DAC_STEP [FORMAT=1] – F1A0h base + 2h offset = F1A2h

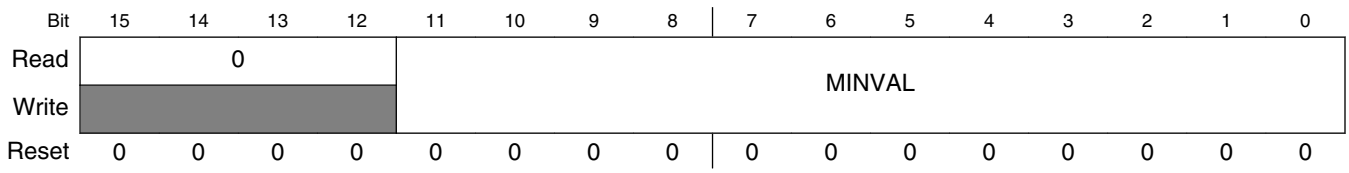
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	STEP												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_STEP [FORMAT=1] field descriptions

Field	Description
15–4 STEP	STEP size (left justified) When the DAC is in automatic mode (CTRL[AUTO] = 1), the step size contained in this register will be added to or subtracted from the current value to create the next value presented to the DAC inputs. This register is not used during normal mode operation, but can still be written to and read from.
3–0 Reserved	This read-only bitfield is reserved and always has the value zero.

5.2.6 Minimum Value Register (DAC_MINVAL [FORMAT=0])

Address: DAC_MINVAL [FORMAT=0] – F1A0h base + 3h offset = F1A3h

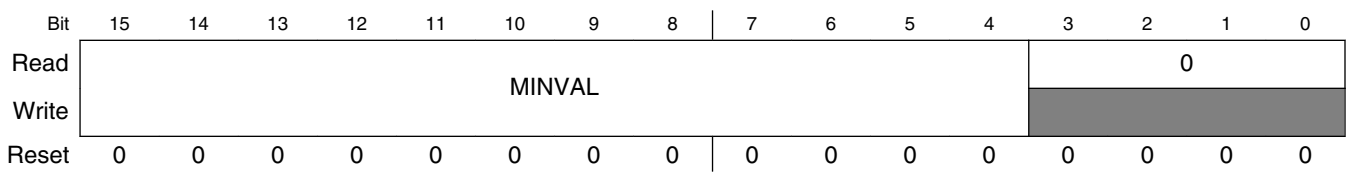


DAC_MINVAL [FORMAT=0] field descriptions

Field	Description
15–12 Reserved	This read-only bitfield is reserved and always has the value zero.
11–0 MINVAL	Minimum value (right justified) When the DAC is in automatic mode (CTRL[AUTO] = 1), the minimum value contained in this register will act as the lower range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Please refer to the chip data sheet for limitations on the low end voltage output of the DAC. **NOTE:** If DAC input data is less than MINVAL, output will be limited to MINVAL during automatic waveform generation.

5.2.7 Minimum Value Register (DAC_MINVAL [FORMAT=1])

Address: DAC_MINVAL [FORMAT=1] – F1A0h base + 3h offset = F1A3h



DAC_MINVAL [FORMAT=1] field descriptions

Field	Description
15–4 MINVAL	Minimum value (left justified) When the DAC is in automatic mode (CTRL[AUTO] = 1), the minimum value contained in this register will act as the lower range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Please refer to the chip data sheet for limitations on the low end voltage output of the DAC. **NOTE:** If DAC input data is less than MINVAL, output will be limited to MINVAL during automatic waveform generation.
3–0 Reserved	This read-only bitfield is reserved and always has the value zero.

5.2.8 Maximum Value Register (DAC_MAXVAL [FORMAT=0])

Address: DAC_MAXVAL [FORMAT=0] – F1A0h base + 4h offset = F1A4h

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

DAC_MAXVAL [FORMAT=0] field descriptions

Field	Description
15–12 Reserved	This read-only bitfield is reserved and always has the value zero.
11–0 MAXVAL	Maximum value (right justified) When the DAC is in automatic mode (CTRL[AUTO] = 1), the maximum value contained in this register will act as the upper range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Please refer to the chip data sheet for limitations on the high end voltage output of the DAC. **NOTE:** If DAC input data is greater than MAXVAL, output will be limited to MAXVAL during automatic waveform generation.

5.2.9 Maximum Value Register (DAC_MAXVAL [FORMAT=1])

Address: DAC_MAXVAL [FORMAT=1] – F1A0h base + 4h offset = F1A4h

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

DAC_MAXVAL [FORMAT=1] field descriptions

Field	Description
15–4 MAXVAL	Maximum value (left justified) When the DAC is in automatic mode (CTRL[AUTO] = 1), the maximum value contained in this register will act as the upper range limit during automatic waveform generation. This register is not used during normal mode operation, but can still be written to and read from. Please refer to the chip data sheet for limitations on the high end voltage output of the DAC. **NOTE:** If DAC input data is greater than MAXVAL, output will be limited to MAXVAL during automatic waveform generation.
3–0 Reserved	This read-only bitfield is reserved and always has the value zero.

5.3 Functional Description

5.3.1 Conversion modes

This DAC supports two conversion modes: asynchronous conversion and synchronous conversion.

5.3.1.1 Asynchronous conversion mode

Data can be immediately presented to the DAC and converted to an analog output when it is written to the DAC buffered data register.

5.3.1.2 Synchronous conversion mode

Data in the DAC buffered data register is controlled by the SYNC_IN signal when the buffered data is presented to the input of the DAC. The update occurs on the rising edge of the SYNC_IN signal. The SYNC_IN signal can come from a timer, comparators, pins, or other sources. The CPU needs to update the buffered data register prior to the next SYNC_IN rising edge. Otherwise, the old buffered data is reused. Note: The SYNC_IN single must be high for at least one IPBus clock cycle and must be low for at least one IPBus clock cycle.

5.3.2 Operation Modes

The DAC operates in either Normal or Automatic mode. In Normal mode, it generates an analog representation of digital words. In Automatic mode, it generates sawtooth, triangle, and square waveforms without CPU intervention.

5.3.2.1 Normal Mode

The DAC receives data words through a memory-mapped register on the IPBus (DATA). A digital word is applied to the DAC inputs based on CTRL[SYNC_EN]. In the worst case with no DAC output load, approximately 240 ns settling time is needed.

5.3.2.2 Automatic Mode

In Automatic mode, the DAC generates sawtooth, triangle, and square wave waveforms without CPU intervention. The update rate, incremental step size, and minimum and maximum values are programmable.

The value in the DATA register is used as a starting-point for the following process:

1. The SYNC_IN input indicates that it is time to update the data presented to the DAC.
2. The STEP value is added to/subtracted from the current DATA value and DATA is updated.
3. If CTRL[UP] is set, then STEP is added to DATA each update until MAXVAL is reached.
4. The generator starts subtracting STEP from DATA if CTRL[DOWN] is set (down counting enabled) or reloading MINVAL if CTRL[DOWN] is clear (no down counting).
5. When DATA reaches MINVAL while counting down, the generator starts counting up if CTRL[UP] is set or reloads MAXVAL if CTRL[UP] is clear (up counting disabled).

The initial count direction depends on which bit (CTRL[UP] or CTRL[DOWN]) is set last.

The following figures show examples of automatically-generated waveforms. The waveforms shown are ideal. Actual waveforms are limited by the slew rate of the DAC output.

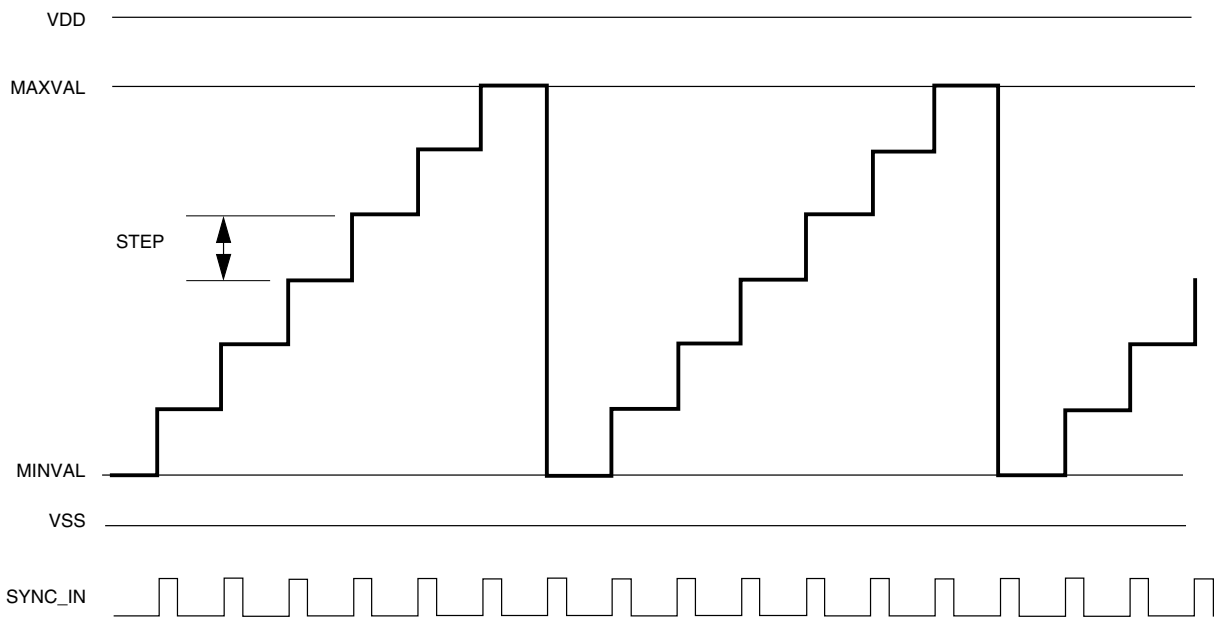


Figure 5-11. Sawtooth Waveform Example with CTRL[UP]=1 and CTRL[DOWN]=0

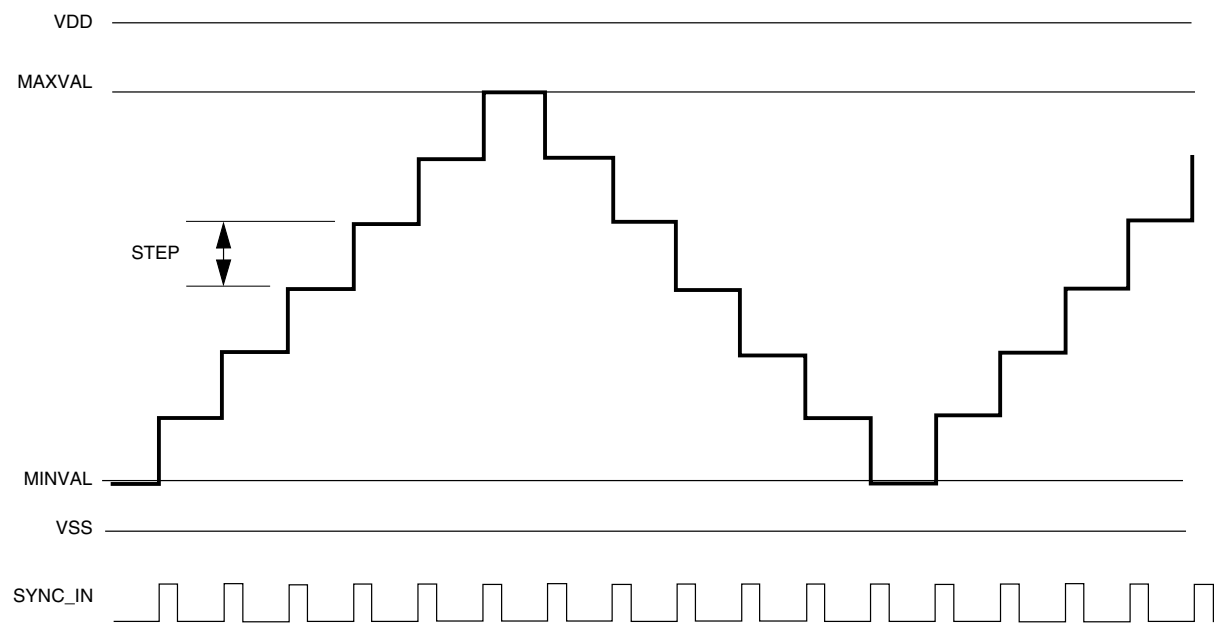


Figure 5-12. Triangle Waveform Example with CTRL[UP]=1 and CTRL[DOWN]=1

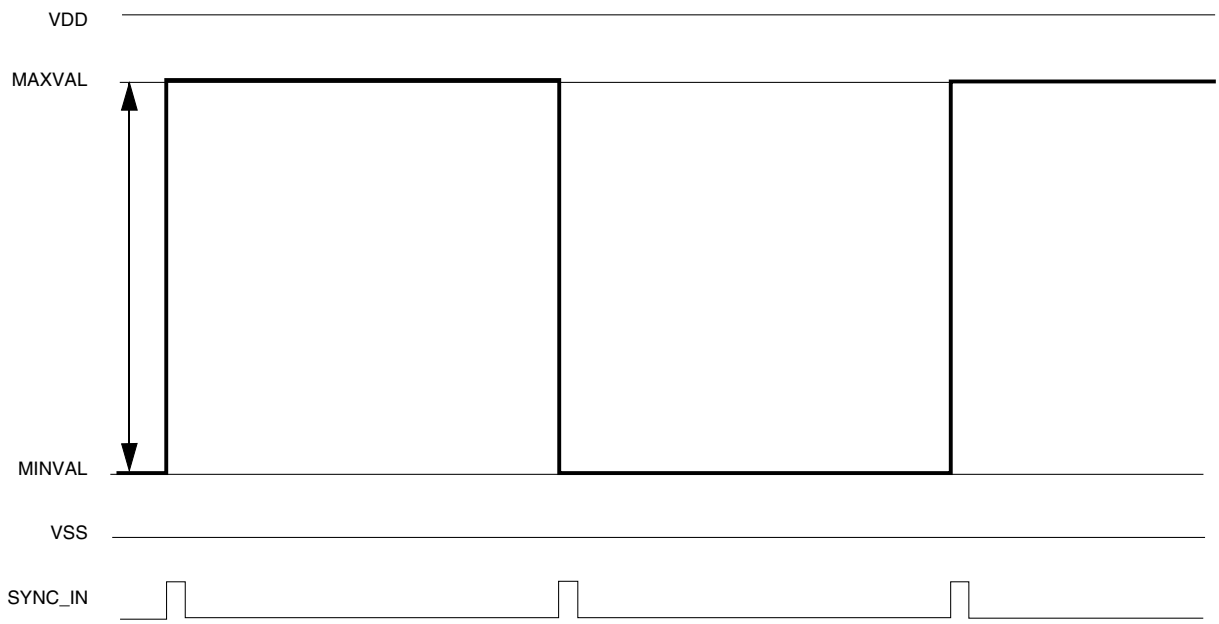


Figure 5-13. Square Waveform Example with CTRL[UP]=1 and CTRL[DOWN]=1

These examples show that the waveform period is a function of the difference between MAXVAL and MINVAL, the STEP size, and the update rate as shown here:

$$\text{Period} = \left[\frac{\text{MAXVAL} - \text{MINVAL}}{\text{STEP}} \times \text{UpdatePeriod}\right] \times 2$$

Increasing STEP decreases the resolution of the output steps. Increasing the update rate decreases the waveform period. Varying MINVAL and MAXVAL changes the DC offset and the amplitude of the waveform.

5.3.3 DAC settling time

Settling time is the interval, within a specified percentage error band, between the time data is presented to the DAC input to update (change) and the time its analog output value reaches its final value. Settling time is affected by circuit propagation delay and the slew rate of the DAC output capability, plus the real load on DAC output.

Approximately 240 ns settling time, which is caused by the DAC conversion glitch, is needed in the worst case situations when DAC output is internally fed to other modules, such as comparator inputs. If DAC output is to a package pin, the additional settling time is affected by the slew rate of an output amplifier and the load on the pin. The maximum settling time will not exceed 2 microseconds with a maximum output load (3 kohm || 400 pf) when the output swings from minimum output to maximum output or vice-versa.

5.3.4 Waveform Programming Example

To create a waveform that goes down from 3.0 V to 1.5 V in 1 millisecond, first calculate MAXVAL and MINVAL. Based on each DAC LSB representing 0.806 mV (assuming the DAC reference is 3.3 V), we find that MAXVAL is 0xE8A and MINVAL is 0x745. These numbers represent a difference of 1861 LSBs to be accomplished in 1 millisecond, so we can safely update the DAC 500 times because the DAC has a maximum 2-microsecond settling time. Programming calculations are as follows:

- To go from MAXVAL to MINVAL in 500 steps, STEP must be 0x004 after rounding the result of 1861/500.
- To go from MAXVAL to MINVAL with a STEP size of 0x004 requires 465.25 steps.
- To keep the waveform period of 1 millisecond using 465.25 updates requires an update period of 465.25 KHz.
- If the system clock operates at 60 MHz, program a timer module to pulse the SYNC_IN input every 60000000/465250 = 129 counts.

When the timer module is programmed along with the MAXVAL, MINVAL, and STEP registers, the DATA register is written with a value equal to MAXVAL as a starting point because the DAC is only down counting. When writing to the DATA, STEP, MAXVAL, and MINVAL registers, ensure that FORMAT has the desired value and that the data values are properly justified to match FORMAT. The SYNC_EN, DOWN, and AUTO bits of the CTRL register should be set. The CTRL[UP] and CTRL[PDN] bits are cleared. To suppress glitches on the output, set CTRL[FILT_CNT] and CTRL[FILT_EN]. The desired waveform starts within 12 microseconds from the clearing of CTRL[PDN] and continues until CTRL[PDN] is set or the timer is stopped.

5.3.5 Sources of Waveform Distortion

5.3.5.1 Switching Glitches

When a new digital value is presented to the DAC input, some glitches may appear on the output as the new values propagate through the circuitry. Eventually, these glitches settle out and the output slews to its new value. To avoid these glitches, set CTRL[FILT_EN] and give CTRL[FILT_CNT] a suitable value to cause the DAC to hold its current output for a number of clock cycles equal to 2 * CTRL[FILT_CNT] + 1 while the switching glitches settle out. After the filter time is satisfied, the output smoothly slews to the new value.

5.3.5.2 Slew Effects

The example waveforms are ideal waveforms and show transitions as step functions. In reality, the DAC output has a finite slew rate that rounds off the steps. Whether this rounding off is noticeable depends on the output step size (larger output changes require longer settling times) and on the update period (longer dwell times make the settling times less noticeable).

5.3.5.3 Clipping Effects (Automatic Mode Only)

One form of clipping occurs during automatic waveform generation when the difference between MAXVAL and MINVAL is not a (near) even multiple of the STEP value. This results in a partial step as the waveform approaches MAXVAL and MINVAL. The following figure shows an example.

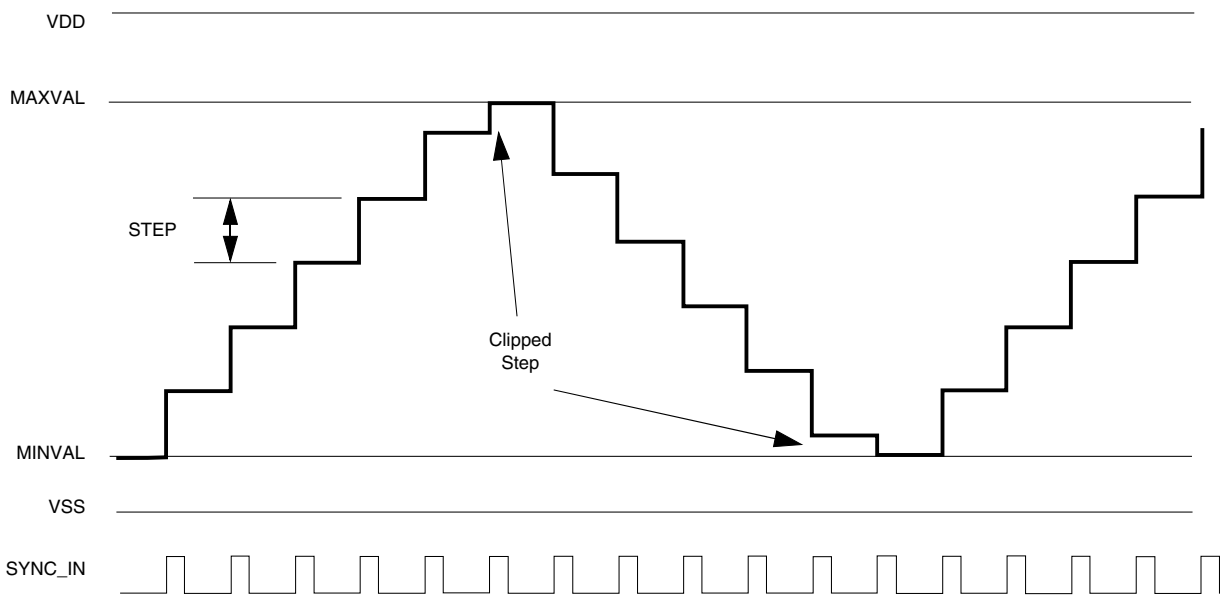


Figure 5-14. Triangle Waveform Example with Clipping

Another form of clipping occurs when the MAXVAL or MINVAL is beyond the output range of the DAC. The maximum and minimum voltages that can be driven out are defined in the device data sheet.

5.4 Resets

When reset, all of the registers return to the reset state.

5.5 Clocks

The DAC uses the system bus clock (IPBus clock).

5.6 Interrupts

The DAC module does not generate any interrupts.

Chapter 6
Quad Timer (TMR)

6.1 Overview

Each timer module (TMR) contains four identical counter/timer groups. Each 16-bit counter/timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status and control registers, and one control register. All of the registers except the prescaler are read/writable. NOTE: This document uses the terms "Timer" and "Counter" interchangeably because the counter/timers may perform either or both tasks.

The load register provides the initialization value to the counter when the counter's terminal value has been reached.

The hold register captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters.

The capture register enables an external signal to take a "snap shot" of the counter's current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The prescaler provides different time bases useful for clocking the counter/timer.

The counter provides the ability to count internal or external events.

Within a timer module (set of four timer/counters), the input pins are shareable.

6.2 Features

The TMR module design includes these distinctive features:

- Four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulo
- Max count rate equals peripheral clock/2 for external clocks
- Max count rate equals peripheral clock for internal clocks
- Count once or repeatedly
- Counters are preloadable
- Compare registers are preloadable (available with compare load feature)
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Programmable operation during debug mode
- Inputs may act as fault inputs
- Programmable input filter
- Counting start can be synchronized across counters

6.3 Modes of Operation

The TMR module design operates in only a single mode of operation: Functional Mode. The various counting modes are detailed in the Functional Description.

6.4 Block Diagram

Each of the timer/counter groups within the quad-timer are shown in this figure.

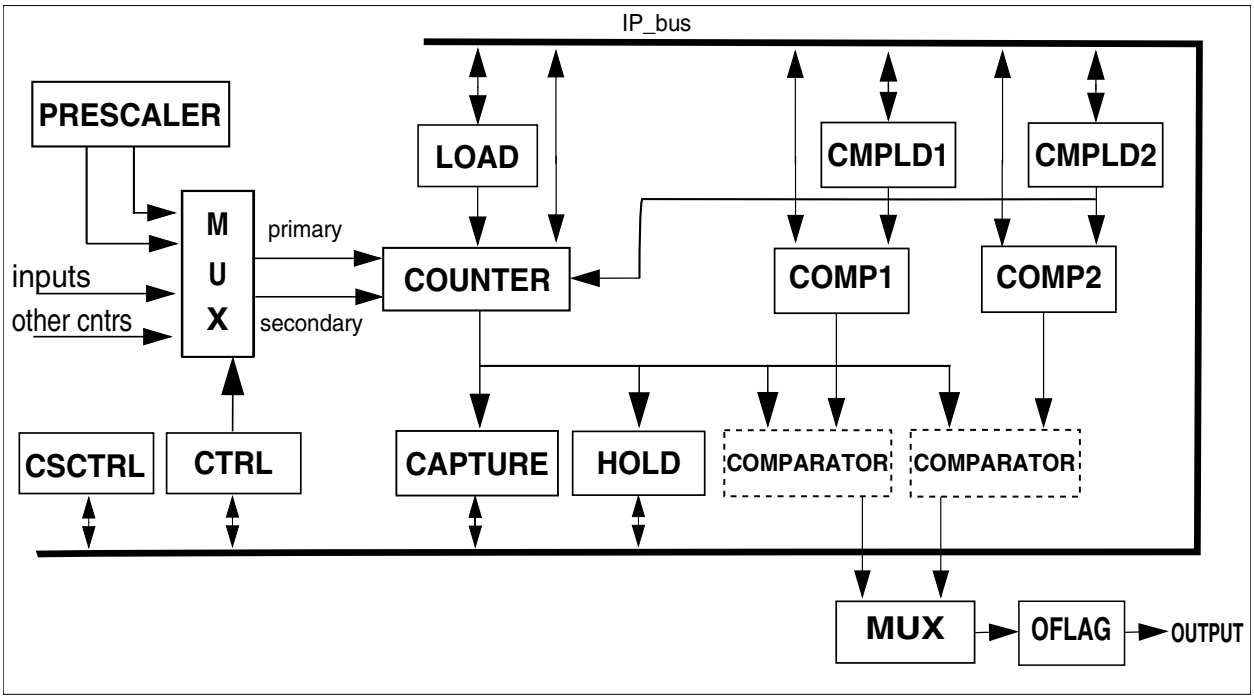


Figure 6-1. Quad Timer Block Diagram

6.5 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level and the address offset is defined at the module level. Make certain to check which quad timer is available on the chip being used, and which timer channels have external I/O.

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TMRA0_COMP1	R W	COMPARISON_1															
1	TMRA0_COMP2	R W	COMPARISON_2															
2	TMRA0_CAPT	R W	CAPTURE															
3	TMRA0_LOAD	R W	LOAD															

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	TMRA0_HOLD	R	HOLD															
		W																
5	TMRA0_CNTR	R	COUNTER															
		W																
6	TMRA0_CTRL	R	CM			PCS				SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
		W																
7	TMRA0_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
8	TMRA0_CMPLD1	R	COMPARATOR_LOAD_1															
		W																
9	TMRA0_CMPLD2	R	COMPARATOR_LOAD_2															
		W																
A	TMRA0_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRA0_FILT	R	0					FILT_CNT			FILT_PER							
		W																
F	TMRA0_ENBL	R	0													ENBL		
		W																
0	TMRA1_COMP1	R	COMPARISON_1															
		W																
1	TMRA1_COMP2	R	COMPARISON_2															
		W																
2	TMRA1_CAPT	R	CAPTURE															
		W																
3	TMRA1_LOAD	R	LOAD															
		W																
4	TMRA1_HOLD	R	HOLD															
		W																
5	TMRA1_CNTR	R	COUNTER															
		W																
6	TMRA1_CTRL	R	CM			PCS				SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7	TMRA1_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
8	TMRA1_CMPLD1	R/W	COMPARATOR_LOAD_1															
9	TMRA1_CMPLD2	R/W	COMPARATOR_LOAD_2															
A	TMRA1_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRA1_FILT	R	0					FILT_CNT			FILT_PER							
		W																
F	Reserved	R	0												RSVD			
		W																
0	TMRA2_COMP1	R/W	COMPARISON_1															
1	TMRA2_COMP2	R/W	COMPARISON_2															
2	TMRA2_CAPT	R/W	CAPTURE															
3	TMRA2_LOAD	R/W	LOAD															
4	TMRA2_HOLD	R/W	HOLD															
5	TMRA2_CNTR	R/W	COUNTER															
6	TMRA2_CTRL	R	CM			PCS				SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
		W																
7	TMRA2_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
8	TMRA2_CMPLD1	R/W	COMPARATOR_LOAD_1															
9	TMRA2_CMPLD2	R/W	COMPARATOR_LOAD_2															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	TMRA2_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRA2_FILT	R	0					FILT_CNT			FILT_PER							
		W																
F	Reserved	R	0												RSVD			
		W																
0	TMRA3_COMP1	R	COMPARISON_1															
		W																
1	TMRA3_COMP2	R	COMPARISON_2															
		W																
2	TMRA3_CAPT	R	CAPTURE															
		W																
3	TMRA3_LOAD	R	LOAD															
		W																
4	TMRA3_HOLD	R	HOLD															
		W																
5	TMRA3_CNTR	R	COUNTER															
		W																
6	TMRA3_CTRL	R	CM				PCS			SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
		W																
7	TMRA3_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
8	TMRA3_CMPLD1	R	COMPARATOR_LOAD_1															
		W																
9	TMRA3_CMPLD2	R	COMPARATOR_LOAD_2															
		W																
A	TMRA3_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRA3_FILT	R	0					FILT_CNT			FILT_PER							
		W																
F	Reserved	R	0												RSVD			
		W																
0	TMRB0_COMP1	R	COMPARISON_1															
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	TMRB0_COMP2	R	COMPARISON_2															
		W																
2	TMRB0_CAPT	R	CAPTURE															
		W																
3	TMRB0_LOAD	R	LOAD															
		W																
4	TMRB0_HOLD	R	HOLD															
		W																
5	TMRB0_CNTR	R	COUNTER															
		W																
6	TMRB0_CTRL	R	CM				PCS				SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE	
		W																
7	TMRB0_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
8	TMRB0_CMPLD1	R	COMPARATOR_LOAD_1															
		W																
9	TMRB0_CMPLD2	R	COMPARATOR_LOAD_2															
		W																
A	TMRB0_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRB0_FILT	R	0					FILT_CNT			FILT_PER							
		W																
F	TMRB0_ENBL	R	0													ENBL		
		W																
0	TMRB1_COMP1	R	COMPARISON_1															
		W																
1	TMRB1_COMP2	R	COMPARISON_2															
		W																
2	TMRB1_CAPT	R	CAPTURE															
		W																
3	TMRB1_LOAD	R	LOAD															
		W																
4	TMRB1_HOLD	R	HOLD															
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

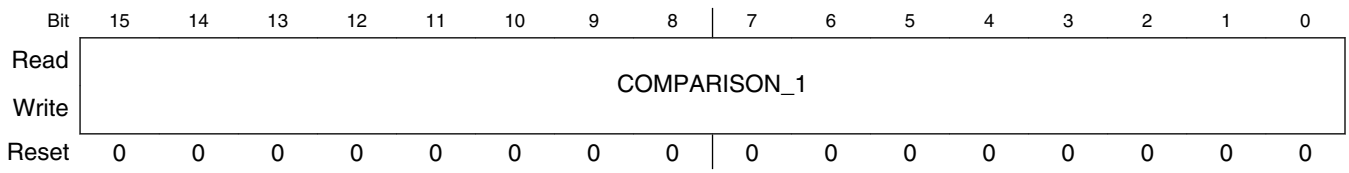
Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
5	TMRB1_CNTR	R	COUNTER															
		W																
6	TMRB1_CTRL	R	CM			PCS				SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
		W																
7	TMRB1_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
8	TMRB1_CMPLD1	R	COMPARATOR_LOAD_1															
		W																
9	TMRB1_CMPLD2	R	COMPARATOR_LOAD_2															
		W																
A	TMRB1_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRB1_FILT	R	0						FILT_CNT		FILT_PER							
		W																
F	Reserved	R	0												RSVD			
		W																
0	TMRB2_COMP1	R	COMPARISON_1															
		W																
1	TMRB2_COMP2	R	COMPARISON_2															
		W																
2	TMRB2_CAPT	R	CAPTURE															
		W																
3	TMRB2_LOAD	R	LOAD															
		W																
4	TMRB2_HOLD	R	HOLD															
		W																
5	TMRB2_CNTR	R	COUNTER															
		W																
6	TMRB2_CTRL	R	CM			PCS				SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
		W																
7	TMRB2_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8	TMRB2_CMPLD1	R	COMPARATOR_LOAD_1															
		W																
9	TMRB2_CMPLD2	R	COMPARATOR_LOAD_2															
		W																
A	TMRB2_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRB2_FILT	R	0					FILT_CNT			FILT_PER							
		W																
F	Reserved	R	0												RSVD			
		W																
0	TMRB3_COMP1	R	COMPARISON_1															
		W																
1	TMRB3_COMP2	R	COMPARISON_2															
		W																
2	TMRB3_CAPT	R	CAPTURE															
		W																
3	TMRB3_LOAD	R	LOAD															
		W																
4	TMRB3_HOLD	R	HOLD															
		W																
5	TMRB3_CNTR	R	COUNTER															
		W																
6	TMRB3_CTRL	R	CM				PCS			SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
		W																
7	TMRB3_SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
8	TMRB3_CMPLD1	R	COMPARATOR_LOAD_1															
		W																
9	TMRB3_CMPLD2	R	COMPARATOR_LOAD_2															
		W																
A	TMRB3_CSCTRL	R	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
		W																
B	TMRB3_FILT	R	0					FILT_CNT			FILT_PER							
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name
F	Reserved

6.5.1 Timer Channel Compare Register 1 (TMRx_COMP1)

Addresses: TMRA0_COMP1 – F000h base + 0h offset = F000h
TMRA1_COMP1 – F010h base + 0h offset = F010h
TMRA2_COMP1 – F020h base + 0h offset = F020h
TMRA3_COMP1 – F030h base + 0h offset = F030h
TMRB0_COMP1 – F040h base + 0h offset = F040h
TMRB1_COMP1 – F050h base + 0h offset = F050h
TMRB2_COMP1 – F060h base + 0h offset = F060h
TMRB3_COMP1 – F070h base + 0h offset = F070h



TMRx_COMP1 field descriptions

Field	Description
15–0 COMPARISON_1	Comparison Value 1 This read/write register stores the value used for comparison with the counter value.

6.5.2 Timer Channel Compare Register 2 (TMRx_COMP2)

Addresses: TMRA0_COMP2 – F000h base + 1h offset = F001h
TMRA1_COMP2 – F010h base + 1h offset = F011h
TMRA2_COMP2 – F020h base + 1h offset = F021h
TMRA3_COMP2 – F030h base + 1h offset = F031h
TMRB0_COMP2 – F040h base + 1h offset = F041h
TMRB1_COMP2 – F050h base + 1h offset = F051h
TMRB2_COMP2 – F060h base + 1h offset = F061h
TMRB3_COMP2 – F070h base + 1h offset = F071h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	COMPARISON_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_COMP2 field descriptions

Field	Description
15–0 COMPARISON_2	Comparison Value 2 This read/write register stores the value used for comparison with the counter value.

6.5.3 Timer Channel Capture Register (TMRx_CAPT)

Addresses: TMRA0_CAPT – F000h base + 2h offset = F002h
TMRA1_CAPT – F010h base + 2h offset = F012h
TMRA2_CAPT – F020h base + 2h offset = F022h
TMRA3_CAPT – F030h base + 2h offset = F032h
TMRB0_CAPT – F040h base + 2h offset = F042h
TMRB1_CAPT – F050h base + 2h offset = F052h
TMRB2_CAPT – F060h base + 2h offset = F062h
TMRB3_CAPT – F070h base + 2h offset = F072h

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

TMRx_CAPT field descriptions

Field	Description
15–0 CAPTURE	Capture Value This read/write register stores the value captured from the counter.

6.5.4 Timer Channel Load Register (TMRx_LOAD)

Addresses: TMRA0_LOAD – F000h base + 3h offset = F003h
TMRA1_LOAD – F010h base + 3h offset = F013h
TMRA2_LOAD – F020h base + 3h offset = F023h
TMRA3_LOAD – F030h base + 3h offset = F033h
TMRB0_LOAD – F040h base + 3h offset = F043h
TMRB1_LOAD – F050h base + 3h offset = F053h
TMRB2_LOAD – F060h base + 3h offset = F063h
TMRB3_LOAD – F070h base + 3h offset = F073h

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

TMRx_LOAD field descriptions

Field	Description
15–0 LOAD	Timer Load Register This read/write register stores the value used to initialize the counter.

6.5.5 Timer Channel Hold Register (TMRx_HOLD)

Addresses: TMRA0_HOLD – F000h base + 4h offset = F004h
TMRA1_HOLD – F010h base + 4h offset = F014h
TMRA2_HOLD – F020h base + 4h offset = F024h
TMRA3_HOLD – F030h base + 4h offset = F034h
TMRB0_HOLD – F040h base + 4h offset = F044h
TMRB1_HOLD – F050h base + 4h offset = F054h
TMRB2_HOLD – F060h base + 4h offset = F064h
TMRB3_HOLD – F070h base + 4h offset = F074h

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

TMRx_HOLD field descriptions

Field	Description
15–0 HOLD	This read/write register stores the counter's values of specific channels whenever any of the four counters within a module is read.

6.5.6 Timer Channel Counter Register (TMRx_CNTR)

Addresses: TMRA0_CNTR – F000h base + 5h offset = F005h
TMRA1_CNTR – F010h base + 5h offset = F015h
TMRA2_CNTR – F020h base + 5h offset = F025h
TMRA3_CNTR – F030h base + 5h offset = F035h
TMRB0_CNTR – F040h base + 5h offset = F045h
TMRB1_CNTR – F050h base + 5h offset = F055h
TMRB2_CNTR – F060h base + 5h offset = F065h
TMRB3_CNTR – F070h base + 5h offset = F075h

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

TMRx_CNTR field descriptions

Field	Description
15–0 COUNTER	This read/write register is the counter for the corresponding channel in a timer module.

6.5.7 Timer Channel Control Register (TMRx_CTRL)

Addresses: TMRA0_CTRL – F000h base + 6h offset = F006h
TMRA1_CTRL – F010h base + 6h offset = F016h
TMRA2_CTRL – F020h base + 6h offset = F026h
TMRA3_CTRL – F030h base + 6h offset = F036h
TMRB0_CTRL – F040h base + 6h offset = F046h
TMRB1_CTRL – F050h base + 6h offset = F056h
TMRB2_CTRL – F060h base + 6h offset = F066h
TMRB3_CTRL – F070h base + 6h offset = F076h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	CM			PCS				SCS		ONCE	LENGTH	DIR	COINIT	OUTMODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_CTRL field descriptions

Field	Description
15–13 CM	Count Mode These bits control the basic counting and behavior of the counter. 000 No operation 001 Count rising edges of primary source 1 010 Count rising and falling edges of primary source 2 011 Count rising edges of primary source while secondary input high active 100 Quadrature count mode, uses primary and secondary sources 101 Count rising edges of primary source; secondary source specifies direction (1 = minus) 3 110 Edge of secondary source triggers primary count until compare 111 Cascaded counter mode (up/down) 4
12–9 PCS	Primary Count Source These bits select the primary count source. **NOTE:** A timer selecting its own output for input is not a legal choice. The result is no counting. 0000 Counter 0 input pin 0001 Counter 1 input pin

Table continues on the next page...

TMRx_CTRL field descriptions (continued)

Field	Description
	0010 Counter 2 input pin 0011 Counter 3 input pin 0100 Counter 0 output 0101 Counter 1 output 0110 Counter 2 output 0111 Counter 3 output 1000 IP bus clock divide by 1 prescaler 1001 IP bus clock divide by 2 prescaler 1010 IP bus clock divide by 4 prescaler 1011 IP bus clock divide by 8 prescaler 1100 IP bus clock divide by 16 prescaler 1101 IP bus clock divide by 32 prescaler 1110 IP bus clock divide by 64 prescaler 1111 IP bus clock divide by 128 prescaler
8–7 SCS	Secondary Count Source These bits identify the external input pin to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of CNTR . The selected input can also be used to specify the count direction. The selected signal can also be used as a fault input when CSCTRL[FAULT] is set. The polarity of the signal can be inverted by SCTRL[IPS]. 00 Counter 0 input pin 01 Counter 1 input pin 10 Counter 2 input pin 11 Counter 3 input pin
6 ONCE	Count Once This bit selects continuous or one shot counting mode. 0 Count repeatedly. 1 Count until compare and then stop. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, the counter re-initializes after reaching the COMP1 value, continues to count to the COMP2 value, and then stops.
5 LENGTH	Count Length This bit determines whether the counter: • counts to the compare value and then re-initializes itself to the value specified in the LOAD (or CMPLD2) register, or • continues counting past the compare value to the binary roll over. 0 Roll over. 1 Count until compare, then re-initialize. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until a COMP1 value is reached, re-initializes, counts until COMP2 value is reached, re-initializes, counts until COMP1 value is reached, and so on.
4 DIR	Count Direction

Table continues on the next page...

TMRx_CTRL field descriptions (continued)

Field	Description
	This bit selects either the normal count direction up, or the reverse direction, down. 0 Count up. 1 Count down.
3 COINIT	Co-Channel Initialization This bit enables another counter/timer within the module to force the re-initialization of this counter/timer when it has an active compare event. 0 Co-channel counter/timers cannot force a re-initialization of this counter/timer 1 Co-channel counter/timers may force a re-initialization of this counter/timer
2–0 OUTMODE	Output Mode These bits determine the mode of operation for the OFLAG output signal. 000 Asserted while counter is active 001 Clear OFLAG output on successful compare 010 Set OFLAG output on successful compare 011 Toggle OFLAG output on successful compare 100 Toggle OFLAG output using alternating compare registers 101 Set on compare, cleared on secondary source input edge 110 Set on compare, cleared on counter rollover 111 Enable gated clock output while counter is active

1. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1. If the primary count source is IP bus clock divide by 1, only rising edges are counted regardless of the value of SCTRL[IPS].
2. IP bus clock divide by 1 cannot be used as a primary count source in edge count mode.
3. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1.
4. The primary count source must be set to one of the counter outputs.

6.5.8 Timer Channel Status and Control Register (TMRx_SCTRL)

Addresses: TMRA0_SCTRL – F000h base + 7h offset = F007h
TMRA1_SCTRL – F010h base + 7h offset = F017h
TMRA2_SCTRL – F020h base + 7h offset = F027h
TMRA3_SCTRL – F030h base + 7h offset = F037h
TMRB0_SCTRL – F040h base + 7h offset = F047h
TMRB1_SCTRL – F050h base + 7h offset = F057h
TMRB2_SCTRL – F060h base + 7h offset = F067h
TMRB3_SCTRL – F070h base + 7h offset = F077h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
Write														FORCE		
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

TMRx_SCTRL field descriptions

Field	Description
15 TCF	Timer Compare Flag This bit is set when a successful compare occurs. This bit is cleared by writing a zero to this bit location.
14 TCFIE	Timer Compare Flag Interrupt Enable This bit (when set) enables interrupts when TCF is set.
13 TOF	Timer Overflow Flag This bit is set when the counter rolls over its maximum value $FFFF or $0000 (depending on count direction). This bit is cleared by writing a zero to this bit location.
12 TOFIE	Timer Overflow Flag Interrupt Enable This bit (when set) enables interrupts when TOF is set.
11 IEF	Input Edge Flag This bit is set when a positive input transition occurs (on an input selected as a secondary count source) while the count mode does not equal 000. This bit is cleared by writing a zero to this bit position. **NOTE:** Setting the input polarity select bit (IPS) enables the detection of negative input edge transitions. Also, the control register's secondary count source determines which external input pin is monitored by the detection circuitry.
10 IEFIE	Input Edge Flag Interrupt Enable This bit (when set) enables interrupts when IEF is set.
9 IPS	Input Polarity Select This bit (when set) inverts the input signal polarity.
8 INPUT	External Input Signal This read only bit reflects the current state of the external input pin selected via the secondary count source after application of IPS and filtering.
7–6 CAPTURE_ MODE	Input Capture Mode These bits specify the operation of the capture register as well as the operation of the input edge flag. The input source is the secondary count source. 00 Capture function is disabled 01 Load capture register on rising edge (when IPS=0) or falling edge (when IPS=1) of input 10 Load capture register on falling edge (when IPS=0) or rising edge (when IPS=1) of input 11 Load capture register on both edges of input
5 MSTR	Master Mode This bit (when set) enables the compare function's output to be broadcasted to the other counters/timers in the module. This signal then can be used to re-initialize the other counters and/or force their OFLAG signal outputs.
4 EEOF	Enable External OFLAG Force This bit (when set) enables the compare from another counter/timer within the same module to force the state of this counter's OFLAG output signal.
3 VAL	Forced OFLAG Value This bit determines the value of the OFLAG output signal when software triggers a FORCE command.

Table continues on the next page...

TMRx_SCTRL field descriptions (continued)

Field	Description
2 FORCE	Force OFLAG Output This write only bit forces the current value of VAL to be written to the OFLAG output. This bit always reads as a zero. VAL and FORCE can be written simultaneously in a single write operation. Write to FORCE only if the counter is disabled. Setting this bit while the counter is enabled may yield unpredictable results.
1 OPS	Output Polarity Select This bit determines the polarity of the OFLAG output signal. 0 True polarity. 1 Inverted polarity.
0 OEN	Output Enable This bit determines the direction of the external pin. 0 The external pin is configured as an input. 1 The OFLAG output signal is driven on the external pin. Other timer groups using this external pin as their input see the driven value. The polarity of the signal is determined by OPS.

6.5.9 Timer Channel Comparator Load Register 1 (TMRx_CMPLD1)

Addresses: TMRA0_CMPLD1 – F000h base + 8h offset = F008h
TMRA1_CMPLD1 – F010h base + 8h offset = F018h
TMRA2_CMPLD1 – F020h base + 8h offset = F028h
TMRA3_CMPLD1 – F030h base + 8h offset = F038h
TMRB0_CMPLD1 – F040h base + 8h offset = F048h
TMRB1_CMPLD1 – F050h base + 8h offset = F058h
TMRB2_CMPLD1 – F060h base + 8h offset = F068h
TMRB3_CMPLD1 – F070h base + 8h offset = F078h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARATOR_LOAD_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_CMPLD1 field descriptions

Field	Description
15–0 COMPARATOR_ LOAD_1	This read/write register is the comparator 1 preload value for the COMP1 register for the corresponding channel in a timer module.

6.5.10 Timer Channel Comparator Load Register 2 (TMRx_CMPLD2)

Addresses: TMRA0_CMPLD2 – F000h base + 9h offset = F009h
TMRA1_CMPLD2 – F010h base + 9h offset = F019h
TMRA2_CMPLD2 – F020h base + 9h offset = F029h
TMRA3_CMPLD2 – F030h base + 9h offset = F039h
TMRB0_CMPLD2 – F040h base + 9h offset = F049h
TMRB1_CMPLD2 – F050h base + 9h offset = F059h
TMRB2_CMPLD2 – F060h base + 9h offset = F069h
TMRB3_CMPLD2 – F070h base + 9h offset = F079h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARATOR_LOAD_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_CMPLD2 field descriptions

Field	Description
15–0 COMPARATOR_ LOAD_2	This read/write register is the comparator 2 preload value for the COMP2 register for the corresponding channel in a timer module.

6.5.11 Timer Channel Comparator Status and Control Register (TMRx_CSCTRL)

Addresses: TMRA0_CSCTRL – F000h base + Ah offset = F00Ah
TMRA1_CSCTRL – F010h base + Ah offset = F01Ah
TMRA2_CSCTRL – F020h base + Ah offset = F02Ah
TMRA3_CSCTRL – F030h base + Ah offset = F03Ah
TMRB0_CSCTRL – F040h base + Ah offset = F04Ah
TMRB1_CSCTRL – F050h base + Ah offset = F05Ah
TMRB2_CSCTRL – F060h base + Ah offset = F06Ah
TMRB3_CSCTRL – F070h base + Ah offset = F07Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_CSCTRL field descriptions

Field	Description
15–14 DBG_EN	Debug Actions Enable These bits allow the TMR module to perform certain actions in response to the chip entering debug mode. 00 Continue with normal operation during debug mode. (default) 01 Halt TMR counter during debug mode. 10 Force TMR output to logic 0 (prior to consideration of SCTRL[OPS]). 11 Both halt counter and force output to 0 during debug mode.
13 FAULT	Fault Enable The selected secondary input acts as a fault signal so that the timer OFLAG is cleared when the secondary input is set. When the secondary input is used in this mode, there is no resynchronization of the input so that there is a combinational path to clear the OFLAG. Fault inputs less than two clock periods wide will not be latched. Latched faults will be cleared the next time that the counter logic sets the OFLAG. 0 Fault function disabled. 1 Fault function enabled.
12 ALT_LOAD	Alternative Load Enable This bit allows for an alternative method for loading the counter during modulo counting. Normally, the counter can be loaded only with the value from the LOAD register. When this bit is set, the counter is loaded from the LOAD register when counting up and a match with COMP1 occurs, or the counter is loaded from the CMPLD2 register when counting down and a match with COMP2 occurs. 0 Counter can be re-initialized only with the LOAD register. 1 Counter can be re-initialized with the LOAD or CMPLD2 registers depending on count direction.
11 ROC	Reload on Capture This bit enables the capture function to cause the counter to be reloaded from the LOAD register. 0 Do not reload the counter on a capture event. 1 Reload the counter on a capture event.
10 TCI	Triggered Count Initialization Control This bit is used during triggered count mode, CTRL[CM] = 110, to enable the counter to be re-initialized when a second trigger occurs while the counter is still counting. Normally, the second trigger causes the counting to stop/pause until a third trigger occurs. With this bit set, a second trigger event causes the counter to re-initialize and continue counting. 0 Stop counter upon receiving a second trigger event while still counting from the first trigger event. 1 Reload the counter upon receiving a second trigger event while still counting from the first trigger event.
9 UP	Counting Direction Indicator This read-only bit is used during quadrature count mode, CTRL[CM] = 100, to read the direction of the last count. CTRL[DIR] reverses the sense of this bit. 0 The last count was in the DOWN direction. 1 The last count was in the UP direction.
8 Reserved	This read-only bit is reserved and always has the value zero.

Table continues on the next page...

TMRx_CSCTRL field descriptions (continued)

Field	Description
7 TCF2EN	Timer Compare 2 Interrupt Enable An interrupt is issued when both this bit and TCF2 are set.
6 TCF1EN	Timer Compare 1 Interrupt Enable An interrupt is issued when both this bit and TCF1 are set.
5 TCF2	Timer Compare 2 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP2 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
4 TCF1	Timer Compare 1 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP1 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
3–2 CL2	Compare Load Control 2 These bits control when COMP2 is preloaded with the value from CMPLD2. 00 Never preload 01 Load upon successful compare with the value in COMP1 10 Load upon successful compare with the value in COMP2 11 Reserved
1–0 CL1	Compare Load Control 1 These bits control when COMP1 is preloaded with the value from CMPLD1. 00 Never preload 01 Load upon successful compare with the value in COMP1 10 Load upon successful compare with the value in COMP2 11 Reserved

6.5.12 Timer Channel Input Filter Register (TMRx_FILT)

Input filter considerations:

- Set the FILT_PER value such that the sampling period is larger than the period of the expected noise. In this way, a noise spike will corrupt only one sample. Choose the FILT_CNT value to reduce the probability that noisy samples cause an incorrect transition to be recognized. The probability of an incorrect transistion is defined as the probability of an incorrect sample raised to the power of (FILT_CNT + 3).
- The values of FILT_PER and FILT_CNT must also be balanced against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of (((FILT_CNT + 3) x FILT_PER) + 2) IP bus clock periods.

Addresses: TMRA0_FILT – F000h base + Bh offset = F00Bh
TMRA1_FILT – F010h base + Bh offset = F01Bh
TMRA2_FILT – F020h base + Bh offset = F02Bh
TMRA3_FILT – F030h base + Bh offset = F03Bh
TMRB0_FILT – F040h base + Bh offset = F04Bh
TMRB1_FILT – F050h base + Bh offset = F05Bh
TMRB2_FILT – F060h base + Bh offset = F06Bh
TMRB3_FILT – F070h base + Bh offset = F07Bh

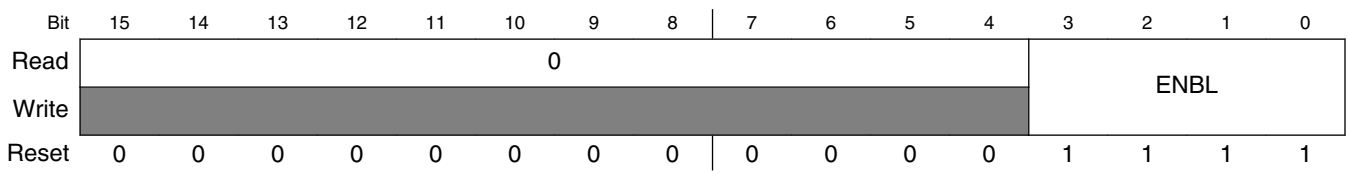
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					FILT_CNT			FILT_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_FILT field descriptions

Field	Description
15–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10–8 FILT_CNT	Input Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency.
7–0 FILT_PER	Input Filter Sample Period These bits represent the sampling period (in IP bus clock cycles) of the TMR input signals. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter.

6.5.13 Timer Channel Enable Register (TMRx_ENBL)

Addresses: TMRA0_ENBL – F000h base + Fh offset = F00Fh
TMRB0_ENBL – F040h base + Fh offset = F04Fh



TMRx_ENBL field descriptions

Field	Description
15–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3–0 ENBL	Timer Channel Enable These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate counters. If an ENBL bit is set, then the corresponding channel starts its counter as soon as the CTRL[CM] field has a value other than 0. When an ENBL bit is clear, the corresponding counter maintains its current value. 0 Timer channel is disabled. 1 Timer channel is enabled. (default)

6.6 Functional Description

6.6.1 General

The counter/timer has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.
- The counter can decode and count quadrature encoded input signals.

- The counter can count up and down using dual inputs in a "count with direction" format.
- The counter's terminal count value (modulo) is programmable.
  - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count "rolls over" to zero.

The external inputs to each counter/timer are shareable among each of the four counter/timers within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be "captured"
- They can be used to generate interrupt requests

The polarity of the external inputs are selectable.

The primary output of each timer/counter is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is selectable.

Any counter/timer can be assigned as a master. A master's compare signal can be broadcast to the other counter/timers within the module. The other counters can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a master's counter/timer compare event occurs.

6.6.2 Functional Modes

The selected external count signals are sampled at the TMR's base clock rate and then run through a transition detector. The maximum count rate is one-half of the TMR's base clock rate. Internal clock sources can be used to clock the counters at the TMR's base clock rate.

If a counter is programmed to count to a specific value and then stop, the CTRL[CM] field is cleared when the count terminates.

6.6.2.1 Stop Mode

If CTRL[CM] is set to '000', the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

6.6.2.2 Count Mode

If CTRL[CM] is set to '001', the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as "widgets" on a conveyor belt passing a sensor. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the negative edge of the selected external input signal is counted.

Example: 6.6.2.2.1 Count Pulses from External Source

```
//        (See Processor Expert PulseAccumulator bean.)
//    This example uses TMRA1 to count pulse (actually counts rising edges of the pulse)
//      from an external source (TA3).
//
void Pulse_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x0600);           /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x00);
  setReg(TMRA1_CNTR,0x00);             /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMRA1_CTRL,CM,0x01);  /* Run counter */
}
```

Example: 6.6.2.2.2 Generate Periodic Interrupt By Counting Internal Clocks

```
//         (See Processor Expert TimerInt bean.)
//   This example generates an interrupt every 100ms,
//     assuming the chip is operating at 60 MHz.
//
// It does this by using the IP_bus_clk divided by 128 as the counter clock source.
// The counter then counts to 46874 where it matches the COMP1 value.
// At that time an interrupt is generated, the counter is reloaded and
// the next COMP1 value is loaded from CMPLD1.
//
void TimerInt_Init(void)
{
  /* TMRA0_CTRL: CM=0,PCS=0,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA0_CTRL,0x20);              /* Stop all functions of the timer */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA0_SCTRL,0x00);
  setReg(TMRA0_LOAD,0x00);             /* Reset load register */
  setReg(TMRA0_COMP1,46874);           /* Set up compare 1 register */
  setReg(TMRA0_CMPLD1,46874);          /* Also set the compare preload register */
  /* TMRA0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,TCF2EN=0,TCF1EN=1,
                   TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMRA0_CSCTRL,0x41);           /* Enable compare 1 interrupt and */
                                       /*   compare 1 preload */
  setRegBitGroup(TMRA0_CTRL,PCS,0xF); /* Primary Count Source to IP_bus_clk / 128 */
  setReg(TMRA0_CNTR,0x00);             /* Reset counter register */
  setRegBitGroup(TMRA0_CTRL,CM,0x01); /* Run counter */
}
```

6.6.2.3 Edge-Count Mode

If CTRL[CM] is set to '010', the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment, such as a simple encoder wheel.

Example: 6.6.2.3.1 Count Both Edges of External Source Signal

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMRA1 to count pulse (actually counts rising edges of the pulse)
//    from an external source (TA3).
//
void Pulse_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x0600);           /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x00);
  setReg(TMRA1_CNTR,0x00);             /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMRA1_CTRL,CM,0x02);  /* Run counter */
}
```

6.6.2.4 Gated-Count Mode

If CTRL[CM] is set to '011', the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the counter will count while the selected secondary input is low.

Example: 6.6.2.4.1 Capture Duration of External Pulse

```
//       (See Processor Expert PulseAccumulator bean.)
//   This example uses TMRA1 to determine the duration of an external pulse.
//
//   The IP_bus clock is used as the primary counter. If the duration of the
//     external pulse is longer than 0.001 seconds one of the other IP_bus clock
//     dividers can be used. If the pulse duration is longer than 0.128 seconds
//     an external clock source will have to be used as the primary clock source.
//
void Pulse1_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=8,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x1080);            /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x00);
  setReg(TMRA1_CNTR,0x00);             /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMRA1_CTRL,CM,0x03);  /* Run counter */
}
```

6.6.2.5 Quadrature-Count Mode

If CTRL[CM] is set to '100', the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information.

This figure shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

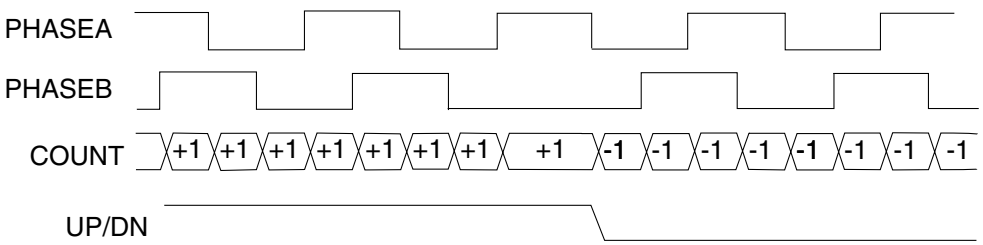


Figure 6-119. Quadrature Incremental Position Encoder

Example: 6.6.2.5.1 Quadrature Count Mode Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMRA0 for counting states of a quadrature position encoder.
//
//  Timer input 0 is used as the primary count source (PHASEA).
//  Timer input 1 is used as the secondary count source (PHASEB).
//
void Pulse_Init(void)
{
  /* TMRA0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRC0_CTRL,0x80);              /* Set up mode */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA0_SCTRL,0x00);
  setReg(TMRA0_CNTR,0x00);             /* Reset counter register */
  setReg(TMRA0_LOAD,0x00);             /* Reset load register */
  setReg(TMRA0_COMP1,0xFFFF);           /* Set up compare 1 register */
  setReg(TMRA0_COMP2,0x00);             /* Set up compare 2 register */
  /* TMRA0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA0_CSCTRL,0x00);
  setRegBitGroup(TMRA0_CTRL,CM,0x04);  /* Run counter */
}
```

6.6.2.6 Quadrature-Count Mode with Index Input

As an extension to the quadrature count mode discussed in the previous paragraph, some rotary shafts have a HOME or INDEX indicator. This would be a third input to the timer that is used to reset the timer's counter.

In this example, channel 0 is used to decode the quadrature inputs, but it doesn't actually count. Because its upper and lower limits are both set to 0, its output is cascaded count up and count down signals each time the quadrature inputs indicate a change in count. Channel 1 works in cascaded count mode receiving its counting instructions from channel 0. When an input capture event occurs, channel 1 is programmed to reset its counter value. The channel 1 counter contains the position value for the shaft.

Example: 6.6.2.6.1 Quadrature Count Mode with Index Input Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMRA0and TMRA1 for counting states of a quadrature position encoder.
//
// Timer input 0 is used as the primary count source (PHASEA).
// Timer input 1 is used as the secondary count source (PHASEB).
// Timer input 2 is used as the index input source (INDEX).
//
void Pulse_Init(void)
{
  /* TMRA0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA0_CTRL,0xA0);              /* Set up mode */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA0_SCTRL,0x00);
  setReg(TMRA0_CNTR,0x00);             /* Reset counter register */
  setReg(TMRA0_LOAD,0x00);             /* Reset load register */
  setReg(TMRA0_COMP1,0x00);           /* Set up compare 1 register */
  setReg(TMRA0_COMP2,0x00);            /* Set up compare 2 register */
  /* TMRA0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA0_CSCTRL,0x00);
/* TMRA1_CTRL: CM=7,PCS=100,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0xEA00);           /* Set up capture edge */
/* TMRA1_SCTRL: Capture_Mode=10 */
  setReg(TMRA1_SCTRL,0x0080);
/* TMRA1_CSTRL: ROC=1 */
  setReg(TMRA1_CTRL,0x0800);           /* Set up reload on capture */
setRegBitGroup(TMRA0_CTRL,CM,0x04);  /* Run counter */
}
```

6.6.2.7 Signed-Count Mode

If CTRL[CM] is set to '101', the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

Example: 6.6.2.7.1 Signed Count Mode Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMRA0 for signed mode counting.
//
//  Timer input 2 is used as the primary count source.
//  Timer input 1 is used to determine the count direction.
//
void Pulse_Init(void)
{
  /* TMRA0_CTRL: CM=0,PCS=2,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA0_CTRL,0x0480);              /* Set up mode */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA0_SCTRL,0x1000);
  setReg(TMRA0_CNTR,0x00);              /* Reset counter register */
  setReg(TMRA0_LOAD,0x00);              /* Reset load register */
  setRegBitGroup(TMRA0_CTRL,CM,0x05);   /* Run counter */
}
```

6.6.2.8 Triggered-Count Mode 1

If CSCTRL[TCI] is clear and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count is reached, counting will stop and SCTRL[TCF] (timer compare flag) will be set. Subsequent secondary input transitions will continue to restart and stop the counting until a compare event occurs.

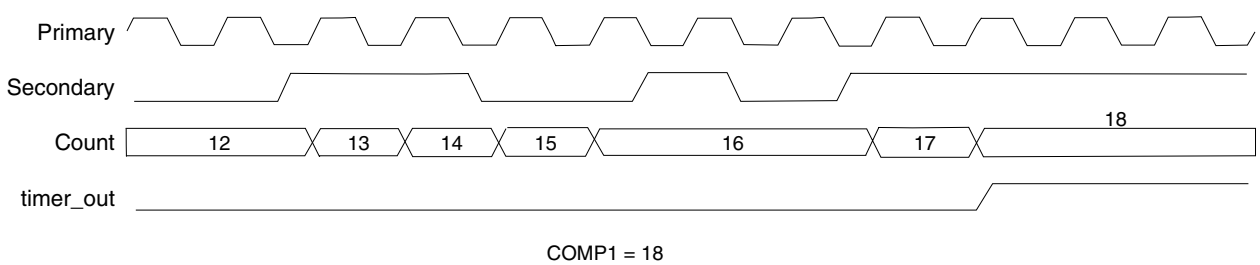


Figure 6-120. Triggered Count Mode 1 (CTRL[LENGTH]=0)

Example: 6.6.2.8.1 Triggered Count Mode 1 Example

```
//        (See Processor Expert PulseAccumulator bean.)
//   This example uses TMRA1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x0700);             /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x1000);
  setReg(TMRA1_CNTR,0x00);               /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);               /* Reset load register */
  setReg(TMRA1_COMP1,0x0012);             /* Set up compare 1 register */
  /* TMRA1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA1_CSCTRL,0x00);
  setRegBitGroup(TMRA1_CTRL,CM,0x06);    /* Run counter */
}
```

6.6.2.9 Triggered-Count Mode 2

If CSCTRL[TCI] is set and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count was reached, the counter will reload and continue counting. When CSCTRL[TCI] is set, the OFLAG output mode, CTRL[OUTMODE], should probably be set to '101' (cleared on init, set on compare) to ensure the output will be in a known state after the second input transition and subsequent reload takes place.

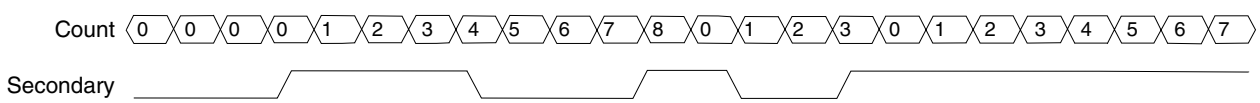


Figure 6-121. Triggered Count Mode 2 (CTRL[LENGTH]=0)

Example: 6.6.2.9.1 Triggered Count Mode 2Example

```
//        (See Processor Expert PulseAccumulator bean.)
//   This example uses TMRA1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x0700);            /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x1000);

  setReg(TMRA1_CNTR,0x00);              /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);              /* Reset load register */
  setReg(TMRA1_COMP1,0x0012);            /* Set up compare 1 register */
  /* TMRA1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA1_CSCTRL,0x00);
  setRegBitGroup(TMRA1_CTRL,CM,0x06); /* Run counter */
}
```

6.6.2.10 One-Shot Mode

If CTRL[CM] is set to '110', and the counter is set to reinitialize at a compare event (CTRL[LENGTH]=1),and CTRL[OUTMODE] is set to '101' (cleared on init, set on compare), the counter works in a one-shot mode. An external event causes the counter to count, and when the terminal count is reached, the output is asserted. This delayed output can be used to provide timing delays.

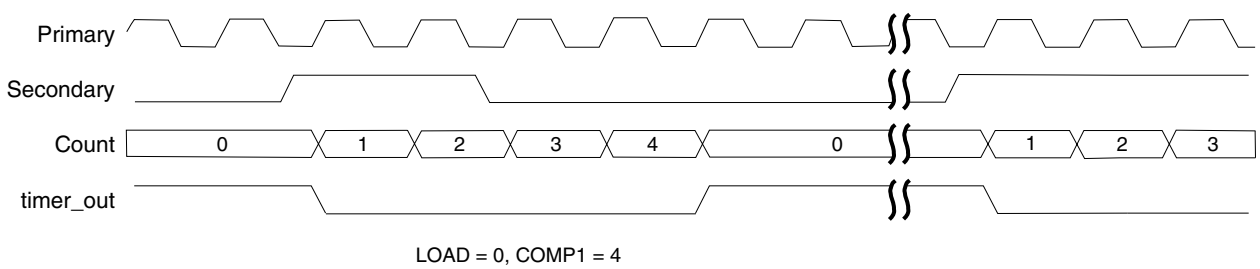


Figure 6-122. One-Shot Mode (CTRL[LENGTH]=1)

Example: 6.6.2.10.1 One-Shot Mode Example

```
//   (See Processor Expert PulseAccumulator bean.)
//  This example uses TMRA1 for one-shot mode counting.
//
//  Timer input 3 is used as the primary count source.
//  Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=5 */
  setReg(TMRA1_CTRL,0x0725);            /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
              Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x1000);
  setReg(TMRA1_CNTR,0x00);              /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);              /* Reset load register */
  setReg(TMRA1_COMP1,0x0004);            /* Set up compare 1 register */
  /* TMRA1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA1_CSCTRL,0x00);
  setRegBitGroup(TMRA1_CTRL,CM,0x06);  /* Run counter */
}
```

6.6.2.11 Cascade-Count Mode

If CTRL[CM] is set to '111', the counter's input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This cascade or daisy-chained mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high-speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to four counters may be cascaded to create a 64-bit wide synchronous counter. Check the data sheet to see if there are any frequency limits for cascaded counting mode.

Whenever any counter is read within a counter module, all of the counters' values within the module are captured in their respective hold registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read the hold registers of the other counters in the chain. The cascaded counter mode is synchronous.

Note

It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a ripple mode, where higher order counters will transition a clock later than a purely synchronous design.

Example: 6.6.2.11.1 Generate Periodic Interrupt Cascading Two Counters

```
//    (See Processor Expert TimerInt bean.)
// This example generates an interrupt every 30 seconds,
//    assuming the chip is operating at 60 MHz.
//
// To do this, counter 2 is used to count 60,000 IP_bus clocks, which means it
//    will compare and reload every 0.001 seconds.
// Counter 3 is cascaded and used to count the 0.001 second ticks and
//    generate the desired interrupt interval.
//
void TimerInt_Init(void)
{
// Set counter 2 to count IP_bus clocks
   /* TMRA2_CTRL: CM=0,PCS=8,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
   setReg(TMRA2_CTRL,0x1020);           /* Stop all functions of the timer */
// Set counter 3 as cascaded and to count counter 2 outputs
  /* TMRA3_CTRL: CM=7,PCS=6,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA3_CTRL,0xEC20);            /* Set up cascade counter mode */
  /* TMRA3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA3_SCTRL,0x00);
  /* TMRA2_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA2_SCTRL,0x00);
  setReg(TMRA3_CNTR,0x00);              /* Reset counter register */
  setReg(TMRA2_CNTR,0x00);
  setReg(TMRA3_LOAD,0x00);              /* Reset load register */
  setReg(TMRA2_LOAD,0x00);
  setReg(TMRA3_COMP1, 30000);           /* milliseconds in 30 seconds */
  setReg(TMRA3_CMPLD1,30000);
  setReg(TMRA2_COMP1, 60000);           // Set to cycle every milisecond
  setReg(TMRA2_CMPLD1,60000);
  /* TMRA3_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMRA3_CSCTRL,0x41);            /* Enable compare 1 interrupt and */
                                        /* compare 1 preload              */
  /* TMRA2_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMRA2_CSCTRL,0x01);            /* Enable Compare 1 preload */
  setRegBitGroup(TMRA2_CTRL,CM,0x01);  /* Run counter */
}
```

6.6.2.12 Pulse-Output Mode

If CTRL[CM]=001, and CTRL[OUTMODE] is set to 111' (gated clock output), and CTRL[ONCE] is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

Note

This does not work if CTRL[PCS] is set to 1000 (IP_bus/1).

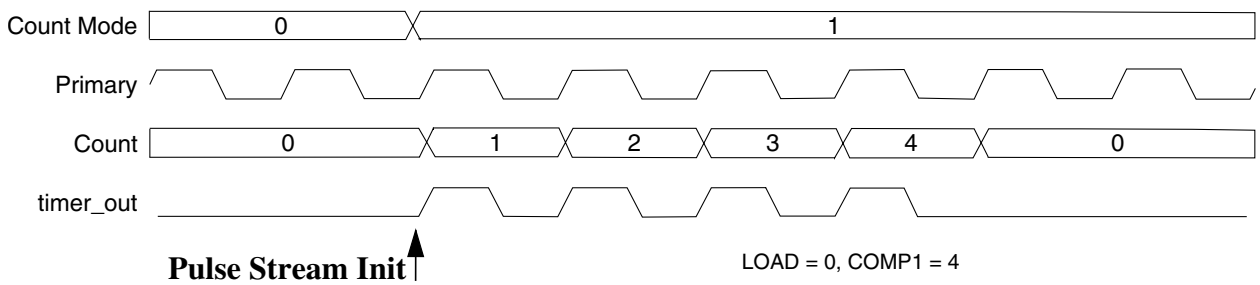


Figure 6-123. Pulse Output Mode

Example: 6.6.2.12.1 Pulse Outputs Using Two Counters

```
//     (See Processor Expert PulseStream bean.)
// This example generates six 10ms pulses, from TA1 output.
// Assuming the chip is operating at 60 MHz.
//
// To do this, timer 3 is used to generate a clock with a period of 10ms.
//
// Timer 1 is used to gate these clocks and count the number of pulses that have
//  been generated.
//
void PulseStream_Init(void)
{
// Select IP_bus_clk/16 as the clock source for Timer A3
  /* TMRA3_CTRL: CM=0,PCS=0x0C,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=3 */
   setReg(TMRA3_CTRL,0x1823);              /* Set up mode */
  /* TMRA3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
   setReg(TMRA3_SCTRL,0x00);
   setReg(TMRA3_LOAD,0x00);               /* Reset load register */
   setReg(TMRA3_COMP1,37500);             /* (16 * 37500 ) / 60e6 = 0.01 sec */
 /* TMRA3_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA3_CSCTRL,0x00);              /* Set up comparator control register */



// Timer 3 output is the clock source for this timer.
  /* TMRA1_CTRL: CM=0,PCS=7,SCS=0,ONCE=1,LENGTH=1,DIR=0,Co_INIT=0,OM=7 */
  setReg(TMRA1_CTRL,0x0E67);              /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
   setReg(TMRA1_SCTRL,0x01);
   setReg(TMRA1_CNTR,0x00);              /* Reset counter register */
   setReg(TMRA1_LOAD,0x00);              /* Reset load register */
   setReg(TMRA1_COMP1,0x04);              /* Set up compare 1 register */

// set to interrupt after the last pulse
  /* TMRA1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
                   TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMRA1_CSCTRL,0x40);              /* Set up comparator control register */
// Finally, start the counters running
  setReg(TMRA3_CNTR,0);                   /* Reset counter */
  setRegBitGroup(TMRA3_CTRL,CM,0x01);     /* Run source clock counter */
  setRegBitGroup(TMRA1_CTRL,CM,0x01);     /* Run counter */
}
```

6.6.2.13 Fixed-Frequency PWM Mode

If CTRL[CM]=001, count through roll-over (CTRL[LENGTH]=0), continuous count (CTRL[ONCE]=0) and CTRL[OUTMODE] is '110' (set on compare, cleared on counter roll-over), then the counter output yields a pulse-width modulated (PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse-width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

Example: 6.6.2.13.1 Fixed-Frequency PWM Mode Example

```
//    (See Processor Expert PWM bean.)
// This example uses TMRA0 for Fixed-Frequency PWM mode timing.
//
// The timer will count IP_bus clocks continuously until it rolls over.
// This results in a PWM period of 65536 / 60e6 = 1092.267 usec
//
// Initially, an output pulse width of 25 usec is generated ( 1500 / 60e6 )
//   giving a PWM ratio of 1500 / 65536 = 2.289%
// This pulse width can be changed by changing the COMP1 register value (using CMPLD1).
//
void PWM1_Init(void)
{
  setReg(TMRA0_CNTR,0);                 /* Reset counter */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                  Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
  setReg(TMRA0_SCTRL,0x05);              /* Enable output */
  setReg(TMRA0_COMP1,1500);          /* Store initial value to the duty-compare register */
  /* TMRA0_CTRL: CM=1,PCS=8,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=6 */
  setReg(TMRA0_CTRL,0x3006);       /* Run counter */
}
```

6.6.2.14 Variable-Frequency PWM Mode

If CTRL[CM]=001, count until compare (CTRL[LENGTH]=1), continuous count (CTRL[ONCE] = 0) and CTRL[OUTMODE] is '100' (toggle OFLAG and alternate compare registers), then the counter output yields a pulse-width modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

To set up the TMR to run in variable frequency PWM mode with compare preload please use the following setup for the specific counter you would like to use. When performing the setup, update the TMR_CTRL register last because the counter will start counting if the count mode is changed to any value other than 3'b000 (assuming the primary count source is already active).

Timer Control Register (CTRL)

- CM=3'b001 (count rising edges of primary source)
- PCS=4'b1000 (IP bus clock for best granularity for waveform timing)
- SCS=Any (ignored in this mode)

- ONCE=1'b0 (want to count repeatedly)
- LENGTH=1'b1 (want to count until compare value is reached and re-initialize counter register)
- DIR=Any (user's choice. The compare register values must be chosen carefully to account for things like roll-under, etc.)
- COINIT=1'b0 (user can set this if they need this function)
- OUTMODE=3'b100 (toggle OFLAG output using alternating compare registers)

Timer Status and Control Register (SCTRL)

- OEN = 1'b1 (output enable to allow OFLAG output to be put on an external pin. Set this bit as needed.)
- OPS = Any (user's choice)
- Make sure the rest of the bits are cleared for this register. We will enable interrupts in the comparator status and control register instead of in this register.

Comparator Status and Control Register (CSCTRL)

- TCF2EN=1'b1 (allow interrupt to be issued when CSCTRL[TCF2] is set)
- TCF1EN=1'b0 (do not allow interrupt to be issued when CSCTRL[TCF1] is set)
- TCF1=1'b0 (clear timer compare 1 interrupt source flag. This is set when counter register equals compare register 1 value and OFLAG is low)
- TCF2=1'b0 (clear timer compare 2 interrupt source flag. This is set when counter register equals compare register 2 value and OFLAG is high)
- CL1=2'b10 (load compare register when CSCTRL[TCF2] is asserted)
- CL2=2'b01 (load compare register when CSCTRL[TCF1] is asserted)

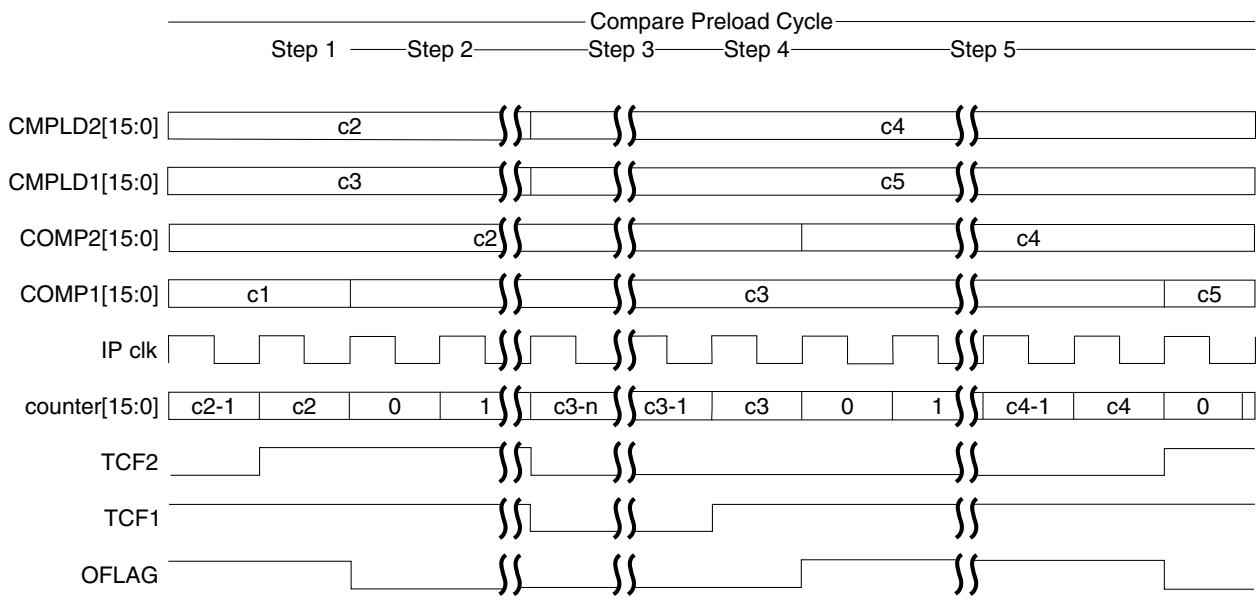
Interrupt Service Routines

To service the CSCTRL[TCF2] interrupts generated by the timer, the interrupt controller must be configured to enable the interrupts for the particular timer being used. Additionally the user will need to write an interrupt service routine to do at a minimum the following:

- Clear CSCTRL[TCF2] and CSCTRL[TCF1] flags.
- Calculate and write new values for both CMPLD1 and CMPLD2.

Timing

This figure contains the timing for using the compare preload feature. The compare preload cycle begins with a compare event on COMP2 causing CSCTRL[TCF2] to be asserted. COMP1 is loaded with the value in the CMPLD1 (c3) one IP bus clock later. In addition an interrupt is asserted by the timer and the interrupt service routine is executed during which both comparator load registers are updated with new values (c4 and c5). When CSCTRL[TCF1] is asserted, COMP2 is loaded with the value in CMPLD2 (c4). And on the subsequent CSCTRL[TCF2] event, COMP1 is loaded with the value in CMPLD1 (c5). The cycle starts over again as an interrupt is asserted and the interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and calculates new values for CMPLD1 and CMPLD2.



Step 1-- CNTR matches COMP2 value. CSCTRL[TCF2] is asserted and an interrupt request is generated.

Step 2-- One clock later, OFLAG toggles, CMPLD1 is copied to COMP1, LOAD is copied to CNTR, the counter starts counting.

Step 3-- The interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and the ISR loads CMPLD1 and CMPLD2 with the values for the next cycle. The counter continues counting until CNTR matches COMP1.

Step 4-- CSCTRL[TCF1] is asserted. One clock later, OFLAG toggles, CMPLD2 is copied to COMP2, LOAD is copied to CNTR and the counter starts counting.

Step 5--The counter continues counting until CNTR matches COMP2.

Figure 6-124. Compare Load Timing

Example: 6.6.2.14.1 Variable Frequency PWM Mode

```
//    (See Processor Expert PPG [Programmable Pulse Generator] bean.)
// This example starts with an 11 msec with a 31 msec cycle.
// Assuming the chip is operating at 60 MHz, the timer use IP_bus_clk/32 as its
//   clock source.
//
// Initial pulse period: 60e6/32 clocks/sec * 31 ms = 58125 total clocks in period
// Initial pulse width: 60e6/32 clocks/sec * 11 ms = 20625 clocks in pulse
//
//
// Once the initial values of COMP1/CMPLD1 and COMP2/CMPLD2 are set the pulse width
//   can be varied by load new values of CMPLD1 and CMPLD2 on each compare interrupt.
//    (See Usage of Compare Load Registers.)
//
void PPG1_Init(void)
{
  setReg(TMRA0_LOAD,0);                /* Clear load register */
  setReg(TMRA0_CNTR,0);                /* Clear counter */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
  setReg(TMRA0_SCTRL,5);                /* Set Status and Control Register */
// Set compare preload operation and enable an interrupt on compare2 events.
  /* TMRA0_CSCTRL: TCF2EN=1,TCF1EN=0,TCF2=0,TCF1=0,CL21=0,CL20=1,CL11=1,CL10=0 */
  setReg(TMRA0_CSCTRL,0x86);          /* Set Comparator Status and Control Register */

  setReg(TMRA0_COMP1,20625);           /* Set the pulse width of the off time */
  setReg(TMRA0_CMPLD1,20625);         /* Set the pulse width of the off time */
  setReg(TMRA0_COMP2,58125-20625);     /* Set the pulse width of the on time */
  setReg(TMRA0_CMPLD2,58125-20625);   /* Set the pulse width of the on time */
  /* TMRA0_CTRL: CM=1,PCS=0xD,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=4 */
  setRegBits(TMRA0_CTRL,0x3A24);       /* Set variable PWM mode and run counter */
}
```

6.6.2.15 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability. The COMP1 register is used when the counter is *counting up*, and the COMP2 register is used when the counter is *counting down*. Alternating compare mode is the only exception.

The COMP1 register should be set to the desired maximum count value or FFFFh to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0000h to indicate the minimum unsigned value prior to roll-under.

If CTRL[OUTMODE] is set to 100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time. The variable frequency PWM mode is defined for positive counting only.

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to FFFFh or 0000h, roll over, then begin counting toward the new value. The check is: Count=CMPx, *not* Count > COMP1 or Count < COMP2.

The use of the CMPLD1 and CMPLD2 registers to compare values will help to minimize this problem.

6.6.2.16 Usage of Compare Load Registers

The CMPLD1, CMPLD2, and CSCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. In the past, a compare register could be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there was the possibility that the counter may have already counted past the new compare value by the time the compare register was updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are now updated in hardware in the same way the counter register is re-initialized to the value stored in the load register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See the following figure.

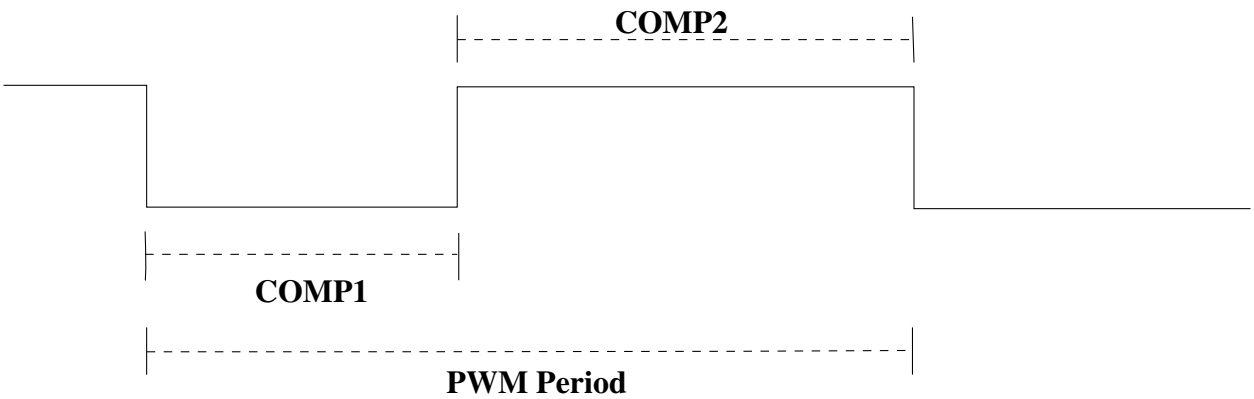


Figure 6-125. Variable PWM Waveform

Should we desire to update the duty cycle or period of the above waveform, we would need to update the COMP1 and COMP2 values using the compare load feature.

6.6.2.17 Usage of the Capture Register

The capture register stores a copy of the counter's value when an input edge (positive, negative, or both) is detected. After a capture event occurs, no further updating of the capture register will occur until the SCTRL[IEF] (input edge flag) is cleared by writing a zero to the SCTRL[IEF].

6.7 Resets

6.7.1 General

The TMR module can be reset only by the RST_B signal. This forces all registers to their reset state and clears the OFLAG signal if it is asserted. The counter will be turned off until the settings in the control register are changed.

Table 6-127. Reset Summary

Reset	Priority	Source	Characteristics
RST_B	n/a	Hardware Reset	Full System Reset

6.8 Clocks

6.8.1 General

The timer only receives the IP bus clock (system clock) at a 1x or 2x rate, which can be selected in the SIM.

6.9 Interrupts

6.9.1 General

The TMR module can generate 20 interrupts, Five for each of the four counters/channels.

Table 6-128. Interrupt Summary

Core Interrupt	Interrupt	Description
TMR Channel 0	TMR0_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 0
	TMR0_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 0
	TMR0_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 0
	TMR0_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 0
	TMR0_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 0
TMR Channel 1	TMR1_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 1
	TMR1_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 1
	TMR1_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 1
	TMR1_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 1
	TMR1_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 1
TMR Channel 2	TMR2_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 2
	TMR2_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 2
	TMR2_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 2
	TMR2_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 2
	TMR2_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 2

Table continues on the next page...

Table 6-128. Interrupt Summary (continued)

Core Interrupt	Interrupt	Description
TMR Channel 3	TMR3_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 3
	TMR3_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 3
	TMR3_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 3
	TMR3_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 3
	TMR3_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 3

6.9.2 Description of Interrupt Operation

6.9.2.1 Timer Compare Interrupts

These interrupts are generated when a successful compare occurs between a counter and it's compare registers while SCTRL[TCFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[TCF].

When a timer compare interrupt is set in TMR_SCTRL and the Compare Load registers are available, one of the following two interrupts will also be asserted.

6.9.2.1.1 Timer Compare 1 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP1 register while CSCTRL[TCF1EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF1].

6.9.2.1.2 Timer Compare 2 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP2 register while CSCTRL[TCF2EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF2].

6.9.2.2 Timer Overflow Interrupts

These interrupts are generated when a counter rolls over its maximum value while SCTRL[TOFIE] is set. These interrupts are cleared by writing zero to the appropriate SCTRL[TOF].

6.9.2.3 Timer Input Edge Interrupts

These interrupts are generated by a transition of the input signal (either positive or negative depending on IPS setting) while SCTRL[IEFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[IEF].

Chapter 7
Enhanced Flex Pulse Width Modulator (eFlexPWM)

7.1 Introduction

The pulse width modulator (PWM) module contains PWM submodules, each of which is set up to control a single half-bridge power stage. Fault channel support is provided.

This PWM module can generate various switching patterns, including highly sophisticated waveforms. It can be used to control all known motor types and is ideal for controlling different Switched Mode Power Supplies (SMPS) topologies as well.

7.1.1 Features

- 16 bits of resolution for center, edge aligned, and asymmetrical PWMs
- Fractional delay for enhanced resolution of the PWM period and edge placement
- PWM outputs that can operate as complementary pairs or independent channels
- Ability to accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Support for synchronization to external hardware or other PWM
- Double buffered PWM registers
    - Integral reload rates from 1 to 16
    - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output

- All outputs can be programmed to change simultaneously via a FORCE_OUT event
- PWMX pin can optionally output a third PWM signal from each submodule
- Channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions
- Enhanced dual edge capture functionality
- The option to supply the source for each complementary PWM signal pair from any of the following:
  - Crossbar module outputs
  - External ADC input, taking into account values set in ADC high and low limit registers

7.1.1.1 Inter-Module Connections

On the MC56F825x/MC56F824x, all external control signals can be configured for a wide range of inter-module connections via the crossbar. Connections can be arbitrarily made to ADC inputs, a comparator output, GPIOs, timers, or a DAC. For more details, refer to the crossbar switch module's chapter and the device's data sheet.

The external control signals are FAULT[n], PWM[n]_EXT_SYNC, EXT_FORCE, PWM[n]_EXTA and PWM[n]_EXTB, EXT_CLK, and PWM[n]_OUT_TRIGx, where n can be 0, 1, 2, or 3 and x can be 0 or 1. The "n" represents the submodule number of the PWM peripheral, which has four submodules. Each submodule is a pulse width modulator in its own right.

7.1.1.2 Submodules and Fault Channel

The MC56F825x/MC56F824x's eFlexPWM module contains four submodules and has one fault channel, fault channel zero, which accommodates four distinct fault inputs.

The following table identifies PWM feature support that varies by submodule (SM).

Table 7-1. PWM Submodule Feature Support

Submodule(s)	Fractional Delay	Input Capture Functions
SM0, SM1, SM2	Yes	No
SM3	No	Yes (FIFO depth is 1)

7.1.2 Modes of Operation

Be careful when using this module in some operating modes.

CAUTION

Some applications (such as three-phase AC motors) require regular software updates for proper operation. Failure to provide regular software updates could result in destroying the hardware setup.

To accommodate this situation, PWM outputs are placed in their inactive states in stop mode, and they can optionally be placed in inactive states in wait and debug (EOnCE) modes. PWM outputs are reactivated (assuming they were active beforehand) when these modes are exited.

Table 7-2. Modes When PWM Operation is Restricted

Mode	Description
Stop	PWM outputs are inactive.
Wait	PWM outputs are driven or inactive as a function of CTRL2[WAITEN].
Debug	CPU and peripheral clocks continue to run, but the CPU may stall for periods of time. PWM outputs are driven or inactive as a function of CTRL2[DBGEN].

7.1.3 Block Diagram

The following figure is a block diagram of the PWM.

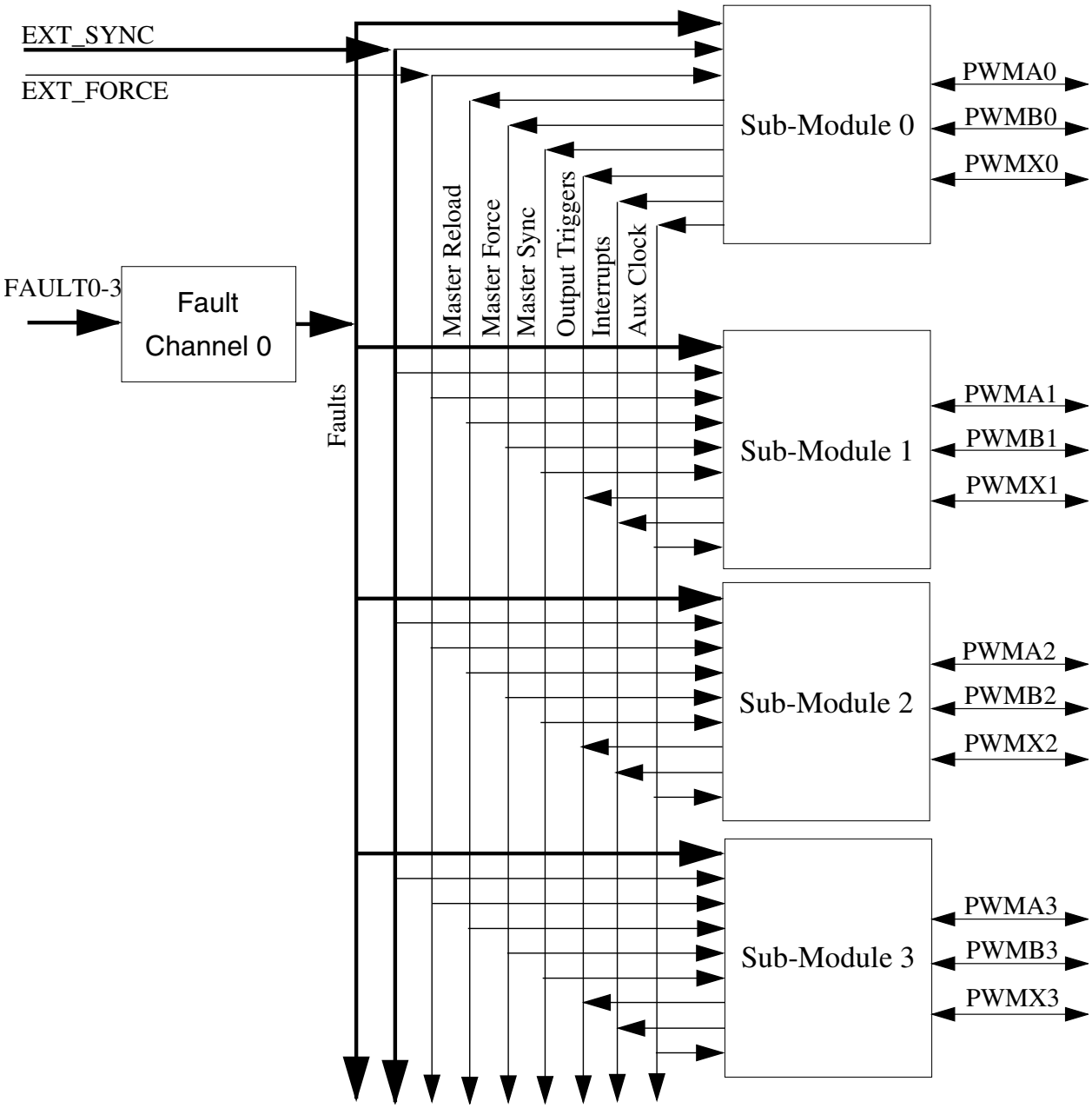


Figure 7-1. PWM Block Diagram

7.1.3.1 PWM Submodule

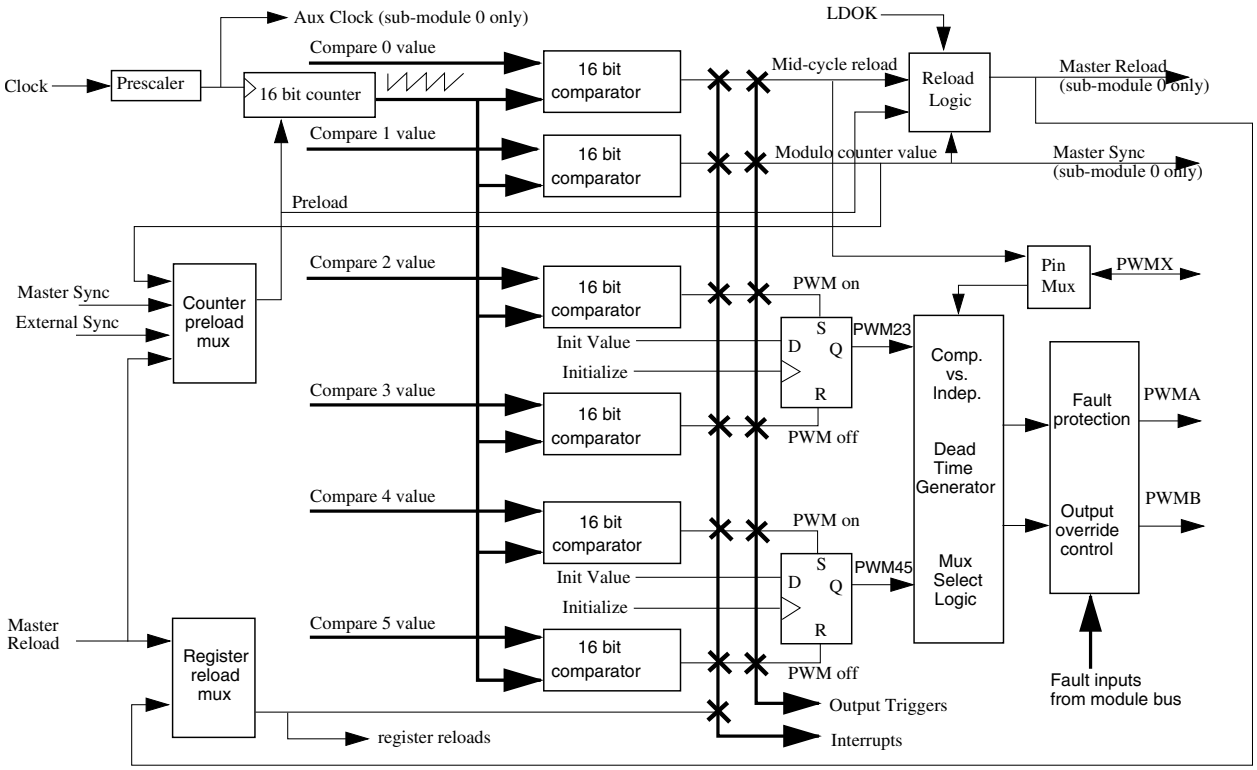


Figure 7-2. PWM Submodule Block Diagram

7.2 Signal Descriptions

The PWM has pins named PWM[n]A, PWM[n]B, PWM[n]X, FAULT[n], PWM[n]_EXT_SYNC, EXT_FORCE, PWM[n]_EXTA, and PWM[n]_EXTB. The PWM also has an on-chip input called EXT_CLK and output signals called PWM[n]_OUT_TRIGx.

7.2.1 PWM Signals and Crossbar Module

On the MC56F825x/MC56F824x, some PWM signals are fed to or from the crossbar switch module or are connected to other modules by the crossbar switch.

- The FAULT[n] input pins are fed from the crossbar module. They are not assigned directly to package pins.
- The PWM[n]_EXT_SYNC input signals are fed from the crossbar module.
- The EXT_FORCE signal is fed from the crossbar module.

- For the possible connections and muxing of the PWM[n]_EXTA and PWM[n]_EXTB signals, refer to the crossbar switch module's chapter.
- The PWM[n]_OUT_TRIG0 and PWM[n]_OUT_TRIG1 output triggers are fed into the crossbar module. For some PWM submodules, OR logic placed between the PWM and crossbar ORs the signals together.

Refer to the crossbar switch module's chapter and the device's data sheet for details.

7.2.2 PWM[n]A and PWM[n]B - External PWM Pair

These pins are the output pins of the PWM channels. They can be independent PWM signals or a complementary pair. When not needed as an output, they can be used as inputs to the input capture circuitry.

7.2.3 PWM[n]X - Auxiliary PWM signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or used to detect the polarity of the current flowing through the complementary circuit at deadtime correction.

7.2.4 FAULT[n] - Fault inputs

These are input pins for disabling selected PWM outputs.

7.2.5 PWM[n]_EXT_SYNC - External Synchronization Signal

These input signals allow a source external to the PWM to initialize the PWM counter. In this manner, the PWM can be synchronized to external circuitry.

7.2.6 EXT_FORCE - External Output Force Signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. In this manner, the PWM can be synchronized to external circuitry.

7.2.7 PWM[n]_EXTA and PWM[n]_EXTB - Alternate PWM Control Signals

These pins allow an alternate source to control the PWMA and PWMB outputs. Typically, either the EXTA or EXTB input (depending on the state of MCTRL[IPOL]) is used for the generation of a complementary pair. Typical control signals include ADC conversion high/low limits, TMR outputs, GPIO inputs, and comparator outputs.

On the MC56F825x/MC56F824x:

- PWM0_EXTA to PWM3_EXTA are connected to crossbar outputs XBAR_OUT12 to XBAR_OUT15.
- PWM0_EXTB connects to ADC sample 0's high/low limit flag. This flag is cleared when the ADC conversion result is higher than the value programmed into the ADC High Limit Register 0, and the flag is set when the ADC conversion result is lower than the value programmed into the ADC Low Limit Register 0.
- PWM1_EXTB connects to ADC sample 1's high/low limit flag. This flag is cleared when the ADC conversion result is higher than the value programmed into the ADC High Limit Register 1, and the flag is set when the ADC conversion result is lower than the value programmed into the ADC Low Limit Register 1.
- PWM2_EXTB connects to ADC sample 2's high/low limit flag. This flag is cleared when the ADC conversion result is higher than the value programmed into the ADC High Limit Register 2, and the flag is set when the ADC conversion result is lower than the value programmed into the ADC Low Limit Register 2.
- PWM3_EXTB is connected to ground.

7.2.8 PWM[n]_OUT_TRIG0 and PWM[n]_OUT_TRIG1 - Output Triggers

These outputs allow the PWM submodules to control timing of ADC conversions. See the description of the Output Trigger Control Register for information about how to enable these outputs and how the compare registers match up to the output triggers.

7.2.9 EXT_CLK - External Clock Signal

This on-chip input signal allows an on-chip source external to the PWM (typically a timer) to control the PWM clocking. In this manner, the PWM can be synchronized to the on-chip source. This signal must be generated synchronously to the PWM's clock because it is not resynchronized in the PWM.

7.3 Memory Map and Registers

Register details appear in these groups:

1. Register descriptions for submodules 0, 1, and 2 appear together, as a single group. Each of these submodules has an identical register set.
2. Submodule 3's register descriptions appear separately because its registers vary from those of submodules 0, 1, and 2.
3. The final set of configuration registers, based at an offset of C0h, applies to the entire PWM peripheral. Some of these registers contain bitfields for configuring an individual submodule, while other registers affect all submodules at once. This final register set also includes fault channel registers.

The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is equal to the base address of submodule 0 plus the offset 30h. The base address of submodule 2 is equal to the base address of submodule 1 plus an additional 30h offset, and so on for the base address of submodule 3. Submodule register names contain SM0, SM1, SM2, or SM3 to designate the applicable submodule.

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PWM_SM0CNT	R	CNT															
		W																
1	PWM_SM0INIT	R	INIT															
		W																
2	PWM_SM0CTRL2	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
		W										FORCE						
3	PWM_SM0CTRL	R	LDFQ				HALF	FULL	DT		0	PRSC			0	LDMOD	0	DBLEN
		W																
4	Reserved	R	Reserved															
		W																
5	PWM_SM0VAL0	R	VAL0															
		W																
6	PWM_SM0FRACVAL1	R	FRACVAL1					0										
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7	PWM_SM0VAL1	R	VAL1															
		W																
8	PWM_SM0FRACVAL2	R	FRACVAL2					0										
		W																
9	PWM_SM0VAL2	R	VAL2															
		W																
A	PWM_SM0FRACVAL3	R	FRACVAL3					0										
		W																
B	PWM_SM0VAL3	R	VAL3															
		W																
C	PWM_SM0FRACVAL4	R	FRACVAL4					0										
		W																
D	PWM_SM0VAL4	R	VAL4															
		W																
E	PWM_SM0FRACVAL5	R	FRACVAL5					0										
		W																
F	PWM_SM0VAL5	R	VAL5															
		W																
10	PWM_SM0FRCTRL	R	RESERVED	0							FRAC_PU	0		FRAC45_EN	0	FRAC23_EN	FRAC1_EN	0
		W																
11	PWM_SM0OCTRL	R	PWMA_IN	PWMB_IN	PWMX_IN	0			POLA	POLB	POLX	0		PWMAFS		PWMBFS		PWMXFS
		W																
12	PWM_SM0STS	R	0	RUF	REF	RF	0					0		CMPF				
		W																
13	PWM_SM0INTEN	R	0		REIE	RIE	0					0		CMPIE				
		W																
14	Reserved	R	Reserved															
		W																
15	PWM_SM0TCTRL	R	0											OUT_TRIG_EN				
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16	PWM_SM0DISMAP	R	1	1	1	1	DISX				DISB				DISA			
		W																
17	PWM_SM0DTCNT0	R	0								DTCNT0							
		W																
18	PWM_SM0DTCNT1	R	0								DTCNT1							
		W																
19	Reserved	R	Reserved															
		W																
30	PWM_SM1CNT	R	CNT															
		W																
31	PWM_SM1INIT	R	INIT															
		W																
32	PWM_SM1CTRL2	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
		W										FORCE						
33	PWM_SM1CTRL	R	LDFQ				HALF	FULL	DT		0	PRSC			0	LDMOD	0	DBLEN
		W																
35	PWM_SM1VAL0	R	VAL0															
		W																
36	PWM_SM1FRACVAL1	R	FRACVAL1					0										
		W																
37	PWM_SM1VAL1	R	VAL1															
		W																
38	PWM_SM1FRACVAL2	R	FRACVAL2					0										
		W																
39	PWM_SM1VAL2	R	VAL2															
		W																
3A	PWM_SM1FRACVAL3	R	FRACVAL3					0										
		W																
3B	PWM_SM1VAL3	R	VAL3															
		W																
3C	PWM_SM1FRACVAL4	R	FRACVAL4					0										
		W																
3D	PWM_SM1VAL4	R	VAL4															
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3E	PWM_SM1FRACVAL5	R	FRACVAL5					0										
		W																
3F	PWM_SM1VAL5	R	VAL5															
		W																
40	PWM_SM1FRCTRL	R	RESERVED	0						FRAC_PU	0			FRAC45_EN	0	FRAC23_EN	FRAC1_EN	0
		W																
41	PWM_SM1OCTRL	R	PWMA_IN	PWMB_IN	PWMX_IN	0		POLA	POLB	POLX	0			PWMAFS		PWMBFS		PWMXFS
		W																
42	PWM_SM1STS	R	0	RUF	REF	RF		0			0			CMPF				
		W																
43	PWM_SM1INTEN	R	0		REIE	RIE		0			0			CMPIE				
		W																
45	PWM_SM1TCTRL	R	0											OUT_TRIG_EN				
		W																
46	PWM_SM1DISMAP	R	1	1	1	1		DISX			DISB			DISA				
		W																
47	PWM_SM1DTCNT0	R	0					DTCNT0										
		W																
48	PWM_SM1DTCNT1	R	0					DTCNT1										
		W																
60	PWM_SM2CNT	R	CNT															
		W																
61	PWM_SM2INIT	R	INIT															
		W																
62	PWM_SM2CTRL2	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0		FORCE_SEL			RELOAD_SEL	CLK_SEL
		W										FORCE						
63	PWM_SM2CTRL	R	LDFQ				HALF	FULL	DT		0	PRSC			0	LDMOD	0	DBLEN
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
65	PWM_SM2VAL0	R	VAL0															
		W																
66	PWM_SM2FRACVAL1	R	FRACVAL1					0										
		W																
67	PWM_SM2VAL1	R	VAL1															
		W																
68	PWM_SM2FRACVAL2	R	FRACVAL2					0										
		W																
69	PWM_SM2VAL2	R	VAL2															
		W																
6A	PWM_SM2FRACVAL3	R	FRACVAL3					0										
		W																
6B	PWM_SM2VAL3	R	VAL3															
		W																
6C	PWM_SM2FRACVAL4	R	FRACVAL4					0										
		W																
6D	PWM_SM2VAL4	R	VAL4															
		W																
6E	PWM_SM2FRACVAL5	R	FRACVAL5					0										
		W																
6F	PWM_SM2VAL5	R	VAL5															
		W																
70	PWM_SM2FRCTRL	R	RESERVED	0							FRAC_PU	0			FRAC45_EN	0	FRAC23_EN	FRAC1_EN	0
		W																	
71	PWM_SM2OCTRL	R	PWMA_IN	PWMB_IN	PWMX_IN	0			POLA	POLB	POLX	0		PWMAFS		PWMBFS		PWMXFS	
		W																	
72	PWM_SM2STS	R	0	RUF	REF	RF		0				0		CMPF					
		W																	
73	PWM_SM2INTEN	R	0		REIE	RIE		0				0		CMPIE					
		W																	
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
75	PWM_SM2TCTRL	R	0										OUT_TRIG_EN					
		W																
76	PWM_SM2DISMAP	R	1	1	1	1	DISX				DISB				DISA			
		W																
77	PWM_SM2DTCNT0	R	0															
		W									DTCNT0							
78	PWM_SM2DTCNT1	R	0															
		W									DTCNT1							
90	PWM_SM3CNT	R	CNT															
		W																
91	PWM_SM3INIT	R	INIT															
		W																
92	PWM_SM3CTRL2	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
		W										FORCE						
93	PWM_SM3CTRL	R	LDFQ				HALF	FULL	DT		0	PRSC			0	LDMOD	0	DBLEN
		W																
95	PWM_SM3VAL0	R	VAL0															
		W																
97	PWM_SM3VAL1	R	VAL1															
		W																
99	PWM_SM3VAL2	R	VAL2															
		W																
9B	PWM_SM3VAL3	R	VAL3															
		W																
9D	PWM_SM3VAL4	R	VAL4															
		W																
9F	PWM_SM3VAL5	R	VAL5															
		W																
A1	PWM_SM3OCTRL	R	PWMA_IN	PWMB_IN	PWMX_IN	0		POLA	POLB	POLX	0		PWMAFS		PWMBFS		PWMXFS	
		W																
A2	PWM_SM3STS	R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	CFX1	CFX0	CMPF					
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A3	PWM_SM3INTEN	R	0										CMPIE					
		W			REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	CX1IE	CX0IE						
A5	PWM_SM3TCTRL	R	0															OUT_TRIG_EN
		W																
A6	PWM_SM3DISMAP	R	1	1	1	1	DISX				DISB				DISA			
		W																
A7	PWM_SM3DTCNT0	R	0								DTCNT0							
		W																
A8	PWM_SM3DTCNT1	R	0								DTCNT1							
		W																
AA	PWM_SM3CAPTCTRLA	R	CA1CNT				CA0CNT				CFAWM	EDGCNTA_EN	INP_SELA	EDGA1		EDGA0	ONESHOTA	ARMA
		W																
AB	PWM_SM3CAPTCOMPA	R	EDGCNTA								EDGCMPA							
		W																
AC	PWM_SM3CAPTCTRLB	R	CB1CNT				CB0CNT				CFBWM	EDGCNTB_EN	INP_SELB	EDGB1		EDGB0	ONESHOTB	ARMB
		W																
AD	PWM_SM3CAPTCOMPB	R	EDGCNTB								EDGCMPB							
		W																
AE	PWM_SM3CAPTCTRLX	R	CX1CNT				CX0CNT				CFXWM	EDGCNTX_EN	INP_SELX	EDGX1		EDGX0	ONESHOTX	ARMX
		W																
AF	PWM_SM3CAPTCOMPX	R	EDGCNTX								EDGCMPX							
		W																
B0	PWM_SM3CVAL0	R	CAPTVAL0															
		W																
B2	PWM_SM3CVAL1	R	CAPTVAL1															
		W																
B4	PWM_SM3CVAL2	R	CAPTVAL2															
		W																
B6	PWM_SM3CVAL3	R	CAPTVAL3															
		W																
B8	PWM_SM3CVAL4	R	CAPTVAL4															
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA	PWM_SM3CVAL5	R	CAPTVAL5															
		W																
C0	PWM_OUTEN	R	0				PWMA_EN				PWMB_EN				PWMX_EN			
		W																
C1	PWM_MASK	R	0				MASKA				MASKB				MASKX			
		W																
C2	PWM_SWCOUT	R	0								SM3OUT23	SM3OUT45	SM2OUT23	SM2OUT45	SM1OUT23	SM1OUT45	SM0OUT23	SM0OUT45
		W																
C3	PWM_DTSRCSEL	R	SM3SEL23			SM3SEL45			SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23	SM0SEL45
		W																
C4	PWM_MCTRL	R	IPOL				RUN								LDOK			
		W									CLDOK							
C5	PWM_MCTRL2	R	0															MONPLL
		W																
C6	PWM_FCTRL	R	FLVL				FAUTO				FSAFE				FIE			
		W																
C7	PWM_FSTS	R	0				FTEST	FFPIN			FFULL				FFLAG			
		W																
C8	PWM_FFILT	R	GSTR	0				FILT_CNT			FILT_PER							
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

7.3.1 PWM SMx Counter Register (PWM_SM*n*CNT)

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible.

Addresses: PWM_SM0CNT – F300h base + 0h offset = F300h
PWM_SM1CNT – F300h base + 30h offset = F330h
PWM_SM2CNT – F300h base + 60h offset = F360h

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

PWM_SM*n*CNT field descriptions

Field	Description
15–0 CNT	Counter Register Bits

7.3.2 PWM SMx Initial Count Register (PWM_SM*n*INIT)

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of CTRL2[INIT_SEL]) or when CTRL2[FORCE] is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

The INIT register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Addresses: PWM_SM0INIT – F300h base + 1h offset = F301h
PWM_SM1INIT – F300h base + 31h offset = F331h
PWM_SM2INIT – F300h base + 61h offset = F361h

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

PWM_SM*n*INIT field descriptions

Field	Description
15–0 INIT	Initial Count Register Bits

7.3.3 PWM SMx Control 2 Register (PWM_SM*n*CTRL2)

Addresses: PWM_SM0CTRL2 – F300h base + 2h offset = F302h

PWM_SM1CTRL2 – F300h base + 32h offset = F332h

PWM_SM2CTRL2 – F300h base + 62h offset = F362h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
Write										FORCE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*CTRL2 field descriptions

Field	Description
15 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in debug mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates will not occur in debug mode. Any motors requiring such updates should be disabled during debug mode. If in doubt, leave this bit set to zero.
14 WAITEN	WAIT Enable When set to one, the PWM will continue to run while the chip is in WAIT mode. In this mode, the peripheral clock continues to run but the CPU clock does not. If the device enters WAIT mode and this bit is zero, then the PWM outputs will be disabled until WAIT mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in WAIT mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in WAIT mode. The key point is PWM parameter updates will not occur in this mode. Any motors requiring such updates should be disabled during WAIT mode. If in doubt, leave this bit set to zero.
13 INDEP	Independent or Complementary Pair Operation This bit determines if the PWMA and PWMB channels will be independent PWMs or a complementary PWM pair. 0 PWMA and PWMB form a complementary PWM pair. 1 PWMA and PWMB outputs are independent PWMs.
12 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT.
11 PWM45_INIT	PWM45 Initial Value

Table continues on the next page...

PWM_SM*n*CTRL2 field descriptions (continued)

Field	Description
	This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT.
10 PWMX_INIT	PWMX Initial Value This read/write bit determines the initial value for PWMX and the value to which it is forced when FORCE_INIT.
9–8 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00 Local sync (PWMX) causes initialization. 01 Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 10 Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 11 EXT_SYNC causes initialization.
7 FRCEN	Force Initialization Enable This bit allows the CTRL2[FORCE] signal to initialize the counter without regard to the signal selected by CTRL2[INIT_SEL]. This is a software controlled initialization. 0 Initialization from a FORCE_OUT event is disabled. 1 Initialization from a FORCE_OUT event is enabled.
6 FORCE	Force Initialization If CTRL2[FORCE_SEL] is set to 000, writing a 1 to this bit results in a FORCE_OUT event. This causes the following actions to be taken: • The PWMA and PWMB output pins will assume values based on DTSRCSEL[SMxSEL23] and DTSRCSEL[SMxSEL45]. • If CTRL2[FRCEN] is set, the counter value will be initialized with the INIT register value.
5–3 FORCE_SEL	This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000 The local force signal, CTRL2[FORCE], from this submodule is used to force updates. 001 The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010 The local reload signal from this submodule is used to force updates. 011 The master reload signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100 The local sync signal from this submodule is used to force updates. 101 The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110 The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111 reserved
2 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, MCTRL[LDOK[0]] for submodule 0 should be used since the local MCTRL[LDOK] will be ignored. 0 The local RELOAD signal is used to reload registers. 1 The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.

Table continues on the next page...

PWM_SM*n*CTRL2 field descriptions (continued)

Field	Description
1–0 CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00 The IPBus clock is used as the clock for the local prescaler and counter. 01 EXT_CLK is used as the clock for the local prescaler and counter. 10 Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11 reserved

7.3.4 PWM SMx Control Register (PWM_SM*n*CTRL)

Addresses: PWM_SM0CTRL – F300h base + 3h offset = F303h
PWM_SM1CTRL – F300h base + 33h offset = F333h
PWM_SM2CTRL – F300h base + 63h offset = F363h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LDFQ				HALF	FULL	DT		0	PRSC			0	LDMOD	0	DBLEN
Write																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*CTRL field descriptions

Field	Description
15–12 LDFQ	0000 Every PWM Opportunity 0001 Every 2 PWM Opportunities 0010 Every 3 PWM Opportunities 0011 Every 4 PWM Opportunities 0100 Every 5 PWM Opportunities 0101 Every 6 PWM Opportunities 0110 Every 7 PWM Opportunities 0111 Every 8 PWM Opportunities 1000 Every 9 PWM Opportunities 1001 Every 10 PWM Opportunities 1010 Every 11 PWM Opportunities 1011 Every 12 PWM Opportunities 1100 Every 13 PWM Opportunities 1101 Every 14 PWM Opportunities 1110 Every 15 PWM Opportunities 1111 Every 16 PWM Opportunities
11 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be half way through the PWM cycle.

Table continues on the next page...

PWM_SM*n*CTRL field descriptions (continued)

Field	Description
	0 Half-cycle reloads disabled. 1 Half-cycle reloads enabled.
10 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either CTRL[HALF] or CTRL[FULL] must be set in order to move the buffered data into the registers used by the PWM generators. If both CTRL[HALF] and CTRL[FULL] are set, then reloads can occur twice per cycle. 0 Full-cycle reloads disabled. 1 Full-cycle reloads enabled.
9–8 DT	Deadtime These read only bits reflect the sampled values of the PWMX input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.
7 Reserved	This read-only bit is reserved and always has the value zero.
6–4 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CTRL2[CLK_SEL]. Reading CTRL[PRSC] reads the buffered values and not necessarily the values currently in effect. CTRL[PRSC] takes effect at the beginning of the next PWM cycle and only when the load okay bit, MCTRL[LDOK], is set or CTRL[LDMOD] is set. This field cannot be written when MCTRL[LDOK] is set. In the bitfield-setting descriptions, f_{clk} is the clock input to the PWM peripheral clock input, which appears in the upper left corner of Figure 7-2. This clock is also known as the IPBus clock or simply as the peripheral clock. 000 PWM clock frequency = fclk 001 PWM clock frequency = fclk/2 010 PWM clock frequency = fclk/4 011 PWM clock frequency = fclk/8 100 PWM clock frequency = fclk/16 101 PWM clock frequency = fclk/32 110 PWM clock frequency = fclk/64 111 PWM clock frequency = fclk/128
3 Reserved	This read-only bit is reserved and always has the value zero.
2 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. 0 Buffered registers of this submodule are loaded and take effect at the next PWM reload if MCTRL[LDOK] is set. 1 Buffered registers of this submodule are loaded and take effect immediately upon MCTRL[LDOK] being set.
1 Reserved	This read-only bit is reserved and always has the value zero.

Table continues on the next page...

PWM_SM*n*CTRL field descriptions (continued)

Field	Description
0 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior. 0 Double switching disabled. 1 Double switching enabled.

7.3.5 PWM SMx Value Register 0 (PWM_SM*n*VAL0)

Addresses: PWM_SM0VAL0 – F300h base + 5h offset = F305h
PWM_SM1VAL0 – F300h base + 35h offset = F335h
PWM_SM2VAL0 – F300h base + 65h offset = F365h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*VAL0 field descriptions

Field	Description
15–0 VAL0	Value Register 0 The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWMX signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE:** The VAL0 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL0 cannot be written when MCTRL[LDOK] is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

7.3.6 PWM SMx Fractional Value Register 1 (PWM_SM*n*FRACVAL1)

Addresses: PWM_SM0FRACVAL1 – F300h base + 6h offset = F306h
PWM_SM1FRACVAL1 – F300h base + 36h offset = F336h
PWM_SM2FRACVAL1 – F300h base + 66h offset = F366h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL1					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*FRACVAL1 field descriptions

Field	Description
15–11 FRACVAL1	Fractional Value 1 Register These bits act as a fractional addition to the value in the VAL1 register which controls the PWM period width. The PWM period is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL1 is temporarily incremented and the PWM cycle is extended by one clock period to compensate for the accumulated fractional values. **NOTE:** The FRACVAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL1 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.
10–0 Reserved	This read-only bitfield is reserved and always has the value zero.

7.3.7 PWM SMx Value Register 1 (PWM_SM*n*VAL1)

Addresses: PWM_SM0VAL1 – F300h base + 7h offset = F307h

PWM_SM1VAL1 – F300h base + 37h offset = F337h

PWM_SM2VAL1 – F300h base + 67h offset = F367h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*VAL1 field descriptions

Field	Description
15–0 VAL1	Value Register 1 The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local sync signal while resetting PWMX. This register is not byte accessible. **NOTE:** The VAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL1 cannot be written when MCTRL[LDOK] is set. Reading VAL1 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

7.3.8 PWM SMx Fractional Value Register 2 (PWM_SM*n*FRACVAL2)

Addresses: PWM_SM0FRACVAL2 – F300h base + 8h offset = F308h
PWM_SM1FRACVAL2 – F300h base + 38h offset = F338h
PWM_SM2FRACVAL2 – F300h base + 68h offset = F368h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL2					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*FRACVAL2 field descriptions

Field	Description
15–11 FRACVAL2	Fractional Value 2 These bits act as a fractional addition to the value in the VAL2 register which controls the PWMA turn on timing. It is also used to control the fractional addition to the turn off delay of PWMB when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL2 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.
10–0 Reserved	This read-only bitfield is reserved and always has the value zero.

7.3.9 PWM SMx Value Register 2 (PWM_SM*n*VAL2)

Addresses: PWM_SM0VAL2 – F300h base + 9h offset = F309h
PWM_SM1VAL2 – F300h base + 39h offset = F339h
PWM_SM2VAL2 – F300h base + 69h offset = F369h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	VAL2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*VAL2 field descriptions

Field	Description
15–0 VAL2	Value Register 2

PWM_SM*n*VAL2 field descriptions (continued)

Field	Description
	The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE:** The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL2 cannot be written when MCTRL[LDOK] is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.10 PWM SMx Fractional Value Register 3 (PWM_SM*n*FRACVAL3)

Addresses: PWM_SM0FRACVAL3 – F300h base + Ah offset = F30Ah

PWM_SM1FRACVAL3 – F300h base + 3Ah offset = F33Ah

PWM_SM2FRACVAL3 – F300h base + 6Ah offset = F36Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL3					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*FRACVAL3 field descriptions

Field	Description
15–11 FRACVAL3	Fractional Value 3 Register These bits act as a fractional addition to the value in the VAL3 register which controls the PWMA turn off timing. It is also used to control the fractional addition to the turn on delay of PWMB when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL3 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.
10–0 Reserved	This read-only bitfield is reserved and always has the value zero.

7.3.11 PWM SMx Value Register 3 (PWM_SM*n*VAL3)

Addresses: PWM_SM0VAL3 – F300h base + Bh offset = F30Bh
PWM_SM1VAL3 – F300h base + 3Bh offset = F33Bh
PWM_SM2VAL3 – F300h base + 6Bh offset = F36Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*VAL3 field descriptions

Field	Description
15–0 VAL3	Value Register 3 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low. This register is not byte accessible. **NOTE:** The VAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL3 cannot be written when MCTRL[LDOK] is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.12 PWM SMx Fractional Value Register 4 (PWM_SM*n*FRACVAL4)

Addresses: PWM_SM0FRACVAL4 – F300h base + Ch offset = F30Ch
PWM_SM1FRACVAL4 – F300h base + 3Ch offset = F33Ch
PWM_SM2FRACVAL4 – F300h base + 6Ch offset = F36Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL4					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*FRACVAL4 field descriptions

Field	Description
15–11 FRACVAL4	Fractional Value 4 These bits act as a fractional addition to the value in the VAL4 register which controls the PWMB turn on timing. It is also used to control the fractional addition to the turn off delay of PWMA when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL4 cannot be

Table continues on the next page...

PWM_SM*n*FRACVAL4 field descriptions (continued)

Field	Description
	written when MCTRL[LDOK] is set. Reading FRACVAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
10–0 Reserved	This read-only bitfield is reserved and always has the value zero.

7.3.13 PWM SMx Value Register 4 (PWM_SM*n*VAL4)

Addresses: PWM_SM0VAL4 – F300h base + Dh offset = F30Dh
PWM_SM1VAL4 – F300h base + 3Dh offset = F33Dh
PWM_SM2VAL4 – F300h base + 6Dh offset = F36Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*VAL4 field descriptions

Field	Description
15–0 VAL4	Value Register 4 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high. This register is not byte accessible. **NOTE:** The VAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL4 cannot be written when MCTRL[LDOK] is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.14 PWM SMx Fractional Value Register 5 (PWM_SM*n*FRACVAL5)

Addresses: PWM_SM0FRACVAL5 – F300h base + Eh offset = F30Eh
PWM_SM1FRACVAL5 – F300h base + 3Eh offset = F33Eh
PWM_SM2FRACVAL5 – F300h base + 6Eh offset = F36Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FRACVAL5					0										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*FRACVAL5 field descriptions

Field	Description
15–11 FRACVAL5	Fractional Value 5 These bits act as a fractional addition to the value in the VAL5 register which controls the PWMB turn off timing. It is also used to control the fractional addition to the turn on delay of PWMa when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0 **NOTE:** The FRACVAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL5 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
10–0 Reserved	This read-only bitfield is reserved and always has the value zero.

7.3.15 PWM SMx Value Register 5 (PWM_SM*n*VAL5)

Addresses: PWM_SM0VAL5 – F300h base + Fh offset = F30Fh
PWM_SM1VAL5 – F300h base + 3Fh offset = F33Fh
PWM_SM2VAL5 – F300h base + 6Fh offset = F36Fh

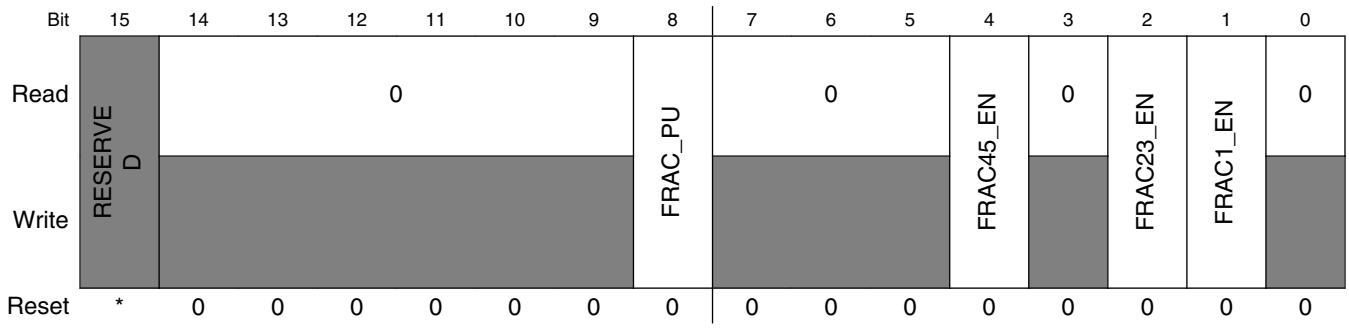
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*VAL5 field descriptions

Field	Description
15–0 VAL5	Value Register 5 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE:** The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL5 cannot be written when MCTRL[LDOK] is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.16 PWM SMx Fractional Control Register (PWM_SM*n*FRCTRL)

Addresses: PWM_SM0FRCTRL – F300h base + 10h offset = F310h
PWM_SM1FRCTRL – F300h base + 40h offset = F340h
PWM_SM2FRCTRL – F300h base + 70h offset = F370h



* Notes:

- RESERVED bitfield:Resets to 0, but may be 0 or 1 when read

PWM_SM*n*FRCTRL field descriptions

Field	Description
15 Reserved	This bit is reserved. This bit resets to 0. When read, its value may be 0 or 1.
14–9 Reserved	This read-only bitfield is reserved and always has the value zero.
8 FRAC_PU	Fractional Delay Circuit Power Up This bit is used to power up the fractional delay analog block. The fractional delay block requires 25 µs to power up after the first FRAC_PU bit in any submodule is set. The fractional delay block powers down only when the FRAC_PU bits in all submodules are 0. The fractional delay logic can be used only when the IPBus clock is running at 60 MHz. When the logic is turned off, fractional placement is disabled. 0 Turn off fractional delay logic. 1 Power up fractional delay logic.
7–5 Reserved	This read-only bitfield is reserved and always has the value zero.
4 FRAC45_EN	Fractional Cycle Placement Enable for PWMB This bit is used to enable the fractional cycle edge placement of PWMB using the FRACVAL4 and FRACVAL5 registers. When the function is disabled, the fractional cycle edge placement of PWMB is bypassed. **NOTE:** The FRAC45_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC45_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC45_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 Bypass fractional cycle placement for PWMB. 1 Enable fractional cycle placement for PWMB.

Table continues on the next page...

PWM_SM*n*FRCTRL field descriptions (continued)

Field	Description
3 Reserved	This read-only bit is reserved and always has the value zero.
2 FRAC23_EN	Fractional Cycle Placement Enable for PWMA This bit is used to enable the fractional cycle edge placement of PWMA using the FRACVAL2 and FRACVAL3 registers. When the function is disabled, the fractional cycle edge placement of PWMA is bypassed. **NOTE:** The FRAC23_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC23_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC23_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 Bypass fractional cycle placement for PWMA. 1 Enable fractional cycle placement for PWMA.
1 FRAC1_EN	Fractional Cycle PWM Period Enable This bit is used to enable the fractional cycle length of the PWM period using the FRACVAL1 register. When the function is disabled, the fractional cycle length of the PWM period is bypassed. **NOTE:** The FRAC1_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC1_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC1_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 Bypass fractional cycle length for the PWM period. 1 Enable fractional cycle length for the PWM period.
0 Reserved	This read-only bit is reserved and always has the value zero.

1. Resets to 0, but may be 0 or 1 when read

7.3.17 PWM SMx Output Control Register (PWM_SM*n*OCTRL)

Addresses: PWM_SM0OCTRL – F300h base + 11h offset = F311h
PWM_SM1OCTRL – F300h base + 41h offset = F341h
PWM_SM2OCTRL – F300h base + 71h offset = F371h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PWMA_IN	PWMB_IN	PWMX_IN	0		POLA	POLB	POLX	0		PWMAFS		PWMBFS		PWMXFS	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM*n*OCTRL field descriptions

Field	Description
15 PWMA_IN	PWMA Input This read only bit shows the logic value currently being driven into the PWMA input.
14 PWMB_IN	PWMB Input This read only bit shows the logic value currently being driven into the PWMB input.
13 PWMX_IN	PWMX Input This read only bit shows the logic value currently being driven into the PWMX input.
12–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10 POLA	PWMA Output Polarity This bit inverts the PWMA output polarity. 0 PWMA output not inverted. A high level on the PWMA pin represents the "on" or "active" state. 1 PWMA output inverted. A low level on the PWMA pin represents the "on" or "active" state.
9 POLB	PWMB Output Polarity This bit inverts the PWMB output polarity. 0 PWMB output not inverted. A high level on the PWMB pin represents the "on" or "active" state. 1 PWMB output inverted. A low level on the PWMB pin represents the "on" or "active" state.
8 POLX	PWMX Output Polarity This bit inverts the PWMX output polarity. 0 PWMX output not inverted. A high level on the PWMX pin represents the "on" or "active" state. 1 PWMX output inverted. A low level on the PWMX pin represents the "on" or "active" state.
7–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–4 PWMAFS	PWMA Fault State These bits determine the fault state for the PWMA output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
3–2 PWMBFS	PWMB Fault State These bits determine the fault state for the PWMB output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.

Table continues on the next page...

PWM_SM*n*OCTRL field descriptions (continued)

Field	Description
1–0 PWMXFS	PWMX Fault State These bits determine the fault state for the PWMX output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.

7.3.18 PWM SMx Status Register (PWM_SM*n*STS)

Addresses: PWM_SM0STS – F300h base + 12h offset = F312h
PWM_SM1STS – F300h base + 42h offset = F342h
PWM_SM2STS – F300h base + 72h offset = F372h

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

PWM_SM*n*STS field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 RUF	Registers Updated Flag This read only flag is set when one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers has been written resulting in non-coherent data in the set of double buffered registers. Clear this bit by a proper reload sequence consisting of a reload signal while MCTRL[LDOK] = 1. Reset clears this bit. 0 No register update has occurred since last reload. 1 At least one of the double buffered registers has been updated since the last reload.
13 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while MCTRL[LDOK] is 0 and the double buffered registers are in a non-coherent state (STS[RUF] = 1). Clear this bit by writing a logic one to this location. Reset clears this bit. 0 No reload error occurred. 1 Reload signal occurred with non-coherent data and MCTRL[LDOK] = 0.
12 RF	Reload Flag

Table continues on the next page...

PWM_SM*n*STS field descriptions (continued)

Field	Description
	This read/write flag is set at the beginning of every reload cycle regardless of the state of MCTRL[LDOK]. Clear this bit by writing a logic one to this location. Reset clears this bit. 0 No new reload cycle since last STS[RF] clearing 1 New reload cycle since last STS[RF] clearing
11–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–0 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 0 No compare event has occurred for a particular VALx value. 1 A compare event has occurred for a particular VALx value.

7.3.19 PWM SMx Interrupt Enable Register (PWM_SM*n*INTEN)

Addresses: PWM_SM0INTEN – F300h base + 13h offset = F313h
PWM_SM1INTEN – F300h base + 43h offset = F343h
PWM_SM2INTEN – F300h base + 73h offset = F373h

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

PWM_SM*n*INTEN field descriptions

Field	Description
15–14 Reserved	This read-only bitfield is reserved and always has the value zero.
13 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag, STS[REF], to generate CPU interrupt requests. Reset clears this bit. 0 STS[REF] CPU interrupt requests disabled 1 STS[REF] CPU interrupt requests enabled
12 RIE	Reload Interrupt Enable This read/write bit enables the reload flag, STS[RF], to generate CPU interrupt requests. Reset clears this bit.

Table continues on the next page...

PWM_SM*n*INTEN field descriptions (continued)

Field	Description
	0 STS[RF] CPU interrupt requests disabled 1 STS[RF] CPU interrupt requests enabled
11–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–0 CMPIE	Compare Interrupt Enables These bits enable the STS[CMPF] flags to cause a compare interrupt request to the CPU. 0 The corresponding STS[CMPF] bit will not cause an interrupt request. 1 The corresponding STS[CMPF] bit will cause an interrupt request.

7.3.20 PWM SMx Output Trigger Control Register (PWM_SM*n*TCTRL)

Addresses: PWM_SM0TCTRL – F300h base + 15h offset = F315h
PWM_SM1TCTRL – F300h base + 45h offset = F345h
PWM_SM2TCTRL – F300h base + 75h offset = F375h

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

PWM_SM*n*TCTRL field descriptions

Field	Description
15–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–0 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of OUT_TRIG0 and OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. VAL0, VAL2, and VAL4 are used to generate OUT_TRIG0 and VAL1, VAL3, and VAL5 are used to generate OUT_TRIG1. The OUT_TRIGx signals are only asserted as long as the counter value matches the VALx value, therefore up to six triggers can be generated (three each on OUT_TRIG0 and OUT_TRIG1) per PWM cycle per submodule. 0 OUT_TRIGx will not set when the counter value matches the VALx value. 1 OUT_TRIGx will set when the counter value matches the VALx value.

7.3.21 PWM SMx Fault Disable Mapping Register (PWM_SM*n*DISMAP)

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register.

Addresses: PWM_SM0DISMAP – F300h base + 16h offset = F316h
PWM_SM1DISMAP – F300h base + 46h offset = F346h
PWM_SM2DISMAP – F300h base + 76h offset = F376h

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

PWM_SM*n*DISMAP field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value one.
14 Reserved	This read-only bit is reserved and always has the value one.
13 Reserved	This read-only bit is reserved and always has the value one.
12 Reserved	This read-only bit is reserved and always has the value one.
11–8 DISX	PWMX Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMX output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7–4 DISB	PWMB Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMB output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
3–0 DISA	PWMA Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMA output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

7.3.22 PWM SMx Deadtime Count Register 0 (PWM_SM*n*DTCNT0)

Deadtime operation is only applicable to complementary channel operation. The 11-bit values written to these registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. These registers are not byte accessible.

Addresses: PWM_SM0DTCNT0 – F300h base + 17h offset = F317h
PWM_SM1DTCNT0 – F300h base + 47h offset = F347h
PWM_SM2DTCNT0 – F300h base + 77h offset = F377h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					DTCNT0										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

PWM_SM*n*DTCNT0 field descriptions

Field	Description
15–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10–0 DTCNT0	Deadtime Count Register 0 The DTCNT0 field is used to control the deadtime during 0 to 1 transitions of the PWMA output (assuming normal polarity).

7.3.23 PWM SMx Deadtime Count Register 1 (PWM_SM*n*DTCNT1)

Deadtime operation is only applicable to complementary channel operation. The 11-bit values written to these registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. These registers are not byte accessible.

Addresses: PWM_SM0DTCNT1 – F300h base + 18h offset = F318h
PWM_SM1DTCNT1 – F300h base + 48h offset = F348h
PWM_SM2DTCNT1 – F300h base + 78h offset = F378h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					DTCNT1										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

PWM_SM*n*DTCNT1 field descriptions

Field	Description
15–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10–0 DTCNT1	Deadtime Count Register 1 The DTCNT1 field is used to control the deadtime during 0 to 1 transitions of the complementary PWMB output.

7.3.24 PWM SM3 Counter Register (PWM_SM3CNT)

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible.

Address: PWM_SM3CNT – F300h base + 90h offset = F390h

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

PWM_SM3CNT field descriptions

Field	Description
15–0 CNT	Counter Register Bits

7.3.25 PWM SM3 Initial Count Register (PWM_SM3INIT)

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of CTRL2[INIT_SEL]) or when CTRL2[FORCE] is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

The INIT register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Address: PWM_SM3INIT – F300h base + 91h offset = F391h

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

PWM_SM3INIT field descriptions

Field	Description
15–0 INIT	Initial Count Register Bits

7.3.26 PWM SM3 Control 2 Register (PWM_SM3CTRL2)

Address: PWM_SM3CTRL2 – F300h base + 92h offset = F392h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
Write										FORCE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CTRL2 field descriptions

Field	Description
15 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in debug mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates will not occur in debug mode. Any motors requiring such updates should be disabled during debug mode. If in doubt, leave this bit set to zero.
14 WAITEN	WAIT Enable When set to one, the PWM will continue to run while the chip is in WAIT mode. In this mode, the peripheral clock continues to run but the CPU clock does not. If the device enters WAIT mode and this bit is zero, then the PWM outputs will be disabled until WAIT mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in WAIT mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in WAIT mode. The key point is PWM parameter updates will not occur in this mode. Any motors requiring such updates should be disabled during WAIT mode. If in doubt, leave this bit set to zero.
13 INDEP	Independent or Complementary Pair Operation This bit determines if the PWMA and PWMB channels will be independent PWMs or a complementary PWM pair. 0 PWMA and PWMB form a complementary PWM pair. 1 PWMA and PWMB outputs are independent PWMs.
12 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT.
11 PWM45_INIT	PWM45 Initial Value This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT.
10 PWMX_INIT	PWMX Initial Value This read/write bit determines the initial value for PWMX and the value to which it is forced when FORCE_INIT.
9–8 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00 Local sync (PWMX) causes initialization. 01 Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 10 Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 11 EXT_SYNC causes initialization.

Table continues on the next page...

PWM_SM3CTRL2 field descriptions (continued)

Field	Description
7 FRCEN	Force Initialization Enable This bit allows the CTRL2[FORCE] signal to initialize the counter without regard to the signal selected by CTRL2[INIT_SEL]. This is a software controlled initialization. 0 Initialization from a FORCE_OUT event is disabled. 1 Initialization from a FORCE_OUT event is enabled.
6 FORCE	Force Initialization If CTRL2[FORCE_SEL] is set to 000, writing a 1 to this bit results in a FORCE_OUT event. This causes the following actions to be taken: • The PWMA and PWMB output pins will assume values based on DTSRCSEL[SMxSEL23] and DTSRCSEL[SMxSEL45]. • If CTRL2[FRCEN] is set, the counter value will be initialized with the INIT register value.
5–3 FORCE_SEL	This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000 The local force signal, CTRL2[FORCE], from this submodule is used to force updates. 001 The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010 The local reload signal from this submodule is used to force updates. 011 The master reload signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100 The local sync signal from this submodule is used to force updates. 101 The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110 The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111 reserved
2 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, MCTRL[LDOK[0]] for submodule 0 should be used since the local MCTRL[LDOK] will be ignored. 0 The local RELOAD signal is used to reload registers. 1 The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.
1–0 CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00 The IPBus clock is used as the clock for the local prescaler and counter. 01 EXT_CLK is used as the clock for the local prescaler and counter. 10 Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11 reserved

7.3.27 PWM SM3 Control Register (PWM_SM3CTRL)

Address: PWM_SM3CTRL – F300h base + 93h offset = F393h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LDFQ				HALF	FULL	DT		0	PRSC			0	LDMOD	0	DBLEN
Write																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

PWM_SM3CTRL field descriptions

Field	Description
15–12 LDFQ	0000 Every PWM Opportunity 0001 Every 2 PWM Opportunities 0010 Every 3 PWM Opportunities 0011 Every 4 PWM Opportunities 0100 Every 5 PWM Opportunities 0101 Every 6 PWM Opportunities 0110 Every 7 PWM Opportunities 0111 Every 8 PWM Opportunities 1000 Every 9 PWM Opportunities 1001 Every 10 PWM Opportunities 1010 Every 11 PWM Opportunities 1011 Every 12 PWM Opportunities 1100 Every 13 PWM Opportunities 1101 Every 14 PWM Opportunities 1110 Every 15 PWM Opportunities 1111 Every 16 PWM Opportunities
11 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be half way through the PWM cycle. 0 Half-cycle reloads disabled. 1 Half-cycle reloads enabled.
10 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either CTRL[HALF] or CTRL[FULL] must be set in order to move the buffered data into the registers used by the PWM generators. If both CTRL[HALF] and CTRL[FULL] are set, then reloads can occur twice per cycle. 0 Full-cycle reloads disabled. 1 Full-cycle reloads enabled.
9–8 DT	Deadtime

Table continues on the next page...

PWM_SM3CTRL field descriptions (continued)

Field	Description
	These read only bits reflect the sampled values of the PWMX input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.
7 Reserved	This read-only bit is reserved and always has the value zero.
6–4 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CTRL2[CLK_SEL]. Reading CTRL[PRSC] reads the buffered values and not necessarily the values currently in effect. CTRL[PRSC] takes effect at the beginning of the next PWM cycle and only when the load okay bit, MCTRL[LDOK], is set or CTRL[LDMOD] is set. This field cannot be written when MCTRL[LDOK] is set. 000 PWM clock frequency = fclk 001 PWM clock frequency = fclk/2 010 PWM clock frequency = fclk/4 011 PWM clock frequency = fclk/8 100 PWM clock frequency = fclk/16 101 PWM clock frequency = fclk/32 110 PWM clock frequency = fclk/64 111 PWM clock frequency = fclk/128
3 Reserved	This read-only bit is reserved and always has the value zero.
2 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. 0 Buffered registers of this submodule are loaded and take effect at the next PWM reload if MCTRL[LDOK] is set. 1 Buffered registers of this submodule are loaded and take effect immediately upon MCTRL[LDOK] being set.
1 Reserved	This read-only bit is reserved and always has the value zero.
0 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior. 0 Double switching disabled. 1 Double switching enabled.

7.3.28 PWM SM3 Value Register 0 (PWM_SM3VAL0)

Address: PWM_SM3VAL0 – F300h base + 95h offset = F395h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3VAL0 field descriptions

Field	Description
15–0 VAL0	Value Register 0 The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWMX signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE:** The VAL0 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL0 cannot be written when MCTRL[LDOK] is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

7.3.29 PWM SM3 Value Register 1 (PWM_SM3VAL1)

Address: PWM_SM3VAL1 – F300h base + 97h offset = F397h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3VAL1 field descriptions

Field	Description
15–0 VAL1	Value Register 1 The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local sync signal while resetting PWMX. This register is not byte accessible. **NOTE:** The VAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL1 cannot be written when MCTRL[LDOK] is set. Reading VAL1 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

7.3.30 PWM SM3 Value Register 2 (PWM_SM3VAL2)

Address: PWM_SM3VAL2 – F300h base + 99h offset = F399h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3VAL2 field descriptions

Field	Description
15–0 VAL2	Value Register 2 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE:** The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL2 cannot be written when MCTRL[LDOK] is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.31 PWM SM3 Value Register 3 (PWM_SM3VAL3)

Address: PWM_SM3VAL3 – F300h base + 9Bh offset = F39Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3VAL3 field descriptions

Field	Description
15–0 VAL3	Value Register 3 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low. This register is not byte accessible. **NOTE:** The VAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL3 cannot be written when MCTRL[LDOK] is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.32 PWM SM3 Value Register 4 (PWM_SM3VAL4)

Address: PWM_SM3VAL4 – F300h base + 9Dh offset = F39Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3VAL4 field descriptions

Field	Description
15–0 VAL4	Value Register 4 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high. This register is not byte accessible. **NOTE:** The VAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL4 cannot be written when MCTRL[LDOK] is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.33 PWM SM3 Value Register 5 (PWM_SM3VAL5)

Address: PWM_SM3VAL5 – F300h base + 9Fh offset = F39Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	VAL5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3VAL5 field descriptions

Field	Description
15–0 VAL5	Value Register 5 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE:** The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL5 cannot be written when MCTRL[LDOK] is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

7.3.34 PWM SM3 Output Control Register (PWM_SM3OCTRL)

Address: PWM_SM3OCTRL – F300h base + A1h offset = F3A1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PWMA_IN	PWMB_IN	PWMX_IN	0		POLA	POLB	POLX	0		PWMAFS		PWMBFS		PWMXFS	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3OCTRL field descriptions

Field	Description
15 PWMA_IN	PWMA Input This read only bit shows the logic value currently being driven into the PWMA input.
14 PWMB_IN	PWMB Input This read only bit shows the logic value currently being driven into the PWMB input.
13 PWMX_IN	PWMX Input This read only bit shows the logic value currently being driven into the PWMX input.
12–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10 POLA	PWMA Output Polarity This bit inverts the PWMA output polarity. 0 PWMA output not inverted. A high level on the PWMA pin represents the "on" or "active" state. 1 PWMA output inverted. A low level on the PWMA pin represents the "on" or "active" state.
9 POLB	PWMB Output Polarity This bit inverts the PWMB output polarity. 0 PWMB output not inverted. A high level on the PWMB pin represents the "on" or "active" state. 1 PWMB output inverted. A low level on the PWMB pin represents the "on" or "active" state.
8 POLX	PWMX Output Polarity This bit inverts the PWMX output polarity. 0 PWMX output not inverted. A high level on the PWMX pin represents the "on" or "active" state. 1 PWMX output inverted. A low level on the PWMX pin represents the "on" or "active" state.
7–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–4 PWMAFS	PWMA Fault State

Table continues on the next page...

PWM_SM3OCTRL field descriptions (continued)

Field	Description
	These bits determine the fault state for the PWMA output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
3–2 PWMBFS	PWMB Fault State These bits determine the fault state for the PWMB output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
1–0 PWMXFS	PWMX Fault State These bits determine the fault state for the PWMX output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.

7.3.35 PWM SM3 Status Register (PWM_SM3STS)

Address: PWM_SM3STS – F300h base + A2h offset = F3A2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	CFX1	CFX0	CMPF					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3STS field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 RUF	Registers Updated Flag

Table continues on the next page...

PWM_SM3STS field descriptions (continued)

Field	Description
	This read only flag is set when one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers has been written resulting in non-coherent data in the set of double buffered registers. Clear this bit by a proper reload sequence consisting of a reload signal while MCTRL[LDOK] = 1. Reset clears this bit. 0 No register update has occurred since last reload. 1 At least one of the double buffered registers has been updated since the last reload.
13 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while MCTRL[LDOK] is 0 and the double buffered registers are in a non-coherent state (STS[RUF] = 1). Clear this bit by writing a logic one to this location. Reset clears this bit. 0 No reload error occurred. 1 Reload signal occurred with non-coherent data and MCTRL[LDOK] = 0.
12 RF	Reload Flag This read/write flag is set at the beginning of every reload cycle regardless of the state of MCTRL[LDOK]. Clear this bit by writing a logic one to this location. Reset clears this bit. 0 No new reload cycle since last STS[RF] clearing 1 New reload cycle since last STS[RF] clearing
11 CFA1	Capture Flag A1 This bit is set when a capture event occurs on the Capture A1 circuit. This bit is cleared by writing a one to this bit position. Reset clears this bit.
10 CFA0	Capture Flag A0 This bit is set when a capture event occurs on the Capture A0 circuit. This bit is cleared by writing a one to this bit position. Reset clears this bit.
9 CFB1	Capture Flag B1 This bit is set when a capture event occurs on the Capture B1 circuit. This bit is cleared by writing a one to this bit position. Reset clears this bit.
8 CFB0	Capture Flag B0 This bit is set when a capture event occurs on the Capture B0 circuit. This bit is cleared by writing a one to this bit position. Reset clears this bit.
7 CFX1	Capture Flag X1 This bit is set when a capture event occurs on the Capture X1 circuit. This bit is cleared by writing a one to this bit position. Reset clears this bit.
6 CFX0	Capture Flag X0 This bit is set when a capture event occurs on the Capture X0 circuit. This bit is cleared by writing a one to this bit position. Reset clears this bit.
5–0 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 0 No compare event has occurred for a particular VALx value. 1 A compare event has occurred for a particular VALx value.

7.3.36 PWM SM3 Interrupt Enable Register (PWM_SM3INTEN)

Address: PWM_SM3INTEN – F300h base + A3h offset = F3A3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	CX1IE	CX0IE	CMPIE					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3INTEN field descriptions

Field	Description
15–14 Reserved	This read-only bitfield is reserved and always has the value zero.
13 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag, STS[REF], to generate CPU interrupt requests. Reset clears this bit. 0 STS[REF] CPU interrupt requests disabled 1 STS[REF] CPU interrupt requests enabled
12 RIE	Reload Interrupt Enable This read/write bit enables the reload flag, STS[RF], to generate CPU interrupt requests. Reset clears this bit. 0 STS[RF] CPU interrupt requests disabled 1 STS[RF] CPU interrupt requests enabled
11 CA1IE	Capture A 1 Interrupt Enable This bit allows the STS[CFA1] flag to create an interrupt request to the CPU. 0 Interrupt request disabled for STS[CFA1]. 1 Interrupt request enabled for STS[CFA1].
10 CA0IE	Capture A 0 Interrupt Enable This bit allows the STS[CFA0] flag to create an interrupt request to the CPU. 0 Interrupt request disabled for STS[CFA0]. 1 Interrupt request enabled for STS[CFA0].
9 CB1IE	Capture B 1 Interrupt Enable This bit allows the STS[CFB1] flag to create an interrupt request to the CPU. 0 Interrupt request disabled for STS[CFB1]. 1 Interrupt request enabled for STS[CFB1].
8 CB0IE	Capture B 0 Interrupt Enable This bit allows the STS[CFB0] flag to create an interrupt request to the CPU.

Table continues on the next page...

PWM_SM3INTEN field descriptions (continued)

Field	Description
	0 Interrupt request disabled for STS[CFB0]. 1 Interrupt request enabled for STS[CFB0].
7 CX1IE	Capture X 1 Interrupt Enable This bit allows the STS[CFX1] flag to create an interrupt request to the CPU. 0 Interrupt request disabled for STS[CFX1]. 1 Interrupt request enabled for STS[CFX1].
6 CX0IE	Capture X 0 Interrupt Enable This bit allows the STS[CFX0] flag to create an interrupt request to the CPU. 0 Interrupt request disabled for STS[CFX0]. 1 Interrupt request enabled for STS[CFX0].
5–0 CMPIE	Compare Interrupt Enables These bits enable the STS[CMPF] flags to cause a compare interrupt request to the CPU. 0 The corresponding STS[CMPF] bit will not cause an interrupt request. 1 The corresponding STS[CMPF] bit will cause an interrupt request.

7.3.37 PWM SM3 Output Trigger Control Register (PWM_SM3TCTRL)

Address: PWM_SM3TCTRL – F300h base + A5h offset = F3A5h

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

PWM_SM3TCTRL field descriptions

Field	Description
15–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–0 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of OUT_TRIG0 and OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. VAL0, VAL2, and VAL4 are used to generate OUT_TRIG0 and VAL1, VAL3, and VAL5 are used to generate OUT_TRIG1. The OUT_TRIGx signals are only asserted as long as the counter value matches the VALx value, therefore up to six triggers can be generated (three each on OUT_TRIG0 and OUT_TRIG1) per PWM cycle per submodule. 0 OUT_TRIGx will not set when the counter value matches the VALx value. 1 OUT_TRIGx will set when the counter value matches the VALx value.

7.3.38 PWM SM3 Fault Disable Mapping Register (PWM_SM3DISMAP)

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register.

Address: PWM_SM3DISMAP – F300h base + A6h offset = F3A6h

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

PWM_SM3DISMAP field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value one.
14 Reserved	This read-only bit is reserved and always has the value one.
13 Reserved	This read-only bit is reserved and always has the value one.
12 Reserved	This read-only bit is reserved and always has the value one.
11–8 DISX	PWMX Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMX output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7–4 DISB	PWMB Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMB output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
3–0 DISA	PWMA Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMA output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

7.3.39 PWM SM3 Deadtime Count Register 0 (PWM_SM3DTCNT0)

Deadtime operation is only applicable to complementary channel operation. The 11-bit values written to these registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. These registers are not byte accessible.

Address: PWM_SM3DTCNT0 – F300h base + A7h offset = F3A7h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					DTCNT0										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

PWM_SM3DTCNT0 field descriptions

Field	Description
15–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10–0 DTCNT0	Deadtime Count Register 0 The DTCNT0 field is used to control the deadtime during 0 to 1 transitions of the PWMA output (assuming normal polarity).

7.3.40 PWM SM3 Deadtime Count Register 1 (PWM_SM3DTCNT1)

Deadtime operation is only applicable to complementary channel operation. The 11-bit values written to these registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. These registers are not byte accessible.

Address: PWM_SM3DTCNT1 – F300h base + A8h offset = F3A8h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					DTCNT1										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

PWM_SM3DTCNT1 field descriptions

Field	Description
15–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10–0 DTCNT1	Deadtime Count Register 1 The DTCNT1 field is used to control the deadtime during 0 to 1 transitions of the complementary PWMB output.

7.3.41 PWM SM3 Capture Control A Register (PWM_SM3CAPTCTRLA)

Address: PWM_SM3CAPTCTRLA – F300h base + AAh offset = F3AAh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CA1CNT			CA0CNT			CFAWM		EDGCNTA_EN	INP_SELA	EDGA1		EDGA0		ONESHOTA	ARMA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CAPTCTRLA field descriptions

Field	Description
15–13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1)
12–10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1)
9–8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1)
7 EDGCNTA_EN	Edge Counter A Enable This bit enables the edge counter which counts rising and falling edges on the PWMA input signal. 0 Edge counter disabled and held in reset 1 Edge counter enabled
6 INP_SELA	Input Select A This bit selects between the raw PWMA input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit.

Table continues on the next page...

PWM_SM3CAPTCTRLA field descriptions (continued)

Field	Description
	0 Raw PWMA input signal selected as source. 1 Output of edge counter/compare selected as source.NOTE:When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and CAPTCTRLA[EDGA1] fields to enable one or both of the capture registers.
5–4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
3–2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely.If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLA[ARMA] is cleared. No further captures will be performed until CAPTCTRLA[ARMA] is set again.If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 Input capture operation is disabled. 1 Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

7.3.42 PWM SM3 Capture Compare A Register (PWM_SM3CAPTCOMPA)

Address: PWM_SM3CAPTCOMPA – F300h base + ABh offset = F3ABh

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

PWM_SM3CAPTCOMPA field descriptions

Field	Description
15–8 EDGCNTA	Edge Counter A This read-only field contains the edge counter value for the PWMA input capture circuitry.
7–0 EDGCMPA	Edge Compare A This read/write field is the compare value associated with the edge counter for the PWMA input capture circuitry.

7.3.43 PWM SM3 Capture Control B Register (PWM_SM3CAPTCTRLB)

Address: PWM_SM3CAPTCTRLB – F300h base + ACh offset = F3ACh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CB1CNT			CB0CNT			CFBWM		EDGCNTB_EN	INP_SELB	EDGB1		EDGB0		ONESHOTB	ARMB
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CAPTCTRLB field descriptions

Field	Description
15–13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1)
12–10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1)
9–8 CFBWM	Capture B FIFOs Water Mark

Table continues on the next page...

PWM_SM3CAPTCTRLB field descriptions (continued)

Field	Description
	This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1)
7 EDGCNTB_EN	Edge Counter B Enable This bit enables the edge counter which counts rising and falling edges on the PWMB input signal. 0 Edge counter disabled and held in reset 1 Edge counter enabled
6 INP_SELB	Input Select B This bit selects between the raw PWMB input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0 Raw PWMB input signal selected as source. 1 Output of edge counter/compare selected as source.NOTE:When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields in order to enable one or both of the capture registers.
5–4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
3–2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely.If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLB[ARMB] is cleared. No further captures will be performed until CAPTCTRLB[ARMB] is set again.If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.

Table continues on the next page...

PWM_SM3CAPTCTRLB field descriptions (continued)

Field	Description
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 Input capture operation is disabled. 1 Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

7.3.44 PWM SM3 Capture Compare B Register (PWM_SM3CAPTCOMPB)

Address: PWM_SM3CAPTCOMPB – F300h base + ADh offset = F3ADh

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

PWM_SM3CAPTCOMPB field descriptions

Field	Description
15–8 EDGCNTB	Edge Counter B This read-only field contains the edge counter value for the PWMB input capture circuitry.
7–0 EDGCMPB	Edge Compare B This read/write field is the compare value associated with the edge counter for the PWMB input capture circuitry.

7.3.45 PWM SM3 Capture Control X Register (PWM_SM3CAPTCTRLX)

Address: PWM_SM3CAPTCTRLX – F300h base + AEh offset = F3AEh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CX1CNT			CX0CNT			CFXWM		EDGCNTX_EN	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CAPTCTRLX field descriptions

Field	Description
15–13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1)
12–10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1)
9–8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWMX input signal. 0 Edge counter disabled and held in reset 1 Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWMX input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0 Raw PWMX input signal selected as source. 1 Output of edge counter/compare selected as source.NOTE:When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields in order to enable one or both of the capture registers.
5–4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
3–2 EDGX0	Edge X 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
1 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed.

Table continues on the next page...

PWM_SM3CAPTCTRLX field descriptions (continued)

Field	Description
	The process continues indefinitely.If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again.If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 Input capture operation is disabled. 1 Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

7.3.46 PWM SM3 Capture Compare X Register (PWM_SM3CAPTCOMPX)

Address: PWM_SM3CAPTCOMPX – F300h base + AFh offset = F3AFh

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

PWM_SM3CAPTCOMPX field descriptions

Field	Description
15–8 EDGCNTX	Edge Counter X This read-only field contains the edge counter value for the PWMX input capture circuitry.
7–0 EDGCMPX	Edge Compare X This read/write field is the compare value associated with the edge counter for the PWMX input capture circuitry.

7.3.47 PWM SM3 Capture Value 0 Register (PWM_SM3CVAL0)

Address: PWM_SM3CVAL0 – F300h base + B0h offset = F3B0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL0															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CVAL0 field descriptions

Field	Description
15–0 CAPTVAL0	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. This register is not byte accessible.

7.3.48 PWM SM3 Capture Value 1 Register (PWM_SM3CVAL1)

Address: PWM_SM3CVAL1 – F300h base + B2h offset = F3B2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CVAL1 field descriptions

Field	Description
15–0 CAPTVAL1	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX1]. This register is not byte accessible.

7.3.49 PWM SM3 Capture Value 2 Register (PWM_SM3CVAL2)

Address: PWM_SM3CVAL2 – F300h base + B4h offset = F3B4h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CVAL2 field descriptions

Field	Description
15–0 CAPTVAL2	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA0]. This register is not byte accessible.

7.3.50 PWM SM3 Capture Value 3 Register (PWM_SM3CVAL3)

Address: PWM_SM3CVAL3 – F300h base + B6h offset = F3B6h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL3															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CVAL3 field descriptions

Field	Description
15–0 CAPTVAL3	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA1]. This register is not byte accessible.

7.3.51 PWM SM3 Capture Value 4 Register (PWM_SM3CVAL4)

Address: PWM_SM3CVAL4 – F300h base + B8h offset = F3B8h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL4															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CVAL4 field descriptions

Field	Description
15–0 CAPTVAL4	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB0]. This register is not byte accessible.

7.3.52 PWM SM3 Capture Value 5 Register (PWM_SM3CVAL5)

Address: PWM_SM3CVAL5 – F300h base + BAh offset = F3BAh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTVAL5															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SM3CVAL5 field descriptions

Field	Description
15–0 CAPTVAL5	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB1]. This register is not byte accessible.

7.3.53 Output Enable Register (PWM_OUTEN)

Address: PWM_OUTEN – F300h base + C0h offset = F3C0h

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

PWM_OUTEN field descriptions

Field	Description
15–12 Reserved	This read-only bitfield is reserved and always has the value zero.
11–8 PWMA_EN	PWMA Output Enables These bits enable the PWMA outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMA pin is being used for input capture. 0 PWMA output disabled. 1 PWMA output enabled.
7–4 PWMB_EN	PWMB Output Enables These bits enable the PWMB outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMB pin is being used for input capture. 0 PWMB output disabled. 1 PWMB output enabled.

Table continues on the next page...

PWM_OUTEN field descriptions (continued)

Field	Description
3–0 PWMX_EN	PWMX_EN - PWMX Output Enables These bits enable the PWMX outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMX pin is being used for input capture or deadtime correction. 0 PWMX output disabled. 1 PWMX output enabled.

7.3.54 Mask Register (PWM_MASK)

MASK is double buffered and does not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading MASK reads the buffered values and not necessarily the values currently in effect.

Address: PWM_MASK – F300h base + C1h offset = F3C1h

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

PWM_MASK field descriptions

Field	Description
15–12 Reserved	This read-only bitfield is reserved and always has the value zero.
11–8 MASKA	PWMA Masks These bits mask the PWMA outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity. 0 PWA output normal. 1 PWA output masked.
7–4 MASKB	PWMB Masks These bits mask the PWMB outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity. 0 PWMB output normal. 1 PWMB output masked.
3–0 MASKX	PWMX Masks These bits mask the PWMX outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity. 0 PWM output normal. 1 PWM output masked.

7.3.55 Software Controlled Output Register (PWM_SWCOUT)

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Address: PWM_SWCOUT – F300h base + C2h offset = F3C2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SM3OUT23	SM3OUT45	SM2OUT23	SM2OUT45	SM1OUT23	SM1OUT45	SM0OUT23	SM0OUT45
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_SWCOUT field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 SM3OUT23	Submodule 3 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM3SEL23] is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
6 SM3OUT45	Submodule 3 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM3SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
5 SM2OUT23	Submodule 2 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM2SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.
4 SM2OUT45	Submodule 2 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM2SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.

Table continues on the next page...

PWM_SWCOUT field descriptions (continued)

Field	Description
3 SM1OUT23	Submodule 1 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM1SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
2 SM1OUT45	Submodule 1 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM1SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
1 SM0OUT23	Submodule 0 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM0SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
0 SM0OUT45	Submodule 0 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM0SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

7.3.56 Deadtime Source Select Register (PWM_DTSRCSEL)

The deadtime source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading the these bits reads the buffered value and not necessarily the value currently in effect.

Address: PWM_DTSRCSEL – F300h base + C3h offset = F3C3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SM3SEL23		SM3SEL45		SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23		SM0SEL45	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM_DTSRCSEL field descriptions

Field	Description
15–14 SM3SEL23	Submodule 3 PWM23 Control Select

Table continues on the next page...

PWM_DTSRCSEL field descriptions (continued)

Field	Description
	This field selects possible over-rides to the generated SM3PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM3PWM23 signal is used by the deadtime logic. 01 Inverted generated SM3PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM3OUT23] is used by the deadtime logic. 11 PWM3_EXTA signal is used by the deadtime logic.
13–12 SM3SEL45	Submodule 3 PWM45 Control Select This field selects possible over-rides to the generated SM3PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM3PWM45 signal is used by the deadtime logic. 01 Inverted generated SM3PWM45 signal is used by the deadtime logic. 10 SWCOUT[SM3OUT45] is used by the deadtime logic. 11 PWM3_EXTB signal is used by the deadtime logic.
11–10 SM2SEL23	Submodule 2 PWM23 Control Select This field selects possible over-rides to the generated SM2PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM2PWM23 signal is used by the deadtime logic. 01 Inverted generated SM2PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM2OUT23] is used by the deadtime logic. 11 PWM2_EXTA signal is used by the deadtime logic.
9–8 SM2SEL45	Submodule 2 PWM45 Control Select This field selects possible over-rides to the generated SM2PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that sudeadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM2PWM45 signal is used by the deadtime logic. 01 Inverted generated SM2PWM45 signal is used by the deadtime logic. 10 SWCOUT[SM2OUT45] is used by the deadtime logic. 11 PWM2_EXTB signal is used by the deadtime logic.
7–6 SM1SEL23	Submodule 1 PWM23 Control Select This field selects possible over-rides to the generated SM1PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM1PWM23 signal is used by the deadtime logic. 01 Inverted generated SM1PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM1OUT23] is used by the deadtime logic. 11 PWM1_EXTA signal is used by the deadtime logic.
5–4 SM1SEL45	Submodule 1 PWM45 Control Select This field selects possible over-rides to the generated SM1PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM1PWM45 signal is used by the deadtime logic. 01 Inverted generated SM1PWM45 signal is used by the deadtime logic.

Table continues on the next page...

PWM_DTSRCSEL field descriptions (continued)

Field	Description
	10 SWCOUT[SM1OUT45] is used by the deadtime logic. 11 PWM1_EXTB signal is used by the deadtime logic.
3–2 SM0SEL23	Submodule 0 PWM23 Control Select This field selects possible over-rides to the generated SM0PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM0PWM23 signal is used by the deadtime logic. 01 Inverted generated SM0PWM23 signal is used by the deadtime logic. 10 SWCOUT[SM0OUT23] is used by the deadtime logic. 11 PWM0_EXTA signal is used by the deadtime logic.
1–0 SM0SEL45	Submodule 0 PWM45 Control Select This field selects possible over-rides to the generated SM0PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 Generated SM0PWM45 signal is used by the deadtime logic. 01 Inverted generated SM0PWM45 signal is used by the deadtime logic. 10 SWCOUT[SM0OUT45] is used by the deadtime logic. 11 PWM0_EXTB signal is used by the deadtime logic.

7.3.57 Master Control Register (PWM_MCTRL)

Address: PWM_MCTRL – F300h base + C4h offset = F3C4h

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

PWM_MCTRL field descriptions

Field	Description
15–12 IPOL	Current Polarity This buffered read/write bit is used to select between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output. MCTRL[IPOL] is ignored in independent mode. MCTRL[IPOL] does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading MCTRL[IPOL] reads the buffered value and not necessarily the value currently in effect. 0 PWM23 is used to generate complementary PWM pair. 1 PWM45 is used to generate complementary PWM pair.
11–8 RUN	Run This read/write bit enables the clocks to the PWM generator. When this bit equals zero, the submodule counter is reset. A reset clears this field.

Table continues on the next page...

PWM_MCTRL field descriptions (continued)

Field	Description
	NOTE: For proper initialization of MCTRL[LDOK] and MCTRL[RUN], see the description of PWM initialization. 0 PWM generator disabled. 1 PWM generator enabled.
7–4 CLDOK	Clear Load Okay This write only bit is used to clear MCTRL[LDOK]. Write a 1 to this location to clear the corresponding MCTRL[LDOK]. If a reload occurs with MCTRL[LDOK] set at the same time that MCTRL[CLDOK] is written, then the reload will not be performed and MCTRL[LDOK] will be cleared. This bit is self clearing and always reads as a 0.
3–0 LDOK	Load Okay This read/set field loads CTRL[PRSC] and the INIT, FRACVALx, and VALx registers of the corresponding submodule into a set of buffers. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if CTRL[LDMOD] is clear or immediately if CTRL[LDMOD] is set. Set MCTRL[LDOK] by reading it when it is logic zero and then writing a logic one to it. The VALx, FRACVALx, INIT, and CTRL[PRSC] registers of the corresponding submodule cannot be written while MCTRL[LDOK] is set. MCTRL[LDOK] is automatically cleared after the new values are loaded, or can be manually cleared before a reload by writing a logic 1 to MCTRL[CLDOK]. This bit cannot be written with a zero. Reset clears this field. **NOTE:** For proper initialization of MCTRL[LDOK] and MCTRL[RUN], see the description of PWM initialization. 0 Do not load new values. 1 Load prescaler, modulus, and PWM values.

7.3.58 Master Control 2 Register (PWM_MCTRL2)

Address: PWM_MCTRL2 – F300h base + C5h offset = F3C5h

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

PWM_MCTRL2 field descriptions

Field	Description
15–2 Reserved	This read-only bitfield is reserved and always has the value zero.
1–0 MONPLL	Monitor PLL State These bits are used to control disabling of the fractional delay block when the chip PLL is unlocked and/or missing its input reference. The fractional delay block requires a continuous 120 MHz clock from the PLL. If this clock turns off when the fractional delay block is being used, then the outputs of the fractional delay block can be stuck high or low even if the PLL restarts. When this control bit is set, PLL problems cause

Table continues on the next page...

PWM_MCTRL2 field descriptions (continued)

Field	Description
	the fractional delay block to be disabled until the PLL returns to a locked state. When the PLL is receiving a proper reference and is locked, the fractional delay block requires a 25 µs startup time, just as if the FRCTRL[FRAC_PU] bits (one per submodule) had been turned off and turned on again. If PLL monitoring is disabled, software should manually clear and then set the FRCTRL[FRAC_PU] bits when the PLL loses its reference or becomes unlocked. This sequence causes the fractional delay block to be disabled and restarted. If the fractional delay block is not being used, the value of these bits does not matter. 00 Not locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. 01 Not locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. 10 Locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. These bits are write protected until the next reset. 11 Locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. These bits are write protected until the next reset.

7.3.59 Fault Control Register (PWM_FCTRL)

Address: PWM_FCTRL – F300h base + C6h offset = F3C6h

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

PWM_FCTRL field descriptions

Field	Description
15–12 FLVL	Fault Level These read/write bits select the active logic level of the individual fault inputs. A reset clears this field. 0 A logic 0 on the fault input indicates a fault condition. 1 A logic 1 on the fault input indicates a fault condition.
11–8 FAUTO	Automatic Fault Clearing These read/write bits select automatic or manual clearing of faults. A reset clears this field. 0 Manual fault clearing. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending the state of FSTS[FFULL]. This is further controlled by FCTRL[FSAFE]. 1 Automatic fault clearing. PWM outputs disabled by this fault are enabled when FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the state of FSTS[FFULL] without regard to the state of FSTS[FFLAGx].

Table continues on the next page...

PWM_FCTRL field descriptions (continued)

Field	Description
7–4 FSAFE	Fault Safety Mode These read/write bits select the safety mode during manual fault clearing. A reset clears this field. FSTS[FFPINx] may indicate a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency. 0 Normal mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the state of FSTS[FFULL] without regard to the state of FSTS[FFPINx]. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAP). 1 Safe mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear and FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the state of FSTS[FFULL].
3–0 FIE	Fault Interrupt Enables This read/write bit enables CPU interrupt requests generated by the FAULTx pins. A reset clears this field. 0 FAULTx CPU interrupt requests disabled. 1 FAULTx CPU interrupt requests enabled.

7.3.60 Fault Status Register (PWM_FSTS)

Address: PWM_FSTS – F300h base + C7h offset = F3C7h

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

PWM_FSTS field descriptions

Field	Description
15–13 Reserved	This read-only bitfield is reserved and always has the value zero.
12 FTEST	Fault Test These read/write bit is used to simulate a fault condition. Setting this bit will cause a simulated fault to be sent into all of the fault filters. The condition will propagate to the fault flags and possibly the PWM outputs depending on the DISMAP settings. Clearing this bit removes the simulated fault condition. 0 No fault. 1 Cause a simulated fault.
11–8 FFPIN	Filtered Fault Pins These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULTx pin. A reset has no effect on this field.

Table continues on the next page...

PWM_FSTS field descriptions (continued)

Field	Description
7–4 FFULL	Full Cycle These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. 0 PWM outputs are re-enabled at the start of a full or half cycle. 1 PWM outputs are re-enabled only at the start of a full cycle.
3–0 FFLAG	Fault Flags These read-only flag is set within two CPU cycles after a transition to active on the FAULTx pin. Clear this bit by writing a logic one to it. A reset clears this field. 0 No fault on the FAULTx pin. 1 Fault on the FAULTx pin.

7.3.61 Fault Filter Register (PWM_FFILT)

Address: PWM_FFILT – F300h base + C8h offset = F3C8h

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

PWM_FFILT field descriptions

Field	Description
15 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0 Fault input glitch stretching is disabled. 1 Input fault signals will be stretched to at least 2 IPBus clock cycles.
14–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10–8 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FFILT[FILT_CNT] affects the input latency
7–0 FILT_PER	Fault Filter Period

Table continues on the next page...

PWM_FFILT field descriptions (continued)

Field	Description
	These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If FFILT[FILT_PER] is 0x00 (default), then the input filter is bypassed. The value of FFILT[FILT_PER] affects the input latency.

7.4 Functional Description

7.4.1 PWM Capabilities

This section describes some capabilities of the PWM module.

7.4.1.1 Center Aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in Figure 7-208.

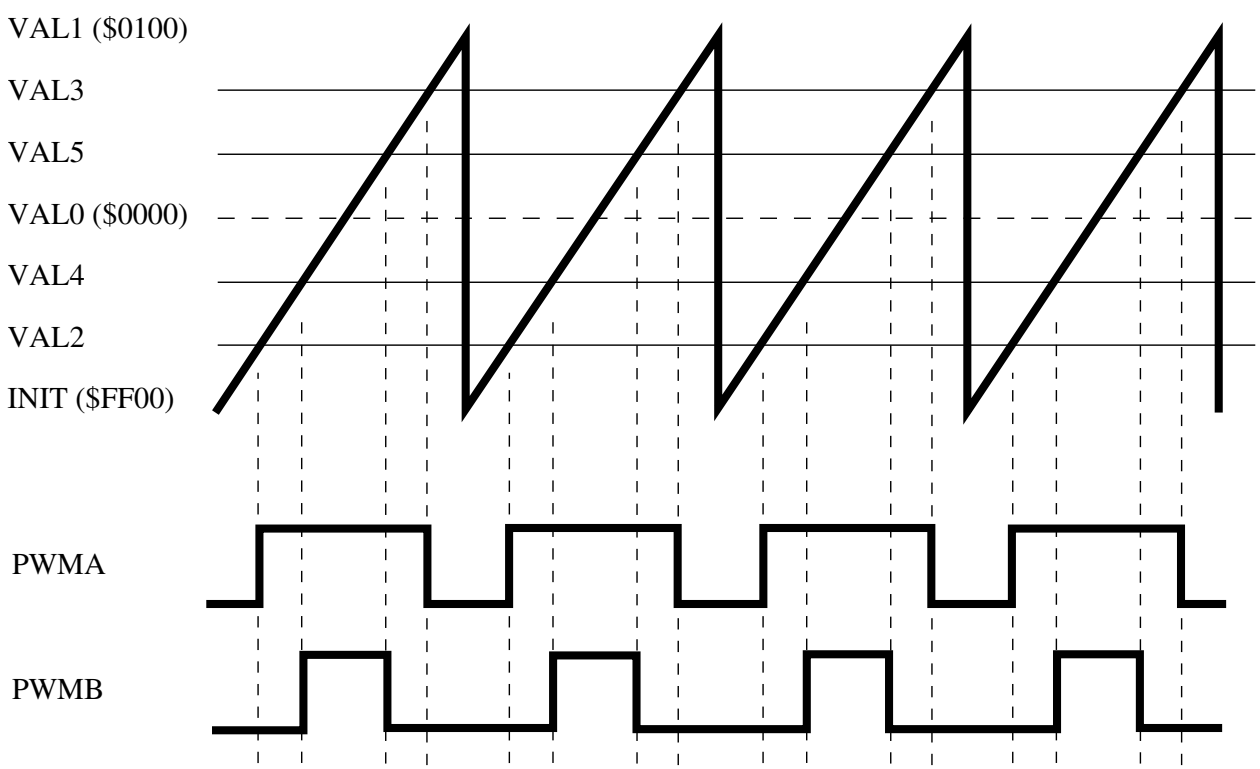


Figure 7-208. Center Aligned Example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn on edge and the turn off edge. This double action edge generation not only gives the user control over the pulse width, but over the relative alignment of the signal as well. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.

Figure 7-208 also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means that if each PWM's turn on and turn off edge values are also the same number but only different in their sign, the "on" portion of the output signal will be centered around a count value of zero. Therefore, only one PWM value needs to be calculated in software and then this value and its negative are provided to the submodule as the turn off and turn on edges respectively. This technique will result in a pulse width that always consists of an odd number of timer counts. If all PWM signal edge calculations follow this same convention, then the signals will be center aligned with respect to each other, which is the goal. Of course, center alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

7.4.1.2 Edge Aligned PWMs

When the turn on edge for each pulse is specified to be the INIT value, then edge aligned operation results, as the following figure shows. Therefore, only the turn off edge value needs to be periodically updated to change the pulse width.

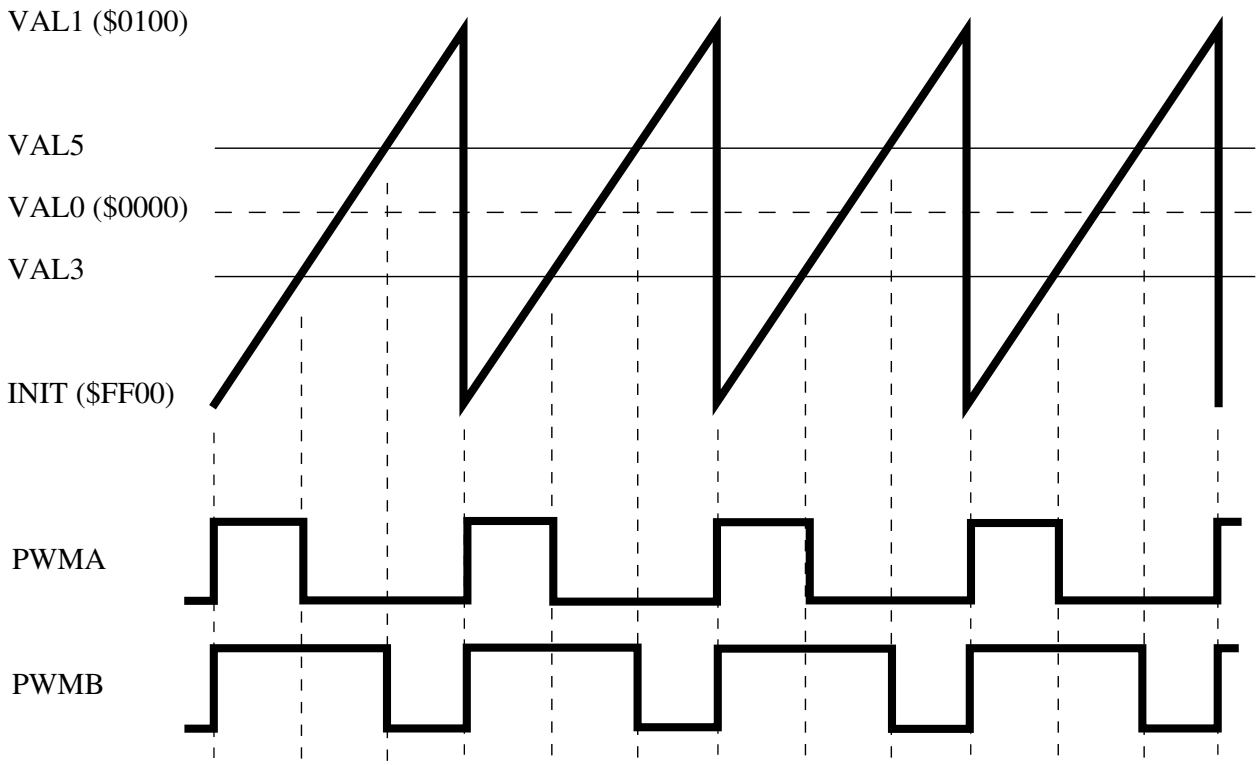


Figure 7-209. Edge Aligned Example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. A common way to drive an H-bridge is to use a technique called "bipolar" PWMs where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% result in negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn off edge value and the motor voltage, INCLUDING the sign. So once again, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

7.4.1.3 Phase Shifted PWMs

In the previous sections, the benefits of signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases are applied to the turn on and turn off edges of different PWM signal, the signals will be phase shifted with respect to each other, as the following figure shows. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from

the different phases occur at nearly the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does NOT affect the duty cycle so average load voltage is not affected.

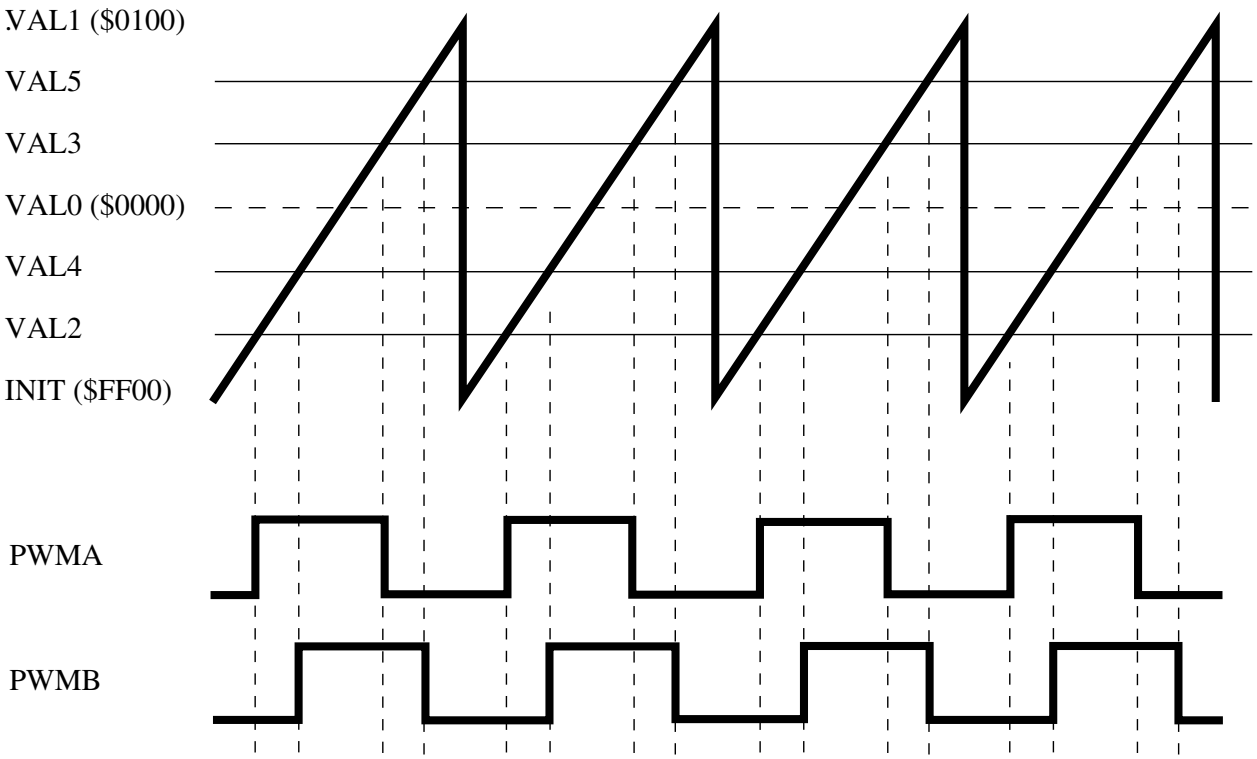


Figure 7-210. Phase Shifted Outputs Example

An additional benefit of phase shifted PWMs appears in Figure 7-211. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to always generate a square wave with 50% duty cycle. This works out nicely for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. Notice that the square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% making this technique ideally suited for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

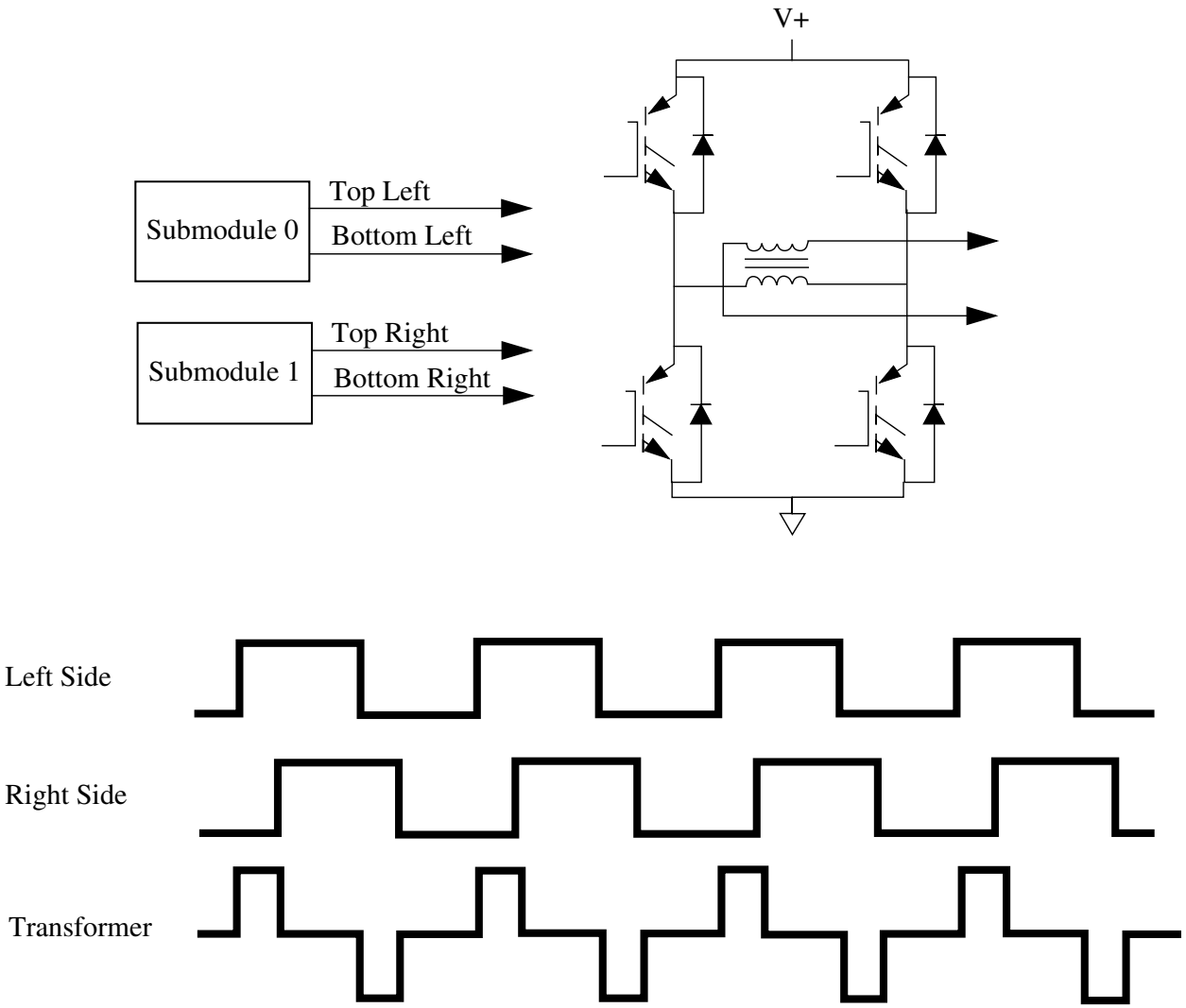


Figure 7-211. Phase Shifted PWMs Applied to a Transformer Primary

7.4.1.4 Double Switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction. This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labelled as PWMA in the figure) while VAL4 and VAL5 are used to generate the odd channel. The two channels are combined using XOR logic (force out logic) as the following figure shows. The DBLPWM signal can be run through the deadtime insertion logic.

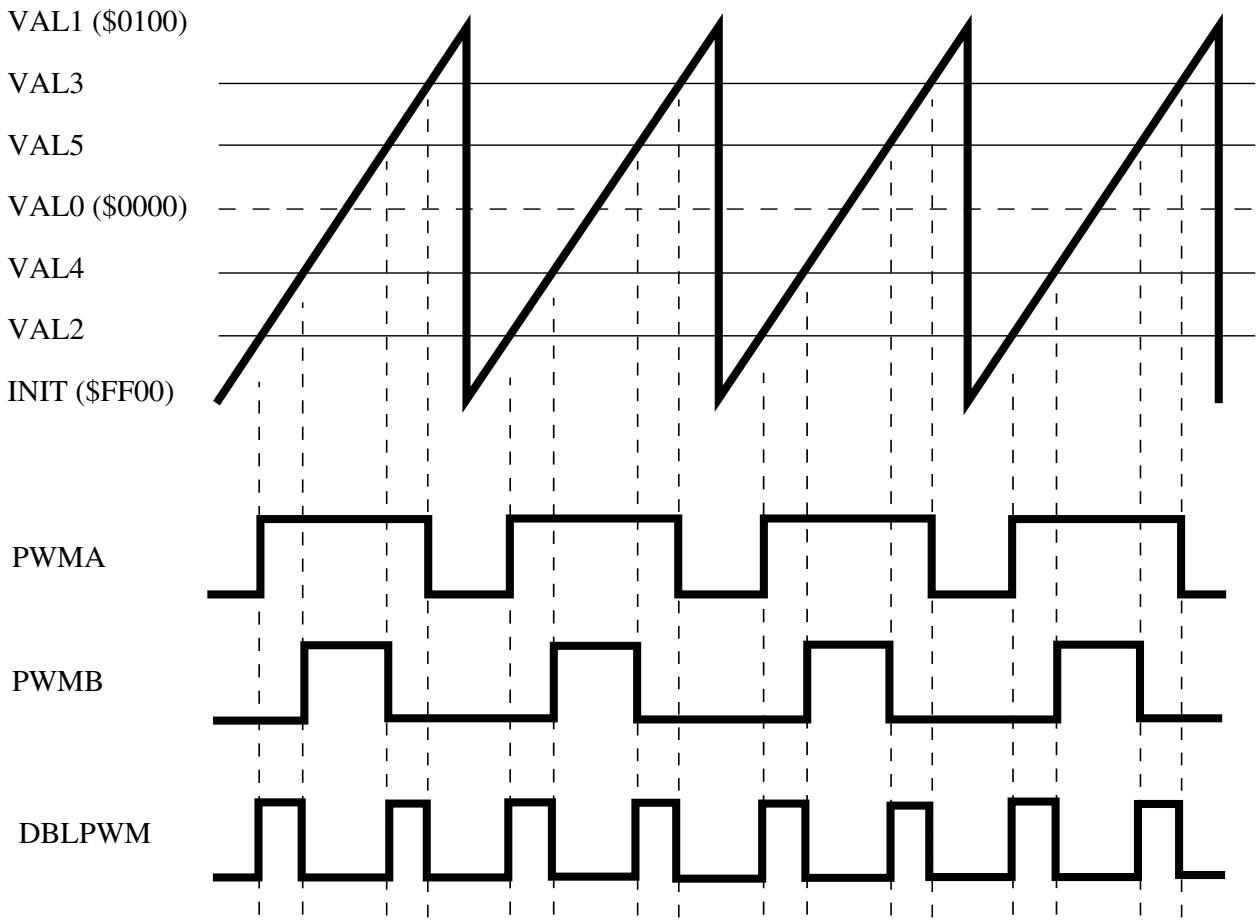


Figure 7-212. Double Switching Output Example

7.4.1.5 ADC Triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. Figure 7-213 shows how this is accomplished. When specifying complementary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators are free to perform other functions. In this example, the software does not need to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

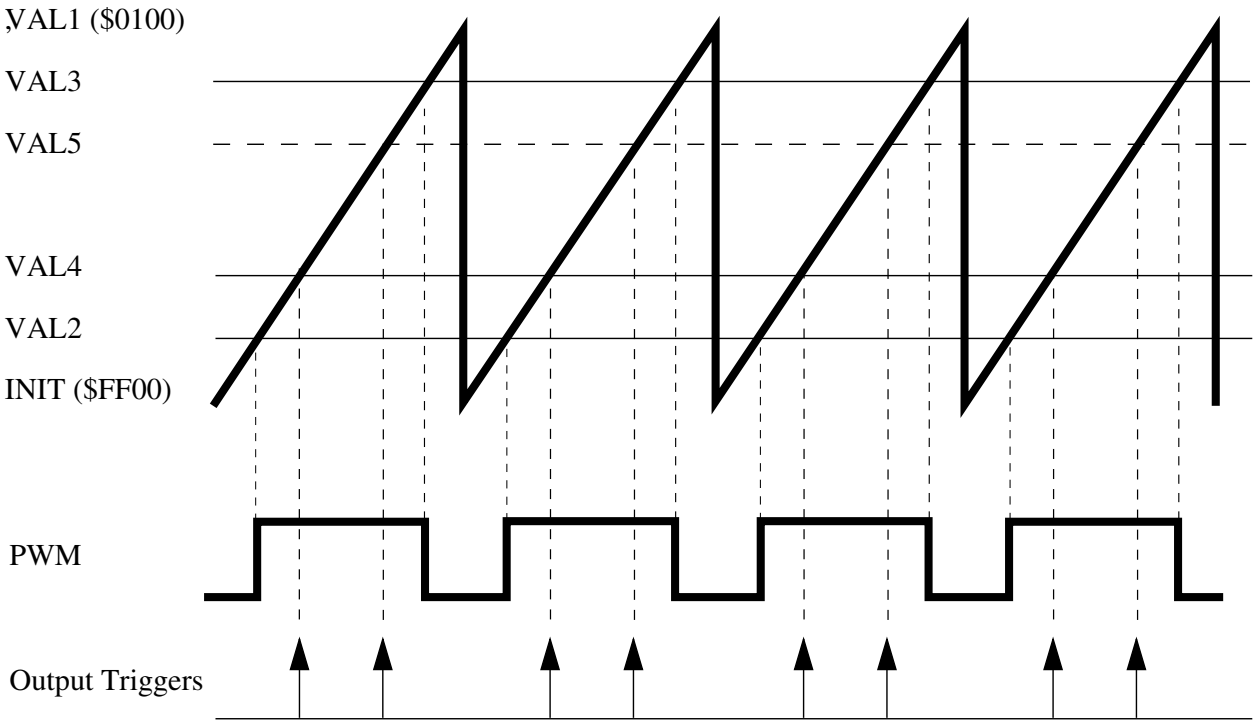


Figure 7-213. Multiple Output Trigger Generation in Hardware

Because each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. Figure 7-214 shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. A suggested use for this configuration would be to use the lower-frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In Figure 7-214 , *all* submodule comparators are shown being used for ADC trigger generation.

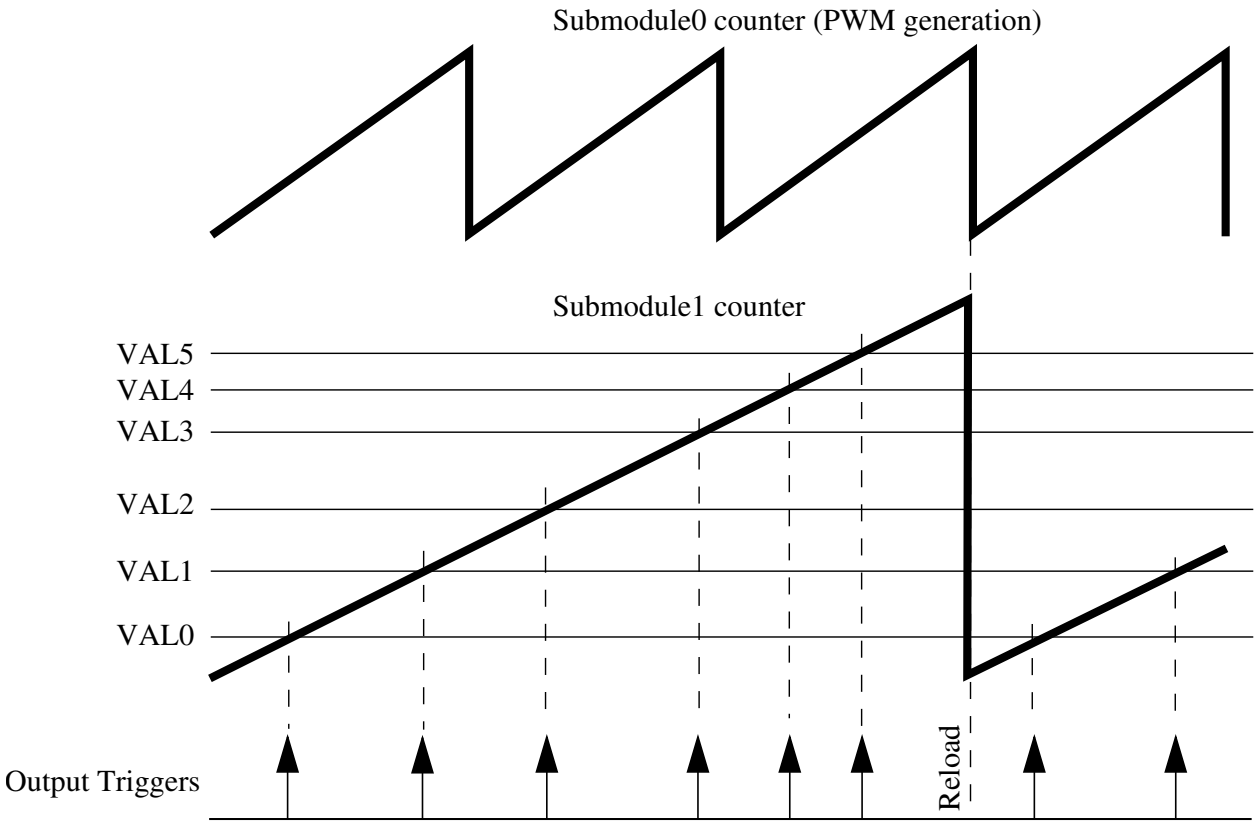


Figure 7-214. Multiple Output Triggers Over Several PWM Cycles

7.4.1.6 Enhanced Capture Capabilities (E-Capture)

When a PWM pin is not being used for PWM generation, it can be used to perform input captures. Recall that for PWM generation BOTH edges of the PWM signal are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. By simply programming the desired edge of each capture circuit, period and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8 bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

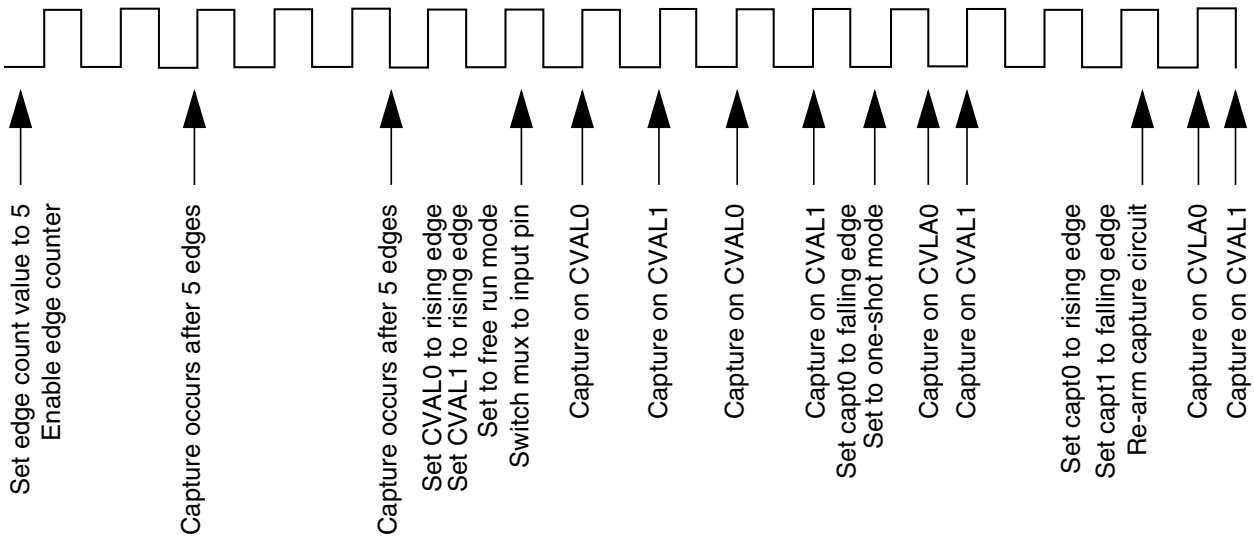


Figure 7-215. Capture Capabilities of the E-Capture Circuit

When a submodule is being used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (e.g, PWMX) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16 bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is perfectly suited for the application. Consider the following figure as an example. In this application the output of a PWM power stage is connected to the PWMX pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This will result in new load pulse width data being acquired every PWM cycle. To calculate the pulse width, simply subtract the CVAL0 register value from the CVAL1 register value. This measurement is extremely beneficial when performing dead-time distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays. For details, refer to the separate discussion of deadtime distortion correction.

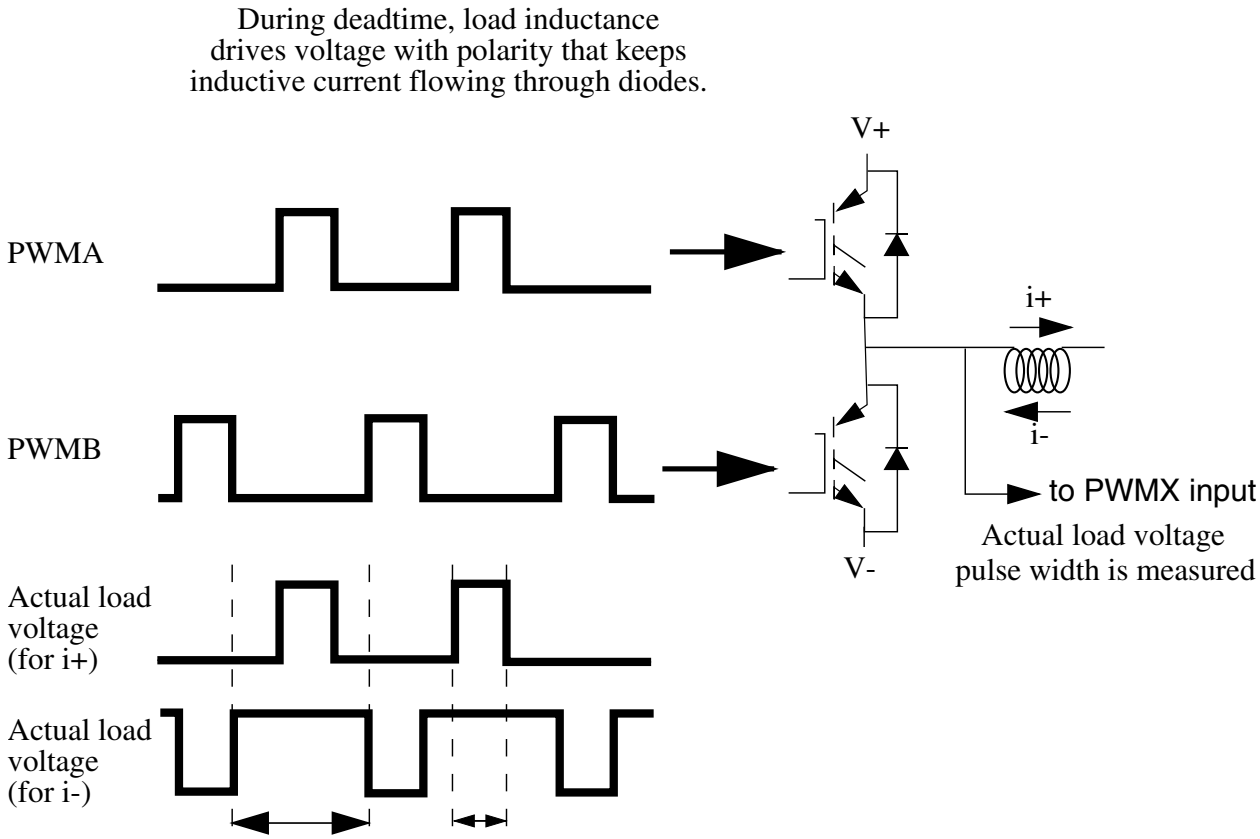


Figure 7-216. Output Pulse Width Measurement Possible with the E-Capture Circuit

7.4.1.7 Synchronous Switching of Multiple Outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature can be extremely useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. Not only do all the changes occur synchronously on all submodule outputs, but they occur IMMEDIATELY after the trigger event occurs eliminating any interrupt latency.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module and, in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are

switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that might occur do not violate deadtime on the power stage when in complementary mode.

Figure 7-217 shows a popular application that can benefit from this feature. On a brushless DC motor it is desirable on many cases to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of Figure 7-217 are a simplistic representation of these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

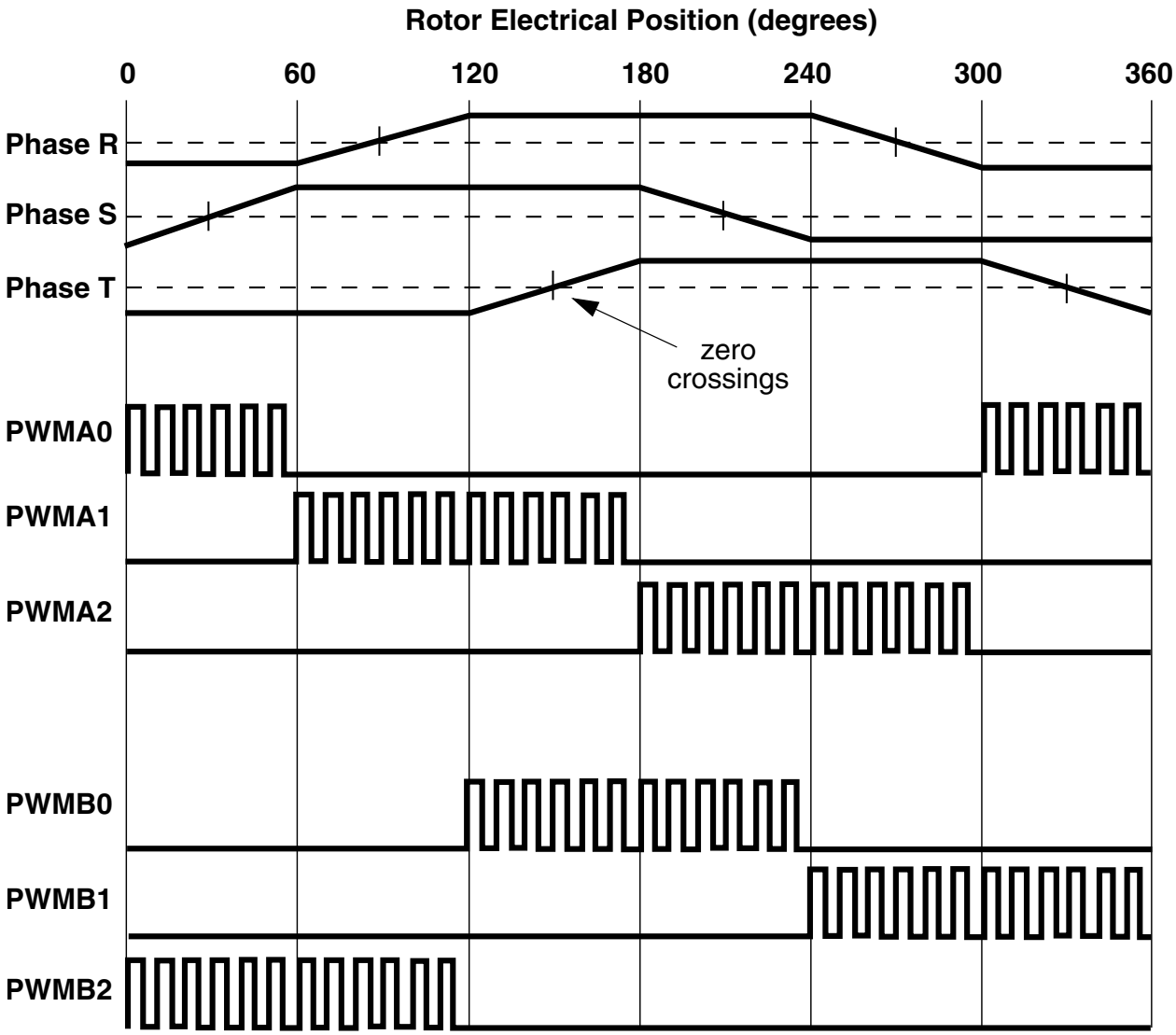


Figure 7-217. Sensorless BLDC Commutation Using the Force Out Function

7.4.2 Functional Details

This section describes the implementation of various sections of the PWM in greater detail.

7.4.2.1 PWM Clocking

Figure 7-218 shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the IPBus clock, EXT_CLK, and AUX_CLK. The EXT_CLK is generated by an on-chip resource such as a Timer module and goes to all of

the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8-bit prescaler and MCTRL[RUN] from submodule0 can control all of the submodules.

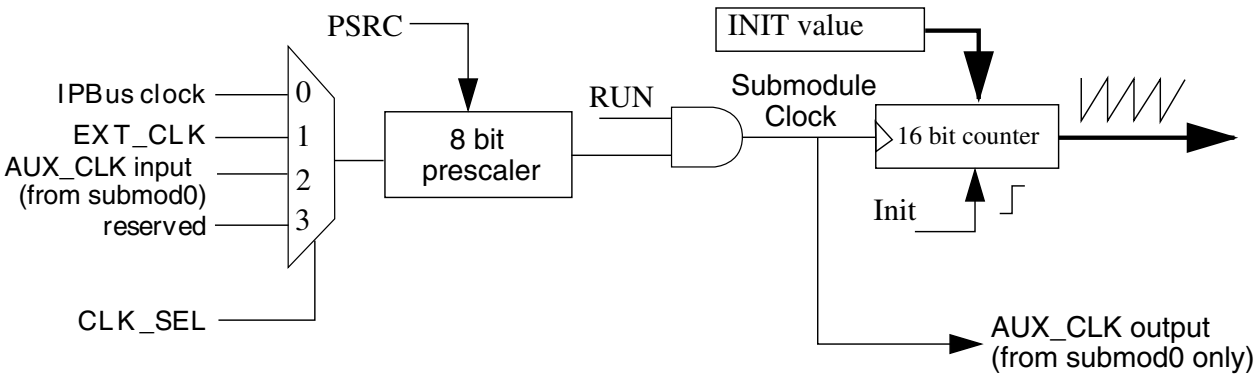


Figure 7-218. Clocking Block Diagram for Each PWM Submodule

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by 1-128. The prescaler bits, CTRL[PRSC], select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until MCTRL[LDOK] is set and a new PWM reload cycle begins or CTRL[LDMOD] is set.

7.4.2.2 Register Reload Logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using CTRL[LDFQ] and CTRL[FULL]. A half cycle reload option is also supported (CTRL[HALF]) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As illustrated in Figure 7-219 the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

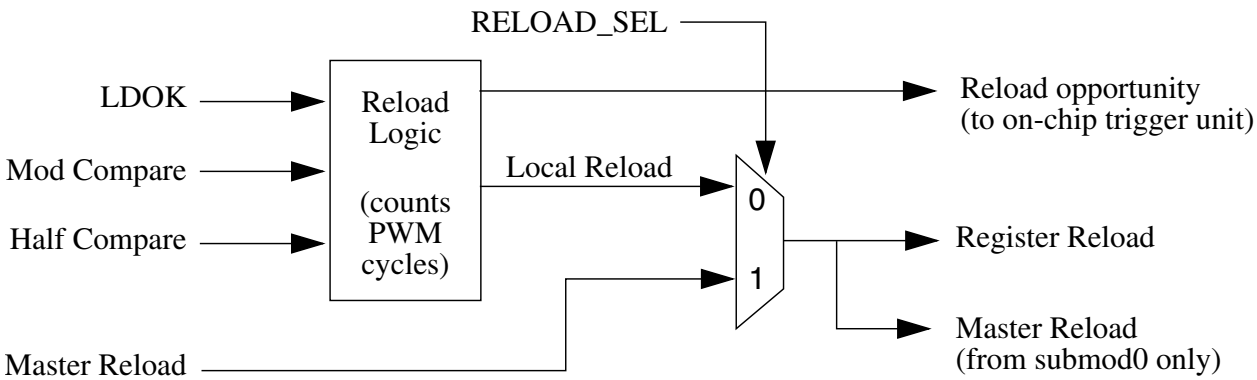


Figure 7-219. Register Reload Logic

7.4.2.3 Counter Synchronization

In the following figure, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

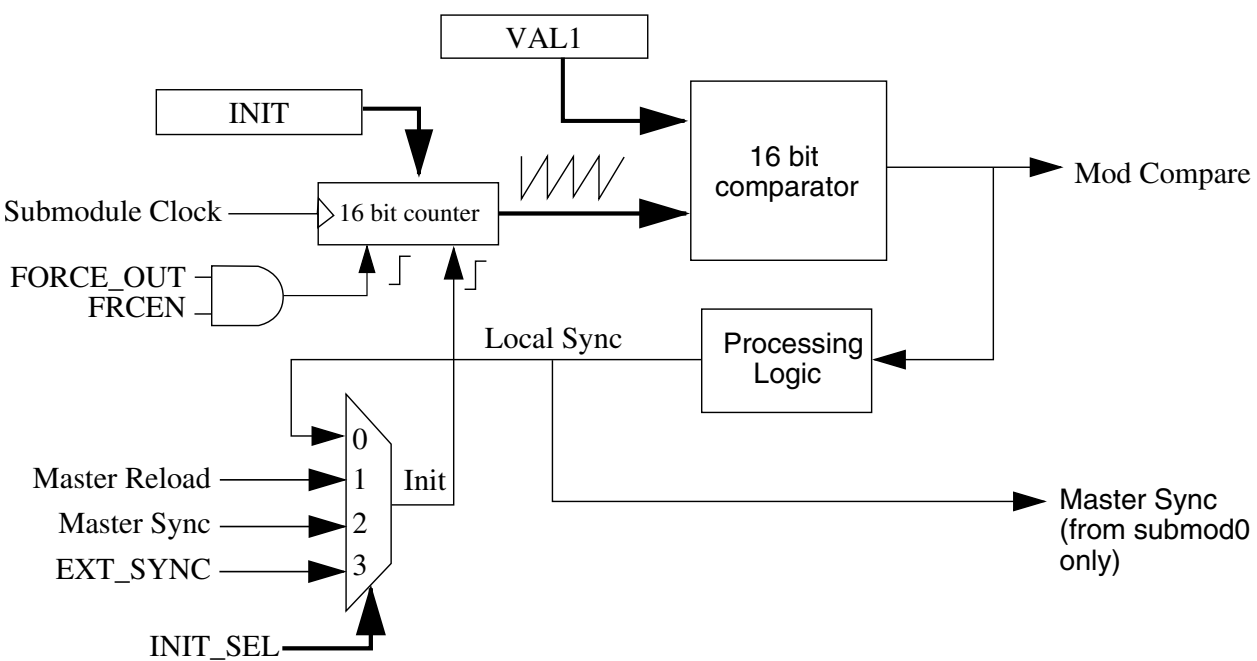


Figure 7-220. Submodule Timer Synchronization

The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0. The VAL1 register and associated comparator of the other submodules can then be freed up for other functions such as PWM generation, input captures, output compares, or output triggers.

The EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that CTRL2[FRCEN] is set. As indicated by the preceding figure, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal. The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event with respect to the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result will be an oscillation caused by the beating between the PWM frequency and the commutation frequency.

7.4.2.4 PWM Generation

Figure 7-221 illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register are used to control the turn-on edge, while a second comparator and VALx register control the turn-off edge.

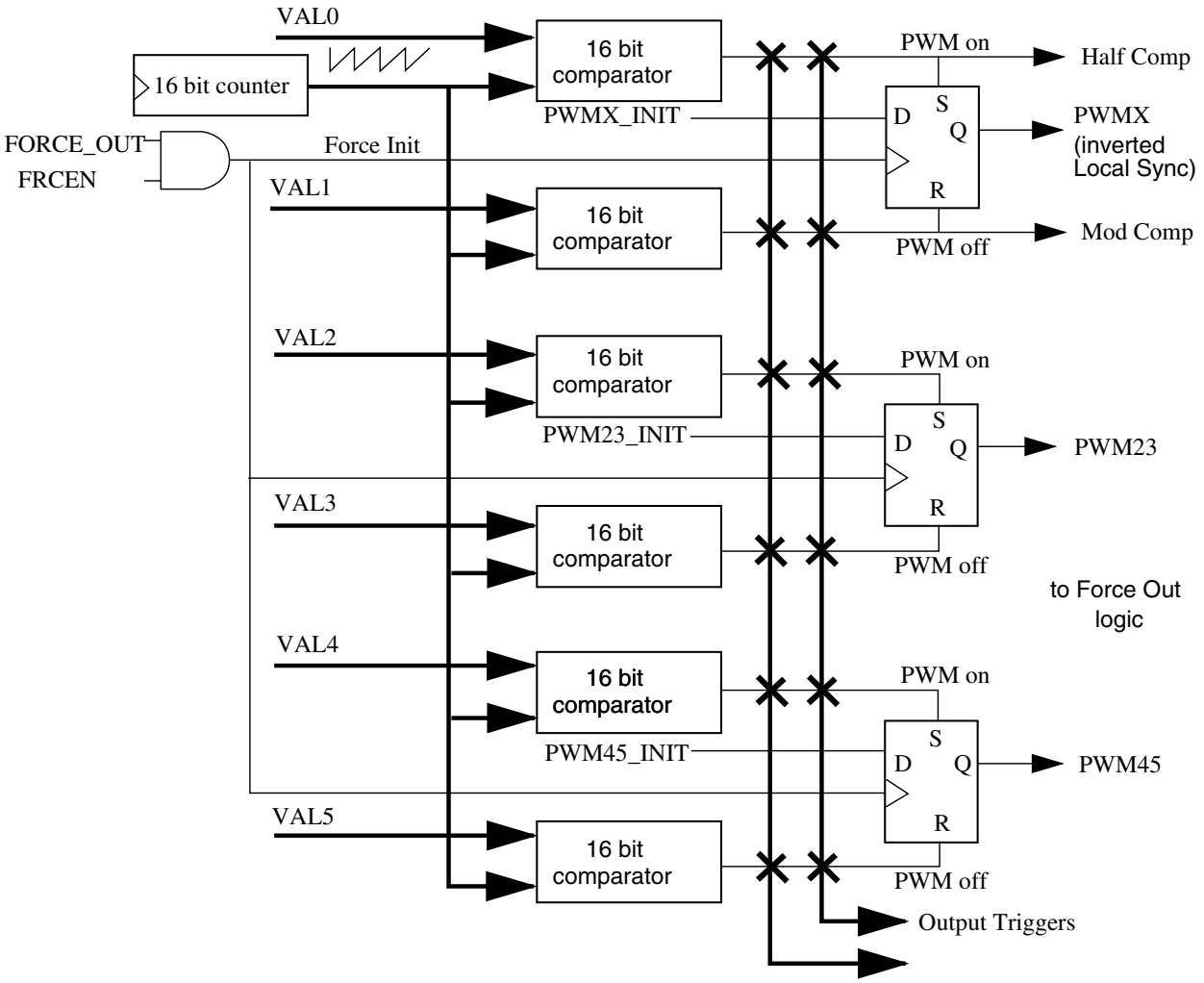


Figure 7-221. PWM Generation Hardware

The generation of the Local Sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a rising edge of the Local Sync signal, comparator 1 generates a falling edge. Comparator 1 is also hardwired to the reload logic to generate the full cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half cycle reload pulse will occur exactly half way through the timer count period and the Local Sync will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Sync signal, effectively turning it into an auxiliary PWM signal (PWMX) assuming that the PWMX pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Sync signal, each submodule is capable of generating three PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The 16-bit comparators shown in Figure 7-221 are "equal to or greater than," not just "equal to," comparators. In addition, if both the set and reset of the flip-flop are asserted, then the flop output goes to 0.

7.4.2.5 Output Compare Capabilities

By using the VALx registers in conjunction with the submodule timer and 16 bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

In PWM generation, an output compare is initiated by programming a VALx register for a timer compare which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWMA signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the VAL3 register must be programmed to a value outside of the modulus range of the counter. Therefore, a compare that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWMA signal that sets it low, the VAL3 register is programmed with the appropriate count value and the VAL2 register is programmed with a value outside the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt or output trigger can be generated when the compare event occurs.

7.4.2.6 Force Out Logic

For each submodule, software can select between seven signal sources for the FORCE_OUT signal: local CTRL2[FORCE], the Master Force signal from submodule0, the local Reload signal, the Master Reload signal from submodule0, the Local Sync signal, the Master Sync signal from submodule0, or the EXT_FORCE signal from on- or off-chip depending on the chip architecture. The local signals are used when the user

simply wants to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master signals or EXT_FORCE signal should be selected.

Figure 7-222 illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the EXTA or EXTB alternate external control signals. The selection can be determined ahead of time and, when a FORCE_OUT event occurs, these values are presented to the signal selection mux that immediately switches the requested signal to the output of the mux for further processing downstream.

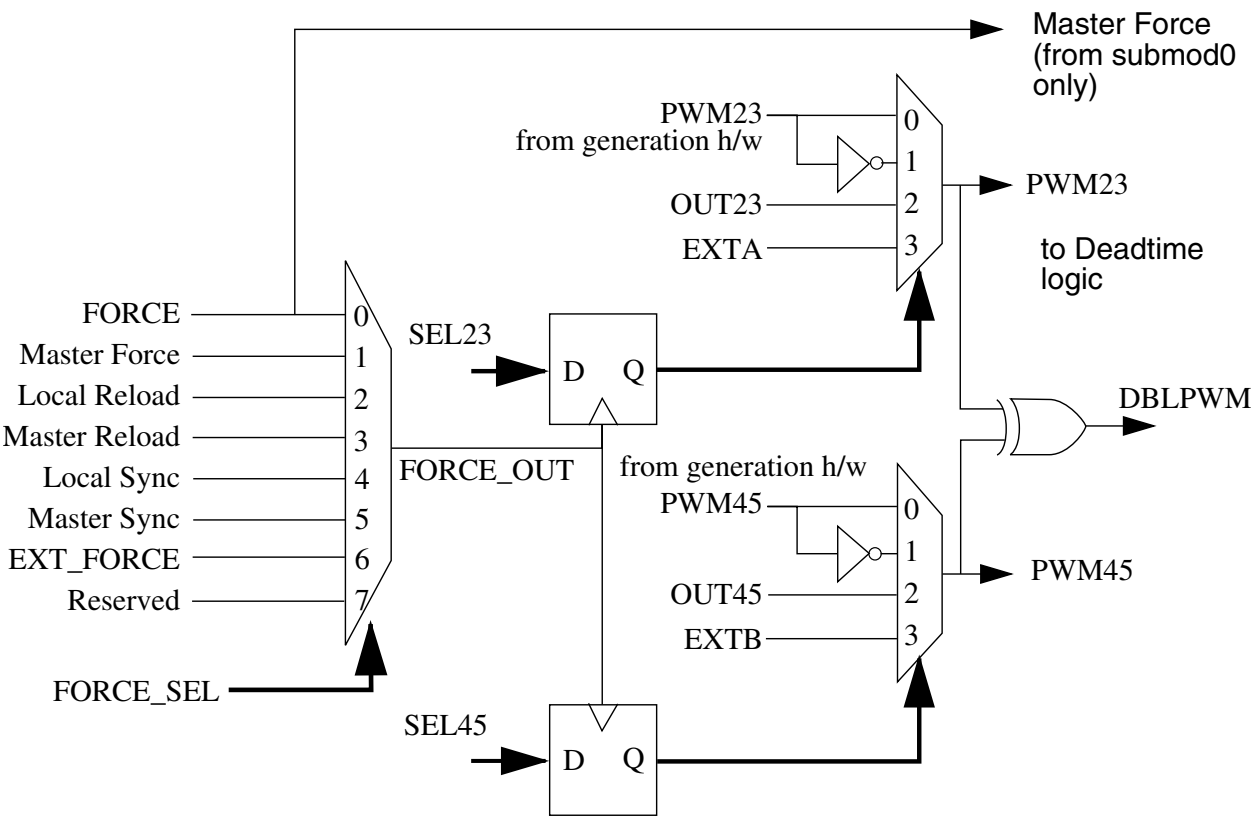


Figure 7-222. Force Out Logic

The local CTRL2[FORCE] signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the CTRL2[FORCE] of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

7.4.2.7 Independent or Complementary Channel Operation

Writing a logic one to CTRL2[INDEP] configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in Figure 7-223 in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by MCTRL[IPOL].

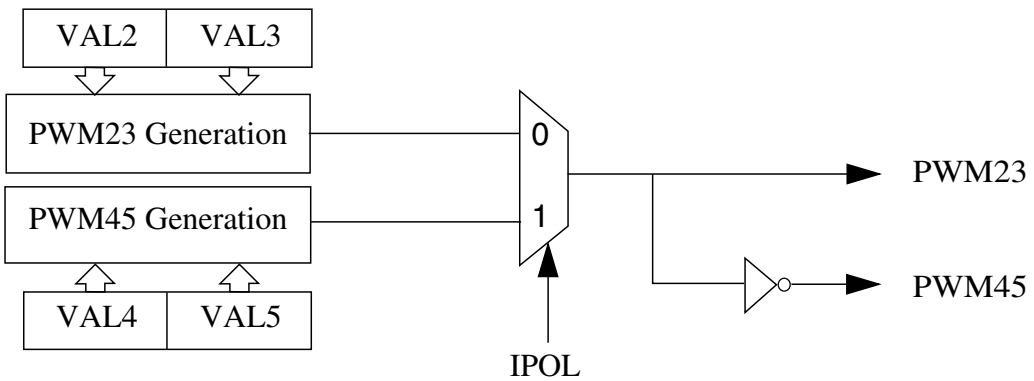


Figure 7-223. Complementary Channel Pair

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in Figure 7-224.

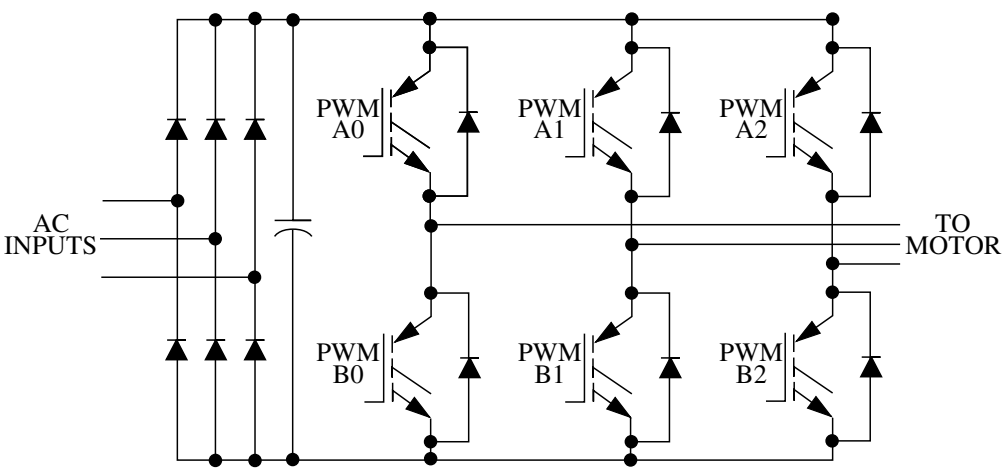


Figure 7-224. Typical 3 Phase AC Motor Drive

Complementary operation allows the use of the deadtime insertion feature.

7.4.2.8 Deadtime Insertion Logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

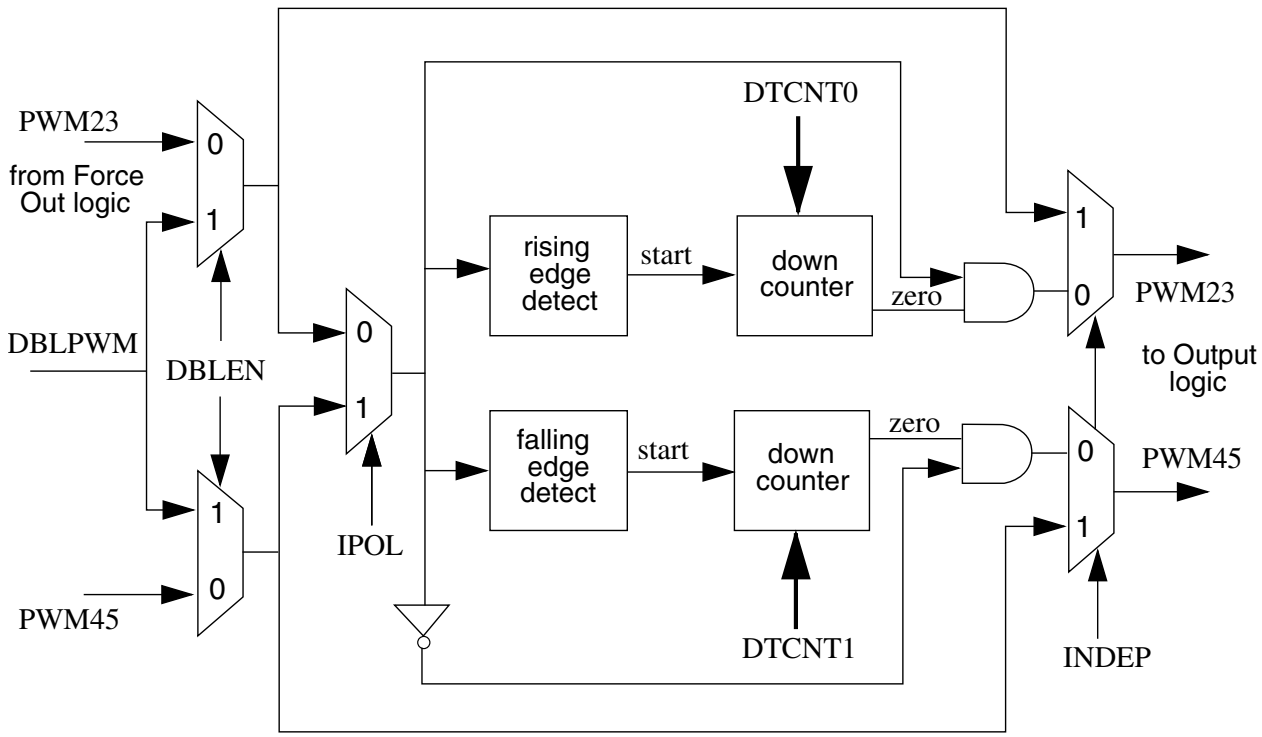


Figure 7-225. Deadtime Insertion Logic

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Note

To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. But the transistor's characteristics may make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as illustrated in the following figure.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

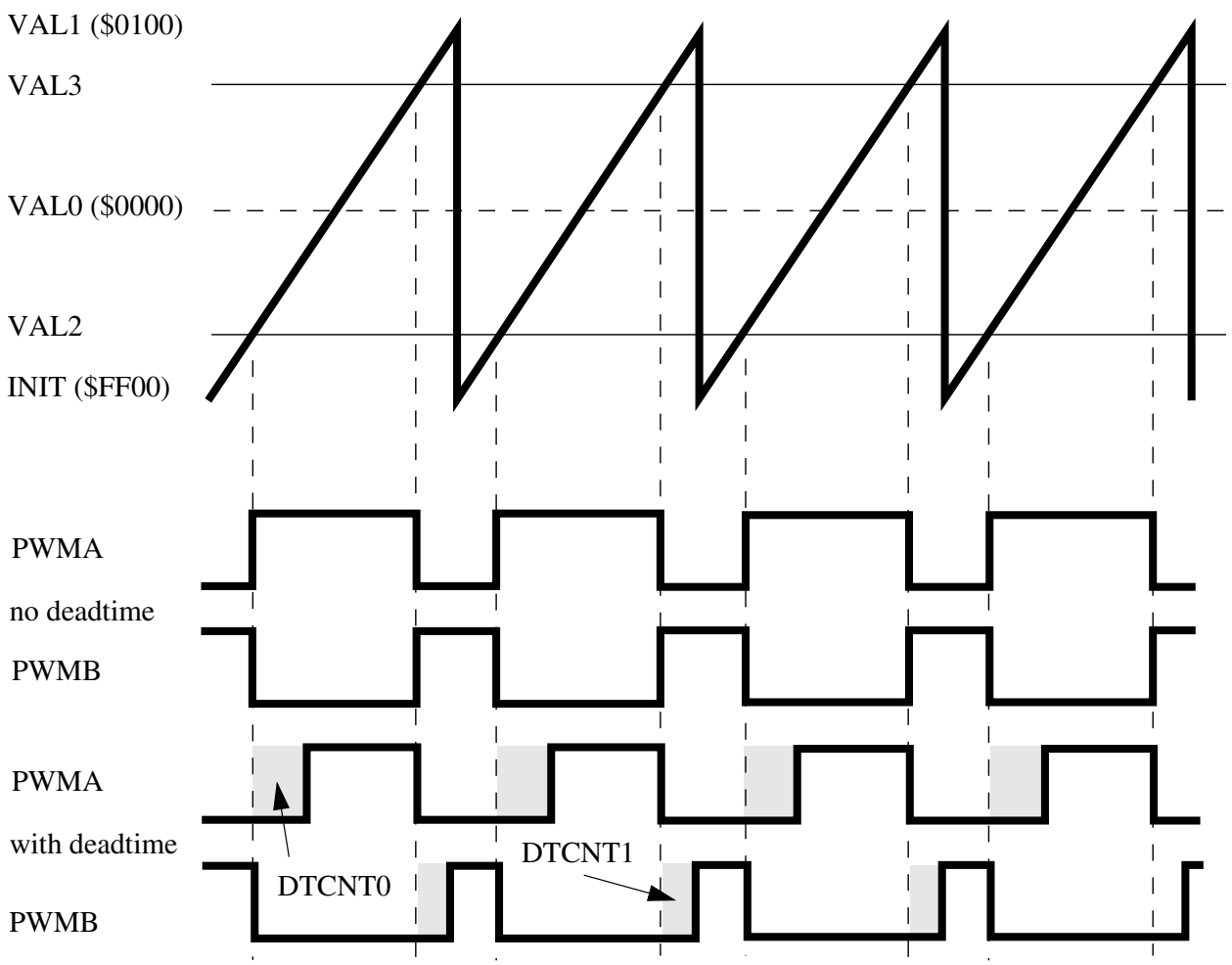


Figure 7-226. Deadtime Insertion

7.4.2.8.1 Top/Bottom Correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage. See the following figure. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

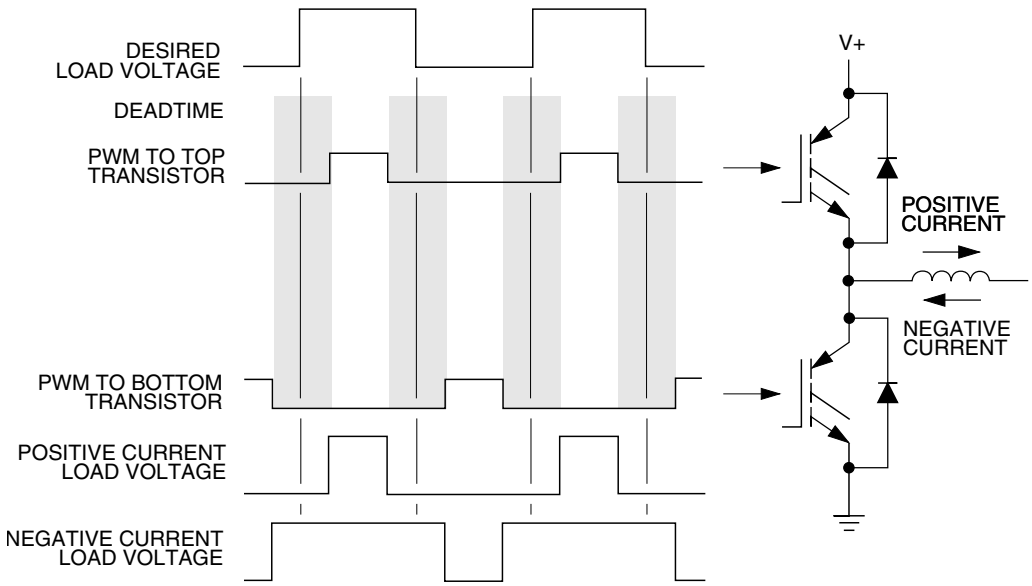


Figure 7-227. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in the motor current waveform. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the motor current for that pair, as the preceding figure shows. To correct

distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the VALx registers. Either the VAL2/VAL3 or the VAL4/VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the VAL2/VAL3 or VAL4/VAL5 pair is active depends on either:

- The state of the current status pin, PWMX, for that driver
- The state of the odd/even correction bit, MCTRL[IPOL], for that driver

To correct deadtime distortion, software can decrease or increase the value in the appropriate VALx register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

7.4.2.8.2 Manual Correction

To detect the current status, the voltage on each PWMX pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in CTRL[DT]. CTRL[DT] is a timing marker especially indicating when to toggle between PWM value registers. Software can then set MCTRL[IPOL] to switch between VAL2/VAL3 and VAL4/VAL5 register pairs according to CTRL[DT] values.

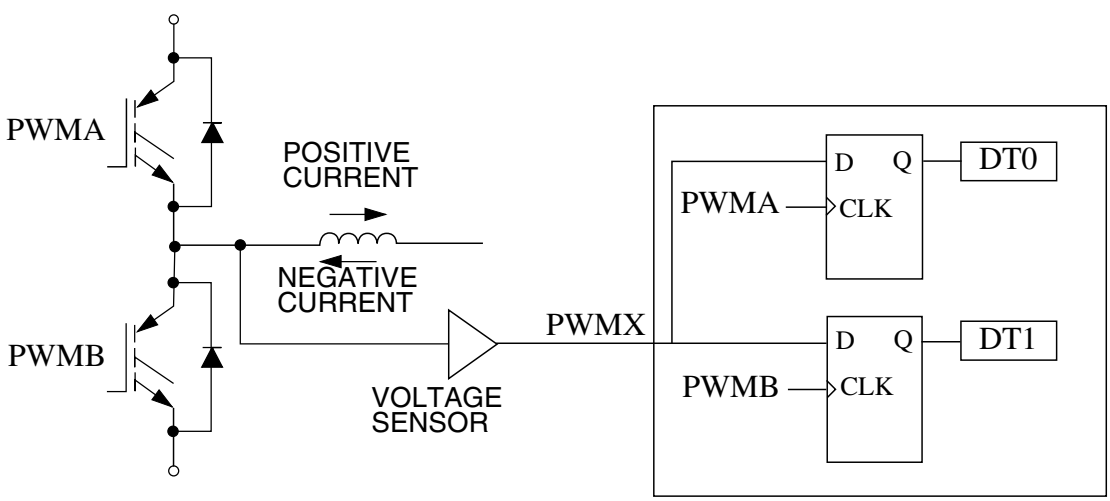


Figure 7-228. Current-status Sense Scheme for Deadtime Correction

Both D flip-flops latch low, CTRL[DT] = 00, during deadtime periods if current is large and flowing out of the complementary circuit. See the preceding figure. Both D flip-flops latch the high, CTRL[DT] = 11, during deadtime periods if current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. **Sampled results will be CTRL[DT] = b10. Thus, the best time to change one PWM value register to another is just before the current zero crossing.**

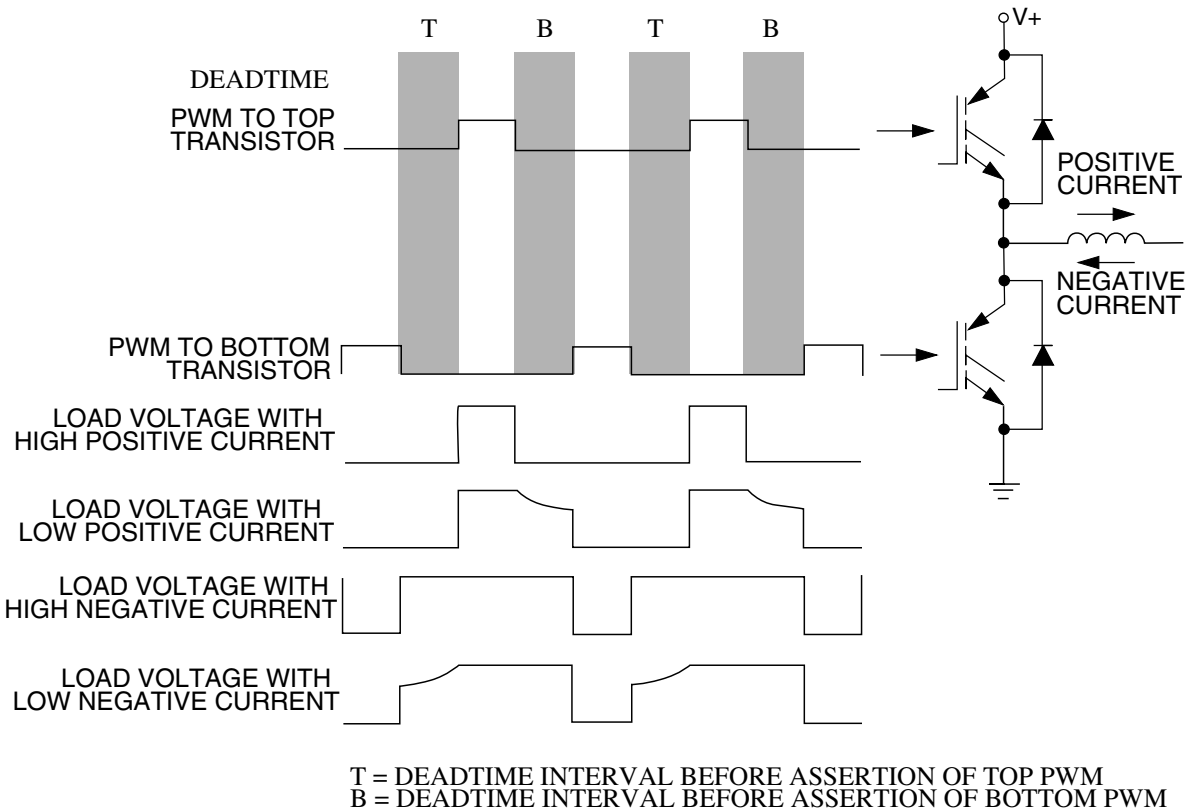


Figure 7-229. Output Voltage Waveforms

7.4.2.9 Fractional Delay Logic

For applications requiring greater resolution than a single IPBus clock period, the fractional delay logic can achieve fine resolution on the rising and falling edges of the PWMA and PWMB outputs. Use the PWM_SMn_FRCTRL registers (where n is 0, 1, or 2) to enable the fractional delay logic where needed (and only where needed). The

FRACVALx registers act as a fractional clock cycle addition to the turn on and turn off count specified by the VAL2, VAL3, VAL4, or VAL5 registers. The FRACVAL1 register acts as a fractional increase in the PWM period as defined by VAL1.

As an example, assume the following conditions:

INIT = 0x0000

VAL1 = 0x000F

VAL2 = 0x0000

VAL3 = 0x0007

FRACVAL3 = 0x00

These settings would cause the PWM output to have a 50% duty cycle: it would be high from a count value of 0x0 to the count value of 0x7, at which point it would go low until the counter reaches the maximum value of 0xF. If FRACVAL3 were changed to a value of 0xb800, then the time the PWM output is active would be:

8 cycles + (23/32) cycle = 8.719 cycles

for a high duty cycle of:

(8.719 cycles / 16 cycles) x 100% = 54.49%

Another case involves fine tuning the PWM period using the FRACVAL1 register. If you want a period of 100.25 clock cycles, program VAL1 with 0x0064 and FRACVAL1 with 0x4000. In this case, the fractional value accumulates so that every 4 PWM cycles are 1 clock cycle longer (101 instead of 100). The rising and falling edges of the PWM outputs also use the accumulated fraction to delay their edges and maintain a consistent spacing of 100.25 cycles between corresponding edges from one cycle to the next.

NOTE

The FRCTRL[FRAC23_EN] bit should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less. Similarly, the FRCTRL[FRAC45_EN] bit should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.

7.4.2.10 Output Logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing to the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic (refer to the figure) are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program OCTRL[POLA] and OCTRL[POLB] before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the OCTRL[PWMxFS] fields.

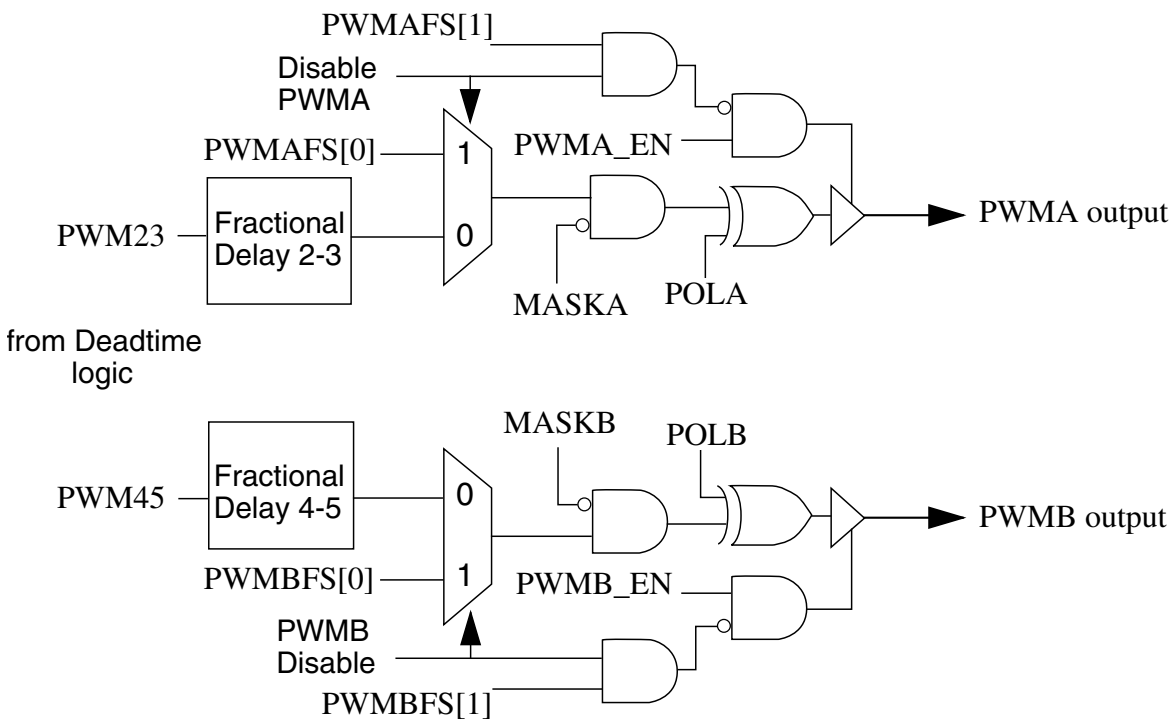


Figure 7-230. Fractional Delay and Output Logic Section

7.4.2.11 E-Capture

Commensurate with the idea of controlling both edges of an output signal, the Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure is a block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8 bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8 bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. this feature is useful for dividing down high frequency signals for capture processing so that capture interrupts don't overwhelm the CPU. Also, this feature can be used to simply generate an interrupt after "n" events have been counted.

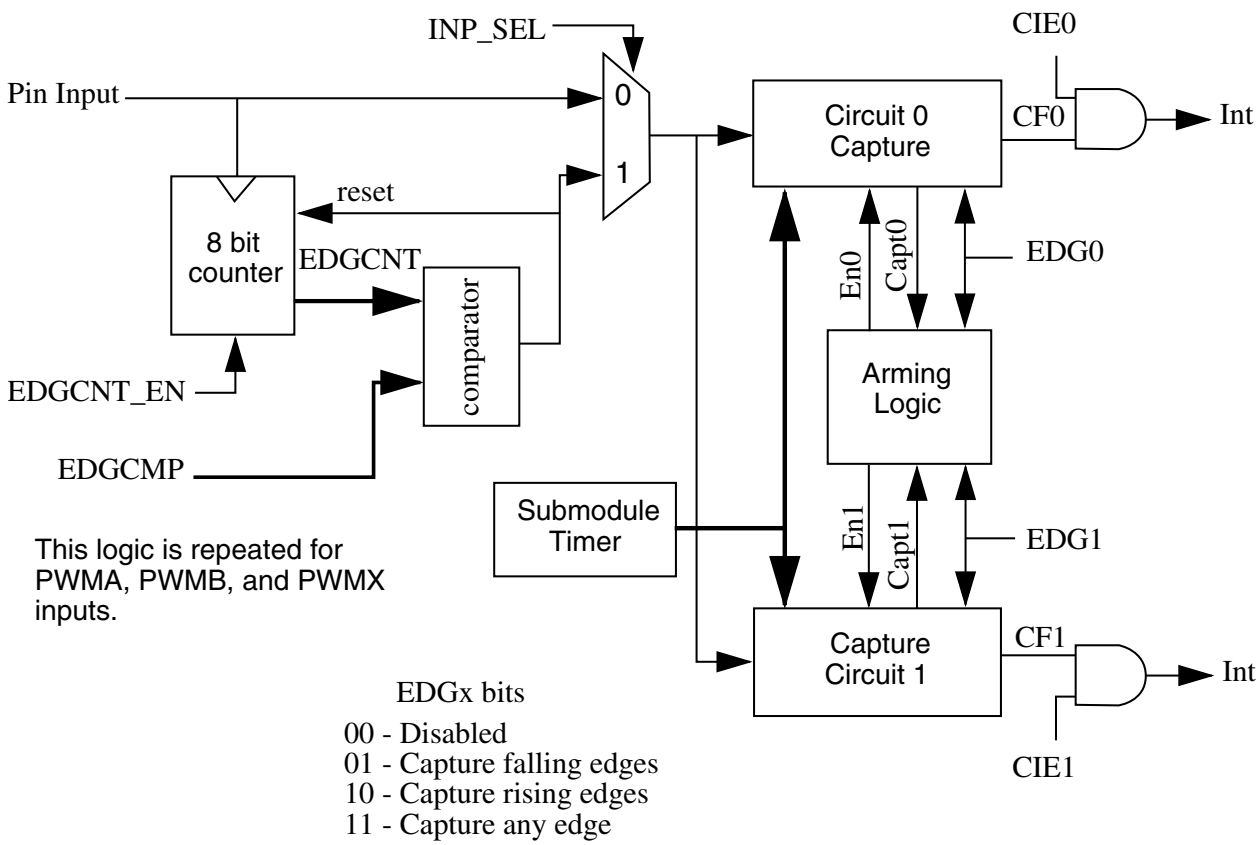


Figure 7-231. Enhanced Capture (E-Capture) Logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by CAPTCTRLx[EDGx1] and CAPTCTRLx[EDGx0], whose functionality is listed in the preceding figure. Also, controlling the operation of the capture circuits is the arming

logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

7.4.2.12 Fault Protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via FCTRL[FLVL]. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or tristated depending the values of OCTRL[PWMxFS].

The fault decoder disables PWM pins selected by the fault logic and the disable mapping register (DISMAP). The following figure shows an example of the fault disable logic. Each bank of bits in DISMAP control the mapping for a single PWM pin. Refer to the following table.

The fault protection is enabled even when the PWM module is not enabled; therefore, a fault will be latched in and must be cleared in order to prevent an interrupt when the PWM is enabled.

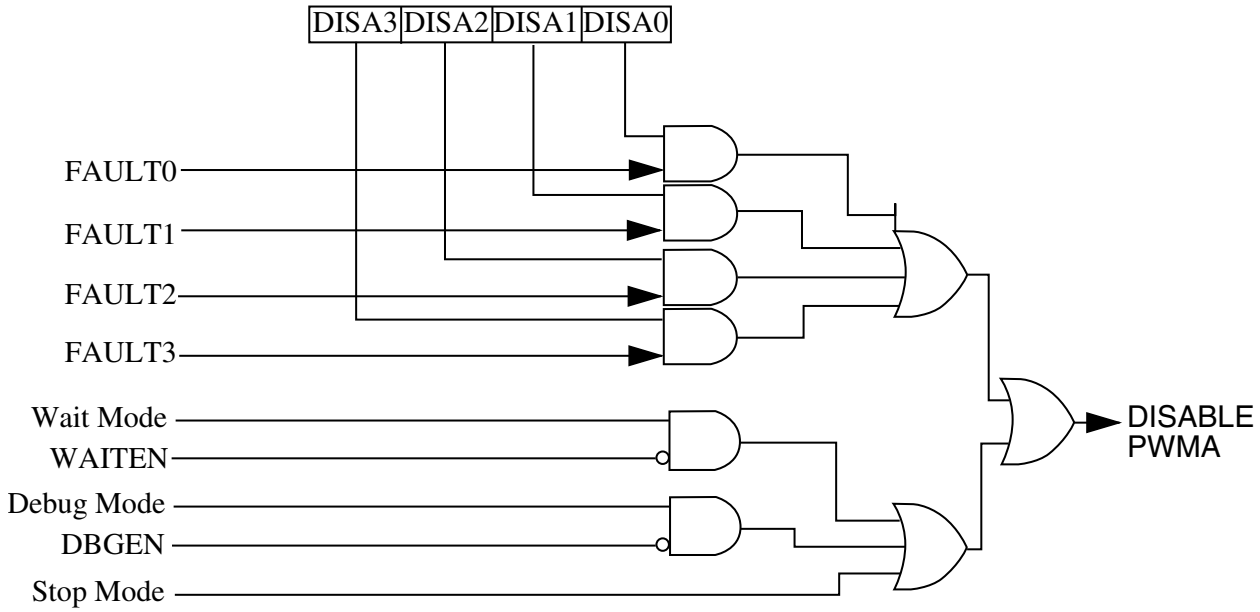


Figure 7-232. Fault Decoder for PWMA

Table 7-209. Fault Mapping

PWM Pin	Controlling Register Bits
PWMA	DISMAP[DISA]
PWMB	DISMAP[DISB]
PWMX	DISMAP[DISX]

7.4.2.12.1 Fault Pin Filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with FFILT[FILT_PER]. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using FFILT[FILT_CNT]. Setting FFILT[FILT_PER] to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FCTRL[FLVLx] is set), the corresponding FSTS[FFPINx] and fault flag, FSTS[FFLAGx], bits are set. FSTS[FFPINx] remains set as long as the filtered FAULTx pin is zero. Clear FSTS[FFLAGx] by writing a logic 1 to FSTS[FFLAGx].

If the FIEx, FAULTx pin interrupt enable bit is set, FSTS[FFLAGx] generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears FSTS[FFLAGx] by writing a logic one to the bit

- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

7.4.2.12.2 Automatic Fault Clearing

Setting an automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for automatic clearing.

When FCTRL[FAUTOx] is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle. Clearing FSTS[FFLAGx] does not affect disabled PWM pins when FCTRL[FAUTOx] is set.

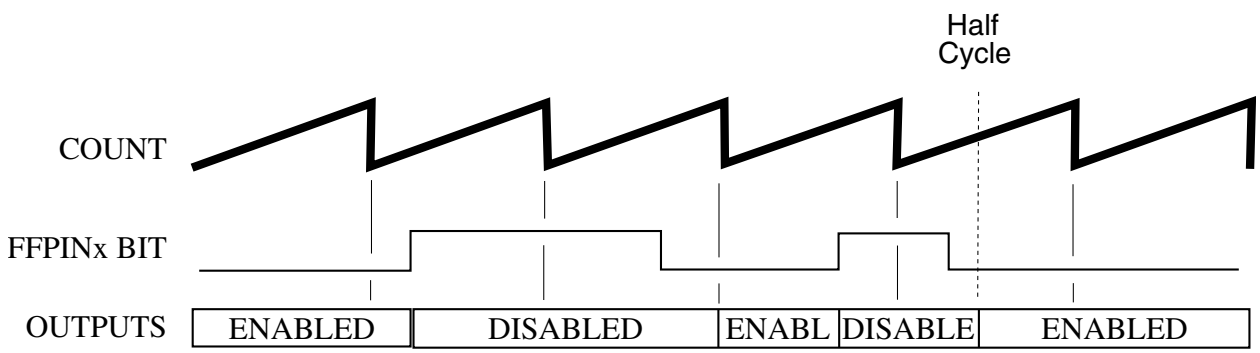


Figure 7-233. Automatic Fault Clearing

7.4.2.12.3 Manual Fault Clearing

Clearing the automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits, FCTRL[FSAFEx], are clear, then PWM pins disabled by the FAULTx pins are enabled when:

- Software clears the corresponding FSTS[FFLAGx] flag
- The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin. See the first following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.

- If the fault safety mode bits, FCTRL[FSAFEx], are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag
  - The filter detects a logic one on the FAULTx pin at the start of the next PWM full or half cycle boundary. See the second following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.

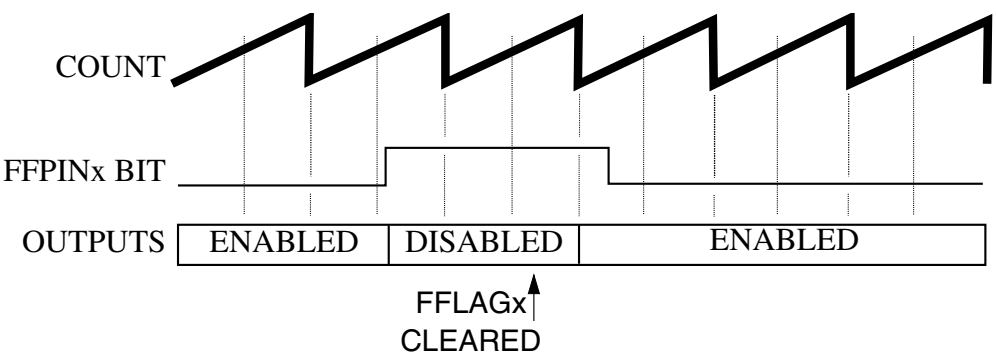


Figure 7-234. Manual Fault Clearing (FCTRL[FSAFE]=0)

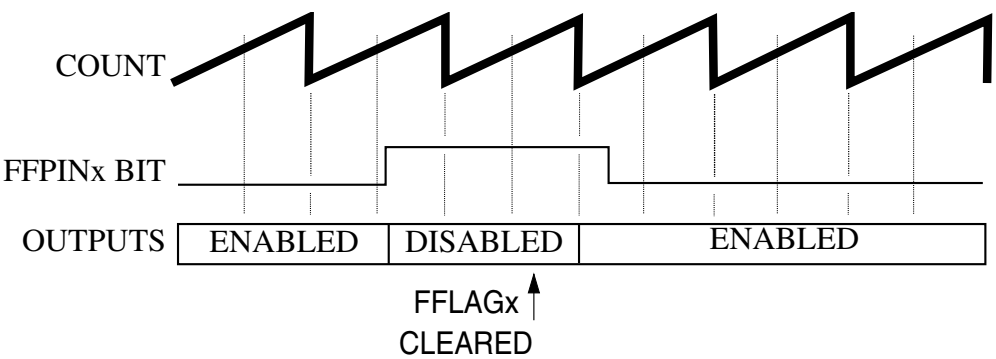


Figure 7-235. Manual Fault Clearing (FCTRL[FSAFE]=1)

Note

Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or EXTA and EXTB. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, MCTRL[RUN] equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, MCTRL[RUN] equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

7.4.2.12.4 Fault Testing

FSTS[FTEST] is used to simulate a fault condition on each of the fault inputs.

7.4.3 PWM Generator Loading

7.4.3.1 Load Enable

MCTRL[LDOK] enables loading of the following PWM generator parameters:

- The prescaler divisor—from CTRL[PRSC]
- The PWM period and pulse width—from the INIT and VALx registers

MCTRL[LDOK] allows software to finish calculating all of these PWM parameters so they can be synchronously updated. The CTRL[PRSC], INIT, and VALx registers are loaded by software into a set of outer buffers. When MCTRL[LDOK] is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator. These values can be transfered to the inner set of registers immediately upon setting MCTRL[LDOK] if CTRL[LDMOD] is set. Set MCTRL[LDOK] by reading it when it is a logic zero and then writing a logic one to it. After loading, MCTRL[LDOK] is automatically cleared.

7.4.3.2 Load Frequency

CTRL[LDFQ] selects an integral loading frequency of one to 16 PWM reload opportunities. CTRL[LDFQ] takes effect at every PWM reload opportunity, regardless the state of MCTRL[LDOK]. CTRL[HALF] and CTRL[FULL] control reload timing. If

CTRL[FULL] is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If CTRL[HALF] is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both CTRL[HALF] and CTRL[FULL] are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

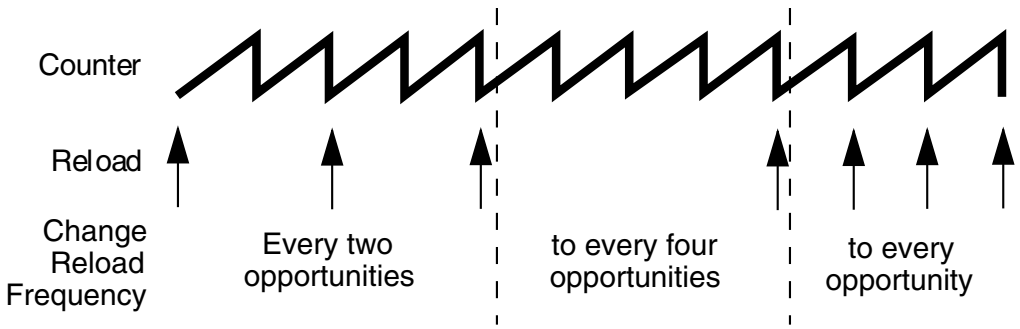


Figure 7-236. Full Cycle Reload Frequency Change

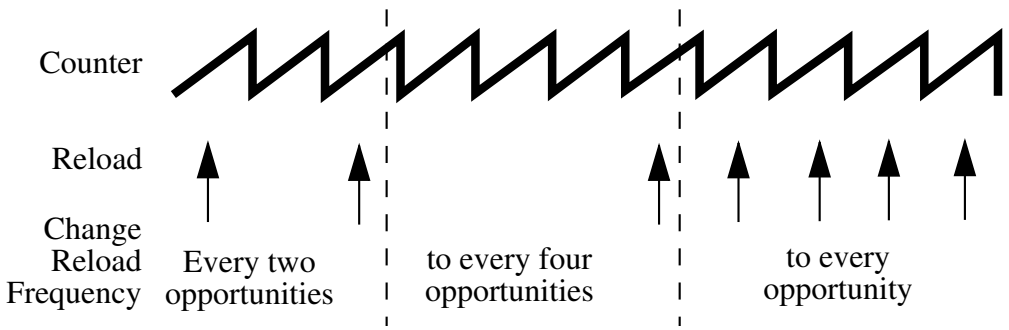


Figure 7-237. Half Cycle Reload Frequency Change

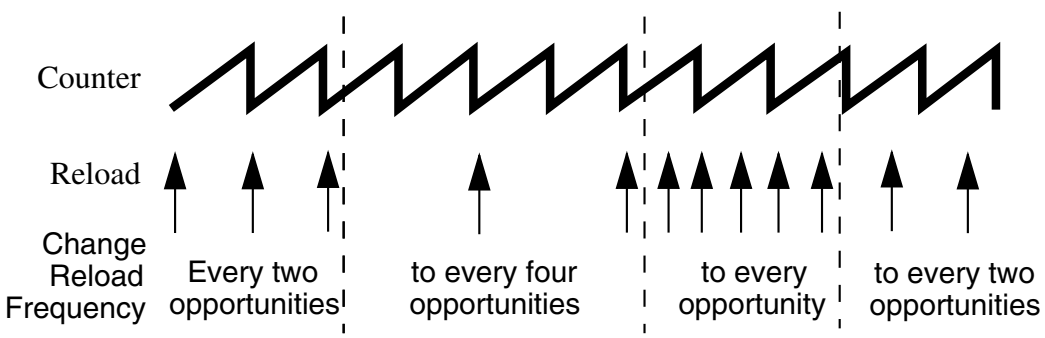


Figure 7-238. Full and Half Cycle Reload Frequency Change

7.4.3.3 Reload Flag

At every reload opportunity the PWM Reload Flag (STS[RF]) is set. Setting STS[RF] happens even if an actual reload is prevented by MCTRL[LDOK]. If the PWM reload interrupt enable bit, INTEN[RIE], is set, the STS[RF] flag generates CPU interrupt

requests allowing software to calculate new PWM parameters in real time. When INTEN[RIE] is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

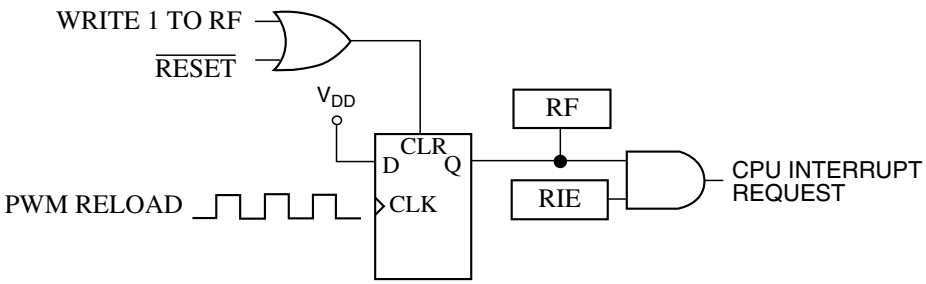


Figure 7-239. PWMF Reload Interrupt Request

7.4.3.4 Reload Errors

Whenever one of the VALx, FRACVALx, or CTRL[PRSC] registers is updated, the STS[RUF] flag is set to indicate that the data is not coherent. STS[RUF] will be cleared by a successful reload which consists of the reload signal while MCTRL[LDOK] is set. If STS[RUF] is set and MCTRL[LDOK] is clear when the reload signal occurs, a reload error has taken place and STS[REF] is set. If STS[RUF] is clear when a reload signal asserts, then the data is coherent and no error will be flagged.

7.4.3.5 Initialization

Initialize all registers and set MCTRL[LDOK] before setting MCTRL[RUN].

Note

Even if MCTRL[LDOK] is not set, setting MCTRL[RUN] also sets the STS[RF] flag. To prevent a CPU interrupt request, clear INTEN[RIE] before setting MCTRL[RUN].

The PWM generator uses the last values loaded if MCTRL[RUN] is cleared and then set while MCTRL[LDOK] equals zero.

When MCTRL[RUN] is cleared:

- The STS[RF] flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active

- Software/external output control remains active
- Deadtime insertion continues during software/external output control

7.5 Resets

All PWM registers are reset to their default values upon any system reset.

The reset forces all registers to their reset states and tri-states the PWM outputs.

7.6 Interrupts

Each of the submodules within the eFlexPWM module can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 7-210. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CMP0	SM0STS[CMPF]	SM0INTEN[CMPIE]	Submodule 0 compare interrupt	Compare event has occurred
PWM_RELOAD0	SM0STS[RF]	SM0INTEN[RIE]	Submodule 0 reload interrupt	Reload event has occurred
PWM_CMP1	SM1STS[CMPF]	SM1INTEN[CMPIE]	Submodule 1 compare interrupt	Compare event has occurred
PWM_RELOAD1	SM1STS[RF]	SM1INTEN[RIE]	Submodule 1 reload interrupt	Reload event has occurred
PWM_CMP2	SM2STS[CMPF]	SM2INTEN[CMPIE]	Submodule 2 compare interrupt	Compare event has occurred
PWM_RELOAD2	SM2STS[RF]	SM2INTEN[RIE]	Submodule 2 reload interrupt	Reload event has occurred
PWM_CMP3	SM3STS[CMPF]	SM3INTEN[CMPIE]	Submodule 3 compare interrupt	Compare event has occurred
PWM_CAP3	SM3STS[CFA1], SM3STS[CFA0], SM3STS[CFB1], SM3STS[CFB0], SM3STS[CFX1], SM3STS[CFX0]	SM3INTEN[CFA1IE], SM3INTEN[CFA0IE], SM3INTEN[CFB1IE], SM3INTEN[CFB0IE], SM3INTEN[CFX1IE], SM3INTEN[CFX0IE	Submodule 3 input capture interrupt	Input capture event has occurred

Table continues on the next page...

Table 7-210. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_RELOAD3	SM3STS[RF]	SM3INTEN[RIE]	Submodule 3 reload interrupt	Reload event has occurred
PWM_RERR	SM0STS[REF]	SM0INTEN[REIE]	Submodule 0 reload error interrupt	Reload error has occurred
	SM1STS[REF]	SM1INTEN[REIE]	Submodule 1 reload error interrupt	
	SM2STS[REF]	SM2INTEN[REIE]	Submodule 2 reload error interrupt	
	SM3STS[REF]	SM3INTEN[REIE]	Submodule 3 reload error interrupt	
PWM_FAULT	FSTS[FFLAG]	FCTRL[FIE]	Fault input interrupt	Fault condition has been detected

Chapter 8
Crossbar Switch (XBAR)

8.1 Introduction

8.1.1 Overview

The crossbar switch module implements an array of 30 outputs and 22 inputs of combinational digital multiplexes. All 30 multiplexes share the same 22 inputs in the same order, but each multiplex has its own independent select field. The module's registers control the select field for each multiplex.

This module is designed to provide a flexible crossbar switching matrix that allows any input (typically from external GPIO or internal module outputs) to be connected to any output (typically to external GPIO or internal module inputs) under user control. This is used to allow user configuration of data paths between internal modules and between internal modules and GPIO.

8.1.2 Features

The crossbar switch (XBAR) module's design includes these distinctive features:

- Capability for generic inter-module connections between on-chip peripherals including ADCs, DAC, comparators, timers, PWM module, and GPIO pins
- No synchronization delay between input and output
- 30 identical 22-input multiplexes with individual select fields
- Register write protection input signal

8.1.3 Block Diagram

The following figure is a block diagram of the XBAR module.

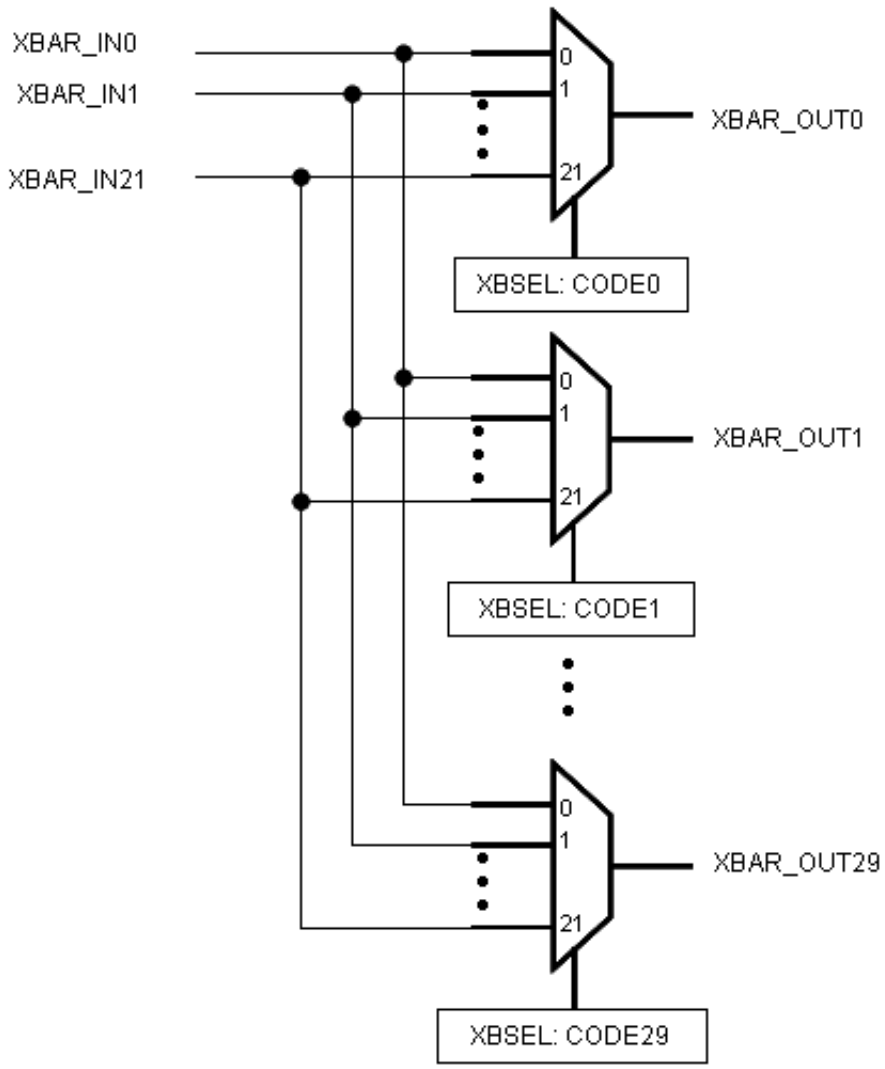


Figure 8-1. Crossbar Switch Module Block Diagram

8.2 Memory Map and Registers

Address offset (hex)	Register name
0	XB_XBC0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			CODE[2n+1]					0			CODE[2n]				
W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	XB_XBC1	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
2	XB_XBC2	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
3	XB_XBC3	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
4	XB_XBC4	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
5	XB_XBC5	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
6	XB_XBC6	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
7	XB_XBC7	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
8	XB_XBC8	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
9	XB_XBC9	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
A	XB_XBC10	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
B	XB_XBC11	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
C	XB_XBC12	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
D	XB_XBC13	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
E	XB_XBC14	R	0			CODE[2n+1]					0			CODE[2n]				
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

8.2.1 Crossbar Control Register n (XB_XBC*n*)

Addresses: F100h base + 0h offset + (1d × *n*), where *n* = 0d to 14d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			CODE[2n+1]					0			CODE[2n]				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XB_XBC*n* field descriptions

Field	Description
15–13 Reserved	This read-only bitfield is reserved and always has the value zero.
12–8 CODE[2n+1]	Input (XBAR_INx) to be muxed to XBAR_OUT[2n+1] (refer to Functional Description section for input/output assignment)
7–5 Reserved	This read-only bitfield is reserved and always has the value zero.
4–0 CODE[2n]	Input (XBAR_INx) to be muxed to XBAR_OUT[2n] (refer to Functional Description section for input/output assignment)

8.3 Functional Description

The crossbar switch module provides a generic mechanism for making connections between on-chip peripherals and between peripheral and pins. It provides a purely combinational path from input to output. The module groups 30 identical multiplexors with 22 shared inputs. All crossbar control registers that are used to select one of the 22 input signals to output are write protected. Control of the write protection setting is in the SIM_PROT register.

In functional mode, the value of each output XBAR_OUT[n] is XBAR_IN[CODEn].

The signal assignments of the crossbar switch's inputs and outputs follow.

Table 8-18. Crossbar Switch Input Encoding

XBAR_INn	Encode	Input from	Function
XBAR_IN0	00h	Logic zero	V_{SS}
XBAR_IN1	01h	Logic one	V_{DD}
XBAR_IN2	02h	XB_IN2	Package pin
XBAR_IN3	03h	XB_IN3	Package pin

Table continues on the next page...

Table 8-18. Crossbar Switch Input Encoding (continued)

XBAR_INn	Encode	Input from	Function
XBAR_IN4	04h	XB_IN4	Package pin
XBAR_IN5	05h	XB_IN5	Package pin
XBAR_IN6	06h	XB_IN6	Package pin
XBAR_IN7	07h	XB_IN7	Package pin
XBAR_IN8	08h	Unused	
XBAR_IN9	09h	CMPA_OUT	Comparator A Output
XBAR_IN10	0Ah	CMPB_OUT	Comparator B Output
XBAR_IN11	0Bh	CMPC_OUT	Comparator C Output
XBAR_IN12	0Ch	TB0	Quad Timer B0 Output
XBAR_IN13	0Dh	TB1	Quad Timer B1 Output
XBAR_IN14	0Eh	TB2	Quad Timer B2 Output
XBAR_IN15	0Fh	TB3	Quad Timer B3 Output
XBAR_IN16	10h	PWM0_TRIG_COMB	PWM_OUT_TRIG0[0] or PWM_OUT_TRIG1[0]
XBAR_IN17	11h	PWM1_TRIG_COMB	PWM_OUT_TRIG0[1] or PWM_OUT_TRIG1[1]
XBAR_IN18	12h	PWM2_TRIG_COMB	PWM_OUT_TRIG0[2] or PWM_OUT_TRIG1[2]
XBAR_IN19	13h	PWM012_TRIG_COMB	PWM0_TRIG_COMB or PWM1_TRIG_COMB or PWM2_TRIG_COMB
XBAR_IN20	14h	PWM3_TRIG0	PWM3_OUT_TRIG[0]
XBAR_IN21	15h	PWM3_TRIG1	PWM3_OUT_TRIG[1]

Table 8-19. Crossbar Switch Output

XBAR_OUTn	Output to	Function
XBAR_OUT0	XB_OUT0	Package pin
XBAR_OUT1	XB_OUT1	Package pin
XBAR_OUT2	XB_OUT2	Package pin
XBAR_OUT3	XB_OUT3	Package pin
XBAR_OUT4	XB_OUT4	Package pin
XBAR_OUT5	XB_OUT5	Package pin
XBAR_OUT6	ADCA	ADCA Trigger
XBAR_OUT7	ADCB	ADCB Trigger
XBAR_OUT8	DAC	12-bit DAC SYNC_IN
XBAR_OUT9	CMPA	Comparator A Window/Sample
XBAR_OUT10	CMPB	Comparator B Window/Sample
XBAR_OUT11	CMPC	Comparator C Window/Sample
XBAR_OUT12	PWM0 EXTA	eFlexPWM submodule 0 Alternate Control signal

Table continues on the next page...

Table 8-19. Crossbar Switch Output (continued)

XBAR_OUTn	Output to	Function
XBAR_OUT13	PWM1 EXTA	eFlexPWM submodule 1 Alternate Control signal
XBAR_OUT14	PWM2 EXTA	eFlexPWM submodule 2 Alternate Control signal
XBAR_OUT15	PWM3 EXTA	eFlexPWM submodule 3 Alternate Control signal
XBAR_OUT16	PWM0 EXT_SYNC	eFlexPWM submodule 0 External Synchronization signal
XBAR_OUT17	PWM1 EXT_SYNC	eFlexPWM submodule 1 External Synchronization signal
XBAR_OUT18	PWM2 EXT_SYNC	eFlexPWM submodule 2 External Synchronization signal
XBAR_OUT19	PWM3 EXT_SYNC	eFlexPWM submodule 3 External Synchronization signal
XBAR_OUT20	PWM EXT_CLK	eFlexPWM External Clock signal
XBAR_OUT21	PWM FAULT0	eFlexPWM module FAULT0
XBAR_OUT22	PWM FAULT1	eFlexPWM module FAULT1
XBAR_OUT23	PWM FAULT2	eFlexPWM module FAULT2
XBAR_OUT24	PWM FAULT3	eFlexPWM module FAULT3
XBAR_OUT25	PWM FORCE	eFlexPWM External Output Force signal
XBAR_OUT26	TB0	Quad Timer B0 Input when SIM_GPS3[12] is set
XBAR_OUT27	TB1	Quad Timer B1 Input when SIM_GPS3[13] is set
XBAR_OUT28	TB2	Quad Timer B2 Input when SIM_GPS3[14] is set
XBAR_OUT29	TB3	Quad Timer B3 Input when SIM_GPS3[15] is set

8.4 Clocks and Resets

The crossbar module runs at standard system bus speeds and assumes reset states as defined in the device's data sheet. Reset occurs whenever any source of system reset occurs (POR, external reset, COP reset, and so on).

8.5 Interrupts

The XBAR module does not generate interrupts.

Chapter 9
General-Purpose Input/Output (GPIO)

9.1 Overview

The general-purpose input/output (GPIO) module allows direct read or write access to pin values or the ability to assign a pin to be used as an external interrupt. GPIO pins are multiplexed with other peripherals on the package. The device's data sheet specifies the assigned GPIO ports and the multiplexed pin package.

A GPIO pin can be configured in different operation modes:

- As GPIO input with, or without, pullup
- As GPIO output with push-pull mode or open-drain mode
- As a peripheral pin when multiplexed with another module

GPIOs are placed on the device in groups of one to sixteen bits, called ports and designated as A, B, C, and so on. Refer to the device's data sheet for the specific definition of each of the GPIO ports on the chip.

9.1.1 Features

The GPIO module's design includes these features:

- Individual control for each pin to be in either peripheral mode or GPIO mode
- Individual direction control for each pin in GPIO mode
- Individual pull-up enable control for each pin in either peripheral mode or GPIO mode
- Individual selection of output push-pull mode or open-drain mode for each pin
- Individual output drive strength control (high-power mode or low-power mode) for each pin
- Ability to monitor each pin's logic values when pin is in either GPIO mode or peripheral mode by using raw data (RDATA) register

- Each pin has the ability to generate an interrupt with programmable rising or falling edge and software interrupt
- 5 V tolerance

9.1.2 Modes of Operation

The GPIO module can operate in two major modes:

- Peripheral mode: The peripheral module controls the pin. However, if the pin is not configured as analog input, then output drive strength, push-pull or open-drain output, and pullup enable are still controlled by GPIO registers.
- GPIO mode: The GPIO module controls the pin. Any data output and input can be written to or read from GPIO data registers. GPIO pins can generate the edge interrupt and insert the software interrupt.

9.2 Memory Map and Registers

Each GPIO register contains up to 16 bits, each of which performs an identical function for one of the GPIO pins controlled by that GPIO port. The only difference that arises is with regard to initial operating conditions at reset; some GPIO modes are on by default, and some are not. Some pullup resistors may be enabled, while others are not. Refer to the device's data sheet for the reset state of each register.

For simplicity, this section shows each GPIO port's registers with the same width of 16 bits, corresponding to 16 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is as follows:

- Port A: 8
- Port B: 8
- Port C: 16
- Port D: 5
- Port E: 8
- Port F: 9

Any register bit for which no corresponding port pin exists is reserved. For example, bits 15-8 of every GPIOA register are reserved.

NOTE

The reset value of these registers may differ depending on the reset function of specific pins. Refer to the device's data sheet.

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	GPIOA_PUR	R W	PU															
1	GPIOA_DR	R W	D															
2	GPIOA_DDR	R W	DD															
3	GPIOA_PER	R W	PE															
4	GPIOA_IAR	R W	IA															
5	GPIOA_IENR	R W	IEN															
6	GPIOA_IPOLR	R W	IPOL															
7	GPIOA_IPR	R W	IP															
8	GPIOA_IESR	R W	IES															
9	GPIOA_ PPMODE	R W	PPMODE															
A	GPIOA_ RAWDATA	R W	RAW_DATA															
B	GPIOA_DRIVE	R W	DRIVE															
0	GPIOB_PUR	R W	PU															
1	GPIOB_DR	R W	D															
2	GPIOB_DDR	R W	DD															
3	GPIOB_PER	R W	PE															

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	GPIOB_IAR	R	IA															
		W																
5	GPIOB_IENR	R	IEN															
		W																
6	GPIOB_IPOLR	R	IPOL															
		W																
7	GPIOB_IPR	R	IP															
		W																
8	GPIOB_IESR	R	IES															
		W																
9	GPIOB_ PPMODE	R	PPMODE															
		W																
A	GPIOB_ RAWDATA	R	RAW_DATA															
		W																
B	GPIOB_DRIVE	R	DRIVE															
		W																
0	GPIOC_PUR	R	PU															
		W																
1	GPIOC_DR	R	D															
		W																
2	GPIOC_DDR	R	DD															
		W																
3	GPIOC_PER	R	PE															
		W																
4	GPIOC_IAR	R	IA															
		W																
5	GPIOC_IENR	R	IEN															
		W																
6	GPIOC_IPOLR	R	IPOL															
		W																
7	GPIOC_IPR	R	IP															
		W																
8	GPIOC_IESR	R	IES															
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
9	GPIOC_PPMODE	R	PPMODE															
		W																
A	GPIOC_RAWDATA	R	RAW_DATA															
		W																
B	GPIOC_DRIVE	R	DRIVE															
		W																
0	GPIOD_PUR	R	PU															
		W																
1	GPIOD_DR	R	D															
		W																
2	GPIOD_DDR	R	DD															
		W																
3	GPIOD_PER	R	PE															
		W																
4	GPIOD_IAR	R	IA															
		W																
5	GPIOD_IENR	R	IEN															
		W																
6	GPIOD_IPOLR	R	IPOL															
		W																
7	GPIOD_IPR	R	IP															
		W																
8	GPIOD_IESR	R	IES															
		W																
9	GPIOD_PPMODE	R	PPMODE															
		W																
A	GPIOD_RAWDATA	R	RAW_DATA															
		W																
B	GPIOD_DRIVE	R	DRIVE															
		W																
0	GPIOE_PUR	R	PU															
		W																
1	GPIOE_DR	R	D															
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2	GPIOE_DDR	R	DD															
		W																
3	GPIOE_PER	R	PE															
		W																
4	GPIOE_IAR	R	IA															
		W																
5	GPIOE_IENR	R	IEN															
		W																
6	GPIOE_IPOLR	R	IPOL															
		W																
7	GPIOE_IPR	R	IP															
		W																
8	GPIOE_IESR	R	IES															
		W																
9	GPIOE_ PPMODE	R	PPMODE															
		W																
A	GPIOE_ RAWDATA	R	RAW_DATA															
		W																
B	GPIOE_DRIVE	R	DRIVE															
		W																
0	GPIOF_PUR	R	PU															
		W																
1	GPIOF_DR	R	D															
		W																
2	GPIOF_DDR	R	DD															
		W																
3	GPIOF_PER	R	PE															
		W																
4	GPIOF_IAR	R	IA															
		W																
5	GPIOF_IENR	R	IEN															
		W																
6	GPIOF_IPOLR	R	IPOL															
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7	GPIOF_IPR	R	IP															
		W																
8	GPIOF_IESR	R	IES															
		W																
9	GPIOF_ PPMODE	R	PPMODE															
		W																
A	GPIOF_ RAWDATA	R	RAW_DATA															
		W																
B	GPIOF_DRIVE	R	DRIVE															
		W																

9.2.1 GPIO Pullup Enable Register (GPIOx_PUR)

This register is for internal pullup enabling and disabling. If the pin is configured as an output, this register is not used. This register is read and write. Unimplemented bits read as 0.

The pullup is intended only to drive an undriven input pin to a known state. It is characteristically a very weak pullup.

Addresses: GPIOA_PUR – F140h base + 0h offset = F140h
GPIOB_PUR – F150h base + 0h offset = F150h
GPIOC_PUR – F160h base + 0h offset = F160h
GPIOD_PUR – F170h base + 0h offset = F170h
GPIOE_PUR – F180h base + 0h offset = F180h
GPIOF_PUR – F190h base + 0h offset = F190h

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

GPIOx_PUR field descriptions

Field	Description
15–0 PU	Pullup Enable Bits 0 Pullup is disabled 1 Pullup is enabled

9.2.2 GPIO Data Register (GPIOx_DR)

This register holds data that comes either from the pin or the data bus. In other words, the register is the data interface between the pin and the data bus.

Data written to this register appears on the pins if the pins are configured as GPIO output. Data read from this register is the same as the read state on the pins if those pins are configured as GPIO input.

Addresses: GPIOA_DR – F140h base + 1h offset = F141h
GPIOB_DR – F150h base + 1h offset = F151h
GPIOC_DR – F160h base + 1h offset = F161h
GPIOD_DR – F170h base + 1h offset = F171h
GPIOE_DR – F180h base + 1h offset = F181h
GPIOF_DR – F190h base + 1h offset = F191h

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

GPIOx_DR field descriptions

Field	Description
15–0 D	Data Bits

9.2.3 GPIO Data Direction Register (GPIOx_DDR)

This read/write register configures the state of the pin as either input or output when the pin is configured as GPIO (the corresponding bit in the GPIO peripheral enable register is set to zero). When the register is set to zero, the pin is an input. When the register is set to one, the pin is an output.

Addresses: GPIOA_DDR – F140h base + 2h offset = F142h
GPIOB_DDR – F150h base + 2h offset = F152h
GPIOC_DDR – F160h base + 2h offset = F162h
GPIOD_DDR – F170h base + 2h offset = F172h
GPIOE_DDR – F180h base + 2h offset = F182h
GPIOF_DDR – F190h base + 2h offset = F192h

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

GPIOx_DDR field descriptions

Field	Description
15–0 DD	Data Direction Bits 0 Pin is an input 1 Pin is an output

9.2.4 GPIO Peripheral Enable Register (GPIOx_PER)

This read/write register determines the configuration of the GPIO pins.

When the Peripheral Enable bitfield value in this register is one, the GPIO module is configured for peripheral mode. In this mode, a peripheral controls the GPIO pin where the data transfer direction depends on the function of the peripheral.

When the Peripheral Enable bitfield value is zero, the pin is configured for GPIO mode. In this mode, the corresponding GPIO Data Direction register controls the data flow.

If write protection (via the SIM PROT register) is implemented on an individual chip, then this register value cannot be changed after the write protect signal has been asserted.

NOTE

The reset value of the GPIOD_PER register differs from the value of the other GPIOx_PER registers. The reset value of the GPIOD_PER register is 0x001F.

Addresses: GPIOA_PER – F140h base + 3h offset = F143h
GPIOB_PER – F150h base + 3h offset = F153h
GPIOC_PER – F160h base + 3h offset = F163h
GPIOD_PER – F170h base + 3h offset = F173h
GPIOE_PER – F180h base + 3h offset = F183h
GPIOF_PER – F190h base + 3h offset = F193h

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

GPIOx_PER field descriptions

Field	Description
15–0 PE	Peripheral Enable Bits 0 Pin is for GPIO (GPIO mode) 1 Pin is for peripheral (peripheral mode)

9.2.5 GPIO Interrupt Assert Register (GPIOx_IAR)

This read and write register is only for software testing, but it also provides a software interrupt capability. When the bit is set to one, an interrupt is asserted. The interrupt is generated continually until this bit is cleared. Clear the bits in the register by writing zeros.

Addresses: GPIOA_IAR – F140h base + 4h offset = F144h
GPIOB_IAR – F150h base + 4h offset = F154h
GPIOC_IAR – F160h base + 4h offset = F164h
GPIOD_IAR – F170h base + 4h offset = F174h
GPIOE_IAR – F180h base + 4h offset = F184h
GPIOF_IAR – F190h base + 4h offset = F194h

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

GPIOx_IAR field descriptions

Field	Description
15–0 IA	Interrupt Assert Bits

GPIOx_IAR field descriptions (continued)

Field	Description
	0 Deassert software interrupt 1 Assert software interrupt

9.2.6 GPIO Interrupt Enable Register (GPIOx_IENR)

This register enables or disables the edge interrupt from each GPIO pin. Set a bit to 1 to enable the interrupt for the associated GPIO pin. The interrupt is recorded in the corresponding GPIO Interrupt Pending register.

Addresses: GPIOA_IENR – F140h base + 5h offset = F145h
GPIOB_IENR – F150h base + 5h offset = F155h
GPIOC_IENR – F160h base + 5h offset = F165h
GPIOD_IENR – F170h base + 5h offset = F175h
GPIOE_IENR – F180h base + 5h offset = F185h
GPIOF_IENR – F190h base + 5h offset = F195h

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

GPIOx_IENR field descriptions

Field	Description
15–0 IEN	Interrupt Enable Bits 0 External Interrupt is disabled 1 External Interrupt is enabled

9.2.7 GPIO Interrupt Polarity Register (GPIOx_IPOLR)

This read/write register is used for polarity detection caused by any external interrupts. The interrupt at the pin is active low when this register is set to one (falling edge causes the interrupt). The interrupt seen at the pin is active high when this register is set to zero (rising edge causes the interrupt).

Addresses: GPIOA_IPOLR – F140h base + 6h offset = F146h
GPIOB_IPOLR – F150h base + 6h offset = F156h
GPIOC_IPOLR – F160h base + 6h offset = F166h
GPIOD_IPOLR – F170h base + 6h offset = F176h
GPIOE_IPOLR – F180h base + 6h offset = F186h
GPIOF_IPOLR – F190h base + 6h offset = F196h

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

GPIOx_IPOLR field descriptions

Field	Description
15–0 IPOL	Interrupt Polarity Bits 0 Interrupt occurred on rising edge 1 Interrupt occurred on falling edge

9.2.8 GPIO Interrupt Pending Register (GPIOx_IPR)

This read-only register is used to record any incoming interrupts. The user can read this register to determine which pin has caused the interrupt. This register can be cleared by writing ones into the GPIO Interrupt Edge Sensitive register if the interrupt is caused by a pin, or by writing zeros into the GPIO Interrupt Assert register if the interrupt is caused by software.

Addresses: GPIOA_IPR – F140h base + 7h offset = F147h
GPIOB_IPR – F150h base + 7h offset = F157h
GPIOC_IPR – F160h base + 7h offset = F167h
GPIOD_IPR – F170h base + 7h offset = F177h
GPIOE_IPR – F180h base + 7h offset = F187h
GPIOF_IPR – F190h base + 7h offset = F197h

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

GPIOx_IPR field descriptions

Field	Description
15–0 IP	Interrupt Pending Bits 0 No Interrupt 1 Interrupt occurred

9.2.9 GPIO Interrupt Edge Sensitive Register (GPIOx_IESR)

When an edge is detected by the edge detector circuit and the GPIO Interrupt Enable register's field is set to one, this register's field records the interrupt. This read/write register clears the corresponding Interrupt Pending bit by writing one to the appropriate Interrupt Edge Sensitive bit. Writing zero to an Interrupt Edge Sensitive bit is ignored.

Addresses: GPIOA_IESR – F140h base + 8h offset = F148h
GPIOB_IESR – F150h base + 8h offset = F158h
GPIOC_IESR – F160h base + 8h offset = F168h
GPIOD_IESR – F170h base + 8h offset = F178h
GPIOE_IESR – F180h base + 8h offset = F188h
GPIOF_IESR – F190h base + 8h offset = F198h

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

GPIOx_IESR field descriptions

Field	Description
15–0 IES	Interrupt Edge-Sensitive Bits 0 No edge detected if read; no effect if writing 1 An edge detected if read; clear corresponding Interrupt Pending bit if writing

9.2.10 GPIO Push-Pull Mode Register (GPIOx_PPMODE)

This register can be used to explicitly set each output driver to either push-pull or open-drain mode. If write protection (via the SIM PROT register) is implemented on an individual device, then this register value cannot be changed after the write protect signal is asserted.

NOTE

Open-drain mode can be used to tri-state any pin on the GPIO port without switching that pin to input mode. This capability is useful for some applications, including a keypad interface.

Addresses: GPIOA_PPMODE – F140h base + 9h offset = F149h
GPIOB_PPMODE – F150h base + 9h offset = F159h
GPIOC_PPMODE – F160h base + 9h offset = F169h
GPIOD_PPMODE – F170h base + 9h offset = F179h
GPIOE_PPMODE – F180h base + 9h offset = F189h
GPIOF_PPMODE – F190h base + 9h offset = F199h

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

GPIOx_PPMODE field descriptions

Field	Description
15–0 PPMODE	Push-Pull Mode Bits 0 Open Drain Mode 1 Push-Pull Mode

9.2.11 GPIO Raw Data Register (GPIOx_RAWDATA)

This read-only register gives the DSP direct access to the logic values on each GPIO pin, even when pins are not in GPIO mode. Values are not clocked and are subject to change at any time. Read several times to ensure a stable value.

Addresses: GPIOA_RAWDATA – F140h base + Ah offset = F14Ah
GPIOB_RAWDATA – F150h base + Ah offset = F15Ah
GPIOC_RAWDATA – F160h base + Ah offset = F16Ah
GPIOD_RAWDATA – F170h base + Ah offset = F17Ah
GPIOE_RAWDATA – F180h base + Ah offset = F18Ah
GPIOF_RAWDATA – F190h base + Ah offset = F19Ah

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

* Notes:
- x = Undefined at reset.

GPIOx_RAWDATA field descriptions

Field	Description
15–0 RAW_DATA	Raw Data Bits

9.2.12 GPIO Drive Strength Control Register (GPIOx_DRIVE)

This register can be used to explicitly set the drive strength of each output driver. If write protection (via the SIM PROT register) is implemented on an individual device, then this register value cannot be changed after the write protect signal is asserted.

Addresses: GPIOA_DRIVE – F140h base + Bh offset = F14Bh
GPIOB_DRIVE – F150h base + Bh offset = F15Bh
GPIOC_DRIVE – F160h base + Bh offset = F16Bh
GPIOD_DRIVE – F170h base + Bh offset = F17Bh
GPIOE_DRIVE – F180h base + Bh offset = F18Bh
GPIOF_DRIVE – F190h base + Bh offset = F19Bh

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

GPIOx_DRIVE field descriptions

Field	Description
15–0 DRIVE	Drive Strength Selector Bits 0 Low drive strength: 4 mA 1 High drive strength: 8 mA

9.3 Functional Description

The following block diagram illustrates the logic associated with just one of the bits in each GPIO port. Each GPIO pin can be configured as:

- An input, with or without pullup functions
- An edge interrupt
- An output with push-pull or open-drain mode

The GPIO's pullup is configured by writing the pullup enable register. When the pin is configured for a peripheral function, the pullups are controlled by the pullup enable register and the direction is specified by the peripheral used. If the pin is set to be an output, the pullup is disabled.

A pin may have several peripheral functions, one of which may be an analog function. To access its analog function, the pin must be in peripheral mode with an analog input enabled. Selecting between an analog peripheral or a digital peripheral is controlled by the GPIO Peripheral Select Register (GPSn) in the SIM module. When the GPIO is in peripheral mode and its analog peripheral function is selected, the digital output buffer and pullup are disabled. The digital input buffer is also disconnected from the pin so that the digital input is not responding to analog voltages on the pin.

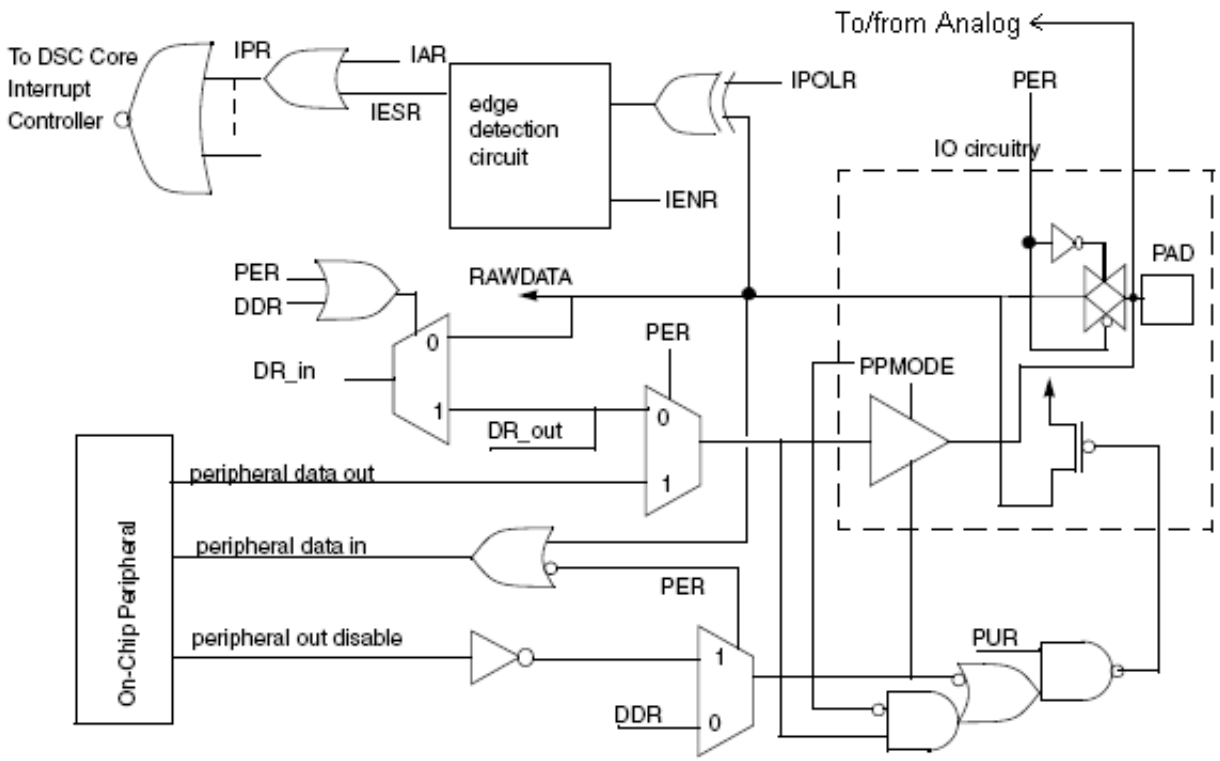


Figure 9-85. Bit View of the GPIO Logic with a Multiplex of Analog Input

9.4 Interrupts

The GPIO module has two types of interrupts:

1. Software interrupt

Write ones to the interrupt assert register to generate the software interrupt. The interrupt pending register records the value of the interrupt assert register. Clear the the interrupt pending register by writing zeroes to the the interrupt assert register.

NOTE

When a software interrupt is asserted, the interrupt polarity register, interrupt edge sensitive register, and the interrupt enable register must be zero to guarantee that the interrupt registered in the interrupt pending register is due only to the interrupt assert register.

2. Hardware interrupt from the pin

   When a GPIO pin is used as an external interrupt source, its IEN bit in the interrupt enable register is set to 1 and the interrupt assert register must be set to 0. The interrupt polarity register must be set to 1 for a falling edge interrupt and to 0 for a rising edge interrupt. When the signal at the pin transitions from high to low or low to high, the value is seen at the interrupt edge sensitive register and recorded by the interrupt pending register.

The interrupt signals in each port are ORed together, presenting only a single interrupt per port to the interrupt controller. The interrupt service routine must then check the contents of the interrupt pending register to determine which pin(s) caused the interrupt. External interrupt sources do not need to remain asserted because the detection mechanism is edge sensitive.

9.5 Clocks and Resets

The GPIO module runs at standard system bus speeds and assumes reset states as defined in the device data sheet. Reset occurs whenever any source of system reset occurs (POR, external reset, COP reset, and so on).

Chapter 10
Inter-Integrated Circuit (I2C)

10.1 Introduction

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 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*.

10.1.1 Features

The I2C module has these distinctive 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

10.1.2 Modes of Operation

To summarize the I2C module's operation in various low power modes:

- 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.

10.1.3 Block Diagram

The following figure is a block diagram of the I2C module.

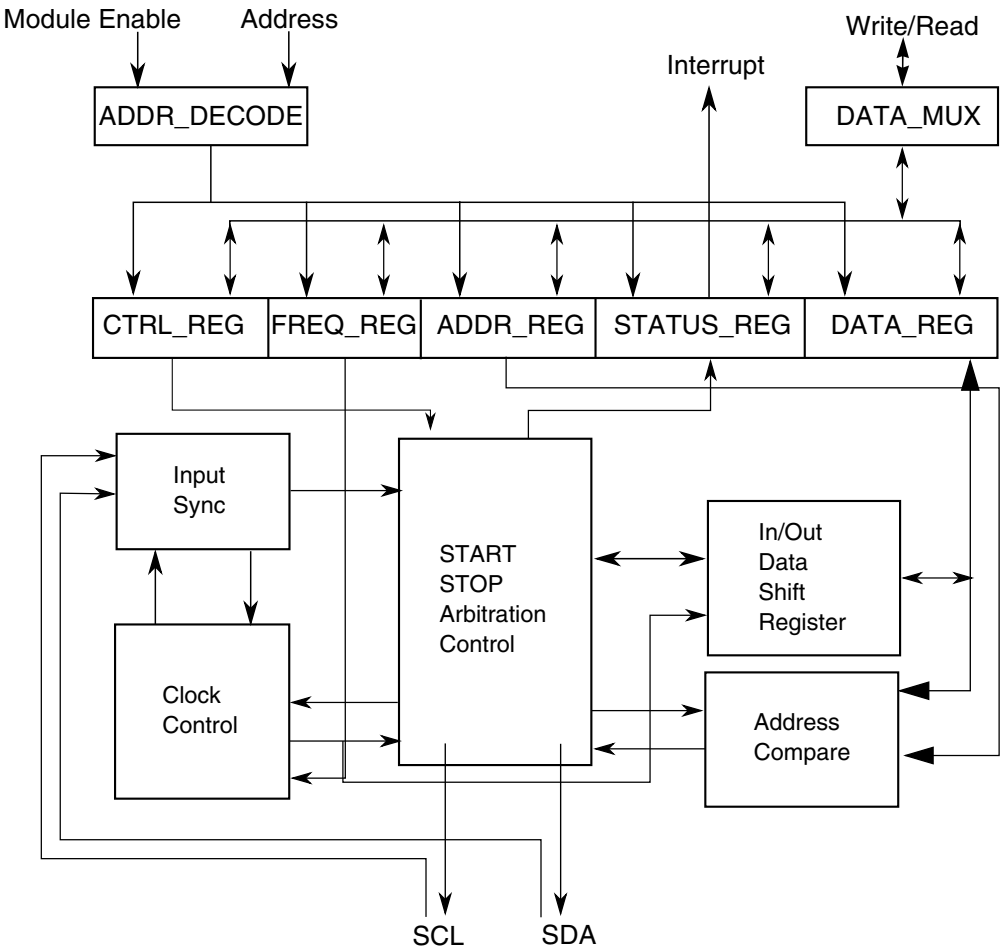


Figure 10-1. I2C Functional Block Diagram

10.2 Signal Descriptions

This section describes each user-accessible pin signal.

10.2.1 Serial Clock Line (SCL)

The bidirectional SCL is the serial clock line of the I2C system.

10.2.2 Serial Data Line (SDA)

The bidirectional SDA is the serial data line of the I2C system.

10.3 Memory Map and Registers

This section describes in detail all I2C registers accessible to the end user.

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	I2C0_ADDR	R	0								AD[7:1]							0
		W																
1	I2C0_FREQDIV	R	0								MULT		ICR					
		W																
2	I2C0_CR1	R	0								IICEN	IICIE	MST	TX	TXAK	0	WUEN	0
		W														RSTA		
3	I2C0_SR	R	0								TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
		W																
4	I2C0_DATA	R	0								DATA							
		W																
5	I2C0_CR2	R	0								GCAEN	ADEXT	0	0	0	AD[10:8]		
		W																
6	I2C0_FILT	R	0								0					FLT		
		W																
7	I2C0_SMB_CSR	R	0								FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
		W																
8	I2C0_ADDR2	R	0								SAD							0
		W																
9	I2C0_SLT1	R	0								SSLT[15:8]							
		W																
A	I2C0_SLT2	R	0								SSLT[7:0]							
		W																
0	I2C1_ADDR	R	0								AD[7:1]							0
		W																
1	I2C1_FREQDIV	R	0								MULT		ICR					
		W																
2	I2C1_CR1	R	0								IICEN	IICIE	MST	TX	TXAK	0	WUEN	0
		W														RSTA		
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3	I2C1_SR	R	0								TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
		W																
4	I2C1_DATA	R	0								DATA							
		W																
5	I2C1_CR2	R	0								GCAEN	ADEXT	0	0	0	AD[10:8]		
		W																
6	I2C1_FILT	R	0								0					FLT		
		W																
7	I2C1_SMB_CSR	R	0								FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
		W																
8	I2C1_ADDR2	R	0								SAD							0
		W																
9	I2C1_SLT1	R	0								SSLT[15:8]							
		W																
A	I2C1_SLT2	R	0								SSLT[7:0]							
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

10.3.1 I2C Address Register 1 (I2Cx_ADDR)

This register contains the slave address to be used by the I2C module.

Addresses: I2C0_ADDR – F210h base + 0h offset = F210h

I2C1_ADDR – F220h base + 0h offset = F220h

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

I2Cx_ADDR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–1 AD[7:1]	Address

Table continues on the next page...

I2Cx_ADDR field descriptions (continued)

Field	Description
	Contains the 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.
0 Reserved	This read-only bit is reserved and always has the value zero.

10.3.2 I2C Frequency Divider register (I2Cx_FREQDIV)

Addresses: I2C0_FREQDIV – F210h base + 1h offset = F211h

I2C1_FREQDIV – F220h base + 1h offset = F221h

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

I2Cx_FREQDIV field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–6 MULT	The MULT bits define 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

5–0
ICR
	Clock rate
	
	Prescales the bus 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. See I2C Divider and Hold Values for a list of values corresponding to each ICR setting.
	
	The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate.
	
	`I2C baud rate = bus 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 = bus 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 = bus period (s) × mul × SCL start hold value`

Table continues on the next page...

I2Cx_FREQDIV field descriptions (continued)

Field	Description
	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 = bus period (s) × mul × SCL stop hold value`

10.3.3 I2C Control Register 1 (I2Cx_CR1)

Addresses: I2C0_CR1 – F210h base + 2h offset = F212h

I2C1_CR1 – F220h base + 2h offset = F222h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								IICEN	IICIE	MST	TX	TXAK	0	WUEN	0
Write														RSTA		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_CR1 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
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 the MST bit is changed from a 0 to a 1, a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 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 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.

Table continues on the next page...

I2Cx_CR1 field descriptions (continued)

Field	Description
	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 the FACK bit affects NACK/ACK generation. 0 An acknowledge signal is sent to the bus on the following (if FACK is cleared) or current (if FACK is set) receiving byte. 1 No acknowledge signal is sent to the bus on the following (if FACK is cleared) or current (if FACK is set) receiving data byte.NOTE:SCL is held low until TXAK is written.
2 RSTA	Repeat START Writing a one to this bit generates a repeated START condition provided it is the current master. This bit will always be read as zero. Attempting a repeat at the wrong time results in loss of arbitration.
1 WUEN	Wakeup enable I2C module can wake the core from stop mode when slave address matching occurs. 0 Normal operation. No interrupt generated when address matching in stop mode. 1 Enables the wakeup function in stop mode.
0 Reserved	This read-only bit is reserved and always has the value zero.

10.3.4 I2C Status Register (I2Cx_SR)

Addresses: I2C0_SR – F210h base + 3h offset = F213h

I2C1_SR – F220h base + 3h offset = F223h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
Write																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

I2Cx_SR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 TCF	Transfer complete flag This bit sets on the completion of a byte and acknowledge bit transfer. This bit is only valid during or immediately following a transfer to or from the I2C module. The TCF bit 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_SR field descriptions (continued)

Field	Description
	0 Transfer in progress 1 Transfer complete
6 IAAS	Addressed as a slave The IAAS bit is set when one of the following conditions is met: • The calling address matches the programmed slave address • GCAEN is set and a general call is received • SIICAEN is set and the calling address matches the second programmed slave address • ALERTEN is set and an SMBus alert response address is received This bit sets before the ACK bit does. The CPU needs to check the SRW bit and set TX/RX accordingly. Writing the I2C Control Register 1 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 a one to it. 0 Standard bus operation. 1 Loss of arbitration.
3 Reserved	This read-only bit is reserved and always has the value zero.
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 The IICIF bit sets when an interrupt is pending. This bit must be cleared by software, by writing a one to it in the interrupt routine. One of the following events can set the IICIF bit: • One byte transfer including ACK/NACK bit completes if FACK = 0 • One byte transfer excluding ACK/NACK bit completes if FACK = 1. An ACK or NACK is sent on the bus by writing 0 or 1 to TXAK after this bit is set in receive mode • Match of slave address to calling address including primary slave address, general call address, alert response address, and second slave address. • Arbitration lost • In SMBus mode, any timeouts except SCL and SDA high timeouts 0 No interrupt pending 1 Interrupt pending

Table continues on the next page...

I2Cx_SR field descriptions (continued)

Field	Description
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

10.3.5 I2C Data I/O register (I2Cx_DATA)

Addresses: I2C0_DATA – F210h base + 4h offset = F214h

I2C1_DATA – F220h base + 4h offset = F224h

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

I2Cx_DATA field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 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 transitioning 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 CR1[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, then 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 modes. 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. 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 comprise of the calling address (in bit 7-1) concatenated with the required R/$\overline{W}$ bit (in position bit 0).

10.3.6 I2C Control Register 2 (I2Cx_CR2)

Addresses: I2C0_CR2 – F210h base + 5h offset = F215h

I2C1_CR2 – F220h base + 5h offset = F225h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								GCAEN	ADEXT	0	0	0	AD[10:8]		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_CR2 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
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 read-only bit is reserved and always has the value zero.
4 Reserved	This read-only bit is reserved and always has the value zero.
3 Reserved	This read-only bit is reserved and always has the value zero.
2–0 AD[10:8]	Slave address Contains the upper three bits of the slave address in the 10-bit address scheme. This field is only valid when the ADEXT bit is set.

10.3.7 I2C Programmable Input Glitch Filter register (I2Cx_FILT)

Addresses: I2C0_FILT – F210h base + 6h offset = F216h

I2C1_FILT – F220h base + 6h offset = F226h

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

I2Cx_FILT field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–3 Reserved	This read-only bitfield is reserved and always has the value zero.
2–0 FLT	I2C programmable filter factor Controls the width of the glitch (in terms of bus clock cycles) that the filter must absorb. In other words, the filter does not allow to pass any glitch whose size is less than or equal to this width setting. 0x0 No filter/bypass 0x1-7 Filter glitches up to width of n bus clock cycles, where n=1-7

10.3.8 I2C SMBus Control and Status Register (I2Cx_SMB_CSR)

NOTE

A master assumes that the bus is free when detecting the clock and data signals being high for greater than high time out period. However, SHTF1 rises in bus transmission process with idle bus state.

NOTE

When TCKSEL is set, there is no meaning to monitor SHTF1 since the bus speed is too high to match the protocol of SMBus.

Addresses: I2C0_SMB_CSR – F210h base + 7h offset = F217h

I2C1_SMB_CSR – F220h base + 7h offset = F227h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Cx_SMB_CSR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
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 SMBus alert response address. 0 Disabled 1 Enabled
5 SIICAEN	Second I2C address enable Enables SMBus device default address 0 Disabled 1 Enabled
4 TCKSEL	Timeout counter clock select Selects the clock source of the timeout counter. 0 Bus clock / 64 frequency 1 Bus clock frequency
3 SLTF	SCL low timeout flag This bit is set when the SLT register (consisting of the SLT1 and SLT2 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 zero. 0 No low timeout occurs 1 Low timeout occurs
2 SHTF1	SCL high timeout flag 1

Table continues on the next page...

I2Cx_SMB_CSR field descriptions (continued)

Field	Description
	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
2 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 a 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 Disabled 1 Enabled

10.3.9 I2C Address Register 2 (I2Cx_ADDR2)

Addresses: I2C0_ADDR2 – F210h base + 8h offset = F218h

I2C1_ADDR2 – F220h base + 8h offset = F228h

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

I2Cx_ADDR2 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–1 SAD	SMBus address Contains the slave address used by the SMBus. This field is used on the device default address or other related addresses.
0 Reserved	This read-only bit is reserved and always has the value zero.

10.3.10 I2C SCL Low Timeout register High (I2Cx_SLT1)

Addresses: I2C0_SLT1 – F210h base + 9h offset = F219h

I2C1_SLT1 – F220h base + 9h offset = F229h

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

I2Cx_SLT1 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 SSLT[15:8]	Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

10.3.11 I2C SCL Low Timeout register Low (I2Cx_SLT2)

Addresses: I2C0_SLT2 – F210h base + Ah offset = F21Ah

I2C1_SLT2 – F220h base + Ah offset = F22Ah

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

I2Cx_SLT2 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 SSLT[7:0]	Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

10.4 Functional Description

This section provides a comprehensive functional description of the I2C module.

10.4.1 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.

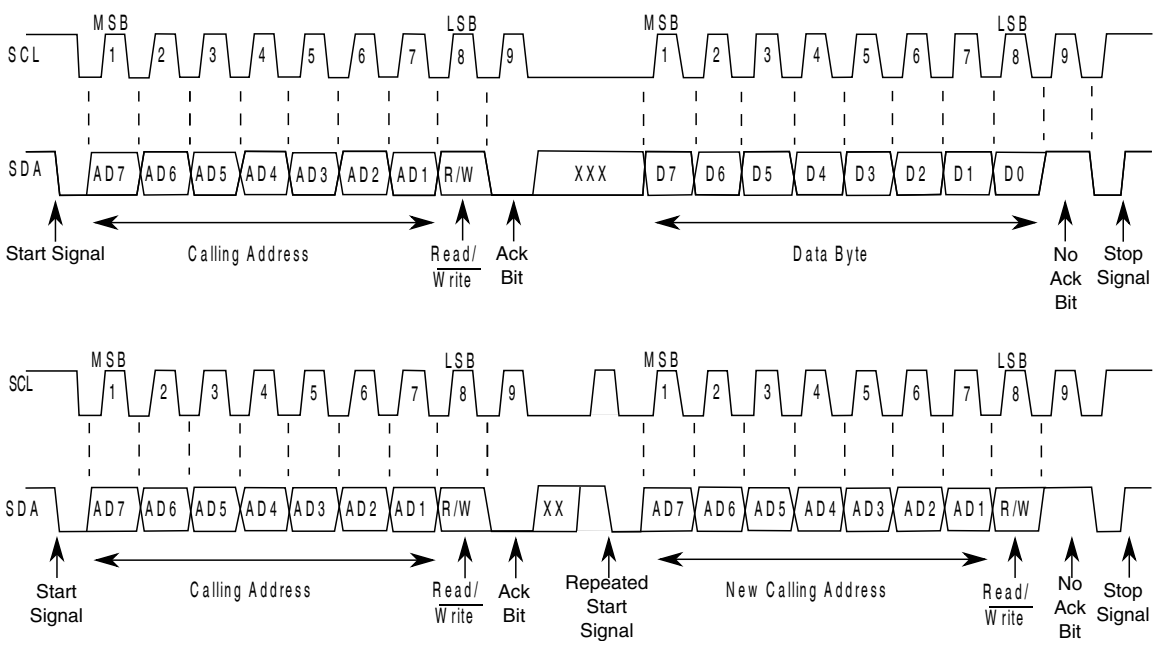


Figure 10-35. I2C Bus Transmission Signals

10.4.1.1 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.

10.4.1.2 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/$\overline{W}$ bit. The R/$\overline{W}$ 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 may 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.

10.4.1.3 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/$\overline{W}$ 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.

10.4.1.4 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.

10.4.1.5 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.

10.4.1.6 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.

10.4.1.7 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.

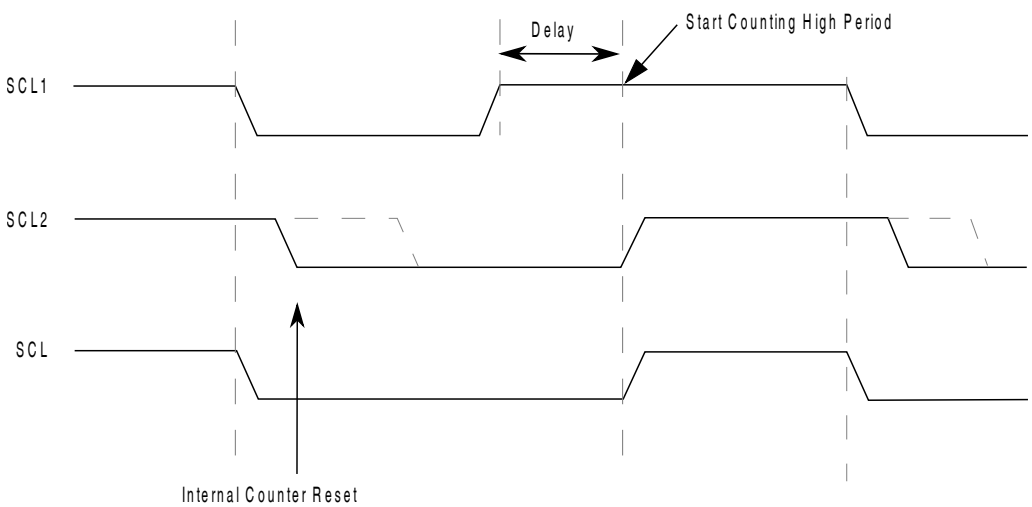


Figure 10-36. I2C Clock Synchronization

10.4.1.8 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.

10.4.1.9 I2C Divider and Hold Values

Table 10-37. 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
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

10.4.2 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.

10.4.2.1 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 seven 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 eight bits of the second byte of the slave address with its own address, but only one slave finds a match and generate 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 10-38. 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.

10.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{W}$ 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/$\overline{W}$) 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/$\overline{W}$) bit. However, none of them are addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

Table 10-39. Master-Receiver Addresses a Slave-Transmitter 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	Sr	Slave address first 7 bits 11110 + AD10 + AD9	R/$\overline{W}$ 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.

10.4.3 Address Matching

All received addresses can be requested in 7-bit or 10-bit address format. The Address Register 1, which contains the I2C primary slave address, always participates in the address matching process. 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, the Address Register 2 participates in the address matching process.

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.

10.4.4 System Management Bus Specification

SMBus provides a control bus for system and power management related tasks. A system may 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 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.

10.4.4.1 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. It is highly recommended that a slave device 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.

10.4.4.1.1 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.

10.4.4.1.2 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 can occur in two ways:

1. HIGH timeout detected after a STOP condition appears on the bus
2. HIGH timeout detected after a START condition, but before a STOP condition appears on the bus

Any master detecting either scenario can assume the bus is free when SHTF1 rises. A HIGH timeout occurs in scenario 2 if a master ever detects that both 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, the other 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 also triggers IICIF.

10.4.4.1.3 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.

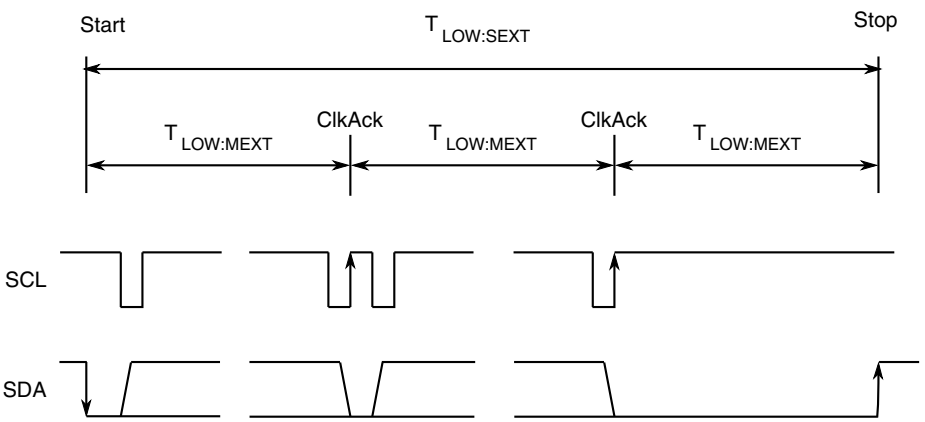


Figure 10-37. Timeout measurement intervals

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.

10.4.4.1.4 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. In order 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.

10.4.5 Resets

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

10.4.6 Interrupts

The I2C module generates an interrupt when any of the events in the following table 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 10-40. 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
SMBus SCL low timeout interrupt flag	SLTF	IICIF	IICIE
SMBus SCL high SDA low timeout interrupt flag	SHTF2	IICIF	IICIE & SHTF2IE
Wakeup from stop interrupt	IAAS	IICIF	IICIE & WUEN

10.4.6.1 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 8th clock to indicate the completion of byte.

10.4.6.2 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.

10.4.6.3 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.

10.4.6.4 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.

10.4.6.5 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.

10.4.6.6 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) bus 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 only specify the size of the glitch (in terms of bus clock cycles) for the filter to absorb and not pass.

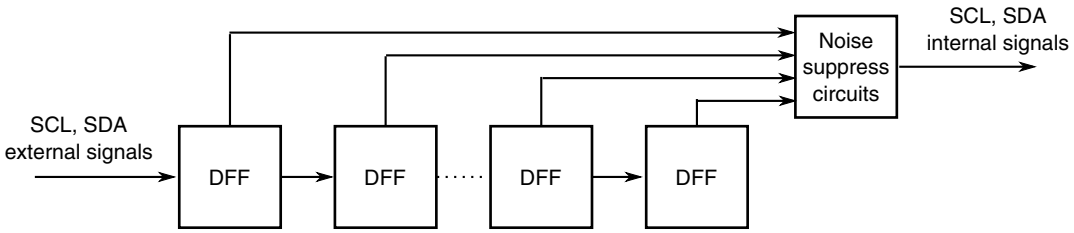


Figure 10-38. Programmable input glitch filter diagram

10.4.6.7 Address Matching Wakeup

When a primary or general call address match occurs when the I2C module is in slave receive mode, the MCU wakes from low power mode with no peripheral bus running. After the address matching IAAS bit is set, an interrupt is sent at the end of address matching to wake the core. The IAAS bit must be cleared after the clock recovery.

NOTE

After the system recovers and is in run mode, restart the I2C module if necessary. The SCL line is not held low until the I2C module resets after address matching.

10.5 Initialization/Application Information

Module Initialization (Slave)

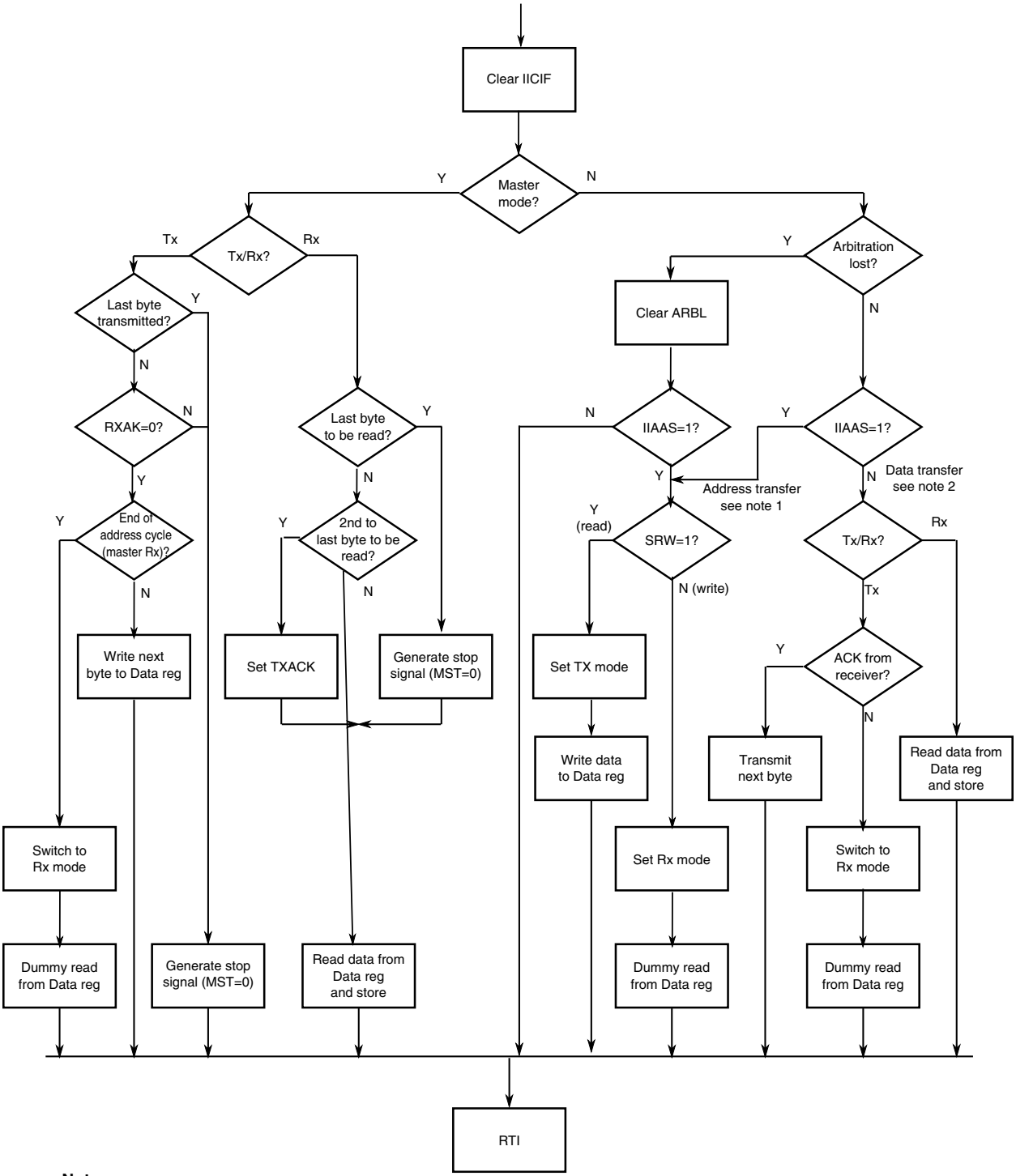
1. Write: Control Register 2
   - to enable or disable general call
   - to select 10-bit or 7-bit addressing mode
2. Write: Address Register 1 to set the slave address
3. Write: Control Register 1 to enable the I2C module and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit 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 (example provided in this chapter)

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)

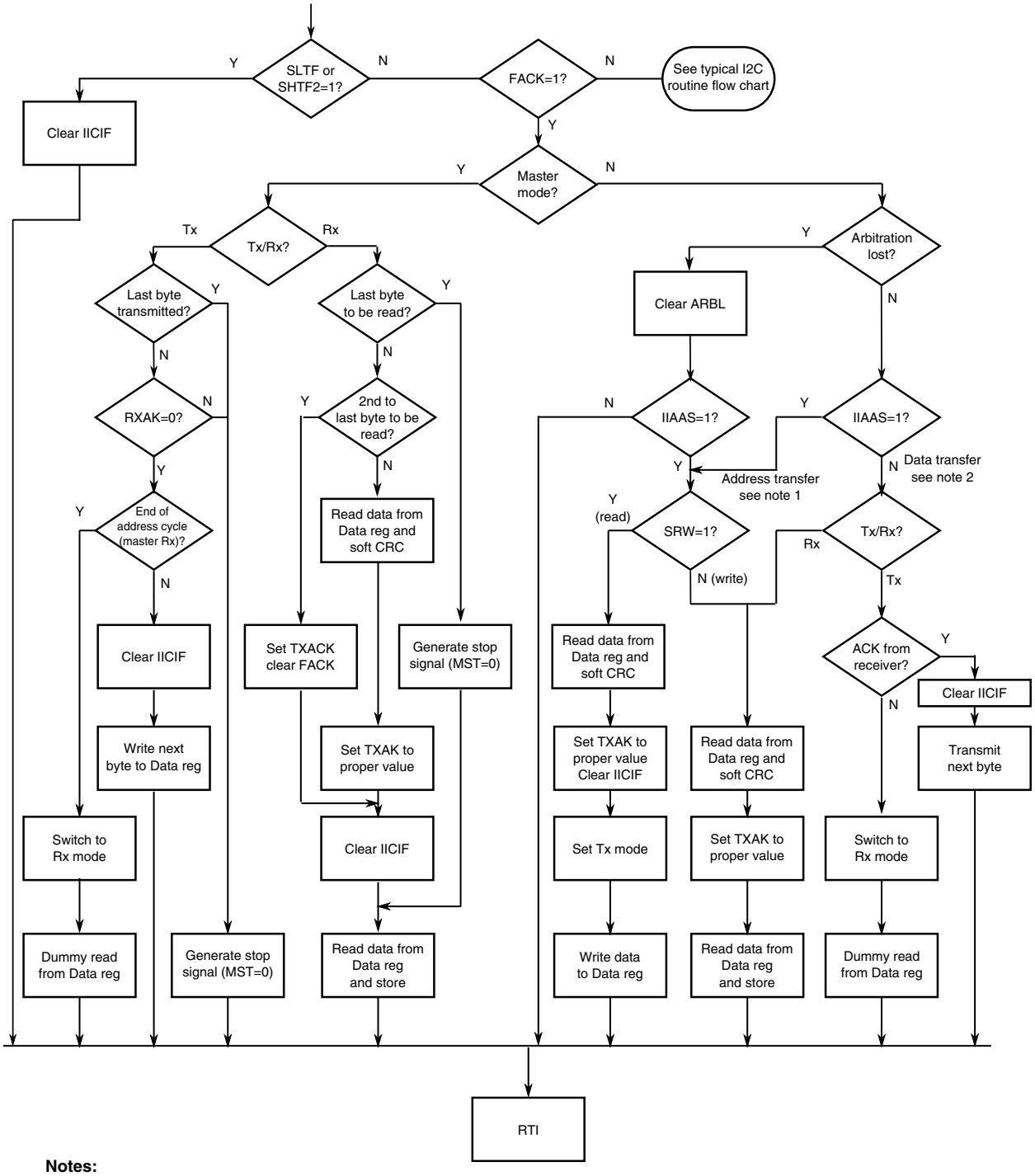
The routine shown in the following figure can handle both master and slave I2C operations. For slave operation, an incoming I2C message that contains the proper address begins I2C communication. For master operation, communication must be initiated by writing the Data register.



Notes:

1. If general call is enabled, check to determine if the received address was a general call address (0x00). If the received address was 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 10-39. Typical I2C Interrupt Routine



Notes:

1. If general call or SIICAEN is enabled, check to determine if the received address was a general call address (0x00) or SMBus device default address. If either, they must be handled by user software.
2. In master receive, complete the second data read after the ninth SCL cycle.

Figure 10-40. Typical I2C SMBus Interrupt Routine

Chapter 11
Queued Serial Communications Interface (QSCI)

11.1 Introduction

The SCI allows asynchronous serial communications with peripheral devices.

11.1.1 Features

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit integer and 3 bit fractional baud rate selection
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter DSC core interrupt requests
- Programmable polarity for transmitter and receiver
- Two receiver wake-up methods: idle line or address mark
- Interrupt-driven operation with seven flags:
  - Transmitter empty
  - Transmitter idle
  - Receiver full
  - Receiver overrun
  - Noise error

- Framing error
- Parity error

- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

11.1.2 SCI Block Diagram

The following figure shows the SCI block diagram.

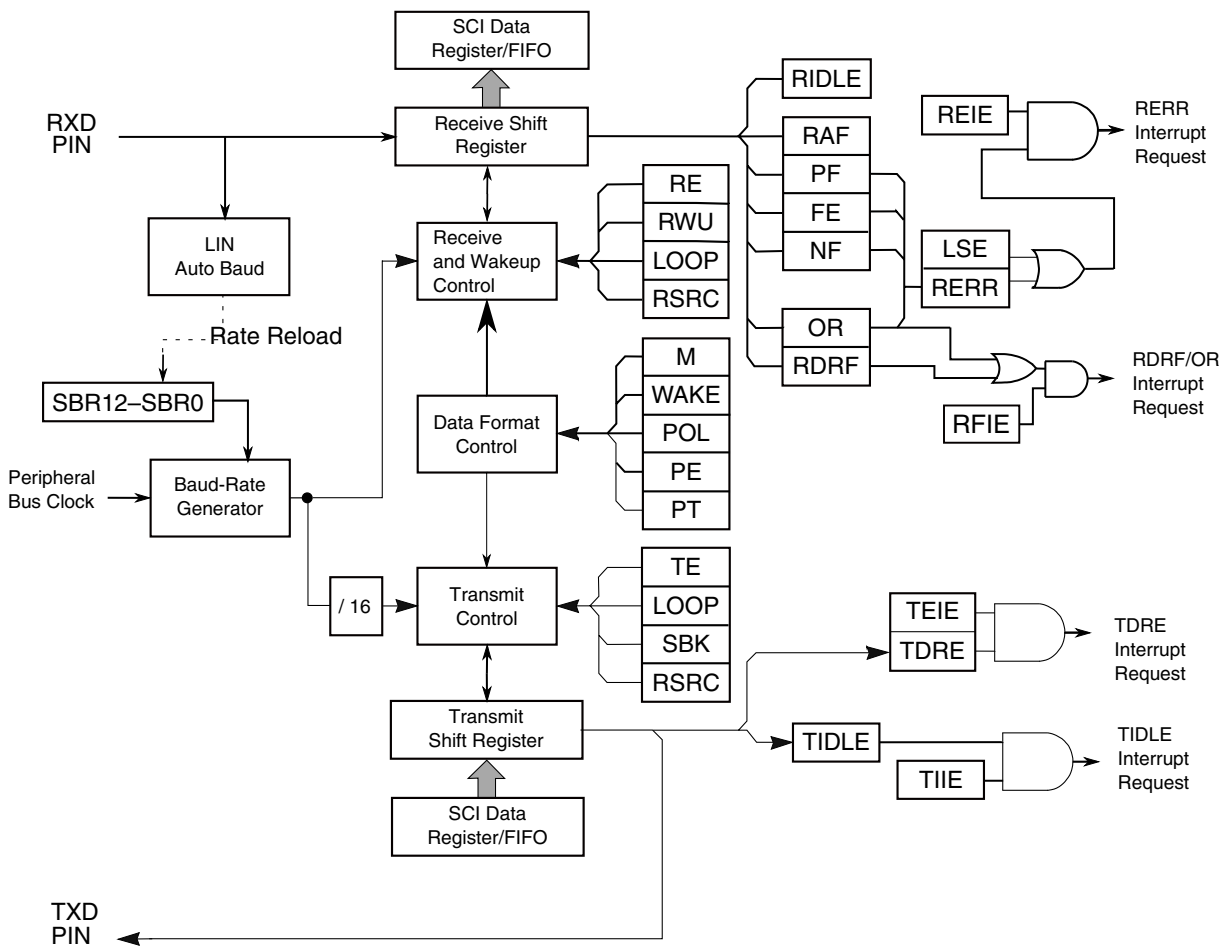


Figure 11-1. SCI Block Diagram

11.2 External Signal Descriptions

Table 11-1. Signal Properties

Name	I/O Type	Function	Reset State
TXD	Output	Transmit Data Pin	1
RXD	Input	Receive Data Pin	—

11.2.1 TXD —Transmit Data

The Transmit Data Pin (TXD) is the SCI transmitter pin. TXD is available for general purpose I/O when it is not configured for transmitter operation, such as when CTRL1[TE] = 0.

11.2.2 RXD —Receiver Data

The Receiver Data Pin (RXD) is the SCI receiver pin. RXD is available for general-purpose I/O when it is not configured for receiver operation, such as when CTRL1[RE] = 0.

11.3 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	QSCI0_RATE	R	SBR													FRAC_SBR		
		W																
1	QSCI0_CTRL1	R	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
		W																
2	QSCI0_CTRL2	R	TFCNT			TFWM		RFCNT			RFWM		FIFO_EN	0	LINMODE	0		
		W																
3	QSCI0_STAT	R	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0			0	LSE	0		RAF
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	QSCI0_DATA	R	0							RECEIVE_TRANSMIT_DATA								
		W																
0	QSCI1_RATE	R	SBR													FRAC_SBR		
		W																
1	QSCI1_CTRL1	R	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
		W																
2	QSCI1_CTRL2	R	TFCNT			TFWM		RFCNT			RFWM		FIFO_EN	0	LINMODE	0		
		W																
3	QSCI1_STAT	R	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0			0	LSE	0		RAF
		W																
4	QSCI1_DATA	R	0							RECEIVE_TRANSMIT_DATA								
		W																

11.3.1 QSCI Baud Rate Register (QSCIx_RATE)

Read: anytime

Write: anytime

Addresses: QSCI0_RATE – F1E0h base + 0h offset = F1E0h

QSCI1_RATE – F1F0h base + 0h offset = F1F0h

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

QSCIx_RATE field descriptions

Field	Description
15–3 SBR	SCI Baud Rate divider, a value from 1 to 8191 Refer to the description of the FRAC_SBR bitfield.
2–0 FRAC_SBR	Fractional SCI Baud Rate divider, a value from 0 to 7 that is divided by 8 The SBR and FRAC_SBR fields combine to form the divider to determine the baud rate of the SCI. SBR represents the integer portion of the baud rate divider, and FRAC_SBR represents the fractional portion.

Table continues on the next page...

QSCIx_RATE field descriptions (continued)

Field	Description
	The FRAC_SBR field can only be used when SBR is greater than 1. Therefore, the value of the divider can be 1.000 or (with fractional values) in the range from 2.000 to 8191.875. The formula for calculating the baud rate is: `Baud rate = peripheral bus clock / (16 * (SBR + (FRAC_SBR / 8)))` **NOTE:** The baud rate generator is disabled until CTRL1[TE] or CTRL1[RE] is set for the first time after reset. The baud rate generator is disabled when RATE[SBR] and RATE[FRAC_SBR] = 0. **NOTE:** If CTRL2[LINMODE] is set, the value of this register is automatically adjusted to match the data rate of the LIN master device. Reading this register yields the auto-baud value set.

11.3.2 QSCI Control Register 1 (QSCIx_CTRL1)

Read: anytime

Write: anytime

Addresses: QSCI0_CTRL1 – F1E0h base + 1h offset = F1E1h

QSCI1_CTRL1 – F1F0h base + 1h offset = F1F1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSCIx_CTRL1 field descriptions

Field	Description
15 LOOP	Loop Select This bit enables loop operation. In loop operation the RXD pin is disconnected from the SCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use the internal loop function as opposed to single wire operation, which requires only one or the other to be enabled. The receiver input is determined by CTRL1[RSRC]. The transmitter output is controlled by CTRL1[TE]. If CTRL1[TE] is set and CTRL1[LOOP]=1, the transmitter output appears on the TXD pin. If CTRL1[TE] is clear and CTRL1[LOOP]=1, the TXD pin is high-impedance. 0 Normal operation, regardless of the value of RSRC 1 When RSRC = 0: Loop mode with internal TXD fed back to RXD 1 When RSRC = 1: Single-wire mode with TXD output fed back to RXD
14 SWAI	Stop in Wait Mode This bit disables the SCI in wait mode

Table continues on the next page...

QSCIx_CTRL1 field descriptions (continued)

Field	Description
	0 SCI enabled in wait mode 1 SCI disabled in wait mode
13 RSRC	Receiver Source When CTRL1[LOOP]=1, CTRL1[RSRC] determines the internal feedback path for the receiver. 0 Receiver input internally connected to transmitter output 1 Receiver input connected to TXD pin
12 M	Data Format Mode This bit determines whether data characters are eight or nine bits long. 0 One start bit, eight data bits, one stop bit 1 One start bit, nine data bits, one stop bit
11 WAKE	Wake-up Condition This bit determines which condition wakes the SCI: a one (address mark) in the most significant bit position of a received data character or an idle condition on the RXD pin. 0 Idle line wake-up 1 Address mark wake-up
10 POL	Polarity This bit determines whether to invert the data as it goes from the transmitter to the TXD pin and from the RXD pin to the receiver. All bits (start, data, and stop) are inverted as they leave the transmit shift register and before they enter the receive shift register. **NOTE:** It is recommended that CTRL1[POL] be toggled only when CTRL1[TE]=0 and CTRL1[RE]=0. 0 Don't invert transmit and receive data bits (normal mode) 1 Invert transmit and receive data bits (inverted mode)
9 PE	Parity Enable This bit enables the parity function. When enabled, the parity function replaces the most significant bit of the data character with a parity bit. 0 Parity function disabled 1 Parity function enabled
8 PT	Parity Type This bit determines whether the SCI generates and checks for even parity or odd parity of the data bits. With even parity, an even number of ones clears the parity bit and an odd number of ones sets the parity bit. With odd parity, an odd number of ones clears the parity bit and an even number of ones sets the parity bit. 0 Even parity 1 Odd parity
7 TEIE	Transmitter Empty Interrupt Enable This bit enables the transmit data register empty flag, STAT[TDRE], to generate interrupt requests.

Table continues on the next page...

QSCIx_CTRL1 field descriptions (continued)

Field	Description
	0 STAT[TDRE] interrupt requests disabled 1 STAT[TDRE] interrupt requests enabled
6 TIIE	Transmitter Idle Interrupt Enable This bit enables the transmitter idle flag, STAT[TIDLE], to generate interrupt requests. 0 STAT[TIDLE] interrupt requests disabled 1 STAT[TIDLE] interrupt requests enabled
5 RFIE	Receiver Full Interrupt Enable This bit enables the receive data register full flag, STAT[RDRF], or the overrun flag, STAT[OR], to generate interrupt requests. 0 STAT[RDRF] and STAT[OR] interrupt requests disabled 1 STAT[RDRF] and STAT[OR] interrupt requests enabled
4 REIE	Receive Error Interrupt Enable This bit enables the receive error flags (STAT[NF], STAT[PF], STAT[FE], and STAT[OR]) to generate interrupt requests. 0 Error interrupt requests disabled 1 Error interrupt requests enabled
3 TE	Transmitter Enable This bit enables the SCI transmitter and configures the TXD pin as the SCI transmitter output. CTRL1[TE] can be used to queue an idle preamble. 0 Transmitter disabled 1 Transmitter enabled
2 RE	Receiver Enable This bit enables the SCI receiver. 0 Receiver disabled 1 Receiver enabled
1 RWU	Receiver Wake-up This bit enables the wake-up function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing CTRL1[RWU]. 0 Normal operation 1 Standby state
0 SBK	Send Break Toggling this bit sends one break character (10 or 11 zeroes). As long as this bit is set, the transmitter sends zeroes. 0 No break characters 1 Transmit break characters

11.3.3 QSCI Control Register 2 (QSCIx_CTRL2)

Read: anytime

Write: anytime

Addresses: QSCI0_CTRL2 – F1E0h base + 2h offset = F1E2h

QSCI1_CTRL2 – F1F0h base + 2h offset = F1F2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TFCNT			TFWM		RFCNT			RFWM		FIFO_EN	0	LINMODE	0		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSCIx_CTRL2 field descriptions

Field	Description
15–13 TFCNT	Transmit FIFO Count These read only bits show how many words are used in the TX FIFO. Writes to DATA cause CTRL2[TFCNT] to increment. As words are pulled for transmission, CTRL2[TFCNT] decrements. Attempts to write new data to DATA are ignored when CTRL2[TFCNT] indicates the FIFO is full (4 words). 000 0 words in Tx FIFO 001 1 word in Tx FIFO 010 2 words in Tx FIFO 011 3 words in Tx FIFO 100 4 words in Tx FIFO 101 Reserved 110 Reserved 111 Reserved
12–11 TFWM	Transmit FIFO Empty Water Mark These bits set the TX FIFO word count level at which STAT[TDRE] is set. If CTRL2[FIFO_EN] is clear (FIFO disabled), then this field is forced to 00 and STAT[TDRE] is set when there is no word in the transmit buffer. 00 TDRE is set when 0 words are in the FIFO 01 TDRE is set when 1 or fewer words are in the FIFO 10 TDRE is set when 2 or fewer words are in the FIFO 11 TDRE is set when 3 or fewer words are in the FIFO
10–8 RFCNT	Receive FIFO Count These read only bits show how many words are used in the RX FIFO. As words are received, CTRL2[RFCNT] is incremented. As words are read from DATA the value of CTRL2[RFCNT] decrements. There is one word time to read DATA between when STAT[RDRF] is set (interrupt asserted), due to the RX FIFO being full, and when an overflow condition is flagged.

Table continues on the next page...

QSCIx_CTRL2 field descriptions (continued)

Field	Description
	000 0 words in RX FIFO 001 1 word in RX FIFO 010 2 words in RX FIFO 011 3 words in RX FIFO 100 4 words in RX FIFO 101 Reserved 110 Reserved 111 Reserved
7–6 RFWM	Receive FIFO Full Water Mark These bits set the RX FIFO word count level at which STAT[RDRF] is set. If CTRL2[FIFO_EN] is clear (FIFO disabled), then this field is forced to 00 and STAT[RDRF] is set if there is a word in the receive buffer. 00 RDRF is set when at least 1 word is in the FIFO 01 RDRF is set when at least 2 words are in the FIFO 10 RDRF is set when at least 3 words are in the FIFO 11 RDRF is set when at least 4 words are in the FIFO
5 FIFO_EN	FIFO Enable This read/write bit enables the 4-word-deep TX and RX FIFOs. Change this bit only when the SCI is idle. 0 FIFOs are disabled. 1 FIFOs are enabled.
4 Reserved	This read-only bit is reserved and always has the value zero.
3 LINMODE	Enable LIN Slave Mode This bit should be used only in Local Interconnect Network (LIN) applications. **NOTE:** During initialization, the RATE register should be loaded to a value that is within 15% of the actual master data rate; otherwise, 0x00 data might be misinterpreted as a break. **NOTE:** If the first character following a break is not the LIN sync character (0x55), the RATE register is not adjusted and STAT[LSE] is set. 0 The LIN auto baud feature is disabled and the RATE register maintains whatever value the processor writes to it. 1 Enable LIN slave functionality. This includes a search for the break character followed by a sync character (0x55) from the master LIN device. When the break is detected (11 consecutive samples of zero), the subsequent sync character is used to measure the baud rate of the transmitting master, and the RATE register is automatically reloaded with the value needed to "match" that baud rate.
2–0 Reserved	This read-only bitfield is reserved and always has the value zero.

11.3.4 QSCI Status Register (QSCIx_STAT)

Read: anytime

Write: this register is not writable

Addresses: QSCI0_STAT – F1E0h base + 3h offset = F1E3h

QSCI1_STAT – F1F0h base + 3h offset = F1F3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0			0	LSE	0		RAF
Write																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSCIx_STAT field descriptions

Field	Description
15 TDRE	Transmit Data Register Empty Flag This bit is set when the TX FIFO word count (CTRL2[TFCNT]) falls to the watermark level (CTRL2[TXWM]). Clear TDRE by reading STAT with TDRE set and then writing to the SCI data register until CTRL2[TFCNT] is above CTRL2[TFWM]. If CTRL2[FIFO_EN] or CTRL2[TDE] is set, then you can clear TDRE by writing to the SCI data register without first reading STAT with TDRE set. 0 TX FIFO word count is above watermark 1 TX FIFO word count is at or below watermark
14 TIDLE	Transmitter Idle Flag This bit is set when the TX FIFO is empty and no data, preamble, or break character is being transmitted. When TIDLE is set, the TXD pin becomes idle (1). Clear TIDLE by reading STAT with TIDLE set and then writing to the SCI data register. TIDLE is not generated when a data character, a preamble, or a break is queued and ready to be sent. 0 Transmission in progress 1 No transmission in progress
13 RDRF	Receive Data Register Full Flag This bit is set when the RX FIFO word count (CTRL2[RFCNT]) rises above the watermark (CTRL2[RXWM]). Clear RDRF by reading STAT with RDRF set and then reading the SCI data register until CTRL2[RFCNT] is no longer above CTRL2[RFWM]. If CTRL2[FIFO_EN] or CTRL2[RDE] is set, then you can clear RDRF by reading the SCI data register without first reading STAT with RDRF set. **NOTE:** When you are using the CodeWarrior debugger, RDRF may be erased when a breakpoint is reached. If a memory window that includes the SCI registers is open when a breakpoint is reached, these memory addresses are read to update the memory window. If RDRF is set at this time, these reads satisfy the requirements for clearing RDRF because the status register is read with RDRF set and then the data register is read, which causes RDRF to clear. 0 RX FIFO word count is at or below watermark 1 RX FIFO word count is above watermark
12 RIDLE	Receiver Idle Line Flag This bit is set when 10 consecutive ones (if CTRL1[M]=0) or 11 consecutive ones (if CTRL1[M]=1) appear on the receiver input. The RIDLE flag is cleared by reading STAT with RIDLE set and then reading the SCI data register. Once RIDLE is cleared, a valid frame must be received before an idle condition can set RIDLE. **NOTE:** When the receiver wake-up bit (CTRL1[RWU]) is set, an idle line condition does not set RIDLE.

Table continues on the next page...

QSCIx_STAT field descriptions (continued)

Field	Description
	0 Receiver input is either active now or has never become active since RIDLE was last cleared 1 Receiver input has become idle (after receiving a valid frame)
11 OR	Overrun Flag This bit is set when software fails to read the SCI data register when the RX FIFO is full before the receive shift register receives the next frame. The data in the shift register is lost, but the data already in the SCI data register/FIFO is not affected. Clear OR by reading STAT with OR set and then writing the SCI status register with any value. 0 No overrun 1 Overrun
10 NF	Noise Flag This bit is set when the SCI detects noise on the receiver input. NF is set during the same cycle as RDRF but is not set in the case of an overrun. Clear NF by reading STAT and then writing the SCI status register with any value. 0 No noise 1 Noise
9 FE	Framing Error Flag This bit is set when a 0 is accepted as the stop bit. FE is set during the same cycle as RDRF but is not set in the case of an overrun. Clear FE by reading STAT with FE set and then writing the SCI status register with any value. 0 No framing error 1 Framing error
8 PF	Parity Error Flag This bit is set when the parity enable bit (CTRL1[PE]) is set and the parity of the received data does not match its parity bit. Clear PF by reading STAT and then writing the SCI status register with any value. 0 No parity error 1 Parity error
7–5 Reserved	This read-only bitfield is reserved and always has the value zero.
4 Reserved	This read-only bit is reserved and always has the value zero.
3 LSE	LIN Sync Error This bit is active only when CTRL2[LINMODE] is set. When LSE is set, a Receive Error interrupt occurs if CTRL1[REIE] is set. LSE is set when a LIN sync search detects a non-sync character (anything other than 0x55). When set, this bit indicates either that a protocol error was detected from the Master LIN device or there is a gross mismatch in data rates. This bit is cleared by reading STAT with LSE set and then writing the SCI status register with any value. 0 No error occurred since CTRL2[LINMODE] was enabled or the bit was last cleared 1 A sync error prevented loading of the RATE register with a revised value after the break was detected.

Table continues on the next page...

QSCIx_STAT field descriptions (continued)

Field	Description
2–1 Reserved	This read-only bitfield is reserved and always has the value zero.
0 RAF	Receiver Active Flag This bit is set when the receiver detects a 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects false start bits (usually from noise or baud rate mismatch) or when the receiver detects a preamble. 0 No reception in progress 1 Reception in progress

11.3.5 QSCI Data Register (QSCIx_DATA)

Read: anytime. Reading accesses the SCI receive data register.

Write: anytime. Writing accesses the SCI transmit data register.

Addresses: QSCI0_DATA – F1E0h base + 4h offset = F1E4h

QSCI1_DATA – F1F0h base + 4h offset = F1F4h

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

QSCIx_DATA field descriptions

Field	Description
15–9 Reserved	This read-only bitfield is reserved and always has the value zero.
8–0 RECEIVE_ TRANSMIT_ DATA	RECEIVE_DATA: Received data (when reading this register) TRANSMIT_DATA: Data to be transmitted (when writing this register) If STAT[TDRE] is set, writes to the transmit data field are ignored unless STAT has been read with STAT[TDRE] set. However, when CTRL2[FIFO_EN] or CTRL2[TDE] are set, you can write to the SCI data register without first reading STAT with STAT[TDRE] set. If STAT[RDRF] is set, reads from the receive data field are ignored unless STAT has been read with STAT[RDRF] set. However, when CTRL2[FIFO_EN] or CTRL2[RDE] are set, you can read from the SCI data register without first reading STAT with STAT[RDRF] set. **NOTE:** When the data format is configured for 8-bit data, bits 7 to 0 contain the data.

11.4 Functional Description

The SCI block diagram shows the structure of the SCI module. The SCI allows full duplex, asynchronous, NRZ serial communication between the DSP and remote devices, including other DSPs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The DSC core monitors the status of the SCI, writes the data to be transmitted, and processes received data.

When initializing the SCI, be sure to set the proper peripheral enable bits in the GPIO as well as any pullup enables.

11.4.1 Data Frame Format

The SCI uses the standard NRZ mark/space data frame format illustrated in the following figure.

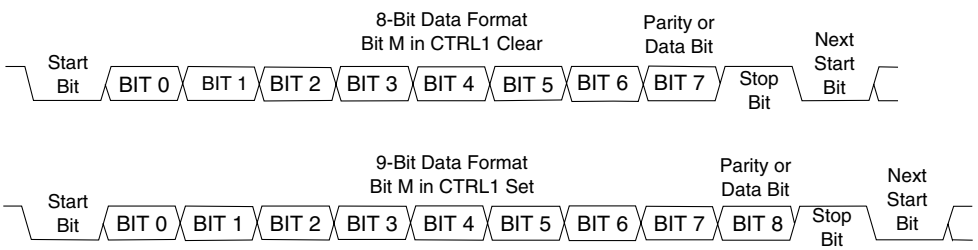


Figure 11-17. SCI Data Frame Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing CTRL1[M] configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits, with formats as shown in the following table.

Table 11-20. Example 8-Bit Data Frame Formats

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1[1]	0	1

1. The address bit identifies the frame as an address character.

Setting CTRL1[M] configures the SCI for 9-bit data characters. A frame with nine data bits has a total of 11 bits with formats as shown in the following table.

Table 11-21. Example 9-Bit Data Frame Formats

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	9	0	0	1
1	8	0	0	2
1	8	0	1	1
1	8	1[1]	0	1

1. The address bit identifies the frame as an address character.

11.4.2 Baud-Rate Generation

A 13-bit modulus counter in the baud-rate generator derives the baud rate for both the receiver and the transmitter. The value written to the RATE[SBR] and RATE[FRAC_SBR] bits determines the peripheral bus clock divisor. The baud rate clock is synchronized with the IP Bus clock and drives the receiver. The baud rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time.

Baud-rate generation is subject to two sources of error:

- Integer division of the peripheral bus clock may not give the exact target frequency.
- Synchronization with the bus clock can cause phase shift.

The following table lists some examples of achieving target baud rates with a peripheral bus clock frequency of 60 MHz.

Table 11-22. Example Baud Rates (Peripheral Bus Clock = 60 MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
32.5	1,846,154	115,385	115,200	0.16
65.125	921,305	57,582	57,600	-0.03
97.625	614,597	38,412	38,400	0.03
195.25	307,298	19,206	19,200	0.03
390.625	153,600	9,600	9600	0.00
781.25	76,800	4,800	4800	0.00
1562.5	38,400	2,400	2400	0.00
3125	19,200	1,200.0	1200	0.00
6250	9,600	600.0	600	0.00

The following table lists some examples of achieving target baud rates with a peripheral bus clock frequency of 32 MHz.

Table 11-23. Example Baud Rates (Peripheral Bus Clock = 32 MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
17.375	1,841,727	115,108	115,200	-0.08
34.75	920,863	57,554	57,600	-0.08
52.125	613,909	38,369	38,400	-0.08
104.125	307,323	19,208	19,200	0.04
208.375	153,569	9,598	9,600	-0.02
416.625	76,808	4,800	4,800	0.01
833.375	38,398	2,400	2,400	-0.01
1666.625	19,200	1,200	1,200	0.00
3333.375	9,600	600	600	0.00

Note

Maximum baud rate is peripheral bus clock rate divided by 16. System overhead may preclude processing the data at this speed.

11.4.3 Transmitter

The following figure is a block diagram of the transmitter functions.

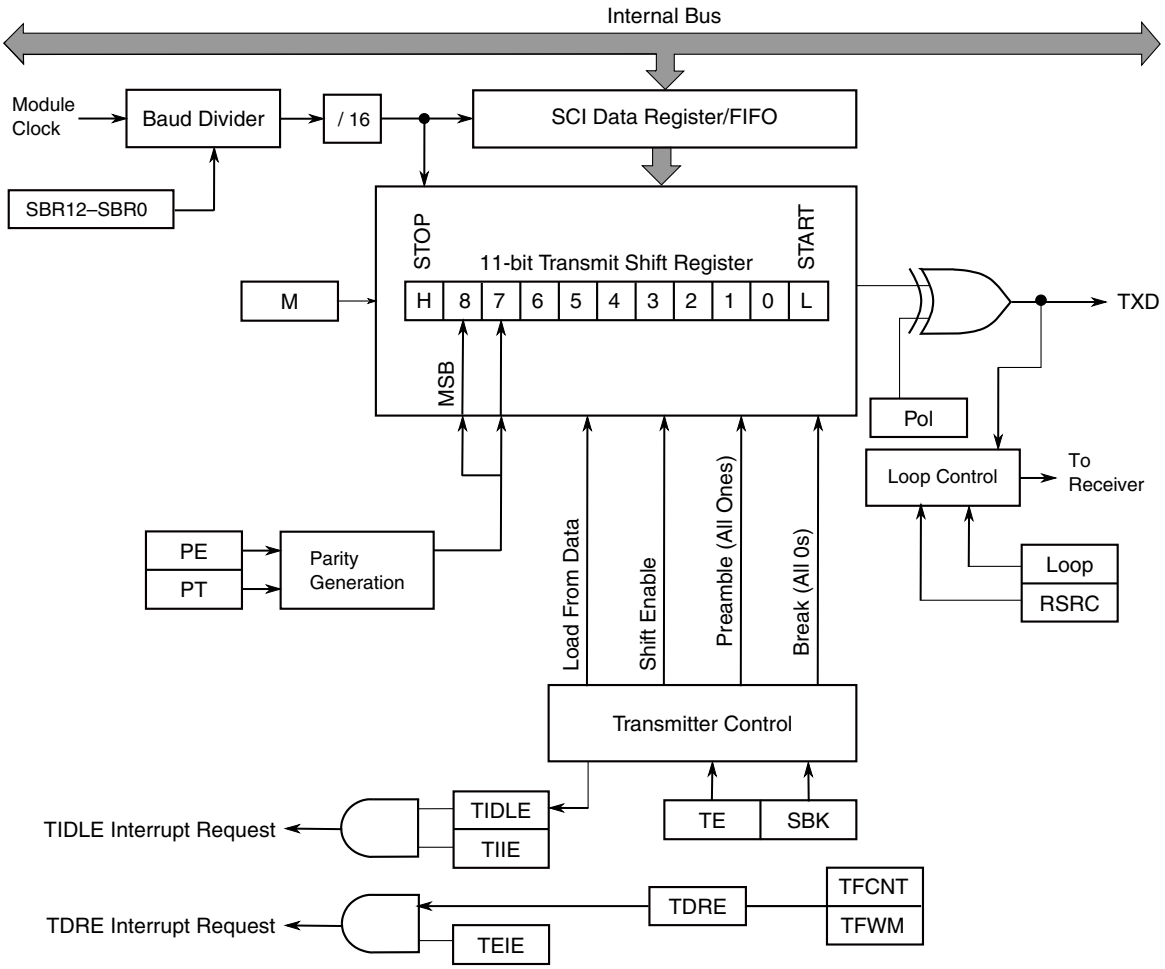


Figure 11-18. SCI Transmitter Block Diagram

11.4.3.1 Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of CTRL1[M] determines the length of data characters.

11.4.3.2 Character Transmission

During an SCI transmission, the transmit shift register shifts a frame out to the TXD pin. The SCI data register is the write-only buffer between the internal data bus and the transmit shift register.

To initiate an SCI transmission:

1. Enable the transmitter by writing a logic 1 to the transmitter enable bit (CTRL1[TE]).

2. Clear the transmit data register empty flag, STAT[TDRE], by first reading the SCI status register (STAT) and then writing to the SCI data register (DATA).
3. Repeat step 2 for each subsequent transmission.

Writing CTRL1[TE] bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 ones (if CTRL1[M] = 0) or 11 ones (if CTRL1[M] = 1). After the preamble shifts out, control logic automatically transfers the data from the SCI data register into the transmit shift register. A logic zero start bit automatically goes into the least significant bit position of the transmit shift register. A logic one stop bit goes into the most significant bit (MSB) position of the frame.

Hardware supports odd or even parity. When parity is enabled, the MSB of the data character is replaced by the parity bit.

The transmit data register empty flag, STAT[TDRE], becomes set when the SCI data register transfers a character to the transmit shift register. STAT[TDRE] indicates that the SCI data register can accept new data from the internal data bus. If the transmitter empty interrupt enable bit, CTRL1[TEIE], is also set, STAT[TDRE] generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame and CTRL1[TE]=1, the TXD pin goes to the idle condition, logic one. If at any time software clears CTRL1[TE], the transmitter relinquishes control of the port I/O pin upon completion of the current transmission, causing the TXD pin to go to a high z state.

If software clears CTRL1[TE] while a transmission is in progress (STAT[TIDLE] = 0), the frame in the transmit shift register continues to shift out. Then transmission stops even if there is data pending in the SCI data register. To avoid accidentally cutting off the last frame in a message, always wait for STAT[TDRE] to go high after the last frame before clearing CTRL1[TE].

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last character of the first message to the DATA register.
2. Wait for STAT[TDRE] to go high while CTRL2[TFWM] = 00, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting CTRL1[TE].
4. Write the first character of the second message to the DATA register.

11.4.3.3 Break Characters

Writing a logic one to the send break bit, CTRL1[SBK], loads the transmit shift register with a break character. A break character contains all logic zeroes and has no start, stop, or parity bit. Break character length depends on CTRL1[M]. As long as CTRL1[SBK] is at logic one, transmitter logic continuously loads break characters into the transmit shift register. After software clears CTRL1[SBK], the shift register finishes transmitting the last break character and then transmits at least one logic one. The automatic logic one at the end of the last break character guarantees the recognition of the start bit of the next frame.

The SCI recognizes a break character when a start bit is followed by eight or nine logic zero data bits and a logic zero where the stop bit should be. Receiving a break character has these effects on SCI registers:

- Sets the framing error flag, STAT[FE]
- Sets the receive data register full flag, STAT[RDRF], if CTRL2[RFWM] = 00
- Clears the SCI data register
- May set the overrun flag (STAT[OR]), noise flag (STAT[NF]), parity error flag (STAT[PF]), or receiver active flag (STAT[RAF])

11.4.3.4 Preambles

A preamble contains all logic ones and has no start, stop, or parity bit. Preamble length depends on CTRL1[M]. The preamble is a synchronizing mechanism that begins the first transmission initiated after writing CTRL1[TE] from 0 to 1.

If CTRL1[TE] is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting CTRL1[TE] during a transmission queues a preamble to be sent after the frame currently being transmitted.

Note

Toggle CTRL1[TE] for a queued preamble when STAT[TDRE] becomes set and immediately before writing the next character to the DATA register.

When queueing a preamble, return CTRL1[TE] to logic one before the stop bit of the current frame shifts out to the TXD pin. Setting CTRL1[TE] after the stop bit appears on TXD causes data previously written to the SCI data register to be lost.

11.4.4 Receiver

The following figure is the block diagram of the SCI receiver.

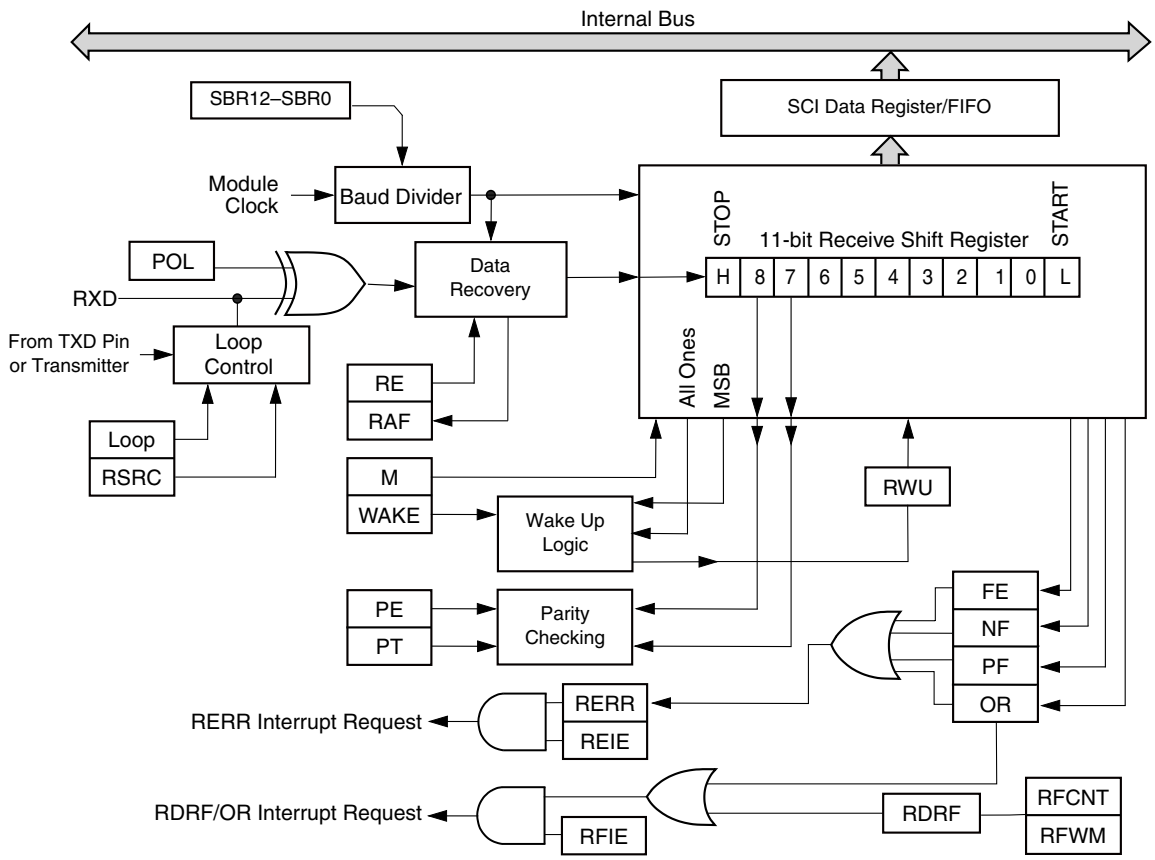


Figure 11-19. SCI Receiver Block Diagram

11.4.4.1 Character Length

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of CTRL1[M] determines the length of data characters.

11.4.4.2 Character Reception

During an SCI reception, the receive shift register shifts a frame in from the RXD pin. The SCI data register/FIFO is the read-only buffer between the internal data bus and the receive shift register.

After a complete frame shifts into the receive shift register, the data portion of the frame along with the STAT[FE], STAT[NF], STAT[PF], and STAT[LSE] status flags transfer to the SCI data register. The receive data register full flag, STAT[RDRF], becomes set when the RX FIFO word count is above the watermark, indicating that a received character can be read. When the FIFO is enabled, there can be received data words to be read even if STAT[RDRF] is not set. If the receive interrupt enable bit, CTRL1[RFIE], is also set, STAT[RDRF] generates a Receiver Full interrupt request.

The STAT[FE], STAT[NF], STAT[PF], and STAT[LSE] flags are associated with the current character to be read from the receive data register/FIFO.

11.4.4.3 Data Sampling

The receiver samples the RXD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (shown in the following figure) is resynchronized:

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after most data bit samples at RT8, RT9, and RT10 return a valid logic 1 and most of the next RT8, RT9, and RT10 samples return a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic ones. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

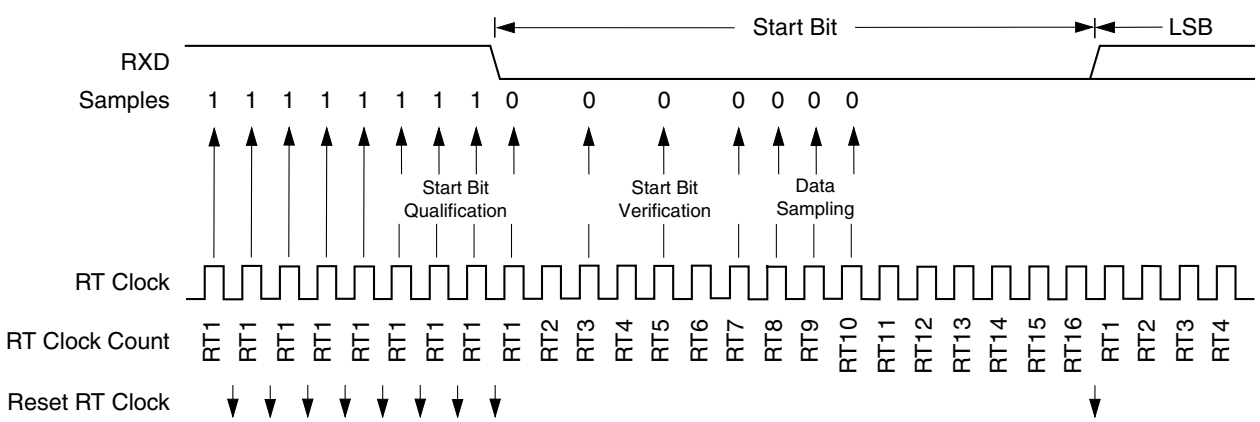


Figure 11-20. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. The following table summarizes the results of the start bit verification samples.

Table 11-24. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the data bit samples.

Table 11-25. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

Note

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic ones following a successful start bit verification, the noise flag (STAT[NF]) is set and the receiver assumes that the bit is a start bit (logic zero).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the stop bit samples.

Table 11-26. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

In the following figure, the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. The RT clock is reset and the start bit search begins again. The noise flag is not set because the noise occurred before the start bit was found.

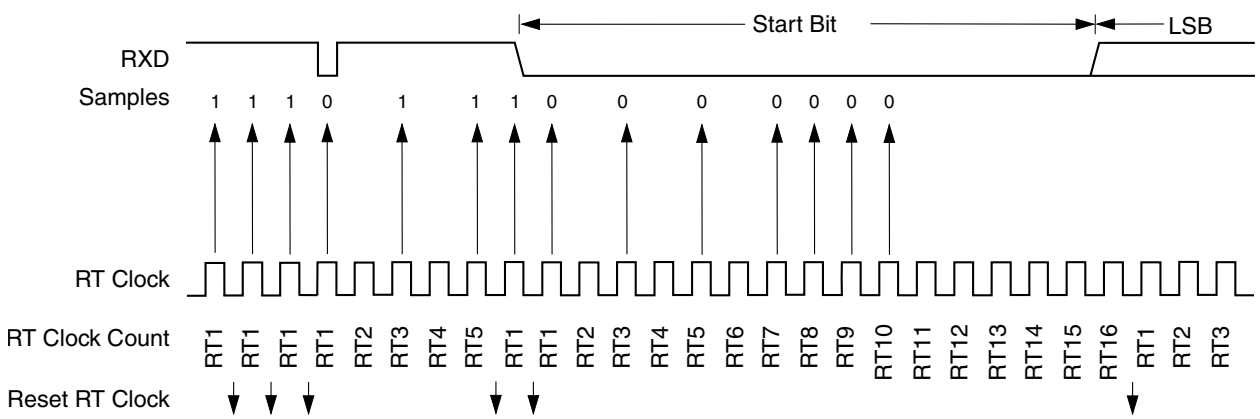


Figure 11-21. Start Bit Search Example 1

In the following figure, noise is perceived as the beginning of a start bit because the verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

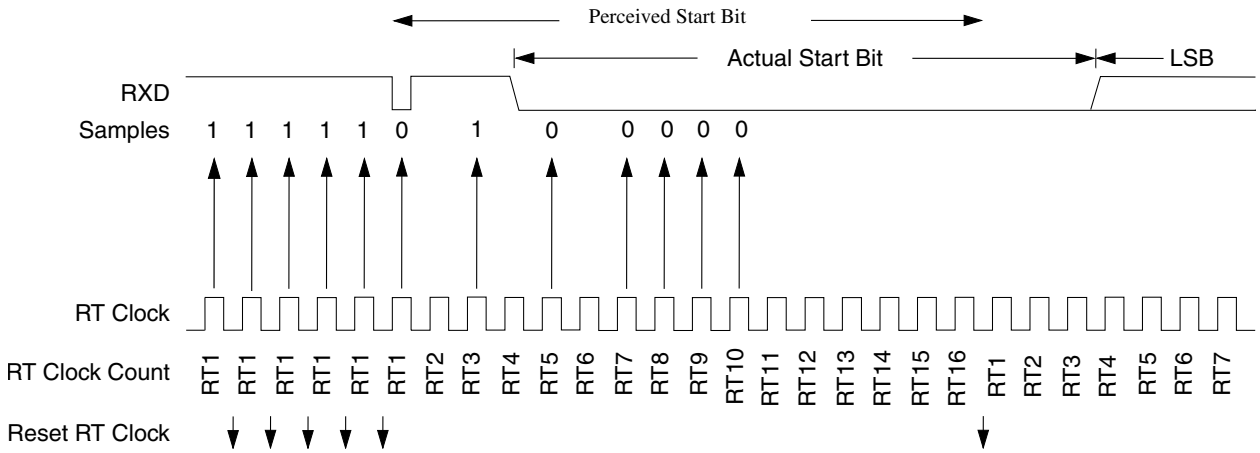


Figure 11-22. Start Bit Search Example 2

In the following figure, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

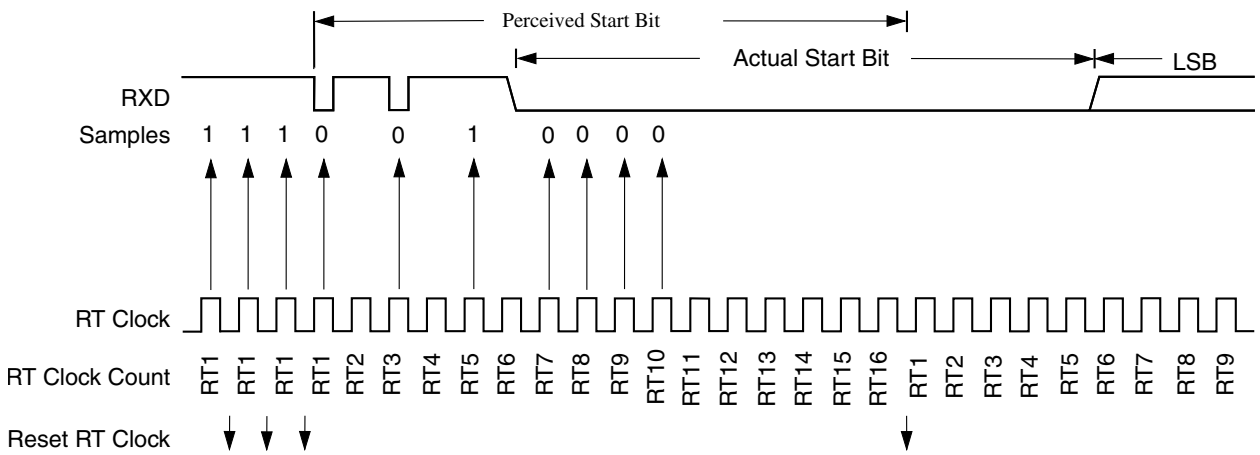


Figure 11-23. Start Bit Search Example 3

The following figure shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

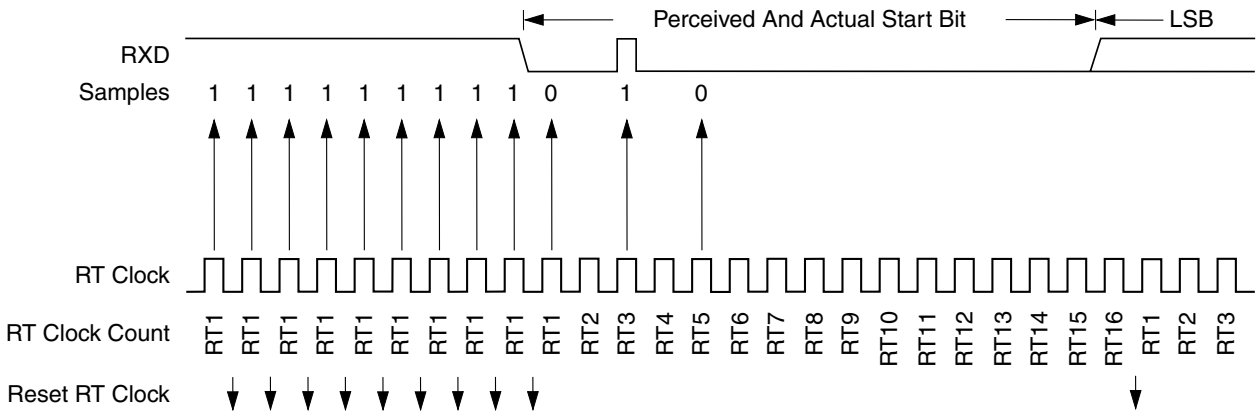


Figure 11-24. Start Bit Search Example 4

The following figure shows a burst of noise near the beginning of the start bit that resets the RT clock. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

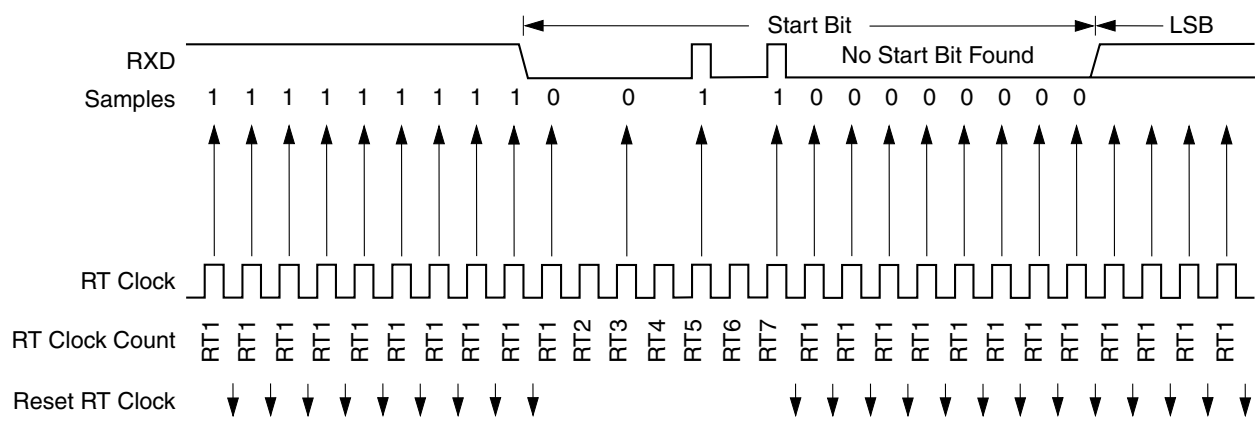


Figure 11-25. Start Bit Search Example 5

In the following figure, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored.

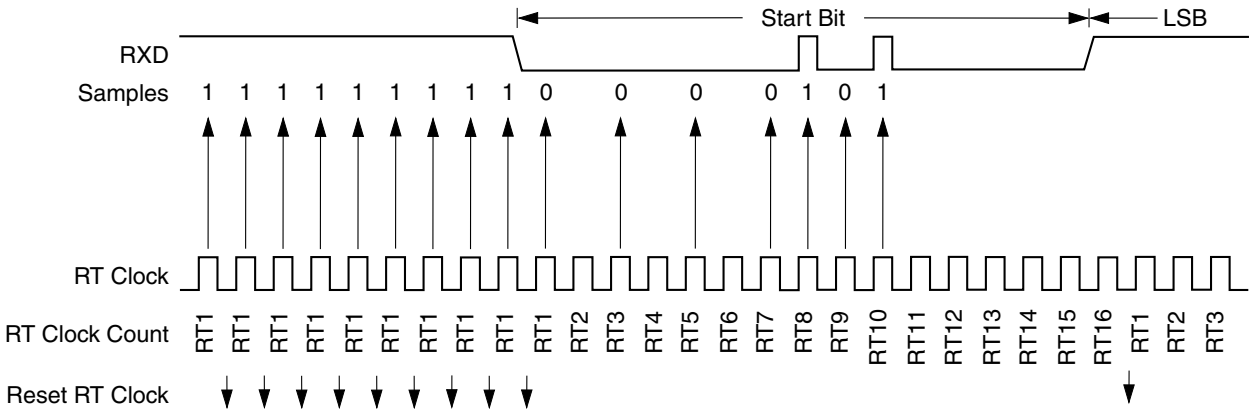


Figure 11-26. Start Bit Search Example 6

11.4.4.4 Framing Errors

If the data recovery logic does not detect a logic one where the stop bit should be in an incoming frame, it sets the framing error flag, STAT[FE]. A break character also sets STAT[FE] because a break character has no stop bit. STAT[FE] is set at the same time that STAT[RDRF] is set.

11.4.4.5 Baud-Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects misalignments between transmitter bit times and receiver bit times.

11.4.4.6 Slow Data Tolerance

The following figure 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.

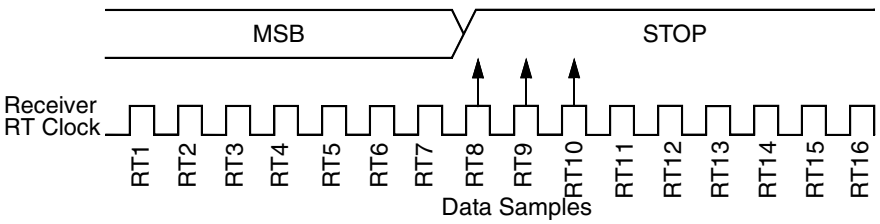


Figure 11-27. Slow Data

For an 8-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
9 bit x 16 RT cycles + 10 RT cycles = 154 RT cycles
```

With the misaligned character shown in the data figure, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

```
9 bit 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 character with no errors is:

```
((154-147) / 154) = 4.54%
```

For a 9-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
10 bit x 16 RT cycles + 10 RT cycles = 170 RT cycles
```

With the misaligned character shown in the slow data figure, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

```
10 bit 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 character with no errors is:

```
((170-163) / 170) = 4.12%
```

11.4.4.7 Fast Data Tolerance

The following figure shows how much a fast received frame can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

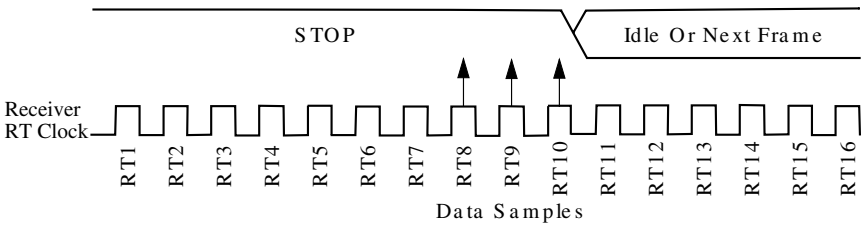


Figure 11-28. Fast Data

For an 8-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
9 bit x 16 RT cycles + 10 RT cycles = 154 RT cycles
```

With the misaligned character shown in the fast data figure, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

```
10 bit x 16 RT cycles = 160 RT cycles
```

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

```
((154-160) / 154) = 3.90%
```

For a 9-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
10 bit x 16 RT cycles + 10 RT cycles = 170 RT cycles
```

With the misaligned character shown in the fast data figure, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

```
11 bit x 16 RT cycles = 176 RT cycles
```

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

```
((170-176) / 170) = 3.53%
```

11.4.4.8 Receiver Wakeup

So that the SCI can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, CTRL1[RWU], puts the receiver into a standby state during which receiver interrupts are disabled.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message.

CTRL1[WAKE] determines how the SCI is brought out of the standby state to process an incoming message. CTRL1[WAKE] enables either idle line wakeup or address mark wakeup:

- Idle input line wakeup (CTRL1[WAKE] = 0): In this wakeup method, an idle condition on the RXD pin clears CTRL1[RWU] and wakes the SCI. The initial frame (or frames) of every message contains addressing information. All receivers evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its CTRL1[RWU] bit and return to the standby state. CTRL1[RWU] remains set and the receiver remains on standby until another preamble appears on the RXD pin.

  The idle line wakeup method requires that messages be separated by at least one preamble and that no message contains preambles.

  The preamble that wakes a receiver does not set the receiver idle bit, STAT[RIDLE], or the receive data register full flag, STAT[RDRF].

- Address mark wakeup (CTRL1[WAKE] = 1): In this wakeup method, a logic one in the most significant bit (MSB) position of a frame clears CTRL1[RWU] and wakes up the SCI. The logic one in the MSB position marks a frame as an address frame that contains addressing information. All receivers evaluate the addressing information, and the receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. CTRL1[RWU] remains set and the receiver remains on standby until another address frame appears on the RXD pin.

  The logic one MSB of an address frame clears the receiver's CTRL1[RWU] bit before the stop bit is received and sets STAT[RDRF].

  The address mark wakeup method allows messages to contain preambles but requires that the MSB be reserved for use in address frames.

Note

With CTRL1[WAKE] clear, setting CTRL1[RWU] after the RXD pin is idle can cause the receiver to wake up immediately.

11.4.4.9 Single-Wire Operation

Normally, the SCI uses two pins for transmitting and receiving. In single-wire operation, the RXD pin is disconnected from the SCI and is available as a general-purpose I/O pin. The SCI uses the TXD pin for both receiving and transmitting.

Setting CTRL1[TE] configures TXD as the output for transmitted data. Clearing CTRL1[TE] configures TXD as the input for received data.

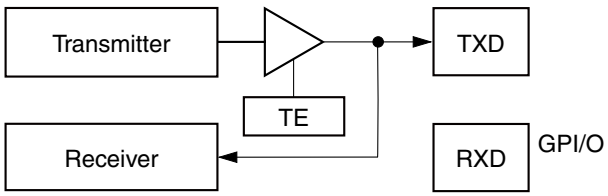


Figure 11-29. Single-Wire Operation (CTRL1[LOOP] = 1, CTRL1[RSRC] = 1)

Enable single-wire operation by setting CTRL1[LOOP] and the receiver source bit, CTRL1[RSRC]. Setting CTRL1[LOOP] disables the path from the RXD pin to the receiver. Setting CTRL1[RSRC] connects the receiver input to the output of the TXD pin driver.

11.4.4.10 Loop Operation

In loop operation the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI and is available as a general-purpose I/O pin.

Setting CTRL1[TE] connects the transmitter output to the TXD pin. Clearing CTRL1[TE] disconnects the transmitter output from the TXD pin.

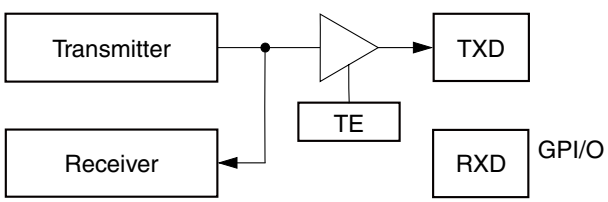


Figure 11-30. Loop Operation (CTRL1[LOOP] = 1, CTRL1[RSRC] = 0)

Enable loop operation by setting CTRL1[LOOP] and clearing CTRL1[RSRC]. Setting CTRL1[LOOP] disables the path from the RXD pin to the receiver. Clearing CTRL1[RSRC] connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (CTRL1[TE] = 1 and CTRL1[RE] = 1).

11.4.5 LIN Slave Operation

LIN slave operation occurs when CTRL2[LIN MODE] is set. The receiver searches for a break character (a start bit, 8 bits of data all 0, and a 0 in the stop bit location). After a break character is detected (11 consecutive samples of logic zero), the next field to be received is the sync field. The sync field is a word with 0x55 data that produces an alternating 0 and 1 pattern. The receiver detects the falling edge at the beginning of the

start bit and starts counting system clocks until the falling edge at the beginning of data bit 7 is detected, at which point it stops counting. This count is divided by 8 (for the 8-bit periods that have passed) and further divided by 16 to provide new RATE[SBR] and RATE[FRAC_SBR] values. If the data value of the sync field is 0x55, then these new RATE[SBR] and RATE[FRAC_SBR] values are placed in the baud rate register. Then the slave is considered synced to the master and further data words are received properly. If the data value of the sync field isn't 0x55, then the LIN sync error (STAT[LSE]) bit is set and subsequent received data bytes should be ignored.

To detect the break character successfully, the initial baud rate for this slave device must be within 15 percent of the nominal baud rate for the LIN master device.

11.4.6 Low-Power Options

11.4.6.1 Run Mode

Clearing the transmitter enable or receiver enable bits (CTRL1[TE] or CTRL1[RE]) reduces power consumption in run mode. SCI registers are still accessible when CTRL1[TE] or CTRL1[RE] is cleared, but clocks to the core of the SCI are disabled.

11.4.6.2 Wait Mode

SCI operation in wait mode depends on the state of CTRL1[SWAI].

- If CTRL1[SWAI] is clear, the SCI operates normally when the DSC core is in wait mode.
- If CTRL1[SWAI] is set, SCI clock generation ceases and the SCI module enters a power-conservation state when the DSC core is in wait mode. SCI registers are not accessible. Setting CTRL1[SWAI] does not affect the state of the receiver enable bit, CTRL1[RE], or the transmitter enable bit, CTRL1[TE].

  If CTRL1[SWAI] is set, any transmission or reception in progress stops at wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the DSC out of wait mode. Exiting wait mode via reset aborts any transmission or reception in progress and resets the SCI.

11.4.6.3 Stop Mode

SCI operation in stop mode depends on the state of the SCI stop disable bit in the SIM's applicable stop disable register.

- If the SCI stop disable bit is clear, the SCI is inactive in stop mode to reduce power consumption. The STOP instruction does not affect the SCI registers' states. SCI operation resumes after an interrupt brings the CPU out of stop mode. Exiting stop mode by reset aborts any transmission or reception in progress and resets the SCI.
- If the SCI stop disable bit is set, the SCI operates normally in stop mode.

11.5 Resets

Any system reset completely resets the SCI.

11.6 Clocks

All timing is derived from the IP Bus clock, which is the main clock for this module. See Baud-Rate Generation about how the data rate is determined.

11.7 Interrupts

Table 11-27. Interrupt Summary

Interrupt	Source	Description
TDRE	Transmitter	Transmit Data Register Empty interrupt
TIDLE	Transmitter	Transmit Idle interrupt
RERR	Receiver	Receive Error (FE, NF, PF, or OR) interrupt
RDRF/OR	Receiver	Receive Data Register Full / Overrun interrupt

11.7.1 Description of Interrupt Operation

Table 11-28. SCI Interrupt Sources

Interrupt Source	Flag	Local Enable	Description
Transmitter	STAT[TDRE]	CTRL1[TEIE]	Transmit Data Register Empty interrupt

Table continues on the next page...

Table 11-28. SCI Interrupt Sources (continued)

Interrupt Source	Flag	Local Enable	Description
Transmitter	STAT[TIDLE]	CTRL1[TIIE]	Transmit Idle interrupt
Receiver	STAT[RDRF] STAT[OR]	CTRL1[RFIE]	Receive Data Register Full / Overrun interrupt
Receiver	STAT[FE] STAT[PE] STAT[NF] STAT[OR]	CTRL1[REIE]	Receive Error (FE, NF, PF, or OR) interrupt

11.7.1.1 Transmitter Empty Interrupt

The transmitter empty interrupt is enabled by setting CTRL1[TEIE]. When this interrupt is enabled, an interrupt is generated while data is transferred from the SCI data register to the transmit shift register. The interrupt service routine should read the STAT register, verify that STAT[TDRE] is set, and write the next data to be transmitted to the DATA register, which clears STAT[TDRE].

11.7.1.2 Transmitter Idle Interrupt

The transmitter idle interrupt is enabled by setting CTRL1[TIIE]. This interrupt indicates that STAT[TIDLE] is set and the transmitter is no longer sending data, preamble, or break characters. The interrupt service routine should read the STAT register, verify STAT[TIDLE] is set, and initiate a preamble, break, or write a data character to the DATA register. Any of these actions clears STAT[TIDLE] because the transmitter is then busy.

11.7.1.3 Receiver Full Interrupt

The receiver full interrupt is enabled by setting CTRL1[RFIE]. This interrupt indicates that receive data is available in the DATA register. The interrupt service routine should read the STAT register, verify that STAT[RDRF] is set, and then read the data from the DATA register, which clears STAT[RDRF].

11.7.1.4 Receive Error Interrupt

The receive error interrupt is enabled by setting CTRL1[REIE]. This interrupt indicates that the receiver detected any of the errors reflected by the following conditions:

- Noise flag (STAT[NF]) set
- Parity error flag (STAT[PF]) set
- Framing error flag (STAT[FE]) set
- Overrun flag (STAT[OR]) set

The interrupt service routine should read the STAT register to determine which of the error flags was set. The error flag is cleared by writing (anything) to the STAT register. Then software should take the appropriate action to handle the error condition.

11.7.2 Recovery from Wait and Stop Mode

Any enabled SCI interrupt request can bring the DSC core out of wait mode or stop mode (if the SCI module is enabled in stop mode).

Chapter 12
Queued Serial Peripheral Interface (QSPI)

12.1 Introduction

12.1.1 Overview

The serial peripheral interface (SPI) module allows full-duplex, synchronous, serial communication between the DSC and peripheral devices, including other DSCs. Software can poll the SPI status flags or SPI operation can be interrupt driven. The block contains six 16-bit memory mapped registers for control parameters, status, and data transfer.

Features of the SPI module include the following:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Programmable Length Transactions (2 to 16 Bits)
- Programmable transmit and receive shift order (MSB or LSB first)
- Fourteen master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency ÷ 4
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts:
  - SPRF (SPI receiver full)
  - SPTE (SPI transmitter empty)
- Mode fault error flag with DSC interrupt capability

- Overflow error flag with DSC interrupt capability
- Wired OR mode functionality enabling connection to multiple SPIs

Maximum SPI data rates are limited by I/O pad performance as specified in the device's data sheet.

12.1.2 Block Diagram

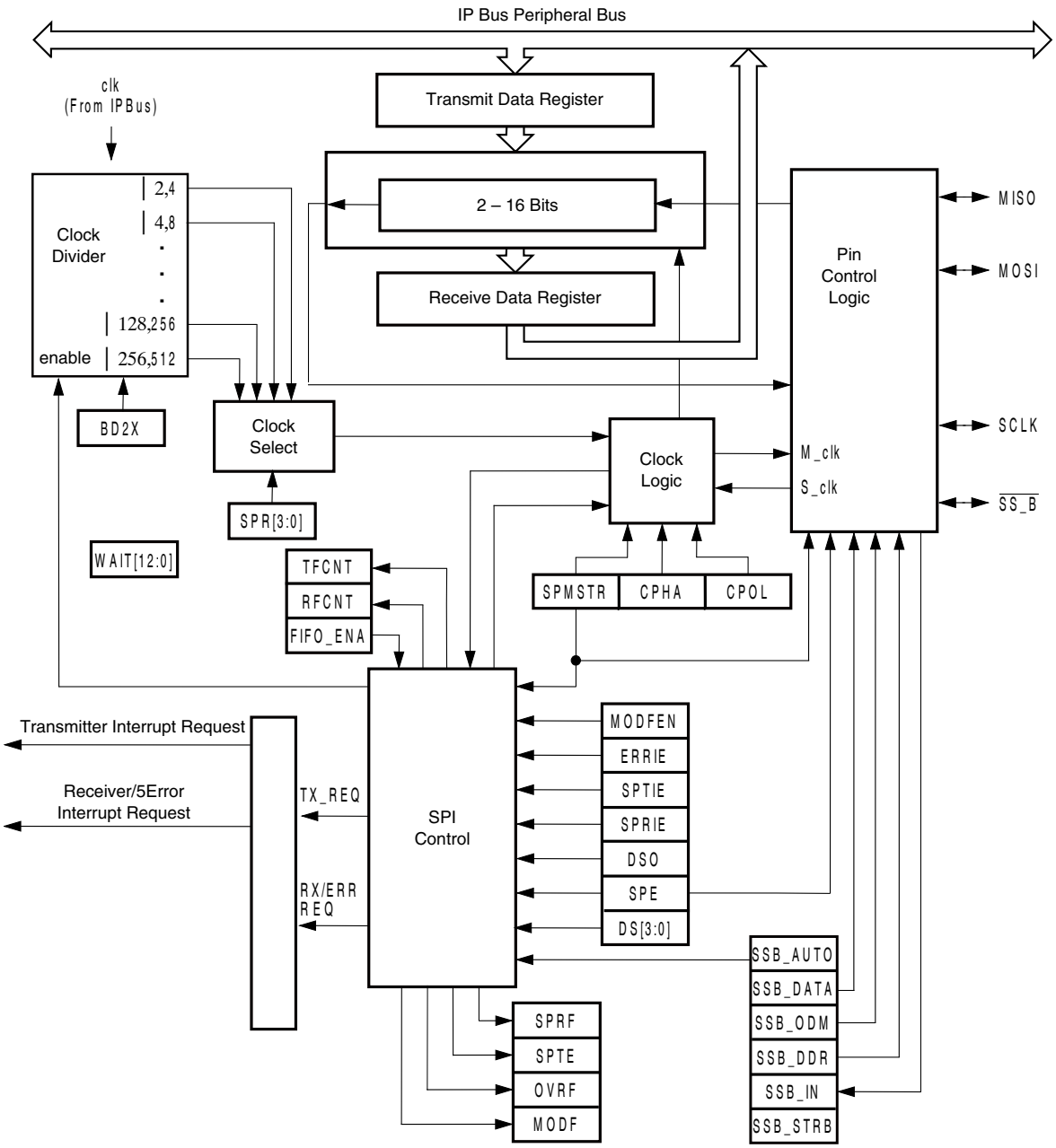


Figure 12-1. SPI Block Diagram

12.2 Signal Descriptions

12.2.1 External I/O Signals

The following are external I/O signals at the device interface. All of these pins are bidirectional.

Table 12-1. External I/O

Signal Name	Description
MOSI	Master-out Slave-in Pad Pin
MISO	Master-in Slave-out Pad Pin
SCLK	Slave Clock Pad Pin
$\overline{SS}$	Slave Select Pad Pin (Active Low)

12.2.1.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmit serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit (see the description of the SPI status and control register) is logic zero and its SS pin is at logic zero. To support a multiple-slave system, a logic one on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

12.2.1.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits serial data. In full-duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

12.2.1.3 SCLK (Serial Clock)

The serial clock synchronizes data transactions between master and slave devices. In a master DSC, the SCLK pin is the clock output. In a slave DSC, the SCLK pin is the clock input. In full duplex operation, the master and slave DSC exchange data in the same number of clock cycles as the number of bits of transmitted data.

12.2.1.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transaction. Because it is used to indicate the start of a transaction, the $\overline{SS}$ must be toggled high and low between each full length set of data transmitted for the CPHA = 0 format. However, it can remain low between transactions for the CPHA = 1 format.

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. The MODFEN bit can prevent the state of the $\overline{SS}$ from creating a MODF error.

NOTE

A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SCLK clocks, even if it is already in the middle of a transaction. A mode fault occurs if the $\overline{SS}$ pin changes state during a transaction.

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SCLK. For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SCTRL register must be set.

Table 12-2. SPI IO Configuration

SPE	SPMSTR	MODFEN	SPI CONFIGURATION	STATE OF SS_B LOGIC
0	X[1]	X	Not Enabled	$\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	$\overline{SS}$ input ignored by SPI, $\overline{SS}$ output may be activated under software or hardware control to select slave devices.
1	1	1	Master with MODF	Input-only to SPI

1. X = don't care

12.3 Memory Map Registers

Six registers control and monitor SPI operation. These registers should be accessed only with word accesses. Accesses other than word lengths result in undefined results. The SPI bit in the SIM module's SIM_PCE register must be 1 before the SPI registers can be changed.

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	QSPI0_SCTRL	R	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
		W																
1	QSPI0_ DSCTRL	R	WOM	0	0	BD2X	SSB_ IN	SSB_ DATA	SSB_ ODM	SSB_ AUTO	SSB_ DDR	SSB_ STRB	SSB_ OVER	SPR3	DS			
		W																
2	QSPI0_DRCV	R	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
		W																
3	QSPI0_DXMIT	R																
		W	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
4	QSPI0_FIFO	R	0	TFCNT			0	RFCNT			0	TFWM		0	RFWM		0	FIFO_ ENA
		W																
5	QSPI0_DELAY	R	0			WAIT												
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

12.3.1 SPI Status and Control Register (QSPI0_SCTRL)

The SCTRL register does the following:

- Select master SPI baud rate
- Determines data shift order
- Enables SPI module interrupt requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase

NOTE

Using BFCLR or BFSET instructions on the SCTRL register can cause unintended side effects on the status bits.

Address: QSPI0_SCTRL – F200h base + 0h offset = F200h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
Write																
Reset	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1

QSPI0_SCTRL field descriptions

Field	Description

15–13
SPR
	SPI Baud Rate Select
	
	In master mode, these read/write bits select one of four baud rates. SPR1 and SPR0 have no effect in slave mode. Reset sets SPR[2:0] to b011.
	Use the following formula to calculate the SPI baud rate:
	
	`Baud rate = clk/BD`
	`where: clk= Peripheral Bus Clock`
	`BD = baud rate divisor`
	
	00 Baud rate divisor 2
	01 Baud rate divisor 4
	10 Baud rate divisor 8
	11 Baud rate divisor 16

12 DSO	Data Shift Order This read/write bit determines which bit is transmitted or received first, either the MSB or LSB. Both master and slave SPI modules need to transmit and receive the same length packets. Regardless of how this bit is set, when reading the from the QSPI0_DRCV or writing to the QSPI0_DXMIT the LSB will always be at bit location 0 and the MSB will at the correct bit position. If the data length is less than 16 bits, the data will be zero padded on the upper bits. 0 MSB transmitted first (MSB -> LSB) 1 LSB transmitted first (LSB -> MSB)
11 ERRIE	Error Interrupt Enable This read/write bit enables the MODF (if MODFEN is also set) and OVRF bits to generate DSC interrupt requests. Reset clears the ERRIE bit. 0 MODF and OVRF cannot generate DSC interrupt requests 1 MODF and OVRF can generate DSC interrupt requests
10 MODFEN	Mode Fault Enable This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the MODFEN bit is low, the level of the SS_B pin does not affect the operation of an enabled SPI configured as a master. If configured as a master and MODFEN=1, a transaction in progress will stop if SS_B goes low. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation.

Table continues on the next page...

QSPI0_SCTRL field descriptions (continued)

Field	Description
9 SPRIE	SPI Receiver Interrupt Enable Bit This read/write bit enables interrupt requests generated by the SPRF bit or the receive FIFO watermark register. 0 SPRF interrupt requests disabled 1 SPRF interrupt requests enabled
8 SPMSTR	SPI Master Bit This read/write bit selects master mode operation or slave mode operation. 0 Slave mode 1 Master mode
7 CPOL	Clock Polarity Bit This read/write bit determines the logic state of the SCLK pin between transactions. To transmit data between SPI modules, the SPI modules must have identical CPOL values. 0 Rising edge of SCLK starts transaction 1 Falling edge of SCLK starts transaction
6 CPHA	Clock Phase Bit This read/write bit controls the timing relationship between the serial clock and SPI data. To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the SS_B pin of the slave SPI module must be set to logic one between data words. DO NOT use CPHA=0 while in DMA mode.
5 SPE	SPI Enable This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. When changing the SPE bit, you must change ONLY the SPE bit. Change any other bits in a separate write statement. In master mode the SPE bit can be cleared by a mode fault condition. 0 SPI module disabled 1 SPI module enabled
4 SPTIE	Transmit Interrupt Enable This read/write bit enables interrupt requests generated by the SPTE bit or the transmit FIFO watermark register. 0 SPTE interrupt requests disabled 1 SPTE interrupt requests enabled
3 SPRF	SPI Receiver Full This clearable, read-only flag is set each time data transfers from the shift register to the receive data register and there is no space available in the Rx queue to receive new data (Rx FIFO is full). SPRF generates an interrupt request if the SPRIE bit in the SPI control register is set. This bit will automatically clear after reading the QSPI0_DRCV register. 0 Receive data register or FIFO is not full. (If using the FIFO, then read the RFCNT field to determine the number of valid words available.) 1 Receive data register or FIFO is full.

Table continues on the next page...

QSPI0_SCTRL field descriptions (continued)

Field	Description
2 OVRF	Overflow 0 No overflow 1 Overflow
1 MODF	Mode Fault This clearable, read-only flag is set in a slave SPI if the SS_B pin goes high during a transaction with the MODFEN bit set. In a master SPI, the MODF flag is set if the SS_B pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing a one to the MODF bit when it is set. 0 SS_B pin at appropriate logic level 1 SS_B pin at inappropriate logic level
0 SPTE	SPI Transmitter Empty This clearable, read-only flag is set each time the transmit data register transfers data into the shift register and there is no more new data available in the Tx queue (Tx FIFO is empty). SPTE generates an interrupt request if the SPTIE bit in the SPI control register is set. SPTE is cleared by writing to the QSPI0_DXMIT register. **NOTE:** Do not write to the SPI data register unless the SPTE bit is high or data may be lost. 0 Transmit data register or FIFO is not empty. (If using the FIFO, then read the TFCNT field to determine how many words can be written safely.) 1 Transmit data register or FIFO is empty.

12.3.2 SPI Data Size and Control Register (QSPI0_DSCTRL)

This read/write register determines the data length for each transaction. The master and slave must transfer the same size data on each transaction. A new value will only take effect at the time the SPI is enabled (SPE bit in SCTRL register set from a zero to a one). In order to have a new value take effect, disable then re-enable the SPI with the new value in the register.

To utilize the SS_B control functions in master mode the appropriate bit must be set in GPIO_x_PER register to enable peripheral control of the SS_B pin.

Address: QSPI0_DSCTRL – F200h base + 1h offset = F201h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WOM	0	0	BD2X	SSB_IN	SSB_DATA	SSB_ODM	SSB_AUTO	SSB_DDR	SSB_STRB	SSB_OVER	SPR3	DS			
Write																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1

QSPI0_DSCTRL field descriptions

Field	Description
15 WOM	Wired-OR Mode The Wired-OR mode (WOM) control bit is used to select the nature of the SPI pins. When enabled (the WOM bit is set), the SPI pins are configured as open-drain drivers with the pull-ups disabled. When disabled (the WOM bit is cleared), the SPI pins are configured as push-pull drivers.
14 Reserved	This read-only bit is reserved and always has the value zero.
13 Reserved	This read-only bit is reserved and always has the value zero.
12 BD2X	Baud Divisor Times When set the Baud Rate Divisor will be multiplied by 2.
11 SSB_IN	SS_B Input This read only bit shows the current state of the SS_B pin in all modes.
10 SSB_DATA	SS_B Data This read/write bit is the value to drive on the SS_B pin. This bit is disabled when SSB_AUTO=1 or SSB_STRB=1. 0 SS_B pin is driven low if SSB_DDR=1 1 SS_B pin is driven high if SSB_DDR=1
9 SSB_ODM	SS_B Open Drain Mode This read/write bit enables open drain mode on the SS_B pin in master mode. 0 SS_B is configured for high and low drive. This mode is generally used in single master systems. 1 SS_B is configured as an open drain pin (only drives low output level). This mode is useful for multiple master systems.
8 SSB_AUTO	SS_B Automatic Mode This read/write bit enables hardware control of the SS_B pin in master mode. (The legacy design requires software to control the SS_B output pin.) The initial falling edge of SS_B will be generated and SS_B will be held low until the Tx buffer or FIFO is empty. This bit may be used alone or in combination with the SS_STRB to generate the required SS_B signal. 0 SS_B output signal is software generated by directly manipulating the various bits in this register or the GPIO registers (Compatible with legacy SPI software). 1 SS_B output signal is hardware generated to create the initial falling edge and final rising edge. The idle state of the SS_B is high. Do not use if MODFEN = 1.
7 SSB_DDR	SS_B Data Direction Register This read/write bit controls input/output mode on the SS_B pin in master mode. 0 SS_B is a configured as an input pin. Use this setting in slave mode or in master mode with MODFEN=1. 1 SS_B is configured as an output pin. Use this setting in master mode with MODFEN=0.
6 SSB_STRB	SS_B Strobe Mode

Table continues on the next page...

QSPI0_DSCTRL field descriptions (continued)

Field	Description
	This read/write bit enables hardware pulse of the SS_B pin in master mode between words. This bit may be used alone or in combination with the SS_AUTO to generate the required SS_B signal. Pulses are generated between words irrespective of the setting of CPHA. 0 No SS_B pulse between words. 1 SS_B output signal is pulsed high between words. This adds 1.5 baud clocks to the total word period. The idle state of the SS_B is low unless SSB_AUTO is high and then the idle state is high. Do not use if MODFEN = 1.
5 SSB_OVER	SS_B Override Register This read/write bit overrides the internal SS_B signal input from the I/O pad and replaces it with a level equal to the setting of the SPMSTR bit. This allows the SPI to function in slave mode, CPHA=1, without commiting a GPIO pin to be tied low. This bit should not be used in multi-slave systems or when CPHA=0. In master mode a mode fault error can not be generated so this bit should not be used in a multi-master system. 0 SS_B internal module input is selected to be connected to a GPIO pin. 1 SS_B internal module input is selected to be equal to SPMSTR.
4 SPR3	SPI Baud Rate Select **Table 12-6. SPI Master Baud Rate Selection** **SPR[2:0]** / **Baud Rate Divisor (BD)** **SPR3=0**: BD2X=0: 2; BD2X=1: 4 **SPR3=1**: BD2X=0: 512; BD2X=1: 1024 000: 2, 4, 512, 1024 001: 4, 8, 1024, 2048 010: 8, 16, 2048, 4096 011: 16, 32, 4096, 8192 100: 32, 64, 8192, 16384 101: 64, 128, 16384, 16384 110: 128, 256, 16384, 16384 111: 256, 512, 16384, 16384
3–0 DS	4'h0 Not Allowed 4'h1 2 bits data size 4'h2 3 bits data size 4'h3 4 bits data size 4'h4 5 bits data size 4'h5 6 bits data size 4'h6 7 bits data size 4'h7 8 bits data size 4'h8 9 bits data size 4'h9 10 bits data size

Table continues on the next page...

QSPI0_DSCTRL field descriptions (continued)

Field	Description
	4'hA 11 bits data size 4'hB 12 bits data size 4'hC 13 bits data size 4'hD 14 bits data size 4'hE 15 bits data size 4'hF 16 bits data size

12.3.3 SPI Data Receive Register (QSPI0_DRCV)

The SPI data receive register consists of a read-only data register. Reading data from the register will show the last data received after a complete transaction. The SPRF bit is set when new data is transferred to this register.

Address: QSPI0_DRCV – F200h base + 2h offset = F202h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPI0_DRCV field descriptions

Field	Description
15 R15	Receive Data Bit 15
14 R14	Receive Data Bit 14
13 R13	Receive Data Bit 13
12 R12	Receive Data Bit 12
11 R11	Receive Data Bit 11
10 R10	Receive Data Bit 10
9 R9	Receive Data Bit 9
8 R8	Receive Data Bit 8
7 R7	Receive Data Bit 7

Table continues on the next page...

QSPI0_DRCV field descriptions (continued)

Field	Description
6 R6	Receive Data Bit 6
5 R5	Receive Data Bit 5
4 R4	Receive Data Bit 4
3 R3	Receive Data Bit 3
2 R2	Receive Data Bit 2
1 R1	Receive Data Bit 1
0 R0	Receive Data Bit 0

12.3.4 SPI Data Transmit Register (QSPI0_DXMIT)

The SPI data transmit register consists of a write-only data register. Writing data to this register writes the data to the transmit data buffer. When the SPTE bit is set, new data should be written to this register. If new data is not written while in master mode, a new transaction will not begin until this register is written.

When selected in slave mode, the old data will be re-transmitted. When NOT selected and in slave mode, transmit data will remain unchanged. All data should be written with the LSB at bit 0. This register can only be written when the SPI is enabled, SPE = 1.

Address: QSPI0_DXMIT – F200h base + 3h offset = F203h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPI0_DXMIT field descriptions

Field	Description
15 T15	Transmit Data Bit 15
14 T14	Transmit Data Bit 14

Table continues on the next page...

QSPI0_DXMIT field descriptions (continued)

Field	Description
13 T13	Transmit Data Bit 13
12 T12	Transmit Data Bit 12
11 T11	Transmit Data Bit 11
10 T10	Transmit Data Bit 10
9 T9	Transmit Data Bit 9
8 T8	Transmit Data Bit 8
7 T7	Transmit Data Bit 7
6 T6	Transmit Data Bit 6
5 T5	Transmit Data Bit 5
4 T4	Transmit Data Bit 4
3 T3	Transmit Data Bit 3
2 T2	Transmit Data Bit 2
1 T1	Transmit Data Bit 1
0 T0	Transmit Data Bit 0

12.3.5 SPI FIFO Control Register (QSPI0_FIFO)

This register is used for FIFO control and status.

Address: QSPI0_FIFO – F200h base + 4h offset = F204h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	TFCNT			0	RFCNT			0	TFWM		0	RFWM		0	FIFO_ENA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

QSPI0_FIFO field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14–12 TFCNT	These read-only bits show how many words are used in the Tx FIFO. Writes to the QSPI0_DXMIT will cause TFCNT to increment; as words are pulled for transmission TX_LEVEL will be decremented. Attempts to write new data to QSPI0_DXMIT will be ignored when TFCNT indicates the FIFO is full. If FIFO_ENA is set to zero the TFCNT will always indicate FIFO empty. If master mode is enabled then transmission will continue till the FIFO is empty even if SPE is set to zero. 000 Tx FIFO Empty (If enabled Transmit Empty Interrupt asserted) 001 One word used in Tx FIFO 010 Two words used in Tx FIFO 011 Three words used in Tx FIFO 100 Tx FIFO Full
11 Reserved	This read-only bit is reserved and always has the value zero.
10–8 RFCNT	Rx FIFO Level These read-only bits show how many words are used in the Rx FIFO. As words are received RFCNT will be incremented; as words are read from the QSPI0_DRCV register the value of RFCNT will be decremented. There is one word time to read QSPI0_DRCV between when the SPRF status bit is set (interrupt asserted) and when an overflow condition will be flagged. If FIFO_ENA is set to zero the RFCNT will always indicate FIFO empty. 000 Rx FIFO Empty 001 One word used in Rx FIFO 010 Two words used in Rx FIFO 011 Three words used in Rx FIFO 100 Rx FIFO Full (If enabled Receiver Full Interrupt asserted)
7 Reserved	This read-only bit is reserved and always has the value zero.
6–5 TFWM	Tx FIFO Watermark These read/write bits determine how many words must remain in the Tx FIFO before an interrupt is generated. Increasing the value of TFWM increases the allowable latency in servicing the Tx interrupt without underrunning the Tx buffer space. Larger values of TFWM may also increase the number of Tx interrupt service requests because the maximum number of Tx words may not be available when the service routine is activated. If TFWM is set to the minimum value then only one SPI word time in interrupt service latency is allowed before an underrun condition results and continuous transmission is stopped in master mode or the last data word is re-transmitted in slave mode. This field is ignored when FIFO_ENA = 0. To clear an interrupt generated by TFWM, new words must be written to QSPI0_DXMIT or the value of TFWM reduced. 00 Transmit interrupt active when Tx FIFO has at least one word used 01 Transmit interrupt active when Tx FIFO has at least two words used 10 Transmit interrupt active when Tx FIFO has at least three words used 11 Transmit interrupt active when Tx FIFO is full
4 Reserved	This read-only bit is reserved and always has the value zero.

Table continues on the next page...

QSPI0_FIFO field descriptions (continued)

Field	Description
3–2 RFWM	Rx FIFO Watermark These read/write bits determine how many words must be used in the Rx FIFO before an interrupt is generated. Decreasing the value RFWM increases the allowable latency in servicing the Rx interrupt without overrunning the Rx buffer space. Smaller values of RFWM may also increase the number of Rx interrupt service requests because the maximum number of Rx words may not have been used when the service routine is activated. If RFWM is set to the maximum value then only one SPI word time in interrupt service latency is allowed before an overrun condition results and receive data is lost. This field is ignored when FIFO_ENA = 0. To clear an interrupt generated by RFWM, words must be read from QSPI0_DRCV or the value of RFWM must be increased. 00 Receive interrupt active when Rx FIFO has at least one word used 01 Receive interrupt active when Rx FIFO has at least two words used 10 Receive interrupt active when Rx FIFO has at least three words used 11 Receive interrupt active when Rx FIFO is full
1 Reserved	This read-only bit is reserved and always has the value zero.
0 FIFO_ENA	FIFO Enable This read/write bit enables Tx and Rx FIFO's mode. 0 FIFO's are disabled and reset. Operation is compatible with legacy SPI. 1 FIFO's are enabled. FIFO's retain their status even if SPE is set to zero.

12.3.6 SPI Word Delay Register (QSPI0_DELAY)

This register is used to control the delay between words.

Address: QSPI0_DELAY – F200h base + 5h offset = F205h

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

QSPI0_DELAY field descriptions

Field	Description
15–13 Reserved	This read-only bitfield is reserved and always has the value zero.
12–0 WAIT	Wait Delay Wait is the 13-bit register that controls the time between data transactions in master mode. The value sets the number of Peripheral Bus Clocks to delay between words as: (SPWAIT + 1).

12.4 Functional Description

12.4.1 Operating Modes

12.4.1.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR, is set.

Note

Configure the SPI module as master or slave before enabling the SPI. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI.

Only a master SPI module can initiate transactions. With the SPI enabled, software begins the transaction from the master SPI module by writing to the transmit data register. If the shift register is empty, the data immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The data begins shifting out on the MOSI pin under the control of the SPI serial clock, SCLK.

The SPR3, SPR2, SPR1, and SPR0 bits in the SPI registers control the baud rate generator and determine the speed of the shift register. Through the SCLK pin, the baud rate generator of the master also controls the shift register of the slave peripheral

As the data shifts out on the MOSI pin of the master, external data shifts in from the slave on the master's MISO pin. The transaction ends when the receiver full bit, SPRF, becomes set. At the same time that SPRF becomes set, the data from the slave transfers to the data receive register, DRCV. In normal operation, SPRF signals the end of a transaction. Software clears SPRF by reading the DRCV register. Writing to the SPI data transmit register, DXMIT, clears the SPTE bit.

The following figure is an example configuration for a full-duplex master-slave configuration. Having the SS bit of the master DSC held high is only necessary if MODFEN = 1. Tying the slave DSC SS bit to ground should only be done if CPHA = 1.

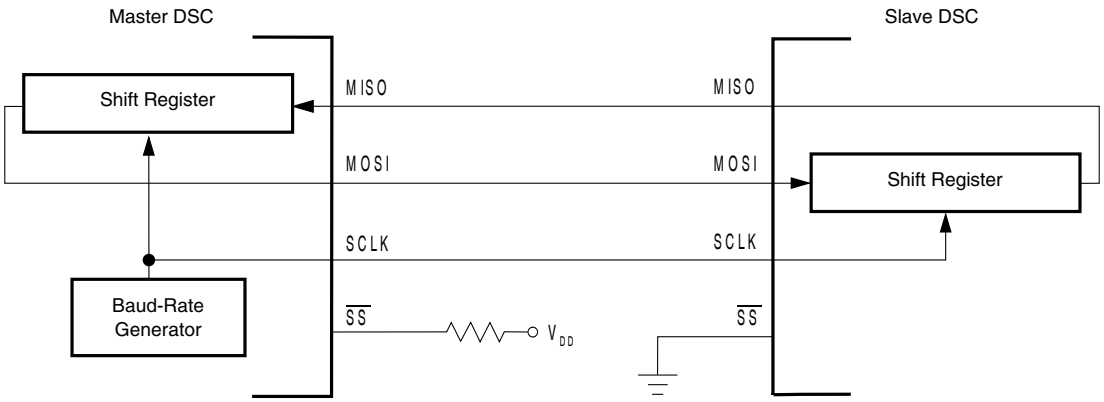


Figure 12-8. Full-Duplex Master-Slave Connections

12.4.1.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode the SCLK pin is the input for the serial clock from the master DSC. Before a data transaction occurs, the SS pin of the slave SPI must be at logic zero. $\overline{SS}$ must remain low until the transaction completes or a mode fault error occurs.

Note

The SPI must be enabled (SPE = 1) for slave transactions to be received.

Note

Data in the transmitter shift register is unaffected by SCLK transitions when the SPI operates as a slave but is deselected ($\overline{SS}$ = 1).

In a slave SPI module, data enters the shift register under the control of the serial clock, SCLK, from the master SPI module. After a full data word enters the shift register of a slave SPI, it transfers to the receive data register, DRCV, and the SPRF bit is set. To prevent an overflow condition, slave software then must read the receive data register before another full data word enters the shift register.

The maximum frequency of the SCLK for an SPI configured as a slave is less than 1/2 the bus clock frequency. The frequency of the SCLK for an SPI configured as a slave does not have to correspond to any SPI baud rate as defined by the SPR bits. The SPR bits control only the speed of the SCLK generated by an SPI configured as a master.

When the master SPI starts a transaction, the data in the slave shift register begins shifting out on the MISO pin. The slave can load its shift register with new data for the next transaction by writing to its transmit data register. The slave must write to its

transmit data register at least one bus cycle before the master starts the next transaction. Otherwise, the data that was last transmitted is reloaded into the slave shift register and shifts out on the MISO pin again. Data written to the slave shift register during a transaction remains in a buffer until the end of the transaction.

When the clock phase bit (CPHA) is set, the first edge of SCLK starts a transaction. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transaction.

Note

SCLK must be in the proper idle state before the slave is enabled to preserve the proper SCLK, MISO, MOSI timing relationships.

12.4.1.3 Wired-OR Mode

Wired-OR functionality is provided to permit the connection of multiple SPIs. Figure 11-7 illustrates the sharing of a single master device between multiple slave SPIs. When the WOM bit is set, the outputs switch from conventional complementary CMOS output to open drain outputs.

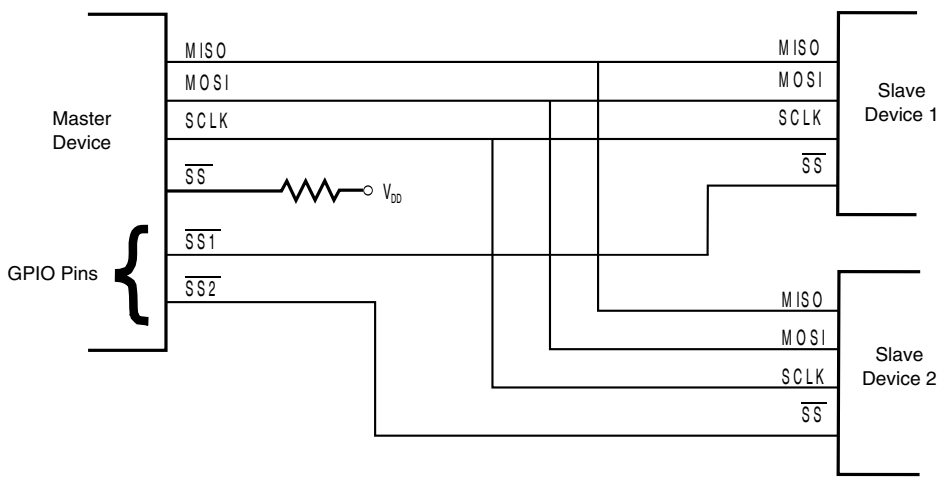


Figure 12-9. Master With Two Slaves

12.4.2 Transaction Formats

During an SPI transaction, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock synchronizes shifting and sampling on the two serial data lines. A slave select line allows selection of an individual slave SPI

device; slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

12.4.2.1 Data Transaction Length

The SPI can support data lengths of two to sixteen bits. This can be configured in the data size register, DSCTRL. When the data length is less than sixteen bits, the receive data register pads the upper bits with zeros.

Note

Data can be lost if the data length is not the same for both master and slave devices.

12.4.2.2 Data Shift Ordering

The SPI can be configured to transmit or receive the MSB of the desired data first or last. This is controlled by the data shift order, DSO, bit in the SCTRL register. Regardless of which bit is transmitted or received first, the data shall always be written to the data transmit register, DXMIT, and read from the receive data register, DRCV, with the LSB in bit 0 and the MSB in correct position depending on the data transaction size.

12.4.2.3 Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SCLK) phase and polarity using two bits in the SPI control register (SCTRL). The clock polarity is specified by the CPOL control bit, which selects an active high or low clock and has no significant effect on the transaction format.

The clock phase (CPHA) control bit selects one of two fundamentally different transaction formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transactions to allow a master device to communicate with peripheral slaves having different requirements.

Note

Before writing to the CPOL bit or the CPHA bit, disable the SPI by clearing the SPI enable bit (SPE).

12.4.2.4 Transaction Format When CPHA = 0

The following figure shows an SPI transaction in which CPHA is logic zero. The figure should not be used as a replacement for data sheet parametric information. It assumes 16 bit data lengths and the MSB shifted out first.

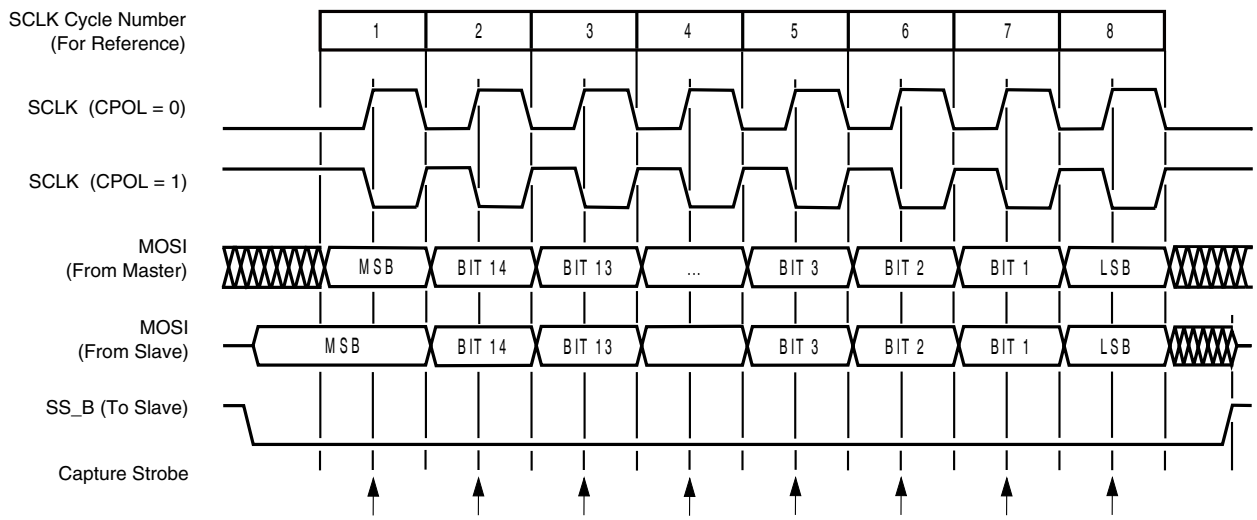


Figure 12-10. Transaction Format (CPHA = 0)

Two waveforms are shown for SCLK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. When CPHA = 0, the first SCLK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SCLK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transaction. The slave $\overline{SS}$ pin must be toggled back to high and then low again between each data word transmitted, as shown in the following figure.

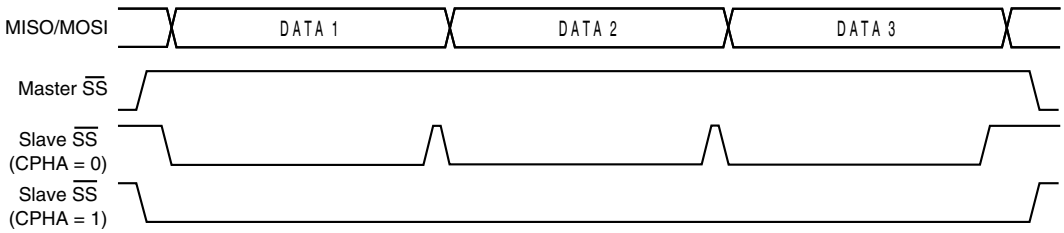


Figure 12-11. CPHA /SS Timing

When CPHA = 0 for a slave, the falling edge of SS indicates the beginning of the transaction. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. After the transaction begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of SS. Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transaction. Also, for correct operation of the slave, SPE must be active before the negative edge of SS to correctly send/receive the first word. The SS line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input (SS) is at logic zero, so that only the selected slave drives to the master.

When CPHA = 0 for a master, normal operation would begin by the master initializing the SS pin of the slave high. A transfer would then begin by the master setting the SS pin of the slave low and then writing the DXMIT register. After a data transfer completes, the master device puts the SS pin back into the high state. While MODFEN = 1, the SS pin of the master must be high or a mode fault error occurs. If MODFEN = 0, the state of the SS pin is ignored.

12.4.2.5 Transaction Format When CPHA = 1

The following figure shows an SPI transaction in which CPHA is logic one. The figure should not be used as a replacement for data sheet parametric information. It assumes 16 bit data lengths and the MSB shifted out first.

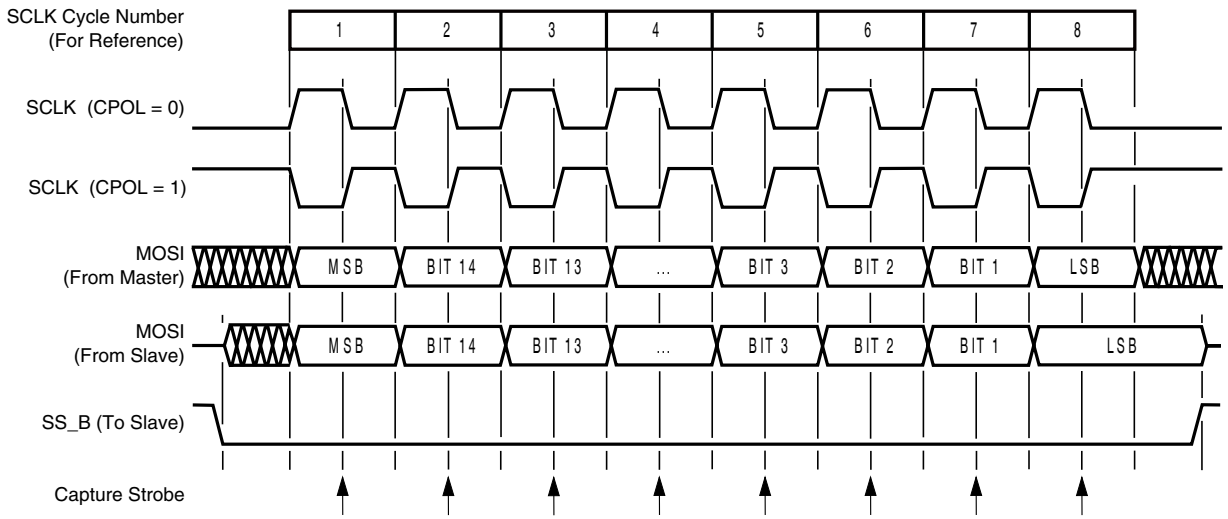


Figure 12-12. Transaction Format (CPHA = 1)

Two waveforms are shown for SCLK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master.

When CPHA = 1 for a slave, the first edge of the SCLK indicates the beginning of the transaction. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. After the transaction begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SCLK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transaction. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic zero, so that only the selected slave drives to the master.

When CPHA = 1 for a master, the MOSI pin begins being driven with new data on the first SCLK edge. If MODFEN = 0 the $\overline{SS}$ pin of the master is ignored. Otherwise, the $\overline{SS}$ pin of the master must be high or a mode fault error occurs. The $\overline{SS}$ pin can remain low between transactions. This format may be preferable in systems with only one master and one slave driving the MISO data line.

12.4.2.6 Transaction Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the DXMIT register starts a transaction. CPHA has no effect on the delay to the start of the transaction, but it does affect the initial state of the SCLK signal. When CPHA = 0, the SCLK signal remains inactive for the first half of the first SCLK cycle. When CPHA = 1, the first SCLK cycle begins with an edge on the SCLK line from its inactive to its active level. The SPI clock rate (selected by SPR2, SPR1, and SPR0) affects the delay from the write to the DXMIT register and the start of the SPI transaction. The internal baud clock in the master is a derivative of the internal DSC clock. To conserve power, it is enabled only after the SPMSTR bit is set and there is a new word written to the DXMIT register. If the DXMIT register has no new word when the current transaction completes, the internal baud clock is stopped. The initiation delay is a single SPI bit time as shown in Figure 11-11. That is, the delay is four DSC bus cycles for DIV4, eight DSC bus cycles for DIV8, sixteen DSC bus cycles for DIV16, thirty-two DSC bus cycles for DIV32, and so on.

Note

The following figure assumes 16 bit data lengths and the MSB shifted out first.

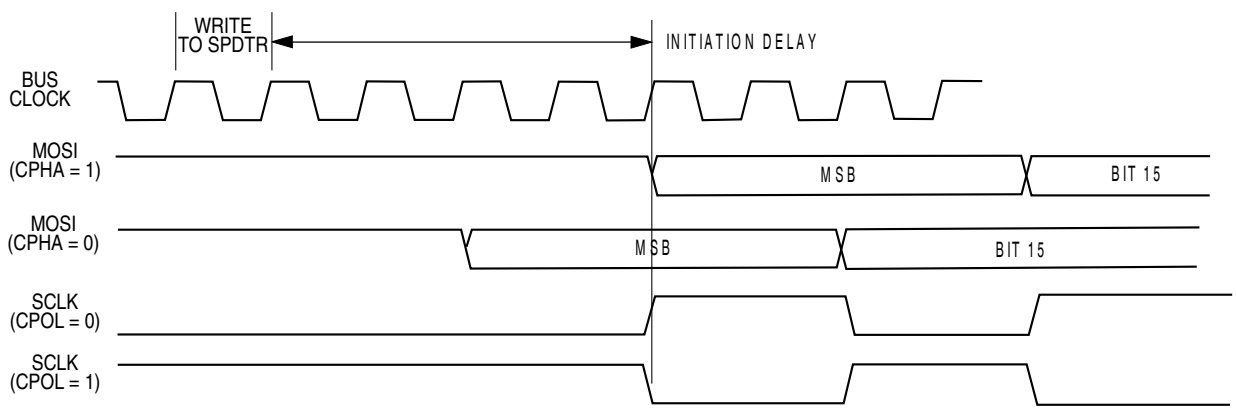


Figure 12-13. Transaction Start Delay (Master)

12.4.2.7 $\overline{SS}$ Hardware-Generated Timing in Master Mode

If the SSB_STRB bit is set in master mode, the SPI generates a word strobe pulse on $\overline{SS}$ for a slave device (see the following figures).

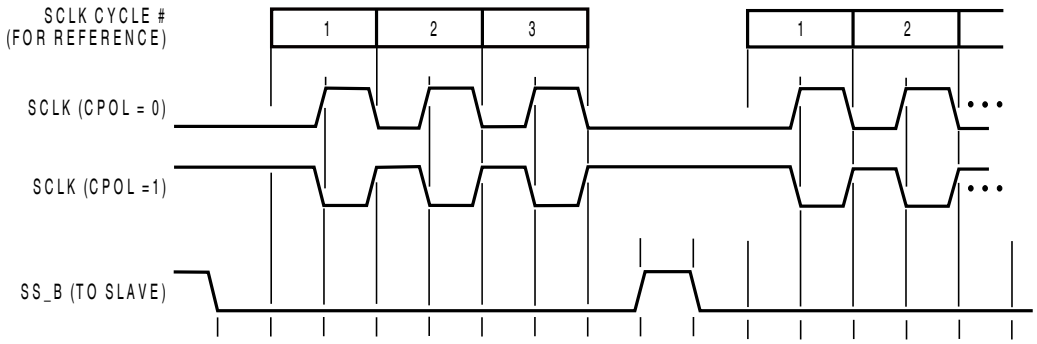


Figure 12-14. $\overline{SS}$ Strobe Timing (CPHA = 0)

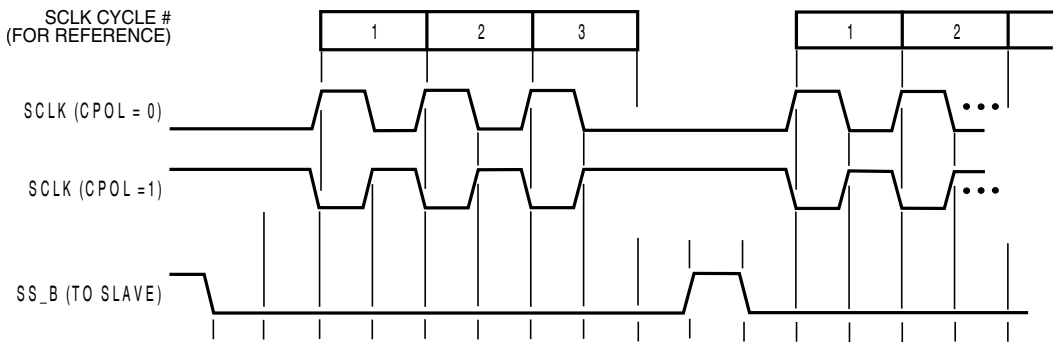


Figure 12-15. $\overline{SS}$ Strobe Timing (CPHA = 1)

If the SSB_AUTO bit is set in master mode, the SPI generates the initial falling edge and the final rising edge of $\overline{SS}$ for a slave device. The $\overline{SS}$ output has a falling edge one bit time before the first edge of SCLK (see the following figure).

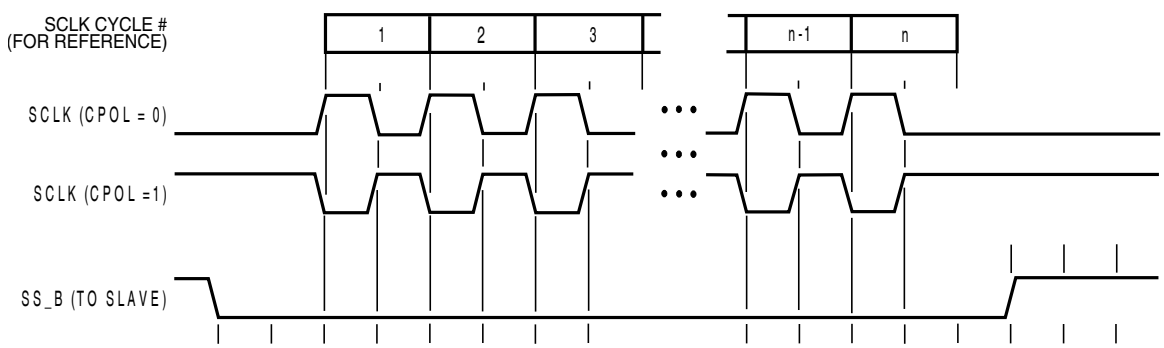


Figure 12-16. $\overline{SS}$ Auto Timing (CPHA = 1)

12.4.3 Transmission Data

The double-buffered data transmit register allows data to be queued and transmitted. For an SPI configured as a master, the queued data is transmitted immediately after the previous transaction has completed. The SPI transmitter empty flag (SPTE) indicates

when the transmit data buffer is ready to accept new data. Write to the data transmit register only when the SPTE bit is high. Figure 11-15 shows the timing associated with doing back-to-back transactions with the SPI (SCLK has CPHA = 1; CPOL = 0).

Note

The following figure assumes 16-bit data lengths and the MSB shifted out first.

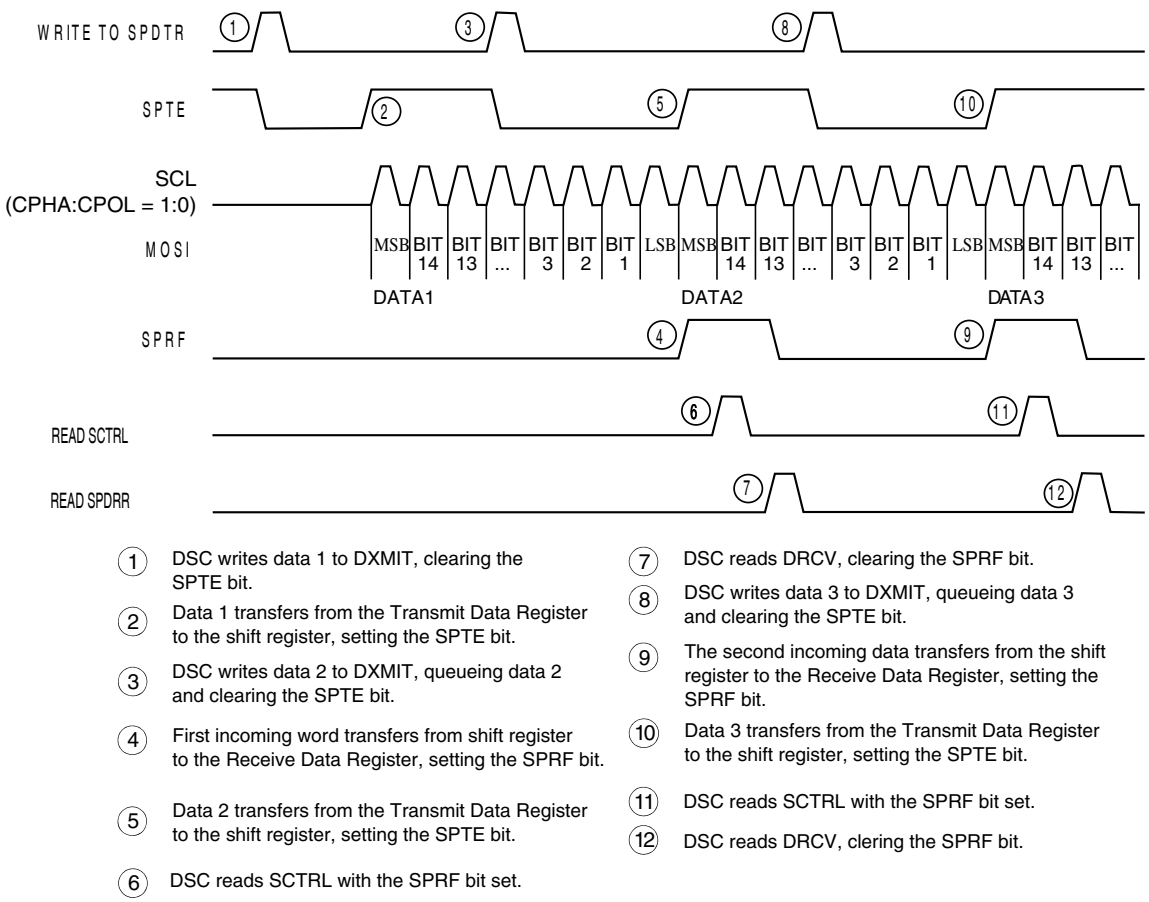


Figure 12-17. SPRF/SPTE DSC Interrupt Timing

The transmit data buffer allows back-to-back transactions without the slave precisely timing its writes between transactions, as occurs in a system with a single data buffer. Also, in slave mode, if no new data is written to the DXMIT register, the last value contained in the DXMIT register is retransmitted if the external master starts a new transaction.

For an idle master that has no data loaded into its transmit buffer and no word currently being transmitted, the SPTE is set again no more than two bus cycles after the DXMIT register is written. This allows the user to queue up at most a 32-bit value to send. For an

SPI operating in slave mode, the load of the shift register is controlled by the external master, and back-to-back writes to the transmit data register are not possible. The SPTE bit indicates when the next write can occur.

12.4.4 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the DRCV register before the next data word completes entering the shift register sets the OVRF bit. The new data word does not transfer to the receive data register, and the unread data word can still be read. OVRF is in the SPI status and control register.
- Mode fault error (MODF) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

12.4.4.1 Overflow Error

The overflow flag (OVRF) is set if the receive data register still has unread data from a previous transaction when the capture strobe of bit 1 of the next transaction occurs. The bit 1 capture strobe occurs in the middle of SCLK when the data length equals transaction data length – 1. If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the receive data register and does not set the SPI receiver full bit (SPRF). The unread data that is transferred to the receive data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI status and control register and then reading the SPI data register.

OVRF generates a receiver/error DSC interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error DSC interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the DSC SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. The following figure shows how it is possible to miss an overflow. The first part of the figure shows how it is possible to read the SCTRL register and the DRCV register to clear the SPRF without problems. However, as illustrated by the second transaction example, the OVRF bit can be set in between the time that the SCTRL register and the DRCV register are read.

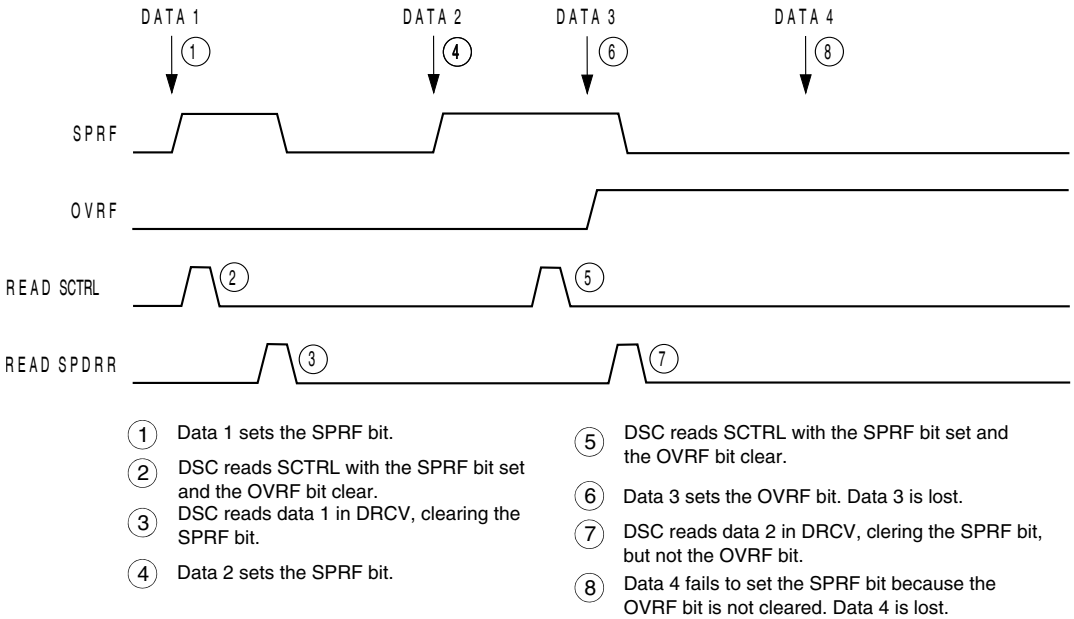


Figure 12-18. Missed Read of Overflow Condition

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that data is being lost as more transactions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SCTRL register following the read of the DRCV register. This ensures that the OVRF was not set before the SPRF was cleared and that future transactions can set the SPRF bit. The following figure illustrates this process. Generally, to avoid this second SCTRL read, enable the OVRF to the DSC by setting the ERRIE bit.

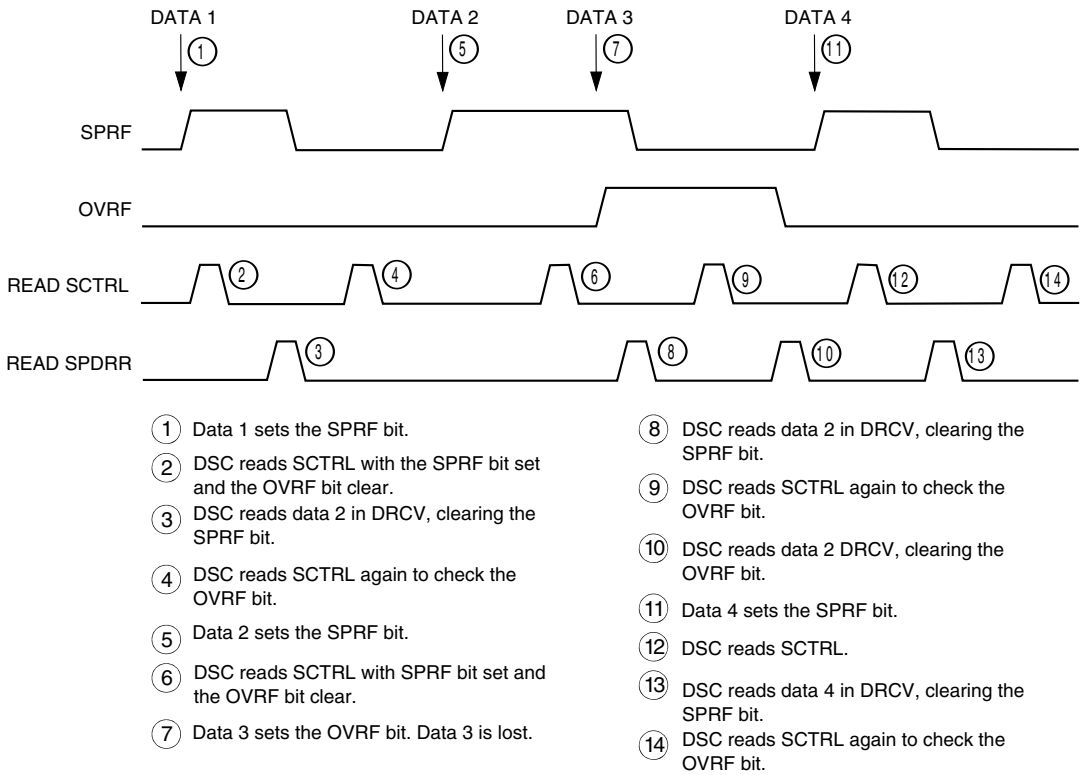


Figure 12-19. Clearing SPRF When OVRF Interrupt Is Not Enabled

12.4.4.2 Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SCLK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SCLK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, is set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR. To prevent SPI pin contention and damage to the DSC, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transaction.
- The $\overline{SS}$ pin of a master SPI goes low at any time.

For the MODF flag to be set, the mode fault error enable bit (MODFEN) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error DSC interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error DSC interrupt request. However, leaving MODFEN low prevents MODF from being set.

12.4.4.2.1 Master Mode Fault

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes to logic zero. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error DSC interrupt request.
- The SPE bit is cleared (SPI disabled).
- The SPTE bit is set.
- The SPI state counter is cleared.

Note

Setting the MODF flag does not clear the SPMSTR bit. The SPMSTR bit has no function when SPE = 0. Reading SPMSTR when MODF = 1 shows the difference between a MODF occurring when the SPI is a master and when it is a slave. When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is at logic zero) and later unselected ($\overline{SS}$ is at logic one) after the first bit of data has been received (SCLK is toggled at least once). This happens because $\overline{SS}$ at logic zero indicates the start of the transaction (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transaction occurring. Therefore, MODF does not occur since a transaction was never begun.

In a master SPI, the MODF flag is not cleared until the $\overline{SS}$ pin is at a logic one or the SPI is configured as a slave.

12.4.4.2.2 Slave Mode Fault

When configured as a slave (SPMSTR = 0), the MODF flag is set if the $\overline{SS}$ pin goes high during a transaction. When CPHA = 0, a transaction begins when $\overline{SS}$ goes low and ends after the incoming SCLK goes back to its idle level following the shift of the last data bit. When CPHA = 1, the transaction begins when the SCLK leaves its idle level and $\overline{SS}$ is already low. The transaction continues until the SCLK returns to its idle level following the shift of the last data bit.

In a slave SPI (SPMSTR = 0), the MODF bit generates an SPI receiver/error DSC interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transaction by clearing the SPE bit of the slave.

Note

A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SCLK clocks, even if it was already in the middle of a transaction. A mode fault occurs if the $\overline{SS}$ pin changes state during a transaction.

In a slave SPI, if the MODF flag is not cleared by writing a one to the MODF bit, the condition causing the mode fault still exists. In this case, the interrupt caused by the MODF flag can be cleared by disabling the ERRIE or MODFEN bits (if set) or by disabling the SPI. Disabling the SPI using the SPE bit causes a partial reset of the SPI and may cause the loss of a message currently being received or transmitted.

To clear the MODF flag, write a one to the MODF bit in the SCTRL register. The clearing mechanism must occur with no MODF condition existing or else the flag is not cleared.

12.4.5 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

1. The SPTE flag is set
2. Any slave mode transaction currently in progress is aborted
3. Any master mode transaction currently in progress is continued to completion
4. The SPI state counter is cleared, making it ready for a new complete transaction
5. All the SPI port logic is disabled.

Items 4 and 5 occur after 2 in slave mode, or after 3 in master mode.

The following items are reset only by a system reset:

- The DXMIT and DRCV registers

- All control bits in the SCTRL register (MODFEN, ERRIE, SPR2, SPR1, and SPR0)
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transactions without having to set all control bits again when SPE is set back high for the next transaction.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI is disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI is also disabled when a mode fault occurs in an SPI configured as a master.

12.5 Interrupts

Four SPI status flags can be enabled to generate DSC interrupt requests.

Table 12-11. SPI Interrupts

Flag	Interrupt Enabled By	Description
SPTE (Transmitter Empty)	SPI EnableSPI Transmitter Interrupt Enable(SPTIE = 1, SPE = 1)	The SPI transmitter interrupt enable bit (SPTIE) enables the SPI transmitter empty (SPTE) flag or TFWM to generate transmitter interrupt requests, provided that the SPI is enabled (SPE = 1). The SPTE bit becomes set every time data transfers from the transmit data register (DXMIT) to the shift register and there is no more new data available in the Tx queue. The clearing mechanism for the SPTE flag is a write to the DXMIT register.
SPRF (Receiver Full)	SPI Receiver Interrupt Enable(SPRIE = 1, SPE = 1)	The SPI receiver interrupt enable bit (SPRIE) enables the SPI receiver full (SPRF) bit or RFWM to generate receiver interrupt requests. The SPRF is set every time data transfers from the shift register to the receive data register (DRCV) and there is no more room available in the RX queue to receive new data. The clearing mechanism for the SPRF flag is to read the DRCV register.
OVRF (Overflow)	SPI Receiver/Error Interrupt Enable(ERRIE = 1)	The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error interrupt request.
MODF (Mode Fault)	SPI Receiver/Error Interrupt Enable MODF Enable(ERRIE = 1, MODFEN=1)	The mode fault enable bit (MODEFEN) enables the mode fault (MODF) bit to generate the receiver/error interrupt request regardless of the state of the SPE bit. The mode fault enable bit (MODFEN) can prevent the MODF flag from being set, so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error DSC interrupt requests.

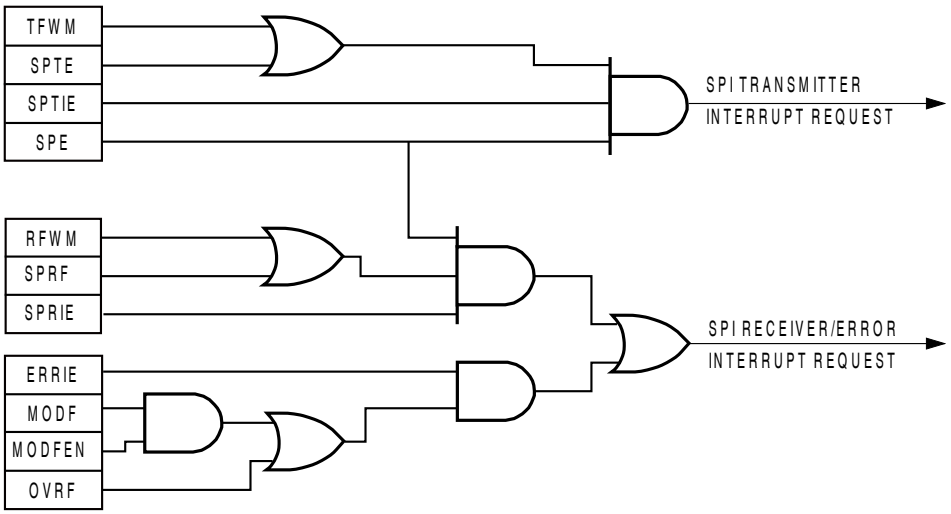


Figure 12-20. SPI Interrupt Request Generation

Chapter 13
Freescale's Scalable Controller Area Network (MSCAN)

13.1 Introduction

Freescale's scalable controller area network definition is based on the MSCAN12 definition, which is the specific implementation of the MSCAN concept targeted for the M68HC12 microcontroller family.

The module is a communication controller implementing the CAN 2.0A/B protocol as defined in the Bosch specification dated September 1991. For users to fully understand the MSCAN specification, it is recommended that the Bosch specification be read first to familiarize the reader with the terms and concepts contained within this document.

Though not exclusively intended for automotive applications, CAN protocol is designed to meet the specific requirements of a vehicle serial data bus: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN uses an advanced buffer arrangement resulting in predictable real-time behavior and simplified application software.

13.1.1 Block Diagram

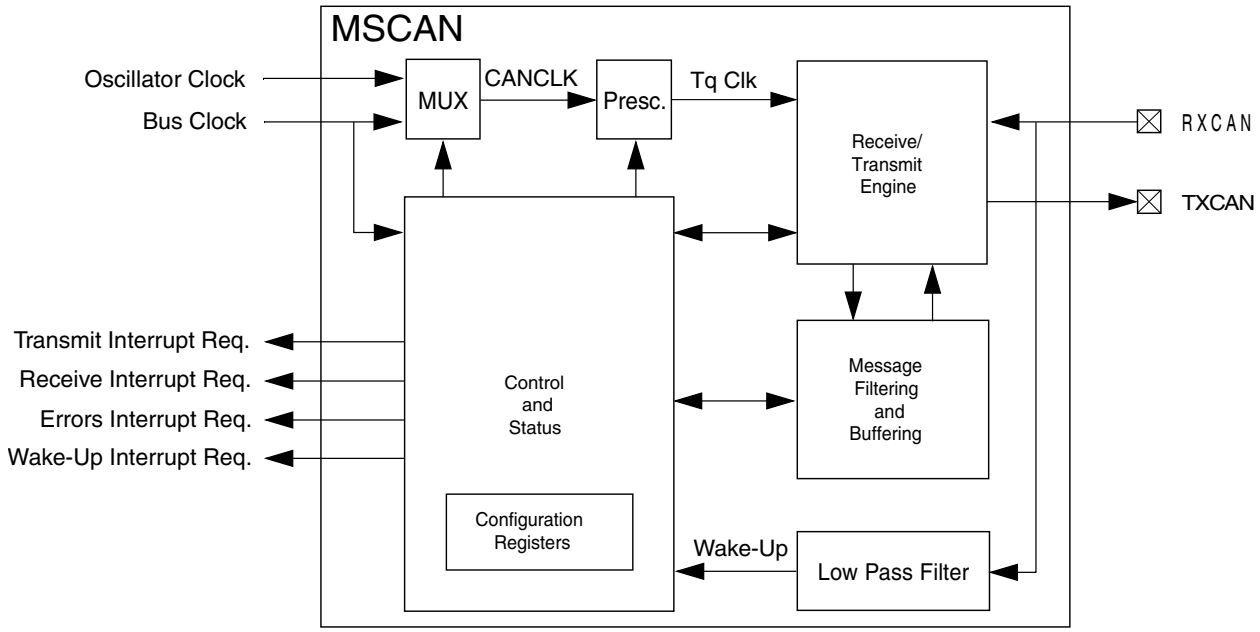


Figure 13-1. MSCAN Block Diagram

13.1.2 Features

The basic features of the MSCAN are as follows:

- Implementation of the CAN protocol — Version 2.0A/B
  - Standard and extended data frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mbps[1]
  - Support for remote frames
- Five receive buffers with FIFO storage scheme
- Three transmit buffers with internal prioritization using a "local priority" concept
- Flexible maskable identifier filter supports two full-size (32-bit) extended identifier filters, or four 16-bit filters, or eight 8-bit filters
- Programmable wakeup functionality with integrated low-pass filter

1. Depending on the actual bit timing and the clock jitter of the PLL.

- Programmable loopback mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Programmable bus-off recovery functionality
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus-off)
- Programmable MSCAN clock source either bus clock or oscillator clock
- Internal timer for time-stamping of received and transmitted messages
- Three low-power modes: sleep, power down, and MSCAN enable
- Global initialization of configuration registers

13.1.3 Modes of Operation

The following modes of operation are specific to the MSCAN. See the functional description for details.

- Listen-only mode
- MSCAN sleep mode
- MSCAN initialization mode
- MSCAN powerdown mode

13.2 External Signal Description

The MSCAN uses two external pins:

13.2.1 RXCAN — CAN Receiver Input Pin

RXCAN is the MSCAN receiver input pin.

13.2.2 TXCAN — CAN Transmitter Output Pin

TXCAN is the MSCAN transmitter output pin. The TXCAN output pin represents the logic level on the CAN bus:

0 = Dominant state

1 = Recessive state

13.2.3 CAN System

A typical CAN system with MSCAN is shown in the following figure. Each CAN station is connected physically to the CAN bus lines through a transceiver device. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against defective CAN or defective stations.

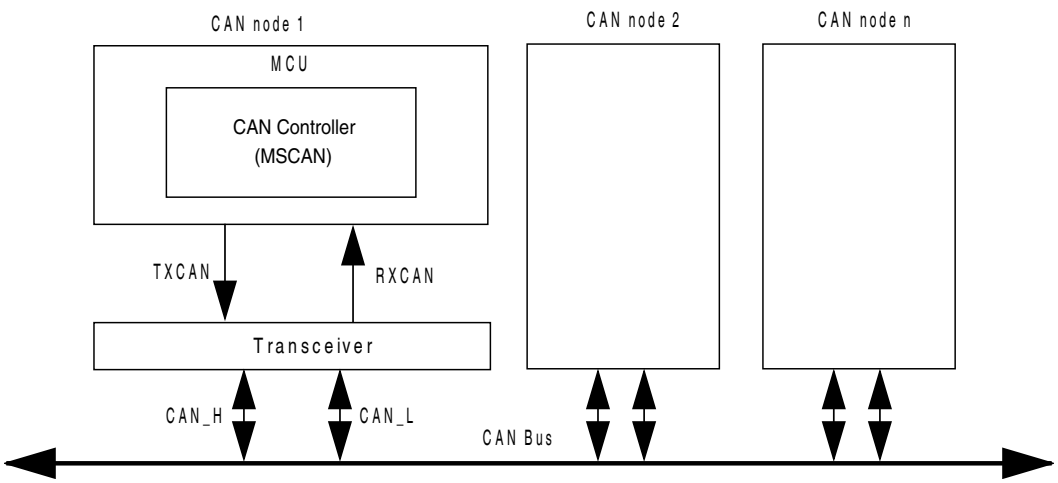


Figure 13-2. CAN System

13.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the MSCAN.

CAUTION

The registers of the CAN module must not be read by the core unless the CAN module is being supplied clocks by the system. Failure to observe this requirement will lock up the system bus.

13.3.1 Programmer's Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers.

To simplify the programmer interface, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13-byte data structure.

An additional transmit buffer priority register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map, the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set.

The time stamp register is written by the MSCAN. The CPU can only read these registers.

Table 13-1. Message Buffer Organization

Mnemonic	Register	Note
IDR0	Identifier Register 0	
IDR1	Identifier Register 1	
IDR2	Identifier Register 2	
IDR3	Identifier Register 3	
DSR0	Data Segment Register 0	
DSR1	Data Segment Register 1	
DSR2	Data Segment Register 2	
DSR3	Data Segment Register 3	
DSR4	Data Segment Register 4	
DSR5	Data Segment Register 5	
DSR6	Data Segment Register 6	
DSR7	Data Segment Register 7	
DLR	Data Length Register	
TBPR	Transmit Buffer Priority Register	Not applicable for receive buffers
TSRH	Time Stamp Register (High Byte)	Read-only for CPU
TSRL	Time Stamp Register (Low Byte)	Read-only for CPU

The next table shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in the table after that .

All bits of the receive and transmit buffers are ' x' out of reset because of RAM-based implementation[2] . All reserved or unused bits of the receive and transmit buffers always read 'x'.

Table 13-2. Receive/Transmit Message Buffer — Extended Identifier Mapping

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
IDR0	RW	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
IDR1	RW	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
IDR2	RW	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
IDR3	RW	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
DSR0	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DSR1	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DSR2	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DSR3	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DSR4	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DSR5	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DSR6	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DSR7	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DLR	RW					DLC3	DLC2	DLC1	DLC0

Blank cell = unused, always read 'x'

Read: For transmit buffers, anytime when TXEx flag is set and the corresponding transmit buffer is selected in CAN_TBSEL. For receive buffers, only when RXF flag is set.

2. Exception: The transmit priority registers are 0 out of reset.

Write: For transmit buffers, anytime when TXEx flag is set and the corresponding transmit buffer is selected in CAN_TBSEL). Unimplemented for receive buffers.

Reset: Undefined (0x00XX) because of RAM-based implementation.

Table 13-3. Receive/Transmit Message Buffer — Standard Identifier Mapping

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
IDR0	RW	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
IDR1	RW	ID2	ID1	ID0	RTR	IDE (=0)			
IDR2	RW								
IDR3	RW								

Blank cell = unused, always read 'x'

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CAN_CTL0	R	0								RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
		W																
1	CAN_CTL1	R	0								CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
		W																
2	CAN_BTR0	R	0								SJW		BRP					
		W																
3	CAN_BTR1	R	0								SAMP	TSEG2			TSEG1			
		W																
4	CAN_RFLG	R	0								WUPIF	CSCIF	RSTAT		TSTAT		OVRIF	RXF
		W																
5	CAN_RIER	R	0								WUPIE	CSCIE	RSTATE		TSTATE		OVRIE	RXFIE
		W																
6	CAN_TFLG	R	0													TXE		
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7	CAN_TIER	R	0													TXEIE		
		W																
8	CAN_TARQ	R	0													ABTRQ		
		W																
9	CAN_TAAK	R	0													ABTAK		
		W																
A	CAN_TBSEL	R	0													TX[2:0]		
		W																
B	CAN_IDAC	R	0										IDAM		0	IDHIT		
		W																
D	CAN_MISC	R	0															BOHOLD
		W																
E	CAN_RXERR	R	0								RXERR							
		W																
F	CAN_TXERR	R	0								TXERR							
		W																
10	CAN_IDAR0	R	0								AC							
		W																
11	CAN_IDAR1	R	0								AC							
		W																
12	CAN_IDAR2	R	0								AC							
		W																
13	CAN_IDAR3	R	0								AC							
		W																
14	CAN_IDMR0	R	0								AM							
		W																
15	CAN_IDMR1	R	0								AM							
		W																
16	CAN_IDMR2	R	0								AM							
		W																
17	CAN_IDMR3	R	0								AM							
		W																
18	CAN_IDAR4	R	0								AC							
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
19	CAN_IDAR5	R	0								AC							
		W																
1A	CAN_IDAR6	R	0								AC							
		W																
1B	CAN_IDAR7	R	0								AC							
		W																
1C	CAN_IDMR4	R	0								AM							
		W																
1D	CAN_IDMR5	R	0								AM							
		W																
1E	CAN_IDMR6	R	0								AM							
		W																
1F	CAN_IDMR7	R	0								AM							
		W																
20	CAN_RXFG_IDR0 (Extended)	R	0								ID[28:21]							
		W																
20	CAN_RXFG_IDR0 (Standard)	R	0								ID[10:3]							
		W																
21	CAN_RXFG_IDR1 (Extended)	R	0								ID[20:18]			SRR (=1)	IDE (=1)	ID[17:15]		
		W																
21	CAN_RXFG_IDR1 (Standard)	R	0								ID[2:0]			RTR	IDE (=0)	0		
		W																
22	CAN_RXFG_IDR2 (Extended)	R	0								ID[14:7]							
		W																
23	CAN_RXFG_IDR3 (Extended)	R	0								ID[6:0]							RTR
		W																
24	CAN_RXFG_DSR0	R	0								DB							
		W																
25	CAN_RXFG_DSR1	R	0								DB							
		W																
26	CAN_RXFG_DSR2	R	0								DB							
		W																
27	CAN_RXFG_DSR3	R	0								DB							
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
28	CAN_RXFG_DSR4	R	0								DB							
		W																
29	CAN_RXFG_DSR5	R	0								DB							
		W																
2A	CAN_RXFG_DSR6	R	0								DB							
		W																
2B	CAN_RXFG_DSR7	R	0								DB							
		W																
2C	CAN_RXFG_DLR	R	0												DLC			
		W																
2E	CAN_RXFG_TSRH	R	0								TSR[15:8]							
		W																
2F	CAN_RXFG_TSRL	R	0								TSR[7:0]							
		W																
30	CAN_TXFG_IDR0 (Extended)	R	0								ID[28:21]							
		W																
30	CAN_TXFG_IDR0 (Standard)	R	0								ID[10:3]							
		W																
31	CAN_TXFG_IDR1 (Extended)	R	0								ID[20:18]			SRR (=1)	IDE (=1)	ID[17:15]		
		W																
31	CAN_TXFG_IDR1 (Standard)	R	0								ID[2:0]			RTR	IDE (=0)	0		
		W																
32	CAN_TXFG_IDR2 (Extended)	R	0								ID[14:7]							
		W																
33	CAN_TXFG_IDR3 (Extended)	R	0								ID[6:0]							RTR
		W																
34	CAN_TXFG_DSR0	R	0								DB							
		W																
35	CAN_TXFG_DSR1	R	0								DB							
		W																
36	CAN_TXFG_DSR2	R	0								DB							
		W																
37	CAN_TXFG_DSR3	R	0								DB							
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
38	CAN_TXFG_DSR4	R	0								DB							
		W																
39	CAN_TXFG_DSR5	R	0								DB							
		W																
3A	CAN_TXFG_DSR6	R	0								DB							
		W																
3B	CAN_TXFG_DSR7	R	0								DB							
		W																
3C	CAN_TXFG_DLR	R	0												DLC			
		W																
3D	CAN_TXFG_TBPR	R	0								PRIO							
		W																
3E	CAN_TXFG_TSRH	R	0								TSR[15:8]							
		W																
3F	CAN_TXFG_TSRL	R	0								TSR[7:0]							
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

13.3.2 MSCAN Control Register 0 (CAN_CTL0)

The CAN control register provides various control bits of the MSCAN module as described below.

The CAN control register, except WUPE, INITRQ, and SLPRQ, is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable again as soon as the initialization mode is exited (INITRQ=0 and INITAK=0).

Read: Anytime

Write: Anytime when out of initialization mode; exceptions are read-only RXACT and SYNCH, RXFRM (which is set by the module only), and INITRQ (which is also writable in initialization mode).

Address: CAN_CTL0 – F440h base + 0h offset = F440h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

CAN_CTL0 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 RXFRM	Received Frame Flag This bit is read and clear only. It is set when a receiver has received a valid message correctly, independently of the filter configuration. After it is set, it remains set until cleared by software or reset. Clearing is done by writing a 1. Writing a 0 is ignored. This bit is not valid in loopback mode. The MSCAN must be in normal mode for this bit to become set. 0 No valid message was received since last clearing this flag 1 A valid message was received since last clearing of this flag
6 RXACT	Receiver Active Status This read-only flag indicates the MSCAN is receiving a message. The flag is controlled by the receiver front end. This bit is not valid in loopback mode. See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states. 0 MSCAN is transmitting or idle 1 MSCAN is receiving a message (including when arbitration is lost)
5 CSWAI	CAN Stops in Wait Mode Enabling this bit allows for lower power consumption in wait mode by disabling all the clocks at the CPU bus interface to the MSCAN module. **NOTE:** To protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the CPU enters wait (CSWAI=1) or stop mode. 0 The module is not affected during wait mode 1 The module ceases to be clocked during wait mode
4 SYNCH	Synchronized Status This read-only flag indicates whether the MSCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the MSCAN. 0 MSCAN is not synchronized to the CAN bus 1 MSCAN is synchronized to the CAN bus
3 TIME	Timer Enable This bit activates an internal 16-bit wide free running timer which is clocked by the bit clock rate. If the timer is enabled, a 16-bit time stamp will be assigned to each transmitted/received message within the active TX/RX buffer. Right after the EOF of a valid message on the CAN bus, the time stamp is written to the time stamp registers in the appropriate buffer. The internal timer is reset (all bits set to 0) when disabled. This bit is held low in initialization mode. 0 Disable internal MSCAN timer 1 Enable internal MSCAN timer
2 WUPE	Wake-Up Enable This configuration bit allows the MSCAN to restart from sleep mode when traffic on CAN is detected. This bit must be configured before sleep mode entry for the selected function to take effect.

Table continues on the next page...

CAN_CTL0 field descriptions (continued)

Field	Description
	NOTE: The CPU has to make sure that the WUPE bit and the WUPIE wake-up interrupt enable bit are enabled, if the recovery mechanism from stop or wait is required. 0 Wake-up disabled — The MSCAN ignores traffic on CAN 1 Wake-up enabled — The MSCAN is able to restart
1 SLPRQ	Sleep Mode Request This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode. The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPAK=1. SLPRQ cannot be set while the WUPIF flag is set. Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself. **Note:** The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ=1 and SLPAK=1). 0 Running — The MSCAN functions normally 1 Sleep mode request — The MSCAN enters sleep mode when CAN bus idle
0 INITRQ	Initialization Mode Request Initialization Mode Request — When this bit is set by the CPU, the MSCAN skips to initialization mode. Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK=1. The following registers enter their hard reset state and restore their default values: • CAN_CTL0 (excluding WUPE, INITRQ, and SLPRQ bits) • CAN_RFLG (TSTAT1, TSTAT0, RSTAT1, and RSTAT0 are not affected by initialization mode) • CAN_RIER • CAN_TFLG • CAN_TIER • CAN_TARQ • CAN_TAAK • CAN_TBSEL The following registers can only be written by the CPU when the MSCAN is in initialization mode (INITRQ=1 and INITAK=1) • CAN_CTL1 • CAN_BTR0 • CAN_BTR1 • CAN_IDAC • CAN_IDAR0-7 • CAN_IDMR0-7 The values of the error counters are not affected by initialization mode. When this bit is cleared by the CPU, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits. Writing to other bits in CAN_CTL0, CAN_RFLG, CAN_RIER, CAN_TFLG, or CAN_TIER must be done only after initialization mode is exited, which is INITRQ=0 and INITAK=0. **NOTE:** The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ=1 and INITAK=1).

Table continues on the next page...

CAN_CTL0 field descriptions (continued)

Field	Description
	To protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ=1 and SLPAK=1) before requesting initialization mode. 0 Normal operation 1 MSCAN in initialization mode

13.3.3 MSCAN Control Register 1 (CAN_CTL1)

The CAN_CTL1 register provides various control bits and handshake status information of the MSCAN module

Read: Anytime

Write: Anytime when INITRQ=1 and INITAK=1, except CANE which is write once when the MSCAN is in initialization mode (INITRQ= 1 and INITAK=1).

Address: CAN_CTL1 – F440h base + 1h offset = F441h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

CAN_CTL1 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 CANE	CAN Enable Used to enable the CAN 0 MSCAN module is disabled 1 MSCAN module is enabled
6 CLKSRC	MSCAN Clock Source This bit defines the clock source for the MSCAN module (only for systems with a clock generation module. 0 MSCAN clock source is the oscillator clock 1 MSCAN clock source is the bus clock

Table continues on the next page...

CAN_CTL1 field descriptions (continued)

Field	Description
5 LOOPB	Loop Back Self Test Mode When this bit is set, the MSCAN performs an internal loopback which can be used for self test operation. The bit stream output of the transmitter is fed back to the receiver internally. The RXCAN input pin is ignored and the TXCAN output goes to the recessive state (logic 1). The MSCAN behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state, the MSCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loopback self test disabled 1 Loopback self test enabled
4 LISTEN	Listen Only Mode This bit configures the MSCAN as a CAN bus monitor. When LISTEN is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out. In addition, the error counters are frozen. Listen only mode supports applications which require “hot plugging” or throughput analysis. The MSCAN is unable to transmit any messages when listen only mode is active. 0 Normal operation 1 Listen only mode activated
3 BORM	Bus-Off Recovery Mode This bits configures the bus-off state recovery mode of the MSCAN. 0 Automatic bus-off recovery (see Bosch CAN 2.0A/B protocol specification) 1 Bus-off recovery upon user request
2 WUPM	Wake-Up Mode If WUPE in CAN_CTL0 is enabled, this bit defines whether the integrated low-pass filter is applied to protect the MSCAN from spurious wake-up. 0 MSCAN wakes up on any dominant level on the CAN bus 1 MSCAN wakes up only in case of a dominant pulse on the CAN bus that has a length of TWAKEUP
1 SLPAK	Sleep Mode Acknowledge This flag indicates whether the MSCAN module has entered sleep mode. It is used as a handshake flag for the SLPRQ sleep mode request. Sleep mode is active when SLPRQ=1 and SLPAK=1. Depending on the setting of WUPE, the MSCAN will clear the flag if it detects activity on the CAN bus while in sleep mode. 0 Running — The MSCAN operates normally 1 Sleep mode active — The MSCAN has entered sleep mode
0 INITAK	Initialization Mode Acknowledge This flag indicates whether the MSCAN module is in initialization mode. It is used as a handshake flag for the INITRQ initialization mode request. Initialization mode is active when INITRQ=1 and INITAK=1. The registers CAN_CTL1, CAN_BTR0, CAN_BTR1, CAN_IDAC, CAN_IDAR0–CAN_IDAR7, and CAN_IDMR0–CAN_IDMR7 can be written only by the CPU when the MSCAN is in initialization mode. 0 Running — The MSCAN operates normally 1 Initialization mode active — The MSCAN has entered initialization mode

13.3.4 MSCAN Bus Timing Register 0 (CAN_BTR0)

The CAN_BTR0 register configures various CAN bus timing parameters of the MSCAN module.

Read: Anytime

Write: Anytime in initialization mode (INITRQ=1 and INITAK=1)

Address: CAN_BTR0 – F440h base + 2h offset = F442h

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

CAN_BTR0 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–6 SJW	Synchronization Jump Width The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve resynchronization to data transitions on the CAN bus. 00 Synchronization Jump Width 1 Tq Clock Cycle 01 Synchronization Jump Width 2 Tq Clock Cycle 10 Synchronization Jump Width 3 Tq Clock Cycle 11 Synchronization Jump Width 4 Tq Clock Cycle
5–0 BRP	Baud Rate Prescaler These bits determine the time quanta (Tq) clock which is used to build up the bit timing.

Table 13-8. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0		Prescaler value (P)
0	0	0	0	0	0		1
0	0	0	0	0	1		2
0	0	0	0	1	0		3
0	0	0	0	1	1		4
:	:	:	:	:	:		:
1	1	1	1	1	1		64

13.3.5 MSCAN Bus Timing Register 1 (CAN_BTR1)

The CAN_BTR1 register configures various CAN bus timing parameters of the MSCAN module.

Address: CAN_BTR1 – F440h base + 3h offset = F443h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SAMP	TSEG2			TSEG1			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_BTR1 field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 SAMP	Sampling This bit determines the number of CAN bus samples taken per bit time. 0 One sample per bit. The resulting bit value is equal to the value of the single bit positioned at the sample point. 1 Three samples per bit. In this case, PHASE_SEG1 must be at least 2 time quanta (Tq). The resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended that only one sample is taken per bit time (SAMP=0)
6–4 TSEG2	Time Segment 2 Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. 000 1 Tq clock cycle (This setting is not valid) 001 2 Tq clock cycles 010 3 Tq clock cycles 011 4 Tq clock cycles 100 5 Tq clock cycles 101 6 Tq clock cycles 110 7 Tq clock cycles 111 8 Tq clock cycles
3–0 TSEG1	Time Segment 1 Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. 0000 1 Tq clock cycles (This setting is not valid) 0001 2 Tq clock cycles (This setting is not valid) 0010 3 Tq clock cycles (This setting is not valid) 0011 4 Tq clock cycles 0100 5 Tq clock cycles

Table continues on the next page...

CAN_BTR1 field descriptions (continued)

Field	Description
	0101 6 Tq clock cycles 0110 7 Tq clock cycles 0111 8 Tq clock cycles 1000 9 Tq clock cycles 1001 10 Tq clock cycles 1010 11 Tq clock cycles 1011 12 Tq clock cycles 1100 13 Tq clock cycles 1101 14 Tq clock cycles 1110 15 Tq clock cycles 1111 16 Tq clock cycles

13.3.6 MSCAN Receiver Flag Register (CAN_RFLG)

A flag can be cleared only by software (writing a 1 to the corresponding bit position) when the condition which caused the setting is no longer valid. Every flag has an associated interrupt enable bit in the CAN_RIER register.

NOTE

The CAN_RFLG register is held in the reset state (RSTAT[1:0] and TSTAT[1:0] bits are not affected by initialization mode) when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable again as soon as the initialization mode is exited (INITRQ=0 and INITAK=0).

Address: CAN_RFLG – F440h base + 4h offset = F444h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								WUPIF	CSCIF	RSTAT		TSTAT		OVRIF	RXF
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RFLG field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 WUPIF	Wake-up Interrupt Flag If the MSCAN detects CAN bus activity while in sleep mode, the module will set WUPIF. If not masked, a wake-up interrupt is pending while this flag is set.

Table continues on the next page...

CAN_RFLG field descriptions (continued)

Field	Description
	0 No wakeup activity observed while in sleep mode 1 MSCAN detected activity on the CAN bus and requested wakeup
6 CSCIF	CAN Status Change Interrupt Flag This flag is set when the MSCAN changes its current CAN bus status due to the actual value of the transmit error counter (TEC) and the receive error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, which is split into separate sections for TEC/REC, informs the system on the actual CAN bus status. If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees that the receiver/transmitter status bits (RSTAT/TSTAT) are only updated when no CAN status change interrupt is pending. If the TECs/RECs change their current value after the CSCIF is asserted, which would cause an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF interrupt is cleared again. 0 No change in CAN bus status occurred since last interrupt 1 MSCAN changed current CAN bus status
5–4 RSTAT	Receiver Status Bits The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the MSCAN. Redundant Information for the most critical CAN bus status which is “bus-off”. This only occurs if the Tx error counter exceeds a number of 255 errors. Bus-off affects the receiver state. As soon as the transmitter leaves its bus-off state the receiver state skips to RxOK too. Refer also to TSTAT coding in this register. 00 RxOK: 0 ≤ transmit error counter ≤ 96 01 RxWRN: 96 < transmit error counter ≤ 127 10 RxERR: 127 < transmit error counter ≤ 255 11 Bus-Off: transmit error counter > 255
3–2 TSTAT	Transmitter Status Bits The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the MSCAN. 00 TxOK: 0 ≤ transmit error counter ≤ 96 01 TxWRN: 96 < transmit error counter ≤ 127 10 TxERR: 127 < transmit error counter ≤ 255 11 Bus-Off: transmit error counter > 255
1 OVRIF	Overrun Interrupt Flag This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set. 0 No data overrun condition 1 A data overrun detected
0 RXF	Receive Buffer Full Flag RXF is set by the MSCAN when a new message is shifted in the receiver FIFO. This flag indicates whether the shifted buffer is loaded with a correctly received message (matching identifier, matching cyclic redundancy code (CRC) and no other errors detected). After the CPU has read that message from the RxFG buffer in the receiver FIFO, the RXF flag must be cleared to release the buffer. A set RXF flag

Table continues on the next page...

CAN_RFLG field descriptions (continued)

Field	Description
	prohibits the shifting of the next FIFO entry into the foreground buffer (RxFG). If not masked, a receive interrupt is pending while this flag is set. To ensure data integrity, do not read the receive buffer registers while the RXF flag is cleared. 0 No new message available within the RxFG 1 The receiver FIFO is not empty. A new message is available in the RxFG

13.3.7 MSCAN Receiver Interrupt Enable Register (CAN_RIER)

This register contains the interrupt enable bits for the interrupt flags described in the CAN_RFLG register.

The CAN_RIER register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0)

Read: Anytime

Write: Anytime when not in initialization mode

Address: CAN_RIER – F440h base + 5h offset = F445h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								WUPIE	CSCIE	RSTATE		TSTATE		OVRIE	RXFIE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RIER field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 WUPIE	Wake-up Interrupt Enable WUPIE and WUPE must both be enabled if the recovery mechanism from stop or wait is required. 0 No interrupt request is generated from this event. 1 A wake-up event causes a Wake-Up interrupt request.
6 CSCIE	CAN Status Change Interrupt Enable 0 No interrupt request is generated from this event. 1 A CAN status change event causes an error interrupt request.
5–4 RSTATE	Receiver Status Change Enable

Table continues on the next page...

CAN_RIER field descriptions (continued)

Field	Description
	These RSTAT enable bits control the sensitivity level in which receiver state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level the RSTAT flags continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending. Bus-off state is defined by the CAN standard (see Bosch CAN 2.0A/B protocol specification: for only transmitters). Because the only possible state change for the transmitter from bus-off to TxOK also forces the receiver to skip its current state to RxOK, the coding of the RSTAT[1:0] flags define an additional bus-off state for the receiver. 00 Do not generate any CSCIF interrupt caused by transmitter state changes. 01 Generate CSCIF interrupt only if the transmitter enters or leaves “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the transmitter enters or leaves “TxErr” or “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes.
3–2 TSTATE	Transmitter Status Change Enable These TSTAT enable bits control the sensitivity level in which transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTAT flags continue to indicate the actual transmitter state and are only updated if no CSCIF interrupt is pending. 00 Do not generate any CSCIF interrupt caused by transmitter state changes. 01 Generate CSCIF interrupt only if the transmitter enters or leaves “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the transmitter enters or leaves “TxErr” or “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes.
1 OVRIE	Overrun Interrupt Enable 0 No interrupt request is generated from this event. 1 An overrun event causes an error interrupt request.
0 RXFIE	Receiver Full Interrupt Enable 0 No interrupt request is generated from this event. 1 A receive buffer full (successful message reception) event causes a receiver interrupt request.

13.3.8 MSCAN Transmitter Flag Register (CAN_TFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CAN_TIER register.

The CAN_TFLG register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0).

Read: Anytime

Write: Anytime for TXEx flags when not in initialization mode; write of 1 clears flag, write of 0 is ignored

Address: CAN_TFLG – F440h base + 6h offset = F446h

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

CAN_TFLG field descriptions

Field	Description
15–3 Reserved	This read-only bitfield is reserved and always has the value zero.
2–0 TXE	Transmitter Buffer Empty TXE[2:0] This flag indicates that the associated transmit message buffer is empty, and thus not scheduled for transmission. The CPU must clear the flag after a message is set up in the transmit buffer and is due for transmission. The MSCAN sets the flag after the message is sent successfully. The flag is also set by the MSCAN when the transmission request is successfully aborted due to a pending abort request. If not masked, a transmit interrupt is pending while this flag is set. Clearing a TXEx flag also clears the corresponding ABTAKx. When a TXEx flag is set, the corresponding ABTRQx bit is cleared. When listen-mode is active, the TXEx flags cannot be cleared and no transmission is started. Read and write accesses to the transmit buffer will be blocked, if the corresponding TXEx bit is cleared (TXEx=0) and the buffer is scheduled for transmission. 0 The associated message buffer is full (loaded with a message due for transmission) 1 The associated message buffer is empty (not scheduled)

13.3.9 MSCAN Transmitter Interrupt Enable Register (CAN_TIER)

NOTE

The CAN_TIER register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Address: CAN_TIER – F440h base + 7h offset = F447h

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

CAN_TIER field descriptions

Field	Description
15–3 Reserved	This read-only bitfield is reserved and always has the value zero.
2–0 TXEIE	Transmitter Empty Interrupt Enable 0 No interrupt request is generated from this event. 1 A transmitter empty (transmit buffer available for transmission) event causes a transmitter empty interrupt request.

13.3.10 MSCAN Transmitter Message Abort Request Register (CAN_TARQ)

The CAN_TARQ register allows abort request of queued messages

NOTE

The CAN_TARQ register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0).

Read: Anytime

Write: Anytime when not in initialization mode

Address: CAN_TARQ – F440h base + 8h offset = F448h

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

CAN_TARQ field descriptions

Field	Description
15–3 Reserved	This read-only bitfield is reserved and always has the value zero.
2–0 ABTRQ	Abort Request The CPU sets the ABTRQx bit to request that a scheduled message buffer (TXEx=0) be aborted. The MSCAN grants the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE and abort acknowledge flags are set and a transmit interrupt occurs if enabled. The CPU cannot reset ABTRQx. ABTRQx is reset whenever the associated TXE flag is set.

Table continues on the next page...

CAN_TARQ field descriptions (continued)

Field	Description
	0 No abort request 1 Abort request pending

13.3.11 MSCAN Transmitter Message Abort Acknowledge Register (CAN_TAAK)

The CAN_TAAK register indicates the successful abort of a queued message, if requested by the appropriate bits in the CAN_TARQ register.

NOTE

The CAN_TAAK register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1).

Read: Anytime

Write: Unimplemented for ABTAK flags

Address: CAN_TAAK – F440h base + 9h offset = F449h

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

CAN_TAAK field descriptions

Field	Description
15–3 Reserved	This read-only bitfield is reserved and always has the value zero.
2–0 ABTAK	Abort Acknowledge This flag acknowledges that a message was aborted due to a pending abort request from the CPU. After a particular message buffer is flagged empty, this flag can be used by the application software to identify whether the message was aborted successfully or was sent anyway. The ABTAKx flag is cleared whenever the corresponding TXE flag is cleared. 0 The message was not aborted. 1 The message was aborted.

13.3.12 MSCAN Transmit Buffer Selection Register (CAN_TBSEL)

The CAN_TBSEL register allows the selection of the actual transmit message buffer, which then will be accessible in the CAN_TXFG register space.

NOTE

The CAN_TBSEL register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0).

Read: Find the lowest ordered bit set to 1, all other bits will be read as 0

Write: Anytime when not in initialization mode

Address: CAN_TBSEL – F440h base + Ah offset = F44Ah

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

CAN_TBSEL field descriptions

Field	Description
14–2 Reserved	This read-only bitfield is reserved and always has the value zero.
2–0 TX[2:0]	Transmit Buffer Select The lowest numbered bit places the respective transmit buffer in the CAN_TXFG register space (e.g., TX1=1 and TX0=1 selects transmit buffer TX0; TX1=1 and TX0=0 selects transmit buffer TX1). Read and write accesses to the selected transmit buffer will be blocked, if the corresponding TXEx bit is cleared and the buffer is scheduled for transmission. The following gives a short programming example of the usage of the CAN_TBSEL register: To get the next available transmit buffer, application software must read the CAN_TFLG register and write this value back into the CAN_TBSEL register. In this example Tx buffers TX1 and TX2 are available. The value read from CAN_TFLG is therefore 0b0000_0110. When writing this value back to CAN_TBSEL, the Tx buffer TX1 is selected in the CAN_TXFG because the lowest numbered bit set to 1 is at bit position 1. Reading back this value out of CAN_TBSEL results in 0b0000_0010, because only the lowest numbered bit position set to 1 is presented. This mechanism eases the application software the selection of the next available Tx buffer. If all transmit message buffers are deselected, no accesses are allowed to the CAN_TXFG registers. 0 The associated message buffer is deselected 1 The associated message buffer is selected, if lowest numbered bit

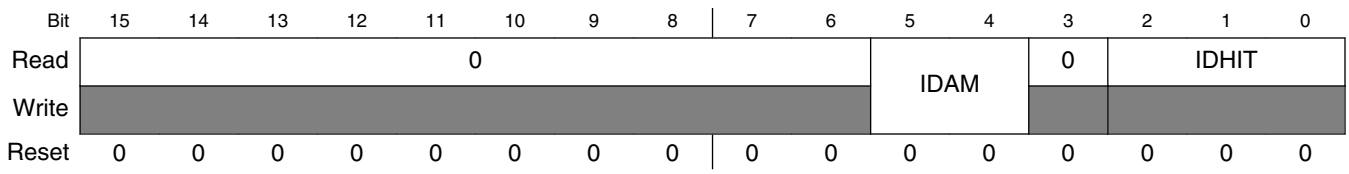
13.3.13 MSCAN Identifier Acceptance Control Register (CAN_IDAC)

The CAN_IDAC register is used for identifier acceptance control.

Read: Anytime

Write: Anytime in initialization mode (INITRQ=1 and INITAK=1), except bits IDHITx, which are read-only

Address: CAN_IDAC – F440h base + Bh offset = F44Bh



CAN_IDAC field descriptions

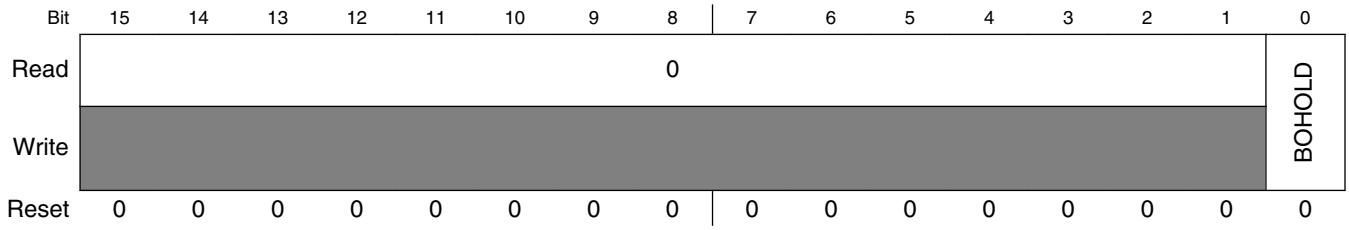
Field	Description
15–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–4 IDAM	00 Two 32-Bit Acceptance Filters 01 Four 16-Bit Acceptance Filters 10 Eight 8-Bit Acceptance Filters 11 Filter Closed
3 Reserved	This read-only bit is reserved and always has the value zero.
2–0 IDHIT	Identifier Acceptance Hit Indicator The MSCAN sets these flags to indicate an identifier acceptance hit. The IDHITx indicators are always related to the message in the foreground buffer (CAN_RXFG). When a message gets shifted into the foreground buffer of the receiver FIFO the indicators are updated as well. 000 Filter 0 Hit 001 Filter 1 Hit 010 Filter 2 Hit 011 Filter 3 Hit 100 Filter 4 Hit 101 Filter 5 Hit 110 Filter 6 Hit 111 Filter 7 Hit

13.3.14 CAN_MISC

Read: Anytime

Write: Anytime; writing 1 clears flag; writing 0 is ignored

Address: CAN_MISC – F440h base + Dh offset = F44Dh



CAN_MISC field descriptions

Field	Description
15–1 Reserved	This read-only bitfield is reserved and always has the value zero.
0 BOHOLD	Bus-off State Hold Until User Request If the BORM bit is set in control register 1, this bit indicates whether the module has entered the bus-off state. Clearing this bit requests recovery from the bus-off state. 0 Module is not in bus-off state, or recovery has been requested by user in bus-off state 1 Module is in bus-off state and holds this state until user requests recovery

13.3.15 MSCAN Receive Error Counter Register (CAN_RXERR)

This register reflects the status of the MSCAN receive error counter.

Read: Only when in sleep mode (SLPRQ=1 and SLPAK=1) or initialization mode (INITRQ=1 and INITAK=1)

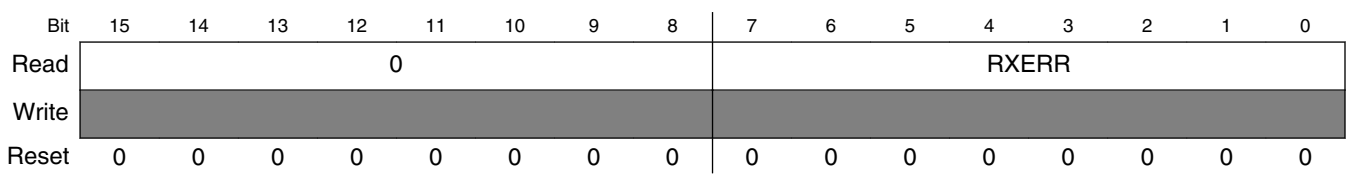
Write: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or initialization mode may return an incorrect value.

Writing to this register when in special modes can alter the MSCAN functionality.

Address: CAN_RXERR – F440h base + Eh offset = F44Eh



CAN_RXERR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 RXERR	MSCAN Receive Error Counter Bits

13.3.16 MSCAN Transmit Error Counter Register (CAN_TXERR)

This register reflects the status of the MSCAN transmit error counter.

Read: Only when in sleep mode (SLPRQ=1 and SLPAK=1) or initialization mode (INITRQ=1 and INITAK=1)

Write: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value.

Writing to this register when in special modes can alter the MSCAN functionality.

Address: CAN_TXERR – F440h base + Fh offset = F44Fh

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

CAN_TXERR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 TXERR	MSCAN Transmit Error Counter Bits

13.3.17 MSCAN Identifier Acceptance Registers (First Bank) (CAN_IDAR*n*)

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

The acceptance registers of the MSCAN are applied on the CAN_IDR0–CAN_IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CAN_IDAR0/1, CAN_IDMR0/1) are applied.

These registers can be read anytime and can be modified by writing anytime in initialization mode (INITRQ=1 and INITAK=1).

Addresses: CAN_IDAR0 – F440h base + 10h offset = F450h
CAN_IDAR1 – F440h base + 11h offset = F451h
CAN_IDAR2 – F440h base + 12h offset = F452h
CAN_IDAR3 – F440h base + 13h offset = F453h

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

CAN_IDAR*n* field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 AC	Acceptance Code Bits AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

13.3.18 MSCAN Identifier Mask Registers (First Bank) (CAN_IDMR*n*)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. To receive standard identifiers in 32 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CAN_IDMR1 and CAN_IDMR5 to “don’t care.” To receive standard identifiers in 16 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CAN_IDMR1, CAN_IDMR3, CAN_IDMR5, and CAN_IDMR7 to “don’t care.”

These registers can be read anytime and can be modified by writing anytime in initialization mode (INITRQ=1 and INITAK=1).

Addresses: CAN_IDMR0 – F440h base + 14h offset = F454h
CAN_IDMR1 – F440h base + 15h offset = F455h
CAN_IDMR2 – F440h base + 16h offset = F456h
CAN_IDMR3 – F440h base + 17h offset = F457h

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

CAN_IDMR*n* field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 AM	Acceptance Mask Bits AM[7:0] If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted. 0 Match corresponding acceptance code register and identifier bits 1 Ignore corresponding acceptance code register bit

13.3.19 MSCAN Identifier Acceptance Registers (Second Bank) (CAN_IDAR*n*)

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

The acceptance registers of the MSCAN are applied on the CAN_IDR0–CAN_IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CAN_IDAR0/1, CAN_IDMR0/1) are applied.

These registers can be read anytime and can be modified by writing anytime in initialization mode (INITRQ=1 and INITAK=1).

Addresses: CAN_IDAR4 – F440h base + 18h offset = F458h
CAN_IDAR5 – F440h base + 19h offset = F459h
CAN_IDAR6 – F440h base + 1Ah offset = F45Ah
CAN_IDAR7 – F440h base + 1Bh offset = F45Bh

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

CAN_IDAR*n* field descriptions

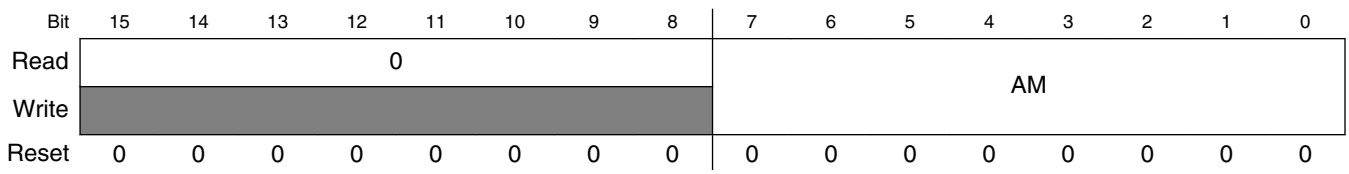
Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 AC	Acceptance Code Bits AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

13.3.20 MSCAN Identifier Mask Registers (Second Bank) (CAN_IDMR*n*)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. To receive standard identifiers in 32 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CAN_IDMR1 and CAN_IDMR5 to “don’t care.” To receive standard identifiers in 16 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CAN_IDMR1, CAN_IDMR3, CAN_IDMR5, and CAN_IDMR7 to “don’t care.”

These registers can be read anytime and can be modified by writing anytime in initialization mode (INITRQ=1 and INITAK=1).

Addresses: CAN_IDMR4 – F440h base + 1Ch offset = F45Ch
CAN_IDMR5 – F440h base + 1Dh offset = F45Dh
CAN_IDMR6 – F440h base + 1Eh offset = F45Eh
CAN_IDMR7 – F440h base + 1Fh offset = F45Fh



CAN_IDMR*n* field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 AM	Acceptance Mask Bits AM[7:0] If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted. 0 Match corresponding acceptance code register and identifier bits 1 Ignore corresponding acceptance code register bit

13.3.21 MSCAN Receive and Transmit Buffer Identifier Register 0 - Extended Identifer Mapping (CAN_*n*XFG_IDR0 (Extended))

The identifier registers for an extended format identifier consists of a total of 32 bits: the ID[28:0], SRR, IDE, and RTR bits.

Addresses: CAN_RXFG_IDR0 (Extended) – F440h base + 20h offset = F460h

CAN_TXFG_IDR0 (Extended) – F440h base + 30h offset = F470h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								ID[28:21]							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_*n*XFG_IDR0 (Extended) field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 ID[28:21]	Extended Format Identifier The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit (MSB) and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

13.3.22 MSCAN Receive and Transmit Buffer Identifier Register 0 - Standard Identifer Mapping (CAN_*n*XFG_IDR0 (Standard))

The identifier registers for a standard format identifier consists of a total of 13 bits: the ID[10:0], RTR, and IDE bits.

Addresses: CAN_RXFG_IDR0 (Standard) – F440h base + 20h offset = F460h

CAN_TXFG_IDR0 (Standard) – F440h base + 30h offset = F470h

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

CAN_*n*XFG_IDR0 (Standard) field descriptions

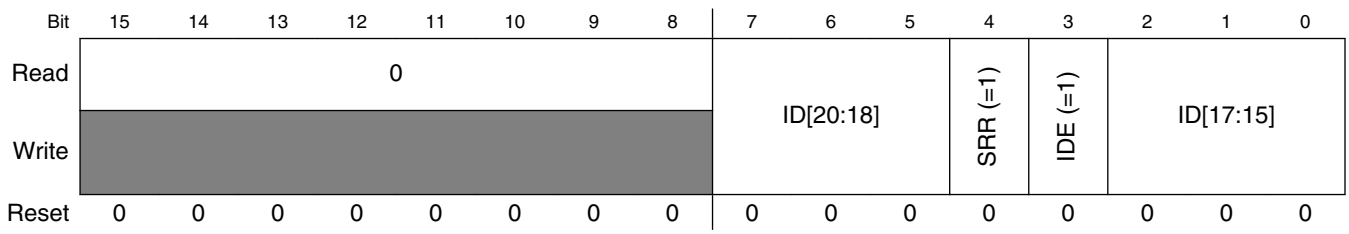
Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 ID[10:3]	Standard Format Identifier The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit (MSB) and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

13.3.23 MSCAN Receive and Transmit Buffer Identifier Register 1 - Extended Identifer Mapping (CAN_*n*XFG_IDR1 (Extended))

The identifier registers for an extended format identifier consists of a total of 32 bits: the ID[28:0], SRR, IDE, and RTR bits.

Addresses: CAN_RXFG_IDR1 (Extended) – F440h base + 21h offset = F461h

CAN_TXFG_IDR1 (Extended) – F440h base + 31h offset = F471h



CAN_*n*XFG_IDR1 (Extended) field descriptions

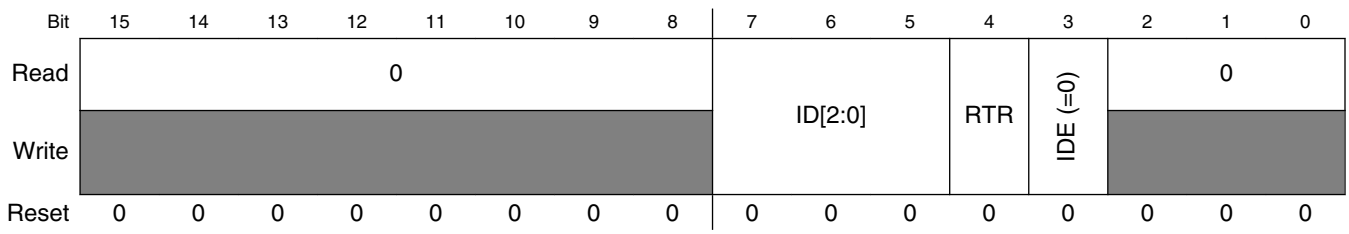
Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–5 ID[20:18]	Extended Format Identifier The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit (MSB) and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
4 SRR (=1)	Substitute Remote Request This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers.
3 IDE (=1)	ID Extended This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)
2–0 ID[17:15]	Extended Format Identifier The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit (MSB) and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

13.3.24 MSCAN Receive and Transmit Buffer Identifier Register 1 - Standard Identifier Mapping (CAN_*n*XFG_IDR1 (Standard))

The identifier registers for a standard format identifier consists of a total of 13 bits: the ID[10:0], RTR, and IDE bits.

Addresses: CAN_RXFG_IDR1 (Standard) – F440h base + 21h offset = F461h

CAN_TXFG_IDR1 (Standard) – F440h base + 31h offset = F471h



CAN_*n*XFG_IDR1 (Standard) field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–5 ID[2:0]	Standard Format Identifier The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit (MSB) and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
4 RTR	Remote Transmission Request This bit reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this bit defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame
3 IDE (=0)	ID Extended This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)
2–0 Reserved	This read-only bitfield is reserved and always has the value zero.

13.3.25 MSCAN Receive and Transmit Buffer Identifier Register 2 - Extended Identifer Mapping (CAN_*n*XFG_IDR2 (Extended))

The identifier registers for an extended format identifier consists of a total of 32 bits: the ID[28:0], SRR, IDE, and RTR bits.

Addresses: CAN_RXFG_IDR2 (Extended) – F440h base + 22h offset = F462h

CAN_TXFG_IDR2 (Extended) – F440h base + 32h offset = F472h

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

CAN_*n*XFG_IDR2 (Extended) field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 ID[14:7]	Extended Format Identifier The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit (MSB) and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

13.3.26 MSCAN Receive and Transmit Buffer Identifier Register 3 - Extended Identifier Mapping (CAN_*n*XFG_IDR3 (Extended))

The identifier registers for an extended format identifier consists of a total of 32 bits: the ID[28:0], SRR, IDE, and RTR bits.

Addresses: CAN_RXFG_IDR3 (Extended) – F440h base + 23h offset = F463h

CAN_TXFG_IDR3 (Extended) – F440h base + 33h offset = F473h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								ID[6:0]							RTR
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_*n*XFG_IDR3 (Extended) field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–1 ID[6:0]	Extended Format Identifier The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit (MSB) and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
0 RTR	Remote Transmission Request This bit reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this bit defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame

13.3.27 Receive Buffer Data Segment Registers (CAN_RXFG_DSR*n*)

The eight data segment registers, each with bits DB[7:0], contain the data to be received. The number of bytes to be received is determined by the data length code in the corresponding DLR register.

Addresses: CAN_RXFG_DSR0 – F440h base + 24h offset = F464h
CAN_RXFG_DSR1 – F440h base + 25h offset = F465h
CAN_RXFG_DSR2 – F440h base + 26h offset = F466h
CAN_RXFG_DSR3 – F440h base + 27h offset = F467h
CAN_RXFG_DSR4 – F440h base + 28h offset = F468h
CAN_RXFG_DSR5 – F440h base + 29h offset = F469h
CAN_RXFG_DSR6 – F440h base + 2Ah offset = F46Ah
CAN_RXFG_DSR7 – F440h base + 2Bh offset = F46Bh

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

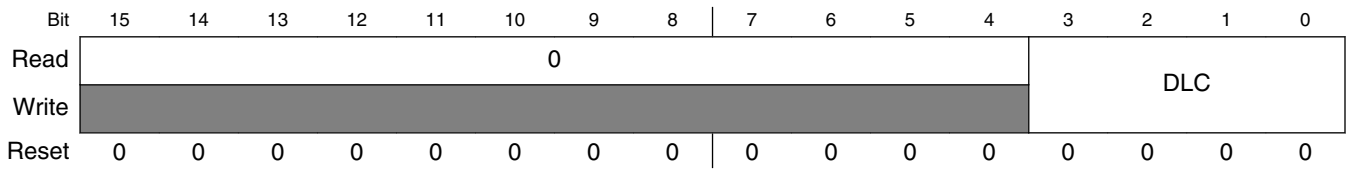
CAN_RXFG_DSR*n* field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 DB	Data bits 7-0

13.3.28 MSCAN Receive Buffer Data Length Register (CAN_RXFG_DLR)

This register keeps the data length field of the CAN frame.

Address: CAN_RXFG_DLR – F440h base + 2Ch offset = F46Ch



CAN_RXFG_DLR field descriptions

Field	Description
15–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3–0 DLC	Data Length Code The data length code (DLC) bit contains the number of bytes (data byte count) of the respective message. During the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted data bytes is always zero. The data byte count ranges from zero to eight for a data frame. 0000 Data Byte Count 0 0001 Data Byte Count 1 0010 Data Byte Count 2 0011 Data Byte Count 3 0100 Data Byte Count 4 0101 Data Byte Count 5 0110 Data Byte Count 6 0111 Data Byte Count 7 1000 Data Byte Count 8

13.3.29 Receive Buffer Time Stamp Register - High Byte (CAN_RXFG_TSRH)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus (see the section “MSCAN Control Register 0 (CANCTL0)” in this chapter). In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Read: Anytime when TXEx flag is set (see the section “MSCAN Transmitter Flag Register (CANTFLG)” in this chapter) and the corresponding transmit buffer is selected in CANTBSEL (see the section “MSCAN Transmit Buffer Selection Register (CANTBSEL)” in this chapter).

Write: Unimplemented

Address: CAN_RXFG_TSRH – F440h base + 2Eh offset = F46Eh

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

CAN_RXFG_TSRH field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 TSR[15:8]	Time Stamp Register Bits 15-8

13.3.30 Receive Buffer Time Stamp Register - Low Byte (CAN_RXFG_TSRL)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus (see the section “MSCAN Control Register 0 (CANCTL0)” in this chapter). In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Read: Anytime when TXEx flag is set (see the section “MSCAN Transmitter Flag Register (CANTFLG)” in this chapter) and the corresponding transmit buffer is selected in CANTBSEL (see the section “MSCAN Transmit Buffer Selection Register (CANTBSEL)” in this chapter).

Write: Unimplemented

Address: CAN_RXFG_TSRL – F440h base + 2Fh offset = F46Fh

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

CAN_RXFG_TSRL field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 TSR[7:0]	Time Stamp Register Bits 7-0

13.3.31 Transmit Buffer Data Segment Registers (CAN_TXFG_DSR*n*)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted. The number of bytes to be transmitted is determined by the data length code in the corresponding DLR register.

Addresses: CAN_TXFG_DSR0 – F440h base + 34h offset = F474h
CAN_TXFG_DSR1 – F440h base + 35h offset = F475h
CAN_TXFG_DSR2 – F440h base + 36h offset = F476h
CAN_TXFG_DSR3 – F440h base + 37h offset = F477h
CAN_TXFG_DSR4 – F440h base + 38h offset = F478h
CAN_TXFG_DSR5 – F440h base + 39h offset = F479h
CAN_TXFG_DSR6 – F440h base + 3Ah offset = F47Ah
CAN_TXFG_DSR7 – F440h base + 3Bh offset = F47Bh

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

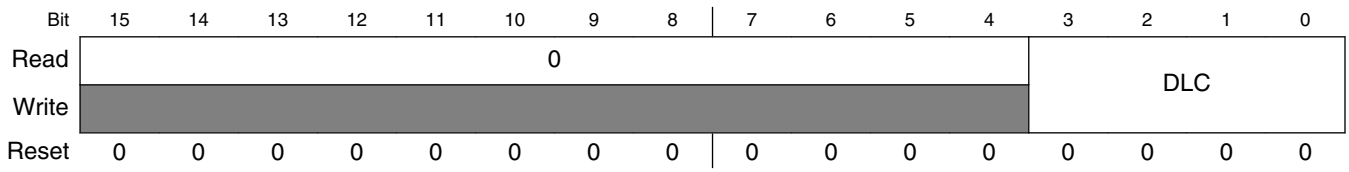
CAN_TXFG_DSR*n* field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 DB	Data bits 7-0

13.3.32 MSCAN Transmit Buffer Data Length Register (CAN_TXFG_DLR)

This register keeps the data length field of the CAN frame.

Address: CAN_TXFG_DLR – F440h base + 3Ch offset = F47Ch



CAN_TXFG_DLR field descriptions

Field	Description
15–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3–0 DLC	0000 Data Byte Count 0 0001 Data Byte Count 1 0010 Data Byte Count 2 0011 Data Byte Count 3 0100 Data Byte Count 4 0101 Data Byte Count 5 0110 Data Byte Count 6 0111 Data Byte Count 7 1000 Data Byte Count 8

13.3.33 MSCAN Transmit Buffer Priority Register (CAN_TXFG_TBPR)

This register defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the CAN and is defined to be highest for the smallest binary number. The CAN implements the following internal prioritization mechanisms:

- All transmission buffers with a cleared TXEn flag participate in the prioritization immediately before the start of frame (SOF) is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.

In cases of more than one buffer having the same lowest priority, the message buffer with the lower index number wins.

This is a read anytime register when TXEn flag is set and the corresponding transmit buffer is selected in TBSEL.

This is a write anytime register when TXEn flag is set and the corresponding transmit buffer is selected in TBSEL.

Address: CAN_TXFG_TBPR – F440h base + 3Dh offset = F47Dh

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

CAN_TXFG_TBPR field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 PRIO	Priority bits

13.3.34 Transmit Buffer Time Stamp Register - High Byte (CAN_TXFG_TSRH)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus. In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Read: Anytime when TXEx flag is set and the corresponding transmit buffer is selected in CAN_TBSEL.

Write: Unimplemented

Address: CAN_TXFG_TSRH – F440h base + 3Eh offset = F47Eh

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

CAN_TXFG_TSRH field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 TSR[15:8]	Time Stamp Register Bits 15-8

13.3.35 Transmit Buffer Time Stamp Register - Low Byte (CAN_TXFG_TSRL)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus. In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Read: Anytime when TXEx flag is set and the corresponding transmit buffer is selected in CAN_TBSEL.

Write: Unimplemented

Address: CAN_TXFG_TSRL – F440h base + 3Fh offset = F47Fh

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

CAN_TXFG_TSRL field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 TSR[7:0]	Time Stamp Register Bits 7-0

13.4 Functional Description

13.4.1 General

This section provides a complete functional description of the MSCAN. It describes each of the features and modes listed in the introduction.

13.4.2 Message Storage

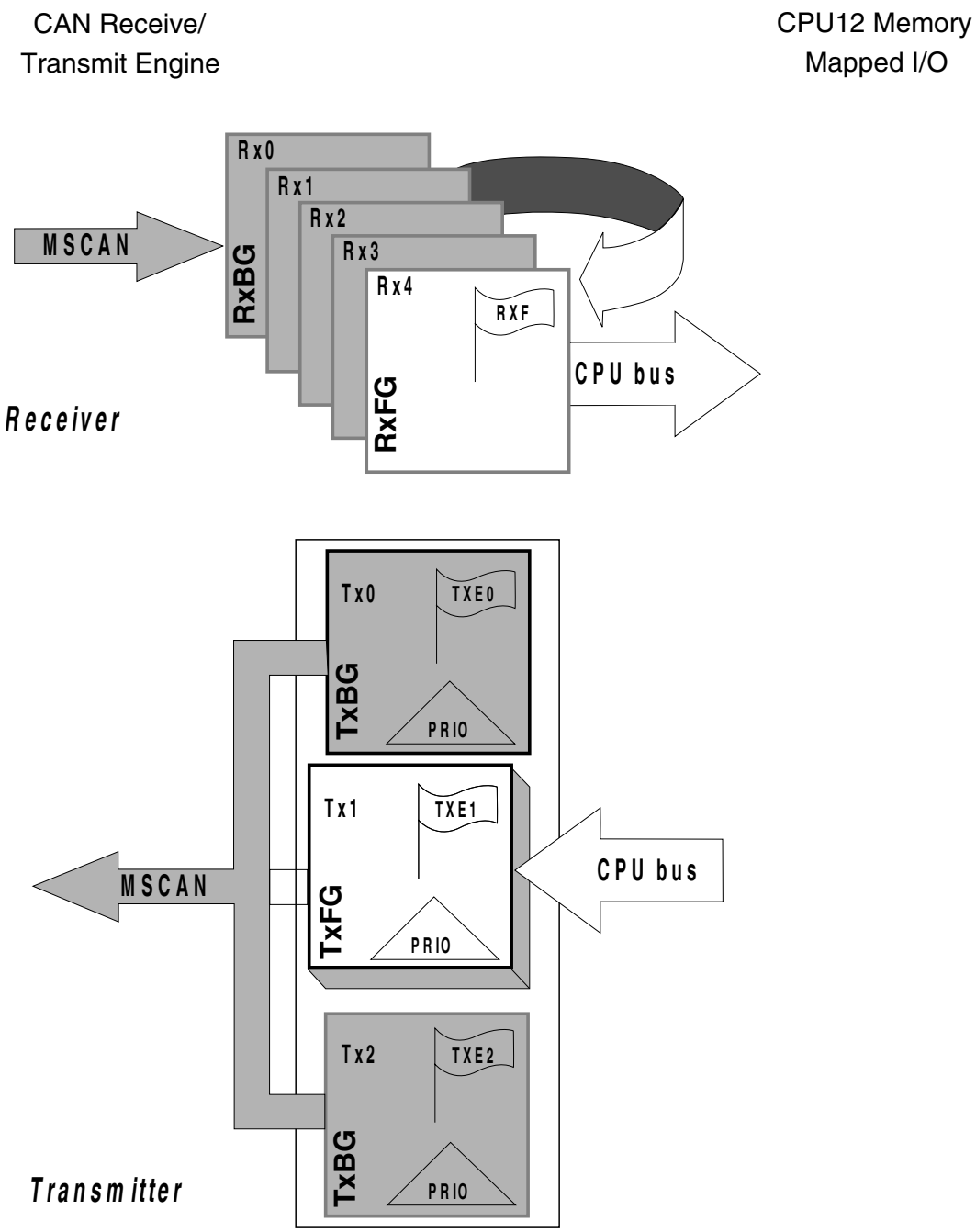


Figure 13-81. User Model for Message Buffer Organization

MSCAN facilitates a sophisticated message storage system that addresses the requirements of a broad range of network applications.

13.4.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the CAN bus between the two messages. Such nodes arbitrate for the CAN bus immediately after sending the previous message and release the CAN bus only in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The behavior described in the bullets above cannot be achieved with a single transmit buffer. That buffer must be reloaded immediately after the previous message is sent. This loading process lasts a finite amount of time and must be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, therefore, reduces the reactiveness required of the CPU. Problems can arise if the message finishes sending while the CPU re-loads the second buffer. No buffer would then be ready for transmission, and the CAN bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization, which the MSCAN implements with the "local priority" concept described in the Transmit Structures section .

13.4.2.2 Transmit Structures

The MSCAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be set up in advance.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit local priority field (PRIO). The remaining two bytes are used for time stamping of a message, if required.

To transmit a message, the CPU must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag. If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the CAN_TBSEL register. This makes the respective buffer accessible within the CAN TXFG address space. The algorithmic feature associated with the CAN_TBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXEx flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXEx flag. A transmit interrupt is generated[3] when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ). The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAKx) in the CAN_TAAK register.
2. Setting the associated TXEx flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAKx flag whether the message was aborted (ABTAKx=1) or sent (ABTAKx=0).

3. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.

13.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area. The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU. This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled.

The receiver full flag (RXF) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO[4], sets the RXF flag, and generates a receive interrupt to the CPU[, 5] . The user's receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loopback mode where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the CAN bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled. The MSCAN

4. Only if the RXF flag is not set.
5. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

remains able to transmit messages while the receiver FIFO being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

13.4.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers define the acceptable patterns of the standard or extended identifier (ID[10:0] or ID[28:0]). Any of these bits can be marked 'don't care' in the MSCAN identifier mask registers.

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and three bits in the CAN_IDAC register. These identifier hit flags (IDHIT[2:0]) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. If more than one hit occurs (two or more filters match), the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes (see Bosch CAN 2.0A/B protocol specification):

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)
    - Substitute remote request (SRR)
  - The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages[6] . This mode implements two filters for a full length CAN 2.0B compliant extended identifier. This figure shows how the first 32-bit filter bank (CAN_IDAR0–CAN_IDAR3, CAN_IDMR0–CAN_IDMR3) produces a filter 0 hit. Similarly, the second filter bank (CAN_IDAR4–CAN_IDAR7, CAN_IDMR4–CAN_IDMR7) produces a filter 1 hit.
- Four identifier acceptance filters, each to be applied to

6. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters for standard identifiers

- a) the 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages or
- b) the 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. The next figure shows how the first 32-bit filter bank (CAN_IDAR0–CAN_IDAR3, CAN_IDMR0–CAN_IDMR3) produces filter 0 and 1 hits. Similarly, the second filter bank (CAN_IDAR4–CAN_IDAR7, CAN_IDMR4–CAN_IDMR7) produces filter 2 and 3 hits.

- Eight identifier acceptance filters, each to be applied to the first eight bits of the identifier. This mode implements eight independent filters for the first eight bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier. The next figure shows how the first 32-bit filter bank (CAN_IDAR0–CAN_IDAR3, CAN_IDMR0–CAN_IDMR3) produces filter 0 to 3 hits. Similarly, the second filter bank (CAN_IDAR4–CAN_IDAR7, CAN_IDMR4–CAN_IDMR7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

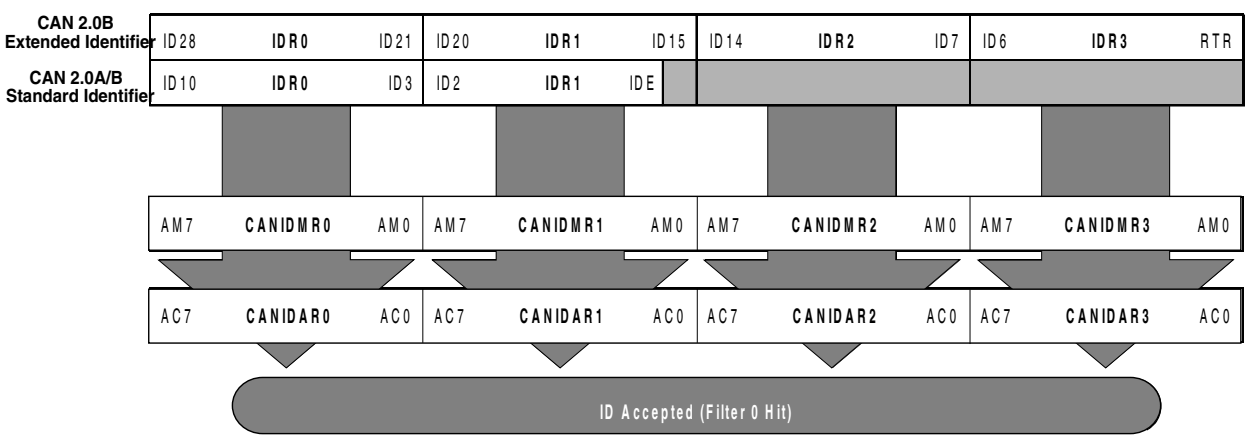


Figure 13-82. 32-bit Maskable Identifier Acceptance Filter

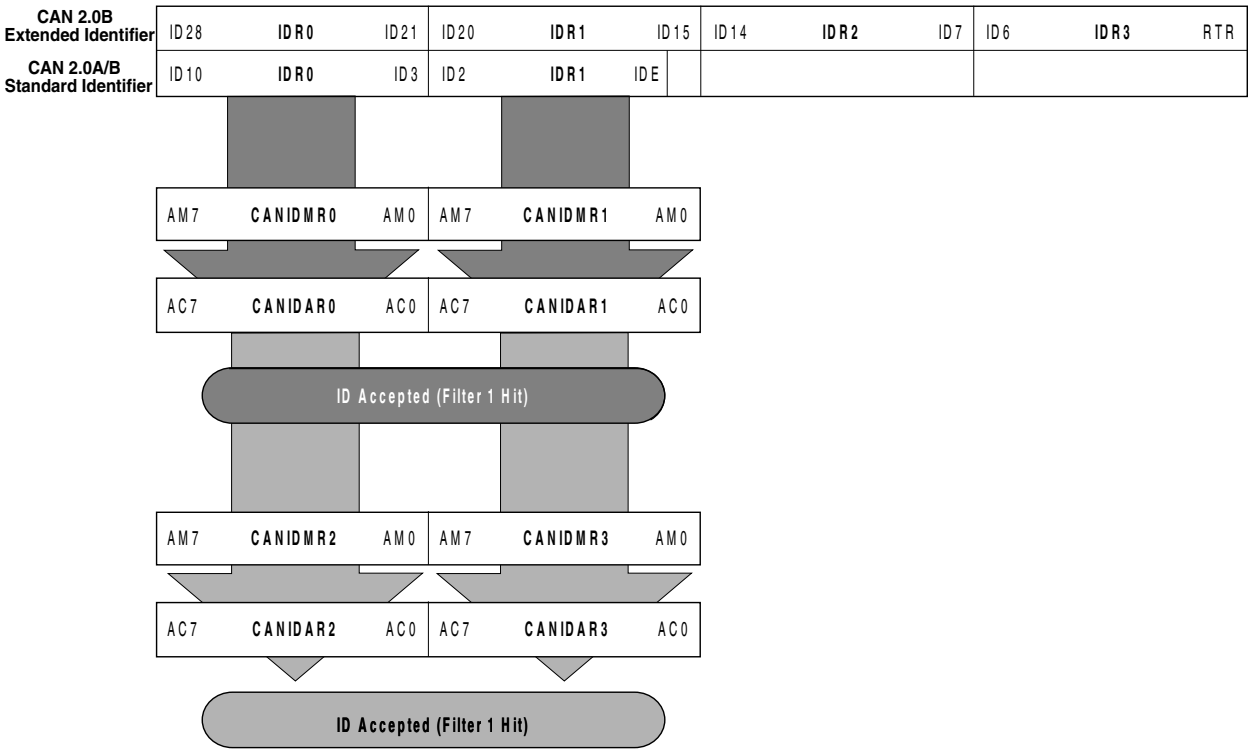


Figure 13-83. 16-bit Maskable Identifier Acceptance Filters

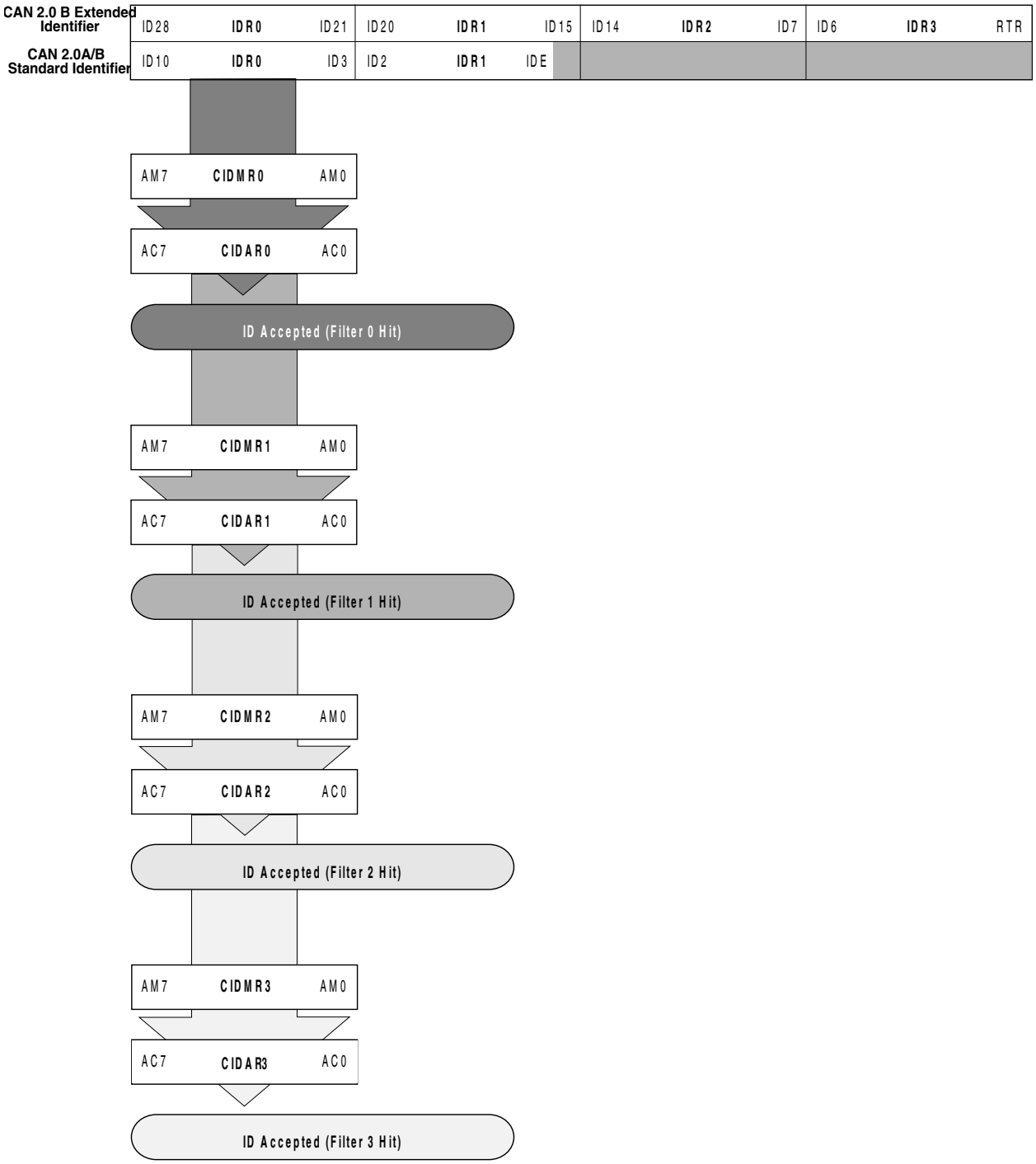


Figure 13-84. 8-bit Maskable Identifier Acceptance Filters

13.4.3.1 Protocol Violation Protection

The MSCAN protects the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers that control the configuration of the MSCAN cannot be modified while the MSCAN is on-line. The MSCAN must be in initialization mode. The corresponding INITRQ/INITAK handshake bits in the CAN_CTL0/CAN_CTL1 registers serve as a lock to protect the following registers:
  - MSCAN control 1 register (CAN_CTL1)
  - MSCAN bus timing registers 0 and 1 (CAN_BTR0, CAN_BTR1)
  - MSCAN identifier acceptance control register (CAN_IDAC)
  - MSCAN identifier acceptance registers (CAN_IDAR0–CAN_IDAR7)
  - MSCAN identifier mask registers (CANIDMR0–CANIDMR7)
- The TXCAN pin is immediately forced to a recessive state when the MSCAN goes into the power down mode or initialization mode.
- The MSCAN enable bit (CANE) is writable only once in normal system operation modes, which provides further protection against inadvertently disabling the MSCAN.

13.4.3.2 Clock System

This figure shows the structure of the MSCAN clock generation circuitry.

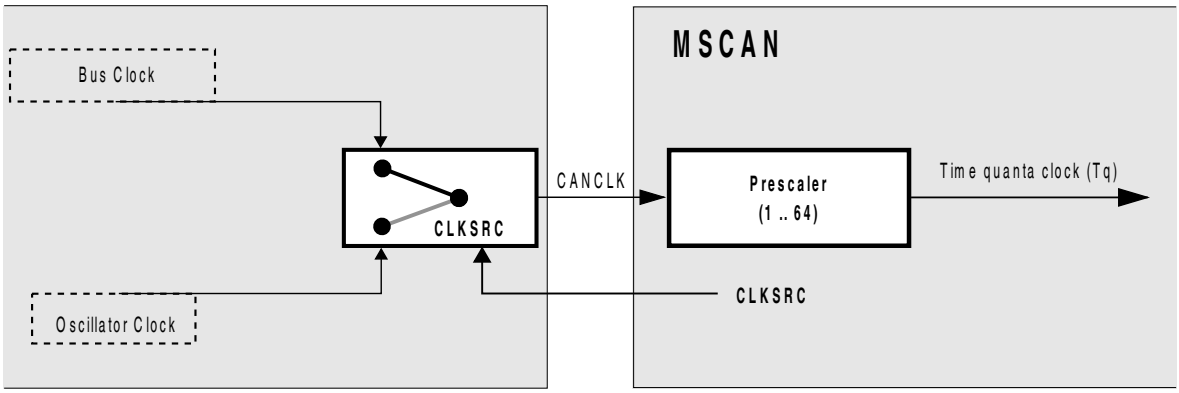


Figure 13-85. MSCAN Clocking Scheme

The clock source bit (CLKSRC) in the CAN_CTL1 register defines whether the internal CAN_CLK is connected to the output of a crystal oscillator (oscillator clock) or to the bus clock.

The clock source must be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 45% to 55% duty cycle of the clock is required.

If the bus clock is generated from a PLL, select the oscillator clock rather than the bus clock because of jitter considerations, especially at the faster CAN bus rates.

A programmable prescaler generates the time quanta (Tq) clock from CAN_CLK. A time quantum is the atomic unit of time handled by the MSCAN.

$$f_{Tq} = f_{CANCLK} / (\text{Prescaler value})$$

A bit time is subdivided into three segments as described in the Bosch CAN specification.

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time Segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

$$\text{Bit Rate} = f_{Tq} / (\text{number of Time Quanta})$$

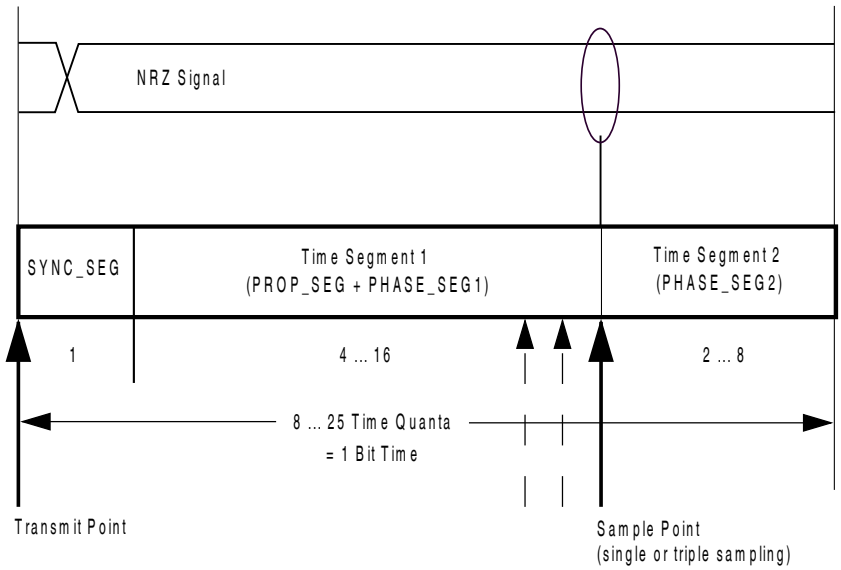


Figure 13-86. Segments within the Bit Time

Table 13-84. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the CAN bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node in receive mode samples the CAN bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The synchronization jump width (see the Bosch CAN specification for details) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The SYNC_SEG, TSEG1, TSEG2, and SJW parameters are set by programming the MSCAN bus timing registers (CAN_BTR0, CAN_BTR1).

This table gives an overview of the CAN compliant segment settings and the related parameter values.

Note

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard.

Table 13-85. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronization Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1

Table continues on the next page...

Table 13-85. CAN Standard Compliant Bit Time Segment Settings (continued)

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronization Jump Width	SJW
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

13.4.4 Modes of Operation

13.4.4.1 Normal Modes

The MSCAN module behaves as described within this specification in all normal system operation modes.

13.4.4.2 Special Modes

The MSCAN module behaves as described within this specification in all special system operation modes.

13.4.4.3 Emulation Modes

In all emulation modes, the MSCAN module behaves just like normal system operation modes as described within this specification.

13.4.4.4 Listen-Only Mode

In an optional CAN bus monitoring mode (listen-only), the CAN node is able to receive valid data frames and valid remote frames, but it sends only "recessive" bits on the CAN bus. In addition, it cannot start a transmision. If the MAC sub-layer is required to send a "dominant" bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the MAC sub-layer monitors this "dominant" bit, although the CAN bus may remain in recessive state externally.

13.4.4.5 Security Modes

The MSCAN module has no security features.

13.4.5 Low-Power Options

If the MSCAN is disabled (CANE=0), the MSCAN clocks are stopped for power saving.

If the MSCAN is enabled (CANE=1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down mode. In sleep mode, power consumption is reduced by stopping all clocks except those to access the registers from the CPU side. In power down mode, all clocks are stopped and no power is consumed.

This table summarizes the combinations of MSCAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

For all modes, an MSCAN wakeup interrupt can occur only if the MSCAN is in sleep mode (SLPRQ=1 and SLPAK=1), wakeup functionality is enabled (WUPE=1), and the wakeup interrupt is enabled (WUPIE=1).

Table 13-86. CPU vs. MSCAN Operating Modes

CPU Mode	MSCAN Mode			
	Normal	Reduced Power Consumption		
		Sleep	Power Down	Disabled (CANE=0)
Run	CSWAI=X[1] SLPRQ=0 SLPAK=0	CSWAI=X SLPRQ=1 SLPAK=1	N/A	CSWAI=X SLPRQ=X SLPAK=X
Wait	CSWAI=0 SLPRQ=0 SLPAK=0	CSWAI=0 SLPRQ=1 SLPAK=1	CSWAI=1 SLPRQ=X SLPAK=X	CSWAI=X SLPRQ=X SLPAK = X
Stop	N/A	N/A	CSWAI=X SLPRQ=X SLPAK=X	CSWAI=X SLPRQ=X SLPAK=X

1. 'X' means don't care.

13.4.5.1 Operation in Run Mode

Only MSCAN sleep mode is available as a low power option when the CPU is in run mode.

13.4.5.2 Operation in Wait Mode

The WAIT instruction puts the MCU in a low power consumption stand-by mode. If the CSWAI bit is set, additional power can be saved in power down mode because the CPU clocks are stopped. After leaving this power down mode, the MSCAN restarts its internal controllers and enters normal mode again.

While the CPU is in wait mode, the MSCAN can be operated in normal mode and generate interrupts (registers can be accessed via background debug mode). The MSCAN can also operate in any of the low-power modes depending on the values of the SLPRQ/SLPAK and CSWAI bits.

13.4.5.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption stand-by mode. In stop mode, the MSCAN is set in power down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits.

13.4.5.4 MSCAN Sleep Mode

The CPU can request the MSCAN to enter this low power mode by asserting the SLPRQ bit in the CAN_CTL0 register. The time when the MSCAN enters sleep mode depends on a fixed synchronization delay and its current activity:

- If there are one or more message buffers scheduled for transmission (TXEx=0), the MSCAN will continue to transmit until all transmit message buffers are empty (TXEx=1, transmitted successfully or aborted) and then goes into sleep mode.

- If the MSCAN is receiving, it continues to receive and goes into sleep mode as soon as the CAN bus next becomes idle.
- If the MSCAN is neither transmitting nor receiving, it immediately goes into sleep mode.

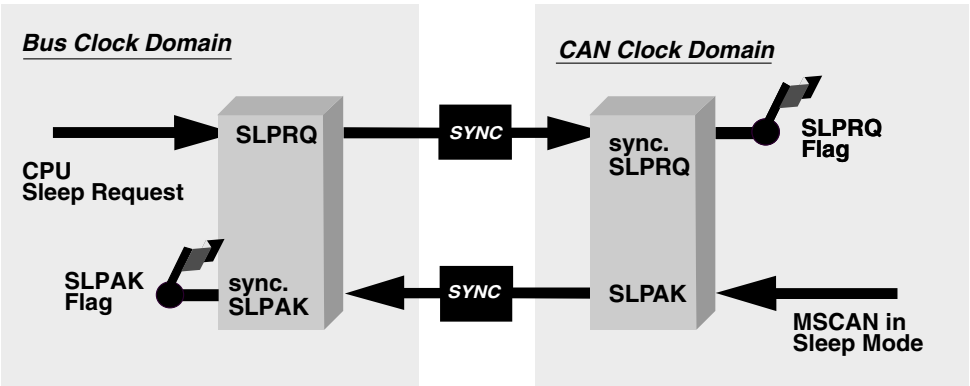


Figure 13-87. Sleep Request / Acknowledge Cycle

Note

The application software must avoid setting up a transmission (by clearing one or more TXEx flag(s)) and immediately request sleep mode (by setting SLPRQ). Whether the MSCAN starts transmitting or goes into sleep mode directly depends on the exact sequence of operations.

If sleep mode is active, the SLPRQ and SLPAK bits are set. The application software must use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode.

When in sleep mode (SLPRQ=1 and SLPAK=1), the MSCAN stops its internal clocks. However, clocks that allow register accesses from the CPU side continue to run.

If the MSCAN is in bus-off state, it stops counting the 128 occurrences of 11 consecutive recessive bits due to the stopped clocks. The TXCAN pin remains in a recessive state. If RXF=1, the message can be read and RXF can be cleared. Shifting a new message into the foreground buffer of the receiver FIFO (RxFG) does not take place while in sleep mode.

It is possible to access the transmit buffers and to clear the associated TXEx flags. No message abort takes place while in sleep mode.

If the WUPE bit in CAN_CTL0 is not asserted, the MSCAN will mask any activity it detects on CAN. The RXCAN pin is therefore held internally in a recessive state. This locks the MSCAN in sleep mode. WUPE must be set before entering sleep mode to take effect.

The MSCAN is able to leave sleep mode (wake) only when:

- CAN bus activity occurs and WUPE = 1

  or

- the CPU clears the SLPRQ bit

Note

The CPU cannot clear the SLPRQ bit before sleep mode (SLPRQ = 1 and SLPAK = 1) is active.

After wakeup, the MSCAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, if the MSCAN is awakened by a CAN frame, this frame is not received.

The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions will be executed upon wakeup; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN remains in bus-off state after sleep mode was exited, it continues counting the 128 occurrences of 11 consecutive recessive bits.

13.4.5.5 MSCAN Initialization Mode

In initialization mode, any on-going transmission or reception is immediately aborted and synchronization to the CAN bus is lost, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives the TXCAN pin into a recessive state.

Note

The user is responsible for ensuring that the MSCAN is not active when initialization mode is entered. The recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before setting the INITRQ bit in the

CAN_CTL0 register. Otherwise, the abort of an on-going message can cause an error condition and can impact other CAN bus devices.

In initialization mode, the MSCAN is stopped. However, interface registers remain accessible. This mode is used to reset the CAN_CTL0, CAN_RFLG, CAN_RIER, CAN_TFLG, CAN_TIER, CAN_TARQ, CAN_TAAK, and CAN_TBSEL registers to their default values. In addition, the MSCAN enables the configuration of the CAN_BTR0, CAN_BTR1 bit timing registers; CAN_IDAC; and the CAN_IDAR, CAN_IDMR message filters.

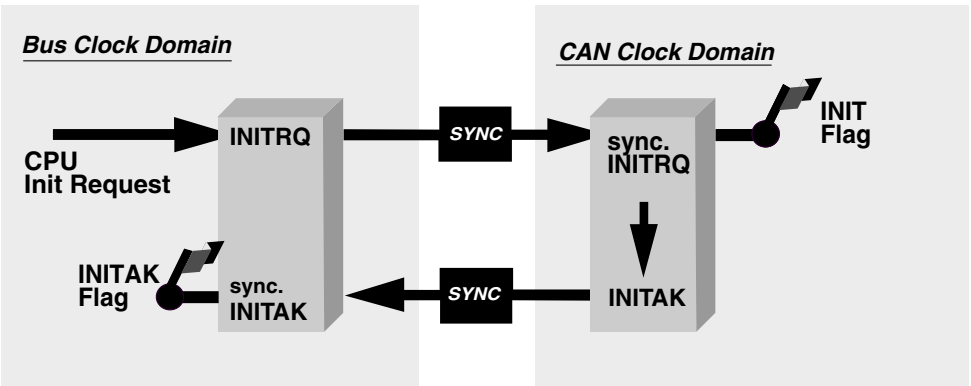


Figure 13-88. Initialization Request/Acknowledge Cycle

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay.

If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

Note

The CPU cannot clear INITRQ before initialization mode (INITRQ = 1 and INITAK = 1) is active.

13.4.5.6 MSCAN Power Down Mode

The MSCAN is in power down mode when

- CPU is in stop mode

or

- CPU is in wait mode and the CSWAI bit is set

When entering the power down mode, the MSCAN immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN immediately drives the TXCAN pin into a recessive state.

Note

The user is responsible for ensuring that the MSCAN is not active when power down mode is entered. The recommended procedure is to bring the MSCAN into Sleep mode before the STOP or WAIT instruction (if CSWAI is set) is executed. Otherwise, the abort of an ongoing message can cause an error condition and impact other CAN bus devices.

In power down mode, all clocks are stopped and no registers can be accessed. If the MSCAN was not in sleep mode before power down mode became active, the module performs an internal recovery cycle after powering up. This causes some fixed delay before the module enters normal mode again.

13.4.5.7 Programmable Wake-Up Function

The MSCAN can be programmed to wake up the MSCAN as soon as CAN bus activity is detected (see control bit WUPE). The sensitivity to existing CAN bus action can be modified by applying a low-pass filter function to the RXCAN input line while in sleep mode (see control bit WUPM).

This feature can be used to protect the MSCAN from wake-up due to short glitches on the CAN bus lines. Such glitches can result from—for example—electromagnetic interference within noisy environments.

13.4.6 Reset Initialization

The reset state of each individual bit is listed in the register definitions, which detail all the registers and their bitfields.

13.4.7 Interrupts

This section describes all interrupts originated by the MSCAN. It documents the enable bits and generated flags. Each interrupt is listed and described separately.

13.4.7.1 Description of Interrupt Operation

The MSCAN supports four interrupt vectors, any of which can be individually masked.

Table 13-87. Interrupt Vectors

Interrupt Source	Interrupt Mask	Local Enable
Wake-Up Interrupt (WUPIF)	INTC_IPR2 Bit 3-2	CAN_RIER (WUPIE)
Error Interrupts Interrupt (CSCIF, OVRIF)	INTC_IPR2 Bit 1-0	CAN_RIER (CSCIE, OVRIE)
Receive Interrupt (RXF)	INTC_IPR1 Bit 15-14	CAN_RIER (RXFIE)
Transmit Interrupts (TXE[2:0])	INTC_IPR1 Bit 13-12	CAN_TIER (TXEIE[2:0])

13.4.7.2 Transmit Interrupt

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The transmit interrupt is generated if the TXEx flag of the empty message buffer is set.

13.4.7.3 Receive Interrupt

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

13.4.7.4 Wakeup Interrupt

A wakeup interrupt is generated if activity on the CAN bus occurs during MSCAN internal sleep mode. WUPE must be enabled.

13.4.7.5 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurrs and the interrupt is not masked. CAN_RFLG indicates one of the following conditions:

- **Overrun** — An overrun condition of the receiver FIFO as described in Receive Structures, occurred.
- **CAN Status Change** — The actual value of the transmit and receive error counters control the CAN bus state of the MSCAN. As soon as the error counters skip into a critical range (Tx/Rx-warning, Tx/Rx-error, bus-off) the MSCAN flags an error condition. The status change, which caused the error condition, is indicated by the TSTAT and RSTAT flags.

13.4.7.6 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either CAN_RFLG or CAN_TFLG ." Interrupts are pending as long as one of the corresponding flags is set. The flags in CAN_RFLG and CAN_TFLG must be reset within the interrupt handler to handshake the interrupt. The flags are reset by writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition prevails.

Note

The CPU must clear only the bit causing the current interrupt. For this reason, bit manipulation instructions (BFSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags that are set after entering the current interrupt service routine.

13.4.7.7 Recovery from Stop or Wait

The MSCAN can recover from stop or wait via the wakeup interrupt. This interrupt can occur only if the MSCAN was in sleep mode (SLPRQ=1 and SLPAK=1) before entering power down mode, the wakeup option is enabled (WUPE=1), and the wakeup interrupt is enabled (WUPIE=1).

13.5 Initialization Application Information

13.5.1 MSCAN initialization

The procedure to initially start up the MSCAN module out of reset is as follows:

1. Assert CANE
2. Write to the configuration registers in initialization mode
3. Clear INITRQ to leave initialization mode and enter normal mode

If the configuration of registers that are writable in initialization mode needs to be changed only when the MSCAN module is in normal mode:

1. Bring the module into sleep mode by setting SLPRQ and awaiting SLPAK to assert after the CAN bus becomes idle.
2. Enter initialization mode: assert INITRQ and await INITAK
3. Write to the configuration registers in initialization mode
4. Clear INITRQ to leave initialization mode and continue in normal mode

13.5.2 Bus-Off Recovery

The bus-off recovery is user configurable. The bus-off state can either be left automatically or on user request.

For reasons of backwards compatibility, the MSCAN defaults to automatic recovery after reset. In this case, the MSCAN will become error active again after counting 128 occurrences of 11 consecutive recessive bits on the CAN bus (See the Bosch CAN specification for details).

If the MSCAN is configured for user request (BORM set in CAN_CTL1), the recovery from bus-off starts after both independent events have become true:

- 128 occurrences of 11 consecutive recessive bits on the CAN bus have been monitored
- BOHOLD in CAN_MISC has been cleared by the user

These two events may occur in any order.

Chapter 14
Interrupt Controller (INTC)

14.1 Introduction

The Interrupt Controller (INTC) module arbitrates among the various interrupt requests (IRQs). The module supports unique interrupt vectors and programmable interrupt priority. It signals to the DSC core when an interrupt of sufficient priority exists and to what address to jump to service this interrupt.

14.1.1 References

- DSP56800E DSC Core Reference Manual

14.1.2 Features

The INTC module design has these distinctive features:

- Programmable priority levels for each IRQ
- Two programmable fast interrupts
- Notification to System Integration Module (SIM) to restart clocks when exiting wait and stop modes
- Driving of initial address on the address bus after reset

14.1.3 Modes of Operation

The INTC module design has these major modes of operation: functional mode and wait and stop modes. Wait and stop modes are really a functional mode, but with limitations, so they are described separately.

- Functional mode

  The INTC is in this mode by default.

- Wait and stop mode operation

  In wait and stop modes, the system clocks and the core are turned off. The INTC signals a pending IRQ to the System Integration Module (SIM) to restart the clocks and service the IRQ. An IRQ can wake up the core only if the IRQ is enabled prior to entering wait or stop mode.

14.1.4 Block Diagram

The following figure is a block diagram of the INTC module.

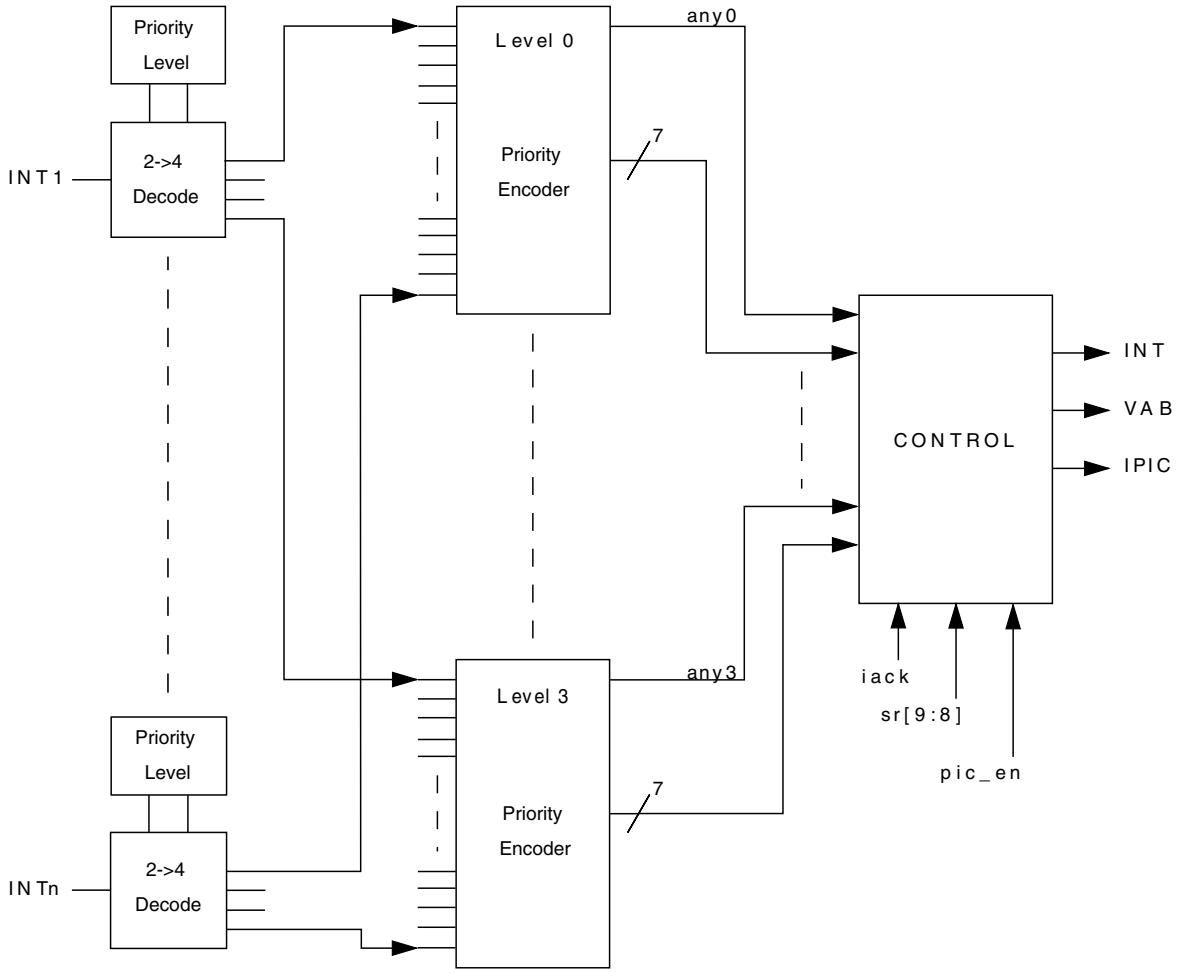


Figure 14-1. Interrupt Controller Block Diagram

14.2 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INTC_IPR0	R	TMRB_3		OCCS		LVI		RX_REG		TX_REG		TRBUF		BKPT		STPCNT	
		W																
1	INTC_IPR1	R	CAN_RX		CAN_TX		ADC_ERR		ADC_CC0		ADC_CC1		TMRB_0		TMRB_1		TMRB_2	
		W																
2	INTC_IPR2	R	QSCI0_ RCV		QSCI0_ RERR		QSCI1_ TDRE		QSCI1_ TIDLE		QSCI1_ RCV		QSCI1_ RERR		CAN_ WAKEUP		CAN_ERR	
		W																
3	INTC_IPR3	R	TMRA_2		TMRA_3		IIC0		IIC1		QSPI_RCV		QSPI_ XMIT		QSCI0_ TDRE		QSCI0_ TIDLE	
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	INTC_IPR4	R	PWM_ RELOAD2		PWM_ CMP3		PWM_ CAP3		PWM_ RELOAD3		PWM_ RERR		PWM_ FAULT		TMRA_0		TMRA_1	
		W																
5	INTC_IPR5	R	HFM_CBE		HFM_CC		HFM_ERR		PWM_ CMP0		PWM_ RELOAD0		PWM_ CMP1		PWM_ RELOAD1		PWM_ CMP2	
		W																
6	INTC_IPR6	R	GPIOB		GPIOC		GPIOD		GPIOE		GPIOF		CMPA		CMPB		CMPC	
		W																
7	INTC_IPR7	R	0														GPIOA	
		W																
8	INTC_VBA	R	0								VECTOR_BASE_ADDRESS							
		W																
9	INTC_FIM0	R	0								FAST_INTERRUPT_0							
		W																
A	INTC_FIVAL0	R	FI_0_VECTOR_ADDRESS_LOW															
		W																
B	INTC_FIVAH0	R	0												FI_0_VECTOR_ADDR_hIGH			
		W																
C	INTC_FIM1	R	0								FAST_INTERRUPT_1							
		W																
D	INTC_FIVAL1	R	FI_1_VECTOR_ADDRESS_LOW															
		W																
E	INTC_FIVAH1	R	0												FI_1_VECTOR_ADDR_hIGH			
		W																
F	INTC_IRQP0	R	PENDING[16:2]															1
		W																
10	INTC_IRQP1	R	PENDING[32:17]															
		W																
11	INTC_IRQP2	R	PENDING[48:33]															
		W																
12	INTC_IRQP3	R	PENDING[64:49]															
		W																

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
13	INTC_IRQP4	R	1														PENDING[66:65]	
		W																
14	INTC_CTRL	R	INT	IPIC			VAB						INT_DIS	1			0	
		W																

14.2.1 Interrupt Priority Register 0 (INTC_IPR0)

Address: INTC_IPR0 – F0C0h base + 0h offset = F0C0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	TMRB_3		OCCS		LVI		RX_REG		TX_REG		TRBUF		BKPT		STPCNT	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR0 field descriptions

Field	Description
15–14 TMRB_3	Timer B Channel 3 Interrupt Priority Level This field is used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 OCCS	PLL Loss of Reference or Change in Lock Status Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
11–10 LVI	Low Voltage Detector Interrupt Priority Level

Table continues on the next page...

INTC_IPR0 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
9–8 RX_REG	EOnCE Receive Register Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
7–6 TX_REG	EOnCE Transmit Register Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
5–4 TRBUF	EOnCE Trace Buffer Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
3–2 BKPT	EOnCE Breakpoint Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
1–0 STPCNT	EOnCE Step Counter Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1

Table continues on the next page...

INTC_IPR0 field descriptions (continued)

Field	Description
	10 IRQ Priority Level 2 11 IRQ Priority Level 3

14.2.2 Interrupt Priority Register 1 (INTC_IPR1)

Address: INTC_IPR1 – F0C0h base + 1h offset = F0C1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	CAN_RX		CAN_TX		ADC_ERR		ADC_CC0		ADC_CC1		TMRB_0		TMRB_1		TMRB_2	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR1 field descriptions

Field	Description
15–14 CAN_RX	CAN Receive Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 CAN_TX	CAN Transmit Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 ADC_ERR	ADC Zero Crossing, High Limit, or Low Limit Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 ADC_CC0	ADC Conversion Complete Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default.

Table continues on the next page...

INTC_IPR1 field descriptions (continued)

Field	Description
	00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 ADC_CC1	ADC Conversion Complete Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 TMRB_0	Timer B Channel 0 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 TMRB_1	Timer B Channel 1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
1–0 TMRB_2	Timer B Channel 2 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

14.2.3 Interrupt Priority Register 2 (INTC_IPR2)

Address: INTC_IPR2 – F0C0h base + 2h offset = F0C2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	QSCI0_RCV		QSCI0_ RERR		QSCI1_ TDRE		QSCI1_ TIDLE		QSCI1_RCV		QSCI1_ RERR		CAN_ WAKEUP		CAN_ERR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR2 field descriptions

Field	Description
15–14 QSCI0_RCV	QSCI 0 Receiver Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 QSCI0_RERR	QSCI 0 Receiver Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 QSCI1_TDRE	QSCI 1 Transmitter Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 QSCI1_TIDLE	QSCI 1 Transmitter Idle Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 QSCI1_RCV	QSCI 1 Receiver Full Interrupt Priority Level

Table continues on the next page...

INTC_IPR2 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 QSCI1_RERR	QSCI 1 Receiver Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 CAN_WAKEUP	CAN Wake Up Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
1–0 CAN_ERR	CAN Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

14.2.4 Interrupt Priority Register 3 (INTC_IPR3)

Address: INTC_IPR3 – F0C0h base + 3h offset = F0C3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	TMRA_2		TMRA_3		IIC0		IIC1		QSPI_RCV		QSPI_XMIT		QSCI0_ TDRE		QSCI0_ TIDLE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR3 field descriptions

Field	Description
15–14 TMRA_2	Timer A Channel 2 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 TMRA_3	Timer A Channel 3 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 IIC0	IIC0 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 IIC1	IIC1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 QSPI_RCV	QSPI Receiver Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 QSPI_XMIT	QSPI Transmit Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default)

Table continues on the next page...

INTC_IPR3 field descriptions (continued)

Field	Description
	01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 QSCI0_TDRE	QSCI 0 Transmitter Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
1–0 QSCI0_TIDLE	QSCI 0 Transmitter Idle Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

14.2.5 Interrupt Priority Register 4 (INTC_IPR4)

Address: INTC_IPR4 – F0C0h base + 4h offset = F0C4h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PWM_ RELOAD2		PWM_CMP3		PWM_CAP3		PWM_ RELOAD3		PWM_RERR		PWM_ FAULT		TMRA_0		TMRA_1	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR4 field descriptions

Field	Description
15–14 PWM_RELOAD2	PWM Submodule 2 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 PWM_CMP3	PWM Submodule 3 Compare Interrupt Priority Level

Table continues on the next page...

INTC_IPR4 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 PWM_CAP3	PWM Submodule 3 Capture Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 PWM_RELOAD3	PWM Submodule 3 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 PWM_RERR	PWM Reload Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 PWM_FAULT	PWM Fault Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 TMRA_0	Timer A Channel 0 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0

Table continues on the next page...

INTC_IPR4 field descriptions (continued)

Field	Description
	10 IRQ Priority Level 1 11 IRQ Priority Level 2
1–0 TMRA_1	Timer A Channel 1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

14.2.6 Interrupt Priority Register 5 (INTC_IPR5)

Address: INTC_IPR5 – F0C0h base + 5h offset = F0C5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	HFM_CBE		HFM_CC		HFM_ERR		PWM_CMP0		PWM_ RELOAD0		PWM_CMP1		PWM_ RELOAD1		PWM_CMP2	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR5 field descriptions

Field	Description
15–14 HFM_CBE	HFM Command, Data, Address Buffers Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 HFM_CC	HFM Command Complete Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 HFM_ERR	HFM Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default.

Table continues on the next page...

INTC_IPR5 field descriptions (continued)

Field	Description
	00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 PWM_CMP0	PWM Submodule 0 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 PWM_RELOAD0	PWM Submodule 0 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 PWM_CMP1	PWM Submodule 1 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 PWM_RELOAD1	PWM Submodule 1 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
1–0 PWM_CMP2	PWM Submodule 2 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

14.2.7 Interrupt Priority Register 6 (INTC_IPR6)

Address: INTC_IPR6 – F0C0h base + 6h offset = F0C6h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	GPIOB		GPIOC		GPIOD		GPIOE		GPIOF		CMPA		CMPB		CMPC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR6 field descriptions

Field	Description
15–14 GPIOB	GPIO B Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 GPIOC	GPIO C Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 GPIOD	GPIO D Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 GPIOE	GPIO E Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 GPIOF	GPIO F Interrupt Priority Level

Table continues on the next page...

INTC_IPR6 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 CMPA	Comparator A Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 CMPB	Comparator B Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
1–0 CMPC	Comparator C Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

14.2.8 Interrupt Priority Register 7 (INTC_IPR7)

Address: INTC_IPR7 – F0C0h base + 7h offset = F0C7h

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

INTC_IPR7 field descriptions

Field	Description
15–2 Reserved	This read-only bitfield is reserved and always has the value zero.
1–0 GPIOA	GPIO A Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

14.2.9 Vector Base Address Register (INTC_VBA)

Address: INTC_VBA – F0C0h base + 8h offset = F0C8h

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

INTC_VBA field descriptions

Field	Description
15–13 Reserved	This read-only bitfield is reserved and always has the value zero.
12–0 VECTOR_ BASE_ ADDRESS	Interrupt Vector Base Address The value in this register is used as the upper 13 bits of the interrupt vector VAB[20:0]. The lower 8 bits are determined based on the highest priority interrupt and are then appended onto the VECTOR_BASE_ADDRESS before presenting the full Vector Address Bus [VAB] to the Core. **NOTE:** The 64KB Flash part resets to a value of 16'h0000. The 48KB part resets to a value of 16'h0020. The 32KB part resets to a value of 16'h0040. The 16KB Flash part resets to a value of 16'h0060. These correspond to reset addresses of 21'h000000, 21'h002000, 21'h004000, and 21'h006000.

14.2.10 Fast Interrupt 0 Match Register (INTC_FIM0)

Address: INTC_FIM0 – F0C0h base + 9h offset = F0C9h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								FAST_INTERRUPT_0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIM0 field descriptions

Field	Description
15–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6–0 FAST_ INTERRUPT_0	Fast Interrupt 0 Vector Number These values are used to declare which two IRQs will be Fast Interrupts. Fast Interrupts vector directly to a service routine based on values in the Fast Interrupt Vector Address registers without having to go to a jump table first. IRQs used as fast interrupts MUST be set to priority level 2. Unexpected results will occur if a fast interrupt vector is set to any other priority. Fast interrupts automatically become the highest priority level 2 interrupt regardless of their location in the interrupt table prior to being declared as fast interrupt. Fast interrupt 0 has priority over fast interrupt 1. To determine the vector number of each IRQ, refer to the vector table in the memory section of the system specification in this document.

14.2.11 Fast Interrupt 0 Vector Address Low Register (INTC_FIVAL0)

This low register is combined with the corresponding high register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: INTC_FIVAL0 – F0C0h base + Ah offset = F0CAh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FI_0_VECTOR_ADDRESS_LOW															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

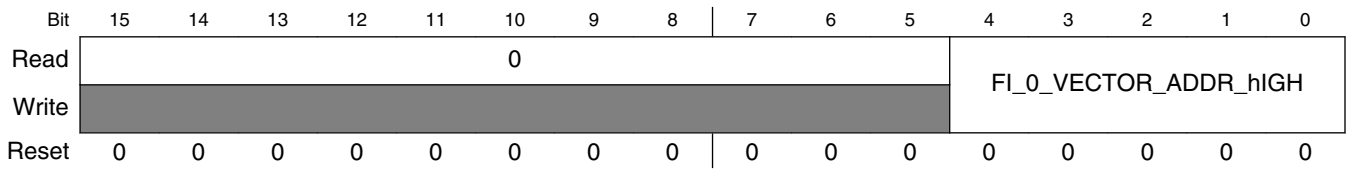
INTC_FIVAL0 field descriptions

Field	Description
15–0 FI_0_VECTOR_ ADDRESS_LOW	Lower 16 bits of vector address for fast interrupt 0

14.2.12 Fast Interrupt 0 Vector Address High Register (INTC_FIVAH0)

This high register is combined with the corresponding low register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: INTC_FIVAH0 – F0C0h base + Bh offset = F0CBh

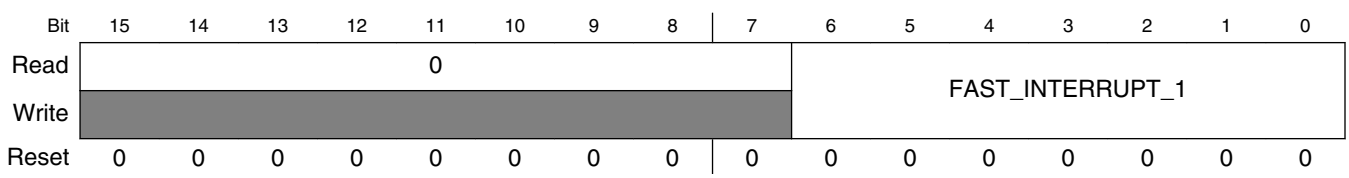


INTC_FIVAH0 field descriptions

Field	Description
15–5 Reserved	This read-only bitfield is reserved and always has the value zero.
4–0 FI_0_VECTOR_ADDR_hIGH	Upper 5 bits of vector address for fast interrupt 0

14.2.13 Fast Interrupt 1 Match Register (INTC_FIM1)

Address: INTC_FIM1 – F0C0h base + Ch offset = F0CCh



INTC_FIM1 field descriptions

Field	Description
15–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6–0 FAST_INTERRUPT_1	Fast Interrupt 1 Vector Number These values are used to declare which two IRQs will be Fast Interrupts. Fast Interrupts vector directly to a service routine based on values in the Fast Interrupt Vector Address registers without having to go to a jump table first. IRQs used as fast interrupts MUST be set to priority level 2. Unexpected results will occur if a fast interrupt vector is set to any other priority. Fast interrupts automatically become the highest priority level 2 interrupt regardless of their location in the interrupt table prior to being declared as fast interrupt. Fast interrupt 0 has priority over fast interrupt 1. To determine the vector number of each IRQ, refer to the vector table in the memory section of the system specification in this document.

14.2.14 Fast Interrupt 1 Vector Address Low Register (INTC_FIVAL1)

This low register is combined with the corresponding high register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: INTC_FIVAL1 – F0C0h base + Dh offset = F0CDh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	FI_1_VECTOR_ADDRESS_LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAL1 field descriptions

Field	Description
15–0 FI_1_VECTOR_ADDRESS_LOW	Lower 16 bits of vector address for fast interrupt 1

14.2.15 Fast Interrupt 1 Vector Address High Register (INTC_FIVAH1)

This high register is combined with the corresponding low register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: INTC_FIVAH1 – F0C0h base + Eh offset = F0CEh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0											FI_1_VECTOR_ADDR_hIGH				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAH1 field descriptions

Field	Description
15–5 Reserved	This read-only bitfield is reserved and always has the value zero.
4–0 FI_1_VECTOR_ADDR_hIGH	Upper 5 bits of vector address for fast interrupt 1

14.2.16 IRQ Pending Register 0 (INTC_IRQP0)

Address: INTC_IRQP0 – F0C0h base + Fh offset = F0CFh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[16:2]															1
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP0 field descriptions

Field	Description
15–1 PENDING[16:2]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 through 66. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number
0 Reserved	This read-only bit is reserved and always has the value one.

14.2.17 IRQ Pending Register 1 (INTC_IRQP1)

Address: INTC_IRQP1 – F0C0h base + 10h offset = F0D0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[32:17]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP1 field descriptions

Field	Description
15–0 PENDING[32:17]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 through 66. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

14.2.18 IRQ Pending Register 2 (INTC_IRQP2)

Address: INTC_IRQP2 – F0C0h base + 11h offset = F0D1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[48:33]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP2 field descriptions

Field	Description
15–0 PENDING[48:33]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 through 66. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

14.2.19 IRQ Pending Register 3 (INTC_IRQP3)

Address: INTC_IRQP3 – F0C0h base + 12h offset = F0D2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[64:49]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP3 field descriptions

Field	Description
15–0 PENDING[64:49]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 through 66. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

14.2.20 IRQ Pending Register 4 (INTC_IRQP4)

Address: INTC_IRQP4 – F0C0h base + 13h offset = F0D3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	1														PENDING[66:65]	
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP4 field descriptions

Field	Description
15–2 Reserved	This read-only bitfield is reserved and always has the value one.
1–0 PENDING[66:65]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 through 66. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

14.2.21 Control Register (INTC_CTRL)

Address: INTC_CTRL – F0C0h base + 14h offset = F0D4h

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

INTC_CTRL field descriptions

Field	Description
15 INT	Interrupt This bit reflects the state of the interrupt to the core.

Table continues on the next page...

INTC_CTRL field descriptions (continued)

Field	Description
	0 No interrupt is being sent to the core. 1 An interrupt is being sent to the core.
14–13 IPIC	Interrupt Priority Level These bits reflect the new interrupt priority level bits being sent to the Core. These bits indicate the priority level needed for a new IRQ to interrupt the current interrupt being sent to the Core. This field is only updated when the DSC core jumps to a new interrupt service routine. 00 Required nested exception priority levels are 0, 1, 2, or 3. 01 Required nested exception priority levels are 1, 2, or 3. 10 Required nested exception priority levels are 2 or3. 11 Required nested exception priority level is 3.
12–6 VAB	Vector number This field shows bits [7:1] of the Vector Address Bus used at the time the last IRQ was taken. In the case of a fast interrupt, it shows the lower address bits of the jump address. This field is only updated when the DSC core jumps to a new interrupt service routine.
5 INT_DIS	Interrupt disable This bit allows the user to disable all interrupts. 0 Normal operation. (default) 1 All interrupts disabled.
4–2 Reserved	This read-only bitfield is reserved and always has the value one.
1–0 Reserved	This read-only bitfield is reserved and always has the value zero.

14.3 Functional Description

The Interrupt Controller is a slave on the IP bus. It contains registers that allow each of the interrupt sources to be set to one of four priority levels (excluding certain interrupts that have fixed priority). All of the interrupt requests of a given level are priority encoded to determine the lowest numerical value of the active interrupt requests for that level. Within a given priority level, number 0 is the highest priority, and number *n*-1 (where *n* is the total number of interrupt sources) is the lowest priority.

14.3.1 Normal Interrupt Handling

After the INTC determines that an interrupt is to be serviced and which interrupt has the highest priority, an interrupt vector address is generated. Normal interrupt handling concatenates the vector base address (VBA) and the vector number to determine the vector address. In this way, an offset into the vector table is generated for each interrupt.

14.3.2 Interrupt Nesting

Interrupt exceptions may be nested to allow the servicing of an IRQ with higher priority than the current exception. The DSC core controls the masking of interrupt priority levels by setting the I0 and I1 bits in its status register (SR).

Table 14-23. Interrupt Mask Bit Settings in Core Status Register

I1 (SR[9])	I0 (SR[8])	Exceptions Permitted	Exceptions Masked
0	0	Priorities 0, 1, 2, 3	None
0	1	Priorities 1, 2, 3	Priority 0
1	0	Priorities 2, 3	Priorities 0, 1
1	1	Priority 3	Priorities 0, 1, 2

The IPIC field of the INTC module's CTRL register reflects the state of the priority level that is presented to the DSC core.

Table 14-24. Interrupt Priority Level Field Settings

IPIC	Current Interrupt Priority Level	Required Nested Exception Priority
00	No interrupt or SWILP	Priorities 0, 1, 2, 3
01	Priority 0	Priorities 1, 2, 3
10	Priority 1	Priorities 2, 3
11	Priorities 2 or 3	Priority 3

14.3.3 Fast Interrupt Handling

Fast interrupt processing is described in section 9.3.2.2 of the *DSP56800E DSC Core Reference Manual*. The Interrupt Controller recognizes fast interrupts before the core does.

A fast interrupt is defined (to the INTC) by:

1. Setting the priority of the interrupt as level 2, using the appropriate field in the IPR registers.
2. Setting the FIMn register to the appropriate vector number.
3. Setting the FIVALn and FIVAHn registers with the address of the code for the fast interrupt.

When an interrupt occurs, its vector number is compared with the FIM0 and FIM1 register values. If a match occurs, and if the interrupt priority is level 2, the INTC handles the interrupt as a fast interrupt. The INTC takes the vector address from the appropriate FIVALn and FIVAHn registers, instead of generating an address that is an offset from the vector base address (VBA).

The core then fetches the instruction from the indicated vector address. If the instruction is not a JSR, the core starts its fast interrupt handling.

14.4 Resets

Table 14-25. Reset Summary

Reset	Priority	Source	Characteristics
Core reset		RST_B	Core reset from the SIM

14.4.1 INTC after Reset

After reset, all INTC registers are in their default states. As a result, all interrupts are disabled except the core IRQs with fixed priorities (Illegal Instruction, SW Interrupt 3, HW Stack Overflow, Misaligned Long Word Access, SW Interrupt 2, SW Interrupt 1, SW Interrupt 0, and SW Interrupt LP), which are enabled at their fixed priority levels.

Chapter 15
On-Chip Clock Synthesis (OCCS)

15.1 Introduction

15.1.1 Overview

This module provides the 2X system clock frequency to the system integration module (SIM), which then generates the various derivative system and peripheral clocks for the chip.

The on-chip clock synthesis module allows product design using several user selectable clock sources including an internal relaxation oscillator and PLL to run at frequencies up to a 60MHz system clock.

15.1.2 Features

The on-chip clock synthesis (OCCS) module interfaces to the oscillator and PLL. The device has more options for clock generation than do the other members of the MC56F8000 family. The OCCS clock sources include:

- On-Chip Relaxation Oscillator (ROSC). This module nominally generates an 8MHz clock signal. It is also capable of operating at 400KHz when the device is in low power mode.
- Crystal Oscillator (COSC). This module is designed for use with a crystal or resonator in the 4 to 16MHz range. When used with the on-chip PLL of this device, the maximum crystal/resonator frequency is 16MHz.
- Off chip external clock source

Additional OCCS module features are as follows:

- Ability to power down internal relaxation oscillator
- Ability to put the internal relaxation oscillator into a 400KHz standby mode
- Ability to power down external oscillator
- 8-bit postscaler provides operates on either PLL output or, in the case where the PLL is not in use, one of the oscillators or external clock source
- Ability to power down the internal PLL
- Provides 2X master clock frequency and 2X High Speed Peripheral clock signals.
- Can be driven from an external clock source

The clock generation module provides the programming interface for the PLL and internal relaxation oscillator and crystal oscillator.

Key features of the crystal oscillator module are:

- Supports 4MHz - 16MHz crystals and resonators
- Automatic Gain Control (AGC) to optimize power consumption in both frequency ranges using low-power mode
- High gain option in both frequency ranges
- Voltage and frequency filtering to guarantee clock frequency and stability

15.2 Modes of Operation

Either an internal relaxation oscillator, crystal oscillator, or an external frequency source can be used to provide a reference clock (sys_clk_2x) to the SIM.

The 2X system clock source output from the OCCS can be described by one of the following equations:

2X system frequency = (oscillator frequency) / (postscaler)

2X system frequency = (oscillator frequency X 15) / (postscaler)

where:

- postscaler = 1, 2, 4, 8, 16, 32, 64, 128 or 256 PLL output divider

The SIM is responsible for further dividing these frequencies by two, which will insure a 50% duty cycle in the system clock output.

The on-chip clock synthesis module of has the following registers:

- PLL control register (CTRL)
- Divide-by register (DIVBY)
- OCCS Status Register (STAT)
- Oscillator Control register (OSCTL)
- Clock Check Reference (CLKCHKR)
- Clock Check Target (CLKCHKT)
- Protect (PROT)

For more information on these registers please refer to the register details section.

15.2.1 Internal Clock Source

The internal relaxation oscillator is optimized for accuracy and programmability while providing several different power saving configurations to accommodate different operating conditions. The internal oscillator has a maximum variation of only +/-3% due to temperature and voltage, but it varies as much as +/-20% as a function of wafer fabrication process. It also is very fast in reaching a stable frequency (well under 1 µs). Under typical conditions the circuit provides an 8 MHz clock at the center of its tuning range. The tuning range is controlled by 10 bits, with each tuning bit providing a binary weighted change from the previous bit. The maximum tuning step size is 40% while the minimum tuning step size is 0.08%. To optimize power, the architecture supports a standby state and a power-down state. During the reset sequence, the internal oscillator will be enabled by default. Application code can then switch to the external source or crystal oscillator and power down the internal oscillator if desired.

15.2.2 Crystal (or Ceramic Resonator) Oscillator

The internal crystal oscillator circuit is designed to interface with a parallel-resonant crystal resonator in the frequency range of 4-16 MHz. A ceramic resonator can be substituted for the external crystal. When used to supply a source to the internal PLL, the crystal/resonator must be in the 4MHz to 16MHz range. Oscillator circuits are shown in the following figures. Follow the crystal supplier's recommendations when selecting a crystal, since crystal parameters determine the component values required to provide maximum stability and reliable start-up. The load capacitance values used in the

oscillator circuit design should include all stray layout capacitances. The crystal and associated components should be mounted as close as possible to the EXTAL and XTAL pins to minimize output distortion and start-up stabilization time.

When using low-frequency, low-power mode, the only external component is the crystal 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 device data sheet. .

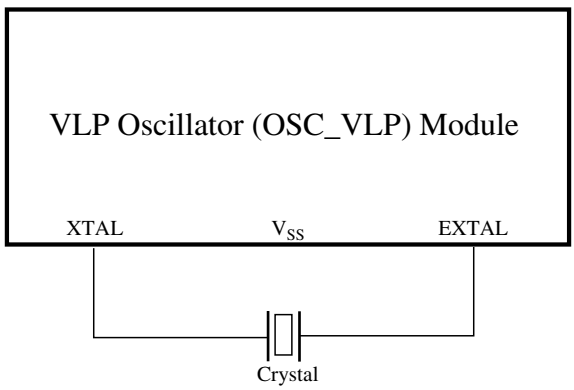


Figure 15-1. Crystal Connections - Low-Frequency, Low-Power Mode

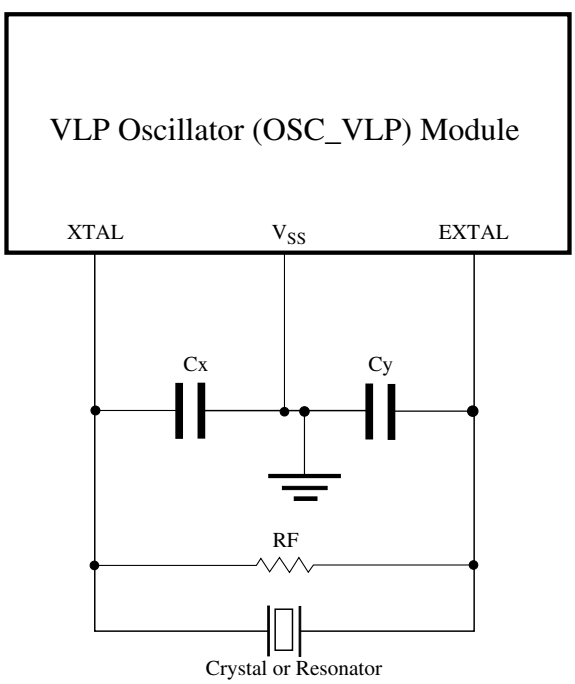


Figure 15-2. Crystal/Resonator Connections - High-Frequency, Low-Power Mode

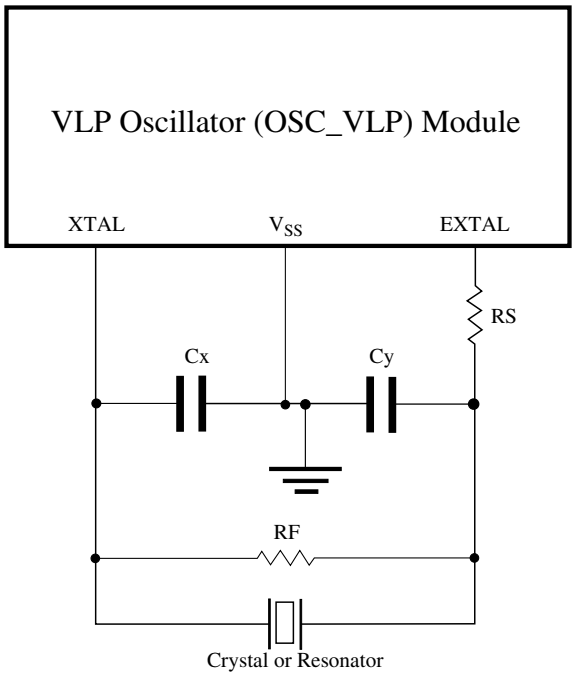


Figure 15-3. Crystal/Resonator Connections - High-Gain Modes

15.2.3 External Clock Source - Clock In (CLKIN)

The recommended method of connecting an external clock is given in the following figure. If the external Clock In (CLKIN) is selected as the device external clock in put, set both the C0 bit in GPS0 in the SIM module and the EXT_SEL bit in OSCTL, and power down the crystal oscillator.

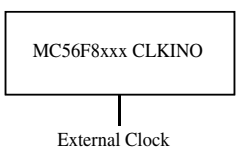


Figure 15-4. Connecting an External Clock Signal using GPIO

15.3 Block Diagram

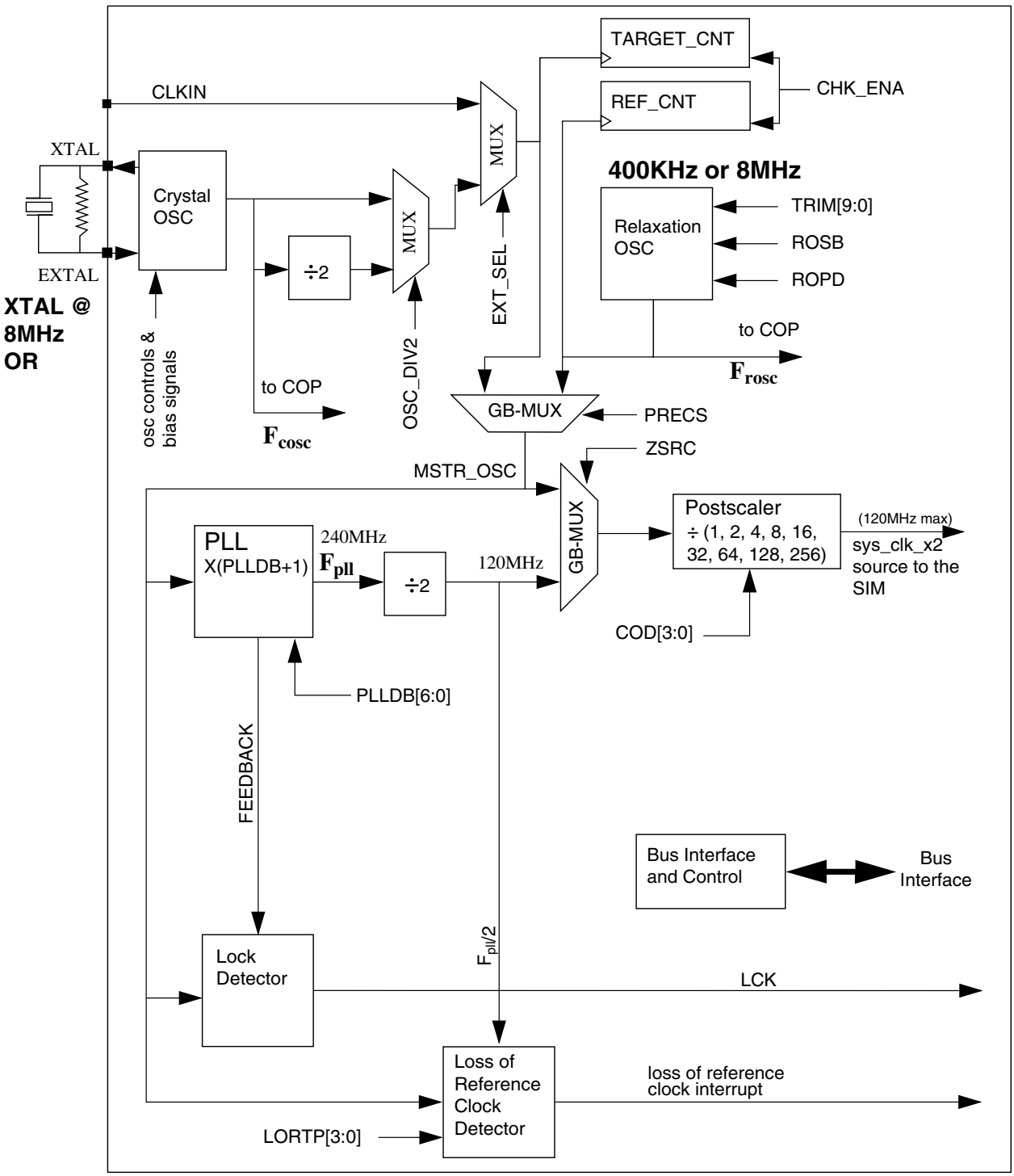


Figure 15-5. OCCS Block Diagram with Crystal Oscillator

The preceding block diagram of the clock generation module shows how this block differs from that found on the MC56F802x/3x series of devices in the following ways:

- The postscaler divisor now can be as large as 256.
- The postscaler can now also be used to divide down an external clock or one of the oscillator outputs for use as sys_clk_x2.

15.4 Pin Descriptions

15.4.1 External Reference

The relaxation oscillator is always included on chip and the reset mode is to use this as the clock source for the chip. The customer then has the option of switching to an external clock reference if desired.

On this device , GPIOC0 can be programmed to be the external clock input through the C0 bit of GPS0 in the SIM module.

15.4.2 Oscillator Inputs (XTAL, EXTAL)

The oscillator inputs can be used to connect an external crystal, ceramic resonator, or to directly drive the chip with an external clock source, thus bypassing the internal crystal oscillator circuit.

15.4.3 CLKO

This family of DSCs will usually have a CLKO pin which can be programmed to bring out any of a number of internal clock signals to the outside world. CLKO functionality exists in the System Integration Module (SIM). It is mentioned here because a number of OCCS clocks will be made available for use on that pin.

Note

WARNING There is no defined relationship between the signals present on CLKO and their internal counterparts. CLKO is useful for observing internal frequencies, but cannot be used to sequence data onto or off of the chip.

15.5 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OCCS_CTRL	R	PLLIE1		PLLIE0		LOCIE	0			LCKON	0		PLLPD	0		PRECS	ZSRC	
		W																
1	OCCS_DIVBY	R	LORTP				COD				0	PLLDB						
		W																
2	OCCS_STAT	R	LOLI1	LOLI0	LOCI	0					LCK1	LCK0	PLLPDN	RESERVED 0				ZSRCS
		W									—	—	—					—
4	OCCS_OSCTL	R	ROPD	ROSB	COHL	CLK_ MODE	OSC_ DIV2	EXT_ SEL	TRIM									
		W																
5	OCCS_ CLKCHKR	R	CHK_ ENA	0							REF_COUNT							
		W																
6	OCCS_ CLKCHKT	R	0								TARGET_COUNT							
		W																
7	OCCS_PROT	R	0											FRQEP	OSCEP		PLLEP	
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

15.5.1 OCCS PLL Control Register (OCCS_CTRL)

Address: OCCS_CTRL – F120h base + 0h offset = F120h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PLLIE1		PLLIE0		LOCIE	0			LCKON	0		PLLPD	0	PRECS	ZSRC	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

OCCS_CTRL field descriptions

Field	Description
15–14 PLLIE1	PLL Interrupt Enable 1 An optional interrupt can be generated when the PLL lock status bit (LCK1) in the OCCS Status Register (STAT) changes. 00 Disable interrupt 01 Enable interrupt on any rising edge of LCK1 10 Enable interrupt on falling edge of LCK1 11 Enable interrupt on any edge change of LCK1
13–12 PLLIE0	PLL Interrupt Enable 0 An optional interrupt can be generated if the PLL lock status bit (LCK0) in the OCCS Status Register (STAT) changes. 00 Disable interrupt 01 Enable interrupt on any rising edge of LCK0 10 Enable interrupt on falling edge of LCK0 11 Enable interrupt on any edge change of LCK0
11 LOCIE	Loss of Reference Clock Interrupt Enable The loss of reference clock circuit monitors the output of the on-chip oscillator circuit. In the event of loss of reference clock, an optional interrupt can be generated. An optional interrupt can be generated if the oscillator circuit output clock is lost. 0 Interrupt disabled 1 Interrupt enabled
10–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 LCKON	Lock Detector On 0 Lock detector disabled 1 Lock detector enabled
6–5 Reserved	This read-only bitfield is reserved and always has the value zero.
4 PLLPD	PLL Power Down The PLL can be turned off by setting the PLLPD bit. There is a four IPbus clock delay from changing the bit to signaling the PLL. When the PLL is powered down, the gear shifting logic automatically switches to ZSRC[1:0] = 01b in order to prevent loss of reference clock to the core. 0 PLL enabled 1 PLL powered down
3 Reserved	This read-only bit is reserved and always has the value zero.
2 PRECS	Prescaler Clock Select This bit is used to select between (the external clock source or oscillator) and the internal relaxation oscillator. **NOTE:** This bit should not be set unless the external reference is enabled in the GPIO/SIM/KTR.

Table continues on the next page...

OCCS_CTRL field descriptions (continued)

Field	Description
	0 Relaxation Oscillator selected (reset value). 1 External reference selected.
1–0 ZSRC	CLOCK Source The CLOCK source determines the system clock source to the SIM module, which generates divided down versions of this signal for use by memories and IP Bus. To prevent loss of reference clock to the core, ZSRC is automatically set to 01b if PLLPD is set. 00 Reserved 01 MSTR_OSC 10 PLL output 11 Reserved

15.5.2 OCCS PLL Divide-By Register (OCCS_DIVBY)

Address: OCCS_DIVBY – F120h base + 1h offset = F121h

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

OCCS_DIVBY field descriptions

Field	Description
15–12 LORTP	Loss of Reference Clock Trip Point These bits control the amount of time required for the loss of reference clock interrupt to be generated. This failure detection time is `((LORTP + 1) x 10) x (reference clock period) / (PLL Multiplier / 2)`. The PLL Multiplier is set by the PLLDB register.
11–8 COD	Clock Output Divide or Postscaler The PLL output clock can be divided down by a 4-bit postscaler. The input of the postscaler is a selectable clock source for the DSP core as determined by the ZSRC[1:0] in the CTRL register. The output of the postscaler is guaranteed to be glitch free, even when COD has been changed. **NOTE:** There are special restrictions on use of 2X high speed peripheral clocks. 0000 Divide by one 0001 Divide by two 0010 Divide by four 0011 Divide by eight 0100 Divide by 16 0101 divide by 32 0110 Divide by 64

Table continues on the next page...

OCCS_DIVBY field descriptions (continued)

Field	Description
	0111 Divide by 128 1xxx Divide by 256
7 Reserved	This read-only bit is reserved and always has the value zero.
6–0 PLLDB	PLL Divide By The output frequency of the PLL is controlled, in part, by the PLLDB[6:0] field. The value written to this field, plus one, is used by the PLL to directly multiply the input frequency and present it at its output. For example, if the input frequency is 8MHz and the PLLDB[6:0] field is set to 29 (the default), then the PLL output frequency is 240MHz. This yields a 120MHz sys_clk_2x. Dividing this by 2 (hardcoded into the SIM) results in a 60MHz system clock assuming that the postscaler is set to one. Before changing the divide-by value, you must switch the core clock to the MSTR_OSC clock. PLLDB settings should be limited such that the output frequency of the PLL does not exceed 240 MHz. **NOTE:** Upon writing to the DIVBY register, the Loss of Reference detector circuit is reset.

15.5.3 OCCS PLL Status Register (OCCS_STAT)

A PLL interrupt is generated if any of the LOLI or LOCI bits are set and the respective interrupt enable is set in the CTRL.

Address: OCCS_STAT – F120h base + 2h offset = F122h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOLI1	LOLI0	LOCI	0						LCK1	LCK0	PLLPDN	RESERVED		ZSRCS	
Write										—	—	—			—	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

OCCS_STAT field descriptions

Field	Description
15 LOLI1	PLL Loss of Lock Interrupt 1 LOLI1 shows the status of the lock detector state from LCK1 circuit. This bit is cleared by writing a one to LOLI0. This bit will not be set (by the hardware) if the corresponding CTRL PLLIE1 bit is cleared (set to zero). 0 PLL locked 1 PLL not locked

Table continues on the next page...

OCCS_STAT field descriptions (continued)

Field	Description
14 LOLI0	PLL Loss of Lock Interrupt 0 LOLI0 shows the status of the lock detector state from LCK0 circuit. This bit is cleared by writing a one to LOLI0. This bit will not be set (by the hardware) if the corresponding CTRL PLLIE0 bit is cleared (set to zero). 0 PLL locked 1 PLL not locked
13 LOCI	Loss of Reference Clock Interrupt LOCI shows the status of the reference clock detection circuit. This bit is cleared by writing a one to LOCI. 0 Oscillator clock normal 1 Lost of oscillator clock detected
12–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6 LCK1	Loss of Lock 1 0 PLL is unlocked 1 PLL is locked (fine)
5 LCK0	Loss of Lock 0 0 PLL is unlocked 1 PLL is locked (coarse)
4 PLLPDN	PLL Power Down PLL power down status is delayed by four IPbus clocks from the PLLPD bit in the CTRL. 0 PLL not powered down 1 PLL powered down
3–2 Reserved	This bitfield is reserved.
1–0 ZSRCS	ZSRCS indicates the current sys_clk_x2 clock source. Since the synchronizing circuit switches the system clock source, ZSRCS takes more than one IPBUS clock to indicate the new selection. 00 Synchronizing in progress 01 MSTR_OSC 10 Postscaler output 11 Synchronizing in progress

15.5.4 OCCS Oscillator Control Register (OCCS_OSCTL)

This register controls aspects of both the internal relaxation oscillator and the crystal/ resonator oscillator.

Address: OCCS_OSCTL – F120h base + 4h offset = F124h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ROPD	ROSB	COHL	CLK_MODE	OSC_DIV2	EXT_SEL	TRIM									
Reset	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0

OCCS_OSCTL field descriptions

Field	Description
15 ROPD	Relaxation Oscillator Power Down This bit is used to power down the relaxation oscillator. The user may power down the relaxation oscillator if the external reference is being used. In order to prevent loss of clock to the core or the PLL, this bit should only be asserted if the clock source has been changed to the external source by setting the PRECS bit in CTRL. 0 Relaxation Oscillator Enabled 1 Relaxation Oscillator Powered Down
14 ROSB	Relaxation Oscillator Standby This bit is used to control the power usage and gross frequency of the relaxation oscillator. It is reset to the more accurate but higher power state. 0 Normal mode. The relaxation oscillator output frequency is 8MHz. 1 Standby mode. The relaxation oscillator output frequency is reduced to 400KHz(+/-50%). The PLL should be disabled in this mode and MSTR_OSC should be selected as the output clock.
13 COHL	Crystal Oscillator High/Low Power Level This bit is used to control the power usage of the crystal oscillator. It is reset to the high power state, which allows either a crystal or resonator to be used. 0 High power mode. This mode is required as default. 1 Low power mode. This is the desired mode when a less than 8 MHz low power crystal is used .
12 CLK_MODE	Crystal Oscillator Clock Mode This bit turns off the Crystal Oscillator for power savings. **NOTE:** If the crystal oscillator is turned off and then turned on again, the clock should not be switched back to the oscillator until after the crystal has had time to stabilize. See the crystal data sheet to determine this time duration. 0 Crystal oscillator enabled. 1 Crystal oscillator disabled.
11 OSC_DIV2	Oscillator Divide By 2 This bit is used to select to the frequency range of the crystal oscillator PRECS should be 0, ROSC selected, before changing the value of OSC_DIV2 to avoid glitches on the system clock. 0 External Oscillator output is not divided. 1 External Oscillator output is divided by 2 befor use as MSTR_OSC.

Table continues on the next page...

OCCS_OSCTL field descriptions (continued)

Field	Description
10 EXT_SEL	External Clock In Select This bit is used to select the source of the external clock input. PRECS should be 0 before changing the value of EXT_SEL to avoid glitches on the system clock. 0 Use XTAL as the external clock input 1 Use CLKIN as the external clock input
9–0 TRIM	Internal Relaxation Oscillator TRIM bits These bits change the size of the internal capacitor used by the internal relaxation oscillator. By testing the frequency of the internal clock and incrementing or decrementing this factor accordingly, the accuracy of the internal clock can be improved by 40%. Incrementing these bits by one increases the clock period by 0.078% of the unadjusted value. Decrementing this register by one decreases the clock period by 0.078%. Reset sets these bits to 200h, centering the range of possible adjustment.

15.5.5 OCCS External Clock Check Reference (OCCS_CLKCHKR)

CLKCHKR and CLKCHKT are used in applications where it is required to verify the activity of an external clock source or crystal/resonator oscillator before it is selected as the active system clock source. PRECS must be 0 to use this function.

Address: OCCS_CLKCHKR – F120h base + 5h offset = F125h

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

OCCS_CLKCHKR field descriptions

Field	Description
15 CHK_ENA	Check Enable This bit is used to start and stop the clock checking function. Allow enough time after the CLK_ENA is cleared to allow for two ROSC clock periods before attempting to start another verification cycle. 0 Writing a low while the clock checking operation is in progress stops the check in it's current state. Reading a low after a check has been started indicates that the check operation is complete and the final values are valid in the REF_CNT and TARGET_CNT registers. 1 Writing a one clears the REF_CNT and TARGET_CNT registers and starts the clock checking function. The CLK_ENA bit will remain high while the operation is in progress.
14–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6–0 REF_COUNT	Reference Count bits

Table continues on the next page...

OCCS_CLKCHKR field descriptions (continued)

Field	Description
	This count is initialized to zero on the positive transition of CHK_ENA. The test is terminated when: • REF_COUNT = 0080h and • Number of ROSC clock cycles that have been counted.

15.5.6 OCCS External Clock Check Target (OCCS_CLKCHKT)

CLKCHKR and CLKCHKT are used in applications where it is required to verify the activity of an external clock source or crystal/resonator oscillator before it is selected as the active system clock source. PRECS must be 0 to use this function.

Address: OCCS_CLKCHKT – F120h base + 6h offset = F126h

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

OCCS_CLKCHKT field descriptions

Field	Description
15–9 Reserved	This read-only bitfield is reserved and always has the value zero.
8–0 TARGET_ COUNT	CLKCHKT Target Count bits Number of external clock cycles that have been counted. **NOTE:** This counter value is not synchronized to the bus clock and any value read while CHK_ENA is high should not be considered accurate.

15.5.7 OCCS Protection Register (OCCS_PROT)

This register provides features for runaway code protection of safety-critical register fields. By choosing an appropriate subset of protection registers the end user can define the trade-off between power management and protection of the OCCS operating configuration.

Flexibility is provided so that write protection control values may themselves be optionally locked (write protected). To this end, protection controls in this register have two bit values. The right bit determines the setting of the control, and the left bit determines whether the value is locked. When a protection control is set to a locked

value, it can only be altered by a chip reset which restores its default non-locked value. While a protection control remains set to non-locked values, it can be re-written to any new value.

Address: OCCS_PROT – F120h base + 7h offset = F127h

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

OCCS_PROT field descriptions

Field	Description
15–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5–4 FRQEP	Frequency Enable Protection Enables write protection of the COD and ZSRC registers. 00 Write protection off (default) 01 Write protection on 10 Write protection off and locked until chip reset 11 Write protection on and locked until chip reset
3–2 OSCEP	Oscillator Enable Protection Enables write protection of the OSCTL and PRECS registers. 00 Write protection off (default) 01 Write protection on 10 Write protection off and locked until chip reset 11 Write protection on and locked until chip reset
1–0 PLLEP	PLL Enable Protection Enables write protection of the PLLPDN, LOCIE, LORTP registers. By write protecting these registers (PLLPD=0, LOCIE=1) the Loss of Reference detector can not be disabled. 00 Write protection off (default) 01 Write protection on 10 Write protection off and locked until chip reset 11 Write protection on and locked until chip reset

15.6 Functional Description

Possible clock source choices are:

- Internal relaxation oscillator

- External ceramic resonator
- External crystal
- External clock source on either XTAL or GPIO port C0.

Each of these clock sources can be selected to drive the remainder of the clock generation circuitry. This circuitry allows direct use of the clock, or the clock can be used as an input to the PLL which will generate a higher frequency clock for use within the chip.

The clock multiplexer (ZSRC MUX) selects the direct clock on power up. A different clock source can be selected by writing to the PLL control register (CTRL). Once a new clock source is selected, the new clock will be activated within four clock periods of the new clock after the clock selection request is re-clocked by the current IPbus clock.

The postscaler output is guaranteed to be glitch free when changing the divide ratio.

Transitions to/from direct to postscaler frequencies are guaranteed to be glitch free on the sys_clk_x2 clock. Before switching to the PLL, the PLL must be locked. The OCCS Status Register (STAT) shows the status of the DSP core clock source. Because the synchronizing circuit changes modes to avoid any glitches, the STAT ZCLOCK source (ZSRCS) shows overlapping modes as an intermediate step. After PLL lock is detected, the DSP core clock can be switched to the PLL by writing to the ZSRC field in the CTRL register.

Frequencies going out of the OCCS are controlled by the postscaler and/or by the divide-by ratio within the PLL. For proper operation of the PLL, the user must keep the VCO, within the PLL, in its operational range of 120 MHz to 240 MHz. The output of the VCO is depicted as F_{pll} in the block diagram. The input frequency multiplied by the divide-by ratio is the frequency at which the VCO runs.

The PLL lock time is 10 ms or less when coming from a powered down state to a power up state. It is recommended when powering down, or powering up, to unselect the PLL as the clocking source. Only after lock is achieved should the PLL be used as a valid clocking source.

The following table shows the possible clock sources and configurations.

Table 15-9. Clock Choices Without Crystal Oscillator

Clock Source	Clock Selected	Configuration Steps
Relaxation Oscillator	Direct	Default. 1. Change TRIM as needed to obtain the desired clock rate

Table continues on the next page...

Table 15-9. Clock Choices Without Crystal Oscillator (continued)

Clock Source	Clock Selected	Configuration Steps
Relaxation Oscillator	Postscaler	1. Change TRIM as needed to obtain the desired clock rate 2. Change COD, if desired 3. Enable the PLL (PLLPD=0) 4. Wait for PLL lock (LCK1=1 and LCK0=1) 5. Change ZSRC to select the postscaler clock (ZSRC=10).
External Clock Source	Direct	1. The clock source (CLKIN) should be enabled in the GPIO and SIM as necessary.(Chip dependent) 2. Select CLKIN as the source clock (PRECS=1, EXT_SEL=1). 3. Wait 6 NOP's for the synchronizing circuit to change clocks. 4. The relaxation oscillator can be powered down (ROPD=1) to conserve power.
External Clock Source	Postscaler	1. The clock source (CLKIN) should be enabled in the GPIO and SIM as necessary.(Chip dependent) 2. Select CLKIN as the source clock (PRECS=1, EXT_SEL=1). 3. Wait 6 NOP's for the synchronizing circuit to change clocks. 4. The relaxation oscillator can be powered down (ROPD=1) to conserve power 5. Change COD, if desired 6. Enable the PLL (PLLPD=0) 7. Wait for PLL lock (LCK1=1 and LCK0=1) 8. Change ZSRC to select the postscaler clock (ZSRC=10).

Table 15-10. Clock Choices with Crystal Oscillator

Clock Source	Clock Selected	Configuration Steps
Relaxation Oscillator	Direct	Default. 1. The crystal oscillator should be powered down (CLK_MODE=1) to conserve power. Default state 2. Change TRIM as needed to obtain the desired clock rate
Relaxation Oscillator	Postscaler	1. The crystal oscillator should be powered down (CLK_MODE=1) to conserve power. 2. Change TRIM as needed to obtain the desired clock rate 3. Change COD, if desired 4. Enable the PLL (PLLPD=0) 5. Wait for PLL lock (LCK1=1 and LCK0=1) 6. Change ZSRC to select the postscaler clock (ZSRC=10).

Table continues on the next page...

Table 15-10. Clock Choices with Crystal Oscillator (continued)

Clock Source	Clock Selected	Configuration Steps
Ceramic Resonator	Direct	1. The crystal oscillator should be powered up (CLK_MODE=0). 2. Change the oscillator to high power mode (COHL=0). 3. Wait for the oscillator to stabilize (up to 10ms) 4. The clock source should be changed to the crystal oscillator (EXT_SEL=0, PRECS=1, in that order) 5. Wait x NOP's for the synchronizing circuit to change clocks. 6. The relaxation oscillator can be powered down (ROPD=1) to conserve power
Ceramic Resonator	Postscaler	1. The crystal oscillator should be powered up (CLK_MODE=0). 2. Change the oscillator to high power mode (COHL=0). 3. Wait for the crystal oscillator to stabilize (up to 10ms) 4. The clock source should be changed to the crystal oscillator (EXT_SEL=0, PRECS=1, in that order) 5. Wait 6 NOP's for the synchronizing circuit to change clocks. 6. The relaxation oscillator can be powered down (ROPD=1) to conserve power 7. Change COD, if desired 8. Enable the PLL (PLLPD=0) 9. Wait for PLL lock (LCK1=1 and LCK0=1) 10. Change ZSRC to select the postscaler clock (ZSRC=10).
Crystal	Direct	1. The crystal oscillator should be powered up (CLK_MODE=0). 2. Change the oscillator to low power mode (COHL=1) 3. Wait for the crystal oscillator to stabilize (up to 10ms) 4. The clock source should be changed to the crystal oscillator (EXT_SEL=0, PRECS=1, in that order) 5. Wait x NOP's for the synchronizing circuit to change clocks. 6. The relaxation oscillator can be powered down (ROPD=1) to conserve power

Table continues on the next page...

Table 15-10. Clock Choices with Crystal Oscillator (continued)

Clock Source	Clock Selected	Configuration Steps
Crystal	Postscaler	1. The crystal oscillator should be powered up (CLK_MODE=0). 2. Change the oscillator to low power mode (COHL=1) 3. Wait for the crystal oscillator to stabilize (up to 10ms) 4. The clock source should be changed to the crystal oscillator (EXT_SEL=0, PRECS=1, in that order) 5. Wait 6 NOP's for the synchronizing circuit to change clocks. 6. The relaxation oscillator can be powered down (ROPD=1) to conserve power 7. Change COD, if desired 8. Enable the PLL (PLLPD=0) 9. Wait for PLL lock (LCK1=1 and LCK0=1) 10. Change ZSRC to select the postscaler clock (ZSRC=10).
External Clock Source	Direct	1. The clock source (CLKIN) should be enabled in the GPIO and SIM as necessary.(Chip dependent) 2. Set the CLK_MODE bit in OSCTL register to 1. 3. Select CLKIN as the source clock (PRECS=1). 4. The clock source should be changed to the crystal oscillator (EXT_SEL=1, PRECS=1, in that order). 5. Wait 6 NOP's for the synchronizing circuit to change clocks. 6. The relaxation oscillator can be powered down (ROPD=1) to conserve power. 7. At this point the EXTAL pin can be used as a GPIO by deasserting the appropriate PE bit in the GPIO_X_PER.
External Clock Source	Postscaler	1. The clock source (CLKIN) should be enabled in the GPIO and SIM as necessary.(Chip dependent) 2. Set the CLK_MODE bit in OSCTL register to 1. 3. Select CLKIN as the source clock (PRECS=1). 4. The clock source should be changed to the crystal oscillator (EXT_SEL=1, PRECS=1, in that order) 5. Wait x NOP's for the synchronizing circuit to change clocks. 6. The relaxation oscillator can be powered down (ROPD=1) to conserve power 7. Change COD, if desired 8. Enable the PLL (PLLPD=0) 9. Wait for PLL lock (LCK1=1 and LCK0=1) 10. Change ZSRC to select the postscaler clock (ZSRC=10). 11. At this point the EXTAL pin can be used as a GPIO by deasserting the appropriate PE bit in the GPIO_X_PER.

15.7 External Reference

If higher clock precision is required the chip can be operated from an external clock source.

15.8 Crystal Oscillator

The crystal oscillator is designed to operate with either an external crystal or an external ceramic oscillator. If a crystal above 8MHz is used, the COHL bit must be set to 0. If an 8MHz or lower crystal/resonator is used on the board the power level of this oscillator can be lowered to reduce power consumption (see the COHL bit in the OSCTL register).

15.8.1 Switching Clock Sources

To robustly switch between the Internal Relaxation Oscillator clock, External oscillator Clock and CLKIN, the changeover switch assumes the clocks are completely asynchronous, so a synchronizing circuit is required to make the transition. When the select input (PRECS) is changed, the switch will continue to operate off the original clock for between 1 and 2 cycles as the select input is transitioned through one side of the synchronizer. Next, the output will be held low for between 1 and 2 cycles of the new clock as the select input transitions through the other side. Then the output starts switching at the new clock's frequency. This transition guarantees that no glitches will be seen on the output even though the select input may change asynchronously to the clocks. The unpredictability of the transition period is a necessary result of the asynchronicity. The switch automatically selects Internal Relaxation Osc. clock during Reset.

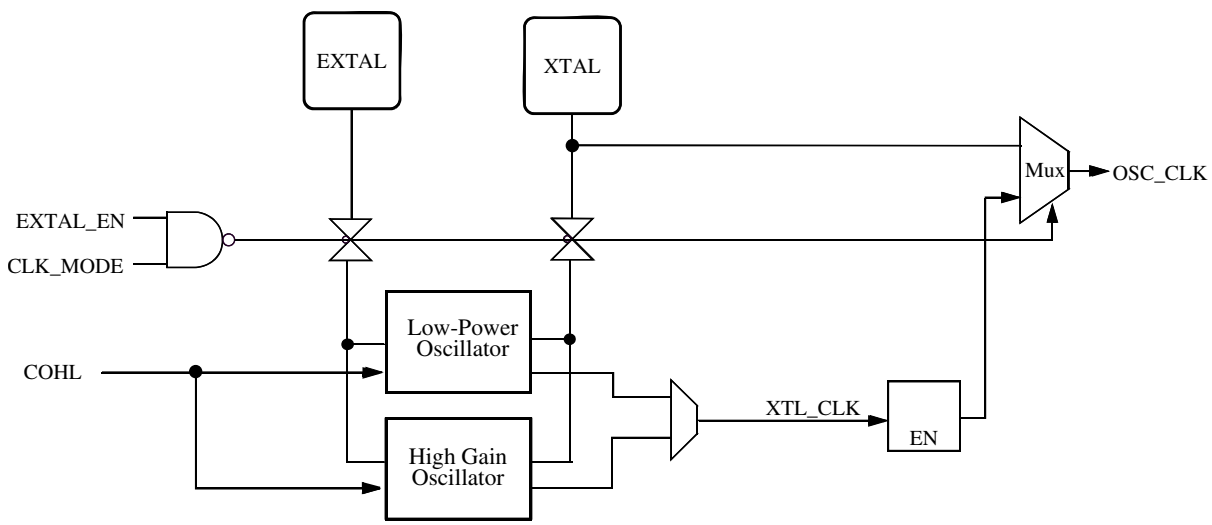
Switching from the Internal Relaxation Oscillator Clock to the Crystal Oscillator Clock source or vice-versa requires both clock sources to be enabled and stable. A simple flow requires:

- If switching to the crystal oscillator, make sure that it has been enabled via GPIO and is powered up (CLK_MODE=0). Check the value of the COSC_RDY bit.
- If switching to the Relaxation Oscillator, make sure that it is powered up i.e. ROPD is clear.
- Wait for the clock to become Active.
- Switch clocks

- Execute 4 NOPs instructions
- Disable previous clock source (i.e. power down Relaxation Osc if Crystal is selected).

The key point to remember in this flow is that the clock source should not be switched unless the desired clock is on and stable.

When a new DSP core clock is selected, the clock generation module will synchronize the request and select the new clock. The OCCS Status Register (STAT) shows the status of the DSP core clock source. Since the synchronizing circuit changes clock sources as to avoid any glitches, the **ZSRCS** bits in STAT will show overlapping modes as an intermediate step.



15.9 Phase Locked Loop

15.9.1 PLL Recommended Range of Operation

The voltage controlled oscillator (VCO) within the PLL has a characterized operating range extending from 120MHz to 240MHz. The output of the PLL, F_{pll}, is fed to the input of the postscaler.

15.9.2 PLL Frequency Lock Detector Block

This digital block monitors the VCO output clock and sets the LCK[1:0] bits in the OCCS Status Register (STAT) based on its frequency accuracy. The lock detector is enabled with the LCKON bit of the PLL control register (CTRL) and the CTRL[PLLPD] bit. Once enabled, the detector starts two counters whose outputs are periodically compared. The input clocks to these counters are the VCO output clock divided by the value in DIVBY[PLLDB], called FEEDBACK, and the PLL input clock. The period of the pulses being compared cover one whole period of each clock.

FEEDBACK and MSTR_OSC clocks are compared after 16, 32, and 64 cycles. If, after 32 cycles, the clocks match, the LCK0 bit is set to one. If, after 64 cycles of MSTR_OSC, there is the same number of MSTR_OSC clocks as FEEDBACK clocks, the LCK1 bit is also set. The LCK bits remain set until any of these events occur:

- Clocks fail to match
- Reset caused by LCKON, PLLPD
- Chip-level reset

When the circuit sets the LCK1, the two counters are reset and start the count again. The lock detector is designed so if LCK1 is reset to zero because clocks did not match, LCK0 can stay high. This provides the processor the accuracy of the two clocks with respect to each other.

15.9.3 Loss of Reference Clock Detector

The loss of reference clock detector is designed to generate an interrupt when the reference clock to the PLL is interrupted. An LOR interrupt should occur after a minimum time of (LORTP+1) × 10 x (reference clock period) / ((PLLDB+1) / 2). This is the minimum time because the PLL output frequency starts to decrease as the PLL tries to track the input reference source. In fact, the phase locked loop can continue running for a time after its reference clock has been disturbed. This provides time for detection of the problem and an orderly system shutdown.

15.9.4 Resets

The following table illustrates the various reset sources present in the implementation of the OCCS module. Any configuration which yields substantially the same operation is acceptable.

Table 15-11. V1 Implementation Reset Summary

Reset	Source	Characteristics
perip_rst_b	SIM	The chip architecture incorporates a 3-phase reset system. Raw resets are comprised of POR, external reset, COP and software reset. These are all active low and ANDed and stretched (see SIM chapter for details) to create the peripheral reset signal (perip_rst_b). That signal is used by this module to reset its state. perip_rst_b is stretched an additional 32 system clocks to generate core_rst_b, which is used to reset the core.

15.9.5 Clocks

The following table summarizes the various clock signals used by the OCCS module. All clocks are ultimately derived from the oscillator output.

Table 15-12. Clock Summary

Clock	Source	Characteristics
IP Bus Clock	IP Bus Bridge	derived from sys_clk and has the same frequency Used for all peripheral register reads & writes
sys_clk_x2	this module	Primary source for all on-chip clocks excluding the oscillator clock used by the MSCAN module. This signal is divided by two by the SIM to generate the master system frequency.
FEEDBACK	PLL	FEEDBACK pin of the PLL
F_{rosc}	Relaxation oscillator	Nominally this is 8MHz. 400KHz in standby.
F_{cosc}	crystal oscillator	Nominally this is 8MHz.
F_{pll}	this module	output of the PLL

Chapter 16
System Integration Module (SIM)

16.1 Introduction

This specification describes the operation and functionality of the system integration module for this devive.

16.1.1 Overview

The SIM is a system catchall for the glue logic that ties together the system-on-chip. It controls distribution of resets and clocks and provides a number of control features.

The system integration module is responsible for the following functions:

- Reset sequencing
- Clock generation and distribution
- Implementation of stop and wait low power modes
- System status registers
- Registers for software access to the JTAG ID of the chip
- Short addressing controls
- External and internal peripheral signal muxing control

These are discussed in more detail in the sections that follow.

16.1.2 Features

The SIM has the following features:

- System bus clocks with pipeline holdoff support
  - Data RAM clock with holdoff control
  - Program flash clock with holdoff support
  - IP bus interface clock with holdoff control
  - HFM IP bus interface clock and PCLK
  - DSC core system clock
  - General-purpose system clock (both standard and inverted versions)
- System clocks for non-pipelined interfaces
  - PCLK clock for DSC core
  - NCLK clock for DSC core
  - A continuously running system clock
  - A continuously running system clock with enable for HFM
- Peripheral clocks including high speed option for TMRs, and SCIs
- ITCK clock to the DSC core TAP interface
- Power saving clock gating for peripherals.
- Three power modes (run, wait, stop) to control power utilization
  - Stop mode shuts down DSC core, system clock, and peripheral clock.
  - Wait mode shuts down DSC core, and unnecessary system clock operation.
  - Run mode supports full part operation
- Controls with write protection to enable/disable the DSC core WAIT and STOP instructions.
- Controls to permit selected peripherals to run in stop mode to generate stop recovery interrupts.
- Controls for programmable peripheral and GPIO connections
- Manages internal reset deassertion sequence
- Software initiated reset

- A holdoff output to abort peripheral bus transactions when the system bus pipeline is held off
- Short addressing base location control
- Peripheral protection control to provide runaway code protection for safety critical applications
- Registers containing the JTAG ID of the chip

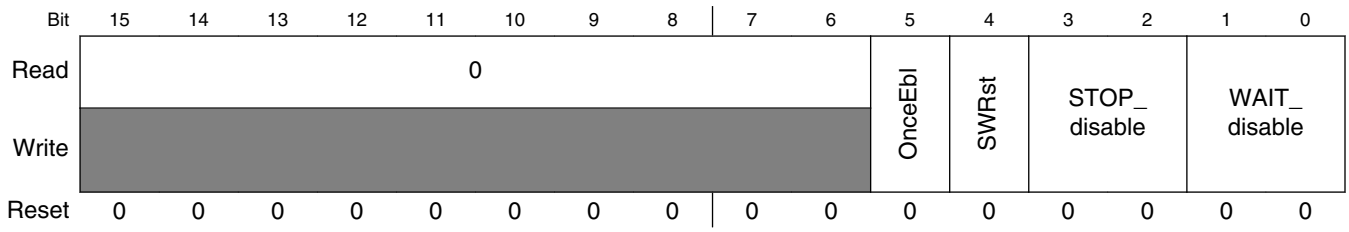
16.2 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SIM_CTRL	R	0										OnceEbl	SWRst	STOP_ disable		WAIT_ disable	
		W																
1	SIM_RSTAT	R	0									SWR	COP _ CPU	COP _ LOR	EXT R	POR	0	
		W																
2	SIM_SCR0	R	SCR0															
		W																
3	SIM_SCR1	R	SCR1															
		W																
4	SIM_SCR2	R	SCR2															
		W																
5	SIM_SCR3	R	SCR3															
		W																
6	SIM_MSHID	R	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1
		W																
7	SIM_LSHID	R	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1
		W																
8	SIM_PWR	R	0														LRSTDBY	
		W																
A	SIM_CLKOUT	R	0									Reserved	CLKDIS	0			CLKOSEL	
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	SIM_PCR	R	TMRA_CR	TMRB_CR	SCI0_CR	SCI1_CR	0											
		W																
C	SIM_PCE0	R	TA0	TA1	TA2	TA3	TB0	TB1	TB2	TB3	ADC	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
		W																
D	SIM_PCE1	R	0	DAC	CMPA	CMPB	CMPC	SCI0	SCI1	QSPI0	IIC0	IIC1	CRC	REFA	REFB	REFC	HFM	MSCAN
		W																
E	SIM_PCE2	R	0												PWMCH0	PWMCH1	PWMCH2	PWMCH3
		W																
F	SIM_SD0	R	TA0	TA1	TA2	TA3	TB0	TB1	TB2	TB3	ADC	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
		W																
10	SIM_SD1	R	0	DAC	CMPA	CMPB	CMPC	SCI0	SCI1	QSPI0	IIC0	IIC1	CRC	REFA	REFB	REFC	0	MSCAN
		W																
11	SIM_SD2	R	0												PWMCH0	PWMCH1	PWMCH2	PWMCH3
		W																
12	SIM_IOSAHI	R	0														ISAL[23:22]	
		W																
13	SIM_IOSALO	R	ISAL[21:6]															
		W																
14	SIM_PROT	R	0												PCEP		GIPSP	
		W																
15	SIM_GPS0	R	0	C7		C6	0	C5	0	C4	C3		C2		0			C0
		W																
16	SIM_GPS1	R	0	C15	0	C14	0	C13	C12		C11		C10		0	C9	0	C8
		W																
17	SIM_GPS2	R	0			F6	0	F5	0	F4	0	F3	0	F2	0	F1	0	
		W																
18	SIM_GPS3	R	TMRB3	TMRB2	TMRB1	TMRB0	0	E7	0	E6	0	E5	0	E4	0	F8	0	A0
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

16.2.1 Control Register (SIM_CTRL)

Address: SIM_CTRL – F0E0h base + 0h offset = F0E0h



SIM_CTRL field descriptions

Field	Description
15–6 Reserved	This read-only bitfield is reserved and always has the value zero.
5 OnceEbl	OnCE Enable 0 The OnCE clock to the DSC core is enabled when the core TAP is enabled. 1 The OnCE clock to the DSC core is always enabled.
4 SWRst	SOFTWARE RESET Writing a 1 to this field causes a device reset.
3–2 STOP_disable	STOP Disable Partial Powerdown mode cannot be entered if the STOP instruction is disabled. 00 Stop mode is entered when the DSC core executes a STOP instruction. 01 The STOP instruction does not cause entry into stop mode. 10 Stop mode is entered when the DSC core executes a STOP instruction, and the STOP_disable field is write protected until the next reset. 11 The STOP instruction does not cause entry into stop mode, and the STOP_disable field is write protected until the next reset.
1–0 WAIT_disable	WAIT Disable 00 WAIT mode is entered when the DSC core executes a WAIT instruction. 01 The WAIT instruction does not cause entry into WAIT. mode 10 WAIT mode is entered when the DSC core executes a WAIT instruction, and the WAIT_disable field is write protected until the next reset. 11 The WAIT instruction does not cause entry into WAIT mode and the WAIT_disable field is write protected until the next reset.

16.2.2 Reset Status Register (SIM_RSTAT)

SIM_RSTAT is updated during any system reset and indicates the cause of the most recent reset. It also controls whether the COP reset vector or regular reset vector in the vector table is used. This register is asynchronously reset during power-on reset and subsequently is synchronously updated based on the level of the external reset, software reset, or COP reset inputs. It is one-hot encoded, and only one source is ever indicated. If multiple reset sources assert simultaneously, the highest-priority source is indicated. The priority from highest to lowest is POR, EXTR, COP_LOR, COP_CPU, and SWR. POR is always set during a power-on reset, but POR is cleared and EXTR is set if the external reset pin is asserted or remains asserted after the power-on reset has deasserted.

Address: SIM_RSTAT – F0E0h base + 1h offset = F0E1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0									SWR	COP_CPU	COP_LOR	EXTR	POR	0	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

SIM_RSTAT field descriptions

Field	Description
15–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6 SWR	Software Reset When set, this bit indicates that the previous system reset occurred as a result of a software reset (write 1 to the SIM_CTRL[SW Rst] bit). SWR is not set if a COP, external, or POR also occurs.
5 COP_CPU	COP CPU Time-out Reset When set, this bit indicates that the previous system reset was caused when the computer operating properly (COP) module signaled a CPU time-out reset. COP_CPU is not set if an external reset, POR, or COP loss of reference reset also occurs. If COP_CPU is set as code starts executing, the COP reset vector in the vector table is used. Otherwise, the normal reset vector is used.
4 COP_LOR	COP Loss of Reference Reset When set, this bit indicates that the previous system reset was caused when the computer operating properly (COP) module signaled a loss of reference clock reset. COP_LOR is not set if an external or POR also occurred. If COP_LOR is set as code starts executing, the COP reset vector in the vector table is used. Otherwise, the normal reset vector is used.
3 EXTR	External Reset When set, this bit indicates that the previous system reset was caused by an external reset. EXTR is set only if the external reset pin is asserted or remains asserted after the power-on reset deasserts.

Table continues on the next page...

SIM_RSTAT field descriptions (continued)

Field	Description
2 POR	Power-on Reset This bit is set at power-on reset.
1–0 Reserved	This read-only bitfield is reserved and always has the value zero.

16.2.3 Software Control Register (SIM_SCR0)

Address: SIM_SCR0 – F0E0h base + 2h offset = F0E2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR0 field descriptions

Field	Description
15–0 SCR0	Software Control Data SCR0 is for general-purpose use by software. It is reset only by a power-on reset.

16.2.4 Software Control Register (SIM_SCR1)

Address: SIM_SCR1 – F0E0h base + 3h offset = F0E3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR1 field descriptions

Field	Description
15–0 SCR1	Software Control Data SCR1 is for general-purpose use by software. It is reset only by a power-on reset.

16.2.5 Software Control Register (SIM_SCR2)

Address: SIM_SCR2 – F0E0h base + 4h offset = F0E4h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR2 field descriptions

Field	Description
15–0 SCR2	Software Control Data SCR2 is for general-purpose use by software. It is reset only by a power-on reset.

16.2.6 Software Control Register (SIM_SCR3)

Address: SIM_SCR3 – F0E0h base + 5h offset = F0E5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR3 field descriptions

Field	Description
15–0 SCR3	Software Control Data SCR3 is for general-purpose use by software. It is reset only by a power-on reset.

16.2.7 Most Significant Half of JTAG ID (SIM_MSHID)

SIM_MSHID is a read-only register that returns the most significant half of the JTAG ID for the device.

Address: SIM_MSHID – F0E0h base + 6h offset = F0E6h

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

SIM_MSHID field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 Reserved	This read-only bit is reserved and always has the value zero.
13 Reserved	This read-only bit is reserved and always has the value zero.
12 Reserved	This read-only bit is reserved and always has the value zero.
11 Reserved	This read-only bit is reserved and always has the value zero.
10 Reserved	This read-only bit is reserved and always has the value zero.
9 Reserved	This read-only bit is reserved and always has the value zero.
8 Reserved	This read-only bit is reserved and always has the value one.
7 Reserved	This read-only bit is reserved and always has the value one.
6 Reserved	This read-only bit is reserved and always has the value one.
5 Reserved	This read-only bit is reserved and always has the value zero.
4 Reserved	This read-only bit is reserved and always has the value zero.
3 Reserved	This read-only bit is reserved and always has the value zero.
2 Reserved	This read-only bit is reserved and always has the value zero.
1 Reserved	This read-only bit is reserved and always has the value one.
0 Reserved	This read-only bit is reserved and always has the value one.

16.2.8 Least Significant Half of JTAG ID (SIM_LSHID)

SIM_LSHID is a read-only register that returns the least significant half of the JTAG ID for the device.

Address: SIM_LSHID – F0E0h base + 7h offset = F0E7h

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

SIM_LSHID field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 Reserved	This read-only bit is reserved and always has the value zero.
13 Reserved	This read-only bit is reserved and always has the value one.
12 Reserved	This read-only bit is reserved and always has the value zero.
11 Reserved	This read-only bit is reserved and always has the value zero.
10 Reserved	This read-only bit is reserved and always has the value zero.
9 Reserved	This read-only bit is reserved and always has the value zero.
8 Reserved	This read-only bit is reserved and always has the value zero.
7 Reserved	This read-only bit is reserved and always has the value zero.
6 Reserved	This read-only bit is reserved and always has the value zero.
5 Reserved	This read-only bit is reserved and always has the value zero.
4 Reserved	This read-only bit is reserved and always has the value one.
3 Reserved	This read-only bit is reserved and always has the value one.
2 Reserved	This read-only bit is reserved and always has the value one.

Table continues on the next page...

SIM_LSHID field descriptions (continued)

Field	Description
1 Reserved	This read-only bit is reserved and always has the value zero.
0 Reserved	This read-only bit is reserved and always has the value one.

16.2.9 Power Control Register (SIM_PWR)

SIM_PWR contains a single control field to enable/disable the standby mode of the primary voltage regulator.

Address: SIM_PWR – F0E0h base + 8h offset = F0E8h

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

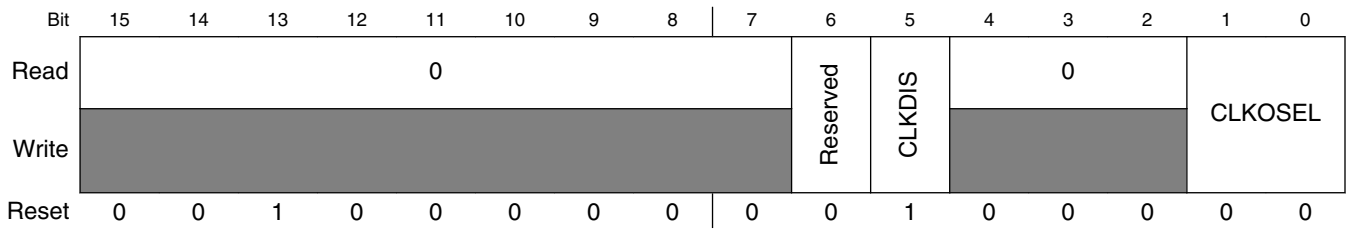
SIM_PWR field descriptions

Field	Description
15–2 Reserved	This read-only bitfield is reserved and always has the value zero.
1–0 LRSTDBY	Large Regulator Standby Control- Controls the standby mode of the primary on-device voltage regulator. Standby mode reduces overall device power consumption but places significant constraints on the allowed operating frequency. Refer to the Power Supervisor module for details on standby mode. The field value can optionally be write protected. 00 Large regulator placed in normal mode (default). 01 Large regulator placed in standby mode. 10 Large regulator placed in normal mode and LRSTDBY is write protected until device reset. 11 Large regulator placed in standby mode and LRSTDBY is write protected until device reset.

16.2.10 Clock Output Select Register (SIM_CLKOUT)

SIM_CLKOUT can be used to multiplex selected clocks generated inside the clock generation, SIM, and other internal modules onto the SIM CLKOUT clock output signal. This signal, in turn, is typically brought out to an external pad. Glitches may be produced when the clock is enabled or switched. The delay from the clock source to the output is unspecified. The visibility of the waveform on the CLKOUT on an external pad is subject to the frequency limitations of the associated I/O cell.

Address: SIM_CLKOUT – F0E0h base + Ah offset = F0EAh



SIM_CLKOUT field descriptions

Field	Description
15–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6 Reserved	This bit is reserved. Always write a 0 to this bit for normal operation.
5 CLKDIS	Disable for CLKOUT 0 CLKOUT output is enabled and will output the signal indicated by CLKOSEL 1 CLKOUT is 0
4–2 Reserved	This read-only bitfield is reserved and always has the value zero. Always write a 0 to this field for normal operation.
1–0 CLKOSEL	CLKOUT Select Selects the clock to be multiplexed on the CLKOUT pin as defined in the following table. Internal delay to CLKOUT output is unspecified. The signal at the output pad is undefined when the CLKOUT signal frequency exceeds the rated frequency of the I/O cell. CLKOUT may glitch when CLKDIS and CLKOSEL settings are changed. CLKDIS and CLKOSEL define CLKOUT. Propagation of CLKOUT to an external pad requires proper setting of the related GPSn and GPIO_X_PER fields.

Table 16-12. CLKOUT Selection Using CLKOSEL

CLKOSEL	USER	FUNCTION	SIGNAL NAME	NOTES
x00	USER	Continuous System Clock	iclk_sys_cont	System frequency, continuous after POR
x01	USER	Peripheral Clock	iclk_per_cont	Peripheral frequency, continuous when not in reset
x02	USER	High-speed Peripheral Clock	iclk_per_cont_2x	2x system frequency continuous when not in reset
x03	USER	Master Clock	mstr_osc	Master clock source before PLL (ROSC, OSC or external clock) continuous

16.2.11 Peripheral Clock Rate Register (SIM_PCR)

By default, all peripherals are clocked at the system clock rate, which is a maximum of 60 MHz. Selected peripherals clocks can be clocked at two times this normal rate, which is a maximum of 120 MHz. SIM_PCR is used to enable high-speed clocking for peripherals that support it. High-speed peripheral clocking is dependent on the mstr_2x clock output of the OCCS.

Peripherals should not be left in an enabled or operating mode while reconfiguring their clocks using the controls in SIM or OCCS. PCR bits should therefore be changed only while the respective peripheral is disabled. Refer to the peripheral user guide for details.

When a peripheral operates in high-speed mode, the I/O rate to the peripheral remains limited to the system clock rate since that is the rate at which the processor operates. For peripherals with a single clock input, that clock operates at the high-speed rate and a high-speed I/O gasket is used to coordinate I/O with the processor. For peripherals with separate I/O and “run” clocks, the I/O clock operates at the normal peripheral clock rate and only the “run” clock operates at the 2x high-speed rate.

Address: SIM_PCR – F0E0h base + Bh offset = F0EBh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TMRA_CR	TMRB_CR	SCI0_CR	SCI1_CR	0											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PCR field descriptions

Field	Description
15 TMRA_CR	General-Purpose Timer A Clock Rate Selects the clock speed for the General-Purpose Timer A module. Timer B should be disabled before TMRA_CR is altered. See the peripheral user guide for details. 0 Timer clock rate equals the core clock rate, a maximum of 60 MHz. (default) 1 Timer clock rate equals the 2X core clock rate. rate
14 TMRB_CR	Selects the clock speed for the General-Purpose Timer B module. Timer B should be disabled before TMRB_CR is altered. See the peripheral user guide for details.
13 SCI0_CR	SCI 0 Clock Rate Selects the clock speed for the SCI0 module. SCI0 should be disabled before SCI0_CR is altered. See the peripheral user guide for details.

Table continues on the next page...

SIM_PCR field descriptions (continued)

Field	Description
	0 SCI0 clock rate equals core clock rate, maximum 60MHz (default) 1 SCI0 clock rate equals 2X core clock rate
12 SCI1_CR	SCI1 Clock Rate Selects the clock speed for the SCI1 module. SCI1 should be disabled before SCI1_CR is altered. See the peripheral user guide for details. 0 SCI1 clock rate equals core clock rate, maximum 60MHz (default) 1 The SCI1 clock rate equals the 2X core clock. rate
11–0 Reserved	This read-only bitfield is reserved and always has the value zero.

16.2.12 Peripheral Clock Enable Register 0 (SIM_PCE0)

Address: SIM_PCE0 – F0E0h base + Ch offset = F0ECh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TA0	TA1	TA2	TA3	TB0	TB1	TB2	TB3	ADC	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PCE0 field descriptions

Field	Description
15 TA0	TMRA0 IPBus Clock Enable
14 TA1	TMRA1 IPBus Clock Enable
13 TA2	TMRA2 IPBus Clock Enable
12 TA3	TMRA3 IPBus Clock Enable
11 TB0	TMRB0 IPBus Clock Enable
10 TB1	TMRB1 IPBus Clock Enable
9 TB2	TMRB2 IPBus Clock Enable
8 TB3	TMRB3 IPBus Clock Enable
7 ADC	ADC IPBus Clock Enable

Table continues on the next page...

SIM_PCE0 field descriptions (continued)

Field	Description
6 GPIOA	GPIOA IPBus Clock Enable
5 GPIOB	GPIOB IPBus Clock Enable
4 GPIOC	GPIOC IPBus Clock Enable
3 GPIOD	GPIOD IPBus Clock Enable
2 GPIOE	GPIOE IPBus Clock Enable
1 GPIOF	GPIOF—GPIOF IPBus Clock Enable
0 Reserved	This read-only bit is reserved and always has the value zero.

16.2.13 Peripheral Clock Enable Register 1 (SIM_PCE1)

Address: SIM_PCE1 – F0E0h base + Dh offset = F0EDh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	DAC	CMPA	CMPB	CMPC	SCI0	SCI1	QSPI 0	IIC0	IIC1	CRC	REFA	REFB	REFC	HFM	MSCAN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

SIM_PCE1 field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 DAC	DAC IPBus Clock Enable
13 CMPA	CMPA IPBus Clock Enable
12 CMPB	CMPB IPBus Clock Enable
11 CMPC	CMPC IPBus Clock Enable
10 SCI0	SCI0 IPBus Clock Enable
9 SCI1	SCI1 IPBus Clock Enable

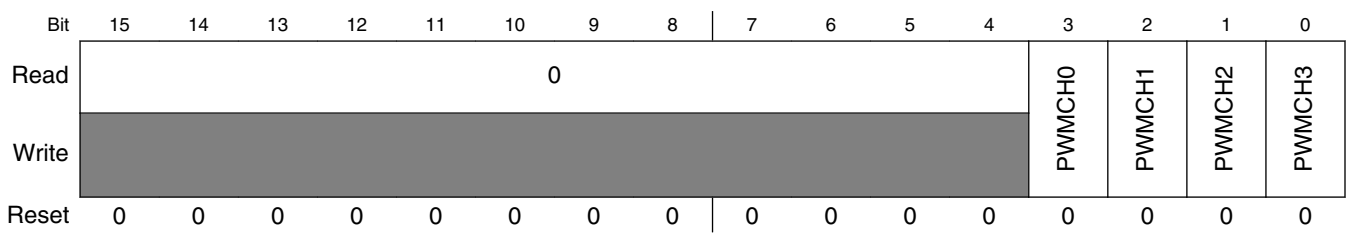
Table continues on the next page...

SIM_PCE1 field descriptions (continued)

Field	Description
8 QSPI0	QSPI0 IPBus Clock Enable
7 IIC0	IIC0 IPBus Clock Enable
6 IIC1	IIC1 IPBus Clock Enable
5 CRC	CRC IPBus Clock Enable
4 REFA	REFA IPBus Clock Enable
3 REFB	REFB IPBus Clock Enable
2 REFC	REFC IPBus Clock Enable
1 HFM	HFM IPBus Clock Enable The PCE bit for HFM is reset to 1 in order to support speical flash functions such as flash lockout recovery which must operate when coming out of reset. This bit may be subsequently set to 0 to save power by disabling the HFM peripheral clock during periods in which no programming state machine operations are in process and the flash is being used for instruction fetch only.
0 MSCAN	MSCAN IPBus Clock Enable

16.2.14 Peripheral Clock Enable Register 2 (SIM_PCE2)

Address: SIM_PCE2 – F0E0h base + Eh offset = F0EEh



SIM_PCE2 field descriptions

Field	Description
15–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3 PWMCH0	PWM Channel 0 IPBus Clock Enable

Table continues on the next page...

SIM_PCE2 field descriptions (continued)

Field	Description
2 PWMCH1	PWM Channel 1 IPBus Clock Enable
1 PWMCH2	PWM Channel 2 IPBus Clock Enable
0 PWMCH3	PWM Channel 3 IPBus Clock Enable Each bit enables peripheral clocking to the indicated peripheral. 0 The corresponding peripheral is not clocked 1 The corresponding peripheral is clocked

16.2.15 STOP Disable Register 0 (SIM_SD0)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral user guide for details. IP bus reads and writes cannot be made to a module with its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is cleared to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: SIM_SD0 – F0E0h base + Fh offset = F0EFh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TA0	TA1	TA2	TA3	TB0	TB1	TB2	TB3	ADC	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SD0 field descriptions

Field	Description
15 TA0	TMRA1 IPBus Clock STOP Disable
14 TA1	TMRA1 IPBus Clock STOP Disable
13 TA2	TMRA2 IPBus Clock STOP Disable
12 TA3	TMRA3 IPBus Clock STOP Disable
11 TB0	TMRB0 IPBus Clock STOP Disable
10 TB1	TMRB1 IPBus Clock STOP Disable
9 TB2	TMRB2 IPBus Clock STOP Disable
8 TB3	TMRB3 IPBus Clock STOP Disable
7 ADC	ADC IPBus Clock STOP Disable
6 GPIOA	GPIOA IPBus Clock STOP Disable
5 GPIOB	GPIOB IPBus Clock STOP Disable
4 GPIOC	GPIOC IPBus Clock STOP Disable
3 GPIOD	GPIOD IPBus Clock STOP Disable
2 GPIOE	GPIOE IPBus Clock STOP Disable
1 GPIOF	GPIOF IPBus Clock STOP Disable
0 Reserved	This read-only bit is reserved and always has the value zero.

16.2.16 Peripheral Clock STOP Disable Register 1 (SIM_SD1)

Address: SIM_SD1 – F0E0h base + 10h offset = F0F0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	DAC	CMPA	CMPB	CMPC	SCI0	SCI1	QSPI 0	IIC0	IIC1	CRC	REFA	REFB	REFC	0	MSCAN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SD1 field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 DAC	DAC IPBus Clock STOP Disable
13 CMPA	CMPA IPBus Clock STOP Disable
12 CMPB	CMPB IPBus Clock STOP Disable
11 CMPC	CMPC IPBus Clock STOP Disable
10 SCI0	SCI0 IPBus Clock STOP Disable
9 SCI1	SCI1 IPBus Clock STOP Disable
8 QSPI0	QSPI0 IPBus Clock STOP Disable
7 IIC0	IIC0 IPBus Clock STOP Disable
6 IIC1	IIC1 IPBus Clock STOP Disable
5 CRC	CRC IPBus Clock STOP Disable
4 REFA	REFA IPBus Clock STOP Disable
3 REFB	REFB IPBus Clock STOP Disable
2 REFC	REFC IPBus Clock STOP Disable
1 Reserved	This read-only bit is reserved and always has the value zero.
0 MSCAN	MSCAN IPBus Clock STOP Disable

16.2.17 Peripheral Clock STOP Disable Register 2 (SIM_SD2)

Address: SIM_SD2 – F0E0h base + 11h offset = F0F1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												PWMCH0	PWMCH1	PWMCH2	PWMCH3
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SD2 field descriptions

Field	Description
15–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3 PWMCH0	PWM Channel 0 IPBus Clock STOP Disable
2 PWMCH1	PWM Channel 1 IPBus Clock STOP Disable
1 PWMCH2	PWM Channel 2 IPBus Clock STOP Disable
0 PWMCH3	PWM Channel 3 IPBus Clock STOP Disable Each bit enables peripheral clocking during STOP mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The corresponding peripheral is not clocked in STOP mode 1 The corresponding peripheral is clocked in STOP mode

16.2.18 I/O Short Address Location Register (SIM_IOSAHI)

The I/O Short Address Location registers specify the memory referenced through the I/O short address mode, which allows the instruction to specify the lower 6 bits of the address. The upper address bits are not directly controllable. SIM_IOSAHI allows limited control of the full address.

The I/O short address is calculated by concatenating the combined ISAL value with the 6-bit short address from the short address opcode.

Address: SIM_IOSAHI – F0E0h base + 12h offset = F0F2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														ISAL[23:22]	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_IOSAHI field descriptions

Field	Description
15–2 Reserved	This read-only bitfield is reserved and always has the value zero.
1–0 ISAL[23:22]	Bits 23:22 of the address.

16.2.19 I/O Short Address Location Register (SIM_IOSALO)

The I/O Short Address Location registers specify the memory referenced through the I/O short address mode, which allows the instruction to specify the lower 6 bits of the address. The upper address bits are not directly controllable. SIM_IOSALO allows limited control of the full address.

The I/O short address is calculated by concatenating the combined ISAL value with the 6-bit short address from the short address opcode.

Address: SIM_IOSALO – F0E0h base + 13h offset = F0F3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ISAL[21:6]															
Write																
Reset	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0

SIM_IOSALO field descriptions

Field	Description
15–0 ISAL[21:6]	Bits 21:6 of the address.

16.2.20 Protection Register (SIM_PROT)

SIM_PROT provides write protection of selected control fields for safety critical applications. The primary purpose is to prevent unsafe conditions due to the unintentional modification of these fields between the onset of a code runaway and a reset by the COP watchdog. GPIO and Internal Peripheral Select Protection (GIPSP) write-protect the registers in the SIM, XBAR, and GPIO modules that control inter-peripheral signal multiplexing and I/O cell configuration. Peripheral Clock Enable Protection (PCEP) write protects the SIM registers that contain peripheral-specific clock controls. Some peripherals provide additional safety features.

GIPSP protects the contents of the SIM registers that control multiplexing of peripheral signals onto GPIO (GPSn) and the XBAR registers that select among optional peripheral inputs (CODEn). GIPSP also write protects some registers in the GPIO module, including the GPIO_X_PER registers that select between peripheral versus GPIO ownership of the I/O cell, the GPIO_X_PPMODE registers the control the I/O cell push/pull mode, and GPIO_X_DRIVE registers that control the I/O cell drive strength.

PCEP write protects the SIM peripheral clock enable registers (PCEn), the SIM peripheral stop disable registers (SDn), and the SIM peripheral clock rate registers (PCR).

For flexibility, write protection control values may themselves be optionally locked (write protected). To this end, protection controls in this register have two bit values. The right bit determines the setting of the control, and the left bit determines whether the value is locked. While a protection control remains unlocked, protection can be disabled and re-enabled as desired. When a protection control is locked, its value can be altered only by a device reset, which restores its default non-locked value.

Address: SIM_PROT – F0E0h base + 14h offset = F0F4h

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

SIM_PROT field descriptions

Field	Description
15–4 Reserved	This read-only bitfield is reserved and always has the value zero.
	00 Write protection off (default). 01 Write protection on.

Table continues on the next page...

SIM_PROT field descriptions (continued)

Field	Description
	10 Write protection off and locked until device reset. 11 Write protection on and locked until device reset.
3–2 PCEP	Peripheral Clock Enable Protection Enables write protection of all fields in the PCEn, SDn, and PCR registers.
1–0 GIPSP	GPIO and Internal Peripheral Select Protection Enables write protection of GPSn registers in the SIM. GIPSP also write protects external registers, including all XBAR, GPIO_X_PER, GPIO_X_PPMODE, GPIO_X_DRIVE, and GPIO_X_IFE registers if present. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until device reset. 11 Write protection on and locked until device reset.

16.2.21 GPIO Peripheral Select Register 0 (SIM_GPS0)

GPIO with only 1 peripheral function do not require a GPS field. That peripheral function is always enabled when the PER field of the corresponding GPIO is set to 1.

Address: SIM_GPS0 – F0E0h base + 15h offset = F0F5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	C7	C6		0	C5	0	C4	C3		C2		0			C0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPS0 field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 C7	Configure GPIO C7 0 Function = SS0_B; Peripheral = SPI0; Direction = IO 1 Function = TXD0; Peripheral = SCI0; Direction = IO
13–12 C6	Configure GPIO C6 00 Function = TA2; Peripheral = TMRA; Direction = IO 01 Function = XB_IN3; Peripheral = XBAR; Direction = IN 10 Function = CMPREF; Peripheral = HSCMP A/B/C; Direction = AN-IN 11 Reserved

Table continues on the next page...

SIM_GPS0 field descriptions (continued)

Field	Description
11 Reserved	This read-only bit is reserved and always has the value zero.
10 C5	Configure GPIO C5 0 Function = DACO; Peripheral = DACO; Direction = AN_OUT 1 Function = XB_IN7; Peripheral = XBAR; Direction = IN
9 Reserved	This read-only bit is reserved and always has the value zero.
8 C4	Configure GPIO C4 0 Function = TA1; Peripheral = TMRA; Direction = IO 1 Function = CMPB_O; Peripheral = HSCMP-B; Direction = OUT
7–6 C3	Configure GPIO C3 00 Function = TA0; Peripheral = TMRA; Direction = IO 01 Function = CMPA_O; Peripheral = HSCMP_A; Direction = O 10 Function = RXD0; Peripheral = SCI0; Direction = IN 11 Reserved
5–4 C2	Configure GPIO C2 00 Function = TXD0; Peripheral = SCI0; Direction = IO 01 Function = TB0; Peripheral = TMRB; Direction = IO 10 Function = XB_IN2; Peripheral = XBAR; Direction = IN 11 Function = CLKO; Peripheral = SIM; Direction = OUT
3–1 Reserved	This read-only bitfield is reserved and always has the value zero.
0 C0	Configure GPIO C0 0 Function = XTAL; Peripheral = OSC; Direction = AN_IO 1 Function = CLKIN; Peripheral = OCCS; Direction = IN

16.2.22 GPIO Peripheral Select Register 1 (SIM_GPS1)

GPIO with only 1 peripheral function does not require a GPS field. That peripheral function is always enabled when the PER field of the corresponding GPIO is set to 1.

Address: SIM_GPS1 – F0E0h base + 16h offset = F0F6h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	C15	0	C14	0	C13	C12		C11		C10		0	C9	0	C8
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPS1 field descriptions

Field	Description
15 Reserved	This read-only bit is reserved and always has the value zero.
14 C15	Configure GPIO C15 0 Function = SCL0; Peripheral = IIC0; Direction = OD_IO 1 Function = XB_OUT1; Peripheral = XBAR; Direction = OUT
13 Reserved	This read-only bit is reserved and always has the value zero.
12 C14	Configure GPIO C14 0 Function = SDA0; Peripheral = IIC0; Direction = OD_IO 1 Function = XB_OUT0; Peripheral = XBAR; Direction = OUT
11 Reserved	This read-only bit is reserved and always has the value zero.
10 C13	Configure GPIO C13 0 Function = TA3; Peripheral = TMRA; Direction = IO 1 Function = XB_IN6; Peripheral = XBAR; Direction = IN
9–8 C12	Configure GPIO C12 00 Function = RX0; Peripheral = MSCAN0; Direction = IN 01 Function = SDA1; Peripheral = IIC1; Direction = OD_IO 10 Function = RXD1; Peripheral = SCI1; Direction = IN 11 Reserved
7–6 C11	Configure GPIO C11 00 Function = TX0; Peripheral = MSCAN0; Direction = OD_OUT 01 Function = SCL1; Peripheral = IIC1; Direction = OD_IO 10 Function = TXD1; Peripheral = SCI1; Direction = IO 11 Reserved
5–4 C10	Configure GPIO C10 00 Function = MOSI0; Peripheral = SPI0; Direction = IO 01 Function = XB_IN5; Peripheral = XBAR; Direction = IN 10 Function = MISO0; Peripheral = SPI0; Direction = IO 11 Reserved
3 Reserved	This read-only bit is reserved and always has the value zero.
2 C9	Configure GPIO C9 0 Function = SCLK0; Peripheral = SPI0; Direction = IO 1 Function = XB_IN4; Peripheral = XBAR; Direction = IN
1 Reserved	This read-only bit is reserved and always has the value zero.
0 C8	Configure GPIO C8

Table continues on the next page...

SIM_GPS1 field descriptions (continued)

Field	Description
	0 Function = MISO0; Peripheral = SPI0; Direction = IO 1 Function = RXD0; Peripheral = SCI0; Direction = IN

16.2.23 GPIO Peripheral Select Register 2 (SIM_GPS2)

GPIO with only 1 peripheral function does not require a GPS field. That peripheral function is always enabled when the PER field of the corresponding GPIO is set to 1.

Address: SIM_GPS2 – F0E0h base + 17h offset = F0F7h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			F6	0	F5	0	F4	0	F3	0	F2	0	F1	0	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPS2 field descriptions

Field	Description
15–13 Reserved	This read-only bitfield is reserved and always has the value zero.
12 F6	Configure GPIO F6 0 Function = TB2; Peripheral = TMRB; Direction = IO 1 Function = PWMX3; Peripheral = PWM; Direction = IO
11 Reserved	This read-only bit is reserved and always has the value zero.
10 F5	Configure GPIO F5 0 Function = RXD1; Peripheral = SCI1; Direction = IN 1 Function = XB_OUT5; Peripheral = XBAR; Direction = OUT
9 Reserved	This read-only bit is reserved and always has the value zero.
8 F4	Configure GPIO F4 1 Function = XB_OUT4; Peripheral = XBAR; Direction = OUT 0 Function = TXD1; Peripheral = SCI1; Direction = IO
7 Reserved	This read-only bit is reserved and always has the value zero.
6 F3	Configure GPIO F3 0 Function = SDA1; Peripheral = IIC1; Direction = OD_IO 1 Function = XB_OUT3; Peripheral = XBAR; Direction = OUT

Table continues on the next page...

SIM_GPS2 field descriptions (continued)

Field	Description
5 Reserved	This read-only bit is reserved and always has the value zero.
4 F2	Configure GPIO F2 0 Function = SCL1; Peripheral = IIC1; Direction = OD_OUT 1 Function = XB_OUT2; Peripheral = XBAR; Direction = OUT
3 Reserved	This read-only bit is reserved and always has the value zero.
2 F1	Configure GPIO F1 0 Function = CLKOUT; Peripheral = SIM; Direction = OUT 1 Function = XB_IN7; Peripheral = XBAR; Direction = IN
1–0 Reserved	This read-only bitfield is reserved and always has the value zero.

16.2.24 GPIO Peripheral Select Register 3 (SIM_GPS3)

GPIO with only 1 peripheral function does not require a GPS field. That peripheral function is always enabled when the PER field of the corresponding GPIO is set to 1.

The TMRBn fields select which signal drives the TMRBn input. The choice in each case is between an external I/O pad or an output. When selecting the external I/O, the GPIO_X_PER bit for that I/O must be set to 1 and the SIM GPS field (if there is one) for that GPIO must be configured to feed TMRB.

Address: SIM_GPS3 – F0E0h base + 18h offset = F0F8h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TMRB3	TMRB2	TMRB1	TMRB0	0	E7	0	E6	0	E5	0	E4	0	F8	0	A0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPS3 field descriptions

Field	Description
15 TMRB3	Select TMRB3 Input 0 Function = GPIOF7; Peripheral = GPIOF; Direction = IN 1 Function = XB_OUT29; Peripheral = XBAR; Direction = IN
14 TMRB2	Select TMRB2 Input 0 Function = GPIOF6; Peripheral = GPIOF; Direction = IN 1 Function = XB_OUT28; Peripheral = XBAR; Direction = IN

Table continues on the next page...

SIM_GPS3 field descriptions (continued)

Field	Description
13 TMRB1	Select TMRB1 Input 0 Function = GPIOF8; Peripheral = GPIOF; Direction = IN 1 Function = XB_OUT27; Peripheral = XBAR; Direction = IN
12 TMRB0	Select TMRB0 Input Settings 0 Function = GPIOC2; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT26; Peripheral = XBAR; Direction = IN
11 Reserved	This read-only bit is reserved and always has the value zero.
10 E7	Configure GPIO E7 0 Function = PWMA3; Peripheral = PWM; Direction = IO 1 Function = XB_IN5; Peripheral = XBAR; Direction = IN
9 Reserved	This read-only bit is reserved and always has the value zero.
8 E6	Configure GPIO E6 0 Function = PWMB3; Peripheral = PWM; Direction = IO 1 Function = XB_IN4; Peripheral = XBAR; Direction = IN
7 Reserved	This read-only bit is reserved and always has the value zero.
6 E5	Configure GPIO E5 0 Function = PWMA2; Peripheral = PWM; Direction = IO 1 Function = XB_IN3; Peripheral = XBAR; Direction = IN
5 Reserved	This read-only bit is reserved and always has the value zero.
4 E4	Configure GPIO E4 0 Function = PWMB2; Peripheral = PWM; Direction = IO 1 Function = XB_IN2; Peripheral = XBAR; Direction = IN
3 Reserved	This read-only bit is reserved and always has the value zero.
2 F8	Configure GPIO F8 0 Function = RXD0; Peripheral = SCI0; Direction = IN 1 Function = TB1; Peripheral = TMRB; Direction = IO
1 Reserved	This read-only bit is reserved and always has the value zero.
0 A0	Configure GPIO A0 0 Function = ANA0/CMPA_P2; Peripheral = ADC/CMPA; Direction = AN_IN 1 Function = CMPC_O; Peripheral = CMPC; Direction = OUT

16.3 Functional Description

16.3.1 Clock Generation Overview

The SIM uses master clocks from the OCCS module to produce the peripheral and system (DSC core and memory) clocks. A mstr_2x clock input from OCCS operates at two times the system and peripheral bus rate and therefore a maximum of 120 MHz. Peripheral and system clocks are generated at a maximum of 60 MHz by dividing the mstr_2x clock by 2 and gating it with appropriate power mode and clock gating controls. The TMR and SCI peripheral clocks can optionally be generated at two times the normal rate, at a maximum of 120 MHz. These clocks are generated by gating the mstr_2x clock with appropriate power mode and clock gating controls.

The OCCS configuration controls the operating frequency of the SIM's master clocks. In the OCCS, an external clock (CLKIN), a crystal oscillator, or the relaxation oscillator can be selected as the master clock source (mstr_osc). An external clock can be operated at any frequency up to 120 MHz. An external clock is presented directly to the SIM as the mstr_2x clock, so the duty cycle of the high-speed peripheral clock would reflect the precision of the duty cycle of the external clock. The crystal oscillator can be operated at between 8 MHz and 16 MHz. The relaxation oscillator can be operated at full speed (8 MHz), standby speed (400 kHz using ROSB), or powered down (using ROPD). An 8 MHz or 16 MHz mstr_osc can be multiplied to 240 MHz using the PLL and postscaled to provide a variety of speed clock rates. Either the postscaled PLL output or mstr_osc signal can be selected to produce the master clocks to the SIM.

In combination with the OCCS module, the SIM provides power modes (see the following section), clock enables (PCEn registers, SDn registers, CLKDIS, ONCEEBL), and clock rate controls (TMRn_CR, SCIn_CR) to provide flexible control of clocking and power use. The SIM's PCEn peripheral clock enable controls can be used to disable individual peripheral clocks when they are not needed. The clock rate controls enable the high speed clocking option for the general-purpose timers and the SCIs. See the OCCS chapter for further details.

16.3.2 Power Down Modes Overview

The 56800E DSC core operates in the power modes shown in the following table.

Table 16-27. Clock Operation in Power Modes

Mode	System Clocks	Peripheral Clocks	Description
Run	Core and memory clocks enabled	Peripheral clocks enabled	Device is fully functional
Wait	Core and memory clocks disabled	Peripheral clocks enabled	Core executes WAIT instruction to enter this mode. Wait mode is typically used for power conscious applications. Possible recoveries from wait mode to run mode are: 1. Any interrupt. 2. Executing a debug mode entry command using the DSC core JTAG interface. 3. Any reset (such as POR, external, software, and COP).
Stop	Master clock generation in the OCCS remains operational, but the SIM disables the generation of system and peripheral clocks.		Core executes STOP instruction to enter this mode. Possible recoveries from stop mode to run mode are: 1. Interrupt from any peripheral configured in SD register to operate in stop mode. 2. Low voltage interrupt 3. Executing a debug mode entry command using the DSC core JTAG interface. 4. Any reset.

Run, wait, and stop modes provide means of enabling/disabling the peripheral and/or core clocking as a group. The stop disable controls in the SDn registers override the default behavior of stop mode. Asserting a peripheral's stop disable bit means the peripheral's clock continues to operate in stop mode. This option is useful for generating interrupts that return the device from stop to run mode.

On-chip peripherals run at the IP bus clock (peripheral bus) frequency,[1] which is the same as the main processor frequency in this architecture. The maximum frequency of operation is sys_clk=60 MHz. The only exceptions are the general-purpose timers and SCIs, which can be configured to operate at two times the system bus rate using TMRn_CR and SCIn_CR controls.

Run, wait, and stop modes may be combined with the low power modes of the PS and choice of clocks to provide a broad palette of power control techniques.

1. The TMR and PWM modules can be operated at three times the IPBus clock frequency.

16.3.3 Stop and Wait Mode Disable Function

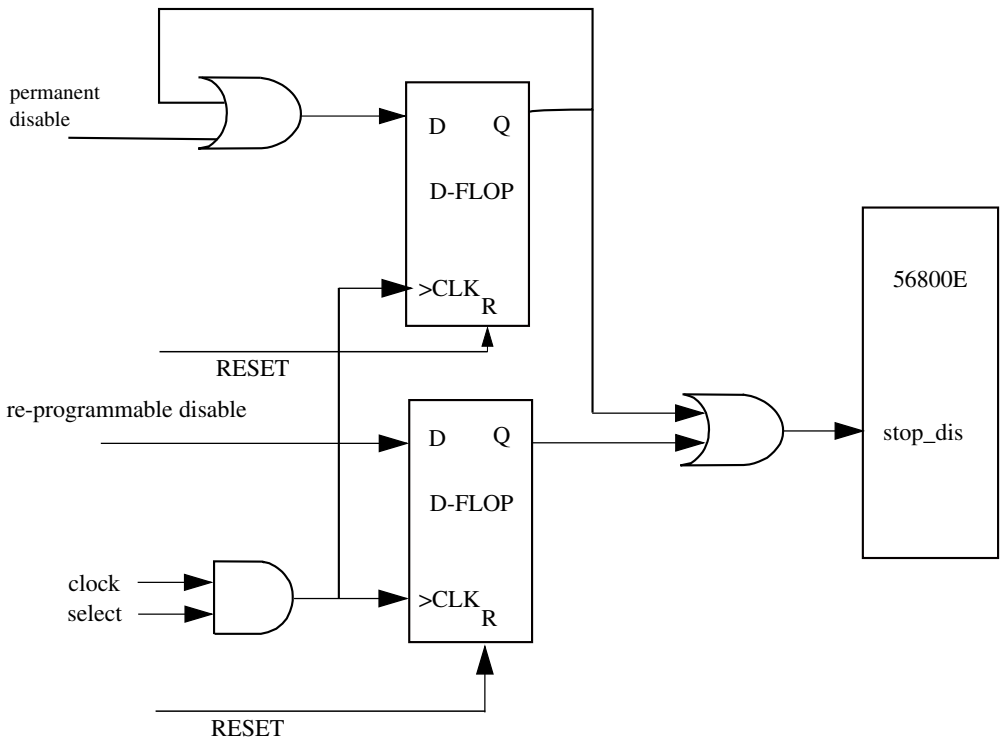


Figure 16-25. Stop Disable Circuit

The DSC core contains both STOP and WAIT instructions. Both put the CPU to sleep. The peripheral bus continues to run in wait mode, but in stop mode, only peripherals whose SDn control is asserted continue to run. Entry into stop or wait mode does not affect the OCCS configuration and affects only the generation of system and peripheral clocks from the master clocks from the OCCS. The OCCS may be reconfigured prior to entering stop or wait, if desired, to reduce master clock frequencies and thus the power use within the OCCS.

Some applications require the DSC STOP/WAIT instructions to be disabled. Control fields are provided in the CTRL register to disable wait and/or stop modes. This procedure can be on either a permanent (until reset) or temporary basis.

16.4 Resets

The SIM supports several sources of reset, as shown in the following table.

Table 16-28. Sources of Reset

Label	Source of Reset	Timing
EXTR	External Reset	Asynchronous
POR	Power-On-Reset (PS)	Asynchronous
COP_CPU	COP CPU reset	Synchronous
COP_LOR	COP Loss of Reference reset	Synchronous
LVDR	Low voltage detect reset (PS)	Synchronous
SWR	Software reset (SIM)	Synchronous

The following figure provides a graphic illustration of the details. Note that the POR_Delay blocks use osc_clk as their time base because other system clocks are inactive during this phase of reset.

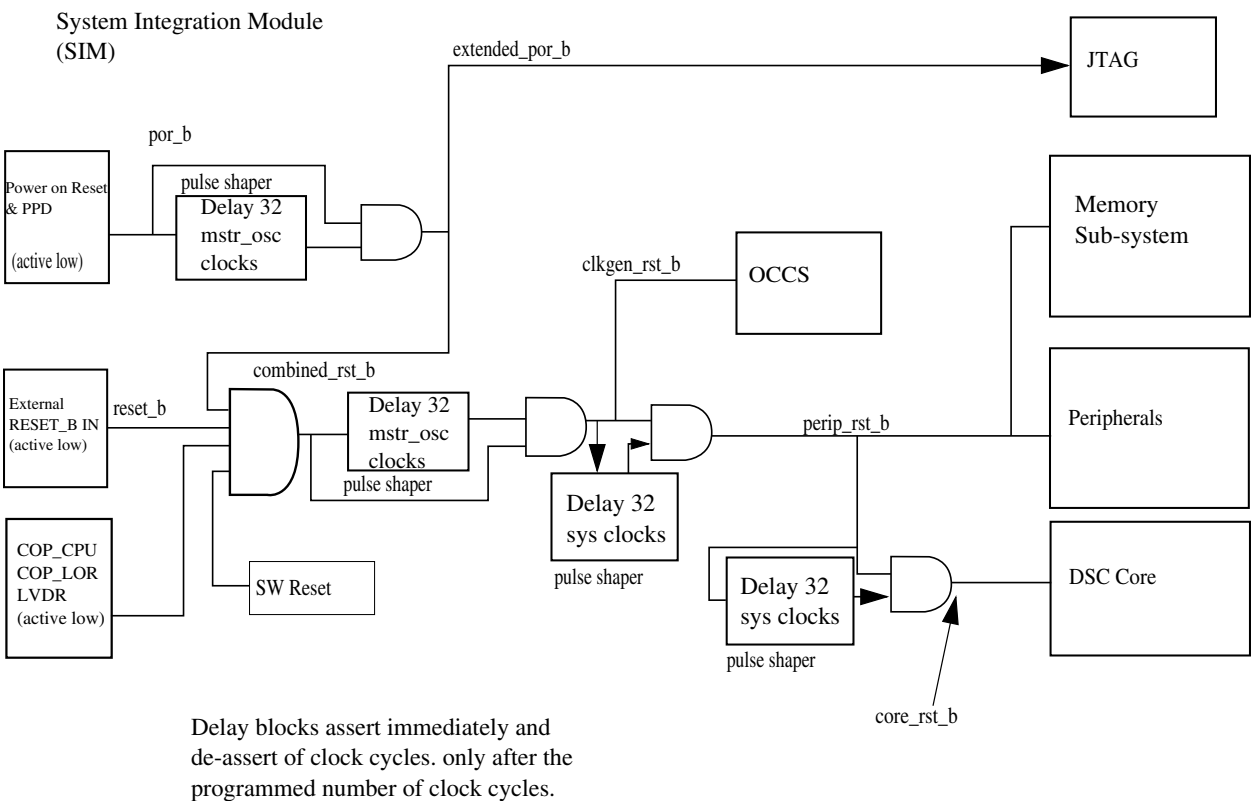


Figure 16-26. Sources of RESET Functional Diagram

POR resets are extended 64 mstr_osc clocks to stabilize the power supply and clock source. All resets are subsequently extended for an additional 32 mstr_osc clocks and 64 system clocks as the various internal reset controls are released. Actual duration is determined by the primary input clock source (mstr_osc) and mstr_2x clock frequency.

An external reset generation chip may also be used. Resets may be asserted asynchronously, but they are always released internally on a rising edge of the system clock.

16.5 Interrupts

The SIM generates no interrupts.

Chapter 17
Power Supervisor (PS)

17.1 Introduction

17.1.1 Overview

The on-device power supervisor (PS) module contains the core voltage regulator and power monitoring circuitry. The PS ensures that the device operates only within legal voltage ranges and assists in the orderly shutdown of the device if the power supply is interrupted. It also regulates the internal voltage rails for the core digital and analog logic.

17.1.2 Features

- Power-on reset ($\overline{\text{POR}}$) is generated until both core logic V_{DD} and I/O V_{DD} exceed their acceptable minimums.
- $\overline{\text{POR}}$ is asserted when the core logic V_{DD} drops below approximately 1.8V.
- A low voltage interrupt is generated when the I/O V_{DD} drops below approximately 2.7V.
- A low voltage interrupt is generated when the core logic V_{DD} drops below approximately 2.1V.
- All LVI levels incorporate 50–100mV of hysteresis.

It is assumed that power supply voltages move relatively slowly with respect to the system clocks, allowing the device some control over its operation.

17.1.3 Modes of Operation

The PS has the most effect on the device when the power supply voltages are moving. The $\overline{\text{POR}}$ is always enabled, and the low voltage interrupts are disabled out of reset. As the device is powered up, the $\overline{\text{POR}}$ is released. The following figure illustrates the integration of the power supervisor into the device. Note that both digital supplies are monitored. Assuming that the I/O supply is used as the source for the core supply regulator, expect to see a 2.7 V LVI prior to seeing the 2.1 V LVI (which can presumably stay in regulation longer).

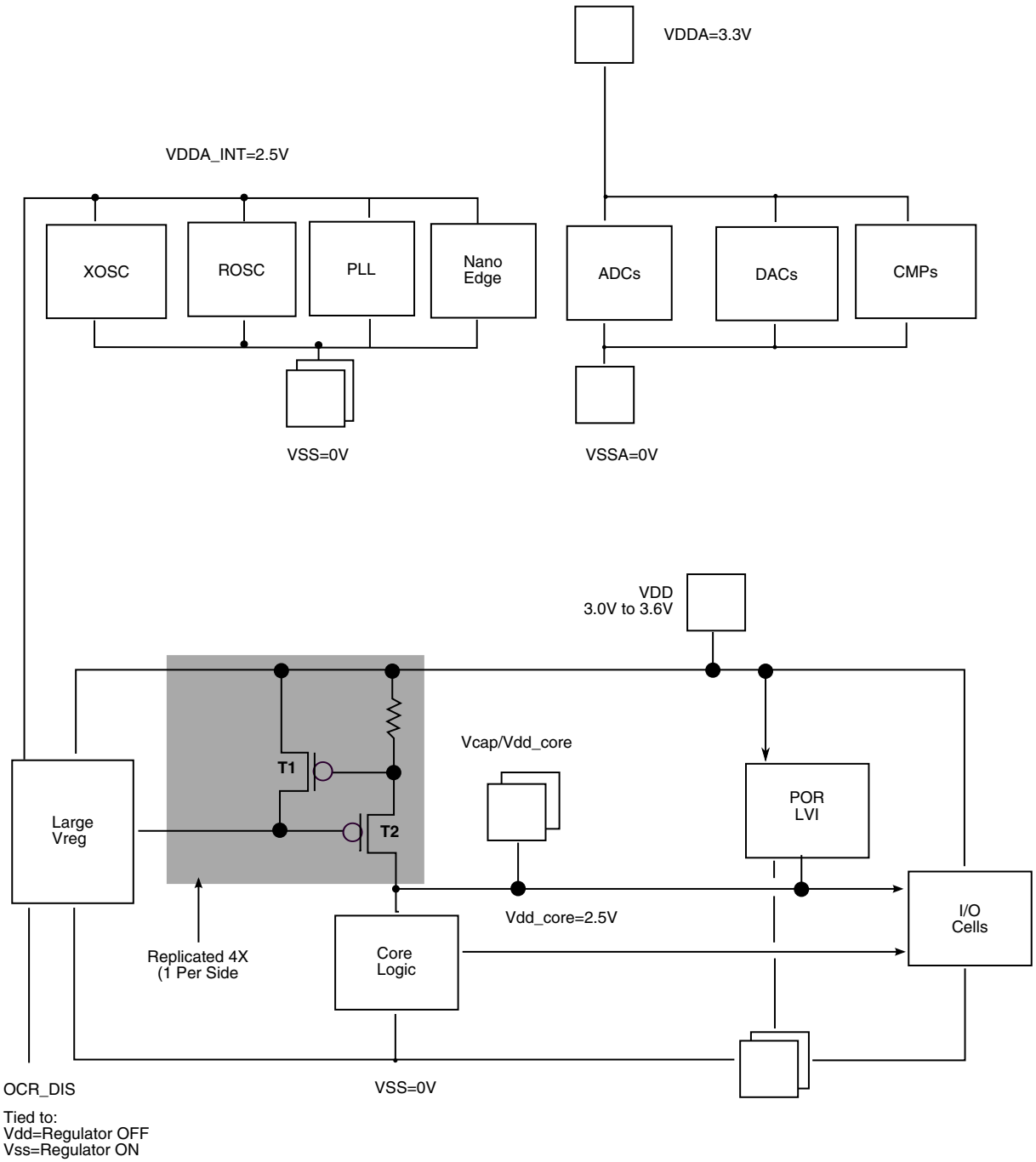


Figure 17-1. Integration of the Power Supervisor

17.1.4 Block Diagram

The basic POR and low voltage detect module appears in the following figure. The POR circuit is designed to assert the internal reset when V_{DD} is in the range from 0 V to 1.8 V. $\overline{POR}$ switching high indicates that the core and I/O voltages are above their thresholds. The deglitch blocks are essentially strings of four flops in series. The outputs of all four must agree before the STS[LV27F] or STS[LV21F] bits change state. This constraint helps to prevent the LVI circuitry from responding to momentary glitches that occur as part of normal operation.

Note

The deglitch flops must be clocked by a continuously running clock so these interrupts can wake the device if it is in stop mode.

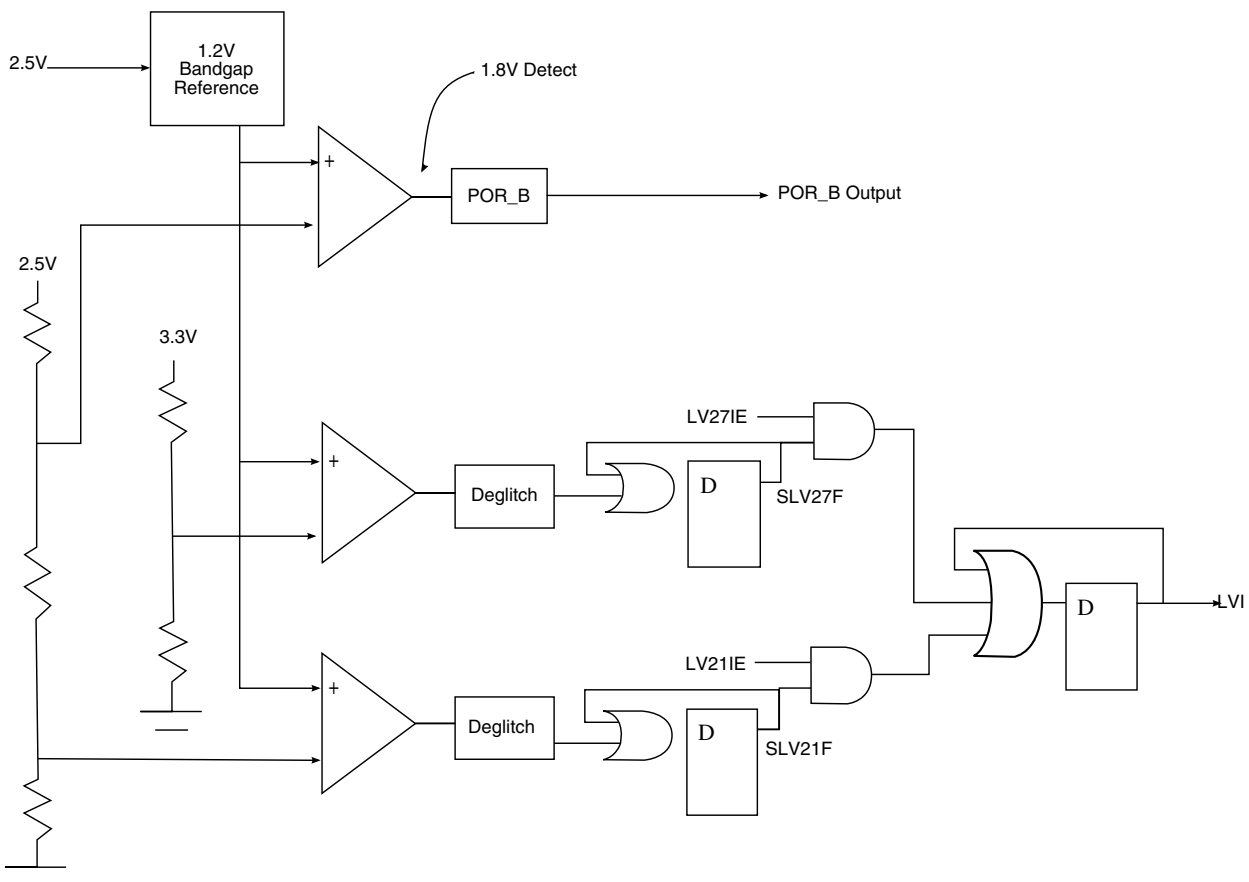


Figure 17-2. PS Block Diagram

17.2 Functional Description

The following figure illustrates the operation of the $\overline{POR}$ versus low voltage detect circuits. The interrupt service routine responsible for shutting down the device when a low voltage is detected should explicitly clear and disable the low-voltage interrupts.

Low-voltage interrupts are masked at $\overline{POR}$. Afterwards, they must be explicitly enabled and disabled. Low-voltage interrupts include roughly 50mV of hysteresis.

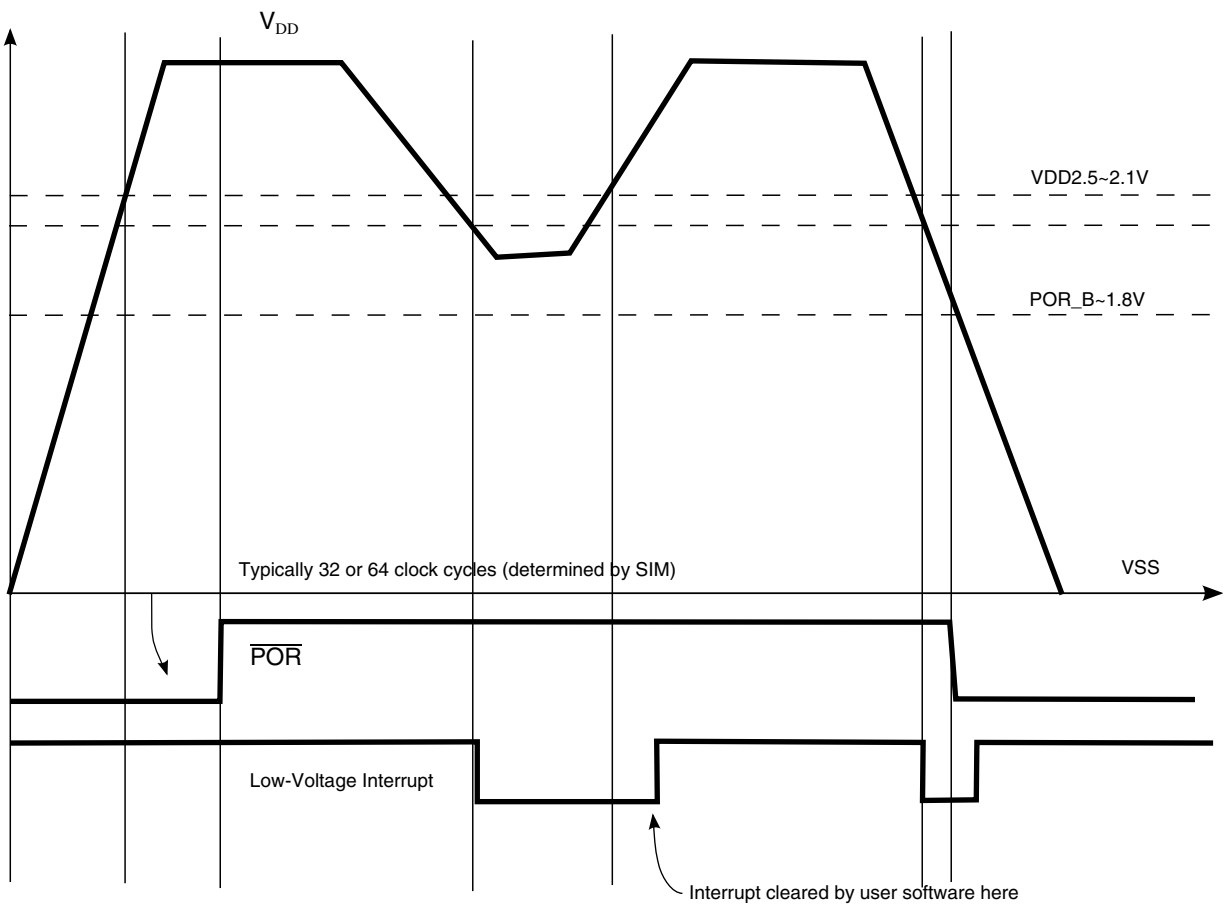


Figure 17-3. $\overline{POR}$ Versus Low-Voltage Interrupts

17.3 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PS_CTRL	R	0												HV27IE	HV21IE	LV27IE	LV21IE
		W																
1	PS_STS	R	0											LVI	SLV27F	SLV21F	LV27F	LV21F
		W																

17.3.1 Power Supervisor Control Register (PS_CTRL)

Address: PS_CTRL – F130h base + 0h offset = F130h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												HV27IE	HV21IE	LV27IE	LV21IE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PS_CTRL field descriptions

Field	Description
15–4 Reserved	This read-only bitfield is reserved and always has the value zero.
3 HV27IE	This bit allows the STS[LVI] bit to be set when the STS[LV27F] bit is clear. After a low voltage condition occurs, set this bit and CTRL[HV21IE], clear CTRL[LV27IE and LV22IE], and an interrupt will be generated once the voltage levels have returned to normal. 2.7 V High Voltage Interrupt Enable 0 Disable setting the high voltage interrupt. 1 Enable setting the high voltage interrupt.
2 HV21IE	This bit allows the STS[LVI] bit to be set when the STS[LV21F] bit is clear. After a low voltage condition occurs, set this bit and CTRL[HV27IE], clear CTRL[LV27IE and LV21IE], and an interrupt will be generated once the voltage levels have returned to normal. **NOTE** If both CTRL[HV27IE and HV21IE] are set, then both STS[LV27F and LV21F] must be clear before STS[LVI] is set. 2.1 V High Voltage Interrupt Enable 0 Disable setting the high voltage interrupt. 1 Enable setting the high voltage interrupt.

Table continues on the next page...

PS_CTRL field descriptions (continued)

Field	Description
1 LV27IE	This bit allows the STS[LVI] bit to be set when the STS[LV27F] bit is set. 2.7 V Low Voltage Interrupt Enable 0 Disable setting the low voltage interrupt. 1 Enable setting the low voltage interrupt.
0 LV21IE	This bit allows the STS[LVI] bit to be set when the STS[LV21F] bit is set. 2.1 V Low Voltage Interrupt Enable 0 Disable setting the low voltage interrupt. 1 Enable setting the low voltage interrupt.

17.3.2 Power Supervisor Status Register (PS_STS)

Address: PS_STS – F130h base + 1h offset = F131h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0											LVI	SLV27F	SLV21F	LV27F	LV21F
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PS_STS field descriptions

Field	Description
15–5 Reserved	This read-only bitfield is reserved and always has the value zero.
4 LVI	Low Voltage Interrupt This read-only bit is the low voltage interrupt. This bit is set by several conditions: • STS[LV21F] and CTRL[LV21IE] • STS[LV27F] and CTRL[LV27IE] • $\overline{\text{STS[LV21F]}}$ and CTRL[HV21IE] • $\overline{\text{STS[LV27F]}}$ and CTRL[HV27IE] **NOTE:** If both CTRL[HV27IE and HV21IE] are set, then both STS[LV27F and LV21F] must be clear before STS[LVI] is set. Once set, this bit will remain set until a 1 is written to this bit position or a reset occurs. Writing a 0 will have no effect. 0 Low voltage interrupt cleared. 1 Low voltage interrupt asserted.
3 SLV27F	Sticky 2.7 V Low Voltage Flag

Table continues on the next page...

PS_STS field descriptions (continued)

Field	Description
	This sticky read-only bit indicates that the 3.3 V supply dropped below the 2.7 V level at some point. Once set, this bit will remain set until a 1 is written to this bit position or a reset occurs. Writing a 0 will have no effect. 0 3.3 V supply has not dropped below the 2.7 V threshold. 1 3.3 V supply has dropped below the 2.7 V threshold.
2 SLV21F	Sticky 2.1 V Low Voltage Flag This sticky read-only bit indicates that the 2.5 V supply dropped below the 2.1 V level at some point. Once set, this bit will remain set until a 1 is written to this bit position or a reset occurs. Writing a 0 will have no effect. 0 2.5 V supply has not dropped below the 2.1 V threshold. 1 2.5 V supply has dropped below the 2.1 V threshold.
1 LV27F	2.7 V Low Voltage Flag This read-only bit indicates that the 3.3 V supply is currently below the 2.7 V level. This bit may reset itself if the supply voltage rises above the threshold. 0 3.3 V supply is not below the 2.7 V threshold. 1 3.3 V supply is below the 2.7 V threshold.
0 LV21F	2.1 V Low Voltage Flag This read-only bit indicates that the 2.5 V supply is currently below the 2.1 V level. This bit may reset itself if the supply voltage rises above the threshold. 0 2.5 V supply is not below the 2.1 V threshold. 1 2.5 V supply is below the 2.1 V threshold.

17.4 Resets

Table 17-4. Reset Summary

Reset	Source	Characteristics
$\overline{\text{POR}}$	This module	When asserted, indicates that the supply voltage is too low for reliable operation.
$\overline{\text{RESET}}$	SIM	Resets the power supervisor registers. This signal is usually derived from $\overline{\text{POR}}$ and other device reset sources.

17.5 Clocks

Table 17-5. Clock Summary

Clock	Source	Used by
IPbus clock	SIM	Used by the glitch filter during normal operation and by control registers at all times.

Table continues on the next page...

Table 17-5. Clock Summary (continued)

Clock	Source	Used by
Oscillator clock	Oscillator	Used by the glitch filter during stop mode and during power-on resets.

17.6 Interrupts

Table 17-6. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name
LVI	STS[SLV27F] STS[SLV21F]	CTRL[LV27IE] CTRL[LV21IE] CTRL[HV27IE] CTRL[HV21IE]	Low-voltage interrupt

Chapter 18
Computer Operating Properly (COP)

18.1 Introduction

The computer operating properly (COP) module is used to help software recover from runaway code. The COP is a free-running down counter that, once enabled, is designed to generate a reset upon reaching zero. Software must periodically service the COP in order to reload the counter and prevent a reset.

18.1.1 Features

The COP module includes these distinctive features:

- Programmable prescaler
- Programmable timeout period = (cop_prescaler*(TIMEOUT + 1)) clock cycles, where TIMEOUT can be from 0x0000 to 0xFFFF
- Programmable wait and stop operation
- COP timer is disabled while the DSC is in debug mode
- Causes loss of reference reset 128 cycles after loss of reference clock to the PLL is detected
- Choice of one of three clock sources for counter
  - Relaxation oscillator (ROSC): 4 MHz or 400 kHz
  - Crystal oscillator (COSC): 4 MHz to 16 MHz
  - System bus clock (IP Bus clock): up to 60 MHz

18.1.2 Block Diagram

The block diagram of the COP module follows.

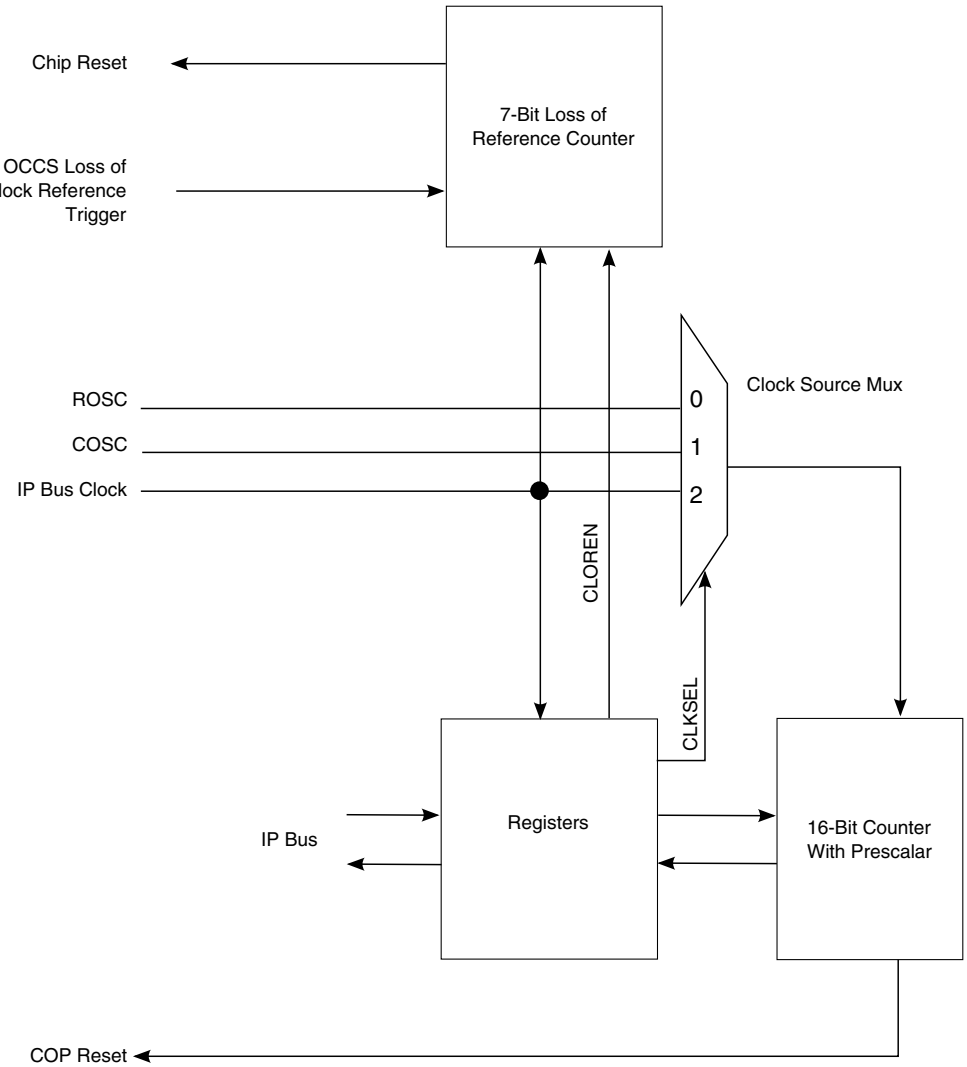


Figure 18-1. COP Module Block Diagram

18.2 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	COP_CTRL	R	0						PSS		0	CLKSEL		CLOREN	CSEN	CWEN	CEN	CWP
		W																
1	COP_TOUT	R	TIMEOUT															
		W																
2	COP_CNTR	R	COUNT_SERVICE															
		W																

18.2.1 COP Control Register (COP_CTRL)

Address: COP_CTRL – F110h base + 0h offset = F110h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0						PSS		0	CLKSEL		CLOREN	CSEN	CWEN	CEN	CWP
Write																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

COP_CTRL field descriptions

Field	Description
15–10 Reserved	This read-only bitfield is reserved and always has the value zero.
9–8 PSS	Prescaler Select This two bit field determines the value of the clock divider (prescaler). You may divide the source clock by 1, 16, 256, or 1024. Generally, you use a lower prescaler value for lower frequency clock sources, but any combination of PSS and timeout may be used as long as they yield the desired timeout value. **Restriction:** This field can be changed only when CWP is set to zero. 00 No division 01 Divide by 16 10 Divide by 256 11 Divide by 1024
7 Reserved	This read-only bit is reserved and always has the value zero.
6–5 CLKSEL	Clock Source Select This bitfield selects the clock source for the COP counter. Some safety applications require the watchdog counter to use a clock source different than the system clock. **Restriction:** This field can be changed only when CWP is set to zero. It also should be changed only when CEN is clear. 00 Relaxation oscillator output (ROSC) is used to clock the counter (default) 01 Crystal oscillator output (COSC) is used to clock the counter 10 IP bus clock is used to clock the counterRestriction:Do not select the IP bus clock to clock the counter if the application requires the COP to wake the device from stop mode. 11 Reserved
4 CLOREN	COP Loss of Reference Enable This bit enables the operation of the COP loss of reference counter. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP loss of reference counter is disabled. (default) 1 COP loss of reference counter is enabled.

Table continues on the next page...

COP_CTRL field descriptions (continued)

Field	Description
3 CSEN	COP Stop Mode Enable This bit controls the operation of the COP counter in stop mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter stops in stop mode. (default) 1 COP counter runs in stop mode if CEN is set to one.
2 CWEN	COP Wait Mode Enable This bit controls the operation of the COP counter in wait mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter stops in wait mode. (default) 1 COP counter runs in wait mode if CEN is set to one.
1 CEN	COP Enable This bit controls the operation of the COP counter. This bit always reads as zero when the chip is in debug mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter is disabled. 1 COP counter is enabled. (default)
0 CWP	COP Write Protect This bit controls the write protection feature of the COP control register (CTRL) and the COP timeout register (TOUT). Once set, this bit can be cleared only by resetting the module. 0 The CTRL and TOUT registers are readable and writable. (default) 1 The CTRL and TOUT registers are read-only.

18.2.2 COP Timeout Register (COP_TOUT)

The value in this register determines the timeout period of the COP counter.

Considerations about setting the timeout value follow:

1. TIMEOUT should be written before the COP is enabled. Changing TIMEOUT while the COP is enabled results in a timeout period that differs from the expected value.
2. After the COP has been enabled:
   - The recommended procedure for changing TIMEOUT is to disable the COP, write to TOUT, and then re-enable the COP. This procedure ensures that the new TIMEOUT is loaded into the counter.
   - Alternatively, the CPU can write to TOUT and then write the proper patterns to CNTR to cause the counter to reload with the new TIMEOUT value.

Address: COP_TOUT – F110h base + 1h offset = F111h

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

COP_TOUT field descriptions

Field	Description
15–0 TIMEOUT	COP Timeout Period The value in this register determines the timeout period of the COP counter. **Restriction:** These bits can be changed only when CWP is set to zero.

18.2.3 COP Counter Register (COP_CNTR)

Address: COP_CNTR – F110h base + 2h offset = F112h

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

COP_CNTR field descriptions

Field	Description
15–0 COUNT_ SERVICE	COP Count/Service COP Count: When this register is read, its value is the current value of the COP counter as it counts down from the timeout value to zero. A reset is issued when this count reaches zero. COP Service: Write to this register to service the counter. When enabled, the COP requires that a service sequence be performed periodically to clear the COP counter and prevent a reset from being issued. • This routine consists of writing 0x5555 to CNTR followed by writing 0xAAAA before the timeout period expires. • These writes to CNTR must be performed in the correct order, but any number of other instructions (and writes to other registers) may be executed between the two writes.

18.3 Functional Description

When the COP is enabled, each positive edge of the prescaled clock (COSC, ROSC, or IP Bus clock) causes the counter to decrement by one. If the count reaches a value of 0x0000, then the device is reset. For the DSC core to show that it is operating properly, it must perform a service routine before the count reaches 0x0000. The service routine consists of writing 0x5555 followed by 0xAAAA to the CNTR register.

18.3.1 COP after Reset

CEN is set out of reset. Thus the counter is enabled by default. In addition, the TOUT register is set to its maximum value of 0xFFFF and PRESCALER is set to 1024 (CTRL[PSS]=11) during reset so the counter is loaded with a maximum timeout period when reset is released.

If the IP bus clock to the COP is not enabled by default after reset, then allow 2 clock cycles to occur after enabling it before performing a write access to the COP.

18.3.2 Wait Mode Operation

If wait mode is entered with both CEN and CWEN set to 1, then the COP counter continues to count down. In that case, a COP reset is issued to wake the device after the counter reaches zero. If either CEN or CWEN is cleared to 0 when wait mode is entered, then the counter is disabled and reloads using the value in the TOUT register.

18.3.3 Stop Mode Operation

If stop mode is entered with both CEN and CSEN set to 1, then the COP counter continues to count down. In that case, a COP reset is issued to wake the device after the counter reaches zero. If either CEN or CSEN is cleared to 0 when stop mode is entered, then the counter is disabled and reloads using the value in the TOUT register.

18.3.4 Debug Mode Operation

The COP counter is not allowed to count when the device is in debug mode. In addition, the CEN bit in the CTRL register always reads as zero when the device is in debug. The actual value of CEN is unaffected by debug, however, and resumes its previously set value upon exiting debug.

18.3.5 Loss of Reference Operation

When the OCCS signals the COP that a loss of the reference clock has occurred and the CLOREN bit is set, then the COP starts a 7-bit counter that runs off of the IP bus clock (which continues to be produced by the PLL for at least 1000 cycles upon losing its reference). The counter continues to count once started counting. When this counter

reaches 0x7F, it causes a loss of reference reset that resets the entire device. If the software has safely shut down the device and does not want a full reset, then the loss of reference timeout count can be delayed by servicing the COP counter in the standard manner of writing 0x5555 followed by 0xAAAA or stopped by setting CLOREN to zero.

18.4 Resets

Any system reset forces all registers to their reset state and clears the COP_RST_B or LOR_RST_B signals if they are asserted. The counter is loaded with its maximum value of 0xFFFF, the prescaler is set to 0x3FF, and the counter restarts when reset is released because CEN is enabled by default.

18.5 Clocks

The COP timer base is the COSC, ROSC, or IP Bus clock divided by the prescaler value (1 - 1024).

18.6 Interrupts

The COP module does not generate any interrupts. It does generate the COP reset signal when the counter reaches a value of 0x0000, which causes a chip-wide reset. It also generates a chip-wide reset signal 128 IP Bus cycles after the loss of the reference clock is detected.

NOTE

The chip reset vector base address is located at P:00h. The COP reset vector base address is located at P:02h. For more details, refer to the interrupt vector table in the device's data sheet.

Chapter 19
Cyclic Redundancy Check Generator (CRC)

19.1 Introduction

The Cyclic Redundancy Check (CRC) generator module uses the 16-bit CRC-CCITT polynomial, $x^{16} + x^{12} + x^5 + 1$, to generate a CRC code for error detection. The 16-bit code is calculated for 8 bits of data at a time and provides a simple check for all accessible memory locations, whether they be in flash memory or RAM.

19.1.1 Features

Features of the CRC module include:

- Hardware CRC generator circuit using 16-bit shift register
- CRC16-CCITT compliancy with $x^{16} + x^{12} + x^5 + 1$ polynomial
- Error detection for all single, double, odd, and most multi-bit errors
- Programmable initial seed value
- High-speed CRC calculation
- Optional ability to transpose input data and CRC result via transpose register, required on applications where bytes are in LSb (Least Significant bit) format.

19.1.2 Modes of Operation

This section defines the CRC operation in run, wait, and stop modes.

- Run Mode - This is the basic mode of operation.

- Wait Mode - The CRC module is operational.
- Stop Mode - The CRC module is not functional in these modes and will be put into its reset state upon recovery from stop.

19.1.3 Block Diagram

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

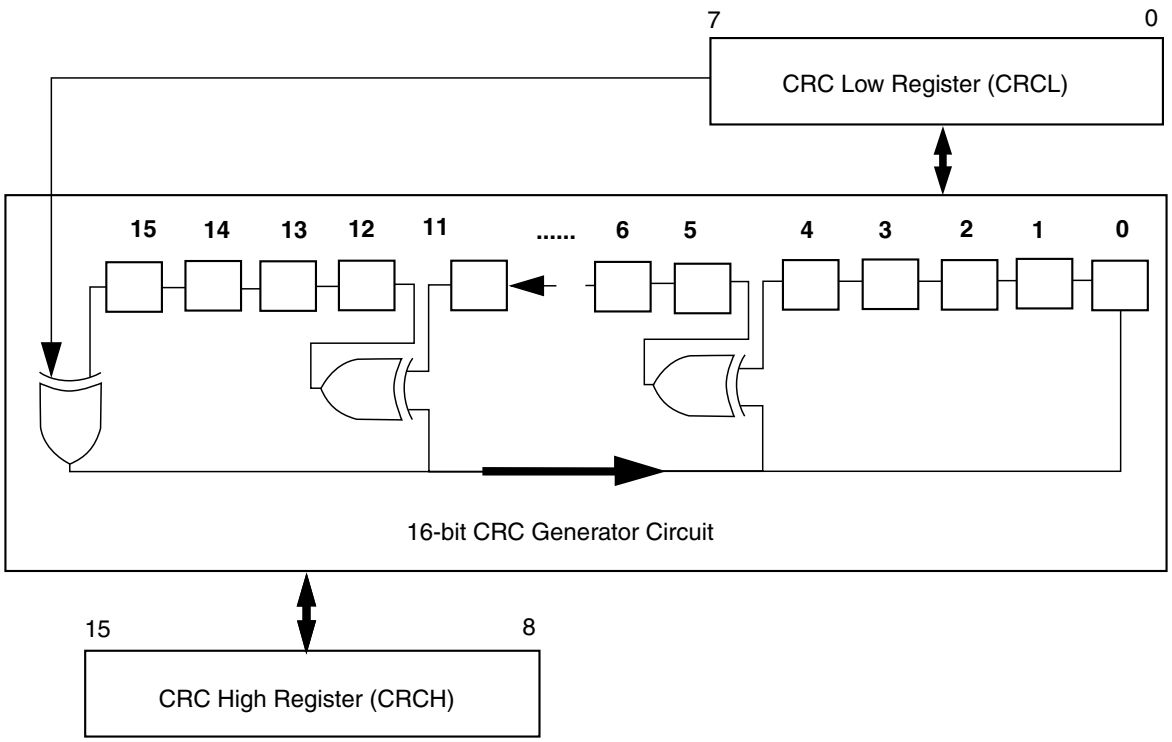


Figure 19-1. Cyclic Redundancy Check (CRC) Module Block Diagram

19.2 External Signal Description

There are no CRC signals that connect off chip.

19.3 Memory Map and Registers

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CRC_CRCH	R	0								CRCH							
		W																
1	CRC_CRCL	R	0								CRCL							
		W																
2	CRC_ TRANSPOSE	R	0								TRANSPOSE							
		W																

19.3.1 CRC High Register (CRC_CRCH)

Address: CRC_CRCH – F230h base + 0h offset = F230h

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

CRC_CRCH field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 CRCH	This is the high byte of the 16-bit CRC register. A write to CRCH will load the high byte of the initial 16-bit seed value directly into bits 15-8 of the shift register in the CRC generator. The CRC generator will then expect the low byte of the seed value to be written to CRCL and loaded directly into bits 7-0 of the shift register. Once both seed bytes written to CRCH:CRCL have been loaded into the CRC generator, and a byte of data has been written to CRCL, the shift register will begin shifting. A read of CRCH will read bits 15-8 of the current CRC calculation result directly out of the shift register in the CRC generator.

19.3.2 CRC Low Register (CRC_CRCL)

Address: CRC_CRCL – F230h base + 1h offset = F231h

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

CRC_CRCL field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 CRCL	This is the low byte of the 16-bit CRC register. Normally, a write to CRCL will cause the CRC generator to begin clocking through the 16-bit CRC generator. As a special case, if a write to CRCH has occurred previously, a subsequent write to CRCL will load the value in the register as the low byte of a 16-bit seed value directly into bits 7-0 of the shift register in the CRC generator. A read of CRCL will read bits 7-0 of the current CRC calculation result directly out of the shift register in the CRC generator.

19.3.3 CRC Transpose Register (CRC_TRANSPOSE)

Address: CRC_TRANSPOSE – F230h base + 2h offset = F232h

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

CRC_TRANSPOSE field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7–0 TRANSPOSE	This register is used to transpose data, converting from LSb to MSb (or vice-versa). The byte to be transposed should be first written to TRANSPOSE and then subsequent reads from TRANSPOSE will return the transposed value of the last written byte (bit 7 becomes bit 0, bit 6 becomes bit 1, and so forth).

19.4 Functional Description

To enable the CRC function, a write to the CRCH register will trigger the first half of the seed mechanism which will place the CRCH value directly into bits 15-8 of the CRC generator shift register. The CRC generator will then expect a write to CRCL to complete the seed mechanism.

As soon as the CRCL register is written to, its value will be loaded directly into bits 7-0 of the shift register, and the second half of the seed mechanism will be complete. This value in CRCH:CRCL will be the initial seed value in the CRC generator.

Now the first byte of the data on which the CRC calculation will be applied should be written to CRCL. This write after the completion of the seed mechanism will trigger the CRC module to begin the CRC checking process. The CRC generator will shift the bits in the CRCL register (MSB first) into the shift register of the generator. One Bus cycle after writing to CRCL all 8 bits have been shifted into the CRC generator, and then the result of the shifting, or the value currently in the shift register, can be read directly from CRCH:CRCL, and the next data byte to include in the CRC calculation can be written to the CRCL register.

This next byte will then also be shifted through the CRC generator's 16-bit shift register, and after the shifting has been completed, the result of this second calculation can be read directly from CRCH:CRCL.

After each byte has finished shifting, a new CRC result will appear in CRCH:CRCL, and an additional byte may be written to the CRCL register to be included within the CRC16-CCITT calculation. A new CRC result will appear in CRCH:CRCL each time 8-bits have been shifted into the shift register.

To start a new CRC calculation, write to CRCH, and the seed mechanism for a new CRC calculation will begin again.

19.4.1 ITU-T (CCITT) Recommendations and Expected CRC Results

The CRC polynomial 0x1021 ($x^{16} + x^{12} + x^5 + 1$) is popularly known as *CRC-CCITT* since it was initially proposed by the ITU-T (formerly CCITT) committee.

Although the ITU-T recommendations are very clear about the polynomial to be used, 0x1021, they accept variations in the way the polynomial is implemented:

- ITU-T V.41 implements the same circuit shown in the block diagram, but it recommends a SEED = 0x0000.
- ITU-T T.30 and ITU-T X.25 implement the same circuit shown in the block diagram, but they recommend the final CRC result to be negated (one-complement operation). Also, they recommend a SEED = 0xFFFF.

Moreover, it is common to find circuits in literature slightly different from the one suggested by the recommendations above, but also known as CRC-CCITT circuits (many variations require the message to be augmented with zeros).

The circuit implemented in the CRC module is exactly the one suggested by the ITU-T V.41 recommendation, with an added flexibility of a programmable SEED. As in ITU-T V.41, no augmentation is needed and the CRC result is not negated. The following table shows some expected results that can be used as a reference. Notice that these are ASCII string messages. For example, message "123456789" is encoded with bytes 0x31 to 0x39 (see ASCII table).

Table 19-5. Expected CRC results

ASCII String Message	SEED (initial CRC value)	CRC result
"A"	0x0000	**0x58e5**
"A"	0xffff	**0xb915**
"A"	0x1d0f[1]	**0x9479**
"123456789"	0x0000	**0x31c3**
"123456789"	0xffff	**0x29b1**
"123456789"	0x1d0f[1]	**0xe5cc**
A string of 256 upper case "A" characters with no line breaks	0x0000	**0xabe3**
A string of 256 upper case "A" characters with no line breaks	0xffff	**0xea0b**
A string of 256 upper case "A" characters with no line breaks	0x1d0f[1]	**0xe938**

1. One common variation of CRC-CCITT require the message to be augmented with zeros and a SEED=0xFFFF. The CRC module will give the same results of this alternative implementation when SEED=0x1D0F and no message augmentation.

19.4.2 Transpose feature

The CRC module provides an phadditional feature to transpose data (invert bit order). This feature is specially useful on applications where the LSb format is used, since the CRC CCITT expects data in the MSb format. In that case, before writing new data bytes to CRCL, these bytes should be transposed as follows:

1. Write data byte to TRANSPOSE register

2. Read data from TRANSPOSE register (subsequent reads will result in the transposed value of the last written data)
3. Write transposed byte to CRCL.

After all data is fed into CRC, the CRCH:CRCL result is available in the MSb format. Then, these two bytes should also be transposed: the values read from CRCH:CRCL should be written/read to/from the TRANSPOSE register.

Although the transpose feature was initially designed to address LSb applications interfacing with the CRC module, it is important to notice that this feature is not necessarily tied to CRC applications. Since it was designed as an independent register, any application should be able to transpose data by writing/reading to/from the TRANSPOSE register (e.g.: Big endian / Little endian conversion in USB).

19.5 Initialization Information

To initialize the CRC Module and initiate a CRC16-CCITT calculation, follow this procedure:

1. Write high byte of initial seed value to CRCH.
2. Write low byte of initial seed value to CRCL.
3. Write first byte of data on which CRC is to be calculated to CRCL.
4. In the next bus cycle after step 3, if desired, the CRC result from the first byte can be read from CRCH:CRCL.
5. Repeat steps 3-4 until the end of all data to be checked.

Chapter 20
Flash Memory (HFM)

20.1 Overview

The embedded flash memory module is a nonvolatile memory module that operates with the program and IP buses.

The programming voltage required to program and erase the flash memory is generated internally by charge pumps. Program and erase operations are performed by a command-driven interface from the DSC core using an internal state machine. It is not possible to read or fetch from a block while it is being programmed or erased.

20.1.1 Features

- Up to 64 KB (32K words) of program flash memory
- 60 MHz (word access) operation for all flash access.
- Automated program and erase operation
- Interrupts on command completion, command buffer empty and access error.
- Fast page erase capability to erase only 2 KB
- Security feature to prevent unauthorized reads of flash contents
- Programmable sector protection feature to protect flash from being erased and programmed unexpectedly
- The flash memory supports byte and word read operations by the DSC core
- Code integrity check using built-in data signature calculation
- In-application programming and in-circuit programming capabilities

20.1.2 Block Diagram

The embedded flash memory module (HFM) consists of a 64 KB flash array and a common set of user-accessible registers, which control the sequence of programming or erasing the flash array and set write protection and flash security. The erase operation supports the mass erasure of an entire block and a page erase to erase only 2048 bytes. An erased bit reads 1 and a programmed bit reads 0. The following figure shows the architecture of the embedded flash module.

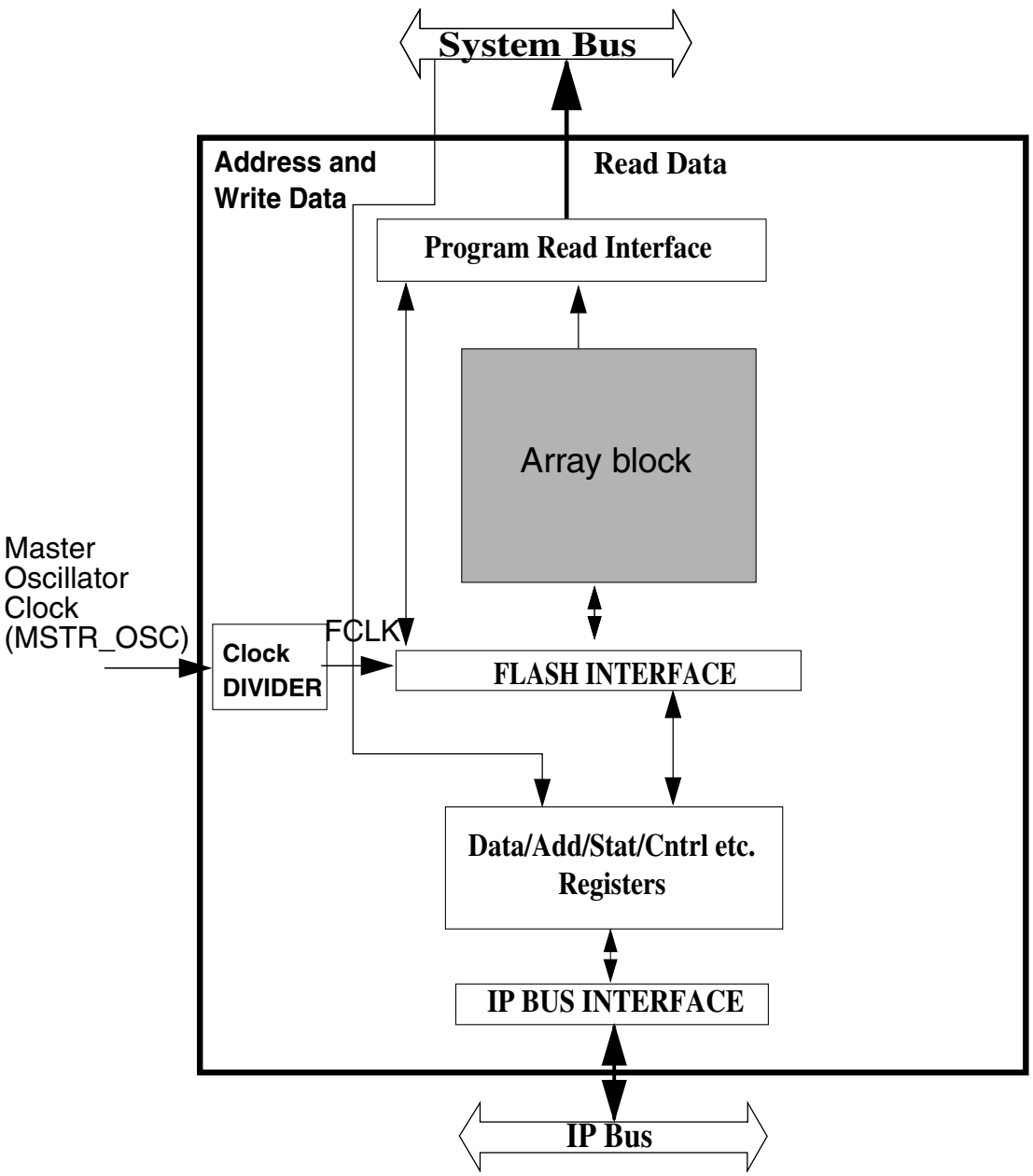


Figure 20-1. Embedded Flash Memory Module

20.1.3 Memory Array Organization

The embedded flash memory array consists of a flash configuration field of 9 words and a program field of 32,759 words. The array is located in program memory space from 0x0000 to 0x7FFF.

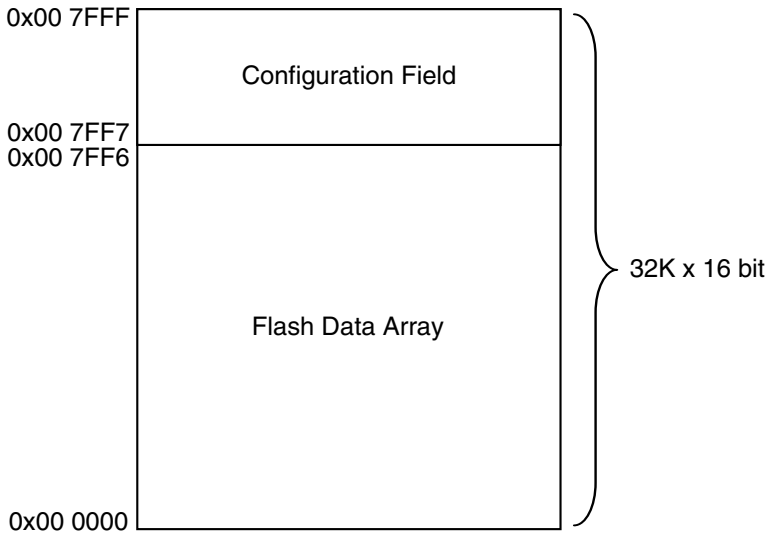


Figure 20-2. Flash Memory Array Organization

The flash configuration field contains the information determining the array's program/erase protection, flash security, and back-door access key. The protection prevents undesired flash writes and erasures. The flash security, if enabled, denies unauthorized access to internal flash contents.

The flash configuration field's arrangement appears in the following table.

Table 20-1. Flash Configuration Field Arrangement

Address	Size (Bytes)	Description	Word Name
0x7FFF	2	Backdoor Comparison Key 3	BACK_KEY_3_VALUE
0x7FFE	2	Backdoor Comparison Key 2	BACK_KEY_2_VALUE
0x7FFD	2	Backdoor Comparison Key 1	BACK_KEY_1_VALUE
0x7FFC	2	Backdoor Comparison Key 0	BACK_KEY_0_VALUE
0x7FFB	2	Not Used	Not Used
0x7FFA	2	Protection Word (see the description of the HFM protection register)	PROT_VALUE
0x7FF9	2	Not Used	Not Used

Table continues on the next page...

Table 20-1. Flash Configuration Field Arrangement (continued)

Address	Size (Bytes)	Description	Word Name
0x7FF8	2	Security Word upper (see the description of the HFM security register high)	SECH_VALUE
0x7FF7	2	Security Word lower (see the description of the HFM security register low)	SECL_VALUE

The top four words contain the back-door access key, which the user can program. To unsecure the device, the correct key must be written to this four-word location. Refer to Back Door Access.

One word contains flash program/erase protection (PROT_VALUE) loaded into the flash protection register upon device reset. The entire flash array is divided into 16 sections. Each bit of the HFM protection register protects one of the sections. The user can modify the protection register to protect/unprotect the sections of the flash array.

The security words (SECH_VALUE and SECL_VALUE) are loaded into the security registers high and low upon reset. The location 0x7FF8 (SECH_VALUE) contains user-programmed information to enable/disable the back-door access scheme. Programming 1 and 0 into location 0x7FF7 (SECL_VALUE), bits 1 and 0, will secure the device. If other bit-value combinations are programmed into these two bits, the device will be unsecured after reset.

20.2 Memory Map and Registers

NOTE

When "w1c" appears in the write row for a bit, it means "write 1 to clear": you must write 1 to the bit to clear it.

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	HFM_CLKD	R	0								DIVLD	PRDIV8	DIV[5:0]					
		W																
1	HFM_CR	R	0						LOCK	0	AEIE	CBEIE	CCIE	KEYACC	0		LBTS	BTS
		W																
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Address offset (hex)	Register name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3	HFM_SECH	R	KEYEN	SECSTAT	0													
		W																
4	HFM_SECL	R	0														SEC	
		W																
10	HFM_PROT	R	PROTECT[15:0]															
		W																
13	HFM_USTAT	R	0								CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	
		W									w1c		w1c	w1c		w1c		
14	HFM_CMD	R	0								CMD							
		W																
18	HFM_DATA	R	HFMDATA															
		W																
1A	HFM_IFR_OPT0	R	RESERVED								IFR_OPT0							
		W																
1D	HFM_TST_SIG	R	TST_AREA_SIG															
		W																

20.2.1 HFM Clock Divider Register (HFM_CLKD)

This register controls the period of the F_{CLK} clock used for timed events in program and erase algorithms within the flash module. While the flash module operates at the system bus's frequency, F_{CLK} must operate in the range between 150 kHz and 200 kHz. F_{CLK} is generated by dividing the master oscillator clock (MSTR_OSC; refer to the OCCS chapter) by a prescaler (determined by the PRDIV8 bit) and a divider (determined by the DIV[5:0] field). F_{CLK} does not vary if the system clock is changed.

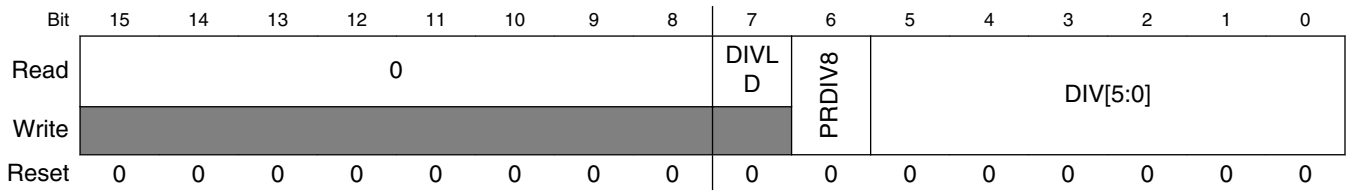
The PRDIV8 and DIV[5:0] fields must be set with appropriate values before programming or erasing the flash array.

NOTE

System bus clock frequencies below 1 MHz are not supported by the flash block for program and erase operations.

All bits in this register are readable and writable except bit 7, which is a read-only status bit.

Address: HFM_CLKD – F400h base + 0h offset = F400h



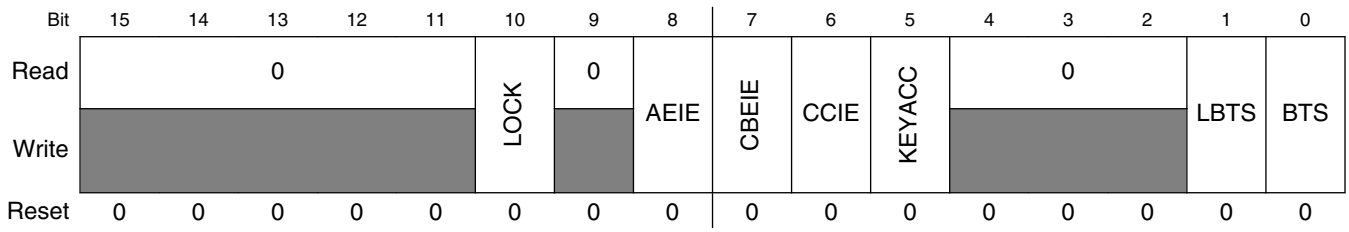
HFM_CLKD field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 DIVLD	Clock Divider Loaded 0 Register has not been written 1 Register has been written to since the last reset
6 PRDIV8	Enable Prescaler by 8 0 Prescaler divides master oscillator clock by 1 1 Prescaler divides master oscillator clock by 8
5–0 DIV[5:0]	Clock Divider The value of this field contributes to the division of the master oscillator clock (MSTR_OSC) to derive F_{CLK}. F_{CLK} is calculated as follows: • If PRDIV8 = 0: F_{CLK} = MSTR_OSC / (DIV[5:0] + 1) • If PRDIV8 = 1: F_{CLK} = MSTR_OSC / 8 / (DIV[5:0] + 1) **NOTE:** 1. Because F_{CLK} is re-timed into the system clock domain, the allowed value ranges for PRDIV8 and DIV[5:0] depend on the system bus frequency as well. 2. The flash clock setting must fulfill the following two requirements to enable proper programming of the embedded flash: • Avoid the case where: ((1 / F_{BUS}[MHz]) - (1 / F_{BUS}[MHz] / 4)) < 5 µs F_{BUS} is the system bus clock frequency, whose maximum value is 60 MHz. • Ensure that 150 kHz < F_{CLK} < 200 kHz. 3. With an F_{BUS} value below 1 MHz, erasing and writing flash are not possible, but reading flash is still supported.

20.2.2 HFM Configuration Register (HFM_CR)

This register controls the operation of the flash memory array.

Address: HFM_CR – F400h base + 1h offset = F401h



HFM_CR field descriptions

Field	Description
15–11 Reserved	This read-only bitfield is reserved and always has the value zero.
10 LOCK	Write Lock Control This bit is always readable. It may be set only once. When it is set, it cannot be cleared except by reset. This bit provides additional security for the flash array by disabling writes to the protection register. **NOTE:** When other bits in this register are written, this bit might also be written accidentally. Be careful not to set this bit accidentally when modifying other bits. 0 The PROT register is writable. 1 The PROT register is write-locked.
9 Reserved	This read-only bit is reserved and always has the value zero.
8 AEIE	Access Error Interrupt Enable This bit enables an interrupt when an ACCERR flag is set. 0 ACCERR interrupts disabled. 1 An interrupt is requested when the ACCERR flag is set.
7 CBEIE	Command Buffer Empty Interrupt Enable This bit enables an interrupt when there is an empty command buffer in the flash memory. 0 Command Buffer Empty interrupts disabled. 1 An interrupt is requested when the CBEIF flag is set.
6 CCIE	Command Complete Interrupt Enable This bit enables an interrupt when all commands are complete in the Flash. 0 Command Complete Interrupts Disabled. 1 An interrupt is requested when the CCIF flag is set.
5 KEYACC	Enable Security Key Writing This bit is always readable. However, it can be written only if the KEYEN bit in the SECH register is set. 0 Flash writes are interpreted as the start of a program or erase sequence. 1 Writes to Flash array are interpreted as keys to open the back door.
4–2 Reserved	This read-only bitfield is reserved and always has the value zero.

Table continues on the next page...

HFM_CR field descriptions (continued)

Field	Description
1 LBTS	BTS Lock Control This bit is always readable. It may be set only once. When it is set, it cannot be cleared except by reset. This bit provides additional security for the flash array by disabling writes to the BTS bit. **NOTE:** When other bits in this register are written, this bit might also be written accidentally. Be careful not to set this bit accidentally when modifying other bits. 0 The BTS bit is writable. 1 The BTS bit is write-locked.
0 BTS	Enable Branch-To-Self Feature This feature allows flash program/erase code to execute from flash. From the time the command is launched, the value of 0xA97F, which is the opcode of the instruction "Loop: BRA Loop," is available to be read on the data bus. When the program or erase command ends, the user data is sent immediately on the read data bus. **NOTE:** The Branch-to-Self feature should be enabled only during controlled program/erase sequences. It should be disabled during normal operation. Failure to do so can result in an inability to recover from or detect an illegal access. When the HFM begins its program/erase sequence, it no longer returns reads from flash memory but instead returns A97Fh, which is a Branch-to-Self instruction. All interrupts to the CPU must be disabled before the BTS is set and remain disabled until BTS is cleared. Attempting an interrupt while the BTS feature is engaged corrupts the stack. Because the processor is currently doing no active calculations (due to the BTS), it simply runs in place until the flash array is available. To enable the Branch To Self, you must first unlock the BTS bit by loading the BTS Lock Control (LBTS) with 0, using one of the methods previously described. The BTS feature works only with the program, page erase, and mass erase commands. 0 An access to the flash memory during program/erase returns invalid data and, meanwhile, the ACCERR flag is not set. 1 A read to the flash array when it is unavailable due to program/erase operations results in 0xA97F being placed on the data bus instead of actual flash data. The code 0xA97F corresponds to a BRANCH TO SELF instruction for the DSC core.

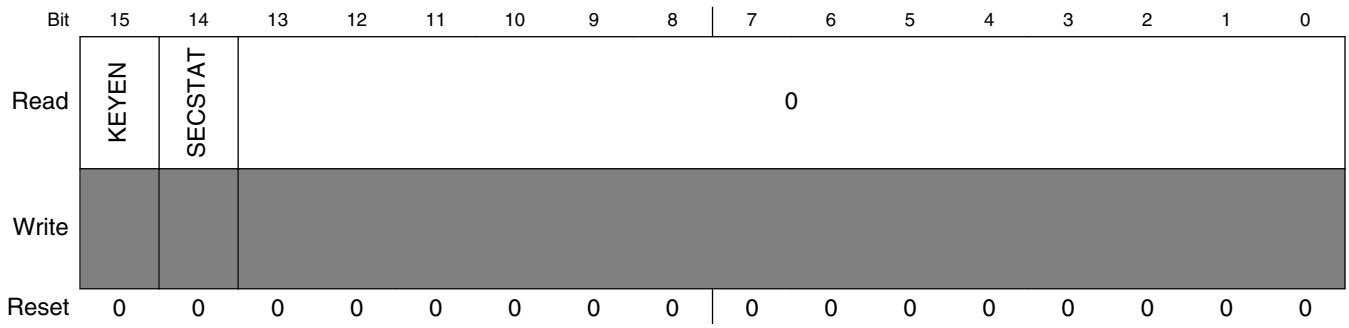
20.2.3 HFM Security Register High (HFM_SECH)

This read-only register stores flash security-related information. It is initialized at reset using the SECH_VALUE from the flash configuration field.

The reset state of the KEYEN bit is loaded from the flash array during reset.

The reset value of the SECSTAT bit is determined by the value loaded into the SEC field at reset. The value of SECSTAT can be modified at runtime.

Address: HFM_SECH – F400h base + 3h offset = F403h



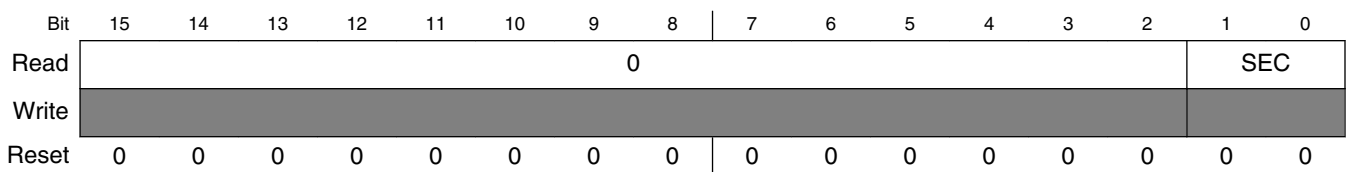
HFM_SECH field descriptions

Field	Description
15 KEYEN	Enable Back-Door Key to Security 0 Back door to flash memory is disabled. 1 Back door to Flash memory is enabled.
14 SECSTAT	Flash Security Status 0 Flash security is disabled. 1 Flash security is enabled.
13–0 Reserved	This read-only bitfield is reserved and always has the value zero.

20.2.4 HFM Security Register Low (HFM_SECL)

This read-only register stores flash security-related information. It is initialized at reset using the SECL_VALUE from the flash configuration field.

Address: HFM_SECL – F400h base + 4h offset = F404h



HFM_SECL field descriptions

Field	Description
15–2 Reserved	This read-only bitfield is reserved and always has the value zero.
1–0 SEC	Memory Security Bits

Table continues on the next page...

HFM_SECL field descriptions (continued)

Field	Description
	The value loaded into SEC from the configuration field at reset determines the state of flash security at reset. **CAUTION:** If security is enabled and you want to perform product analysis, you must either disable security using the back-door key or totally erase the array by performing either the lockout recovery sequence or a mass erase followed by a read-verify. 00 Flash unsecured (SECSTAT cleared at reset) 01 Flash unsecured (SECSTAT cleared at reset) 10 Flash secured (SECSTAT set at reset) 11 Flash unsecured (SECSTAT cleared at reset)

20.2.5 HFM Protection Register (HFM_PROT)

This register defines which flash array sectors are protected against program and erase.

The register is always readable. It is writable only when the LOCK bit in the CR register is cleared. The PROT register's bits corresponding to unimplemented sectors become reserved and always read 0.

This register's reset state is loaded from the flash array during reset.

Address: HFM_PROT – F400h base + 10h offset = F410h

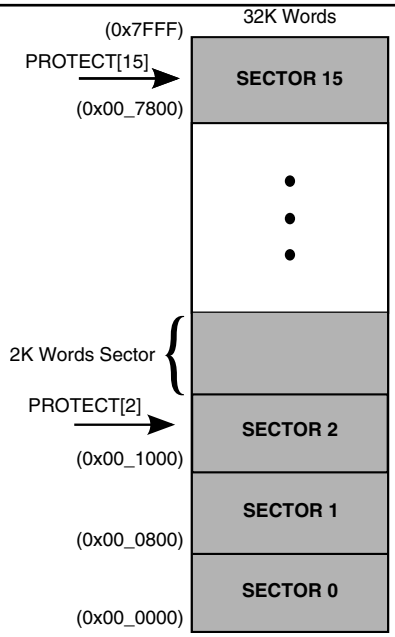
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PROTECT[15:0]															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

HFM_PROT field descriptions

Field	Description
15–0 PROTECT[15:0]	HFM Protection Each HFM array sector can be protected from program and erase by setting a PROTECT[index] bit. The PROT register controls the protection of 16 array sectors. The size of each array sector is 2K words (4 KB). Refer to the following figure.

HFM_PROT field descriptions (continued)

Field	Description
	 The protection scheme is operational only for the commands PGM, PGERS, and MASERS launched by array writes. PROTECT[index] = 0Indexed array sector is not protected PROTECT[index] = 1Indexed array sector is protected

20.2.6 HFM User Status Register (HFM_USTAT)

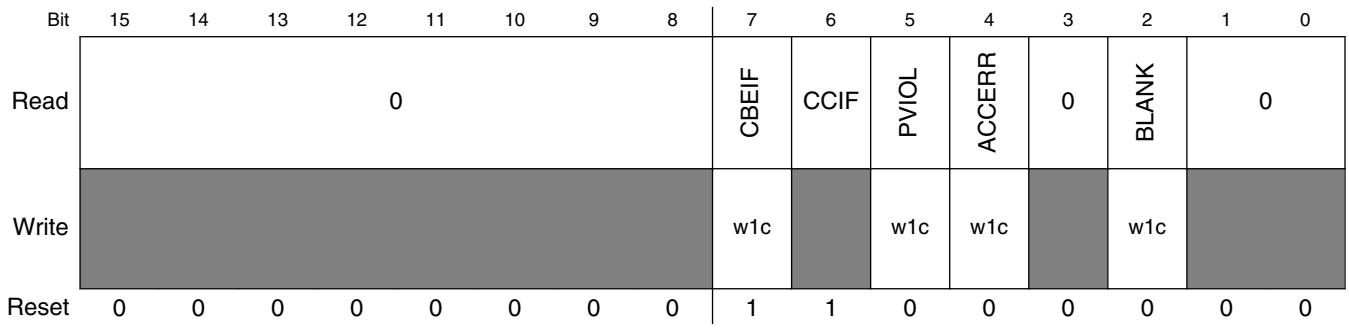
This register defines the flash state machine command status and flash array access, protection, and blank verify status.

USTAT register bits 7, 5, 4, and 2 are readable and writable; bits 3, 1, and 0 read zero and are not writable. Bit 6 is read-only.

NOTE

Only one bit in this register should be cleared at a time.

Address: HFM_USTAT – F400h base + 13h offset = F413h



HFM_USTAT field descriptions

Field	Description
15–8 Reserved	This read-only bitfield is reserved and always has the value zero.
7 CBEIF	Command Buffer Empty Interrupt Flag The CBEIF flag indicates that the address, data, and command buffers are empty so that a new command sequence can be started. The CBEIF flag is cleared by writing a 1. Writing a 0 has no effect on CBEIF bit but can be used to abort a command sequence. The CBEIF bit can generate an interrupt if the CBEIE bit in the CR register is set. While the CBEIF flag is clear, the CMD and DATA registers are not writable. 0 Buffers are full. 1 Buffers are ready to accept a new command.
6 CCIF	Command Complete Interrupt Flag The CCIF flag indicates that there are no more commands pending. The CCIF flag is set and cleared automatically upon the start and completion of a command. Writing to the CCIF bit has no effect. The CCIF bit can generate an interrupt if the CCIE bit in the CR register is set. 0 Command in progress. 1 All commands are completed.
5 PVIOL	Protection Violation The PVIOL flag indicates an attempt was made to program or erase an address in a protected flash memory area. The PVIOL flag is cleared by writing a 1. Writing a 0 has no effect on the PVIOL bit. While the PVIOL flag is set, it is not possible to launch another command. 0 No failure. 1 A protection violation has occurred.
4 ACCERR	Access Error The ACCERR flag indicates the occurrence of an illegal access to the flash array or registers, caused by a bad program or erase sequence. The ACCERR flag is cleared by writing a 1. Writing a 0 to the ACCERR bit has no effect. While the ACCERR flag is set, it is not possible to launch another command. See the description of Flash User Mode Illegal Operations for details on what sets the ACCERR flag. 0 No failure. 1 Access error has occurred.
3 Reserved	This read-only bit is reserved and always has the value zero.

Table continues on the next page...

HFM_USTAT field descriptions (continued)

Field	Description
2 BLANK	Flash Block Verified as Erased The BLANK flag indicates that an Erase Verify command (RDARY1) has checked the flash block and found it to be blank. The BLANK flag is cleared by writing a 1. BLANK is also automatically cleared by the start of any new command. Writing a 0 has no effect. 0 If an Erase Verify command has been requested, and the CCIF flag is set, then a zero in BLANK indicates that the block is not erased. 1 Flash block verifies as erased.
1–0 Reserved	This read-only bitfield is reserved and always has the value zero.

20.2.7 HFM Command Register (HFM_CMD)

This register defines the flash commands.

All of this register's bits are readable and writable except bit 7.

Address: HFM_CMD – F400h base + 14h offset = F414h

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

HFM_CMD field descriptions

Field	Description
15–7 Reserved	This read-only bitfield is reserved and always has the value zero.
6–0 CMD	Valid commands appear in the following table. Writing any other combination sets the ACCERR flag in the USTAT register.

Command	Name	Description
05h	RDARY1	Erase Verify (All Ones)
06h	RDARYM	Calculate Data Signature
20h	PGM	Word Program
40h	PGERS	Page Erase
41h	MASERS	Mass Erase
66h	RDARYMI	Calculate IFR Block Signature

Table continues on the next page...

HFM_CMD field descriptions (continued)

Field	Description
	For more detailed command descriptions, refer to the separate Flash Commands information.

20.2.8 HFM Data Register (HFM_DATA)

This register stores the result of the calculate flash array signature and calculate IFR block signature generated by RDARYM and RDARYMI commands.

The register is read-only.

Address: HFM_DATA – F400h base + 18h offset = F418h

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

HFM_DATA field descriptions

Field	Description
15–0 HFMDATA	HFM Data

20.2.9 HFM IFR Option Register 0 (HFM_IFR_OPT0)

Bits 9-0 of this read-only register store the trim value for the on-chip relaxation oscillator. At reset, the bits are loaded with a factory-programmed trim value stored in the flash information block.

The register's reset state is loaded from the flash array during reset.

Address: HFM_IFR_OPT0 – F400h base + 1Ah offset = F41Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RESERVED						IFR_OPT0									
Write																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

HFM_IFR_OPT0 field descriptions

Field	Description
15–10 Reserved	This bitfield is reserved.
9–0 IFR_OPT0	Trim value of on-chip relaxation oscillator

20.2.10 HFM Test Array Signature (HFM_TST_SIG)

This read-only register stores the flash information block signature that was generated by the RDARYMI command during factory test. At any time during the life of the part, execute the RDARYMI command and compare the result with the value in this register to confirm that the IFR data has not been compromised.

The register's reset state is loaded from the flash array during reset.

Address: HFM_TST_SIG – F400h base + 1Dh offset = F41Dh

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

* Notes:
- x = Undefined at reset.

HFM_TST_SIG field descriptions

Field	Description
15–0 TST_AREA_SIG	TST_AREA_SIG bits

20.3 Functional Description

The embedded flash memory module consists of a data array that is mapped into program memory space and a set of control and status registers that is mapped into data memory space. Data array read operations are formed by using 56800E program memory read instructions. The 56800E data-memory access instructions are used to access the control and status register set. Data array write and erase operations are formed by (in this sequence) using 56800E program-memory access instructions, writing a command to the command register, and writing a 1 to the CBEIF bit in the flash user status register.

20.3.1 Read Operation

The embedded flash memory module provides transparent read access to all memory locations in its flash data array. Each read is achieved without wait states, providing memory access without delays. Read operations execute through either instruction fetches or 56800E program memory read instructions.

20.3.2 Write Operation

Normal write access, where a value is written to a memory location within the program flash memory array, is part of a programming attempt. Writing the data must be followed by writing a program command to the HFM module's command register. Address and data in a 56800E program memory write instruction specifies the desired location and data to be programmed. To launch the module's state machine to process and effect the write:

1. Execute a program memory write instruction.
2. Write the program command code (0020h) to the command register.
3. Write 1 to the CBEIF bit.

The flash command register and address and data buffer operate as a two-stage FIFO. Consequently, a new command, along with the necessary data and address, can be stored in the buffer while the previous command is still in progress. This technique can be applied to all commands except RDARYM and RDARYMI. A new command can be buffered only if the buffer is empty, which is indicated when the CBEIF flag in the user status register reads 0; otherwise the attempt to launch the new command will be flagged as an error. This feature increases the rate at which the HFM state machine receives commands, reducing command processing time, and expedites program/erase cycles. Another flag in the flash user status register signals command completion. Interrupts are generated when they are enabled.

Erased flash consists of all ones, so any programming attempt writes only the required zeros to create the desired bit pattern. The user must verify the erased state of the destination in flash before programming it.

20.3.3 Erase Operation

To perform an erase operation:

1. Execute a 56800E program memory write instruction.
2. Write the erase command code (0040h or 0041h) to the command register.
3. Write 1 to the CBEIF bit to launch the erase process.

If the operation is to erase a page of flash, the address specified in the 56800E program memory write instruction must be any one of the addresses within the page to be erased. If the operation is a mass erase, the address specified in the 56800E program memory write instruction must be any one of the addresses within the data array. Data in the 56800E program memory write instruction is irrelevant for the erase to proceed.

20.3.4 Flash Commands

The following table summarizes the valid flash commands.

Table 20-13. Flash Commands

FM_CMD CMD	Meaning	Description	Array Write Address/Data Usage
0x05	Erase Verify	Verify that the flash array is erased. If the array is verified to be blank, the BLANK bit sets in the USTAT register upon command completion.	The address must be any address within the array. Data is ignored.
0x06	Calculate Data Signature	Calculate a signature over user-specified range of words in the flash array. The resulting signature is returned in the DATA register at the completion of the command. The new signature can be compared with a signature previously calculated by the user to verify program flash integrity.	The array write address specifies the starting address and data specifies the number of words to compress.
0x20	Program	Program a 16-bit word. Programming is only possible when the protection bit for the applicable sector is not set.	The address and data specify the address and value to program.
0x40	Page Erase	Erase a specific page (sector) of the flash array. A page erase is possible only when the PROTECT bit for the selected page is not set.	The address must be any address within the page to be erased. Data is ignored.
0x41	Mass Erase	Erase all flash memory. A mass erase is possible only when no PROTECT bits are set.	The address must be any address within the array. Data is ignored.

Table continues on the next page...

Table 20-13. Flash Commands (continued)

FM_CMD CMD	Meaning	Description	Array Write Address/Data Usage
0x66	Calculate IFR Block Signature	Calculate a signature over the flash Information Row (IFR) of the flash array. The resulting signature is returned in the DATA register at the completion of the command. The entire IFR row 1 is compressed except the last word containing the expected compression result. The flash information block contains data set by the factory for proper operation of the device. The IFR block signature calculated at the factory is available, for comparison to the new value, in the TST_SIG register. If the value in the DATA register differs from the value in the TST_SIG register, then flash programming parameters stored in the IFR block have been corrupted.	The address must be any address within the array. Data is ignored.

20.3.5 Set the Flash Program/Erase Clock (F_{CLK})

Commands that alter HFM content—programming and erasing—must be properly timed to ensure a successful update without damage. The programming and erasing algorithm is controlled by a state machine whose time base (F_{CLK}) is derived from the oscillator clock output (8 MHz typical) using a programmable prescaler and divider controlled by fields in the clock divider register. The resulting FCLK frequency is required to be between 150 kHz and 200 kHz. Program and erase command execution time increases proportionally with the period of F_{CLK}.

NOTE

Due to the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the flash memory cannot be performed while the system bus clock frequency is less than 1 MHz. However, reading the flash array is still supported.

After a reset and before issuing any flash command, the user must write the clock divider register.

CAUTION

Avoid programming or erasing the flash memory when F_{CLK} < 150 kHz. Setting the clock divider register to a value such that F_{CLK} < 150 kHz can damage the flash memory due to overstress. Setting FM_CLKDIV to a value such that (1/F_{CLK} -

$1/F_{BUS}/4) < 5$ µs, where F_{BUS} is the system bus clock frequency (maximum value 60 MHz), can result in incomplete programming or erasure of the flash memory cells.

If the clock divider register is written, the DIVLD bit is set automatically. If the DIVLD bit is zero, the FM_CLKDIV register has not been written since the last reset. If the clock divider register has not been written, the flash command loaded during a command write sequence does not execute and the ACCERR flag in the user status register sets.

The clock divider register bits can be modified even after their initialization.

20.3.6 Command Sequence

A command state machine supervises the command processing. To prepare for executing a command, test the CBEIF flag in the user status register to ensure that the address, data, and command buffers are empty. Failure to do so may cause the ACCERR bit to set. If the CBEIF flag is set, the command sequence can be started.

To successfully program flash, follow this command write sequence *exactly*. Intermediate writes to the HFM are not permitted between the steps.

NOTE

If the BTS bit in the configuration register is cleared, the command write sequence must execute from RAM.

1. Using a *program* memory write instruction, write the desired value to the address in flash memory that you wish to program.
2. Write the numeric code for the command to the command register, which is a buffered register.
3. To launch the command, clear the CBEIF flag by writing 1. After the CBEIF flag is cleared, hardware clears the CCIF flag in the user status register, indicating that the command was successfully launched. Then the CBEIF flag is set again, indicating that the address, data, and command buffers are ready for a new command write sequence to begin.

The completion of the command operation is indicated when the CCIF flag sets.

The command state machine flags errors in program or erase write sequences by means of the access error (ACCERR) and protection violation (PVIOL) flags in the HFM user status register. An erroneous command-write sequence aborts and sets the appropriate flag. If a flag is set, the user must clear the ACCERR or PVIOL flag before starting another command write sequence.

Note

By writing a 0 to the CBEIF flag the command sequence can be aborted after the word write to the flash address space or after writing a command to the HFMCMD register, and before the command is launched. The ACCERR flag will be set on aborted commands and must be cleared before a new command is launched.

Refer to the following flow chart of the entire command sequence.

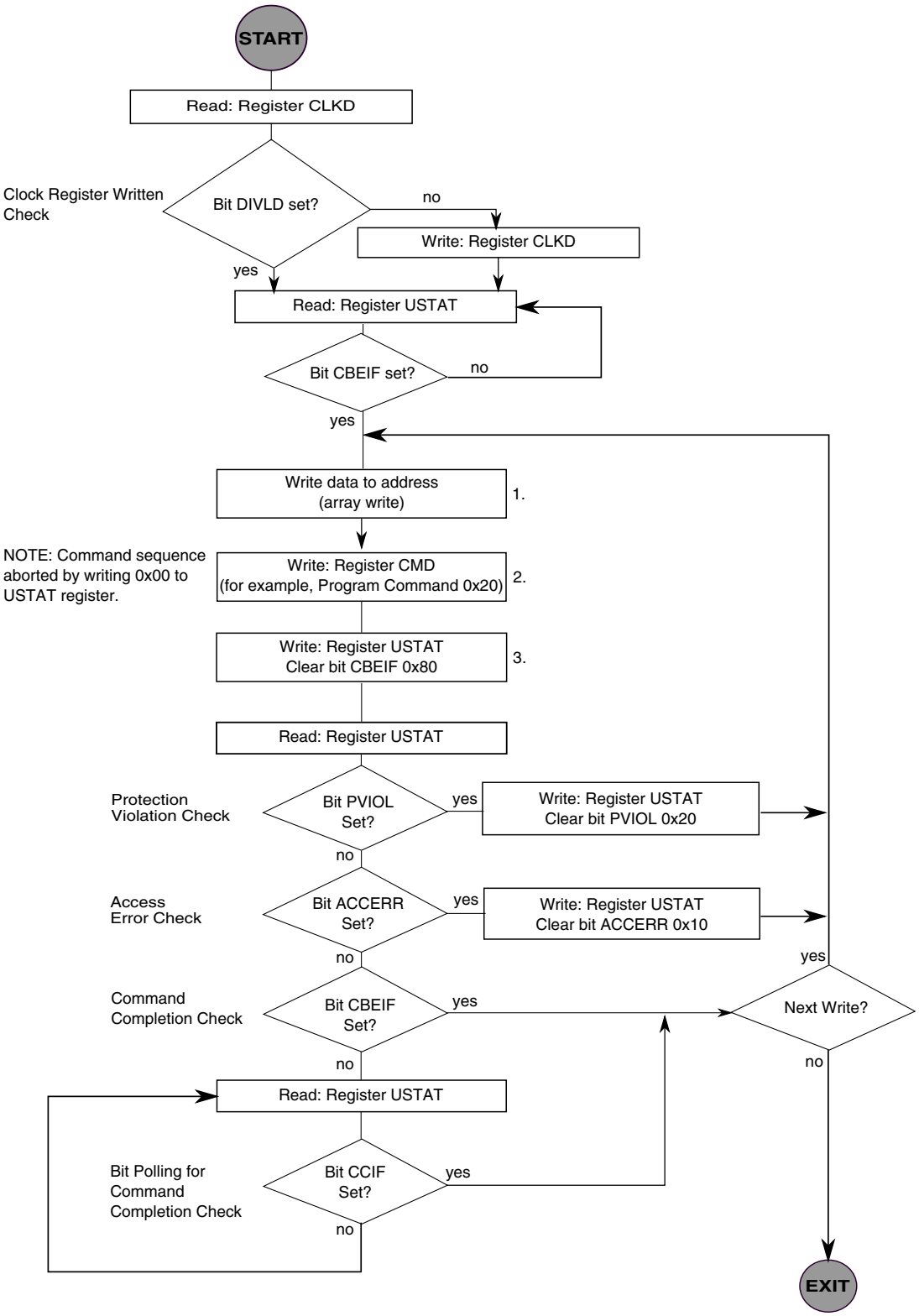


Figure 20-13. Command Sequence Flow Chart

20.3.7 Flash User Mode Illegal Operations

The ACCERR flag is set during the command write sequence if any of the following illegal operations are performed. Such operations cause the command sequence to abort immediately. Writes to the flash array address space occur through the CPU *program* memory write instructions and not through the register set accesses.

- Writing to the flash array address space before initializing FM_CLKDIV.
- Writing to the flash array address space while CBEIF is not set.
- Writing a second word to the flash array address space.
- Writing an invalid user command to the command register.
- Writing to any flash register other than the command register after writing a word to the flash array address space.
- Writing a second command to the command register before executing the previously written command.
- Writing to any flash register other than the user status register (to clear CBEIF) after writing to the command register.
- The device enters stop or wait mode and a command is in progress. The command is aborted.
- Aborting a command sequence by writing a 0 to the CBEIF flag after the word write to the flash address space or after writing a command to the command register and before the command is launched.
- Writing to the flash array while a calculate data signature command is running.

The PVIOL flag is set during the command-write sequence if any of the following illegal operations are performed. Such operations cause the command sequence to abort immediately.

- Writing a flash array address to program in a protected area.
- Writing a flash array address to erase in a protected area.
- Writing a mass erase command to the command register while any protection is enabled (see the description of the HFM protection register).

If a flash block is read during execution of an algorithm on that block (that is, CCIF is low), the read returns invalid data; meanwhile, the ACCERR flag does not set if the BTS bit = 0.

20.3.8 Calculate Data and IFR Block Signature Commands

The RDARYM command calculates data in the flash array into a checksum (signature). The RDARYMI command performs a similar function for the factory configuration data stored in the device.

The data calculation uses an MISR algorithm with the following polynomial:

$p(x) = x + x^2 + x^4 + x^{15}$

The command RDARYM uses the data and address within CPU *program* memory write instructions to specify the start address and data length.

If the length passed as an argument to calculate data signature command RDARYM is greater than the array size, addresses keep wrapping around the block limits until the length count is reached.

If the length passed as argument to calculate data signature command RDARYM is zero, the algorithm runs the maximum number of cycles that can be generated by the length counter. It happens regardless of the array maximum address, and it continues wrapping around the address limits.

The command RDARYMI has the start address and length set by default.

20.3.9 Effects of Wait and Stop Mode

If a command is active (CCIF=0) when the device enters wait or stop mode, the command is aborted and the data being programmed or erased is lost. A pending command (CBEIF=0) is not executed after the device exits wait or stop mode. The CCIF and ACCERR flags are set if a command is active when the device enters wait or stop mode.

When entering wait or stop mode, the flash module immediately (within the next clock period) sets the ACCERR flag, regardless of the operation being executed.

Note

Because active commands are immediately aborted when the device enters wait mode, it is strongly recommended that user software does not execute the WAIT instruction during program and erase execution.

20.3.10 Flash Security Operation

Flash security provides a means to protect the embedded code within the flash array from unauthorized external access. The state of flash security is reflected in the state of the SECSTAT bit in the security register high. The value of the SECSTAT bit at reset is determined by the values in the security words stored in the flash configuration field. Thus, by appropriately programming the configuration field, the part can be forced into secure mode at reset or power-up.

Flash security prevents unauthorized external access by the JTAG/EOnCE port. After flash security is set, an external user is unable to view or change embedded software and is thus unable to introduce code sequences to undo security or export code.

There are three methods of disabling flash security at run time:

1. Execute a back door access scheme built into the application.
2. Pass an erase-verify check.
3. Execute the JTAG lockout-recovery routine to mass erase the flash .

Only the first method preserves the contents of flash memory.

20.3.10.1 Set Flash Security

The user can secure the device by programming the security code 10b into the program memory location 0x00_7FF7. The nonvolatile security word ensures the device remains secure after a reset.

20.3.10.2 Back Door Access

The following directions explain how to disable flash security at run-time. This method is the only one for disabling security without the erasure of flash contents. The KEYEN bit in the security register high must be set to enable back-door key access.

1. Set the KEYACC bit in the configuration register.
2. Write the correct 4-word (64-bit) back-door comparison key to the back-door key locations (0x7FFC to 0x7FFF) of the flash memory configuration field. The four write cycles can be separated by any number of other operations.
3. Clear the KEYACC bit.

If all four words written match the flash content in the configuration field, security is bypassed until the next reset.

After the next reset sequence, the device is secured again.

The back-door method of unsecuring the device has no effect on the program and erase protections defined in the protection register.

NOTE

The user application must execute the back-door access that disables flash security at run-time, according to the proper access sequence provided in the preceding directions. Otherwise, the flash security is not opened, and therefore the code after this access operation will not be executed.

20.3.10.3 JTAG Lockout Recovery

To unsecure a secured device, mass erase the flash memory by using a sequence of JTAG commands. On the next reset, the part will be unsecured.

See the information about security features in the device's data sheet.

20.4 Resets

The flash module responds to a system reset.

20.5 Interrupts

20.5.1 General

The flash module can generate an interrupt when all flash commands are completed, when the data and command buffers are empty, or when an access error occurs.

20.5.2 Interrupt Operation

The following figure outlines the operation of the interrupt request generated by the flash module. The flash module interrupt request lines are connected to the Interrupt Controller. Software can also be used to poll the CCIF, CBEIF, and ACCERR flags.

This system uses the CBEIE, CCIE, and AEIE bits in the configuration register to enable interrupt generation.

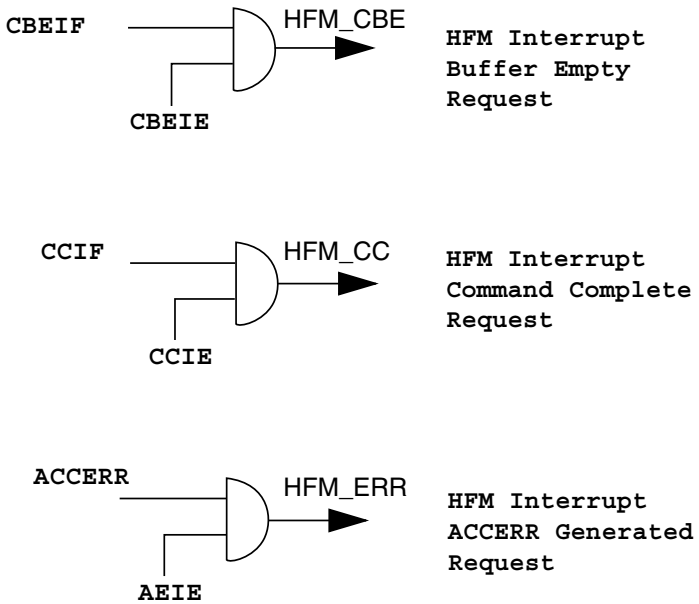


Figure 20-14. HFM Interrupt Implementation

Chapter 21
Joint Test Action Group (JTAG) Port

21.1 Introduction

Because this device also has a DSC core containing its own test access port (TAP), or core TAP, a TAP linking module (TLM) is included to manage the TAP access. Normal operation of this device uses the device TAP as the master TAP controller, thereby disabling the core TAPcontroller. This chapter discusses the master TAP only.

21.1.1 Features

TAP characteristics include:

- Access the EOnCE module controller and circuits to control a target system.
- Query the IDCODE from any TAP in the system.
- Force test data onto the peripheral outputs while replacing its Boundary Scan Register (BSR) with a single bit register.
- Enable/Disable pullup devices on peripheral boundary scan pins.

21.1.2 Block Diagram

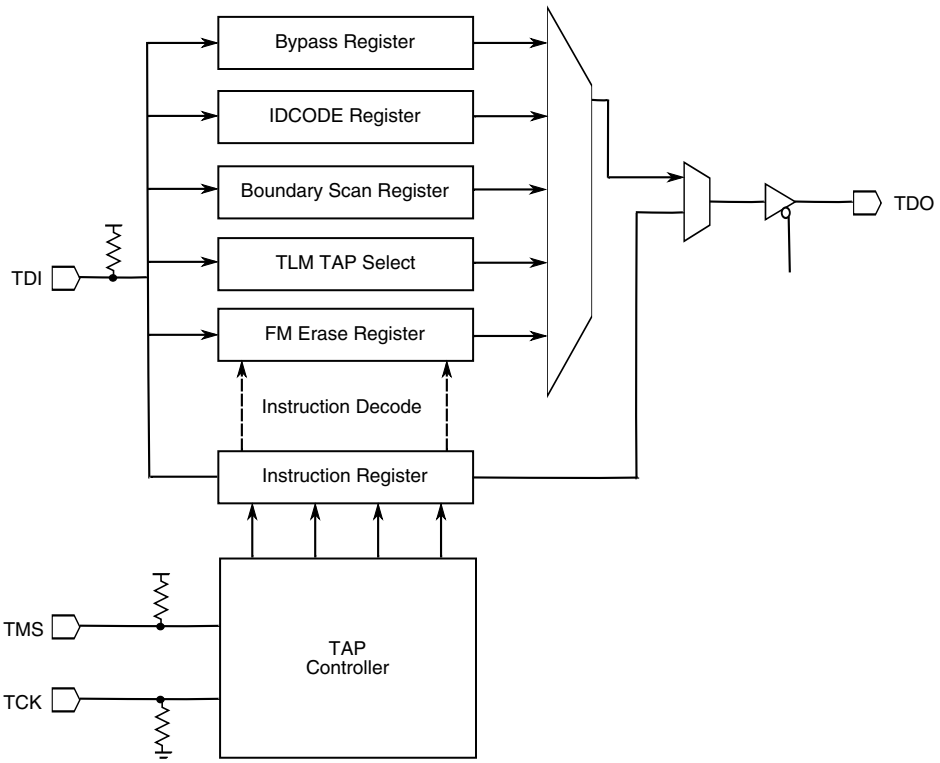


Figure 21-1. JTAG Block Diagram

21.2 External Signal Description

The following table summarizes the JTAG signals.

Table 21-1. JTAG Pin Descriptions

Pin	Description
TCK	Test Clock Input — This input pin provides the clock to synchronize the test logic and shift serial data to and from all TAP controllers and the TLM. If the EOnCE module is not being accessed using the master or 56800E core TAP controllers, the maximum TCK frequency is 1/4 the maximum frequency for the 56800E core. When accessing the EOnCE module through the 56800E core TAP controller, the maximum frequency for TCK is 1/8 the maximum frequency for the 56800E core. The TCK pin has a pulldown non-disabled resistor.
TDI	Test Data Input — This input pin provides a serial input data stream to the TAP and the TLM. It is sampled on the rising edge of TCK. TDI has an on-chip pullup resistor that can be disabled through PUPEN register in the GPIO module.
TMS	Test Mode Select Input — This input pin is used to sequence the TAP controller's TLM state machine. It is sampled on the rising edge of TCK. TMS has an on-chip pullup resistor that can be disabled through PUPEN register in the GPIO module. Note: Always tie the TMS pin to VDD through a 2.2K resistor.
TDO	Test Data Output — This three-state output pin provides a serial output data stream from the master TAP, or 56800E core TAP controller. It is driven in the shift-IR and shift-DR controller states of the TAP controller state machines. Output data changes on the falling edge of TCK.

21.3 Memory Map

JTAG has no memory-mapped registers.

21.4 Functional Description

The master TAP consists of a synchronous finite 16-bit state machine, an eight-bit instruction register, a bypass register, and an identification code register.

21.4.1 JTAG Port Architecture

The TAP controller is a simple state machine used to sequence the JTAG port through its varied operations:

- Serially shift in or out a JTAG port command.
- Update and decode the JTAG port Instruction Register (IR).
- Serially input or output a data value.
- Update a JTAG port or EOnCE module register.

NOTE

The JTAG port supervises the shifting of data into and out of the EOnCE module through the TDI and TDO pins, respectively. In this case, the shifting is guided by the same controller used when shifting JTAG information.

A block diagram of the JTAG port is provided in Figure 20-1. The JTAG port has four read/write registers:

- Instruction Register (JTAGIR)
- Chip Identification (CID) register
- Bypass Register (JTAGBR)
- Boundary Scan Register (BSR)

Access to the EOnCE registers is described in the device data sheet.

21.4.2 Master TAP Instructions

The eight-bit master TAP Instruction register is in support of all JTAG functions. It is described in Table 20-1. This register includes all IEEE 1149.1 required instructions plus several additional instruction registers accommodating debug and BIST testing.

Table 21-2. Master TAP Instruction Opcodes

Instruction	Target Register	Opcode
BYPASS	BYPASS	11111111
IDCODE	IDCODE	00000010
—	Reserved	00000011
—	Reserved	00000100
TLM_SEL	TLM	00000101
Lock Out Recovery (Flash_Erase)	FLASH_ERASE	00001000

21.4.2.1 Bypass Instruction (BYPASS)

The BYPASS instruction is a required JTAG instruction, selecting the TAP Bypass register. This register is a single stage shift register providing a serial path between the TDI and the TDO pins illustrated in Figure 20-2. This instruction enhances test efficiency by shortening the overall path between TDI and TDO when no test operation of a component is required.

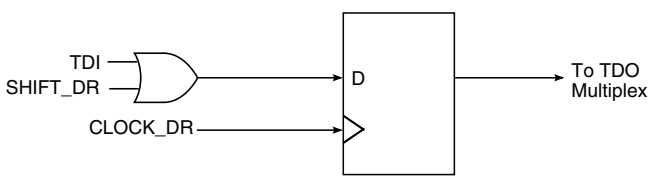


Figure 21-2. Bypass Register Diagram

21.4.2.2 IDCODE

The IDCODE instruction is an optional JTAG instruction enabling the Chip Identification (CID) register between TDI and TDO. This 32-bit register identifies the manufacturer, part, and version numbers.

21.4.2.3 TLM_SEL

The TLM_SEL instruction is a user-defined JTAG instruction, disabling the master TAP and enabling the TAP linking module, or TLM. The TLM then selects the DSC core TAP or the master TAP as the enabled TAP.

21.4.2.4 TAP Controller

The TAP controller is a synchronous 16-bit finite state machine illustrated in the following figure. The TAP controller responds to changes at the TMS and TCK pins. Transitions from one state to another occur on the rising edge of TCK. The value shown adjacent to each state transition represents the signal present on TMS at the time of a rising edge of TCK.

The TDO pin remains in the high impedance state except during the shift-DR and shift-IR TAP controller states. In shift-DR and shift-IR controller states, TDO updates on the falling edge of TCK. TDI is sampled on the rising edge of TCK.

The TAP controller executes the last instruction decoded until a new instruction is entered at the update-IR state, or test-logic-reset is entered.

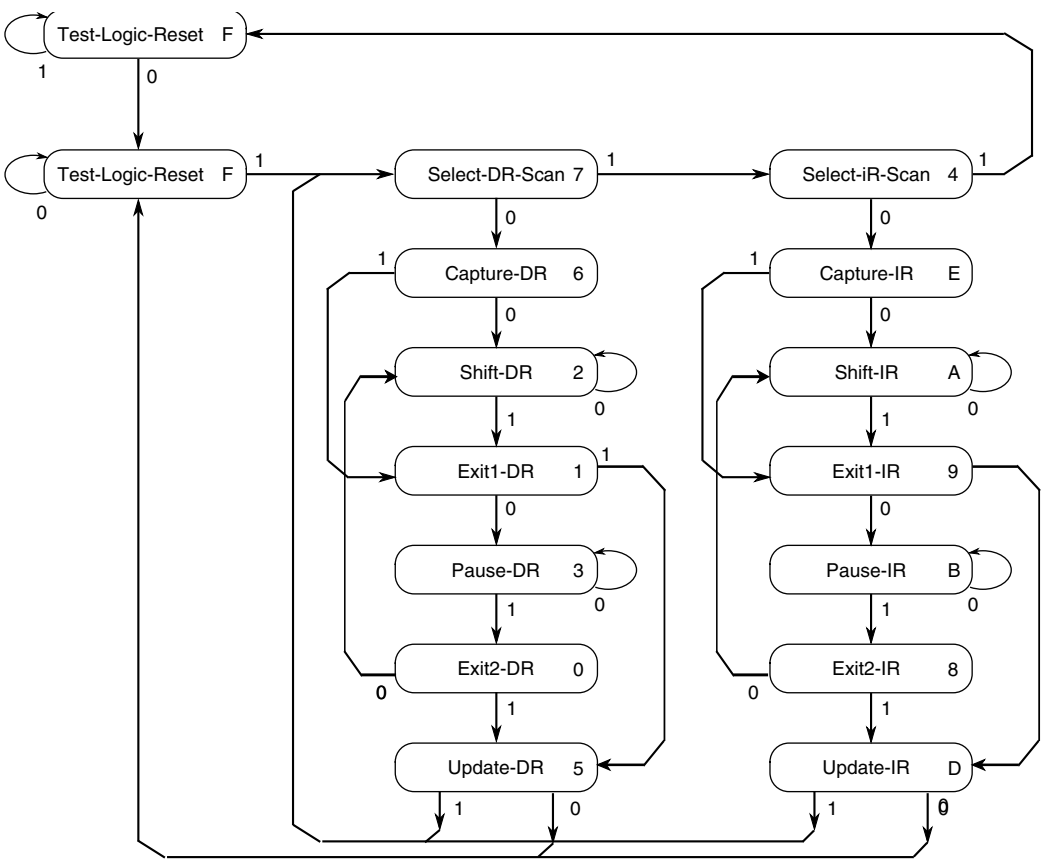


Figure 21-3. TAP Controller State Diagram

All state transitions of the TAP controller occur based on the value of TMS at the time of a rising edge of TCK. Actions of the instructions occur on the falling edge of TCK in each controller state illustrated in the TAP controller state diagram. The following table lists the and descibes the JTAG states.

Table 21-3. JTAG States

State	Description
Test Logic Reset (pstate=F)	During test-logic-reset, all JTAG test logic is disabled so the device can operate in normal mode. This is achieved by initializing the instruction register (IR) with the IDCODE instruction. By holding TMS high for five rising edges of TCK, the device always remains in test-logic-reset no matter what state the TAP controller was in previously.
Run-Test-Idle (pstate-C)	Run-test-idle is a controller state between scan operations. When EOnCE is entered, the controller remains in run-test-idle mode as long as TMS is held low. When TMS is high and a rising edge of TCK occurs, the controller moves to the select-DR state.
Select Data Register (pstate-7)	The select-DR state is a temporary state in which all test data registers selected by the current instruction retain their previous states. If TMS is held low and a rising edge of TCK occurs when the controller is in this state, the controller moves into the capture-DR state and a scan sequence for the selected test date register is initiated. If TMS is held high and a rising edge of TCK occurs, the controller moves to the select-IR state.
Select Instruction Register (pstate=4)	The select-IR state is a temporary state in which all test data registers selected by the current instruction retain their previous states. If TMS is held low and a rising edge of TCK occurs when the controller is in this state, the controller moves into the capture-IR state and a scan sequence for the instruction register is initiated. If TMS is held high and a rising edge of TCK occurs, the controller moves to the test-logic-reset state.
Capture Data Register (pstate=6)	In this controller state, data can be loaded in parallel into test registers selected by the current instruction on the rising edge of TCK. If a test data register selected by the current instruction does not have a parallel input, the register retains its previous value.
Shift Data Register (pstate=1)	In this controller state, the test data register is connected between TDI and TDO. This data is then shifted one stage towards its serial output on each rising edge of TCK. The TAP controller remains in this state while TMS is held at low. When 1 is applied to TMS and a positive edge of TCK occurs, the controller moves to the exit1-DR state.
Exit 1 Data Register (pstate=1)	This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while in this state, the controller advances to the update-DR state. This terminates the scanning process.
Pause Data Register (pstate=3)	This controller state permits shifting of the test data register in the serial path between TDI and TDO to be temporarily halted. All test data registers selected by the current instruction retain their previous state unchanged. The controller remains in this state while TMS is held low. When TMS goes high and a rising edge is applied to TCK, the controller advances to the exit2-DR state.
Exit2 Data Register (pstate=0)	This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while it is in this state, the scanning process terminates and the TAP controller advances to the update-DR state. If TMS is held low and a rising edge of TCK occurs, the controller advances to the shift-DR state.
Update Data Register (pstate=5)	All boundary scan registers contain a two-stage data register. It isolates the shifting and capturing of data on the peripheral from what is applied to internal logic during scan mode. This register is the second stage, or parallel output, and applies a stimulus to internal logic. Data is latched on the parallel output of these test data registers from the shift register path on the falling edge of TCK in the update-DR state. On a rising edge of TCK, the controller advances to the select_dr state if TMS is held high or the run-test-idle state if TMS is held low.

Table continues on the next page...

Table 21-3. JTAG States (continued)

State	Description
Capture Instruction Register (pstate = E)	When the TAP controller is in this state and a rising edge of TCK occurs, the controller advances to the exit1-IR state if TMS is held at one, or the shift-IR state if TMS is held at zero.
Shift Instruction Register (pstate = A)	In this controller state, the shift register contained in the instruction register is connected between TDI and TDO and shifts data one stage toward its serial output on each rising edge of TCK. When the TAP controller is in this state and a rising edge of TCK occurs, the controller advances to the exit1-IR state if TMS is held at one or remains in the shift-IR state if TMS is held at zero.
Exit1 Instruction Register (pstate = 9)	This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while in this state, the controller advances to the update-IR state. This terminates the scanning process. If TMS is held low and a rising edge of TCK occurs the controller advances to the pause-IR state.
Pause Instruction Register (pstate = B)	This controller state allows shifting of the instruction register in the serial path between TDI and TDO to be temporarily halted. All test data registers selected by the current instruction retain their previous state unchanged. The controller remains in this state while TMS is held low. When TMS goes high and a rising edge is applied to TCK, the controller advances to the exit2-IR state.
Exit2 Instruction Register (pstate = 8)	This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while in this state, the scanning process terminates and the TAP controller advances to the update-IR state. If TMS is held low and a rising edge of TCK occurs, the controller advances to the shift-IR state.
Update Instruction Register (pstate = D)	During this state, the instruction shifted into the instruction register is latched from the shift register path on the falling edge of TCK and into the instruction latch. It becomes the current instruction. On a rising edge of TCK, the controller advances to the selector state if TMS is held high, or the run-test-idle state if TMS is held low.

21.5 Clocks

TCK is the sole clock used by the master TAP module. If the EOnCE module is not being accessed using the master or DSC core TAP controllers, the maximum TCK frequency is one-quarter the maximum frequency for the DSC core. When the EOnCE module is accessed through the DSC core TAP controller, the maximum frequency for TCK is one eighth the maximum frequency for the DSC core.

Table 21-4. JTAG Clock Summary

Clock	Priority	Source	Characteristics
TCK	1	External	This user-provided clock shifts data and controls the state machine.

21.6 Interrupts

The JTAG module has no interrupt capabilities.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: MC56F825XRM
Rev. 2, 10/2010
Preliminary