MC56F80XXRM
Rev. 3.1, 03/2012

56F802X and 56F803X Peripheral Reference Manual

by: Microcontroller Solutions Group

This is the 56F802X and 56F803X Peripheral Reference Manual set consisting of the following files:

- 56F802X and 56F803X Peripheral Reference Manual Addendum, Rev 2
- 56F802X and 56F803X Peripheral Reference Manual, Rev 3

Freescale Semiconductor
Reference Manual Addendum

MC56F80XXRMAD
Rev. 2, 01/2012

56F802X and 56F803X Peripheral Reference Manual Addendum

by: Microcontroller Solutions Group

This errata document describes corrections to the *56F802X and 56F803X Peripheral Reference Manual*, order number MC56F80XXRM. For convenience, the addenda items are grouped by revision. Please check our website at http://www.freescale.com for the latest updates.

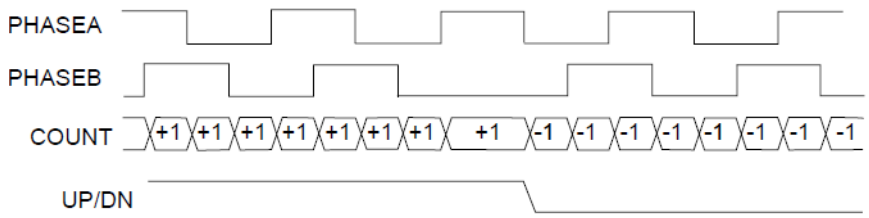
The current version available of the *56F802X and 56F803X Peripheral Reference Manual* is Revision 3.0.

Table of Contents

freescale™ semiconductor

1 Errata for Revision 3.0

Table 1. 56F802X and 56F803X Peripheral Reference Manual Rev 3.0 Errata

Location	Description
Section 2.7.14, “Power Control (PWR) Register”/Figure 2-31/Page 2-37	Change the reset value of the PWR Register (address 0xF0B4) to binary 0001 1100 1101 0111 = 0x1cd7.
Section 2.7.14.4, “Power Status 11 (PSTS1)—Bit 11”/Page 2-38	Change the heading of the section to “Power Status 1 (PSTS1)—Bit 11”.
Section 3.4.2, “COP After Reset”/Page 3-3	Update the text to “the COP is enabled and CTRL is set to 0x22.”
Section 3.5.1.3, “COP Loss of Reference Enable (CLOREN)—Bit 5”/Page 3-5	Change the default value to 1.
Section 8.7.14.9 “Source 2 (SRC2)Bits 6-5”/Page 8-39	Change “SCTRL0“ to “SRC0”.
Section 14.5.5 “Quadrature Count Mode”/Figure 14-3/Page 14-6	Update the timing diagram to the following. 

2 Revision History

Table 2 provides a revision history for this document.

Table 2. Revision History Table

Rev. Number	Substantive Changes	Date of Release
1.0	Not publicly released. Correct errors in the following chapters: • Chapter 2, “Analog Digital Converter (ADC)” • Chapter 3, “Computer Operating Properly (COP)” • Chapter 14, “Quad-Timer (TMR)”	09/2011
2.0	Added the timing diagram in the description column of the section 14.5.5.	01/2012

THIS PAGE IS INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
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

Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MC56F80XXRMAD
Rev. 2
January 2012

freescale™
semiconductor

56F802X and 56F803X

Peripheral Reference Manual

16-Bit Digital Signal Controllers (DSC)

MC56F80XXRM
Rev. 3
02/2007

freescale.com

freescale™ semiconductor

Contents

Chapter 2
Analog Digital Converter (ADC)

Chapter 3
Computer Operating Properly (COP)

Chapter 4
Inter-Integrated Circuit Interface (I^2C)

Chapter 5
On-Clock Chip Synthesis (OCCS)

Chapter 6
Flash Memory (FM)

Chapter 7
General Purpose Input/Output (GPIO)

Chapter 8
Pulse Width Modulator (PWM)

Chapter 9
Multi-Scalable Controller Area Network (MSCAN)

Chapter 10
Joint Test Action Group Port (JTAG)

Chapter 11
Power Supervisor (PS)

Chapter 12
Queued Serial Communications Interface (QSCI)

Chapter 13
Queued Serial Peripheral Interface (QSPI)

Chapter 14
Quad-Timer (TMR)

Chapter 15
Voltage Regulator (VREG)

Chapter 16
Programmable Interval Timer (PIT)

Chapter 17
Digital-to-Analog Converter (DAC)

Chapter 18
Comparator (CMP)

Appendix A
Definitions, Acronyms, and Abbreviations

Appendix B
Programmer Sheets

List of Figures

List of Tables

Preface

This manual is one of a set of three documents. For complete product information, it is necessary to have all three documents. They are:

- DSP56800E Reference Manual
- MC56F802x and MC56F803x Peripheral User Manual
- Device Technical Data Sheet

Order this document by MC56F80xxRM.

About This Manual

Features of the 56F80xx Series of 16-bit digital signal controllers (DSC) are described in this preliminary manual release. Peripheral modules are documented here. This manual is intended to be used with the *DSP56800E Reference Manual* (DSP56800RM), describing the central processing unit (CPU), programming models, and instruction set details. The *5680xx Technical Data Sheet* provides electrical specifications as well as timing, pinout, packaging descriptions and chip specific details of architecture and memory map particulars.

Audience

Information in this manual is intended to assist design and software engineers integrate the 56F8000 digital signal processors into a design and/or while developing application software.

Manual Organization

This manual is arranged in chapters described below:

- **Chapter 1, Overview**—provides a brief overview to the core, devices, and peripherals, describing the structure of this document, including lists of other documentation necessary to use these chips.
- **Chapter 2, Analog Digital Converter (ADC)**—describes features, functions and registers of the analog-to-digital converter.
- **Chapter 3, Computer Operating Properly (COP)**—This module is used to help software recover from runaway code.
- **Chapter 4, Inter-Integrated Circuit Interface (I2C)**—describes a two-wire, bidirectional serial bus suitable for short distance communication among devices.
- **Chapter 5, On-Clock Chip Synthesis (OCCS)**—elaborates on the internal oscillator, relaxation oscillator, phase lock loop (PLL), and timer distribution chain for the 568000.
- **Chapter 6, Flash Memory (FM)**—describes the program Flash and boot Flash features and registers.
- **Chapter 7, General Purpose Input/Output (GPIO)**—describes how peripheral pins are multiplexed with GPIO functions.

- **Chapter 8, Pulse Width Modulator (PWM)**—describes the function, configuration, and registers of the PWM.
- **Chapter 9, Multi-Scalable Controller Area Network (MSCAN)**—describes the capabilities of CAN as a communication controller
- **Chapter 10, Joint Test Action Group Port (JTAG)**—explains the joint test action group (JTAG) testing methodology and its capabilities with test access port (TAP) and enhanced OnCE (fully explained in the DSP56800E Core Family Reference Manual).
- **Chapter 11, Power Supervisor (PS)**—details how the on-chip PS module monitors on-chip voltages.
- **Chapter 12, Queued Serial Communications Interface (QSCI)**—communicates with devices such as other DSCs, microprocessors, or peripherals providing the primary data input path as codecs.
- **Chapter 13, Queued Serial Peripheral Interface (QSPI)**—is described with the capability to communicate with external devices, such as liquid crystal displays (LCDs) and microcontroller units (MCUs).
- **Chapter 14, Quad-Timer (TMR)**—outlines the internal quad timer devices available, including features and registers.
- **Chapter 15, Voltage Regulator (VREG)**—describes the on-chip mechanism used to regulate an external 3.3 V supply down to 2.5 V levels for use with internal logic.
- **Chapter 16, Programmable Interval Timer (PIT)**—provides information about a 16-bit up counter, a modulo register, and a control register.
- **Chapter 17, Digital-to-Analog Converter (DAC)**—discusses the architecture, programming model, operating modes, and initialization of the peripheral interface.
- **Chapter 18, Comparator (CMP)**—provides a digital peripheral interface used to control one analog comparator module.
- **Appendix A, Definitions, Acronyms, and Abbreviations**—provides definitions of terms, peripherals, acronyms and register names used in this manual.
- **Appendix B, Programmer Sheets—**supplies concise, one-location registers and their preference tables intended to simplify programming of 56F80xx peripherals.

Suggested Reading

A list of DSC-related books is provided here as an aid to those who may be new to DSCs:

1. *Advanced Topics in Signal Processing*, Jae S. Lim and Alan V. Oppenheim (Prentice-Hall: 1988).
2. *Applications of Digital Signal Processing*, A. V. Oppenheim (Prentice-Hall: 1978).
3. *Digital Processing of Signals: Theory and Practice*, Maurice Bellanger (John Wiley and Sons: 1984).
4. *Digital Signal Processing*, Alan V. Oppenheim and Ronald W. Schafer (Prentice-Hall: 1975).
5. *Digital Signal Processing: A System Design Approach*, David J. DeFatta, Joseph G. Lucas, and William S. Hodgkiss (John Wiley and Sons: 1988).
6. *Discrete-Time Signal Processing*, A. V. Oppenheim and R.W. Schafer (Prentice-Hall: 1989).
7. *Foundations of Digital Signal Processing and Data Analysis*, J. A. Cadzow (Macmillan: 1987).
8. *Handbook of Digital Signal Processing*, D. F. Elliott (Academic Press: 1987).
9. *Introduction to Digital Signal Processing*, John G. Proakis and Dimitris G. Manolakis (Macmillan: 1988).
10. *IP Bus Specifications*, Semiconductor Reuse Standard, SRSIPB1, v 2.0, Draft 1.6.
11. *Multirate Digital Signal Processing*, R. E. Crochiere and L. R. Rabiner (Prentice-Hall: 1983).

12. *Signal Processing Algorithms*, S. Stearns and R. Davis (Prentice-Hall: 1988).
13. *Signal Processing Handbook*, C. H. Chen (Marcel Dekker: 1988).
14. *Signal Processing: The Modern Approach*, James V. Candy (McGraw-Hill: 1988).
15. *Theory and Application of Digital Signal Processing*, Lawrence R. Rabiner and Bernard Gold (Prentice-Hall: 1975).

Manual Conventions

Conventions used in this manual:

- Bits within registers are always listed from most significant bit (MSB) to least significant bit (LSB).
- Bits within a register are formatted AA[n:0] when more than one bit is involved in a description. For purposes of description, the bits are presented as if they are contiguous within a register. However, this is not always the case. Refer to the programming model diagrams or to the programmer's sheets to see the exact location of bits within a register.
- When a bit is described as *set*, its value is set to one. When a bit is described as cleared, its value is set to zero.
- The word pin is a generic term for any pin on the chip.
- Pins or signals asserted low, made active when pulled to ground, have an over-bar above their name. For example, the $\overline{SS0}$ pin is asserted low.
- Hex values are indicated with a dollar sign ($) preceding the hex value, as follows: $F1A0 is the X memory address for an interrupt priority register (IPR).
- Code examples follow in a single spaced font.

```
BFSET     #$0007,X:PCC     ; Configure:                              line
1

                           ; MISO0, MOSI0, SCK0 for SPI master       line
2

                           ; ~SS0 as PC3 for GPIO                    line
3
```

- Pins or signals listed in code examples asserted as low have a tilde in front of their names. In the previous example, line three refers to the $\overline{SS0}$ pin, shown as ~SS0.
- The word reset is used in three different contexts in this manual. They are described as:
  - A reset pin is always written as $\overline{RESET}$ or ($\overline{RST}$), in uppercase, using the over bar
  - The processor state occurs when the $\overline{RESET}$ or ($\overline{RST}$) pin is asserted. It is always written as Reset, with a capitalized first letter
  - The word reset refers to the reset function. It is written in lowercase, without italics, used here only for differentiation. The word may require a capital letter as style dictates, such as in headings and captions
- The word assert means a high true (active high) signal is pulled high to V_{DD}, or a low true (active low) signal is pulled low to ground. The word deassert means a high true signal is pulled low to ground, or a low true signal is pulled high to V_{DD}.

Pin Conventions

Signal/Symbol	Logic State	Signal State	Voltage[1]
$\overline{\text{PIN}}$	True	Asserted	V_{IL}/V_{OL}
$\overline{\text{PIN}}$	False	Deasserted	V_{IH}/V_{OH}
PIN	True	Asserted	V_{IH}/V_{OH}
PIN	False	Deasserted	V_{IL}/V_{OL}

1. Values for V_{IL}, V_{OL}, V_{IH}, and V_{OH} are defined by individual product specifications.

Register and Table Conventions

Throughout the manual, registers and tables displaying a grayed area designate reserved bits.

= Reserved or not implemented bit, write or read as zero for future compatibility.

Chapter 1
Overview

1.1 Introduction to the 56800E Core

The 56F800E core combines digital processing power with the functionality of a microcontroller. Adding a flexible set of peripherals creates an extremely cost-effective system solution on a single chip. Because of its low cost, configuration flexibility, and compact program code, the 56F80xx is well-suited for many applications.

The DSP56800 core is based on a 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 straight forward generation of efficient, compact 16-bit control code. The instruction set is also highly efficient for C/C++ compilers, enabling rapid development of optimized control applications.

1.1.1 DSP56800 Core Enhancements

The DSP56800 core architecture is C source-code compatible with 56F800E family devices, extending the family architecture, thereby adding these features:

- Byte and long data types, supplementing the word data type of the 56F80xx
- 24-bit data memory address space
- 21-bit program memory address space
- Two additional 24-bit address registers
- Two additional 36-bit accumulator registers
- Full-precision integer multiplication
- 32-bit logical and shifting operations
- Second read in dual read instruction can access off-chip memory
- Loop count (LC) register extended to 16 bits
- Support for nested DO looping through additional loop address and count registers
- Loop address and hardware stack extended to 24 bits
- Three additional interrupt levels with a software interrupt for each level
- Enhanced on-chip emulation (EOnCE) with three debugging modes:
  - Non-intrusive real-time debugging
  - Minimally intrusive real-time debugging
  - Breakpoint and step modes (core is halted)

1.1.2 DSP56800 Core

The DSP56800 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 thereby increasing throughput. There is also an independent bit manipulation unit to enable efficient bit field operations. Each functional unit interfaces with the other units, memory, and the memory-mapped peripherals over the core's internal address and data buses. A block diagram of the DSP56800 core architecture is illustrated in Figure 1-1.

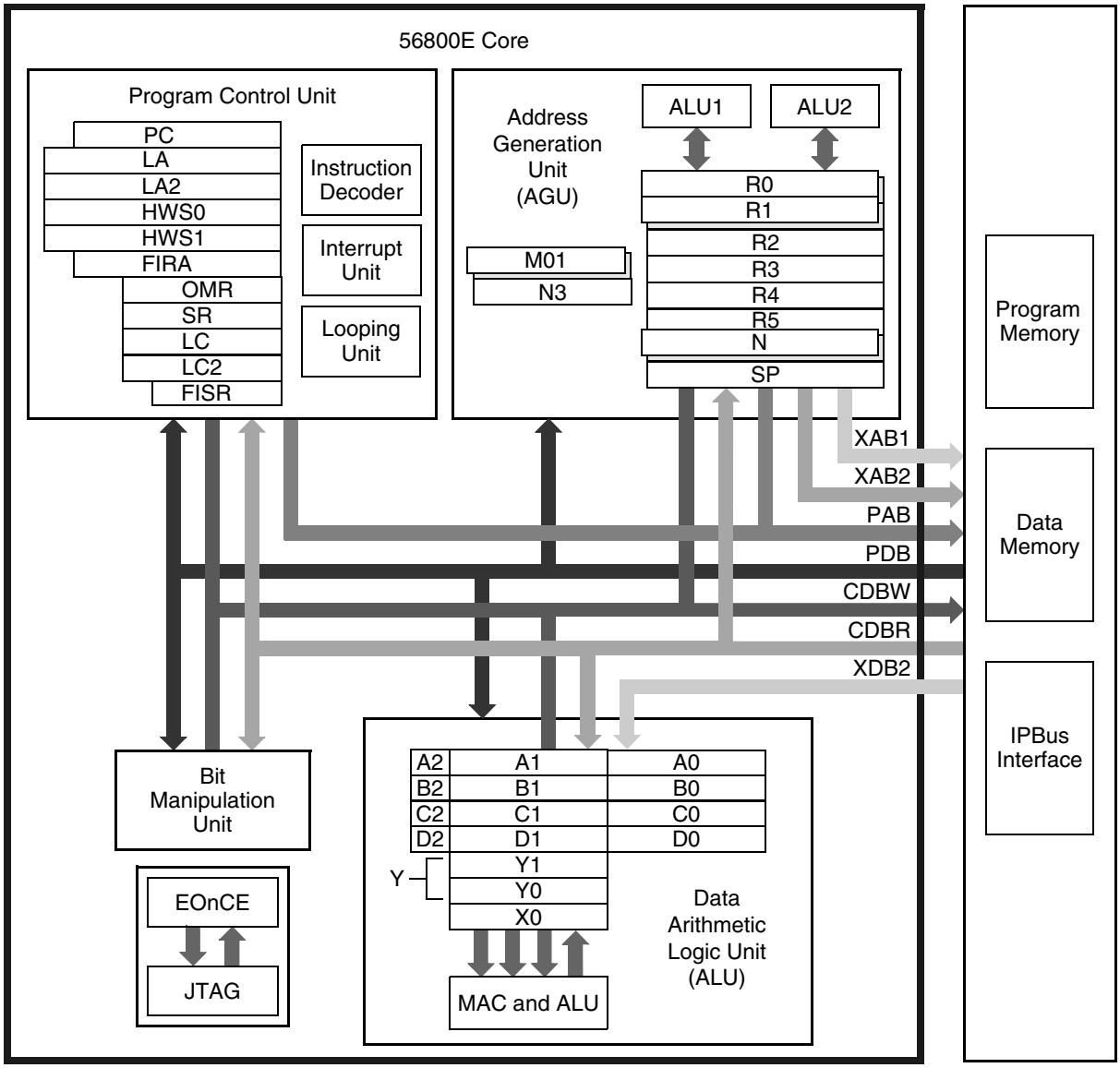


Figure 1-1. DSP56800 Core Block Diagram

Instruction execution is pipelined to take advantage of the parallel units, thereby significantly decreasing execution time for each instruction. For example, within a single execution cycle, it is possible for the:

- Data ALU to perform a multiplication operation
- AGU to generate up to two addresses
- Program controller to pre fetch the next instruction

The major components of the DSP56800 core include:

- Address buses
- Data buses
- Data arithmetic logic unit (ALU)
- Address generation unit (AGU)
- Program controller and hardware looping unit
- Bit manipulation unit
- Enhanced OnCE debugging module
- Clock generation
- Reset circuitry

1.1.2.1 Address Buses

The core contains three address buses:

1. Program memory address bus (PAB) – 21 bits
2. Primary data address bus (XAB1) – 24 bits
3. Secondary data address bus (XAB2) – 24 bits

The program address bus is used to access (16-bit) instruction words in program memory. The two data address buses allow for two simultaneous accesses to data (X) memory. The XAB1 bus can address byte, word, and long data types. The XAB2 bus is limited to (16-bit) word accesses.

All three buses address on-chip memory. They can also address off-chip memory on devices containing an external bus interface unit.

1.1.2.2 Data Buses

Data transfers inside the chip occur over these buses:

- Two unidirectional 32-bit buses:
  - Core data bus for reads (CDBR)
  - Core data bus for writes (CDBW)
- Two unidirectional 16-bit buses:
  - Secondary X data bus (XDB2)
  - Program data bus (PDB)
- IPBus interface

Data transfers between the data ALU and data memory use the core data bus for reads (CDBR) and core data bus for writes (CDBW) when a single memory read or write is performed. When two simultaneous memory reads are performed, the transfers use the CDBR and XDB2 buses. All other data transfers to core blocks occur using the CDBR and CDBW buses. Peripheral transfers occur through the IPBus interface. Instruction word fetches occur over the PDB.

This bus structure supports up to three simultaneous 16-bit transfers. Any one of the following can occur in a single clock cycle:

- One instruction fetch
- One read from data memory
- One write to data memory
- Two reads from data memory
- One instruction fetch and one read from data memory
- One instruction fetch and one write to data memory
- One instruction fetch and two reads from data memory

An instruction fetch takes place on every clock cycle, although it is possible for data memory accesses to be performed without an instruction fetch. Such accesses typically occur when a hardware loop is executed and the repeated instruction is only fetched on the first loop iteration.

1.1.2.3 Data Arithmetic Logic Unit (Data ALU)

The ALU performs all of the arithmetic, logical, and shifting operations on data operands. The data ALU contains the following components:

- Three, 16-bit data registers (X0, Y0, and Y1)
- Four, 36-bit accumulator registers (A, B, C, and D)
- One multiply-accumulator (MAC) unit
- A single-bit accumulator shifter
- One arithmetic and logical multi-bit shifter
- One MAC output limiter
- One data limiter

The data ALU can perform multiplication, multiply-accumulation (with positive or negative accumulation), addition, subtraction, shifting, and logical operations in a single cycle. Division and normalization operations are provided by iteration instructions. Signed and unsigned multiple precision arithmetic is also supported. All operations are performed using two’s-complement fractional or integer arithmetic.

Data ALU source operands can be 8, 16, 32, or 36 bits in size and can be located in memory, in immediate instruction data, or in the data ALU registers. Arithmetic operations and shifts can have 16-, 32-, or 36-bit results. Logical operations are performed on 16- or 32-bit operands and yield results of the same size. The results of data ALU operations are stored either in one of the data ALU registers or directly in memory.

1.1.2.4 Address Generation Unit (AGU)

The address generation unit (AGU) performs all of the calculations of effective addresses for data operands in memory. It contains two address ALUs, allowing up to two 24-bit data addresses to be generated every instruction cycle:

- One for the primary data address bus (XAB1)
- One for the secondary data address bus (XAB2)

The address ALU can perform both linear and modulus address arithmetic. The AGU operates independently of the other core units, minimizing address-calculation overhead.

The AGU can directly address 2^{24} (16M) words on the XAB1 and XAB2 buses. It can access 2^{21} (2M) words on the PAB. The XAB1 bus can address byte, word, and long data operands. The PAB and XAB2 buses can only address words in memory.

The AGU consists of the following registers and functional units:

- Seven 24-bit address registers (R0–R5 and N)
- Four 24-bit shadow registers (for R0, R1, N, and M01)
- A 24-bit dedicated stack pointer (SP) register
- Two offset registers (N and N3)
- A 16-bit modifier register (M01)
- A 24-bit adder unit
- A 24-bit modulus arithmetic unit

Each of the address registers (R0–R5) can contain either data or an address. All of these registers can provide an address for the XAB1 and PAB address buses; addresses on the XAB2 bus are provided by the R3 register. The N offset register can be used either as a general-purpose address register, as an offset, or update value for the addressing modes supporting those values. The second 16-bit offset register (N3) is used only for offset or update values. The modifier register (M01) selects between linear and modulus address arithmetic.

1.1.2.5 Program Controller and Hardware Looping Unit

The program controller is responsible for instruction fetching and decoding, interrupt processing, hardware interlocking, and hardware looping. Actual instruction execution takes place in the other core units, such as in the data ALU, AGU, or bit manipulation unit.

The program controller contains the following

- An instruction latch and decoder
- The hardware looping control unit
- Interrupt control logic
- A program counter (PC)
- Two special registers for Fast Interrupts
  - Fast interrupt return address (FIRA) register
  - Fast interrupt status register (FISR)
- Seven user-accessible status and control registers
  - Two-level deep hardware stack (HWS)
  - Loop address (LA) register
  - Loop address 2 (LA2) register
  - Loop count (LC) register
  - Loop count 2 (LC2) register
  - Status register (SR)
  - Operating mode register (OMR)

The operating mode register (OMR) is a programmable register controlling the operation of the DSP56800E core, including the memory map configuration. The initial operating mode is typically latched on reset from an external source; it can subsequently be altered under program control.

The loop address (LA) and loop count (LC) registers work in conjunction with the hardware stack (HWS) to support no overhead hardware looping. The hardware stack is an internal last-in-first-out (LIFO) buffer consisting of two 24-bit words to store the address of the first instruction of a hardware DO loop. When executing the DO

instruction, the address of the first instruction in the loop is pushed onto the HWS. When a loop finishes normally or an ENDDO instruction is encountered, the value is popped from the HWS. This process allows for one hardware DO loop to be nested inside another.

1.1.2.6 Bit Manipulation Unit

The bit manipulation unit performs bit field operations on data memory words, peripheral registers, and registers within the DSP56800 core. It is capable of testing, setting, clearing, or inverting individual or multiple bits within a 16-bit word. The bit manipulation unit can also test bytes for branch-on-bit field instructions.

1.1.2.7 Enhanced On-Chip Emulation (EOnCE) Module

The enhanced on-chip emulation (EOnCE) module allows interaction in a debug environment with the DSP56800 core and its peripherals. Its capabilities include:

- Examining registers
- Accessing memory, or on-chip peripherals
- Setting breakpoints in memory
- Stepping or tracing instructions

The EOnCE module provides simple, inexpensive, and speed independent access to the DSP56800 core for sophisticated debugging and economical system development. The JTAG port allows access to the EOnCE module and through the 56F80xx device to its target system, retaining debug control without sacrificing other user accessible on-chip resources. This technique eliminates the costly cabling and access to processor pins required by traditional emulator systems. The EOnCE interface is fully described in the DSP56800 Reference Manual.

1.1.3 System Bus Controller

The DSP56800 system bus controller (SBC) provides an interface between the DSP56800 core and other modules on the system bus. The SBC is composed of a set of buffers for the address and control signals originating at the core, and a separate set of multiplexers, routing data from each memory-mapped block back to the core.

The DSP56800 architecture includes two separate bus models:

1. System bus
2. IPBus

Internal memories, the external memory interface, and the core are located on the system buses. All peripherals connect to the IPBus. Access to the IPBus by the 16-bit controller core is facilitated by the IPBus bridge. The system bus controller does not participate in IPBus transactions. Within this document, all descriptions of bus operations pertain only to the system bus.

For performance reasons, all system bus signals in the DSP56800 architecture have a single driver, as opposed to the more common three-state bus configurations. Read data from each memory-mapped device is multiplexed to avoid contention. Since the 16-bit controller core is the only system bus master, there is no need for multiplexers on the address, control or write data buses.

1.1.4 Operation Method

The DSP56800 system utilizes a pipelined memory architecture and separate program and data buses. Each memory cycle is completed in three or more system clock cycles. During the first of these cycles, the core presents an address, along with control signals, indicating the type of memory cycle being initiated. This clock cycle is

referred to as the address phase of a memory cycle. The following cycle is an intermediate step not involving bus activity related to the memory cycle in progress. Finally, the data phase occurs. During this phase data is transferred to or from the master, depending upon the type of cycle initiated during the address phase.

Memory cycles can overlap in each clock cycle, a new address phase can begin while the data phase for a preceding memory cycle occurs. In certain cases, memory devices or the 16-bit controller core may require additional time to complete operations. When this occurs, clock edges to other modules are withheld by the clock generation circuitry.

1.2 Introduction to 56F80xx Devices

1.2.1 Applications

The 56F8000 family of products includes many peripherals useful for applications such as:

- Three-phase motor control
- Dimming lamp ballast
- Switched mode power supply
- Soft-switching PFC
- Appliance motor control
- DC-DC power supplies
- Smart sensors
- Instrumentation

1.2.2 Features

The 56F80xx family of products provides a variety of memory options and peripherals, thereby enhancing performance and reducing application cost while promoting an ease of product development. Features making these benefits possible include:

- Competitive cost sensitive packaging
- Also see Section 1.2.4, IPBus Bridge (IPBB)

1.2.3 System Architecture and Peripheral Interface

The DSP56800 system architecture encompasses all on-chip components, including the core, on-chip memory, peripherals, and the buses necessary to connect them. Figure 1-2 illustrates the overall system architecture for a device with an external bus.

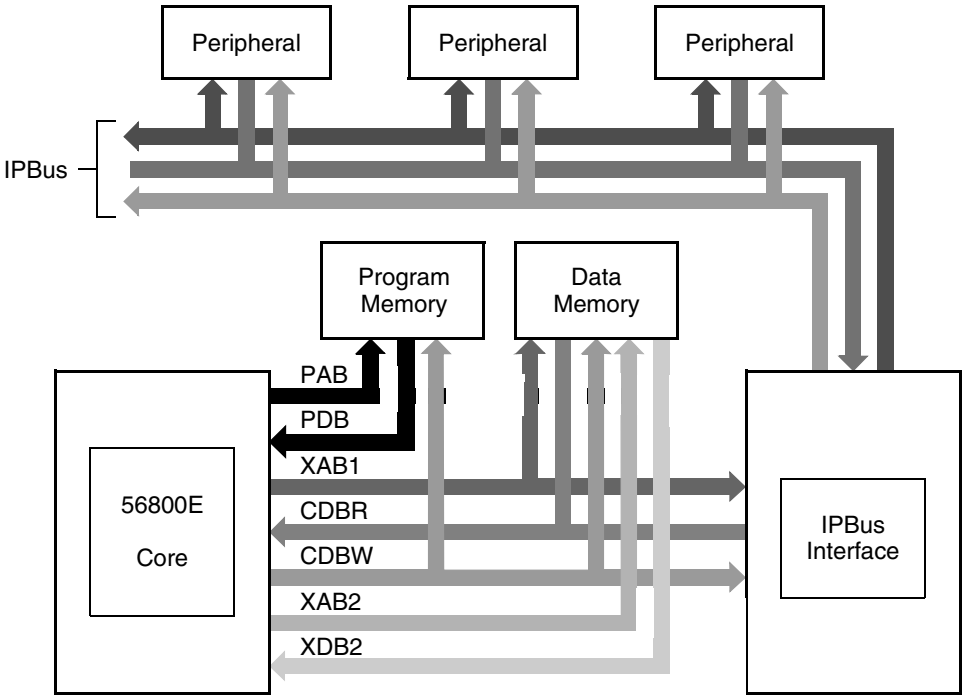


Figure 1-2. 56800E Chip Architecture with External Bus

The complete architecture includes the following components:

- DSP56800 core
- On-chip program memory
- On-chip data memory
- On-chip peripherals
- IPBus peripheral interface

Some DSP56800 devices might not implement an external bus interface. Regardless of the implementation, all peripherals communicate with the DSP56800 core via the IPBus interface. The program and secondary X data (XDB2) buses are not connected to peripherals.

1.2.4 IPBus Bridge (IPBB)

The IPBus Bridge (IPBB) provides a means for communication between the high speed core and the low-bandwidth devices on the IP peripheral bus. Among other functions, the bridge is responsible for maintaining an orderly and synchronized communication between devices on both sides potentially running at different clock frequencies.

The IPBus architecture supports a variety of on-chip peripherals:

- Analog-to-digital converters (ADC)
- Computer operating properly (COP) module
- Phase-locked loop (PLL) module
- Flash memory (FM) module
- Programmable general-purpose I/O (GPIO) modules

- Joint test action group port (JTAG) module
- Power supervisor (PS) module
- Pulse width module (PWM)
- Queued serial communication interface (QSCI/LIN) modules
- Queued serial peripheral interface (QSPI) modules
- 16-bit quad timer (TMR) modules
- Inter-integrated circuit interface bus (I^2C) module
- Voltage regulator (VREG) module
- Controller area network (CAN) module
- Programmable interval timer (PIT) module
- Digital to converter (DAC) module
- Comparator (CMP) module

Figure 1-3 denotes the position and interface of the IPBus bridge with other main blocks within the chip.

Other connections in the figure not pertaining to the primary function of the bridge are omitted for clarity; nevertheless they are discussed as appropriate. A brief description of the bridge interface with various main components on both sides is also provided.

1.2.4.1 System Side Operation

On the system side the IPBus bridge operates at 16-bit controller core frequency and fully supports pipelined communication with the core. The bridge acts as a slave device on this bus. The bridge is responsible for initiating IPBus transactions only per requests initiated by the 16-bit controller core.

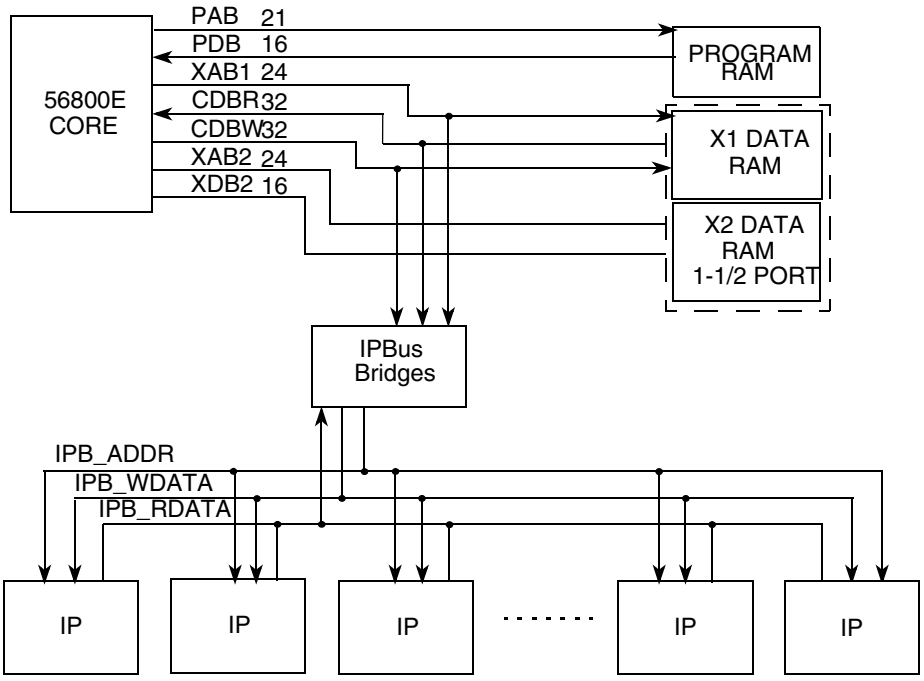


Figure 1-3. IPBus Bridge Interface With Other Main Components System Side Operation

1.2.4.2 Peripheral Side Operation

On the peripheral side, the IPBus bridge accesses various devices through a standard non-pipelined IPBus interface. Separate bus lines are used for read and write transactions.

1.2.5 Peripheral Interrupts/Interrupt Controller Module

The peripherals on the 56F80xx use the interrupt channels found on the DSP56800 core. Each peripheral has its own interrupt vector (often more than one interrupt vector for each peripheral), and can selectively be enabled or disabled via the interrupt priority register (IPR) found in the interrupt controller (ITCN) module. Detailed information regarding the interrupt controller is located in the data sheet. Design includes these distinctive features:

- Programmable priority levels for each IRQ
- Two programmable fast interrupts
- Notification to the SIM module to restart clocks out of wait and stop modes

NOTE:

Please see the device data sheet for detailed information about this module.

1.2.5.1 System Integration Module (SIM)

The SIM module is a system catchall for the glue logic tieing 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 control and distribution
- STOP/WAIT control
- System status registers
- I/O pad multiplexing
- Registers for software access to the JTAG ID of the chip
- Enforcing Flash security

NOTE:

Please see the device data sheet for detailed information about this module.

1.2.6 56F80xx Peripheral Features

1.2.6.1 Analog-to-Digital Converter (ADC)

Detailed information regarding the ADC peripheral is located in Chapter 2 of this manual.

- Twelve-bit resolution
- Maximum ADC clock frequency is 5.33 MHz with 187 ns period
- Sampling rate up to 1.78 million samples per second[1]
- Single conversion time of 8.5 ADC clock cycles (8.5 × 187 ns = 1.59 μs)

- Additional conversion time of 6 ADC clock cycles (6 × 187 ns = 1.125 µs)
- Eight conversions in 26.5 ADC clock cycles (26.5 × 187 ns = 4.97 µs) using parallel mode
- ADC can be synchronized to the PWM via the SYNC0/1 input signal provided the integration permits the PWM to trigger a timer channel connected to that input
- Ability to sequentially scan and store up to 16 measurements
- Ability to scan and store up to eight measurements, on each ADC converter while operating simultaneously and in parallel
- Ability to scan and store up to eight measurements, on each ADC converter while operating asynchronously to each other in parallel
- Optional interrupts at end of scan if an out-of-range limit is exceeded or at 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

1.2.6.2 Computer Operating Properly (COP)

Detailed information regarding this peripheral is located in Chapter 3 of this manual.

- Programmable timeout period = (cop_prescaler × (TIMEOUT + 1)) MSTR_OSC clock cycles, where TIMEOUT can be from $0000 to $FFFF
- Programmable wait and stop mode operation
- COP timer is disabled while host CPU is in debug mode
- Causes loss of reference reset 128 cycles after loss of reference clock to the PLL is detected
- Choice of clock sources for counter

1.2.6.3 Inter-Integrated Circuit Interface (I^2C)

Detailed information regarding the I^2C peripheral is located in Chapter 4 of this manual.

- 2-wire I^2C serial interface – consists of a serial data (SDA) line and a serial clock (SCL) line
- Two speed modes:
  - Standard mode
  - Fast mode
- Clock synchronization
- Master or slave I^2C operation
- Supports multi-master operation (bus arbitration)
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Slave bulk transmit mode

1. Once in loop mode, the time between each conversion is 6 ADC clock cycles (1.125 µs). Using simultaneous conversion, two samples can be obtained in 1.125 µs. Samples per second is calculated according to 1.125 µs per two samples or 1,780,000 samples per second.

- Ignores CBUS addresses (an older ancestor of I^2C previously shared the I^2C bus)
- Transmit and receive buffers
- Interrupt or polled-mode operation
- Handles bit and byte waiting at all bus speeds
- Digital filter for the received SDA and SCL lines
- Component parameters for configurable software driver support

1.2.6.4 On-Chip Clock Synthesis (OCCS)

Detailed information regarding this peripheral is located in Chapter 5 of this manual.

- Internal relaxation oscillator
- Crystal oscillator control
- Ability to power down internal relaxation oscillator or crystal oscillator
- Ability to put the internal relaxation oscillator into a standby mode
- Three-bit postscaler provides control for the PLL output
- Ability to power down the internal PLL
- Provides 2× master clock frequency and OSC_CLK signals.
- Safety shutdown feature available in the event that the PLL reference clock disappears
- Can be driven from an external clock source

1.2.6.5 Flash Memory (FM)

Detailed information regarding this peripheral is located in Chapter 6 of this manual.

- Program Flash memory can also be used to store data
- 32 MHz single cycle operation for all program Flash accesses
- Automated program and erase operation
- Interrupts on command completion, command buffer empty, and access error
- Fast page erase
- Single power supply program and erase
- Security feature
- Sector protection system
- The Flash memory (FM) supports byte and word/read operations by the host digital signal controller (DSC)
- Code integrity check using built-in data signature calculation

1.2.6.6 General Purpose Input/Output (GPIO)

Detailed information regarding this peripheral is located in Chapter 7 of this manual.

- Individual control for each pin to be in either peripheral or GPIO mode
- Individual input/output direction control for each pin in GPIO mode
- Individual pullup enable control for each pin in either peripheral or GPIO mode

- Individual output push-pull mode or open drain mode control for each pin in either peripheral or GPIO mode
- Individual output drive strength control for each pin
- Usable as a keypad interface
- Ability to monitor pin logic values even when GPIO are not enabled by using the RDATA register
- Interrupt assert capability
- 5 V tolerant

1.2.6.7 Pulse Width Modulator (PWM)

Detailed information regarding this peripheral is located in Chapter 8 of this manual.

- PWM operation clock runs at either system clock or 3× system clock
- Six PWM signals
  - All independent
  - Complementary pairs
  - Mix independent and complementary
- Features of complementary channel operation
  - Separate deadtime insertions for rising and falling edges
  - Separate top and bottom pulse width correction via software
  - Asymmetric PWM output within center align operation
  - Separate top and bottom polarity control
- Edge- or center-aligned PWM signals
- 15 bits of resolution
- Half-cycle reload capability
- Integral reload rates from 1 to 16
- Individual software controlled PWM output
- Programmable fault protection
- PWM compare output polarity control
- PWM output polarity control
- Push-pull and open drain modes are available on PWM pins
- External sync control with sync window
- Programmable input filters for the fault signals
- Write-protected registers
- Selectable PWM supply source for each complementary PWM signal pair
  - PWM generator
  - External GPIO pin
  - Internal timer channel
  - ADC conversion result, taking into account values set in ADC high and low limit registers

All three pairs can be driven by any one of the external sources. The following features are disabled when any of external sources is used as PWM source:

- PWM sync is not applicable
- PWM reload registers have no effect

1.2.6.8 Scalable Controller Area Network (MSCAN)

Detailed information regarding this peripheral is located in Chapter 9 of this manual.

- Implementation of the CAN protocol – Version 2.0 A/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 wake-up functionality with integrated low-pass filter
- 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 CAN 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
  - CAN enable
- Global initialization of configuration registers

1.2.6.9 Joint Test Action Group Port (JTAG)

Detailed information regarding this peripheral is located in Chapter 10 of this manual.

- Provide a means of accessing 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 pull-up devices on peripheral boundary scan pins

1.2.6.10 Power Supervisor (PS)

Detailed information regarding this peripheral is located in Chapter 11 of this manual.

- Power-on reset (POR)
  - Holds device in reset until:

1. Depending on the actual bit timing and the clock jitter of the PLL.

  - V_{DD} core voltage exceeds 1.8 V
  - Regulator voltages have risen above LVI thresholds (2.2 V and 2.7 V)

- Core Low Voltage Interrupt (LVI22)
  — Generated when the 2.5 V rail drops below 2.2 V
- I/O Low Voltage Interrupt (LVI27)
  — Generated when the 3.3 V rail drops below 2.7 V

1.2.6.11 Queued Serial Communications Interface (QSCI)

Detailed information regarding this peripheral is located in Chapter 12 of this manual.

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- Thirteen-bit integer and 3-bit fractional baud rate selection
- Programmable 8- or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter CPU interrupt requests
- Programmable polarity for transmitter and receiver
- Two receiver wake-up methods:
  — Idle line
  — Address mark
- Interrupt-driven operation with seven flags:
  — Transmitter empty
  — Transmitter idle
  — Receiver full
  — Receiver overrun
  — Noise error
  — Framing error
  — Parity error
  — LIN sync error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection
- Four-word deep FIFOs available on transmit and receive buffers

1.2.6.12 Queued Serial Peripheral Interface (QSPI)

Detailed information regarding this peripheral is located in Chapter 13 of this manual.

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four-word deep FIFOs available on transmit and receive buffers
- Programmable length transmissions (2–16 bits)

- Programmable transmit and receive shift order (MSB as first bit transmitted)
- Nine 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
  - QSPI receiver full/error, supporting three interrupt flags:
    - Receiver full
    - Mode fault error flag
    - Overflow error flag
  - SPTE (QSPI transmitter empty)
- Wired OR mode functionality enabling connection to multiple QSPIs

1.2.6.13 Quad Timer (TMR)

Detailed information regarding this peripheral is located in Chapter 14 of this manual.

- Four, 16-bit counters/timers
- Count up/down
- Counters can be cascaded
- Programmable count modulo
- Maximum count rate equals peripheral clock ÷2 for external clocks
- Maximum count rate equals peripheral clock for internal clocks
- Count once or repeatedly
- Counters are capable of being preloaded
- Compare registers are capable of being preloaded
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Programmable operation during debug mode
- Programmable input filter
- Counting start can be synchronized across counters

1.2.6.14 Voltage Regulator (V_{REG})

Detailed information regarding this peripheral is located in Chapter 15 of this manual.

- Provide a 2.5 V ±10% accuracy
- Provide a maximum current of 125 mA for the large regulator
- Provide a maximum current of a 4 mA for the small regulator

1.2.6.15 Programmable Interval Timer (PIT)

Detailed information regarding this peripheral is located in Chapter 16 of this manual.

- Sixteen-bit counter/timer
- Programmable count modulo
- Max count rate equals peripheral clock rate
- Slave mode allows synchronization of multiple PIT count enables

1.2.6.16 Digital-to-Analog Converter (DAC)

Detailed information regarding this peripheral is located in Chapter 17 of this manual.

- Twelve-bit resolution
- Two microsecond conversion rate
- Power down mode
- Output can be routed to internal comparator, ADC, or optionally off chip
- DAC can drive 3 KΩ, 400 pF load
- Choice of asynchronous or synchronous updates
  - Sync input can be connected to on chip PITs, TMRs, PWM reload flag, or GPIO inputs
- Automatic mode allows the DAC to generate its own output waveforms including square, triangle, and sawtooth waveforms
- Automatic mode allows programmable period, update rate, and range

1.2.6.17 Comparator (CMP)

Detailed information regarding this peripheral is located in Chapter 18 of this manual.

- Continuous-time differential-input analog comparator
- Internal switching matrix supports the independent connection of the analog inputs to the + and – input of the analog comparator and to the comparator's EXPORT output
- Input sources include five analog inputs include three GPIO (referred to as CIN1, CIN2, and CIN3), a DAC output, and an IMPORT input from another comparator module. Analog comparator polarity control (optional inversion)
- Programmable low-pass filter
- Power saving mode
- Falling and rising comparator edge detection with compare interrupt on rising and/or falling comparator edge
- Output can be used to control timer inputs, PWM faults, PWM control, external pin output, or as a interrupt source

1.3 Energy Information

- Fabricated in high-density CMOS with 5 V tolerant, TTL-compatible digital inputs
- On-board 3.3 V down to 2.5 V voltage regulator for powering internal logic and memories
- On-chip regulators for digital and analog circuitry to lower cost and reduce noise

- Wait and Stop modes available
- ADC smart power management
- Each peripheral can be individually disabled to save power

Table 1-1. Document Revision History for Chapter 1

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 2
Analog Digital Converter (ADC)

2.1 Introduction

This chapter describes the analog-to-digital interface, its architecture, programming model, operating modes, and initialization of the ADC. The module is instantiated as one dual 12-bit ADC in which both ADC converters share a common voltage reference and control block. This is illustrated in Figure 2-1.

2.2 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 (S/H) circuit. A common digital control module configures and controls the functioning of the converters. ADC characteristics include:

- 12-bit resolution
- Maximum ADC clock frequency is 5.33 MHz with 187 ns period
- Sampling rate up to 1.78 million samples per second[1]
- Single conversion time of 8.5 ADC clock cycles (8.5 x 187 ns = 1.59 µs)
- Additional conversion time of 6-ADC clock cycles (6 x 187 ns = 1.125 µs)
- Eight conversions in 26.5-ADC clock cycles (26.5 x 187ns = 4.97µs) using parallel mode
- ADC can be synchronized to the PWM via the SYNC0/1 input signal provided the integration permits the PWM to trigger a timer channel connected to that input
- Ability to sequentially scan and store up to 16 measurements
- Ability to scan and store up to eight measurements, on each ADC converter while operating simultaneously and in parallel
- Ability to scan and store up to eight measurements, on each ADC converter while operating asynchronously to each other in parallel
- Optional interrupts at end of scan if an out-of-range limit is exceeded or at zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result

1. While in loop mode, the time between each conversion is 6 ADC clock cycles (1.125 µs). Using simultaneous conversion, two samples can be obtained in 1.125 µs. Samples per second is calculated according to 1.125 µs per two samples or 1,780,000 samples per second.

- Single ended or differential inputs
- PWM outputs with hysteresis for three of the analog inputs

2.3 Block Diagram

Figure 2-1 illustrates the dual ADC configuration.

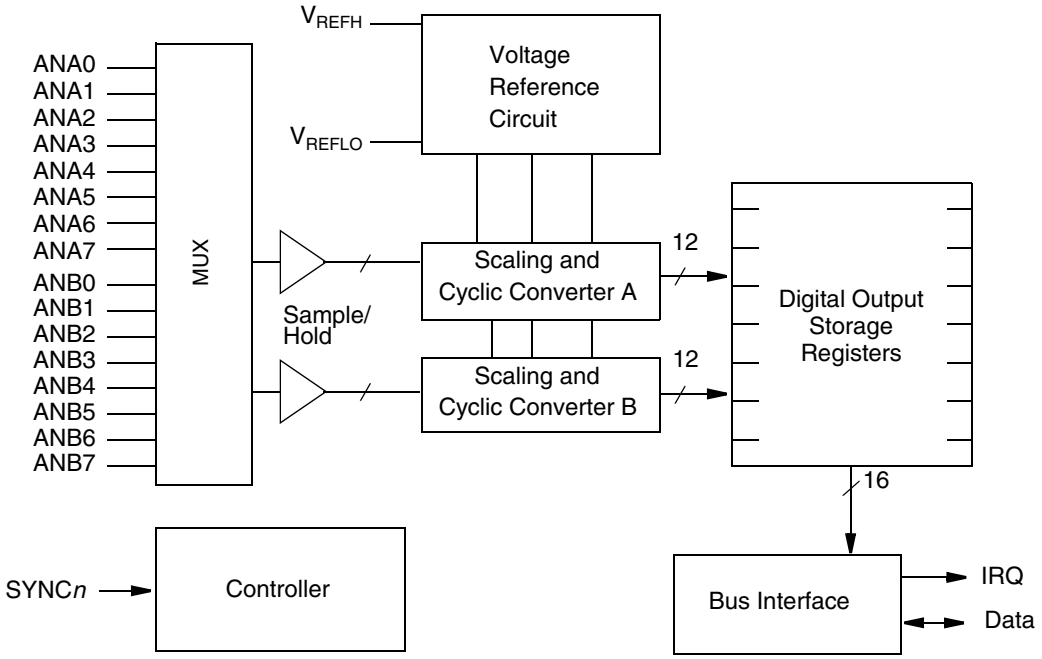


Figure 2-1. Option 1: Dual ADC Block Diagram

2.4 Functional Description

The ADC function, illustrated in Figure 2-1, consists of two 8-channel input select function, 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 by the internal functions.

The conversion process is either initiated by a SYNC signal from one of the on-chip timer channels (see chip specific documentation for the specific timer channel associated with the ADC) or by writing one to a START bit.

Starting a single conversion actually begins a sequence of conversions, or a scan. A conversion, or scan, takes up to 16-single ended or differential samples, one at a time in sequential scan mode. In parallel scan mode, the 16 samples are allocated, eight to converter A and eight to converter B. The two converters operate in parallel. In parallel scan modes, converter A can only sample analog inputs ANA0-7 while converter B can only sample analog inputs ANB0-7. Each converter can take eight samples at most.

Scan sequence is determined by defining 16 sample slots, processed in order SAMPLE0-15 during sequential scan mode. In parallel scan mode the SAMPLE0-3 and SAMPLE8-11 are processed in order by converter A, and SAMPLE4-7 and SAMPLE12-15 are processed in order by converter B. SAMPLE slots may be disabled using the SDIS register to terminate a scan early.

The following pairs of analog inputs can be configured as a differential pair; ANA0-1, ANA2-3, ANA4-5, ANA6-7, and ANB0-1, ANB2-3, ANB4-5, and ANB6-7. When configured as a differential pair, 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. During simultaneous scan mode, the parallel scans in the two converters are executed simultaneously, always resulting in simultaneous pairs of conversions, one by converter A and one by converter B. The two converters share the same start, stop, sync, and end-of-scan interrupt enable control, and interrupt. Scanning in both converters is terminated when either converter encounters a disabled sample. In non-simultaneous scan mode, the parallel scans in the two converters are done independently. The two converters have their own start, stop, sync, and 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 anytime after the SYNC pulse occurs, including while the scan it initiated is still in process.

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

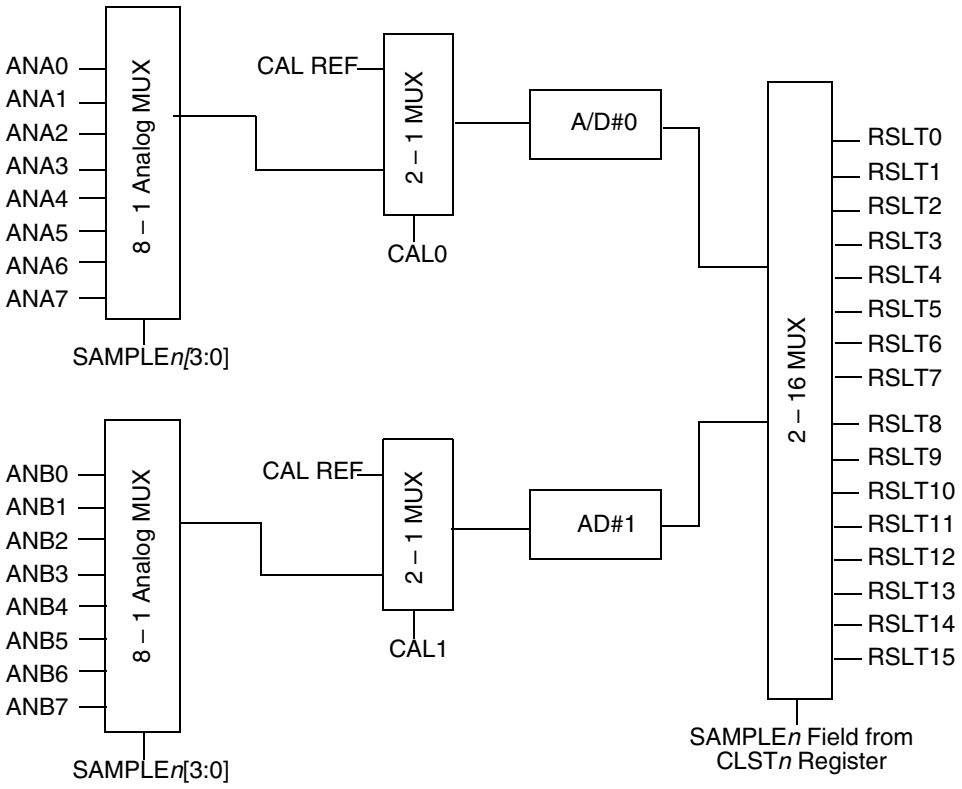


Figure 2-2. ADC Sequential Operation Mode

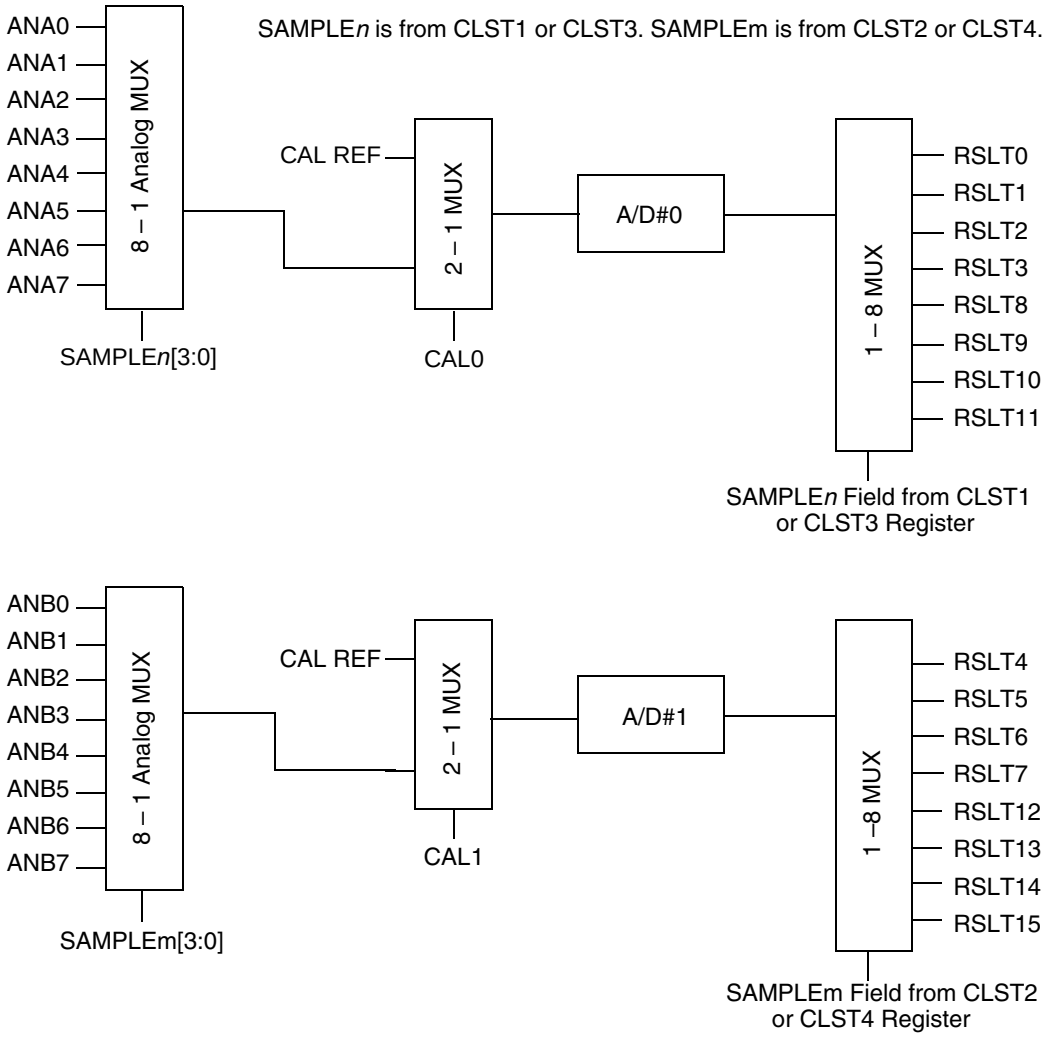


Figure 2-3. ADC Parallel Operation Mode

2.5 Input MUX Function

The input MUX function is illustrated in Figure 2-4. The channel select and single ended differential switches are indirectly controlled based on settings within the CLST1, CLST2, CLST3, CLST4, SDIS registers and the CHNCFG_L bit field of the CTRL1 register and CHNCFG_H bit field of the CTRL2 register.

- MUXing for sequential, single ended mode conversions – During each conversion cycle (sample), any one input of the two eight input groups can be directed to it's corresponding output.
- MUXing for 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.
- MUXing for parallel, single ended mode conversions – During any conversion cycle (sample), any of ANA0-ANA7 can be directed to the converter A input and any of ANB0-ANB7 can be directed to the converter B input.

- MUXing for parallel, differential mode conversions – During any conversion cycle (sample), either member of differential pair ANA0/1, ANA2/3, ANA4/5, or ANA6/7 can be referenced resulting in a differential measurement of that pair at converter A input. Likewise either member of differential pair ANB0/1, ANB2/3, ANB4/5 or ANB6/7can be referenced resulting in a differential measurement of that pair at the converter B input.

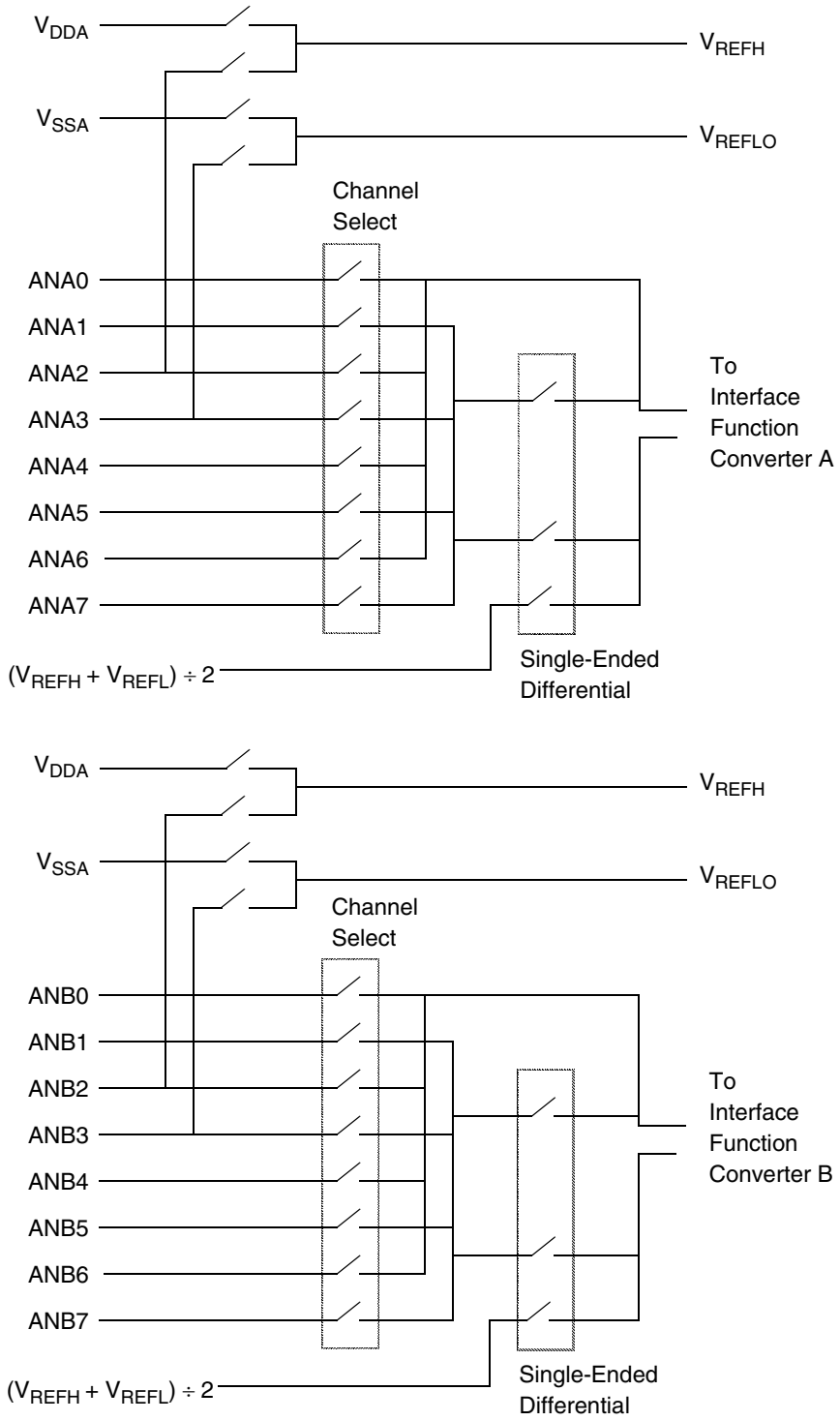


Figure 2-4. Input Select MUX

Details of single ended and differential measurement, from user perspective, are described under the CHNCFG_L, CHNCFG_H bit fields. Internally, all measurements are performed differentially. During single ended measurements, $(V_{REFH}+V_{REFLO})/2$ is used as the (–) input voltage while the selected analog input is used as the (+) input.

Table 2-1. Analog MUX Controls for Each Conversion Mode

Conversion Mode	Channel Select Switches	Single Ended Differential Switches
General Comments		The two lower switches within the dashed box are controlled such that one switch is always closed and the other open.
Sequential, Single Ended	The two 1-of-8 select muxes 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. In this mode, 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, such 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 muxes 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. In this mode, the upper switch is always closed so that any of the four inputs can get to the A/D input.
Parallel, Differential	The channel select switches are turned on in pairs, providing a dual 1-of-4 select function, such 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.

2.6 ADC Sample Conversion Modes of Operation

The ADC consists of a cyclic, algorithmic architecture using two recursive sub-ranging sections (RSD#1 and RSD#2), illustrated in Figure 2-5. Each sub-ranging section resolves a single bit for each conversion clock, resulting in an overall conversion rate of two bits per clock cycle. Each sub-ranging section is designed to run at a maximum clock speed of 5.33 MHz so a complete 12-bit conversion can be accommodated in 1.125 µs, not including sample or post processing time.

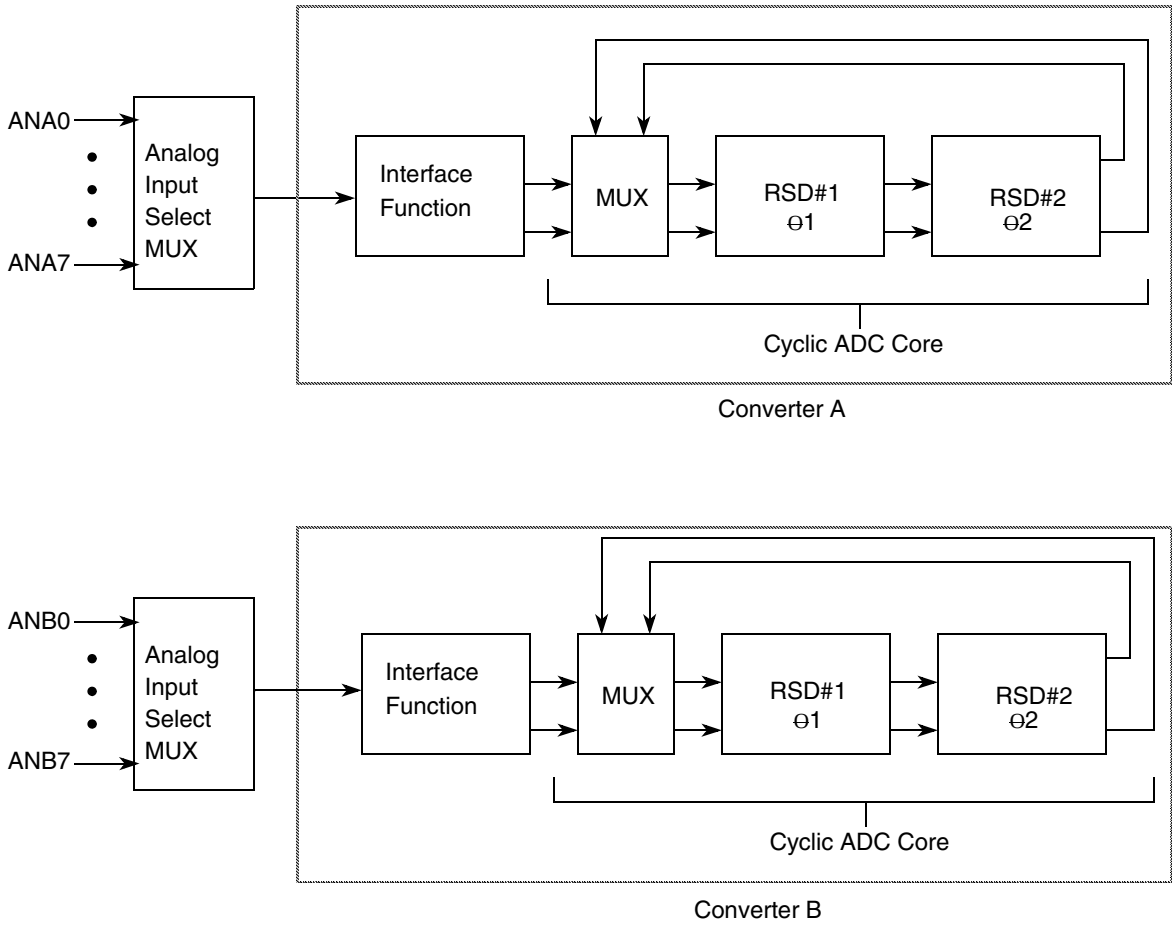


Figure 2-5. Cyclic ADC – Top Level Block Diagram

2.6.1 Normal Mode Operation

The ADC has two modes of normal operation. The mode of operation for a given sample is determined by the CHNCFG_L bit field in the CTRL1 register and CHNCFG_H in the CTRL2 register. The two normal modes of operation are:

1. Single-ended mode (CHNCFG bit = 0)– In the single-ended mode, the input MUX of the ADC selects one of the eight 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 during this mode. The ADC measures the voltage of the selected analog input and compares it against the (V_{REFH} - V_{REFLO}) reference voltage range.

2. Differential mode (CHNCFG bit = 1)– In the differential mode, the ADC measures the voltage difference between two analog inputs and compares that 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, or ANB2/3, ANB4/5, and ANB6/7. In this mode, the plus (+) terminal of the A/D core is connected to the even analog input while 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:

- ANA0 and ANA1 differential, ANA2 and ANA3 single-ended
- ANB0 and ANB1 differential, and ANB2 and ANB3 single-ended

2.6.1.1 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 in the following diagram.

$$\text{Single Ended Value} = \text{round}\left(\frac{V_{IN} - V_{REFLO}}{V_{REFH} - V_{REFLO}} \times 4096 \times 8\right)$$

V_{IN} = Applied voltage at the input pin

V_{REFH} and V_{REFLO} = Voltage at the external reference pins on the device (typically $V_{REFH} = V_{SS}$ and $V_{REFLO} = V_{DD}$)

NOTE:

The 12-bit result is rounded to the nearest LSB.

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 its magnitude, as read from the data bus, is now 32760.

2.6.1.2 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 converting differential measurements, the following formula is useful:

$$\text{Differential Value} = \text{round}\left\{\left(\frac{V_{IN+} - V_{IN-}}{V_{REFH} - V_{REFLO}} \times 2048\right) + 2048\right\} \times 8)$$

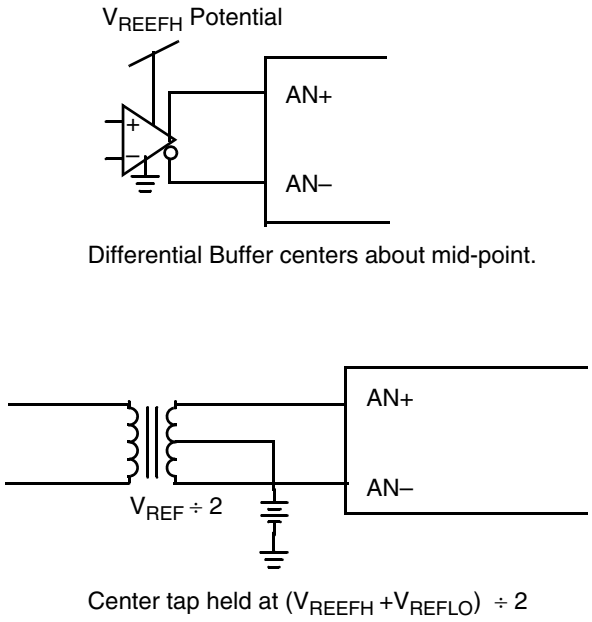
V_{IN} = Applied voltage at the input pin

V_{REFH} and V_{REFLO} = Voltage at the external reference pins on the device (typically $V_{REFH} = V_{SS}$ and $V_{REFLO} = V_{DD}$)

NOTE:

The 12-bit result is rounded to the nearest LSB.

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 its magnitude, as read from the data bus, is now 32760.



Note: Normally V_{REFLO} is set to VSS = 0V.

Figure 2-6. Typical Connections for Differential Measurements

2.6.2 ADC Data Processing

Figure 2-7 illustrates the result of the ADC conversion process is normally sent to an adder for offset correction. The adder subtracts the OFFST*n* (OFFST0-7) register value from each sample and the resultant value is then stored in the result*n* (RSLT0-15) registers. At the same time, the raw ADC value and the RSLT*n* 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 OFFST*n* register value. If the OFFST*n* register is programmed with a value of zero, the RSLT*n* register value is unsigned and equals the cyclic converter unsigned result. The range of the RSLT*n* register is $0000–$7FF8 assuming the entire OFFST*n* register is set to *zero*. This is equal to the raw value of the ADC core.

The RSLT*n* registers used for the results of a scan may be written to by the processor when the STOP bit for that scan is asserted. This write operation is treated as if it came from the ADC analog core; therefore, the limit checking, zero crossing, and the OFFST*n* 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 muxed 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*n* definition and does not include the sign bit.

NOTE:

Offset adjustments and limit comparisons are only available on result registers 0 – 7. Result registers 15 – 8 only store raw ADC conversion data.

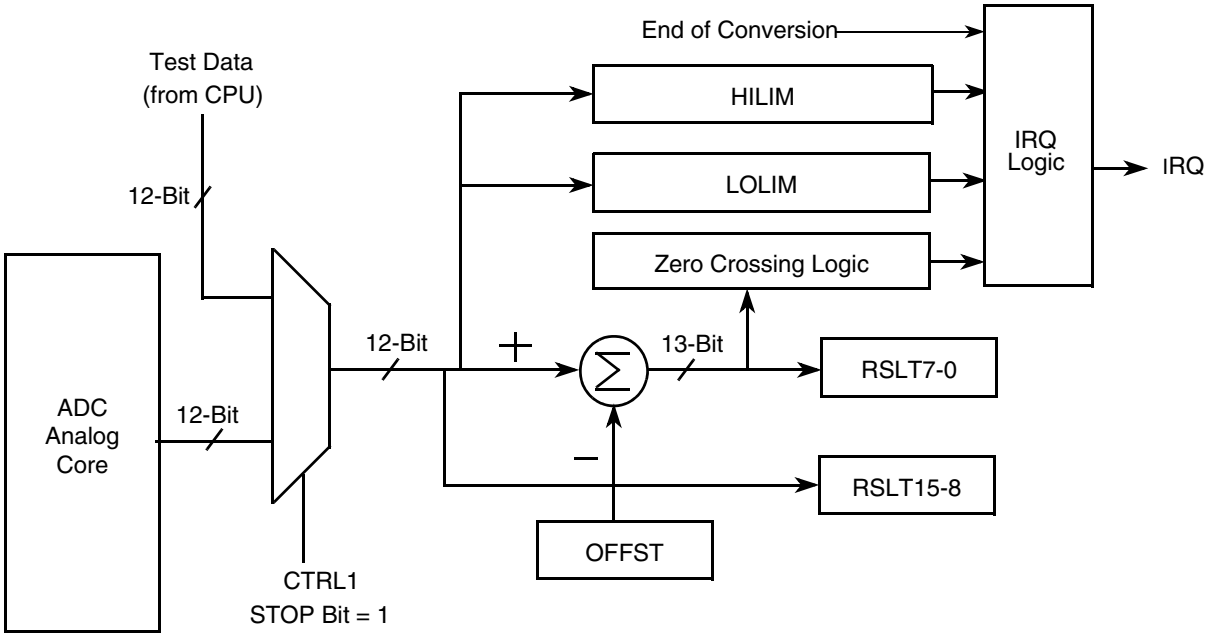


Figure 2-7. Result Register Data Manipulation

2.6.3 Sequential vs. Parallel Sampling

All scan modes make use of the 16 sample slots in the CLST*n* registers. Slots are used to define which input or differential pair to measure at each step in a scan sequence. The SDIS register is used to define which of these sample slots are enabled. Input pairs ANA0/1, ANA2/3, ANA6/7, ANA4/5, ANB0/1, and ANB2/3, ANB4/5, ANB6/7 can be set to be measured differentially using the CHNCFG_L/H bit field. If a sample refers to an input 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. Refer to the CHNCFG_L/H bit field description for details of differential and single-ended measurement.

Scan modes are either sequential or parallel. In sequential scans, up to 16 sample slots are sampled one at a time in order SAMPLE 0-15. Each sample may refer to any of the 16 analog inputs ANA0-ANB7, thus the same input may be referenced by more than one sample slot. Scanning is initiated when the START0 bit is written one or, if the SYNC0 bit is one, when 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 EOSI0 interrupt provided the EOSIEN0 interrupt enable is set. The START0 bit and SYNC0 input are ignored while a scan is in process. Scanning stops and cannot be initiated when the STOP0 bit is set.

Parallel scans differ in that converter A performs up to eight samples (SAMPLE 0-3, SAMPLE 8-11) in parallel to converter B (SAMPLE 4-7, SAMPLE12-15). SAMPLEs 0-3 and SAMPLEs 8-11 may only reference inputs ANA0-7. SAMPLEs 4-7 and SAMPLEs 12-15 may only reference inputs ANB0-7. Within that constraint, any sample may reference any pin and the same input may be referenced by more than one sample slot. By default (when SIMULT = 1), the scans in both converters are initiated when the START0 bit is written one or, if the SYNC0 bit is one, when the SYNC0 input goes high and the scan in both converters terminates when either converter encounters a disabled sample slot. Completion of a scan triggers the EOSI0 interrupt provided the EOSIEN0 interrupt enable is set. Samples are always taken simultaneously in both the A and B converters. Setting the STOP0 bit stops and prevents the initiation of scanning in both converters.

Setting non-simultaneous mode (SIMULT = 0) causes parallel scanning to operate independently in the A and B converter. Each converter has its own set of START*n*, STOP*n*, SYNC*n*, and EOSIEN*n* control bits, SYNC*n* input, EOSI*n* interrupt, and conversion in progress (CIP*n*) status indicators (*n* = 0 for converter A, *n* = 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 may be simultaneous, phase shifted, or asynchronous depending on when scans are initiated on the respective converters. The A and B converter may be of different length (still up to a maximum of eight) and each converter's scan completes when a disabled sample is encountered in that converters sample list only. STOP0 stops the A converter only and STOP1 stops the B converter only. Loop Scan modes iterate independently, the A converter processes input selections from SAMPLE 0-3 and SAMPLE8-11 and the B converter processes input selections from SAMPLE 4-7 and SAMPLE 12-15. Each converter independently restarts its scan after completing its list or encountering a disabled sample slot.

2.6.4 Scan Sequencing

Scan modes breakdown into three types based on how they repeat. See Section 2.6.3 first to understand the operation of sequential and parallel scan modes and their related controls. These types are:

1. Once scan modes execute a sequential or parallel scan only one time each time it is started, but differ from the triggered scan modes in that sync inputs must be re-armed after each use. All scan modes ignore sync pulses occurring while a scan is in process. Once scan modes continue to ignore sync pulses even after the scan completes until the sync input is re-armed. Re-arming, however, can occur any time including during the scan. The SYNC0 input is re-armed by writing to the CTRL1 register and the SYNC1 input is re-armed by writing to the CTRL2 register. Re-arming can be done any time after a scan has started. Re-arming, however, can occur any time including during the scan by writing to a CTRL*n* register. If operating in a sequential or simultaneous parallel mode write to the CTRL1 register. If operating in a non-simultaneous parallel mode, re-arm converter A by writing to the CTRL1 register and converter B by writing to the CTRL2 register.
2. Triggered scan modes are identical to the corresponding once scan modes except re-arming of sync inputs is not necessary.
3. Loop scan modes automatically restart a scan, either parallel or sequential, as soon as the previous scan completes. In parallel loop 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. Scanning can only be terminated by setting the STOP bit.

2.6.5 Power Management

The simplest power management technique is to turn off ADCs not being used. This is described in Section 2.6.5.1. The five supported power modes are described below. They are presented in order from highest to lowest power utilization at the expense of increased conversion latency and/or startup delay. It is recommended changes to the SIM and OCCS affecting the power modes be made while the PD0 and PD1 bits are both asserted. See the Clocks section for details of the various clocks referenced below.

2.6.5.1 Manual Power Down of Unused Converters

If the channel list registers (see Section 2.6.5) do not use input pins ANB0-ANB3 then ADC B is never used. Please see Figure 2-2. Similarly, if ANA0-ANA3 are not used then ADC A is never used. In such cases the unused ADC can be manually powered down using the PD0 or PD1 bit in the power control register. Please see Section 2.7.13.

Since the default value of the PD0 and PD1 bits is powered down, this technique can be used in most applications by simply not powering up the unused ADC.

2.6.5.2 Power Management Mode

1. NORMAL POWER MODE

   This mode operates when:

   a) At least one ADC converter is powered up (PD0 or PD1 = 0 in the PWR register);

   b) Both auto power down and auto standby modes are disabled (APD = 0, ASB = 0 in ADCPOWER);

   c) The ADC's clock is enabled (ADC = 1 in the SIM module's SIM_PCE register).

   In this mode the ADC uses the conversion clock as the ADC clock source both when active or idle. To minimize conversion latency it is recommended the conversion clock be configured to 5.33 MHz. No startup delay (defined by PUDELAY in the PWR register) is imposed.

2. AUTO POWER DOWN MODE

   This mode operates when:

   a) At least one ADC converter is powered up (PD0 or PD1 = 0 in the PWR register);

   b) Auto power down mode enabled (APD = 1 in the PWR register);

   c) The ADC's clock is enabled (ADC = 1 in the SIM module's SIM_PCE register).

   It is recommended the conversion clock be configured at or near 5.33 MHz to minimize conversion latency when active. In this mode, the ADC uses the conversion clock when active and gates off the conversion clock and powers down the converters when idle. A startup delay of PUDELAY ADC clocks is executed at the start of all scans, allowing the ADC to stabilize when switching to normal current mode from a completely powered off condition. This mode uses less power than normal and more power than auto standby. It requires more startup latency (than auto standby) when leaving the idle state to start a scan (higher PUDELAY value).

3. AUTO STANDBY MODE

   This mode operates when:

   a) At least one ADC converter is powered up (PD0 or PD1 = 0 in the PWR register)

   b) Auto power down is disabled (APD = 0 in the PWR register)

   c) Auto standby is enabled (ASB = 1 in the PWR register)

   d) The ADC's clock is enabled (ADC = 1 in the SIM module's SIM_PCE register)

   e) The relaxation oscillator is powered up and operating at 8 MHz. (ROPD = ROSB = 0 in OCCS register OCTRL). Even when using an external clock source (PRECS = 1 in ODDS's OCTRL register) the relaxation oscillator is required to generate a 8 MHz reference.

   In auto standby mode, the ADC uses the conversion clock when active and the100 kHz standby clock when idle. The standby (low current) state automatically engages when the ADC is idle. It is recommended the conversion clock be configured at or near 5.33 MHz to minimize conversion latency. The ADC executes a startup delay of PUDELAY ADC clocks at the start of all scans, allowing the ADC to switch to the conversion clock and to revert from standby to normal current mode.

4. STANDBY MODE

   This mode operates when:

   a) At least one ADC converter is powered up (PD0 or PD1 = 0 in the PWR register)

   b) Auto standby mode is disabled (ASB = 0 in the PWR register)

   c) The ADC's clock is enabled (ADC = 1 in the SIM module's SIM_PCE register)

   d) The PLL is bypassed (ZSRC = 01 in the OCCS module's CTRL register)

   e) The MSTR_OSC clock is driven from the relaxation oscillator (PRECS = 0 in OCCS's CTRL register) in standby mode (ROSB = 1 and ROPD = 0 in OCCS's OCTRL register) (at 400 kHz).

   In this configuration the ADC clock operates at 100 kHz and standby current mode is enabled without loss of conversion accuracy. While no startup delay (PUDELAY) is imposed for each scan, the latency of a scan is increased by the 100 kHz frequency of the ADC clock. This mode is not available while using an external clock source because there is no way to determine the MSTR_OSC clock is operating at the correct frequency.

   Auto power down and standby modes can be used together by setting APD = 1 in the above configuration. This hybrid mode converts at an ADC clock rate of 100 kHz using standby current mode when active and gates off the ADC clock and powers down the converters when idle. A startup delay of PUDELAY ADC clock cycles execute at the start of all scans while the ADC engages the conversion clock and the ADC powers up, stabilizing in the standby current mode. This provides the lowest possible power configuration for ADC operation.

5. POWER DOWN MODE

   This mode operates when:

   a) Both ADC converters are powered down (PD0 = PD1 = 1 in the PWR register).

   b) The ADC's clock is disabled (ADC = 0 in the SIM module's SIM_PCE register).

In this configuration, the clock trees to the ADC and all of its analog components are shut down and the ADC uses no power.

2.6.6 Power Management Details

The ADC voltage reference and converters are powered down (PD*n* = 1) on reset. Individual converters can be manually powered down when not in use (PD0 = 1 or PD1 = 1) and the voltage reference can be manually powered down when no converter is in use (PD2 = 1). When the ADC reference is powered down the output reference voltages are set to low (V_{SSA}) and the ADC data output is driven low.

A delay of PUDELAY ADC clock cycles is imposed whenever PD0 or PD1 are cleared to power up a regulator and also whenever going from an idle (neither converter has a scan in process) to an active state (at least one converter has a scan in process). Provided ADC data sheets recommend the use of two PUDELAY values, a large value for full power up, a moderate value for going from standby current levels to full power up. Following is an explanation of how to use PUDELAY when starting the ADC up or changing modes.

When starting up in normal mode or standby power mode, first set PUDELAY to the large power up value, then clear auto standby (ASB) and auto power down (APD), then clear the PD0 and or PD1 bits to power up the required converters, then poll the status bits until all required converters are powered up. This provides a full power up delay before scans begin. Scan operations can now be started. Normal mode does not use PUDELAY at start of scan thus no further delay is imposed.

When starting up in normal mode clear the PD0 and or PD1 bits to power up the required converters. Poll the status bits (PSTS*n* in the PWR register) until all required converters are powered up. Following polling, start scan operations.

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

When starting up using APD mode, first set PUDELAY to the large power up value, then clear ASB and set APD, and then clear the PD0 and or PD1 bits for the required converters. Converters remain powered off until scanning goes active then the large PUDELAY is imposed to go from the powered down to fully powered before starting scan operations.

It is recommended to power off both regulators (PD0 = PD1 = 1) when re-configuring clocking or power controls to avoid ambiguity and ensure the proper delays are applied when powering up or starting scans.

Attempts to start a scan during the PUDELAY is ignored until the appropriate PSTS*n* bits in the PWR register are cleared.

Any attempt to use a converter when powered down or with the voltage reference disabled results in invalid results. It is possible to read RSLT0:7 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 APD mode, when the ADC goes from idle to active, a converter is only powered up if it is required for the scan as determined by the CLST*n* and SDIS registers.

2.6.7 STOP Mode Operation

Any conversion sequence in progress 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. While in this stop mode, the RSLT*n* registers can be modified by writes from the processor. Any write to the RSLT*n* register in the ADC stop mode is treated as if the analog core supplied the data, therefore, limit checking and zero crossing and associated interrupts can occur if enabled.

2.6.8 ADC Clock

2.6.8.1 General

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

Table 2-2. ADC Clock Summary

Clock Input	Source	Characteristics
Peripheral Clock (=System Clock)	OCCS by way of the SIM	Max rate is 32 Mhz. When PLL is on and selected, it is PLL output divided by 6. When PLL is not selected, it is MSTR_OSC ÷ 2
ADC 8MHz Clock	Relaxation Oscillator	Provides 8 Mhz for auto standby power saving mode.

2.6.8.2 Description of Clock Operation

The peripheral clock rate is determined by the OCCS module configuration. One of two sources (an external clock pin, or the relaxation oscillator) can be selected via PRECS in the OCCS CTRL register to generate the peripheral clock. The maximum rate of the peripheral clock to the ADC is therefore 32 MHz. The ADC is clocked only when the ADC bit in the SIM_PCE register is set. The ADC bit must be enabled while the ADC is in use.

As shown in Figure 2-8, 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 DIV value in the CTRL2 register should be configured so the conversion clock frequency falls between 100 kHz and 5.33 MHz. Operating the ADC at out-of-spec clock frequencies degrades conversion accuracy. Similarly, modifying the parameters affect clock rates or power modes while the regulators are powered up (PD0 = 0 or PD1 = 0) also degrades conversion accuracy.

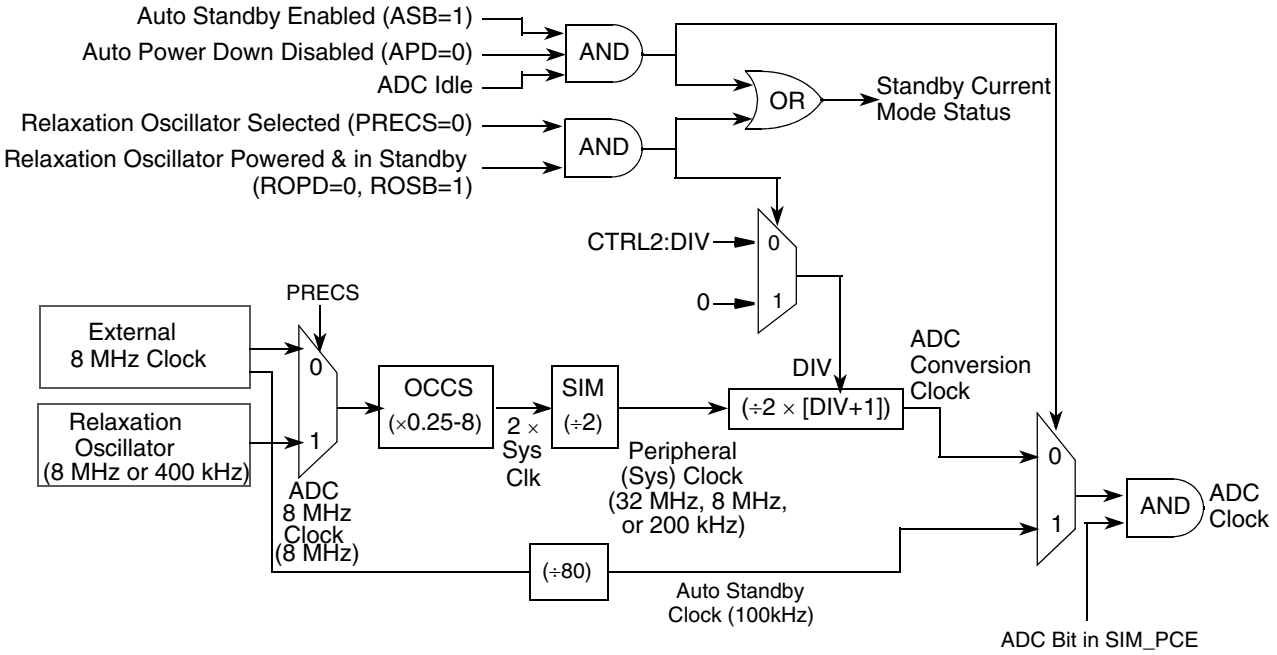


Figure 2-8. ADC Clock Generation

The conversion clock ADC uses for sampling is calculated using the IPBus clock and the clock divisor bits within the CTRL2 register. Please see Section 2.7.1 or Section 2.7.2. The ADC clock is active 100 percent of the time while in loop modes, or if power management is set to normal. It is also active during all ADC power up for a period of time determined by the PUDELAY field in the power (PWR) register. After the power up delay times out, the ADC clock continues until the completion of the ADC*n* scan when operating in auto standby or auto power down modes.

The ADC 8 MHz clock feeds a 80:1 divider, generating the auto standby clock. The auto standby clock is selected as the ADC clock during the auto standby power mode when both converters are idle. The auto standby power mode requires the relaxation oscillator to be powered and normal mode (ROPD ROSB=0 in OCCS's OCTRL register).

2.6.8.3 ADC Clock Re-synchronization at Start of Scan

At the fastest ADC speed, each ADC clock period is six system clock periods long. When asserting the start of a scan, either by writing to a START*n* bit or by a SYNC*n* signal, the ADC clock is re-synchronized to align it to the system clock. This allows the commanded scan to begin as soon as possible rather than wait up to five additional system clocks for the start of the next ADC clock period. This is shown in Figure 2-9 for both sequential and simultaneous parallel modes of operation. In these modes both ADC operate off of the same start signal.

In a parallel scan mode when SIMULT = 0 both ADCs operate using independent START*n* bits and SYNC*n* signals. Figure 2-10 illustrates the first scan started is re-synchronized to the system clock but the second scan may wait up to five additional system clocks before starting. Also, please note which converter is synchronized to the

system clock depends on which convert first starts to use the ADC. The case shown has ADCA synchronized, but one could easily imagine the case where the ADCA start comes after instead of before the ADCB start. In this case ADCAs start would be delayed up to five additional system clock periods instead of ADCBs.

If there is a known timing relationship between ADCA and ADCB when operating in a non-simultaneous parallel mode then the application can control which ADC starts first and gets the re-synchronized clock. The application can also control the delay to starting the second ADC scan so its start signal aligns with the ADC clock and the start of the second ADC is not delayed.

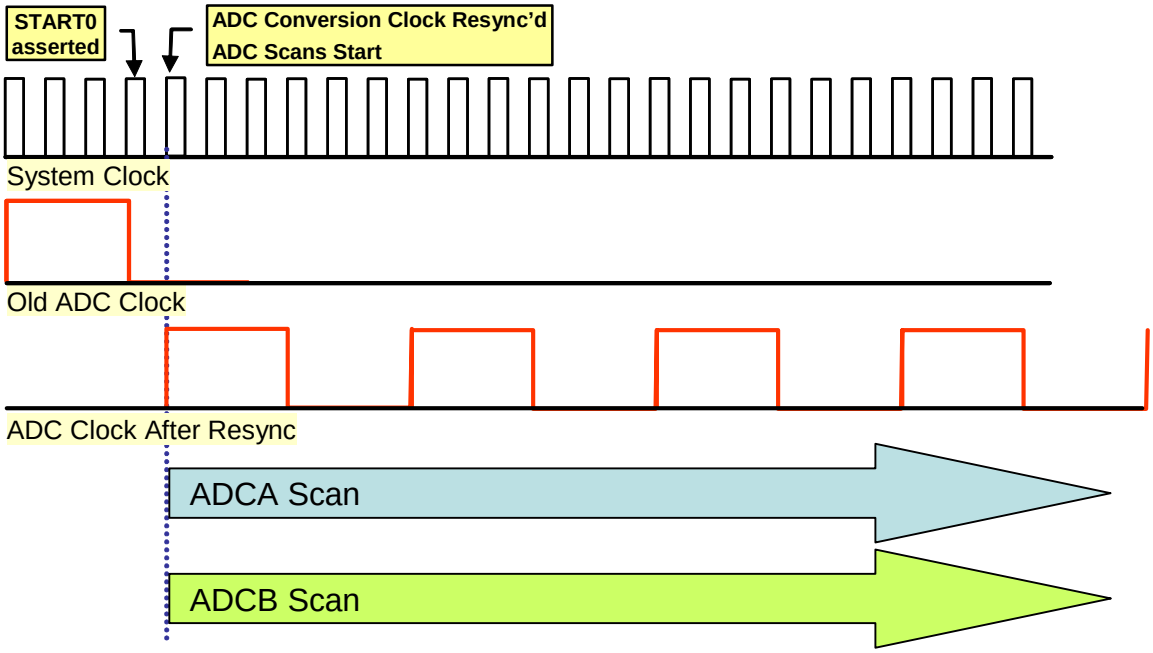


Figure 2-9. ADC Clock Re-synchronization for Sequential and Simultaneous Parallel Modes

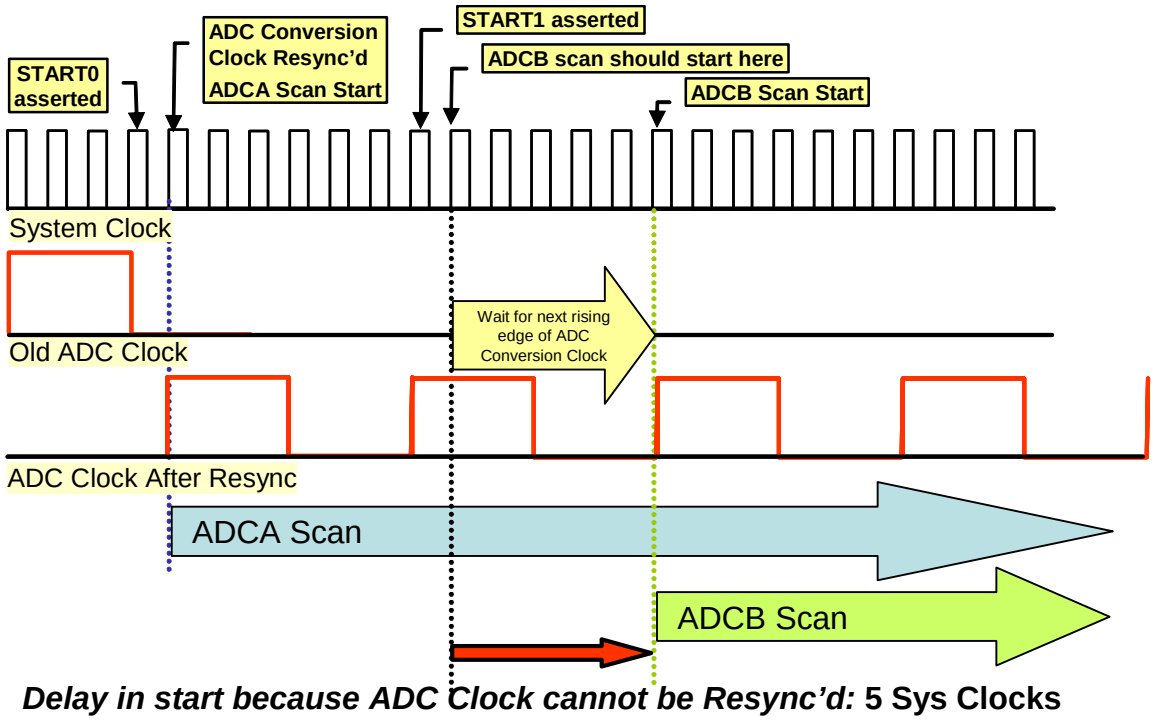


Figure 2-10. ADC Clock Re-synchronization for Non-Simultaneous Parallel Modes

2.6.9 Voltage Reference Pins V_{REFH} & V_{REFLO}

The voltage difference between V_{REFH} and V_{REFLO} provides the reference voltage all analog inputs are measured against. V_{REFH} is nominally set to V_{DDA}. V_{REFLO} is nominally set to V_{SSA}. An external reference voltage should be provided from a low noise filtered source capable of providing up to 1 mA of reference current.

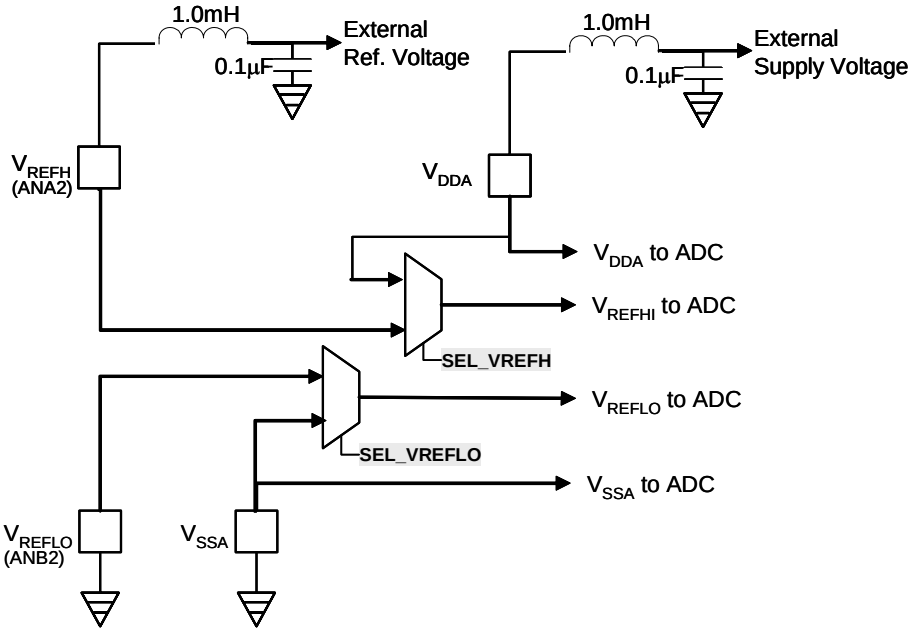


Figure 2-11. ADC Voltage Reference Circuit

When tying V_{REFH} to the same potential as V_{DDA} relative measurements are being made with respect to the amplitude of V_{DDA}. It is imperative special precautions be taken assuring the voltage applied to V_{REFH} be as noise free as possible. Any noise residing on the V_{REFH} voltage is directly transferred to the digital result.

Figure 2-11 illustrates the internal workings of the ADC voltage reference circuit.V_{REFH} must be noise filtered; a minimum configuration is shown in the figure.

2.6.10 Supply Pins V_{DDA} and V_{SSA}

Dedicated power supply pins are provided for the purposes of reducing noise coupling and to improve accuracy. The power provided to these pins is suggested to come from a low noise filtered source. Uncoupling capacitors ought to be connected between V_{DDA} and V_{SSA}.

2.7 Register Description

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. The address offset is defined at the module level. The ADC bit in the SIM module's SIM_PCE register must be one before the ADC registers can be changed.

The ADC and OCCS should not be re configured during scan operations. Allowed accesses during scan operations include:

- Reading status
- Reading conversion results
- Clearing interrupts
- Clearing zero crossing and limit status flags
- Starting/stopping scans using START and STOP bits.

Re-configuring during scan operations does not damage the part but can lead to unpredictable results.

Table 2-3. ADC Memory Map

Device	Peripheral	Base Address
56F80xx	ADC	$00F080

Table 2-4 lists the ADC registers in ascending address order, including their acronyms and address offset of each register.

Table 2-4. ADC Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL1	Control 1 register	Read/Write	Section 2.7.1
Base + $1	CTRL2	Control 2 register	Read/Write	Section 2.7.2
Base + $2	ZXCTRL	Zero crossing control register	Read/Write	Section 2.7.3
Base + $3 – $6	CLIST1–4	Channel list 1–4 registers	Read/Write	Section 2.7.4
Base + $7	SDIS	Sample disable register	Read/Write	Section 2.7.5
Base + $8	STAT	Status register	Read/Write	Section 2.7.6
Base + $9	RDY	Conversion ready register	*Read-Only*	Section 2.7.7
Base + $A	LIMSTAT	Limit status register	Read/Write	Section 2.7.8
Base + $B	ZXSTAT	Zero crossing status register	Read/Write	Section 2.7.9
Base + $C – $13	RSLT0–7	Result 0–7 registers	Read/Write	Section 2.7.10
Base + $14 – $1B	RSLT8–15	Result 8–15 registers	Read/Write	Section 2.7.11
Base + $1C – $23	LOLIM0–7	Low limit 0–7 registers	Read/Write	Section 2.7.12
Base + $24 – $2B	HILIM0–7	High limit 0-7 registers	Read/Write	Section 2.7.12
Base + $2C – $33	OFFST0–7	Offset 0–7 registers	Read/Write	Section 2.7.13
Base + $34	PWR	Power control register	Read/Write	Section 2.7.14
Base + $35	CAL	Calibration register	Read/Write	Section 2.7.15

There are 44 registers in the ADC peripheral summarized in Figure 2-12. Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL1	R	0	STOP0	0	SYNC0	EOSIE0	ZCIE	LLMTIE	HLMTIE	CHNCFG_L				0	SMODE		
		W			START0													
$1	CTRL2	R	0	STOP1	0	SYNC1	EOSIE1	0	CHNCFG_H				SIMULT	DIV				
		W			START1													
$2	ZXCTRL	R	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
		W																
$3	CLIST1	R	SAMPLE3				SAMPLE2				SAMPLE1				SAMPLE0			
		W																
$4	CLIST2	R	SAMPLE7				SAMPLE6				SAMPLE5				SAMPLE4			
		W																
$5	CLIST3	R	SAMPLE11				SAMPLE10				SAMPLE9				SAMPLE8			
		W																
$6	CLIST4	R	SAMPLE15				SAMPLE14				SAMPLE13				SAMPLE12			
		W																
$7	SDIS	R	DS15	DS14	DS13	DS12	DS11	DS10	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
		W																
$8	STAT	R	CIP0	CIP1	0	EOSI1	EOS10	ZCI	LLMTI	HLMTI	RDY							
		W																
$9	RDY	R	RDY															
		W																
$A	LIMSTAT	R	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0
		W																
$B	ZXSTAT	R	0	0	0	0	0	0	0	0	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
		W																
$C-13	RSLT0-7	R	SEXT	RSLT												0	0	0
		W		TEST_DATA														
$14-1B	RSLT8-15	R	0	RSLT												0	0	0
		W		TEST_DATA														
$1C-23	LOLIM0-7	R	0	LLMT												0	0	0
		W																
$24-2B	HILIM0-7	R	0	HLMT												0	0	0
		W																
$2C-33	OFFST0-7	R	0	OFFSET												0	0	0
		W																
$34	PWR	R	ASB	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						APD	PD2	PD1	PD0
		W																
$35	CAL	R	SEL_VREFH_1	SEL_VREFLO_1	SEL_VREFH_0	SEL_VREFLO_0	0	0	0	0	0	0	0	0	SEL_TEST_1	SEL_TEST_0	SEL_DAC1	SEL_DAC0
		W																
		R	0	Read as 0														
		W		Reserved														

Figure 2-12. ADC Register Map Summary

2.7.1 Control 1 (CTRL1) Register

This register controls all types of scans except parallel scans in the B converter when 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 register are used to control B converter scans in non-simultaneous parallel scan modes.

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	STOP0	0	SYNC0	EOSIE0	ZCIE	LLMTIE	HLMTIE	CHNCFG_L				0	SMODE		
Write			START0													
Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Figure 2-13. Control 1 (CTRL1) Register

2.7.1.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.1.2 Stop (STOP0)—Bit 14

When STOP0 is asserted, the current scan is stopped and no further scans can begin. Any further SYNC0 input pulses (see SYNC0 bit) or writes to the START0 bit are ignored until the STOP0 bit is cleared. After the ADC is in stop mode, the RSTL*n* registers can be modified by the processor. Any changes to the RSTL*n* 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 DSC STOP mode.

- 0 = Normal operation
- 1 = Stop mode

2.7.1.3 Start Conversion (START0)—Bit 13

A scan is started by writing one to the START0 bit. This is a write-only bit. Writing one to the START0 bit again while the scan remains in process, is ignored.

- 0 = No action
- 1 = Start command is issued

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.

2.7.1.4 SYNC0 Enable (SYNC0)—Bit 12

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.

- 0 = Scan is initiated by a write to START0 bit only
- 1 = Use a SYNC0 input pulse or START0 bit to initiate a scan

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. Subsequent SYNC0 input pulses are ignored until the SYNC0 input is re-armed by writing to the CTRL1 register. This can be done at any time, including while the scan remains in process.

2.7.1.5 End Of Scan Interrupt 0 Enable (EOSIE0)—Bit 11

This bit enables an EOSI0 interrupt to be generated upon completion of the scan. For loop scan modes, the interrupt triggers after the completion of each iteration of the loop.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

2.7.1.6 Zero Crossing Interrupt Enable (ZCIE)—Bit 10

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

2.7.1.7 Low Limit Interrupt Enable (LLMTIE)—Bit 9

This bit enables the low limit exceeded interrupt when the current result value is less than the low limit (LOLIM*n*) register value. The raw result value is compared to the LOLIM*n* register, bit field LLMT, before the offset (OFFST*n*) register value is subtracted.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

2.7.1.8 High Limit Interrupt Enable (HLMTIE)—Bit 8

This bit enables the high limit exceeded interrupt if the current result value is greater than the high limit (HILIM*n)* register value. The raw result value is compared to the HILIM register, bit field HLMT, before the offset (OFFST*n*) register value is subtracted.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

2.7.1.9 Channel Configure Low (CHNCFG_L)—Bits 7–4

The bits configure the analog inputs for either single ended or differential conversions.

Table 2-5. CHNCFG_L Bit Settings

Bit Settings	Inputs	Description
xxx1	ANA0 – ANA1	Configured as differential pair (ANA0 is + and ANA1 is –)
xxx0		Both configured as single-ended inputs
xx1x	ANA2 – ANA3	Configured as differential pair (ANA2 is + and ANA3 is –)
xx0x		Both configured as single inputs
x1xx	ANB0 – ANB1	Configured as differential pair (ANB0 is + and ANB1 is –)
x0xx		Both configured as single-ended inputs
1xxx	ANB2 – ANB3	Configured as differential pair (ANB2 is + and ANB3 is –)
0xxx		Both configured as single-ended inputs

Differential measurements return the max value (2^{12} –1) when the positive input is V_{REFH} and the negative input is V_{REFLO}, return zero when the + input is at V_{REFLO} and the negative 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 zero when the input is at V_{REFLO}, and scale linearly between based on the amount by which the input exceeds V_{REFLO}.

2.7.1.10 Scan Mode Control (SMODE)—Bits 2-0

SMODE controls the scan mode of the ADC module. All scan modes make use of 16 sample slots defined by the CLIST*n* registers. A scan is the process of stepping through a subset of these sample slots, converting the input indicated by that 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, and ANB2:3, ANB4:5, and ANB6:7 may be configured as differential pairs using the CHNCFG_H/L bit field. 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 details of differential and single ended measurement are described in the description of the CHNCFG bit field.

The SMODE 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 one time or repetitively. See Section 2.6.3 for functional descriptions of parallel and sequential scan modes and of scan sequencing.

Parallel scans may be simultaneous (SIMULT = 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 converter independently with each converter using its own independent set of controls. Refer to the SIMULT bit description for further details. Setting any sequential mode overrides the setting of SIMULT.

- 000 = Once Sequential—Upon start, or an enabled sync signal, samples are taken one at a time starting with SAMPLE0, until the first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after SAMPLE15. If the scan is initiated by a SYNC signal only one scan is completed until the converter is rearmed by writing to the CTRL1 register.
- 001 = Once Parallel—Upon start, or an armed and enabled sync signal, converter A converts SAMPLEs 0:3, SAMPLEs 8:11 and converter B converts SAMPLEs 4:7, SAMPLEs 12:15 in parallel. By default (SIMULT = 1), scanning stops when either converter encounters a disabled sample or both converters complete all 16 samples. When SIMULT = 0, scanning stops in a converter when that converter encounters a disabled sample or that converter completes all eight of its samples. If the scan is initiated by a SYNC signal, only one scan is completed until the converter is rearmed by writing to the CTRL1 register. If SIMULT = 0 then the B converter must be rearmed by writing to the CTRL2 register.

- 010 = Loop sequential—Upon an initial start, or enabled sync pulse, up to 16 samples in order SAMPLEs 0:15 are taken one at a time until a disabled sample is encountered. The process repeats perpetually until the STOP0 bit is set. While a loop mode is running, any additional start commands or sync pulses are ignored. If auto standby (ASB) or auto power down (APD) is the selected power mode control, PUDELAY is only applied on the first conversion.
- 011 = Loop parallel—Upon an initial start, or enabled sync pulse, converter A converts SAMPLEs 0:3, SAMPLEs 8:11 and converter B converts SAMPLEs 4:7, SAMPLEs 12:15. Each time a converter completes it's current scan, it immediately restarts it's scan sequence. This continues until the STOP bit is asserted. While a loop mode is running, any additional start commands or sync pulses are ignored. By default (SIMULT = 1), scanning restarts when either converter encounters a disabled sample. When SIMULT = 0, scanning restarts in a converter when that converter encounters a disabled sample. If auto standby (ASB) or auto power down (APD) is the selected power mode control, PUDELAY is only applied on the first conversion.
- 100 = Triggered sequential—Upon start, or an enabled sync signal, samples are taken one at a time starting with SAMPLE0, until a first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after SAMPLE15. If external sync is enabled new scans are started for each SYNC non-overlapping pulse with a current scan in progress.
- 101 = Triggered parallel (default)—Upon start, or an enabled sync signal, converter A converts SAMPLEs 0:3, SAMPLEs 8:11 and converter B converts SAMPLEs 4:7, SAMPLEs 12:15 in parallel. By default (SIMULT = 1), scanning stops when either converter encounters a disabled sample. When SIMULT = 0, scanning stops in a converter when that converter encounters a disabled sample. If external sync is enabled new scans are started for each non-overlapping SYNC pulse with a current scan in progress.
- 110 = Reserved use
- 111 = Reserved use

2.7.2 Control 2 (CTRL2) Register

Bits 14, 13, 12, and 11 in CTRL2 register, along with the SYNC1 module input are used only to control the B converter during parallel scan modes when SIMULT = 0 (non-simultaneous parallel scan modes). EOSIE1 enables the EOSI1 interrupt used to signal the end of a B converter scan in parallel scan mode when SIMULT = 0. The CIP1 bit is used to indicate a B converter scan is active during parallel scan mode when SIMULT = 0.

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

Figure 2-14. Control 2 (CTRL2) Register

2.7.2.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.2.2 Stop 1 (STOP1)—Bit 14

During parallel scan modes when SIMULT = 0, setting STOP1 stops parallel scans in the B converter and prevents new ones from starting. Any further SYNC1 input pulses (see SYNC1 bit) or writes to the START1 bit are ignored until the STOP1 bit has been cleared. After the ADC is in stop mode, the B converter RSLT*n* registers can be

modified by the processor. Any changes to the RSLT*n* 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 DSC STOP mode.

- 0 = Normal operation
- 1 = Stop command issued

2.7.2.3 Start Conversion 1 (START1)—Bit 13

During parallel scan modes when SIMULT = 0, a B converter parallel scan is started by writing one to the START1 bit. This is a write-only bit. Writing one to the START1 bit again while the scan remains in process, is ignored.

- 0 = No action
- 1 = Start a B converter parallel scan

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.

2.7.2.4 SYNC1 Enable (SYNC1)—Bit 12

During parallel scan modes when SIMULT = 0, setting SYNC1 to one permits a B converter parallel scan to be initiated by asserting the SYNC1 input for at least one ADC clock cycle. Any SYNC1 input pulses while the scan remains in process are ignored.

- 0 = B converter parallel scan is initiated by a write to START1 bit only
- 1 = Use a SYNC1 input pulse or START1 bit to initiate a B converter parallel scan

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. Subsequent SYNC1 input pulses are ignored until the SYNC1 input is re-armed by writing to the CTRL2 register. This may be completed at any time, including while the scan remains in process.

2.7.2.5 End of Scan Interrupt 1 Enable (EOSIE1)—Bit 11

During parallel scan modes when SIMULT = 0, this bit enables an EOSI1 interrupt to be generated upon completion of a B converter parallel scan. For loop scan mode, the interrupt triggers upon the completion of each iteration of the loop.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

2.7.2.6 Reserved—Bit 10

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.2.7 Channel Configure High (CHNCFG_H)—Bits 9–6

The bits configure the analog inputs for either single ended or differential conversions.

Table 2-6. CHNCFG_H Bit Settings

Bit Settings	Inputs	Description
xxx1	ANA4–ANA5	Configured as differential pair (ANA4 is + and ANA5 is –)
xxx0		Both configured as Single-ended inputs
xx1x	ANA6–ANA7	Configured as differential pair (ANA6 is + and ANA7 is –)
xx0x		Both configured as single-ended inputs
x1xx	ANB4–ANB5	Configured as differential pair (ANB4 is + and ANB5 is –)
x0xx		Both configured as single-ended inputs
1xxx	ANB6–ANB7	Configured as differential pair (ANB6 is + and ANB7 is –)
0xxx		Both configured as single-ended inputs

2.7.2.8 Simultaneous Mode (SIMULT)—Bit 5

This bit only affects parallel scan modes. By default (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. START0, 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 EOSI0 triggers if EOSIEN0 is set. The CIP0 status bit indicates a parallel scan is in process.

When 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 one suffix) controlling its operation and report its status. Each converter's scan continues until its sample list is exhausted (eight samples) or a disabled sample in its list is encountered. For loop parallel scan mode, each converter starts its next iteration as soon as the previous iteration in that converter is complete and continues until the STOP bit for that converter is asserted.

- 0 = Parallel scans done independently
- 1 = Parallel scans done simultaneously (default)

2.7.2.9 Clock Divisor Select (DIV)—Bits 4–0

The divider circuit generates the ADC clock by dividing the system clock by 2×DIV[4:0]+1). A DIV value must be chosen so the ADC clock does not exceed 5.33 Mhz. The following table shows ADC clock frequency based on the value of DIV for these various OCCS configurations.

Table 2-7. ADC Clock Frequency for Various Conversion Clock Sources

DIV	Divisor	ROSC Standby 400Khz	ROSC Normal 8Mhz	PLL/64 32 Mhz	External CLK
0_0000	2	100K	2.00M	16.0M	CLK/4
0_0001	4	100K	1.00M	8.00M	CLK/8
0_0010	6	100K	500K	5.33M	CLK/12
0_0011	8	100K	250K	4.00M	CLK/16
0_0100	10	100K	125K	3.20M	CLK/20
—	—	—	—	—	—
—	—	—	—	—	—
1_1111	64	100K	62.5K	500K	CLK/128

2.7.3 ADC Zero Crossing Control (ZXCTRL) Register

The zero crossing control (ZXCTRL) register provides the ability to monitor the selected channels and determine the direction of zero crossing triggering the optional interrupt. Zero crossing logic monitors only the sign change between current and previous sample. ZCE0 monitors the sample stored in RSLT0 and bit ZCE7 monitors RSLT7. When the zero crossing is disabled for a selected RESULT*n*, sign changes are not monitored or updated in the ZXSTAT register.

Zero crossing functionality is only available on the first eight conversions in sequential mode and only available on the first four conversions assoicated with each converter in parallel modes.

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

Figure 2-15. Zero Crossing Control (ZXCTRL) Register

2.7.3.1 Zero Crossing Enable *n* (ZCE*n*)—Bits15–0

For each channel (*n*) setting the ZCE*n* field allows detection of the indicated zero crossing condition.

- 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

2.7.4 Channel List *n* (CLIST1-4) Registers

The channel list (CLIST*n*) registers contain an ordered list of the channels to be converted when the next scan is initiated. If all samples are enabled, in the SDIS register, a sequential scan of inputs proceeds in order of: SAMPLE0 through SAMPLE15. If one of the parallel sampling modes is selected instead, the converter A sampling order is SAMPLE0:3 and the converter B sampling order is SAMPLE4:7, and then SAMPLE8:11 in parallel with SAMPLE12:15.

In sequential conversion mode full functionality, offset subtraction and high/low limit checks, is only available on the first eight conversion slots, SAMPLE0:7. In parallel conversion mode full functionality is only available on the first four conversion slots of each channel, SAMPLE0:3 for converter A and SAMPLE4:7 for converter B.

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

Figure 2-16. Channel List 1 (CLIST1) Register

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

Figure 2-17. Channel List 2 (CLIST2) Register

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

Figure 2-18. Channel List 3 (CLIST3) Register

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

Figure 2-19. Channel List 4 (CLIST4) Register

The value of the SAMPLE*n* field is used to select the input channel to be sampled.

Table 2-8. ADC Input Conversion for Sample Bits

SAMPLEx[3:0]	Single Ended	Differential
0000	ANA0	ANA0+, ANA1–
0001	ANA1	ANA0+, ANA1–
0010	ANA2	ANA2+, ANA3–
0011	ANA3	ANA2+, ANA3–
0100	ANB0	ANB0+, ANB1–
0101	ANB1	ANB0+, ANB1–
0110	ANB2	ANB2+, ANB3–
0111	ANB3	ANB2+, ANB3–
1000	ANA4	ANA4+, ANA5–
1001	ANA5	ANA4+, ANA5–
1010	ANA6	ANA6+, ANA7–
1011	ANA7	ANA6+, ANA7–
1100	ANB4	ANB4+, ANB5–
1101	ANB5	ANB4+, ANB5–
1110	ANB6	ANB6+, ANB7–
1111	ANB7	ANB6+, ANB7–

In sequential modes, the sample slots are converted in order from SAMPLE0 to SAMPLE15. Any sample slot may reference any analog input (may contain a binary value between 0000 - 1111).

In parallel modes, converter A processes sample slots SAMPLE0:3 and SAMPLE8:11 while converter B processes sample slots SAMPLE4:7 and SAMPLE12:15. Because converter A only has access to analog inputs ANA0 through ANA7, sample slots SAMPLE0:3 and SAMPLE8:11 should only contain binary values between 0000 and

0011 or between 1000 and 1011. Likewise, because converter B only has access to analog inputs ANB0 through ANB7, sample slots SAMPLE4:7 should only contain binary values between 0100 and 0111 or between 1100 and 1111. No damage occurs if this constraint is violated but results are undefined.

When inputs are configured as differential pairs, a reference to either analog input in a differential pair by a sample slot implies a differential measurement on the pair. The details of single ended and differential measurement are described under the CHNCFG_H/L bit field.

Sample slots are disabled using the SDIS register.

2.7.5 Sample Disable (SDIS) Register

This register is an extension to the CLST*n*, providing the ability to enable only the desired samples programmed in the SAMPLE0–SAMPLE15 fields. At reset all samples are enabled.

Base + $7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DS15	DS14	DS13	DS12	DS11	DS10	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
Write																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 2-20. Sample Disable (SDIS) Register

2.7.5.1 Disable Sample (DS)—Bits 15–0

The respective SAMPLE*n* field can be enabled or disabled where *n* = 0–15.

- Enable SAMPLE*n*
- Disable SAMPLE*n* and all subsequent samples. Which samples are actually disabled depends on the conversion mode, sequential/parallel, and the value of SIMULT.

2.7.6 Status (STAT) Register

This register provides the current status of the ADC module. RDY*n* bits are cleared by reading their corresponding result*n* (RSLT*n*) registers. HLMTI and LLMTI bits are cleared by writing one to all asserted bits in the limit status (LIMSTAT) register. Likewise, the ZCI bit (bit 10) is cleared by writing one to all asserted bits in the ZXSTAT register. The EOSI*n* bits are cleared by writing one to them.

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

Base + $8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMT	RDY							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-21. Status (STAT) Register

2.7.6.1 Conversion in Progress 0 (CIP0)—Bit 15

This bit indicates if a scan is in progress.

- 0 = Idle state
- 1 = A scan cycle is in progress

The ADC ignores all sync pulses or start commands. This refers to any scan except a B converter scan in non-simultaneous parallel scan modes.

2.7.6.2 Conversion in Progress 1 (CIP1)—Bit 14

This bit indicates if a scan is in progress.

- 0 = Idle state
- 1= A scan cycle is in progress

The ADC ignores all sync pulses or start commands. This refers only to a B converter scan in non-simultaneous parallel scan modes.

2.7.6.3 Reserved—Bit 13

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.6.4 End of Scan Interrupt 1 (EOSI1)—Bit 12

This bit indicates if a scan of analog inputs were completed since the last read of the status register or since a reset. The EOSI1 bit is cleared by writing one to it. This bit cannot be set by software.

- 0 = A scan cycle was not completed, no end of scan IRQ pending
- 1 = A scan cycle was completed, end of scan IRQ pending

In loop 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.

2.7.6.5 End of Scan Interrupt 0 (EOSI0)—Bit 11

This bit indicates if a scan of analog inputs were completed since the last read of the status register, or since a reset. The EOSI0 bit is cleared by writing one to it. This bit cannot be set by software. EOSI0 is the preferred bit to poll for scan completion if interrupts are not enabled.

- 0 = A scan cycle was not completed, no end of scan IRQ pending
- 1 = A scan cycle was completed, end of scan IRQ pending

In loop scan modes, this interrupt is triggered at the completion of each iteration of the loop mode.

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.

2.7.6.6 Zero Crossing Interrupt (ZCI)—Bit 10

If the respective offset (OFFST*n*) register is configured by having a value greater than 0000h, zero crossing checking is enabled. If the OFFST*n* register is programmed with 7FF8h, the result always is less than, or equal to zero. On the other hand, if 0000h is programmed into the OFFST*n* register, the result is always greater than, or equal to zero and no zero crossing can occur because the sign of the result does not change. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan.

The ZCI bit is cleared by writing one to all active ZCS*n* bits.

- 0 = No ZCI interrupt request
- 1 = Zero crossing encountered, IRQ pending if ZCIE is set

2.7.6.7 Low Limit Interrupt (LLMTI)—Bit 9

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.

The LLMTI bit is cleared by writing one to all active LLS*n* bits.

- 0 = No low limit interrupt request
- 1 = Low limit exceeded, IRQ pending if LLMTIE is set

2.7.6.8 High Limit Interrupt (HLMTI)—Bit 8

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.

The HLMTI bit is cleared by writing one to all active HLS*n* bits.

- 0 = No high limit interrupt request
- 1 = High limit exceeded, IRQ pending if HLMTIE is set

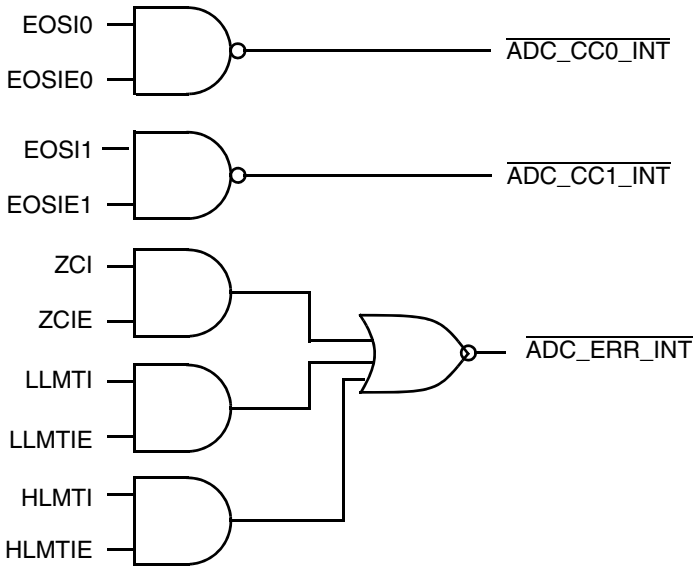


Figure 2-22. ADC Interrupt

2.7.6.9 Ready Sample 7–0 (RDY)—Bits 7–0

These bits indicate samples seven through zero are ready to read. These bits are cleared after a read from the respective results register. The RDY*n* bits are set as the individual channel conversions are completed. If polling the RDY*n* bits to determine if a particular sample is completed care should be taken not to start an new scan until all enabled samples are done.

- 0 = Sample not ready or was read
- 1 = Sample ready to read

2.7.7 Conversion Ready (RDY) Register

This register provides the current status of the ADC conversions. RDY*n* bits are cleared by reading their corresponding result*n* (RSLT*n*) registers

NOTE:

Bits 7 – 0 are equivalent to the RDY bits in the STAT register.

Base + $9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RDY															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-23. Conversion Ready (RDY) Register

2.7.7.1 Ready Sample 15–0 (RDY)—Bits 15–0

These bits indicate samples 15 through 0 are ready to be read. These bits are cleared after a read from the respective RSLT*n* register. The RDY*n* bits are set as the individual channel conversions are completed. If polling the RDY*n* bits to determine if a particular sample is completed care should be taken not to start an new scan until all enabled samples are done.

- 0 = Sample not ready or has been read
- 1 = Sample ready to be read

2.7.8 Limit Status (LIMSTAT) Register

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

Base + $A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HLS7	HLS6	HLS5	HLS44	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-24. Limit Status (LIMSTAT) Register

2.7.9 Zero Crossing Status (ZXSTAT) Register

The zero crossing status (ZXSTAT) register latches in the result of the comparison between the current result of the sample and the previous result of the same sample register. For example, if the result for the channel programmed in SAMPLE0 changes sign from the previous conversion and the respective ZCE*n* bit in register ZXSTAT is set to 11b, any edge change, then the ZCS0 bit is set to one. An interrupt is generated if the ZCIE bit is set in CTRL1 register. A bit can only be cleared by writing a value of one to that specific bit. These bits are sticky. When set, they require a write to clear them. They are not cleared automatically by subsequent conversions.

Base + $B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-25. Zero Crossing Status (ZXSTAT) Register

2.7.9.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.9.2 Zero Crossing Status (ZCS)—Bits 7–0

The zero crossing condition is determined by examining the ADC value after it is adjusted by the offset for the RSLT*n* register. Please see Figure 2-7. Each bit of the register is cleared by writing one to that register bit.

- 0 = A sign change did not occur in a comparison between the current channel*n* result and the previous channel result, or zero crossing control is disabled for channel*n* in the zero crossing control (ZXCTRL) register
- 1 = In a comparison between the current channel*n* result and the previous channel*n* result, a sign change condition occurred as defined in the zero crossing control (ZXCTRL) register

2.7.10 Result 0–7 (RSLT0–7) Registers

These eight RSLT registers contain the converted results from a scan. The SAMPLE0 result is loaded into RSLT0, SAMPLE1 develop in RSLT1, and so on. In a parallel scan mode, the first channel pair designated by SAMPLE0 and SAMPLE4 in register CLST1:2 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, illustrated in Figure 2-7 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.

ADC Result Register 0 (RSTL0)—Address: BASE + $C
ADC Result Register 1 (RSTL1)—Address: BASE + $D
ADC Result Register 2 (RSTL2)—Address: BASE + $E
ADC Result Register 3 (RSTL3)—Address: BASE + $F
ADC Result Register 4 (RSTL4)—Address: BASE + $10
ADC Result Register 5 (RSTL5)—Address: BASE + $11
ADC Result Register 6 (RSTL6)—Address: BASE + $12
ADC Result Register 7 (RSTL7)—Address: BASE + $13

Base + $C-13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SEXT	RSLT												0	0	0
Write		TEST_DATA														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-26. Result 0–7 (RSLT0–7) Register

2.7.10.1 Sign Extend (SEXT)—Bit 15

SEXT bit is the sign-extend bit of the result. SEXT set to one implies a negative result. SEXT set to zero implies a positive result. If all positive results are required, the respective OFFST*n* register must be set to a value of zero.

2.7.10.2 Digital Result of the Conversion (RSLT)—Bits 14–3

RSLT*n* can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the RSLT*n* can be used directly. As a signed integer, it is an option to right shift with sign extend, arithmetic shift right (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, SEXT = 1, are always presented in two's compliment format. If it is a requirement of the application of the RSLT*n* registers always be positive, the OFFST*n* registers must always be set to zero.

The interpretation of the numbers programmed into the LOLIM, HILIM, and OFFST*n* registers should match the interpretation of the RSLT register.

2.7.10.3 Test Data (TEST_DATA)—Bits 14–3

See Section 2.6.2 for a description of this field and how/when it can be used.

2.7.10.4 Reserved—Bits 2–0

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.11 Result 8–15 (RSLT8–15) Registers

These eight RSLT registers contain the converted results from a scan. The SAMPLE8 result is loaded into RSLT8, SAMPLE9 result in RSLT9, and so on. In a parallel scan mode, the first channel pair designated by SAMPLE8 and SAMPLE12 in register CLST3:4 are stored in RSLT8 and RSLT12, respectively.

NOTE:

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

ADC Result Register 8 (RSTL8)—Address: BASE + $14
ADC Result Register 9 (RSTL9)—Address: BASE + $15
ADC Result Register 10 (RSTL10)—Address: BASE + $16
ADC Result Register 11 (RSTL11)—Address: BASE + $17
ADC Result Register 12 (RSTL12)—Address: BASE + $18
ADC Result Register 13 (RSTL13)—Address: BASE + $19
ADC Result Register 14 (RSTL14)—Address: BASE + $1A
ADC Result Register 15 (RSTL15)—Address: BASE + $1B

Base + $14-1B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	RSLT												0	0	0
Write		TEST_DATA														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-27. Result 8–15 (RSLT8–15) Registers

2.7.11.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.11.2 Digital Result of the Conversion (RSLT)—Bits 14–3

RSLT is the raw conversion results, no offset calculation has been applied.

2.7.11.3 Test Data (TEST_DATA)—Bits 14–3

See Section 2.3 for a description of this field and how/when it can be used.

2.7.11.4 Reserved—Bits 2–0

This bit field is reserved or not implemented. It is read as zero and cannot be modified writing.

2.7.12 Low Limit (LOLIM0–7) and High Limit (HILIM0–7) Registers

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. Please refer to Figure 2-7. The limit registers used corresponds to the RSLT registers to which the value is written. 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 $0000 for the low limit. At reset, limit checking is disabled.

ADC Low Limit Register 0 (LLIM0)—Address: BASE + $1C
ADC Low Limit Register 1 (LLIM1)—Address: BASE + $1D
ADC Low Limit Register 2 (LLIM2)—Address: BASE + $1E
ADC Low Limit Register 3 (LLIM3)—Address: BASE + $1F
ADC Low Limit Register 4 (LLIM4)—Address: BASE + $20
ADC Low Limit Register 5 (LLIM5)—Address: BASE + $21
ADC Low Limit Register 6 (LLIM6)—Address: BASE + $22
ADC Low Limit Register 7 (LLIM7)—Address: BASE + $23

Base + $1C-23	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	LLMT												0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-28. Low Limit 0–7 (LOLIM0–7) Register

ADC High Limit Register 0 (HLIM0)—Address: BASE + $24
ADC High Limit Register 1 (HLIM1)—Address: BASE + $25
ADC High Limit Register 2 (HLIM2)—Address: BASE + $26
ADC High Limit Register 3 (HLIM3)—Address: BASE + $27
ADC High Limit Register 4 (HLIM4)—Address: BASE + $28
ADC High Limit Register 5 (HLIM5)—Address: BASE + $29
ADC High Limit Register 6 (HLIM6)—Address: BASE + $2A
ADC High Limit Register 7 (HLIM7)—Address: BASE + $2B

Base + $24-2B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0													0	0	0
Write																
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

Figure 2-29. High Limit 0–7 (HILIM0–7) Register

2.7.13 Offset (OFFST0–7) Registers

Value of the OFFST0:7 registers are used to correct the ADC result before it is stored in the RSLT0:7 registers.

ADC Offset Register 0 (OFFST0)—Address: BASE + $2C
ADC Offset Register 1 (OFFST1)—Address: BASE + $2D
ADC Offset Register 2 (OFFST2)—Address: BASE + $2E
ADC Offset Register 3 (OFFST3)—Address: BASE + $2F
ADC Offset Register 4 (OFFST4)—Address: BASE + $30
ADC Offset Register 5 (OFFST5)—Address: BASE + $31
ADC Offset Register 6 (OFFST6)—Address: BASE + $32
ADC Offset Register 7 (OFFST7)—Address: BASE + $33

Base + $2C-33	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

Figure 2-30. Offset 0–7 (OFFST0–7) Registers

The offset value is subtracted from the ADC result. In order to obtain unsigned results, the respective OFFST*n* register should be programmed with a value of $0000, thus giving a result range of $0000 to $7FF8.

2.7.14 Power Control (PWR) Register

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 generator. 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 the 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. The voltage reference generator and at least one converter must be powered up to use the ADC module.
- Manual power down controls—Each converter and the voltage reference generator have a manual power control bit capable of putting that component into the power down state. Converters have other mechanisms with the capacity to 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 100 kHz. standby current mode uses less power and is engaged only when the ADC clock is at 100 kHz. 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 power down 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.

See the discussion of power modes in Section 2.6.5 for details of the five power modes and how to configure them. See Section 2.8 for a more detailed description of the clocking system and the control of current mode.

Base + $34	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ASB	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						APD	PD2	PD1	PD0
Write																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1

Figure 2-31. Power (PWR) Control Register

2.7.14.1 Auto Standby (ASB)—Bit 15

The ASB bit selects auto standby (ASB) mode. ASB is ignored if APD is set to one. When the ADC is idle, ASB mode selects the standby clock as the ADC clock source, putting 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 PUDELAY ADC clock cycles is imposed for current levels to stabilize. After this delay, the ADC initiates 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.

- 0 = ASB mode disabled
- 1 = ASB mode enabled

NOTE:

This mode is not recommended for conversion clock rates at, or below 100 kHz. Instead, set ASB = APD = 0 and use standby power mode (normal mode with a sufficiently slow conversion clock so standby current mode automatically engages). This provides the advantages of standby current mode while avoiding the clock switching and the PUDELAY.

2.7.14.2 Reserved—Bits 14–13

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.14.3 Power Status 2 (PSTS2)—Bit 12

This read-only bit PSTS2 simply reflects whether the voltage reference circuit is currently powered up.

- 0 = Voltage reference circuit is currently powered up
- 1 = Voltage reference circuit is currently powered down

2.7.14.4 Power Status 11 (PSTS1)—Bit 11

PSTS1 is a read-only bit and is asserted immediately following a write of one to PD1. It is deasserted PUDELAY ADC clock cycles after a write of zero to PD1 if APD is zero. This bit can be read as a status bit to determine when the ADC is ready for operation. During auto power down (APD) 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

2.7.14.5 Power Status 0 (PSTS0)—Bit 10

PSTS0 is a read-only bit and is asserted immediately following a write of one to PD0. It is deasserted PUDELAY ADC clock cycles after a write of zero to PD0 if ADP is zero. This bit can be read as a status bit to determine when the ADC is ready for operation. During APD 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

2.7.14.6 Power Up Delay (PUDELAY)—Bits 9–4

This 6-bit field determines the number of ADC clocks provided to power up an ADC converter (after setting PD0 or PD1 to zero) before allowing a scan to start. It also determines the number of ADC clocks of delay provided in APD and 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 is degraded if PUDELAY is set to too small of a value.

NOTE:

PUDELAY defaults to a value typically sufficient for any power mode. The latency of a scan can be reduced by reducing PUDELAY to the lowest value for which accuracy is not degraded. Refer to the data sheet for further details.

2.7.14.7 Auto Power Down (APD)—Bit 3

Auto power down (APD) mode powers down converters when not in use for a scan. APD takes precedence over ASB. When a scan is started in APD mode, a delay of PUDELAY ADC clock cycles is imposed during which the needed converter(s), if idle, are powered up. The ADC then initiates a scan equivalent to that when APD is not active. When the scan is completed, the converter(s) are powered down again.

- 0 = APD mode is not active
- 1 = APD mode is active

NOTE:

If ASB or APD is asserted while a scan is in progress, that scan is unaffected and the ADC waits to enter its low power state until after all conversions are complete and both ADCs are idle.

ASB and APD are not useful in loop modes. The continuous nature of scanning means the low power state can never be entered.

2.7.14.8 Power Down 2 (PD2)—Bit 2

This bit forces voltage reference circuit to power down.

- 0 = Manually power up voltage reference circuit
- 1 = Power down voltage reference circuit is controlled by PD0 and PD1

The voltage reference circuit is shared by both converters. When PD2 = 1 the voltage reference is activated whenever PD1 or PD0 are powered up.

2.7.14.9 Power Down 1 (PD1)—Bit 1

This bit forces ADC converter B to manually power down.

- 0 = Power up ADC converter B
- 1 = Power down ADC converter B

Asserting PD1 powers down converter B immediately. The results of a scan using converter B is invalid while PD1 is asserted. When PD1 is cleared, converter B is either continuously powered up (APD = 0) or automatically powered up when needed (APD = 1).

When clearing PD1 in any power mode except Auto power down (APD = 1), wait PUDELAY ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PSTS1 bit can be polled to determine when the PUDELAY time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples.

2.7.14.10 Power Down 0 (PD0)—Bit 0

This bit forces ADC converter A to manually power down.

- 0 = Power up ADC converter A
- 1 = Power down ADC converter A

Asserting PD0 powers down converter A immediately. The results of a scan using converter A is invalid while PD0 is asserted. When PD0 is cleared, converter A is either continuously powered up (APD = 0) or automatically powered up when needed (APD = 1).

When clearing PD0 in any power mode except auto power down (APD = 1), wait PUDELAY ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PSTS0 bit can be polled to determine when the PUDELAY time is elapsed. Failure to follow this procedure may result in loss of accuracy of the first two samples.

2.7.15 Calibration (CAL) Register

The ADC provides for off-chip references used for ADC conversions.

Base + $35	15	14	13	12	11	10	9	8
Read	SEL_ VREFH_1	SEL_ VREFLO_1	SEL_ VREFH_0	SEL_ VREFLO_0	0	0	0	0
Write								
Reset	0	0	0	0	0	0	0	0

Base + $35	7	6	5	4	3	2	1	0
Read	0	0	0	0	SEL_TEST1	SEL_TEST0	SEL_DAC1	SEL_DAC0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 2-32. Calibration (CAL) Register

2.7.15.1 Select V_{REFH} Source 1 (SEL_VREFH_1)—Bit 15

This bit selects the source of the V_{REFH} reference for all conversions in converter1.

- 0 = Internal V_{DDA}
- 1 = ANB2

2.7.15.2 Select V_{REFLO} Source 1 (SEL_VREFLO_1)—Bit 14

This bit selects the source of the V_{REFLO} reference for all conversions in converter1.

- 0 = Internal V_{SSA}
- 1 = ANB3

2.7.15.3 Select V_{REFH} Source 0 (SEL_VREFH_0)—Bit 13

This bit selects the source of the V_{REFH} reference for all conversions in converter0.

- 0 = Internal V_{DDA}
- 1 = ANA2

2.7.15.4 Select V_{REFLO} Source 0 (SEL_VREFLO_0)—Bit 12

This bit selects the source of the V_{REFLO} reference for all conversions in converter0.

- 0 = Internal V_{SSA}
- 1 = ANA3

2.7.15.5 Reserved—Bits 11–4

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

2.7.15.6 Select Analog BIST Mode Converter 1 (SEL_TEST1)—Bit 3

This bit forces the analog input select mux on ANB7 to always be in pass-through mode. In this mode additional capacitance from either the internal node or added as an external capacitor can be used to filter the DAC1 output. SEL_DAC1 must have been previously set to one before SEL_TEST1 can be set. Clearing SEL_DAC1 also clears SEL_TEST1.

- 0 = Normal operation
- 1 = ANB7 input mux is always enabled

2.7.15.7 Select Analog BIST Mode Converter 0 (SEL_TEST0)—Bit 2

This bit forces the analog input select mux on ANA7 to always be in pass-through mode. In this mode additional capacitance from either the internal node or added as an external capacitor can be used to filter the DAC0 output. SEL_DAC0 must have been previously set to one before SEL_TEST0 can be set. Clearing SEL_DAC0 also clears SEL_TEST0.

- 0 = Normal operation
- 1 = ANA7 input mux is always enabled

2.7.15.8 Select DAC1 Alternate Source 1 (SEL_DAC1)—Bit 1

This bit selects the source of the ADCB7 input as being either the input pin GPIOC15 or DAC1 output.

- 0 = Normal operation (GPIOC15)
- 1 = ANB7 input is replaced with DAC1 output

2.7.15.9 Select DAC0 Alternate Source 0 (SEL_DAC0)—Bit 0

This bit selects the source of the ADCA7 input as being either the input pin GPIOC11 or DAC0 output.

- 0 = Normal operation (GPIOC11)
- 1 = ANA7 input is replaced with DAC0 output

2.8 Interrupts

Table 2-9. Interrupt Summary

Interrupt	Source	Description
ERR_INT	ADC	Zero crossing, low limit, and high limit interrupt
$\overline{\text{CC0\_INT}}$	ADC	Conversion complete interrupt for any scan type except converter B scan in non-simultaneous parallel scan mode (see EOSI0)
$\overline{\text{CC1\_INT}}$	ADC	Conversion complete interrupt for converter B scan in non-simultaneous parallel scan mode (see EOSI1)

2.8.1 Interrupt Operation Description

Additional bits may need to be set in the interrupt control module to enable the CPU to receive ADC interrupt signals.

2.8.1.1 Threshold Interrupts

This interrupt can occur from three different events:

1. Zero Crossing—occurs if the current result value has a sign change from the previous result as configured by the ZXCTRL register.
2. 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 register, bit field LLMT, before the OFFST register value is subtracted.
3. 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 register, bit field HLMT, before the OFFST register value is subtracted.

All of these interrupts are optional and enabled through CTRL1 register.

2.8.1.2 Conversion Complete Interrupt

The conversion complete interrupt is generated upon completion of any scan and convert sequence when EOSIE0 = 1.

2.9 Resets

Table 2-10. Reset Summary

Reset	Source	Characteristics
RST	SIM	

2.9.1 Reset Operation Description

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

Table 2-11. Document Revision History for Chapter 2

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 3
Computer Operating Properly (COP)

3.1 Introduction

The computer operating properly (COP) module is used to help software recover from runaway code. The COP is a free-running down counter when 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.

3.2 Features

The COP module design includes these distinctive characteristics:

- Programmable timeout period = (cop_prescaler × (TIMEOUT + 1)) × (Selected OSC clock cycles), where TIMEOUT can be from $0000 to $FFFF
- Programmable wait and stop mode operation
- COP timer is disabled while host CPU is in debug mode
- Causes loss of reference reset 128 cycles after loss of reference clock to the PLL is detected
- Choice of clock sources for counter

3.3 Block Diagram

The following is the block diagram of the COP module.

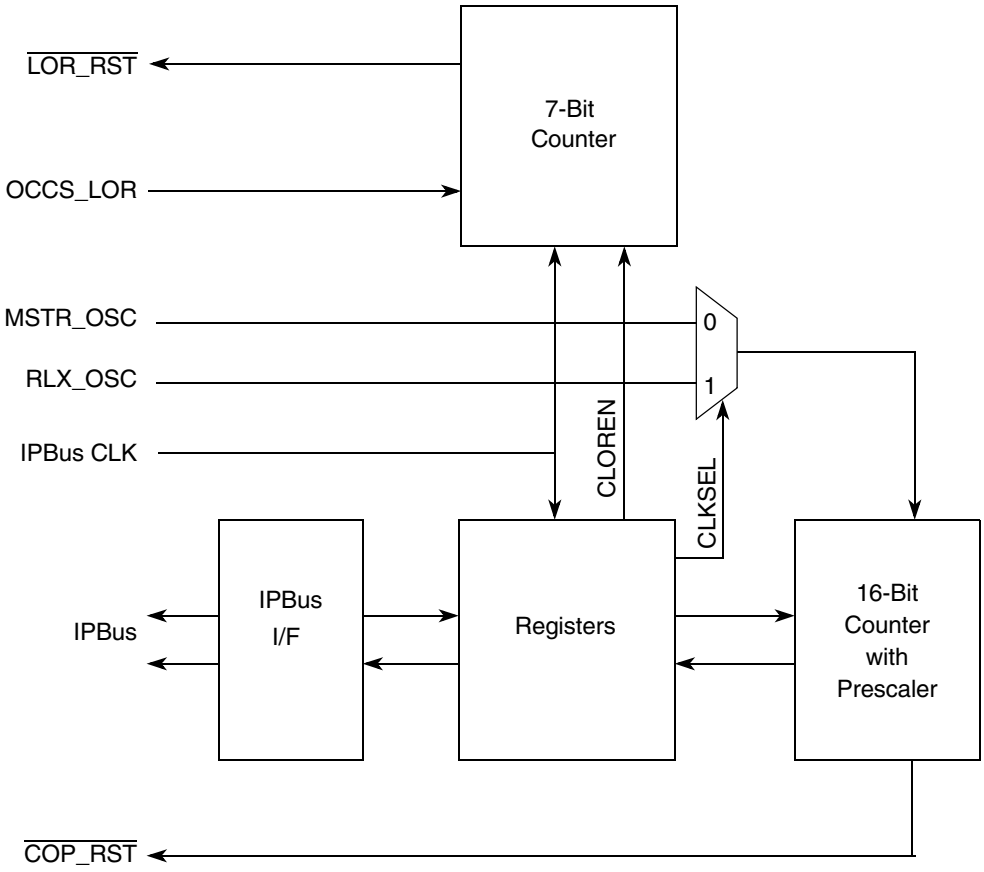


Figure 3-1. COP Block Diagram and Interface Signals

3.4 Functional Description

When the COP is enabled, each positive edge of the prescaled MSTR_OSC (or RLX_OSC when CLKSEL is set) causes the counter to decrement by one. If the count reaches a value of $0000, the $\overline{\text{COP\_RST}}$ signal is asserted and the chip is reset. In order for the CPU to show it is operating properly, it must perform a service routine prior to the count reaching $0000. The service routine consists of writing $5555 followed by $AAAA to the CNTR register.

NOTE:

The COP timer is stopped while the processor is in debug mode (that is, typically at a debugger breakpoint). If the COP is enabled, the timer will resume counting upon exiting debug mode (typically when the debugger releases the device). The CEN bit in the COP control (CTRL) register always reads as zero when in debug mode, even when it has a value of one.

3.4.1 Timeout Specifications

The COP uses a 16-bit counter being clocked by either MSTR_OSC or RLX_OSC prescaled by cop_prescaler. This value is set in hardware and varies per design family. For the following example, the cop_prescaler is set to 1024.

Table 3-1. COP Timeout Ranges as a Function of Oscillator Frequency

TOUT	4 MHz	8 MHz
$0000	256 µsec	128 µsec
$FFFF	16.8 sec	8.4 sec

For a crystal operating at 8 MHz the clock to the COP counter will be 7.8125 kHz. The value of the timeout (TOUT) register can be programmed from one to 65535 giving a timeout period range from 128 µsec minimum to 8.4 sec maximum.

3.4.2 COP After Reset

The control (CTRL) register is cleared out of reset. Thus the counter is disabled by default. In addition, the TOUT register is set to it's maximum value of $FFFF during reset so the counter is loaded with a maximum timeout period when reset is released.

3.4.3 Wait Mode Operation

If wait mode is entered with both CEN and CWEN bits set to one, the COP counter continues to count down. If either CEN or CWEN is set to zero when wait mode is entered, the counter is disabled and reloads using the value in the TOUT register.

3.4.4 Stop Mode Operation

If stop mode is entered with both CEN and CSEN bits set to one, the COP counter continues to count down. If either CEN or CSEN is set to zero when stop mode is entered, the counter will be disabled and reloads using the value in the TOUT register.

3.4.5 Debug Mode Operation

The COP counter is not allowed to count when the chip is in debug mode. Additionally, the CEN bit in the CTRL register always reads as zero when the chip is in debug mode. The actual value of CEN is unaffected by debug mode however, and resumes it's previously set value upon exiting debug mode.

3.4.6 Loss of Reference Operation

When the OCCS signals the COP a loss of the reference clock has occurred by using the OCCS_LOR signal and the CLOREN bit is set, the COP starts a 7-bit counter running off of the IPBus clock (continuing to be produced by the PLL for at least 1000 cycles upon losing its reference). The counter continues to count even if the OCCS_LOR signal deasserts. When this counter reaches $7F it causes a loss of reference reset resetting the entire chip. If the

software safely shut down the chip and does not want a full reset, the loss of reference timeout count can be delayed by servicing the COP counter in the standard manner of writing $5555 followed by $AAAA or stopped by setting CLOREN to zero.

NOTE:

Because this counter runs off of IPBus clock, it does not work when the chip is in stop mode. The way around this issue is to ensure the OCCS and ITCN modules have enabled the loss of reference clock to create an interrupt. This interrupt automatically wakes the part from stop mode and restarts the IPBus clock, allowing the counter to start counting to $7F.

3.5 Register Description

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. The address offset is defined at the module level.

Table 3-2. COP Memory Map

Device	Peripheral	Base Address
56F80xx	COP	$00F120

Table 3-3 lists the COP registers in ascending address order, including their acronyms and address offset of each register.

Table 3-3. COP Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL	Control register	Read/Write	Section 3.5.1
Base + $1	TOUT	Timeout register	Read/Write	Section 3.5.2
Base + $2	CNTR	Counter register	Read/Write	Section 3.5.3

There are three registers in the COP peripheral summarized in Figure 3-2. Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL	R	0	0	0	0	0	0	0	0	0	CLK_SEL	CLO_REN	BYPS	CSEN	CWEN	CEN	CWP
		W																
$1	TOUT	R	TIMEOUT															
		W																
$2	CNTR	R	COUNT															
		W	SERVICE															

R 0 Read as 0
W Reserved

Figure 3-2. COP Register Map Summary

3.5.1 Control (CTRL) Register

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	CLKSEL	CLOREN	BYPS	CSEN	CWEN	CEN	CWP
Write																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Figure 3-1. Control (CTRL) Register

3.5.1.1 Reserved—Bits 15–7

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

3.5.1.2 Clock Source Select (CLKSEL)—Bit 6

This bit selects the clock source for the counter. Some safety applications require the watchdog counter to use a different clock source than the system clock. This bit can only be changed when CWP bit is set to zero. It is recommended this bit only be changed when CEN bit is clear. This bit can be changed with CEN set if the MSTR_OSC is coming from the relaxation oscillator (that is, MSTR_OSC and RLX_OSC are the same signal).

- 0 = MSTR_OSC clocks the counter (default)
- 1 = RLX_OSC clocks the counter

3.5.1.3 COP Loss of Reference Enable (CLOREN)—Bit 5

This bit enables the operation of the loss of reference counter. This bit can only be changed when CWP bit is set to zero.

- 0 = COP loss of reference counter is enabled (default)
- 1 = COP loss of reference counter is disabled

3.5.1.4 Bypass MSTR_OSC (BYPS)—Bit 4

This bit is intended for factory use only. Setting this bit allows testing of COP to be speeded up by routing the IPBus clock to the counter instead of the prescaled selected oscillator. This bit should not be set during normal operation of the chip. If this bit is used it should only be changed while CEN bit is zero.

- 0 = Counter uses selected oscillator (default)
- 1 = Counter uses IPBus clock

3.5.1.5 COP Stop Mode Enable (CSEN)—Bit 3

This bit controls the operation of the counter in stop mode. This bit can only be changed when the CWP bit 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

3.5.1.6 COP Wait Mode Enable (CWEN)—Bit 2

This bit controls the operation of the counter in wait mode. This bit can only be changed when CWP bit 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

3.5.1.7 COP Enable (CEN)—Bit 1

This bit controls the operation of the counter. This bit can only be changed when CWP bit is set to zero. This bit always reads as zero when the chip is in debug mode.

- 0 = COP counter is disabled
- 1 = COP counter is enabled (default)

3.5.1.8 COP Write Protect (CWP)—Bit 0

This bit controls the write protection feature of the control (CTRL) register and the timeout (TOUT) register. When set, this bit can only be cleared by resetting the module.

- 0 = CTRL and TOUT can be read and written (default)
- 1 = CTRL and TOUT are read-only

3.5.2 Timeout (TOUT) Register

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TIMEOUT															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3-1. Timeout (TOUT) Register

3.5.2.1 Timeout Period (TIMEOUT)—Bits 15–0

This register determines the timeout period of the COP counter. TIMEOUT field should be written before the COP is enabled. When the COP is enabled, the recommended procedure for changing TIMEOUT is to disable the COP, write to TOUT, and then re-enable the COP. This procedure ensures the new TIMEOUT is loaded into the counter. Alternatively, the CPU can write to TOUT and then write the proper patterns to the CNTR register to cause the counter to reload with the new TIMEOUT value. Changing TIMEOUT while the COP is enabled results in a timeout period differing from the expected value. These bits can only be changed when CWP bit is set to zero.

3.5.3 Counter (CNTR) Register

Base + $2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNT															
Write	SERVICE															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3-1. Counter (CNTR) Register

3.5.3.1 Count (COUNT)—Bits 15–0

This 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.

3.5.3.2 Service (SERVICE)—Bits 15–0

When enabled, the COP requires a service sequence be performed periodically in order to clear the COP counter and prevent a reset from being issued. This routine consists of writing $5555 to CNTR followed by writing $AAAA before the timeout period expires. The writes to CNTR register 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.

Table 3-4. Document Revision History for Chapter 3

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 4
Inter-Integrated Circuit Interface (I^2C)

4.1 Introduction

The I^2C module is a programmable control bus providing the communications link between integrated circuits in a system. It is a simple two-wire bus with a software-defined protocol for system control, used in temperature sensors and voltage level translators to EEPROMs, general-purpose I/O, A/D and D/A converters, CODECs, and many types of microprocessors.

The I^2C bus is a two-wire serial interface, consisting of a serial data (SDA) line and a serial clock (SCL) line. These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a transmitter or receiver, depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device designed to initiate a data transfer on the bus and generate the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

The I^2C module can operate in standard mode with data rates up to 100 kHz and fast mode with data rates up to 400 kb/s. The I^2C module can communicate with devices only of either of these modes as long as they are attached to the bus. Additionally, fast mode devices are downward compatible. For instance, fast mode devices can communicate with standard mode devices in zero to 100 kb/s I^2C bus system. However, according to the Philips I^2C specification, standard mode devices are not upward compatible and should not be incorporated in a fast mode I^2C bus system as they cannot follow the higher transfer rate and unpredictable states would occur.

Most maintenance and control applications, the common use for the I^2C bus, typically operate at 100 kHz in standard and fast modes.

Any I^2C device can be attached to an I^2C bus and every device can talk with any master, passing information back and forth. At least one master is required (such as a microcontroller or DSC) on the bus, but there can be multiple masters, requiring them to arbitrate for ownership. Multiple masters and arbitration are explained later in this chapter.

The I^2C module is made up of an AMBA APB slave interface, an I^2C interface, and FIFO logic to maintain coherency between the two interfaces. A detailed block diagram of the component is illustrated in Section 4-1.

4.2 Features

I2C module has the following characteristics:

- Two-wire I2C serial interface – consists of a serial data (SDA) line and a serial clock (SCL) line
- Two speed modes:
  - Standard mode
  - Fast mode
- Clock synchronization
- Master or slave I2C operation
- Supports multi-master operation (bus arbitration)
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Slave bulk transmit mode
- Ignores CBUS addresses (an older ancestor of I2C previously shared the I2C bus)
- Four transmit and four receive buffers
- Interrupt or polled-mode operation
- Handles bit and byte waiting at all bus speeds
- Digital filter for the received SDA and SCL lines
- Component parameters for configurable software driver support

4.3 Block Diagram

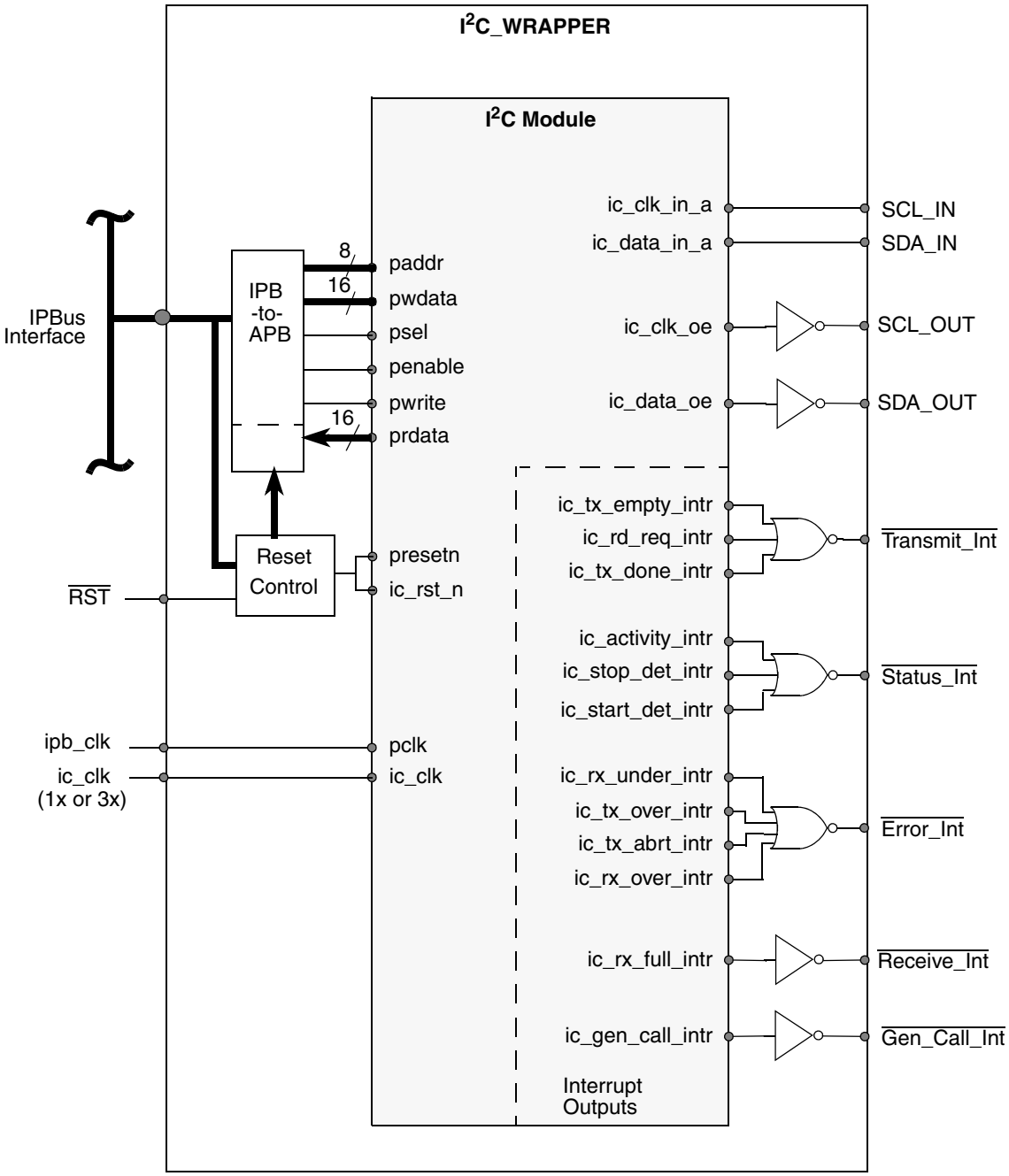


Figure 4-1. I2C Module Block Diagram

NOTE:

The ic_clk frequency must be greater than, or equal to pclk frequency. This restriction occurs because the configuration registers are programmed on pclk and the peripheral enable is the last bit to be programmed; it is then transferred to the other domain, validating the other bits.

4.4 Functional Description

4.4.1 I2C Behavior

The I2C module can be either a master or slave. The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device receiving data, either a master or a slave. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2C bus, using an arbitration procedure to determine bus ownership.

Each slave has a unique address determined by the system designer. When a master wants to communicate with a slave, the master transmits a START condition followed by the slave's address and a control bit read/write (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an Acknowledge (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the slave receives one or more bytes of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave drives its transmit data onto SDA (slave-transmitter) while the master is generating SCL clock pulses, and the master ACKs each byte of the transaction by driving SDA low during the ACK pulses. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a repeated START condition. This behavior is illustrated in Section 4-2.

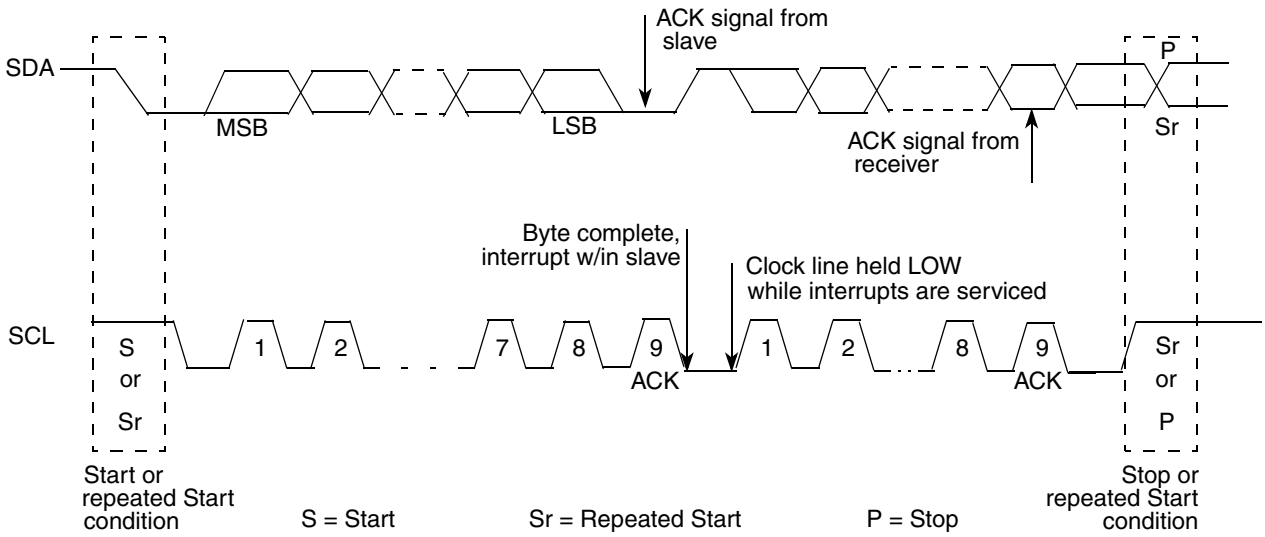


Figure 4-2. Data Transfer on the I2C Bus

The I2C module is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and repeated START conditions. The output drivers are open drain or open collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF. Data is transmitted in byte packages.

The I2C protocols implemented in I2C module are described in more details in the following section and Section 4-3.

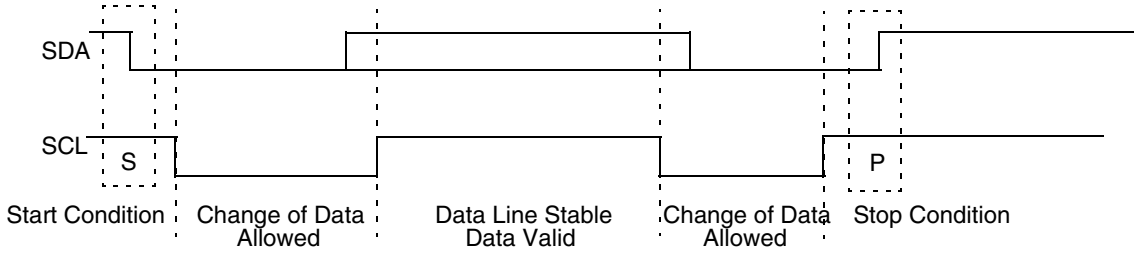


Figure 4-3. Start and Stop Conditions

4.4.2 Start and Stop Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pullup resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is one. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is one. Section 4-3 illustrates the START and STOP conditions. When data is being transmitted on the bus, the SDA line must be stable when SCL is one.

4.4.3 Addressing Slave Protocol

There are two address formats:

1. 7-bit address format
2. 10-bit address format

4.4.3.1 Seven-Bit Address Format

During the 7-bit address format, the first seven bits of the first byte contain the slave address and the LSB bit is the R/W bit illustrated in Section 4-4. When bit zero R/W is set to zero, the master writes to the slave. When bit zero R/W is set to one, the master reads from the slave.

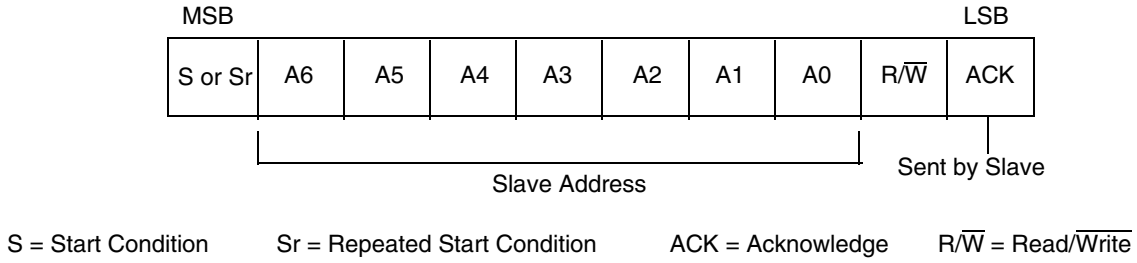


Figure 4-4. Seven-Bit Address Format

During 10-bit addressing, two bytes are needed to present the address. The transfer of the first byte contains the following bit definition. The first five bits notify the slaves this is a 10-bit transfer followed by the next two bits, containing the slave's address bits nine and eight, and the LSB bit is the R/W bit. The second byte transferred contains bits seven through zero of the slave address. Section 4-5 illustrates the 10-bit address format, and Table 4-1 defines the special purpose and reserved first byte addresses.

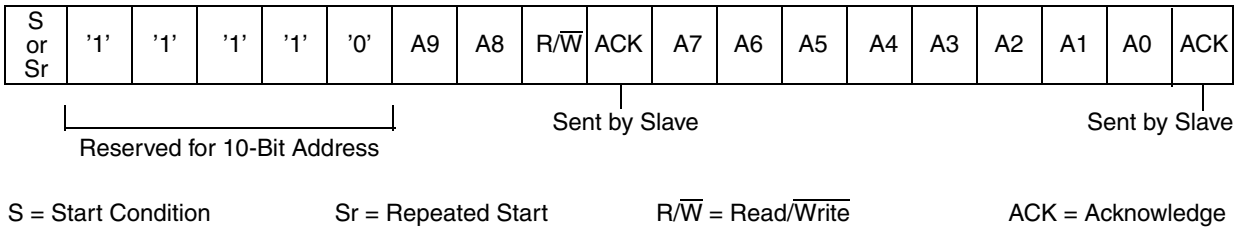


Figure 4-5. 10-Bit Address Format

Table 4-1. Definition of Bits in the First Address Byte

Slave Address	R/W Bit	Description
0000 000	0	General call address
0000 000	1	Start byte. Refer to the Section 4-8
0000 001	X	CBUS address. I2C ignores these accesses.
0000 010	X	Reserved
0000 011	X	Reserved
0000 1XX	X	High-speed master code
1111 1XX	X	Reserved
1111 0XX	X	10-bit slave addressing

4.4.4 Transmitting and Receiving Protocol

The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively. All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer.

4.4.4.1 Master-Transmitter and Slave-Receiver

After the master sends the address and R/W bit, or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge (ACK) signal. When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so the master can abort the transfer.

If the master-transmitter is transmitting data as shown in Section 4-6, then the slave-receiver responds to the master-transmitter with an acknowledge after every byte of data is received.

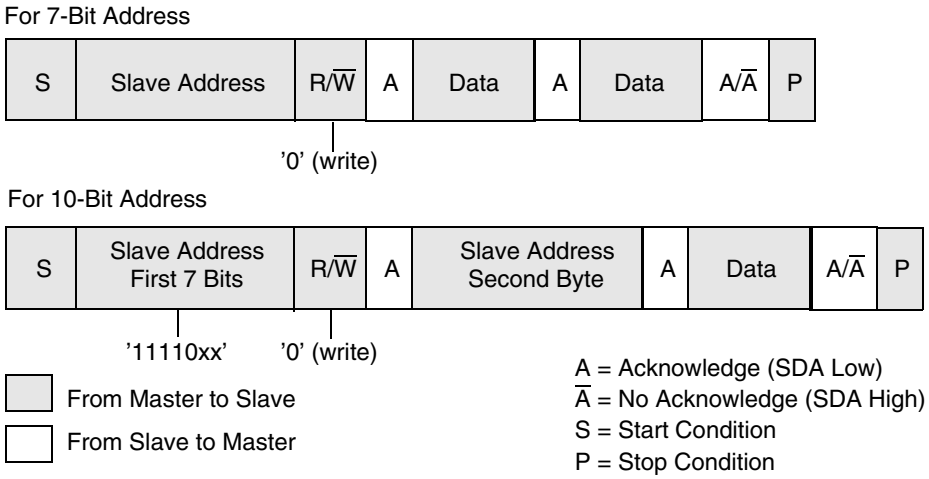


Figure 4-6. Master-Transmitter Protocol

4.4.4.2 Master-Receiver and Slave-Transmitter

If the master is receiving data, illustrated in Section 4-7, the master responds to the slave-transmitter with an ACK pulse after a byte of data is received, except for the last byte. The no acknowledge (NACK) is the way the master-receiver notifies the slave-transmitter this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the NACK so the master can issue a STOP condition.

When a master does not want to relinquish the bus with a STOP condition, the master can issue a repeated START condition. This is identical to a START condition except it occurs after the ACK pulse. The master can then communicate with the same, or a different slave.

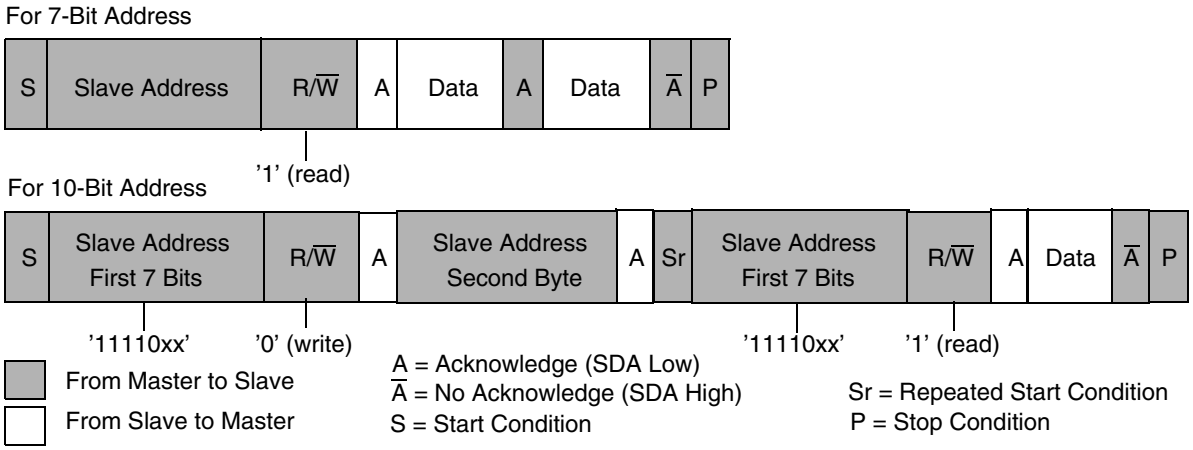


Figure 4-7. Master-Receiver Protocol

4.4.5 Start Byte Protocol

The START byte protocol is set up for systems not having an on-board dedicated I^2C hardware module. When the I^2C module is addressed as a slave, it always samples the I^2C bus at the highest speed supported so it never requires a START byte protocol. However, when I^2C module is a master, it supports the generation of START byte protocols at the beginning of every transfer in case a slave device requires it. This protocol consists of seven zeros being transmitted and followed by a one, illustrated in Section 4-8. This allows the bus polling processor to

under-sample the address phase until zero is detected. Once the microcontroller detects a zero, the processor switches to a higher sampling rate.

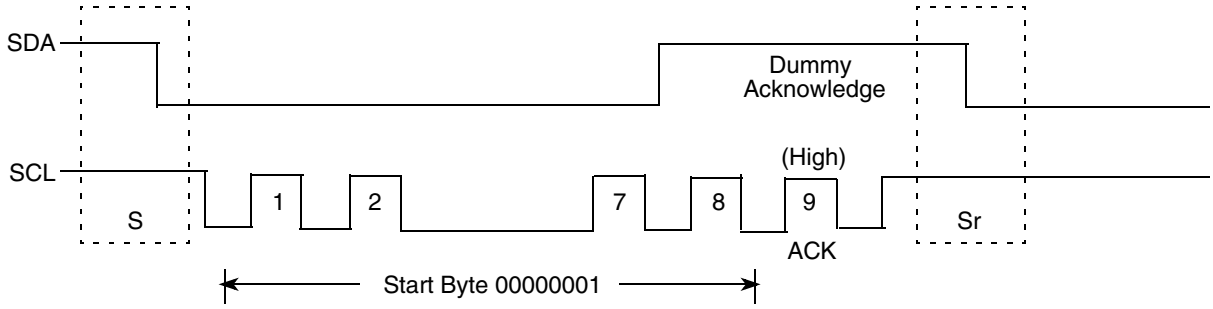


Figure 4-8. Start Byte Procedure

The START byte procedure is outlined here:

- Master generates a START condition
- Master transmits the START byte (0000 0001)
- Master transmits the ACK clock pulse (present only to conform with the byte handling format used on the bus)
- No slave drives the ACK signal to zero
- Master generates a repeated START condition

A hardware receiver ignores the START byte because it is a reserved address and resets after the repeated START condition is received.

4.4.6 Multiple Master Arbitration

The I^2C bus protocol defines an arbitration procedure allowing multiple masters to reside on the same bus. Arbitration occurs when multiple masters attempt to control the bus at the same time by generating a START condition. When a master (for example, a microcontroller) has control of the bus and arbitration occurred, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is one. A master transmitting one while the other master transmits zero, loses arbitration and turns off its data output stage. The master losing arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase. Section 4-9 illustrates timing when two masters are arbitrating on the bus.

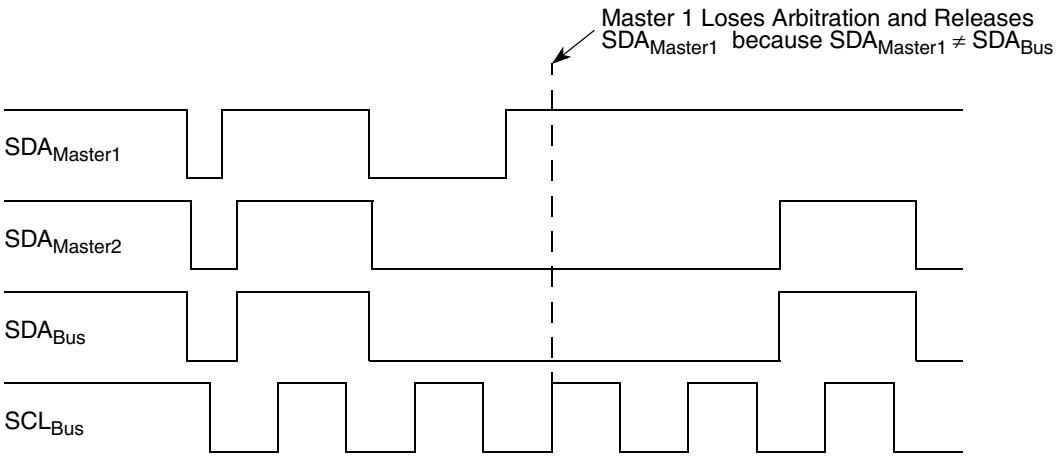


Figure 4-9. Multiple Master Arbitration

Bus control is decided solely on the address and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

4.4.7 Clock Synchronization

When two or more masters attempt to access the bus at the same time, they must arbitrate and synchronize their serial clocks (SCLs). All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL. Clock synchronization is performed using the wired-AND connection to the SCL signal. When a master transitions the SCL to zero, the master starts counting the low time of the SCL, transitioning the SCL signal to one when the clock high state is reached. However, if another master is holding the SCL line to zero, the master goes into a high wait state until the SCL line transitions to one.

All masters then count off their high time, and the master with the shortest high time transitions the SCL line to zero. The masters then count out their low time and the one with the longest low time forces the other master into a high wait state. Therefore, a synchronized SCL is generated, illustrated in Section 4-10. Optionally, slaves may hold the SCL line low, thereby reducing the speed of the I^2C bus.

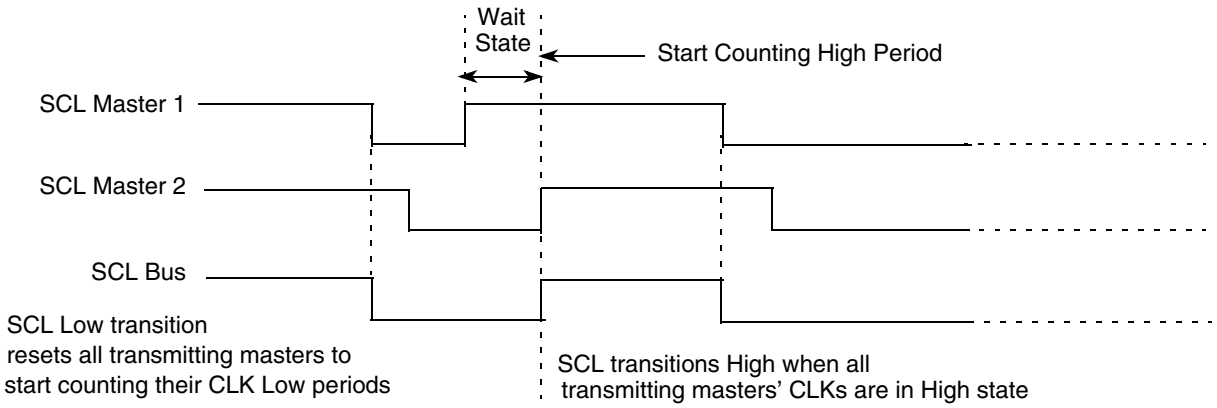


Figure 4-10. Multi-Master Clock Synchronization

4.5 Operation Modes

4.5.1 Slave Mode Operation

4.5.1.1 Initial Configuration

To use the I2C module as a slave, perform the following four steps:

1. Disable the I2C module by writing zero to bit zero of the enable (ENBL) register.
2. To set the slave address, write to the slave address register (SAR). The I2C module responds to this address.
3. To specify the type of supported addressing (7- or 10-bit by writing bit three), write to the control (CRTL) register. Specify whether the I2C module is in slave-only mode (by writing a zero to bit six and a zero to bit zero), or master-slave mode (by writing a zero to bit six and a one to bit zero).

NOTE:

When configuring the I2C module, the addressing type specified for the slave mode is not required to match the addressing type specified for the master mode. For example, master mode can be configured with 10-bit addressing and slave mode can be configured with 7-bit addressing.

4. Enable the I2C module by writing one to bit one of the ENBL register.

4.5.1.2 Slave-Transmitter Operation for a Single Data Byte

When a remote I2C master device on the bus addresses the I2C module, requesting data, the I2C module acts as a slave-transmitter and the following eight steps occur:

1. The remote I2C master device initiates an I2C transfer with an address matching the slave address in the SAR register.
2. The I2C module acknowledges the sent address and recognizes the direction of the transfer.
3. The I2C module asserts the read request (RDREQ) interrupt bit in the RISTAT register and holds the SCL line low. It is in a wait state until software responds.
4. If there is any data remaining in the transmit FIFO (TXFIFO) before receiving the read request, the module asserts a transmit abort (TXABRT) interrupt bit in the RISTAT register to flush the old data from the TXFIFO.
5. Software then writes the data (DATA) register with the data to be written by writing zero in bit eight.
6. The I2C module releases SCL, allowing the remote master to generate SCL transitions to clock the transmit byte out of the I2C.
7. Before proceeding, software must clear the RDREQ and TXABRT interrupts bits, respectively in the RISTAT register.
8. The remote master may hold the I2C bus by issuing a repeated START condition or release the bus by issuing a STOP condition.

4.5.1.3 Slave-Receiver Operation for a Single Data Byte

When a remote I^2C master device on the bus addresses the I^2C module and is sending data, the I^2C module acts as a slave-receiver and the following six steps occur:

1. The remote I^2C master device initiates an I^2C transfer with an address matching the I^2C's slave address in the SAR register.
2. The I^2C module acknowledges the sent address and recognizes the direction of the transfer.
3. The I^2C module receives the transmitted byte and places it in the receive buffer, assuming there is enough space.
4. The status and interrupt bits corresponding to the receive buffer are updated.

NOTE:

For RXFULL interrupt assertion with every received data byte, the RXFT register must contain a value of zero prior to the start of slave-receiver activity.

5. Software may read the byte from the DATA register.
6. The remote master device may hold the I^2C bus by issuing a repeated START condition or release the bus by issuing a STOP condition.

4.5.1.4 Slave Bulk Transmit Operation

There should be a minimum of one entry placed into a slave transmitter's TXFIFO when a slave (slave transmitter) is issued with a read request from a remote master (master receiver).

The I^2C module is designed to handle more data in the TXFIFO allowing subsequent read requests to take that data without raising an interrupt to get more data. Ultimately, this procedure eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TXFIFO.

The RDREQ interrupt is asserted and SCL held low upon receiving a read request from a remote master. The interrupt service routine can either write one byte or more than one byte into the TXFIFO. During the transmission of these bytes to the remote master, if the TXFIFO is allowed to go empty by the CPU and the remote master acknowledges the last byte remaining in the TXFIFO, the slave must reassert the RDREQ interrupt and hold SCL low. This is because the remote master is looking for more data.

Assuming *n* is less than, or equal to the depth of the TXFIFO, if the programmer knows in advance the remote master is requesting a packet of *n* bytes, when a remote master addresses the I^2C module and requests data, the TXFIFO could be written with *n* number bytes. The remote master receives it as a continuous stream of data. For example, the I^2C slave permits the remote master to clock TX data out of the slave as long as the remote master is acknowledging the data sent and there is data available in the TXFIFO. There is no need to hold SCL low or to issue RDREQ again.

If the remote master is to receive *n* bytes from the I^2C module, but the programmer wrote *m* bytes to the TXFIFO, where *m* is larger than *n*, but not greater than the depth of the TXFIFO, when the slave finishes sending the requested *n* bytes, it clears the TXFIFO and ignores any excess bytes.

There is no interrupt to signify the clearing of the TXFIFO in this example. At the time an acknowledge is expected, if a NACK is received, the remote master has all the data it wants. At this time, a flag is raised within the slave's state machine to clear the leftover data. This bit is transferred to the processor domain where the TXFIFO exists and the contents of the TXFIFO is cleared at that time. This prevents the master within the I^2C module thinking it has data to send when it does not.

4.5.1.5 Slave Bulk Receive Operation

This operation is similar to slave receiver operation for a single data byte. The RXFT register value determines the number of RXFIFO entries required to assert the RXFULL interrupt.

4.5.2 Master Mode Operation

4.5.2.1 Initial Configuration

To use the I2C module as a master, perform the following seven steps:

1. Disable the I2C module by writing zero to the ENBL register.
2. Write to the SSHCNT and SSLCNT registers if operating in the standard mode. Write to the FSHCNT and FSLCNT registers if operating in fast mode.
3. Write to the slave address register (SAR) to set the slave address, the address to which the I2C module responds.
4. Write to the SPD bit field in the control (CTRL) register to set the speed mode desired for I2C module master initiated transfers. If configuring for master slave operation, also set the desired addressing of the I2C module for slave mode on ADDRSLV bit, either 7- or 10-bit addressing. When writing to the CTRL register the I2C module can be configured for either master only or master slave operation. This is achieved by writing one to bit six and one to bit zero or by writing zero to bit six and one to bit zero respectively.

   NOTE:

   When configuring the I2C module, the addressing type specified for the slave mode is not required to match the addressing type specified for the master mode. For example, master mode can be configured with 10-bit addressing and slave mode can be configured with 7-bit addressing.

5. Write to the target address register (TAR) the address of the I2C device to be addressed. It also indicates whether a general call, or a START byte command, is going to be performed by I2C. The desired addressing of the I2C module master-initiated transfers, either 7- or 10-bit addressing, is controlled by the address master (ADDRMST) bit in TAR.
6. Enable the I2C module by writing one in the ENBL register.
7. Now, write the transfer direction and data to be sent to the DATA register. If the DATA register is written before the I2C is enabled, the data and commands are lost because the buffers are kept cleared when I2C is not enabled.

This step generates the START condition and the address byte on the I2C bus if it is idle. After generating the START condition and the address byte, the I2C begins reading or writing the data.

4.5.2.2 Dynamic TAR or ADDR MST Update

The I2C module supports dynamic updating of the TAR register. By dynamic, this means software can change the value of the TAR even while the slave interface of the I2C module is involved in an I2C transfer assuming certain conditions detailed in the first point below are met.

- Before performing a dynamic TAR update, make certain the following conditions are met:

— MSTACT bit in the STAT register must be idle; that is, register STAT[5] = 0
— Transmit FIFO completely empty must occur; that is, register STAT[2] = 1

NOTE:

If a bulk read is performed when the I^2C is acting as a slave over the I^2C bus, only MSTACT bit must be idle (STAT[5] = 0); that is, the TXFIFO does not need to be completely empty. This is a very specific case and should be monitored in software.

4.5.2.3 Master Transmit and Master Receive

The I^2C module supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the RX/TX data (DATA) buffer and command register. Bit eight (CMD) should be written to zero for write operations. Subsequently, a read command may be issued by writing don't cares to the lower byte of the DATA register, where each write to the DATA register sets CMD bit. As data is transmitted and received, the transmit and receive buffer status bits and interrupts change.

4.5.2.4 Disabling I^2C Module

I^2C module consists of two functional bus functions/modules, an I^2C master and I^2C slave. Either or both of these can be enabled at any time, but the configuration of this can only be performed if the I^2C module is disabled (when the ENBL register[0] is set to 0). Therefore, the following instructions for disabling I^2C module are also applicable when changing whether the master or slave I^2C bus device is enabled.

The ENBL register allows software to exercise enabling or disabling controls on the I^2C module hardware, which in turn allows the master and slave functions/modules to logically appear or disappear from the I^2C bus. It is necessary for software to specifically manage and coordinate the graceful shutdown of the I^2C module hardware in order there is no data loss or bus lockup.

4.5.2.4.1 Prior Conditions

- I^2C master (transmit or receive)—Software must ensure all existing and/or pending transfers are completed but no new I^2C transfers are started or accepted. The MSTACT bit of the STAT register determines whether I^2C module has ceased all I^2C master-related activity. A value of zero for MSTACT bit indicates the I^2C master FSM is in the IDLE state. This is a pre-condition for safely shutting down I^2C module.
- I^2C slave—Software must cooperate with any I^2C remote master(s) to complete all outstanding I^2C transfers. It is not possible for software to instruct I^2C module to cease such activity without forcefully shutting down the I^2C module hardware, potentially resulting in data loss or bus lockup. The SLVACT bit of the STAT register determines whether I^2C module has ceased all I^2C slave-related activity. A value of zero for SLVACT indicates the I^2C slave FSM is in the IDLE state. This is a precondition for safely shutting down I^2C module.

NOTE:

A remote master can initiate a transfer at any time so any action based on the value of SLVACT should be performed immediately after reading the STAT register.

NOTE:

When either MSTACT or SLVACT, or both, indicates the master or slave FSM respectively is in the IDLE state, this does not mean the I^2C bus is idle.

4.5.2.4.2 Procedure

1. Define a timer interval (t_{i2c_poll}) equal to M times the signaling period for the highest I^2C transfer speed used in the system and supported by I^2C module. For example, if the highest I^2C transfer mode is 400 kb/s, then M can be set to a value of 10 and this t_{i2c_poll} will then be 25 µs.
2. Define a maximum timeout parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.
3. Execute a blocking thread/process/function, ensuring all outstanding I^2C master transactions are completed while at the same time prevents further I^2C master transactions to be started by software.
4. The variable POLL_COUNT is initialized to zero.
5. Read the STAT register and test the MSTACT bit. Increment POLL_COUNT by one. If POLL_COUNT > = MAX _T_POLL_COUNT, exit with the relevant error code.
6. If MSTACT is one, then sleep for t_{i2c_poll} and proceed to step five.
7. The variable POLL_COUNT is initialized to zero.
8. Read the STAT register and test the SLVACT bit. Increment POLL_COUNT by one. If POLL_COUNT > = MAX_T_POLL_COUNT, exit with the relevant error code.
9. If SLVACT bit is one, then sleep for t_{i2c_poll} and proceed to step eight.
10. Write zero to the IENBL register. ISRs must account for the possibility of interrupt sources being truncated. Perform a read of this register before hand and save the value for later possible retrieval.
11. Write zero to ENBL to disable I^2C module.
12. Sleep for t_{i2c_poll} and initialize the variable POLL_COUNT to zero.
13. Read the STAT register and test the SLVACT bit.
14. If SLVACT is equal to one, write one to the ENBL register, re-enable the interrupts by restoring and writing in the contents of IENBL. Sleep for t_{i2c_poll} and proceed to step eight.
15. Read the RISTAT register. Increment POLL_COUNT by one. If POLL_COUNT > = MAX_T_ POLL_COUNT, exit with the relevant error code.

NOTE:

On exiting this step with the indicated error code, one must note the IENBL and ENBL registers are already set to zero.

16. If RISTAT register is not equal to zero, then sleep for t_{i2c_poll} and proceed to step 13.
17. Return with the relevant success code.

4.5.3 IC_CLK Frequency Configuration

When the I^2C module is configured as a master, the *x*HCNT and *x*LCNT registers must be set before any master I^2C bus transaction can take place to ensure proper I/O timing. The *x*HCNT and *x*LCNT registers are:

- SSHCNT
- SSLCNT

- FSHCNT
- FSLCNT

Setting the *x*LCNT registers configures the number of ic_clk periods required for setting the low time of the SCL clock in each speed mode. Setting the *x*HCNT registers configures the number of ic_clk periods required for setting the high time of the SCL clock in each speed mode. Setting the registers to the correct value is described as follows.

The equations for calculating the high and low counts are as follows:

x_HCNT = (ROUNDDOWN(SCL_HIGH_{time}*CLKFREQ,0) - DELAY_ADJ_HIGH)
x_LCNT = (ROUNDUP(SCL_LOW_{time}*CLKFREQ,0) - DELAY_ADJ_LOW)
ROUNDDOWN and ROUNDUP are explicit Excel function calls that are used to convert a real number to the desired integer number.

CLKFREQ = ic_clk Clock Frequency (Hz).
DELAY_ADJ_HIGH = 8
DELAY_ADJ_LOW = 1

For example:
I^2C Speed Mode = fast, 400k/bs
CLKFREQ = 32MHz
SCL_HIGH_{time} = 1200ns for 400k/bs
SCL_LOW_{time} = 1300ns for 400kb/s (Philips I2C spec min is 1300ns)

x_HCNT = (ROUNDDOWN(SCL_HIGH_{time}*CLKFREQ,0) - DELAY_ADJ_HIGH)
x_HCNT = (ROUNDDOWN(1200ns * 32MHz,0) - 8)
x_HCNT = 30 decimal

x_LCNT = (ROUNDUP(SCL_LOW_{time}*CLKFREQ,0) - DELAY_ADJ_LOW)
x_LCNT = (ROUNDUP(1300ns * 100MHz,0) - 1)
x_LCNT = 41 decimal

Actual SCL_HIGH_{time} = (30+8)*(1/32MHz) = 1187.5ns
Actual SCL_LOW_{time} = (41+1)*(1/32MHz) = 1312.5ns

The minimum value for *x*_LCNT is eight while the minimum *x*_HCNT is six. Also, because of the digital filtering on the receiver, the actual SCL high and low times are slightly longer than the specified count value; eight more ic_clks for SCL high and one additional ic_clk for SCL low period. Subtract eight from the high count and subtract one from the low count values to account for this. The following six points describe why this occurs:

The state machine controlling the data transfer needs a few clock cycles before being able to drive the I^2C bus with the required levels.

The digital filter used for both incoming SDA and SCL signals needs about four ic_clk cycles to process the data.

- The master state machine must be monitoring the bus while it is driving the bus for arbitration lost and acknowledge detection
- The design is a fully synchronous D-flip-flop based design

- The I2C module does not use the I2C bus SCL line as a clock source

The minimum ic_clk frequency for:

- Standard mode is 2.7 MHz
- Fast mode is 10 MHz

The I2C module does not have to be running at the clock frequencies listed previously to support all different modes. However, it is necessary to run at, or above those frequencies to operate the master at the maximum I2C baud rates. The observed SCL_HIGH time and SCL_LOW time values on the I2C bus can vary depending on loading in the system. Please see the I2C bus specification and information from Philips for more detail.

http://www.semiconductors.philips.com/buses/i2c/index.html

The 16-bit range on these registers allows a wide range of input clock frequencies to be used.

4.6 Register Description

Registers are on the pclk domain, but status bits reflect actions occurring in the ic_clk domain. Therefore, there is a two-clock delay when the pclk register reflects the activity having taken place on the ic_clk side. Some registers may be written only when the I2C module is disabled, programmed by the ENBL register.

To disable the module, please see Section 4.5.2.4.2.

Some register can be written while the I2C module is enabled. Other registers can only be written while the I2C module is in a disabled state. Registers requiring the module to be in a disabled state when writing are indicated in their descriptions.

Table 4-2. I2C Memory Map

Device	Peripheral	Base Address
56F80xx	I2C	$00F280

Table 4-3 lists the I²C registers in ascending address order, including their acronyms and address offset of each register.

Table 4-3. I²C Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL	Control register	Read/Write	Section 4.6.1
Base + $2	TAR	Target address register	Read/Write	Section 4.6.2
Base + $4	SAR	Slave address register	Read/Write	Section 4.6.3
Base + $8	DATA	Data buffer & command register	Read/Write	Section 4.6.4
Base + $A	SSHCNT	Standard speed clock high count register	Read/Write	Section 4.6.5
Base + $C	SSLCNT	Standard speed clock low count register	Read/Write	Section 4.6.6
Base + $E	FSHCNT	Fast speed clock high count register	Read/Write	Section 4.6.7
Base + $10	FSLCNT	Fast speed clock low count register	Read/Write	Section 4.6.8
Base + $16	ISTAT	Interrupt status register	Read-Only	Section 4.6.9
Base + $18	IENBL	Interrupt enable register	Read/Write	Section 4.6.10
Base + $1A	RISTAT	Raw interrupt status register	Read-Only	Section 4.6.11
Base + $1C	RXFT	Receive FIFO threshold register	Read/Write	Section 4.6.12
Base + $1E	TXFT	Transmit FIFO threshold register	Read/Write	Section 4.6.13
Base + $20	CLRINT	Clear interrupt register	Read-Only	Section 4.6.14
Base + $22	CLRRXUND	Clear receive under interrupt register	Read-Only	Section 4.6.15
Base + $24	CLRRXOVR	Clear receive over interrupt register	Read-Only	Section 4.6.16
Base + $26	CLRTXOVR	Clear transmit over interrupt register	Read-Only	Section 4.6.17
Base + $28	CLRRDREQ	Clear read request interrupt register	Read-Only	Section 4.6.18
Base + $2A	CLRTXABRT	Clear transmit abort interrupt register	Read-Only	Section 4.6.19
Base + $2C	CLRTXDONE	Clear transmit done interrupt register	*Read-Only*	Section 4.6.20
Base + $2E	CLRACT	Clear activity interrupt register	Read-Only	Section 4.6.21
Base + $30	CLRSTPDET	Clear sop detect interrupt register	Read-Only	Section 4.6.22
Base + $32	CLRSTDET	Clear start detect interrupt register	Read-Only	Section 4.6.23
Base + $34	CLRGC	Clear general call register	Read-Only	Section 4.6.24
Base + $36	ENBL	Enable register	Read/Write	Section 4.6.25
Base + $38	STAT	Status register	Read-Only	Section 4.6.26
Base + $3A	TXFLR	Transmit FIFO level register	Read-Only	Section 4.6.27
Base + $3C	RXFLR	Receive FIFO level register	Read-Only	Section 4.6.28
Base + $40	TXABRTSRC	Transmit abort source register	Read-Only	Section 4.6.29

There are 29 registers in the I2C peripheral summarized in Section 4-11. Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL	R	0	0	0	0	0	0	0	0	0	SLV DIS	RST EN	ADDR MST	ADDR SLV	SPD		MST EN
		W																
$2	TAR	R	0	0	0	ADDR MST	SPCL	GC STRT	TA									
		W																
$4	SAR	R	0	0	0	0	0	0	SA									
		W																
$8	DATA	R	0	0	0	0	0	0	0	CMD	DAT							
		W																
$A	SSHCNT	R	SSHCNT															
		W																
$C	SSLCNT	R	SSLCNT															
		W																
$E	FSHCNT	R	FSHCNT															
		W																
$10	FSLCNT	R	FSLCNT															
		W																
$16	ISTAT	R	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
		W																
$18	IENBL	R	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
		W																
$1A	RISTAT	R	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
		W																
$1C	RXFT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXTL	
		W																
$1E	TXFT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXTL	
		W																
$20	CLRINT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
		W																
$22	CLRRXUND	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXUND
		W																
$24	CLRRXOVR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXOVR
		W																
$26	CLRTXOVR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXOVR
		W																
$28	CLRRDREQ	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RD REQ
		W																
$2A	CLRTXABRT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TX ABRT
		W																

Figure 4-11. I2C Register Map Summary

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$2C	CLRTXDONE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TX DONE
		W																
$2E	CLRACT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ACT
		W																
$30	CLRSTPDET	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STP DET
		W																
$32	CLRSTDET	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ST DET
		W																
$34	CLRGC	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GC
		W																
$36	ENBL	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																
$38	STAT	R	0	0	0	0	0	0	0	0	0	SLV ACT	MST ACT	RFF	RFNE	TFE	TFNF	ACT
		W																
$3A	TXFLR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	TXFL		
		W																
$3C	RXFLR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	RXFL		
		W																
$40	TXABRTSRC	R	SLV RD	SLV AL	SLV FLSH	AL	MST DIS	RNOR ST	SNOR ST	0	SACK DET	0	GC READ	GC NACK	TD NACK	AD2 NACK	AD1 NACK	AD7 NACK
		W																

R	0	Read as 0
W		Reserved

Figure 4-11. I²C Register Map Summary (Continued)

4.6.1 Control (CTRL) Register

All bits are read/write (R/W) except for bit four. Bit four (ADDRMST) is read-only. This register can be written only when the I2C module is disabled, corresponding to the ENBL register being set to zero. Writes at other times have no effect.

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	SLV DIS	RST EN	ADDRMST	ADDRSLV	SPD		MST EN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

Figure 4-12. Control (CTRL) Register

4.6.1.1 Reserved—Bits 15–7

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.1.2 Slave Disable (SLVDIS)—Bit 6

This bit controls whether I2C has its slave disabled. If this bit is set, the slave is disabled.

- 0 = Slave is enabled
- 1 = Slave is disabled

NOTE:

Avoid enabling both the master and the slave modes when writing to the CTRL register. Operation of the I2C module when SLVDIS=0 and MSTEN=1 may result in unpredictable behavior of the module.

4.6.1.3 Repeated Start Enable (RSTEN)—Bit 5

This bit determines whether repeated START conditions may be sent when acting as a master. Some older slaves do not support handling repeated START conditions; however, repeated START conditions are used in several I2C module operations.

- 0 = Disable
- 1 = Enable

When repeated START is disabled, the master cannot perform the following functions:

- Send a START byte
- Read operation with a 10-bit address

If the above non-split operations are attempted, it results in setting TXABRT bit in the RISTAT register.

When repeated START is disabled, the master modifies the following functions:

- Change direction within a transfer (split)
- Combined format transfers in 7-bit addressing mode (split)
- Combined format transfers in 10-bit addressing mode not involving read operations (split)

By replacing repeated START condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple I^2C module transfers.

4.6.1.4 Address Mode Master (ADDRMST)—Bit 4

This bit is a read-only copy of the ADDRMST bit in the target address register (TAR).

- 0 = 7-bit addressing
- 1 = 10-bit addressing

4.6.1.5 Address Mode Slave (ADDRSLV)—Bit 3

When acting as a slave, this bit controls whether the I^2C module responds to 7- or 10-bit addresses.

- 0 = 7-bit addressing—The I^2C module ignores transactions involving 10-bit addressing; for 7-bit addressing, only the lower seven bits of the slave address register (SAR) are compared.
- 1 = 10-bit addressing—The I^2C module responds to only 10-bit addressing transfers matching the full 10 bits of the SAR.

NOTE:

ADDRSLV and TAR ADDRMST can be programmed differently and in any combination depending on which format is required for the transfers. For example, master mode can be configured with 10-bit addressing and slave mode can be configured with 7-bit addressing.

4.6.1.6 Speed (SPD)—Bits 2–1

This bit field controls the speed the I^2C module operates; its setting is relevant only if one is operating the I^2C module in master mode. Hardware protects against illegal values being programmed by software. This register should be programmed only with a value in the range of one to two; otherwise hardware updates this register with the value of two.

- 00 = Reserved
- 01 = Standard mode (0-100 k/bs)
- 10 = Fast mode (0-400 k/bs) default
- 11 = Reserved

4.6.1.7 Master Enable (MSTEN)—Bit 0

This bit controls whether the I^2C module master is enabled.

- 0 = Master disabled
- 1 = Master enabled

NOTE:

Avoid enabling both the master and the slave modes when writing to the CTRL register. Operation of the I^2C module when SLVDIS=0 and MSTEN=1 may result in unpredictable behavior of the module.

4.6.2 Target Address Register (TAR)

All bits can be dynamically updated as long as the following are true:

- The MSTACT bit of the STAT register must be IDLE; that is STAT[5] = 0
- Transmit FIFO completely empty must occur; that is STAT[2] = 1

Base + $2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	ADDRMST	SPCL	GCSTRT	TA									
Write																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Figure 4-13. Target Address Register (TAR)

4.6.2.1 Reserved—Bits 15–13

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.2.2 Address Mode Master (ADDRMST)—Bit 12

This bit controls whether the I2C module performs its transfers in 7- or 10-bit addressing mode when acting as a master.

- 0 = 7-bit addressing
- 1 = 10-bit addressing

4.6.2.3 Special (SPCL)—Bit 11

This bit indicates whether software performs a general call or START byte command.

- 0 = Ignore bit 10 (GCSTRT) and use TAR normally
- 1 = Perform special I2C command as specified in GCSTRT bit

4.6.2.4 General Call or Start (GCSTRT)—Bit 10

If the SPCL bit is set to one, it indicates whether a general call or START byte command is to be performed by the I2C module.

- 0 = General call address
- 1 = Start byte

When issuing a general call, only writes may be performed to the TXFIFO. Attempting to issue a read command results in setting TXABRT bit in the RISTAT register. The I2C module remains in general call mode until the SPCL bit is cleared, or until the GCSTRT bit is set.

4.6.2.5 Target Address (TA)—Bits 9–0

This is the target address for any master transaction. This bit field is ignored when transmitting a general call.

If TA and SA bits are the same, loop back exists but the FIFOs are shared between master and slave, therefore full loop back is not feasible. Only one direction (simplex) loop back mode is supported, consisting of master mode transmit and slave mode receive.

4.6.3 Slave Address Register (SAR)

Base + $4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	SA									
Write																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Figure 4-14. Slave Address Register (SAR)

4.6.3.1 Slave Address (SA)—Bits 9–0

The SA bit field holds the slave address when the I^2C is operating as a slave. For 7-bit addressing only bits [6:0] are used. This register can be written only when the I^2C interface is disabled. Writes at other times have no effect.

NOTE:

The programmed value cannot be any of the reserved address locations: that is, 0×00 to 0×07, or 0×78 to 0×7F. These addresses should be obtained from Philips as part of the licensing agreement. Philips will assign a slave address. The correct operation of the device is not guaranteed if registers SAR or TAR are programmed to a reserved value.

4.6.4 Data (DATA) Buffer and Command Register

This register is written by the CPU when filling the TXFIFO. The CPU reads from this register when retrieving bytes from the RXFIFO.

Base + $8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	CMD	DAT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-15. Data (DATA) Buffer and Command Register

4.6.4.1 Reserved—Bits 15–9

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.4.2 Command (CMD)—Bit 8

When writing to the data register, this bit controls whether a read or a write is performed.

- 0 = Write
- 1 = Read

In slave receiver mode, writes to this register are treated as master mode commands. When in the slave transmitter mode, this bit must be written with a value of zero whenever writing to this register. If this bit is written with a value of one while in slave transmitter mode, a transmit abort will occur and the SLVRD bit in the TXABRTSRC register is set.

When writing this bit, attempting to perform a read operation while register TAR is configured for the general call command results in a TXABRT bit interrupt in the RISTAT register, unless either SPCL bit in the TAR is cleared, or GCSTRT bit in the TAR is set. If one is written to the CMD bit after receiving a read request interrupt, a transmit abort interrupt occurs.

When reading from the DATA register, this bit should be ignored.

4.6.4.3 Data (DAT)—Bits 7–0

This register contains the data to be transmitted or received in the I2C bus. If writing a read command to this register, the DAT bit field is ignored by the I2C module. However, when this register is read, this bit field returns the value of data received in the I2C module interface.

NOTE:

Avoid writing to the DATA register when the ENBL register is zero. Writing to the DATA register while the ENBL register is zero may result in unpredictable operation of the I2C module.

4.6.5 Standard Speed I2C Clock SCL High Count (SSHCNT) Register

Base + $A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SSHCNT															
Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1

Figure 4-16. Standard Speed I2C Clock SCL High Count (SSHCNT) Register

4.6.5.1 Standard Speed High Count (SSHCNT)—Bits 15–0

To ensure proper I/O timing, this register must be set before any standard speed master mode I2C operations can take place. This register sets the SCL clock high-period count for standard speed. Table 4-4 provides recommended SSHCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I2C interface is disabled corresponding to the ENBL register being set to zero. Writes at other times have no effect.

The minimum valid value is six; hardware prevents values less than six being written. If the CPU attempts to write a value less than six a value of six results.

Table 4-4. Recommended SSHCNT Values

$pclk_{freq}$ (MHz)	I^2C Run Clock Rate	ic_clk_{freq} (MHz)	SSHCNT Value (hex)	Actual SCL High Time (ns)	I^2C Data Rate (kbps)
32	1x	32	0098	5000	100
16	1x	16	0048	5000	100
8	1x	8	0020	5000	100
4	1x	4	000C	5000	100
2	1x	2	0006	7000	83.33
2	3x	6	0016	5000	100
1	3x	3	0007	5000	100

4.6.6 Standard Speed I^2C Clock SCL Low Count (SSLCNT) Register

Base + $C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SSLCNT															
Write																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0

Figure 4-17. Standard Speed I^2C Clock SCL Low Count (SSLCNT) Register

4.6.6.1 Standard Speed Low Count (SSLCNT)—15–0

To ensure proper I/O timing, this register must be set before any standard speed master mode I^2C operations can take place. This register sets the SCL clock low period count for standard speed. Table 4-5 provides recommended SSLCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I^2C interface is disabled, corresponding to the ENBL register being set to zero. Writes at other times have no effect.

The minimum valid value is eight; hardware prevents values less than eight being written. If the CPU attempts to write a value less than eight a value of eight results.

Table 4-5. Recommended SSLCNT Values

$pclk_{freq}$ (MHz)	I^2C Run Clock Rate	ic_clk_{freq} (MHz)	SSLCNT Value (hex)	Actual SCL Low Time (ns)	I^2C Data Rate (kbps)
32	1x	32	009F	5000	100
16	1x	16	004F	5000	100
8	1x	8	0027	5000	100
4	1x	4	0013	5000	100
2	1x	2	0009	5000	83.33
2	3x	6	001D	5000	100
1	3x	3	000E	5000	100

4.6.7 Fast Speed I2C Clock SCL High Count (FSHCNT) Register

Base + $E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FSHCNT															
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Figure 4-18. Fast Speed I2C Clock SCL High Count (FSHCNT) Register

4.6.7.1 Fast Speed High Count (FSHCNT)—Bits 15–0

To ensure proper I/O timing, this register must be set before any fast speed master mode I2C operations can take place. This register sets the SCL clock high-period count for fast speed. Table 4-6 provides recommended FSHCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I2C interface is disabled, corresponding to the ENBL register being set to zero. Writes at other times have no effect. The minimum valid value is six; hardware prevents values less than six being written. If the CPU attempts to write a value less than eight, a value of eight results.

Table 4-6. Recommended FSHCNT Values

$pclk_{freq}$ (MHz)	I2C Run Clock Rate	ic_clk_{freq} (MHz)	FSHCNT value (hex)	Actual SCL High Time (ns)	I2C Data Rate (kbps)
32	1x	32	001E	1187.5	400
16	1x	16	000B	1187.5	400
8	1x	8	0006	1750	320
4	1x	4	0006	3500	173.91
2	1x	2	0006	7000	86.96
1	1x	1	0006	14000	43.48
4	3x	12	0006	1166.67	400
8	3x	24	0014	1166.67	400

4.6.8 Fast Speed I2C Clock SCL Low Count (FSLCNT) Register

Base + $10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FSLCNT															
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

Figure 4-19. Fast Speed I2C Clock SCL Low Count (FSLCNT) Register

4.6.8.1 Fast Speed SCL Low Count (FSLCNT)—Bits 15–0

To ensure proper I/O timing, this register must be set before any fast speed master mode I^2C operations can take place. This register sets the SCL clock low period count for fast speed. Table 4-7 provides recommended FSLCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I^2C interface is disabled, corresponding with the ENBL register being set to zero. Writes at other times have no effect.

The minimum valid value is eight; hardware prevents values less than eight being written. If the CPU attempts to write a value less than eight a value of eight results.

Table 4-7. Recommended FSLCNT Values

$pclk_{freq}$ (MHz)	I^2C Run Clock Rate	ic_clk_{freq} (MHz)	FSLCNT value (hex)	Actual SCL Low Time (ns)	I^2C Data Rate (kbps)
32	1x	32	0029	1312.5	400
16	1x	16	0014	1312.5	400
8	1x	8	000A	1375	320
4	1x	4	0008	2250	173.91
2	1x	2	0008	4500	86.96
1	1x	1	0008	9000	43.48
4	3x	12	000F	1333.33	400
8	3x	24	001F	1333.33	400

4.6.9 Interrupt Status (ISTAT) Register

Each bit in this register has a corresponding mask bit in the interrupt mask (IMASK) register. Except for TXEMPTY and RXFULL, these bits are cleared by reading the matching interrupt clear register. The unmasked raw versions of these bits are available in the raw interrupt status (RISTAT) register.

NOTE:

Writing zero to the ENBL register while the I^2C is in an enabled state can affect the values of the ISTAT and RISTAT registers. Please see Section 4.6.25 for additional details.

Base + $16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-20. Interrupt Status (ISTAT) Register

4.6.9.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.9.2 General Call (GC)—Bit 11

This bit is set only when a general call (GC) address is received and it is acknowledged. The GC bit is cleared by reading the CLRGC register.

4.6.9.3 Start Detect (STDET)—Bit 10

This bit indicates a START or repeated START condition occurred in the I2C interface regardless whether the I2C module is operating in slave or master mode. The STDET bit is cleared by reading the CLRSTDET register.

4.6.9.4 Stop Detect (STPDET)—Bit 9

This bit indicates a STOP condition occurred in the I2C interface regardless whether the I2C module is operating in slave or master mode. The STPDET bit is cleared by reading the CLRSTPDET register.

4.6.9.5 Activity (ACT)—Bit 8

This bit is set by hardware and cleared by software. It is set to one when the bus transitions from an idle state to a busy state. The ACT bit is cleared by reading the CLRACT register.

NOTE:

This bit remains set if the CPU attempts to clear it while the I2C bus is busy.

4.6.9.6 Transmit Done (TXDONE)—Bit 7

When the I2C module is acting as a slave-transmitter, this bit is set to one if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating the transmission is completed. The TXDONE bit is cleared by reading the CLRTXDONE register.

4.6.9.7 Transmit Abort (TXABRT)—Bit 6

This bit is set to one when the I2C module is unable to complete a command the processor requested. The TXABRT bit is cleared by reading the CLRTXABRT register. Conditions to set TXABRT are:

- If the processor attempts to issue a read command during slave-transmitter operation
- The I2C loses arbitration while transmitting data in slave mode
- When a read request interrupt occurs and the processor previously placed data in the TX buffer not yet transmitted. This data could be intended to service a multi-byte read request (RDREQ) bit resulting in fewer numbers of bytes requested.
- The I2C loses arbitration while operating in master mode
- Attempting to send a master command when configured only to be a slave
- RSTEN bit in the control register is set to zero, disabling the repeated START condition, and the processor attempts to issue a START byte in the I2C bus. This is considered a TXABRT because the I2C function is impossible to perform without using repeated START conditions.
- RSTEN bit in the control register is set to zero, disabling the repeated START condition, and the processor attempts to issue a 10-bit addressing read command in the I2C bus. This is considered a TXABRT because the I2C function is impossible to perform without using repeated START conditions.

- Start byte is generated in master mode but is acknowledged
- If a read command is issued after a general call command was issued
- General call address is generated in master mode but is not acknowledged
- The addressed remote slave receiver does not acknowledge a byte of data
- No remote slave acknowledged the second address byte of a 10-bit address
- No remote slave acknowledged the first address byte of a 10-bit address
- No remote slave acknowledged the address byte of a 7-bit address

Anytime this bit is set, the contents of the transmit and receive buffers are flushed. The transmit buffer is held in a flushed state until the CPU clears the TXABRT bit.

4.6.9.8 Read Request (RDREQ)—Bit 5

This bit is set to one when a remote I^2C master is attempting to read data from the I^2C module. The I^2C module operates as a slave-transmitter. The I^2C module holds the I^2C bus in a wait state (SCL = 0) until the processor writes data to the TXFIFO. The processor services the read request by writing data to the data register. The RDREQ bit is cleared by reading the CLRRDREQ register.

4.6.9.9 Transmit Empty (TXEMPTY)—Bit 4

This bit is set to one when the transmit buffer is at, or below the threshold value set in the transmit FIFO threshold (TXFT) register. TXEMPTY is automatically cleared by hardware when the buffer level goes above the threshold.

4.6.9.10 Transmit Overrun (TXOVR)—Bit 3

Set during transmit if the transmit buffer is filled to transmit buffer depth and the processor attempts to issue another I^2C command by writing to the data register. Clear the TXOVR bit by reading the CLRTXOVR register.

4.6.9.11 Receive Full (RXFULL)—Bit 2

This bit is set when the receive buffer reaches, or goes above the receive threshold level in the receive FIFO threshold (RXFT) register. RXFULL is automatically cleared by hardware when the buffer level goes below the threshold. If the module is disabled (EN = 0), the RXFIFO is flushed and held in reset; therefore the RXFIFO is not full. So this bit is cleared once the EN bit in the ENBL register is programmed with a zero, regardless of the continuing activity.

4.6.9.12 Receive Overrun (RXOVR)—Bit 1

This bit is set if the receive buffer is completely filled to receive buffer depth and an additional data byte is received from a remote I^2C device. The I^2C module acknowledges this additional data byte, but any data bytes received after the FIFO is full are lost. The RXOVR bit is cleared by reading the CLRRXOVR register.

4.6.9.13 Receive Underrun (RXUND)—Bit 0

This bit is set if the processor attempts to read the receive buffer when it is empty by reading from the DATA register. The RXUND bit is cleared by reading the CLRRXUND register.

4.6.10 Interrupt Enable (IENBL) Register

Bits in this register enable their corresponding interrupt status bits in the ISTAT register. They are active high; a value of zero prevents a bit from generating an interrupt.

Base + $18	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
Write																
Reset	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1

Figure 4-21. Interrupt Enable (IENBL) Register

4.6.11 Raw Interrupt Status (RISTAT) Register

Unlike the ISTAT register, these bits are not masked, so they always show the true status of the I2C module.

NOTE:

Writing zero to the ENBL register while the I2C is in an enabled state can affect the values of the ISTAT and RISTAT registers. Please see Section 4.6.25 for additional details.

Base + $1A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-22. Raw Interrupt Status (RISTAT) Register

4.6.11.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.11.2 General Call (GC)—Bit 11

This bit is set only when a general call (GC) address is received and it is acknowledged. The GC bit is cleared by reading the CLRGC register.

4.6.11.3 Start Detect (STDET)—Bit 10

This bit indicates a START or repeated START condition occurred in the I2C interface regardless whether the I2C module is operating in slave or master mode. The STDET bit is cleared by reading the CLRSTDET register.

4.6.11.4 Stop Detect (STPDET)—Bit 9

This bit indicates a STOP condition occurred in the I2C interface regardless whether the I2C module is operating in slave or master mode. The STPDET bit is cleared by reading the CLRSTPDET register.

4.6.11.5 Activity (ACT)—Bit 8

This bit is set by hardware and cleared by software. It is set to one when the bus transitions from an idle state to a busy state. The ACT bit is cleared by reading the CLRACT register.

NOTE:

This bit remains set if the CPU attempts to clear it while the I^2C bus is busy.

4.6.11.6 Transmit Done (TXDONE)—Bit 7

When the I^2C module is acting as a slave-transmitter, this bit is set to one if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating the transmission is completed. The TXDONE bit is cleared by reading the CLRTXDONE register.

4.6.11.7 Transmit Abort (TXABRT)—Bit 6

This bit is set to one when the I^2C module is unable to complete a command the processor requested. The TXABRT bit is cleared by reading the CLRTXABRT register. Conditions to set TXABRT are:

- If the processor attempts to issue a read command during slave-transmitter operation
- The I^2C loses arbitration while transmitting data in slave mode
- When a read request interrupt occurs and the processor previously placed data in the TX buffer not yet transmitted. This data could be intended to service a multi-byte read request (RDREQ) bit resulting in fewer numbers of bytes requested.
- The I^2C loses arbitration while operating in master mode
- Attempting to send a master command when configured only to be a slave
- RSTEN bit in the control register is set to zero, disabling the repeated START condition, and the processor attempts to issue a START byte in the I^2C bus. This is considered a TXABRT because the I^2C function is impossible to perform without using repeated START conditions.
- RSTEN bit in the control register is set to zero, disabling the repeated START condition, and the processor attempts to issue a 10-bit addressing read command in the I^2C bus. This is considered a TXABRT because the I^2C function is impossible to perform without using repeated START conditions.
- Start byte is generated in master mode but is acknowledged
- If a read command is issued after a general call command was issued
- General call address is generated in master mode but is not acknowledged
- The addressed remote slave receiver does not acknowledge a byte of data
- No remote slave acknowledged the second address byte of a 10-bit address
- No remote slave acknowledged the first address byte of a 7-bit address
- No remote slave acknowledged the address byte of a 7-bit address

Anytime this bit is set, the contents of the transmit and receive buffers are flushed. The transmit buffer is held in a flushed state until the CPU clears the TXABRT bit.

4.6.11.8 Read Request (RDREQ)—Bit 5

This bit is set to one when a remote I2C master is attempting to read data from the I2C module. The I2C module operates as a slave-transmitter. The I2C module holds the I2C bus in a wait state (SCL = 0) until the processor writes data to the TXFIFO. The processor services the read request by writing data to the data register. The RDREQ bit is cleared by reading the CLRRDREQ register.

4.6.11.9 Transmit Empty (TXEMPTY)—Bit 4

This bit is set to one when the transmit buffer is at, or below the threshold value set in the transmit FIFO threshold (TXFT) register. TXEMPTY is automatically cleared by hardware when the buffer level goes above the threshold.

4.6.11.10 Transmit Over (TXOVR)—Bit 3

Set during transmit if the transmit buffer is filled to transmit buffer depth and the processor attempts to issue another I2C command by writing to the data register. The TXOVR bit is cleared by reading the CLRTXOVR register.

4.6.11.11 Receive Full (RXFULL)—Bit 2

This bit is set when the receive buffer reaches, or goes above the receive threshold level in the receive FIFO threshold (RXFT) register. RXFULL is automatically cleared by hardware when the buffer level goes below the threshold. If the module is disabled (EN = 0), the RXFIFO is flushed and held in reset; therefore the RXFIFO is not full. So this bit is cleared once the EN bit in the ENBL register is programmed with a zero, regardless of the continuing activity.

4.6.11.12 Receive Over (RXOVR)—Bit 1

This bit is set if the receive buffer is completely filled to receive buffer depth and an additional data byte is received from a remote I2C device. The I2C module acknowledges this additional data byte, but any data bytes received after the FIFO is full are lost. The RXOVR bit is cleared by reading the CLRRXOVR register.

4.6.11.13 Receiver Under (RXUND)—Bit 0

This bit is set if the processor attempts to read the receive buffer when it is empty by reading from the DATA register. The RXUND bit is cleared by reading the CLRRXUND register.

4.6.12 Receive FIFO Threshold (RXFT) Register

Base + $1C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXTL	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 4-23. Receive FIFO Threshold (RXFT) Register

4.6.12.1 Receive FIFO Threshold Level (RXTL)—Bits 7–0

Receive FIFO threshold level controls the level of entries (or above) triggering the RXFULL bit in the RISTAT register. The valid range is 0–3, with the additional restriction hardware does not allow this threshold level to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set is the maximum depth of the buffer minus one.

A value of zero sets the threshold for one entry, and a value of three sets the threshold for four entries.

4.6.13 Transmit FIFO Threshold (TXFT) Register

Base + $1E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXTL	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-24. Transmit FIFO Threshold (TXFT) Register

4.6.13.1 Transmit FIFO Threshold Level (TXTL)—Bits 7–0

Transmit FIFO threshold level controls the level of entries (or below) triggering the TXEMPTY bit in the RISTAT register. The valid range is 0–3, with the additional restriction it may not be set to a threshold level larger than the depth of the buffer. If an attempt is made to do that, the actual value set is the maximum depth of the buffer minus one.

A value of zero sets the threshold for zero entries, and a value of three sets the threshold for three entries.

4.6.14 Clear Individual Interrupts (CLRINT) Register

Base + $20	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-25. Clear Individual Interrupts (CLRINT) Register

4.6.14.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.14.2 Clear Interrupts (INT)—Bit 0

This read-only bit is read to clear all individual interrupts, and the TXABRTSRC register. This bit clears only software interrupts capable of being cleared. Refer to bit nine in the TXABRTSRC register for an exception to clearing the TXABRTSRC register.

4.6.15 Clear Receive Under Interrupt (CLRRXUND) Register

Base + $22	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXUND
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-26. Clear Receive Under Interrupt (CLRRXUND) Register

4.6.15.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.15.2 Clear Receive Under (RXUND)—Bit 0

This read-only bit is read to clear the RXUND interrupt bit in the RISTAT register.

4.6.16 Clear Receive Over Interrupt (CLRRXOVR) Register

Base + $24	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXOVR
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-27. Clear Receive Over Interrupt (CLRRXOVR) Register

4.6.16.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.16.2 Clear Receive Over (RXOVR)—Bit 0

This read-only bit is read to clear the RXOVR interrupt bit in the RISTAT register.

4.6.17 Clear Transmit Over Interrupt (CLRTXOVR) Register

Base + $26	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXOVR
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-28. Clear Transmit Over Interrupt (CLRTXOVR) Register

4.6.17.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.17.2 Clear Transmit Over (TXOVR)—Bit 0

This read-only bit is read to clear the TXOVR interrupt bit in the RISTAT register.

4.6.18 Clear Read Request Interrupt (CLRRDREQ) Register

Base + $28	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RDREQ
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-29. Clear Read Request Interrupt (CLRRDREQ) Register

4.6.18.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.18.2 Clear Read Request Interrupt (RDREQ)—Bit 0

This read-only bit is read to clear the RDREQ interrupt bit in the RISTAT register.

4.6.19 Clear Transmit Abort Interrupt (CLRTXABRT) Register

Base + $2A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXABRT
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-30. Clear Transmit Abort Interrupt (CLRTXABRT) Register

4.6.19.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.19.2 Clear Transmit Abort Interrupt (TXABRT)—Bit 0

This read-only bit is read to clear the TXABRTSRC register and the TXABRT interrupt bit on RISTAT register Refer to bit nine in the TXABRTSRC register for an exception to clearing that register.

4.6.20 Clear Transmit Done Interrupt (CLRTXDONE) Register

Base + $2C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXDONE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-31. Clear Transmit Done Interrupt (CLRTXDONE) Register

4.6.20.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.20.2 Clear Transmit Done Interrupt (TXDONE)—Bit 0

This read-only bit is read to clear the TXDONE interrupt bit in the RISTAT register.

4.6.21 Clear Activity Interrupt (CLRACT) Register

Base + $2E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ACT
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-32. Clear Activity Interrupt (CLRACT) Register

4.6.21.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.21.2 Clear Activity Interrupt (ACT)—Bit 0

This read-only bit is read to clear the ACT interrupt bit in the RISTAT register if the I2C is no longer active. If the I2C module is still active in the bus, the ACT interrupt bit remains set.

4.6.22 Clear Stop Detect Interrupt (CLRSTPDET) Register

Base + $30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STPDET
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-33. Clear Stop Detect Interrupt (CLRSTPDET) Register

4.6.22.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.22.2 Clear Stop Detect Interrupt (STPDET)—Bit 0

This read-only bit is read to clear the STPDET interrupt bit in the RISTAT register.

4.6.23 Clear Start Detect Interrupt (CLRSTDET) Register

Base + $32	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STDET
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-34. Clear Start Detect Interrupt (CLRSTDET) Register

4.6.23.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.23.2 Clear Start Detect Interrupt (STDET)—Bit 0

This read-only bit is read to clear the STDET interrupt bit of the RISTAT register.

4.6.24 Clear General Call Interrupt (CLRGC) Register

Base + $34	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GC
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-35. Clear General Call Interrupt (CLRGCL) Register

4.6.24.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.24.2 Clear General Call Interrupt (GC)—Bit 0

This read-only bit is read to clear the GC interrupt bit on RISTAT register.

4.6.25 Enable (ENBL) Register

Base + $36	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-36. Enable (ENBL) Register

4.6.25.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.25.2 Enable (EN)—Bit 0

This bit controls when the I2C module is enabled.

- 0 = Disables I2C module
- 1 = Enables I2C module

The ACT interrupt bit in the RISTAT register can be polled to determine if the I2C module is active. When I2C module is disabled, (see Section 4.5.2.4.2) the following occurs:

- RXFIFO gets flushed
- The TXFLR and RXFLR registers are cleared.
- Status bits in the RISTAT register remain active until I2C module goes into idle state.
- Status bits in the ISTAT register remain active until I2C module goes into idle state.

NOTE:

Avoid writes to the DATA register when the ENBL register is zero. Writing to the DATA register while the ENBL register is zero may result in unpredictable operation of the I2C module.

4.6.26 Status (STAT) Register

Base + $38	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	SLV ACT	MST ACT	RFF	RFNE	TFE	TFNF	ACT
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 4-37. Status (STAT) Register

4.6.26.1 Reserved—Bits 15–7

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.26.2 Slave FSM Activity Status (SLVACT)—Bit 6

This bit is set when the slave finite sate machine (FSM) is not in the idle state.

- 0 = Slave FSM is in idle state so the slave part of I^2C module is not active
- 1 = Slave FSM is not in idle state so the slave part of I^2C module is active

NOTE:

Bit 0 (ACT) is the OR of bits six (SLVACT) and five (MSTACT).

4.6.26.3 Master FSM Activity Status (MSTACT)—Bit 5

This bit is set when the master finite state machine (FSM) is not in the idle state.

- 0 = Master FSM is in idle state so the master part of I^2C module is not active
- 1 = Master FSM is not in idle state so the master part of I^2C module is active

NOTE:

Bit 0 (ACT) is the OR of bits six (SLVACT) and five (MSTACT).

4.6.26.4 Receive FIFO Completely Full (RFF)—Bit 4

This bit is set when the receive FIFO is completely full. This bit is cleared when the receive FIFO contains one or more empty locations.

- 0 = Receive FIFO is not full
- 1 = Receive FIFO is full

4.6.26.5 Receive FIFO Not Empty (RFNE)—Bit 3

This bit is set when the receive FIFO contains one or more entries, and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO.

- 0 = Receive FIFO is empty
- 1 = Receive FIFO is not empty

4.6.26.6 Transmit FIFO Completely Empty (TFE)—Bit 2

This bit is set when the transmit FIFO is completely empty. It is cleared when it contains one or more valid entries.

- 0 = Transmit FIFO is not empty
- 1 = Transmit FIFO is empty

4.6.26.7 Transmit FIFO Not Full (TFNF)—Bit 1

This bit is set when the transmit FIFO contains one or more empty locations. It is cleared when the FIFO is full.

- 0 = Transmit FIFO is full
- 1 = Transmit FIFO is not full

4.6.26.8 Activity Status (ACT)—Bit 0

I2C activity status.

NOTE:

Bit 0 (ACT) is the OR of bits six (SLVACT) and five (MSTACT).

4.6.27 Transmit FIFO Level Register (TXFLR)

This register contains the number of valid data entries in the TXFIFO buffer. It is cleared whenever:

- There is no data in the TXFIFO
- The I2C is disabled
- There is a transmit abort—that is, TXABRT bit is set in the RISTAT register
- The slave bulk transmit mode is aborted

The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

Base + $3A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	TXFL		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-38. Transmit FIFO Level Register (TXFLR)

4.6.27.1 Reserved—Bits 15–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.27.2 Transmit FIFO Level (TXFL)—Bits 2–0

This bit field contains the number of valid data entries in the TXFIFO.

4.6.28 Receive FIFO Level Register (RXFLR)

This register contains the number of valid data entries in the RXFIFO buffer. It is cleared whenever:

- There is no data in the RXFIFO
- The I^2C is disabled
- There is a transmit abort—that is, TXABRT bit is set in the RISTAT register
- The slave bulk transmit mode is aborted

The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

Base + $3C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	RXFL		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-39. Receive FIFO Level Register (RXFLR)

4.6.28.1 Reserved—Bits 15–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.28.2 Receive FIFO Level (RXFL)—Bits 2–0

This bit contains the number of valid data entries in the RXFIFO.

4.6.29 Transmit Abort Source (TXABRTSRC) Register

This read-only register has 14 bits indicating the source of the TXABRT bit in the RISTAT register. Except for bit nine in this register, all bits are cleared whenever the processor reads the CLRTXABRT or the CLRINT registers.

To clear bit nine, abort START byte no repeated START (SNORST) in this register, its source must be fixed first; repeated START must be enabled (CTRL[5] = 1), bit 11 (SPCL) must be cleared (TAR[11] = 0), or bit 10 (GCSTRT) must be cleared (TAR[10] = 0). Once the source of SNORST is fixed, this bit can be cleared in the same manner as other bits in this register. If the source of the SNORST is not fixed before attempting to clear this bit, bit nine clears for one cycle and is then re-asserted.

Base + $40	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SLV RD	SLV AL	SLV FLSH	AL	MST DIS	RNOR ST	SNOR ST	0	SACK DET	0	GC READ	GC NACK	TD NACK	AD2 NACK	AD1 NACK	AD7 NACK
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-40. Transmit Abort Source (TXABRTSRC) Register

4.6.29.1 Abort Slave Read in Transmit (SLVRD)—Bit 15

The I2C was expecting the processor to service a read request by writing a write command to the DATA register (DATA[8] = 0), but the processor wrote a read command to the DATA register (DATA[8] = 1).

4.6.29.2 Slave Arbitration Lost (SLVAL)—Bit 14

Slave lost arbitration while transmitting data to a remote master. Bit 12 in the TXABRTSRC register is set at the same time.

4.6.29.3 Abort Slave Flush Transmit FIFO (SLVFLSH)—Bit 13

Slave received a read command but data and/or commands still exists in the TXFIFO, so the slave issues a TXABRT interrupt to flush old data in TXFIFO.

4.6.29.4 Arbitration Lost (AL)—Bit 12

This bit indicates an arbitration loss occurred. If bits 12 and 14 are set, the slave lost arbitration. However, if bit 12 is set and bit 14 is clear, the master lost arbitration.

4.6.29.5 Abort Master Disabled (MSTDIS)—Bit 11

The processor attempts a Master mode command, but the master mode is disabled.

4.6.29.6 Abort 10B Read No Repeated Start (RNORST)—Bit 10

The repeated START is disabled (RSTEN bit (CTRL[5] = 0) and the processor attempts a read command in master 10-bit addressing mode.

4.6.29.7 Abort Start Byte No Repeated Start (SNORST)—Bit 9

The repeated START is disabled (RSTEN) bit (CTRL[5] = 0) and the processor attempts to generate a START byte on the bus in master mode. To clear bit nine, the source of the abort START byte no restart (SNORST) bit in this register must be fixed first; repeated START must be enabled (CTRL[5] = 1), the SPCL bit must be cleared (TAR[11] = 0), or the GCSTRT bit must be cleared (TAR[10] = 0). Once the source of the SNORST bit is fixed, this bit can be cleared in the same manner as other bits in this register. However, if the source of the SNORST bit is not fixed before attempting to clear this bit, bit nine clears for one cycle and then becomes reasserted.

4.6.29.8 Reserved—Bit 8

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.29.9 Abort Start Byte Acknowledge Detect (SACKDET)—Bit 7

The I2C sent a START byte in master mode, and the START byte was acknowledged.

4.6.29.10 Reserved—Bit 6

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

4.6.29.11 Abort General Call Read (GCREAD)—Bit 5

The TAR register was configured for general call (TAR[11] = 1 and TAR[10] = 0), but the processor wrote a read command to the DATA register (DATA[8] = 1).

4.6.29.12 Abort General Call No Acknowledge (GCNACK)—Bit 4

The I^2C attempted a general call transfer in master mode, but none of the bus slaves acknowledged the general call address.

4.6.29.13 Abort Transmit Data No Acknowledge (TDNACK)—Bit 3

This is a master mode only bit. Master received an acknowledgement for the address, but did not receive an acknowledgement on one of the transmitted data bytes.

4.6.29.14 Abort 10B Address2 No Acknowledge (AD2NACK)—Bit 2

The I^2C attempted a 10-bit addressing transfer in master mode, but none of the bus slaves acknowledged the second address byte of the 10-bit address.

4.6.29.15 Abort 10B Address1 No Acknowledge (AD1NACK)—Bit 1

The I^2C attempted a 10-bit addressing transfer in master mode, but none of the bus slaves acknowledged the first address byte of the 10-bit address.

4.6.29.16 Abort 7B Address No Acknowledge (AD7NACK)—Bit 0

The I^2C attempted a 7-bit addressing transfer in master mode, but none of the bus slaves acknowledged the address byte of the 7-bit address.

4.7 Interrupt

4.7.1 Operation of the Interrupt Registers

Table 4-8. I^2C Interrupt Summary

Interrupt	Interrupt Vector	Description
Error	I^2C_Vector_Base + $0	Error interrupt
General	I^2C_Vector_Base + $1	General call interrupt
Receive	I^2C_Vector_Base + $2	Receive interrupt
Transmit	I^2C_Vector_Base + $3	Transmit interrupt
Status	I^2C_Vector_Base + $4	Status interrupt

Bits in the IENBL register enable their corresponding status bits in the ISTAT register. Unlike the ISTAT register, bits in the RISTAT register are not masked, so they always show the true status of the I^2C module.

The following tables and figure illustrate the operation of the I^2C module ISTAT bits and how they are set and cleared.

Table 4-9. I^2C Interrupt Sources

ISTAT Bit	IENBL Local Enable	Interrupt	Description
TXABRT	TXABRT	Error	Transmit abort
TXOVR	TXOVR		Transmit overrun
RXOVR	RXOVR		Receive overrun
RXUND	RXUND		Receive underrun
GC	GC	General	General call
RXFULL	RXFULL	Receive	Receive full
TXDONE	TXDONE	Transmit	Transmit done
RDREQ	RDREQ		Read request
TXEMPTY	TXEMPTY		Transmit empty
STDET	STDET	Status	Start detect
STPDET	STPDET		Stop detect
ACT	ACT		Activity

Table 4-10. Setting and Clearing of Interrupt Bits

ISTAT Bit	Hardware Action To Set	Action Required to Clear
GC	General call address occurred on the I^2C bus	CPU read of CLRGC register
STDET	START or repeated START condition occurred on the I^2C bus	CPU read of CLRSTDET register
STPDET	STOP condition occurred on the I^2C bus	CPU read of CLRSTPDET register
ACT	Detection of I^2C bus busy	CPU read of CLRACT register (please see note in Section 4.7.1.5)
TXDONE	Detection of a NACK data byte during slave-transmitter operation	CPU read of CLRTXDONE register
TXABRT	The I^2C module was unable to complete a command the processor requested	CPU read of CLRTXABRT register
RDREQ	I^2C is operating in slave-transmitter mode and data is needed in the Tx FIFO	CPU read of CLRRDREQ register
TXEMPTY	Tx FIFO level is at or below the threshold determined by the TXFT register	CPU write(s) to the Tx FIFO until level is above TXFT threshold
TXOVR	CPU attempted a write to the Tx FIFO, but the Tx FIFO is full	CPU read of CLRTXOVR register
RXFULL	Rx FIFO level is at or above the threshold determined by the RXFT reg	CPU read(s) of the Rx FIFO until level is below RXFT threshold
RXOVR	A data byte was received from the I^2C bus, but the Rx FIFO was full	CPU read of CLRRXOVR register
RXUND	CPU attempted to read the Rx FIFO, but the Rx FIFO was empty	CPU read of CLRRXUND register

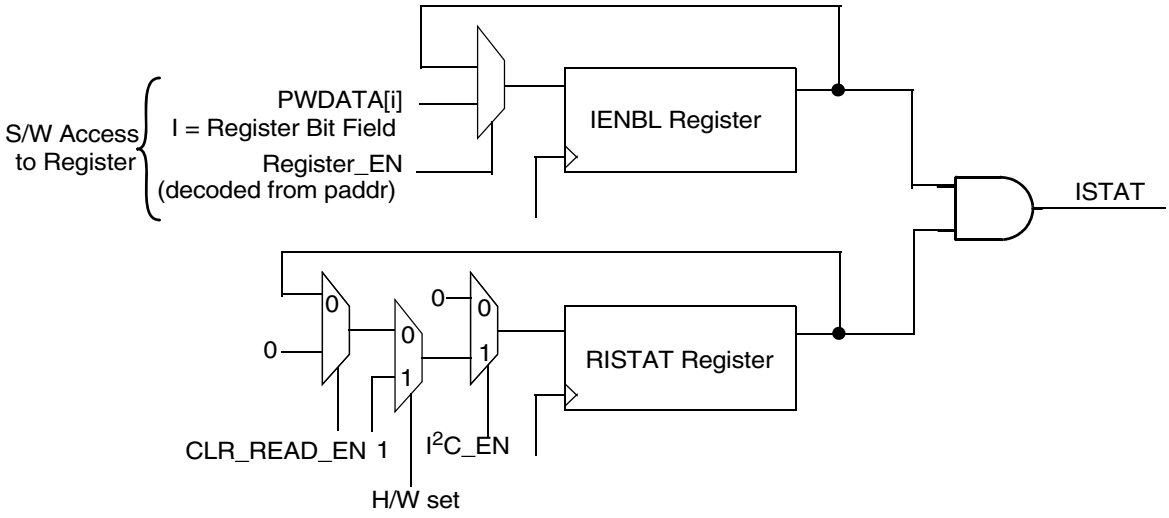


Figure 4-41. Interrupt Scheme for Software Cleared Bits

4.7.1.1 Error Interrupt

This interrupt is enabled by setting one or more of the interrupt enables associated with error. The enables are named TXABRT, TXOVR, RXOVR, and RXUND and are located in the IENBL register. The interrupt service routine should read the ISTAT register to determine what error handling is required. The TXABRT bit is cleared by reading the CLRTXABRT register. The TXOVR bit is cleared by reading the CLRTXOVR register. The RXOVR bit is cleared by reading the CLRRXOVR register. The RXUND bit is cleared by reading the CLRRXUND register.

4.7.1.2 General Interrupt

This interrupt is enabled by setting the GC bit located in the IENBL register. The GC bit is cleared by reading the CLRGC register.

4.7.1.3 Receive Interrupt

This interrupt is enabled by setting the RXFULL bit located in the IENBL register. The RXFULL bit is automatically cleared by hardware when the receive buffer level goes below the threshold.

4.7.1.4 Transmit Interrupt

This interrupt is enabled by setting one or more of the interrupt enables associated with transmit. The enables are named TXDONE, RDREQ, and TXEMPTY and are located in the IENBL register. The interrupt service routine should read the ISTAT register to determine what transmit handling is required. The TXDONE bit is cleared by reading the CLRTXDONE register. The RDREQ bit is cleared by reading the CLRRDREQ register. The TXEMPTY bit is automatically cleared by hardware when the transmit buffer level goes above the threshold.

4.7.1.5 Status Interrupt

This interrupt is enabled by setting one or more of the interrupt enables associated with bus condition detects and activity. The enables are named STDET, STPDET, and ACT and are located in the IENBL register. The interrupt service routine should read the ISTAT register to determine what status handling is required. The STDET bit is cleared by reading the CLRSTDET register. The STPDET bit is cleared by reading the CLRSTPDET register. The ACT bit is cleared by reading the CLRACT register.

NOTE:

The ACT bits in the ISTAT and RISTAT registers remain set if the CPU attempts to clear ACT while the I^2C bus is busy.

4.7.1.6 Recover From Wait or Stop Mode

Any enabled I^2C interrupt request can bring the CPU out of wait or stop mode.

Table 4-11. Document Revision History for Chapter 4

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 5
On-Clock Chip Synthesis (OCCS)

5.1 Introduction

This module provides the raw 2× system clock frequency to the system integration module (SIM), further modifying it to generate the various chip clocks. This module also produces the OSC_CLK control signals.

The on-chip clock synthesis module allows product design using an internal relaxation oscillator to run the 56F8000 at user selectable frequencies up to 32 MHz.

5.2 Features

The on-chip clock synthesis (OCCS) module interfaces to the oscillator and PLL. The OCCS module features follow:

- Internal relaxation oscillator
- Crystal oscillator control
- Ability to power down internal relaxation oscillator or crystal oscillator
- Ability to put the internal relaxation oscillator into a standby mode
- Three-bit postscaler provides control for the PLL output
- Ability to power down the internal PLL
- Provides 2× master clock frequency and OSC_CLK signals
- Safety shutdown feature available in the event the PLL reference clock disappears
- Can be driven from an external clock source
- Provides 3× high speed clock to timer and PWM modules

The clock generation module provides the programming interface for both the PLL and internal relaxation oscillator.

5.3 Block Diagram

Figure 5-1 illustrates the block diagrams of the clock generation module.

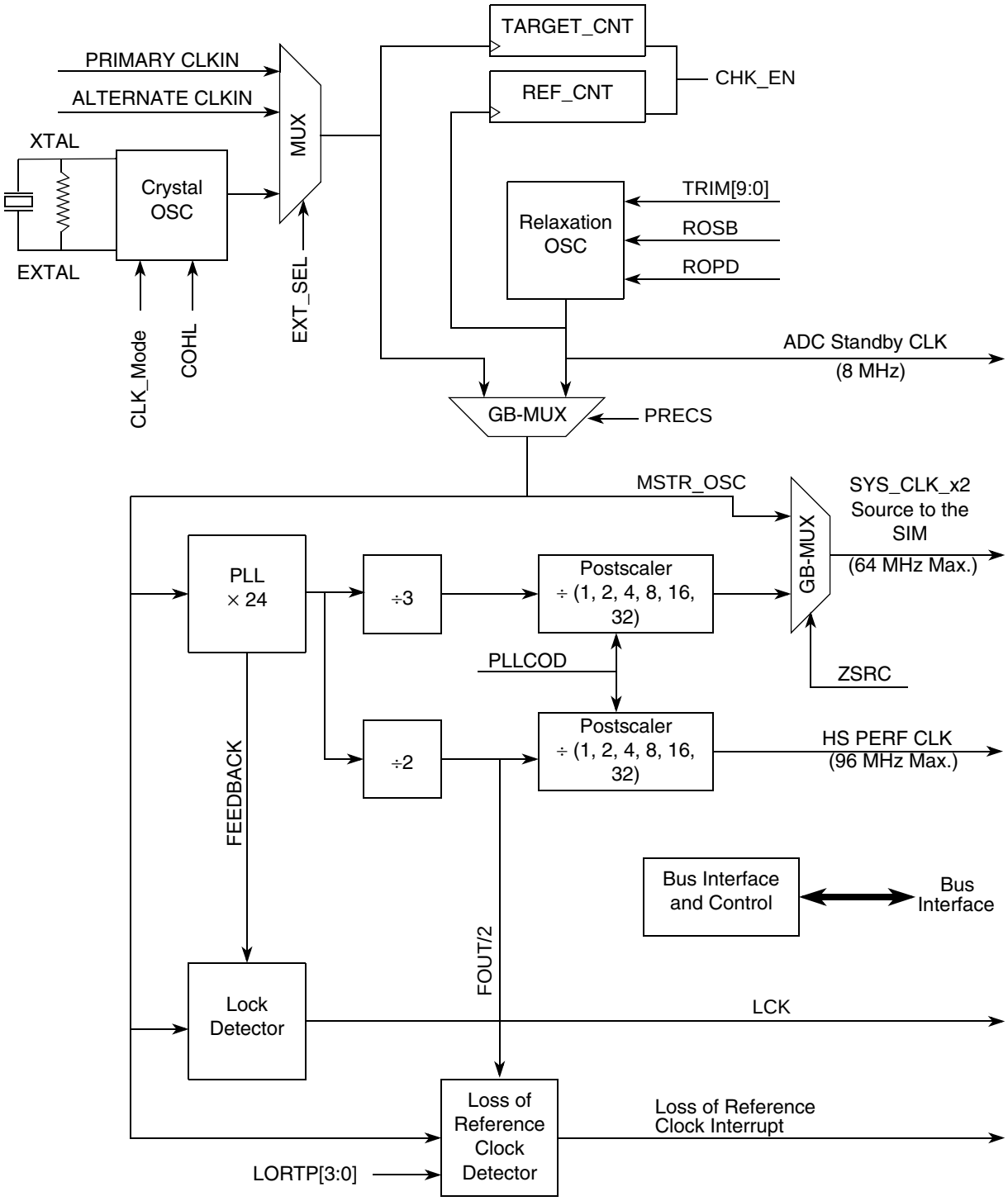


Figure 5-1. OCCS Block Diagram with Crystal Oscillator

5.4 Operation Modes

Either an internal oscillator, or an external frequency source can be used to provide a reference clock (SYS_CLK_×2) to the 56F8000 SIM.

The 2× system clock source output from the OCCS can be described by one of the following equations:

- 2× system frequency = oscillator frequency
- 2× system frequency = (oscillator frequency × 8) / (postscaler)

  while:

  postscaler = 1, 2, 4, 8, 16, or 32

The SIM is responsible for further dividing these frequencies by two, ensuring a 50% duty cycle in the system clock output.

5.4.1 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 very little variability with temperature and voltage, but it does vary as much as ± 20% as a function of wafer fabrication process. It also is very fast in reaching a stable frequency, well under 1 µsec. 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 is enabled by default. Application code can then switch to the external source and power down the internal oscillator if desired.

5.4.2 Crystal Oscillator

The internal crystal oscillator circuit is designed to interface with a parallel resonant crystal resonator in the frequency range, specified for the external crystal, of 4 – 8 MHz. Figure 5-2 illustrates a typical crystal oscillator circuit. Follow the crystal supplier's recommendations when selecting a crystal because 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.

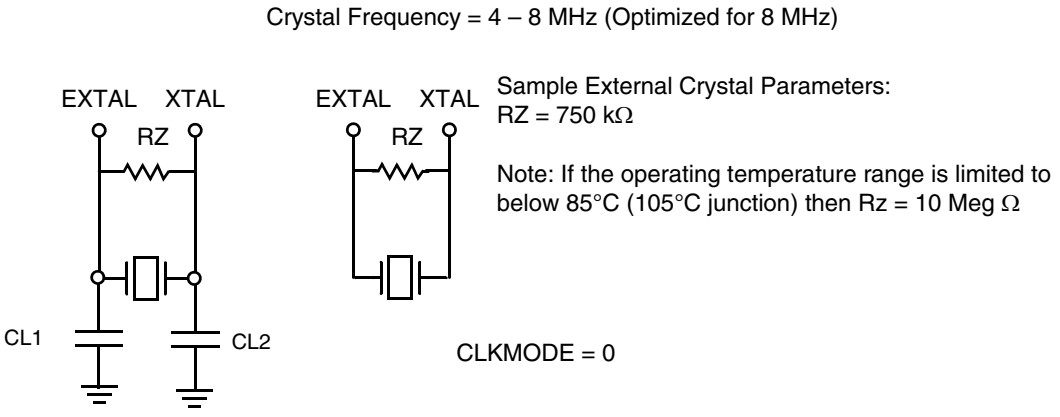


Figure 5-2. External Crystal Oscillator Circuit

5.4.3 Ceramic Resonator

The internal crystal oscillator circuit is also designed to interface with a ceramic resonator in the frequency range, of 4 – 8 MHz. Figure 5-3 illustrates the typical two and three terminal ceramic resonators and their circuits. Follow the resonator supplier's recommendations when selecting a resonator because their parameters determine the component values required to provide maximum stability and reliable start-up. The load capacitance values used in the resonator circuit design should include all stray layout capacitances. The resonator 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.

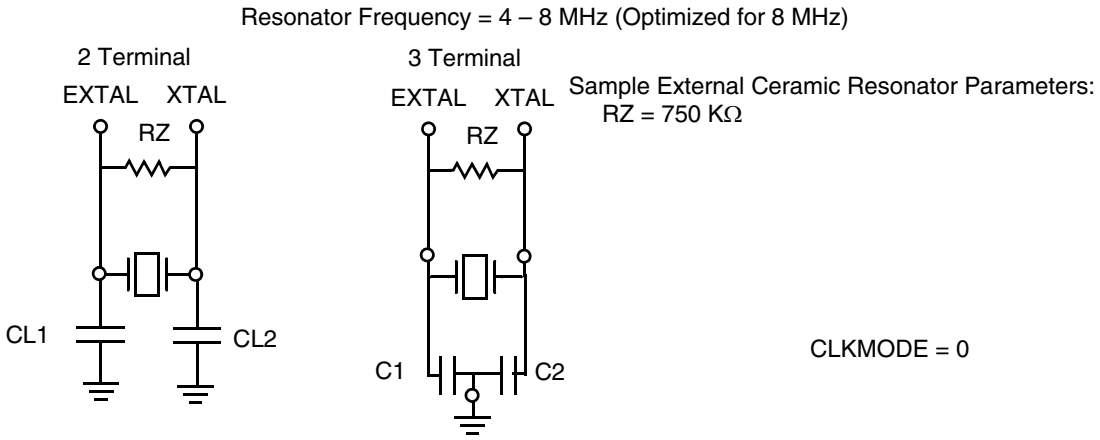


Figure 5-3. External Ceramic Resonator Circuit

5.4.4 External Clock Source: Crystal Oscillator Option

The recommended method of connecting an external clock is provided in Figure 5-6. The external clock source is connected to XTAL. The EXTAL pin is grounded. The external clock input must be generated using a relatively low impedance driver, because the XTAL pin is actually the output pin of the oscillator; it has a very weak driver.

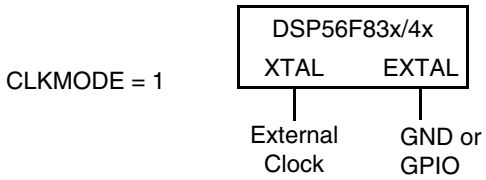


Figure 5-4. Connecting an External Clock Signal Using XTAL

It is possible to drive EXTAL with an external clock, though this is not the recommended method. To drive EXTAL with an external clock source the following conditions must be met:

1. XTAL must be completely unloaded
2. Maximum frequency of the applied clock must be less than 8 MHz

Figure 5-5 illustrates how to connect an external clock circuit with an external clock source using EXTAL as the input.

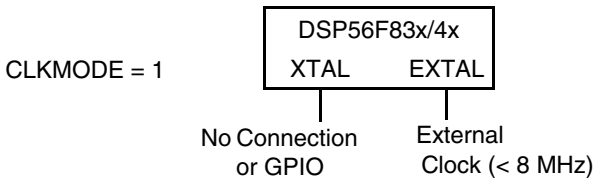


Figure 5-5. Connecting an External Clock Signal Using EXTAL

5.4.5 External Clock Source: GPIO

The recommended method of connecting an external clock is illustrated in Figure 5-6. The external clock source is connected to a GPIO pin.

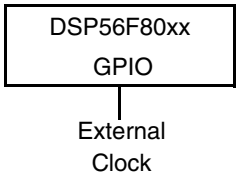


Figure 5-6. Connecting an External Clock Signal Using GPIO

5.5 Functional Description

A block diagram of the OCCS module is provided in Figure 5-1 or Figure 5-2, depending on whether the crystal oscillator is included on the chip.

Possible clock source choices are:

- Internal relaxation oscillator
- External ceramic resonator (on some chips)
- External crystal (on some chips)
- External clock source

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 therefore generating a higher frequency clock for use within the chip. This allows for two final clock output selections:

1. Direct oscillator output
2. Postscaler output

The clock multiplexer (ZSRC MUX) selects the direct clock on power up. A different clock source can be selected by writing to the control (CTRL) register. 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.

Transitions to/from direct to postscaler frequencies are guaranteed to be glitch free. Before switching to the PLL, the PLL must be locked. The status (STAT) register shows the status of the core clock source. Because the synchronizing circuit changes modes to avoid any glitches, the STAT register ZCLOCK (ZSRC) source shows overlapping modes as an intermediate step. After PLL lock is detected the core clock can be switched to the PLL by writing to the ZSRC bits in the CTRL register.

Frequencies going out of the PLL are controlled by the postscaler. The PLL operates at a fixed divide-by ratio of 24. The input frequency multiplied by the divide-by ratio is the frequency at which the VCO is running.

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 up or down, the PLL be deselected as the clocking source. Only after lock is achieved should the PLL be used as a valid clocking source.

Table 5-1 provides the possible clock sources and configurations.

Table 5-1. Clock Choices

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. 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 PLLCOD, 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)
Ceramic Resonator	Direct	1. Crystal Oscillator should be powered up (CLK_MODE = 0) 2. Set Ext-Sel for Crystal Oscillator 3. Wait for the crystal oscillator to stabilize (up to 10 ms) 4. The clock source should be changed to the Crystal Oscillator (PRECS = 1) 5. Wait 4 NOPs for the synchronizing circuit to change clocks 6. Relaxation oscillator can be powered down (ROPD = 1) to conserve power
Ceramic Resonator	Postscaler	1. Crystal Oscillator should be powered up (CLK_MODE = 0) 2. Set Ext-Sel for Crystal Oscillator 3. Wait for the Crystal Oscillator to stabilize (up to 10 ms) 4. The clock source should be changed to the Crystal Oscillator (PRECS = 1) 5. Wait 4 NOPs for the synchronizing circuit to change clocks 6. Relaxation oscillator can be powered down (ROPD = 1) to conserve power 7. Change PLLCOD, 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. Crystal Oscillator should be powered up (CLK_MODE = 0) 2. Set Ext-Sel for Crystal Oscillator 3. Wait for the Crystal Oscillator to stabilize (up to 10 ms) 4. The clock source should be changed to the Crystal Oscillator (PRECS = 1) 5. Wait 4 NOPs for the synchronizing circuit to change clocks 6. Relaxation oscillator can be powered down (ROPD = 1) to conserve power

Table 5-1. Clock Choices (Continued)

Clock Source	Clock Selected	Configuration Steps
Crystal	Postscaler	1. Crystal Oscillator should be powered up (CLK_MODE = 0) 2. Set Ext-Sel for Crystal Oscillator 3. Wait for the crystal oscillator to stabilize (up to 10 ms) 4. The clock source should be changed to the Crystal Oscillator (PRECS = 1) 5. Wait 4 NOPs for the synchronizing circuit to change clocks 6. Relaxation oscillator can be powered down (ROPD = 1) to conserve power 7. Change PLLCOD, 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. Clock source (CLKIN) should be enabled in the GPIO and SIM as necessary. (Chip dependent) 2. Set Ext-Sel for desired external clock pin 3. Set the CLK_MODE bit in OSCTL register to 1 4. Select CLKIN as the source clock (PRECS = 1) 5. The clock source should be changed to the Crystal Oscillator (PRECS = 1) 6. Wait 4 NOPs for the synchronizing circuit to change clocks 7. Relaxation oscillator can be powered down (ROPD = 1) to conserve power 8. Change PLLCID, if desired 9. 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. Clock source (CLKIN) should be enabled in the GPIO and SIM as necessary. (Chip dependent) 2. Set Ext-Sel for the desired external clock pin 3. Set the CLK_MODE bit in OSCTL register to 1 4. Select CLKIN as the source clock (PRECS = 1) 5. The clock source should be changed to the Crystal Oscillator (PRECS = 1) 6. Wait 4 NOPs for the synchronizing circuit to change clocks 7. Relaxation oscillator can be powered down (ROPD = 1) to conserve power 8. Change PLLCOD, if desired 9. Enable the PLL (PLLPD = 0) 10. Wait for PLL lock (LCK1=1 and LCK0 = 1) 11. Change ZSRC to select the postscaler clock (ZSRC = 0) 12. At this point the EXTAL pin can be used as a GPIO by deasserting the appropriate PE bit in the GPIO_X_PER

5.5.1 Relaxation Oscillator

5.5.1.1 Trimming Frequency on the Internal Relaxation Oscillator

The internal relaxation oscillator frequency varies as much as ± 20% due to process, temperature, and voltage dependencies. These dependencies are in the voltage and current references, the offset of the comparators, and the internal capacitor. The voltage and temperature dependencies are designed to be a maximum of approximately two percent error. The process dependencies account for the rest.

Fortunately, for an individual part, the process dependencies are constant. An individual part can operate at approximately 2% variance from its unadjusted operating point over the entire specification range of the application. If the unadjusted operating point can be changed, the entire variance can be limited to 2%.

The method of changing the unadjusted operating point is by changing the size of the capacitor. This capacitor value is controlled by the trim factor (TRIM) in the oscillator control (OCTRL) register. The default value for TRIM is $200. Each unit added or removed adjusts the output period by about 0.078% of the unadjusted period. Adding to TRIM increases the clock period, thereby decreasing the frequency. With TRIM containing 10-bits, the clock period of the relaxation oscillator clock can be changed to ± 40% of its unadjusted value, enough to cancel the process variability previously mentioned.

The best way to trim the internal clock is to use the timer to measure the width of an input pulse on an input capture pin. This pulse must be supplied by the application and should be as long or wide as possible. Considering the prescale value of the timer and the theoretical (zero error) frequency of the bus, the error can be calculated. This error, expressed as a percentage, can be divided by resolution of the trim, that is 0.078% and the resultant factor added or subtracted from TRIM. This process should be repeated to eliminate any residual error.

5.5.2 External Reference

If higher clock precision is required the chip can be operated from an external clock source.

5.5.3 Crystal Oscillator

The crystal oscillator is designed to operate with either an external crystal or an external ceramic resonator. The ceramic resonator requires a larger current source from the amplifier and this is the default. When a crystal is used, the lower-power setting can be used when the crystal's equivalent series resistance (ESR) is less than 40Ω. Please see the COHL bit in the OSCTL register.

5.5.4 Switching Clock Sources

To robustly switch between the internal relaxation oscillator clock, external oscillator clock, and CLKIN, the change over 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 continues to operate off the original clock for between one and two cycles as the select input is transitioned through one side of the synchronizer. Next, the output is held low for between one and two cycles of the new clock as the select input transitions through the other side. The output begins switching at the new clock's frequency. This transition guarantees no glitches are 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 oscillator 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 it is enabled via GPIO and is powered up (CLK_MODE = 0).
- If switching to the relaxation oscillator, make sure it is powered up; that is, the ROPD bit is clear.
- Wait for a few cycles in order for the clock to become active.
- Switch clocks
- Execute four NOPs instructions
- Disable previous clock source (that is, power down relaxation oscillator if crystal is selected).

The key point to remember in this flow is the clock source should not be switched unless the desired clock is on and stable.

When a new core clock is selected, the clock generation module synchronizes the request and selects the new clock. The status (STAT) register shows the status of the clock switching process. Since the synchronizing circuit changes modes in order to avoid any glitches, the ZSRCS bits in the STAT register shows overlapping modes as an intermediate step.

5.5.5 Phase Locked Loop (PLL)

5.5.5.1 PLL Recommended Range of Operation

The voltage controlled oscillator (VCO) within the PLL has a characterized operating range extending from 60 MHz to 68 MHz. The output of the PLL F_{OUT} is fed to the input of the postscaler.

5.5.5.2 PLL Lock Time Specification

In many applications, the lock time of the PLL is the most critical PLL design parameters. Proper design and use of the PLL ensures the highest stability and lowest lock time.

5.5.5.3 Lock Time Definition

Typical control systems refer to the lock time as the reaction time within specified tolerances of the system to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percent of the step input or when the output settles to the desired value plus or minus a percent of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input.

When the PLL is coming from a powered down state (PLL_PDN_H) to a powered up condition (PLL_PDN_L) the maximum lock time with a divide-by count of 16 or less, is 10 ms. Other systems refer to lock time as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the lock time varies according to the original error in the output. Minor errors may be shorter or longer in many cases.

5.5.5.4 Parametric Influences on Reaction Time

Lock time is designed to be as short as possible while still providing the highest possible stability. The reaction time is not constant, however. Many factors directly and indirectly affect the lock time.

The most critical parameter affecting the reaction time of the PLL is the reference frequency through the MSTR_OSC, illustrated in Figure 5-1. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, it is desirable for the corrections to be small and frequent. Therefore, a higher reference frequency provides optimal performance; 8 MHz is preferred.

5.5.6 PLL Frequency Lock Detector Block

This digital block monitors the VCO output clock and sets the LCK bits in the STAT register based on its frequency accuracy. The lock detector is enabled with the LCKON and PLLPD bits, both on the CTRL register. 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 24 called feedback, and the PLL input clock**.** The period of the pulses being compared cover one whole period of each clock.

Feedback and reference 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 are the same number of reference clocks as feedback clocks, the LCK1 bit is also set. The LCK*n* bits stay set until:

- Clocks fail to match
- On reset caused by LCKON, PLL_PDN
- 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.

5.5.7 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 (LORTP+1) × 10/12 reference clocks. Figure 5-7 illustrates the general operation of the LOR detector, in turn relying on the ability of the PLL to continue running for a time after its reference clock is disturbed. This provides time for detection of the problem and an orderly system shutdown.

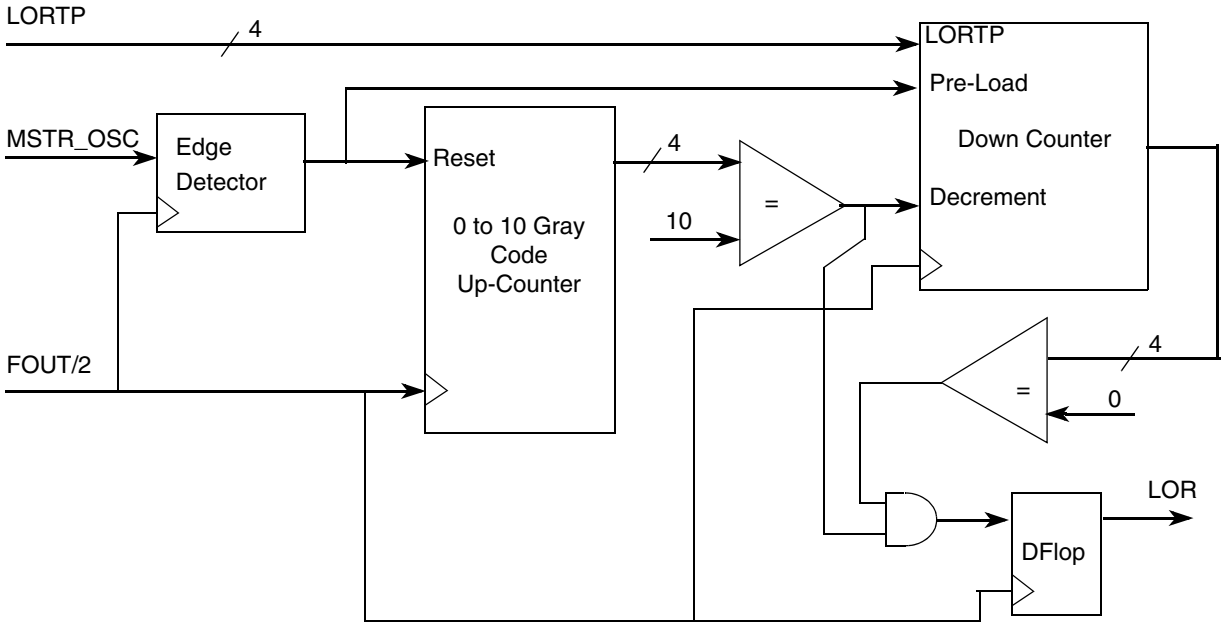


Figure 5-7. Simplified Block Diagram: Loss Of Reference Clock Detector

5.6 Register Description

The address of a register is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level.

Table 5-2. OCCS Memory Map

Device	Peripheral	Base Address
56F80xx	OCCS	$00F130

Table 5-3 lists the OCCS registers in ascending address order, including their acronyms and address offset of each register.

Table 5-3. OCCS Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL	Control register	Read/Write	Section 5.6.1
Base + $1	DIVBY	Divide-by register	Read/Write	Section 5.6.2
Base + $2	STAT	Status register	Read/Write	Section 5.6.3
		Reserved		
Base + $5	OCTRL	Oscillator control register	Read/Write	Section 5.6.4
Base + $6	CLKCHK	External clock check register	Read/Write	Section 5.6.5
Base + $7	PROT	Protection register	Read/Write	Section 5.6.6

There are six registers in the OCCS peripheral summarized in Figure 5-8. Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL	R	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	0	0	PLLPD	0	PRECS	ZSRC	
		W																
$1	DIVBY	R	LORTP				0	PLLCOD			0	0	0	0	0	0	0	0
		W																
$2	STAT	R	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLLPDN	0	0	ZSRCS	
		W																
$5	OCTRL	R	ROPD	ROSB	COHL	CLK_MODE	EXT_SEL	TRIM										
		W																
$6	CLKCHK	R	CHK_EN	REF_CNT							TARGET_CNT							
		W																
$7	PROT	R	0	0	0	0	0	0	0	0	0	0	FREQEP		OSCEP		PLLEP	
		W																

R 0 Read as 0
W Reserved

Figure 5-8. OCCS Register Map Summary

5.6.1 Control (CTRL) Register

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PLLIE1		PLLIC0		LOCIE	0	0	0	LCKON	0	0	PLLPD	0	PRECS	ZSRC	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 5-9. Control (CTRL) Register

5.6.1.1 PLL Interrupt Enable 1 (PLLIE1)—Bits 15–14

An optional interrupt can be generated when the PLL lock status bit (LCK1) in the PLL status register (PLLSR) 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

5.6.1.2 PLL Interrupt Enable 0 (PLLIE0)—Bits 13–12

An optional interrupt can be generated if the PLL lock status bit (LCK0) in the PLL status register (PLLSR) 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

5.6.1.3 Loss of Reference Clock Interrupt Enable (LOCIE)—Bit 11

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

5.6.1.4 Reserved—Bits 10–8

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.1.5 Lock Detector On (LCKON)—Bit 7

- 0 = Lock detector disabled
- 1 = Lock detector enabled

5.6.1.6 Reserved—Bits 6–5

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.1.7 PLL Power Down (PLLPD)—Bit 4

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 = 0 1b in order to prevent loss of reference clock to the core.

- 0 = PLL enabled
- 1 = PLL powered down

5.6.1.8 Reserved—Bit 3

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.1.9 Prescaler Clock Select (PRECS)—Bit 2

This bit is used to select between the external clock source or the internal relaxation oscillator.

- 0 = Relaxation oscillator selected (reset value)
- 1 = External reference selected (either crystal oscillator or external clock source pins)

NOTE:

This bit should not be set unless the external reference is enabled in the GPIO/SIM/KTR.

5.6.1.10 ZCLOCK Source (ZSRC)—Bits 1–0

The CLOCK source determines the SYS_CLK_×2 source to the SIM module, which generates divided down versions of this signal for use by memories and the IPBus. The ZSRC bit in the STAT register is automatically set to 01b during STOP_MODE, or when PLLPD is set in order to prevent loss of reference clock to the core. For the 56F8000, the ZSRC bit may have the following values. Please refer to Section 5.6.3.9.

- 00 = Reserved
- 01 = MSTR_OSC
- 10 = Postscaler output
- 11 = Reserved

5.6.2 Divide-By (DIVBY) Register

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LORTP				0	PLLCOD			0	0	0	0	0	0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-10. Divide-By (DIVBY) Register

5.6.2.1 Loss of Reference Clock Trip Point (LORTP)—Bits 15–12

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) × 10) ÷ (PLL Multiplier ÷ 2) reference clocks. The PLL Multiplier is fixed at 24.

Please see Section 5.5.7 for more information on how the loss of reference clock interrupt is generated.

5.6.2.2 Reserved—Bit 11

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.2.3 PLL Clock Out Divide or Postscaler (PLLCOD)—Bits 10–8

The PLL output clock can be divided down by a postscaler, illustrated in Figure 5-1. The output of the postscaler is a selectable clock source for the core as determined by the ZSRC bit in the CTRL register.

- 000 = divide by 1
- 001 = divide by 2
- 010 = divide by 4
- 011 = divide by 8
- 100 = divide by 16
- 101 = divide by 32
- 11x = divide by 32

5.6.2.4 Reserved—Bit 7–0

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.3 Status (STAT) Register

Base + $2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLL PDN	0	0	ZSRCS	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 5-11. Status (STAT) Register

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 register.

5.6.3.1 Loss of Lock Interrupt 1 (LOLI1)—Bit 15

LOLI1 shows the status of the lock detector state from LCK1 circuit. This bit is cleared by writing one to the LOLI1 bit.

- 0 = PLL locked
- 1 = PLL not locked

This bit is not set by the hardware if the corresponding CTRL register PLLIE1 bit is cleared, that is set to zero.

5.6.3.2 Loss of Lock Interrupt 0 (LOLI0)—Bit 14

LOLI0 shows the status of the lock detector state from LCK0 circuit. This bit is cleared by writing one to the LOLI0 bit.

- 0 = PLL locked
- 1 = PLL not locked

This bit is not set by the hardware if the corresponding CTRL register PLLIE0 bit is cleared, that is set to zero.

5.6.3.3 Loss of Reference Clock Interrupt (LOCI)—Bit 13

LOCI shows the status of the reference clock detection circuit. This bit is cleared by writing one to the LOCI bit.

- 0 = Oscillator clock normal
- 1 = Loss of oscillator clock detected

5.6.3.4 Reserved—Bits 12–7

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.3.5 Loss of Lock 1 (LCK1)—Bit 6

- 0 = PLL is unlocked
- 1 = PLL is locked (fine)

5.6.3.6 Loss of Lock 0 (LCK0)—Bit 5

- 0 = PLL is unlocked
- 1 = PLL is locked (coarse)

5.6.3.7 PLL Power Down (PLLPDN)—Bit 4

PLL power down status is delayed by four IPBus clocks from the PLLPD bit in the CTRL register.

- 0 = PLL not powered down
- 1 = PLL powered down

5.6.3.8 Reserved—Bits 3–2

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.3.9 Clock Source Status (ZSRCS)—Bits 1–0

ZSRCS bit indicates the current SYS_CLK_×2 clock source. Since the synchronizing circuit switches the system clock source, the ZSRCS bit 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

5.6.4 Oscillator Control (OCTRL) Register

This register controls aspects of both the internal relaxation oscillator and the crystal/resonator oscillator, illustrated in Figure 5-2.

Base + $5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ROPD	ROSB	COHL	CLK_ MODE	EXT_SEL		TRIM									
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1

Figure 5-12. Oscillator Control (OCTRL)

5.6.4.1 Relaxation Oscillator Power Down (ROPD)—Bit 15

This bit powers down the relaxation oscillator. The relaxation oscillator may powered down 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 was changed to the external source by setting the PRECS bit in CTRL register.

- 0 = Relaxation oscillator enabled
- 1 = Relaxation oscillator powered down

5.6.4.2 Relaxation Oscillator Standby (ROSB)—Bit 14

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 8 MHz.
- 1 = Standby mode—The relaxation oscillator output frequency is reduced to 400 kHz (± 50%). The PLL should be disabled in this mode and MSTR_OSC should be selected as the output clock.

5.6.4.3 Crystal Oscillator High/Low Power Level (COHL)—Bit 13

This bit controls the power usage of the crystal oscillator. It is reset to the high power state, allowing either a crystal or resonator to be used.

- 0 = High power mode
- 1 = Low power mode. This mode can be used with a crystal if the crystal's ESR is less than 40Ω

5.6.4.4 Crystal Oscillator Clock Mode (CLK_MODE)—Bit 12

This bit controls the crystal/resonator clock selection. When direct clock mode is selected, this bit also turns off the crystal oscillator for power savings.

- 0 = Crystal oscillator enabled
- 1 = Direct clock mode—Setting this bit shuts down the crystal oscillator and allows an external clock source on the XTAL pin to drive the clock input to the chip directly.

NOTE:

If the crystal oscillator is turned off and 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.

5.6.4.5 External Clock In Select (EXT_SEL)—Bits 11–10

These bits selects the source of the external clock input.

- 00 = Select primary external clock input (GPIO B6)
- 01 = Select alternate external clock input (GPIO B5)
- 10 = Select crystal oscillator (XTAL)
- 11 = Select crystal oscillator

To avoid glitches on the system clock, the PRECS bit in the CTRL register should be zero before changing the value of EXT_SEL.

5.6.4.6 Internal Relaxation Oscillator Trim (TRIM)—Bits 9–0

These bits change the size of the internal capacitor used by the internal relaxation oscillator. By testing the frequency of the internal clock and increasing or decreasing 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 $200, centering the range of possible adjustment. (A factory calibrated value is available in the FM_IFROPT_1 register.)

5.6.5 External Clock Check (CLKCHK) Register

This register is used for 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 as illustrated in Figure 5-2. The PRECS bit in the CTRL register must be zero to use this function.

Base + $6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CHK_EN	REF_CNT							TARGET_CNT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-13. External Clock Check (CLKCHK) Register

5.6.5.1 Check Enable (CHK_EN)—Bit 15

This bit starts and stops the clock checking function. Allow enough time after the CLK_CHK 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 its current state. Reading a *low* after a check has been started indicates the check operation is complete and the final values are valid in the REF_CNT and TARGET_CNT registers.
- 1 = Writing one clears the REF_CNT and TARGET_CNT bit fields and begins the clock checking function. The CHK_EN bit remains high while the operation is in progress.

5.6.5.2 Reference Count (REF_CNT)—Bits 14–8

This bit field provides the number of counted ROSC clock cycles. At the end of a clock check operation this count reads as all lows. (The reference counter has counted through its whole range and rolled over to zero.)

NOTE:

This counter value is not synchronized to the bus clock and any value read while CHK_EN is high should not be considered accurate.

5.6.5.3 Target Count (TARGET_CNT)—Bits 7–0

This bit field provides the number of counted external clock cycles.

NOTE:

This counter value is not synchronized to the bus clock and any value read while CHK_EN is high should not be considered accurate.

5.6.6 Protection (PROT) Register

This register provides features for protection of safety-critical register fields during runaway code failures. By choosing an appropriate subset of the protection register it is possible to define the trade-off between power management and protection of the OCCS operating configuration.

Flexibility is provided by allowing the write protection control values themselves to be optionally locked, or write protected. To this end, protection controls in this register have two bit values. The right bit determines the setting of the control while the left bit determines if the value is locked. When a protection control is set to a locked value, it can only be altered by a chip reset thereby restoring its default non-locked value. While a protection control remains set to non-locked values, it can be re-written to any new value.

Base + $7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	FRQEP		OSCEP		PLLEP	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-14. Protection (PROT) Register

5.6.6.1 Reserved—Bits 15–6

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

5.6.6.2 Frequency Enable Protection (FRQEP)—Bits 5–4

Enables write protection of the PLLCOD and ZSRCS bits in the DIVBY and STAT registers respectively.

- 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

5.6.6.3 Oscillator Enable Protection (OSCEP)—Bits 3–2

Enables write protection of the OSCTL and PRECS bits.

- 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

5.6.6.4 PLL Enable Protection (PLLEP)—Bits 1–0

Enables write protection of the PLLPDN, LOCIE and LORTP bits. By write protecting these bits (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

5.7 Interrupts

The interrupts listed in Table 5-4 may be OR’ed into a single processor core interrupt for some chip integrations. Inspect the interrupt table in the chip specification to determine what permutation of these interrupts are actually used.

Table 5-4. Interrupt Summary

Interrupt	Source	Description	Reference
LOLI1	PLLSR	Lock 1 interrupt	Section 5.6.1
LOLI0	PLLSR	Lock 2 interrupt	Section 5.6.1
LOCI	PLLSR	Loss of reference clock interrupt	Section 5.6.1

5.8 Resets

Table 5-5 illustrates the various reset sources present in the 56F80xx implementation of the OCCS module. Any configuration yielding substantially the same operation is acceptable.

Table 5-5. V1 Implementation Reset Summary

Reset	Source	Characteristics
$\overline{\text{PERIP\_RST}}$	SIM	The chip architecture incorporates a three-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 ($\overline{\text{PERIP\_RST}}$). That signal is used by this module to reset its state. $\overline{\text{PERIP\_RST}}$ is stretched an additional 32-system clocks to generate $\overline{\text{CORE\_RST}}$, which is used to reset the core.

Table 5-6. Document Revision History for Chapter 5

Version History	Description of Change
Rev. 3	• Added revision history table. • Deleted the step involving a change to low- or high-power mode from Table 5-1. • In Section 5.5, replaced the sentence "*If a crystal is used on the board the power level of this oscillator should be lowered to prevent over driving the crystal and to reduce overall power consumption*" with "*When a crystal is used, the lower-power setting can be used when the crystal's equivalent series resistance (ESR) is less than 40Ω*". • Revised OCTRL[COHL] bit descriptions.

Chapter 6
Flash Memory (FM)

6.1 Introduction

The Flash memory (FM) module is a non-volatile memory module, it serves as electrically erasable and programmable memory. It is ideal for program storage for single-chip applications, and supports field reprogramming without requiring external programming voltage sources.

Program and erase operations are performed by a command driven interface from the DSC core controller using a state machine internal to the module.

6.2 Features

- 32 MHz single cycle operation for all program Flash accesses
- Automated program and erase operation
- Interrupts on command completion, command buffer empty, and access error
- Fast page erase
- Single power supply program and erase
- Security feature
- Sector protection system
- The Flash memory (FM) supports byte and word/read operations by the host CPV

6.3 Block Diagram

The Flash memory (FM) illustrated in Figure 6-1, contains Flash array blocks, program bus, system bus interface control, Flash interface, and register blocks. The Flash block is an array of electrically erasable and programmable, non-volatile memory for use as program (P:) memory.

Program memory of 64 k to 32 k bytes in length is constructed from a 64 k byte block. Read access occurs within one cycle with no penalty.

A 64 k block is organized as 1024 rows of 64 bytes each. A 32 K block is organized as 512 rows of 64 bytes each.

An erase page contains eight rows of 64 bytes for a total of 512 bytes. The erase operation also supports mass erase of an entire block. An erased bit reads as one and a programmed bit reads as zero.

All registers are memory mapped for easy access to control program and erase operations.

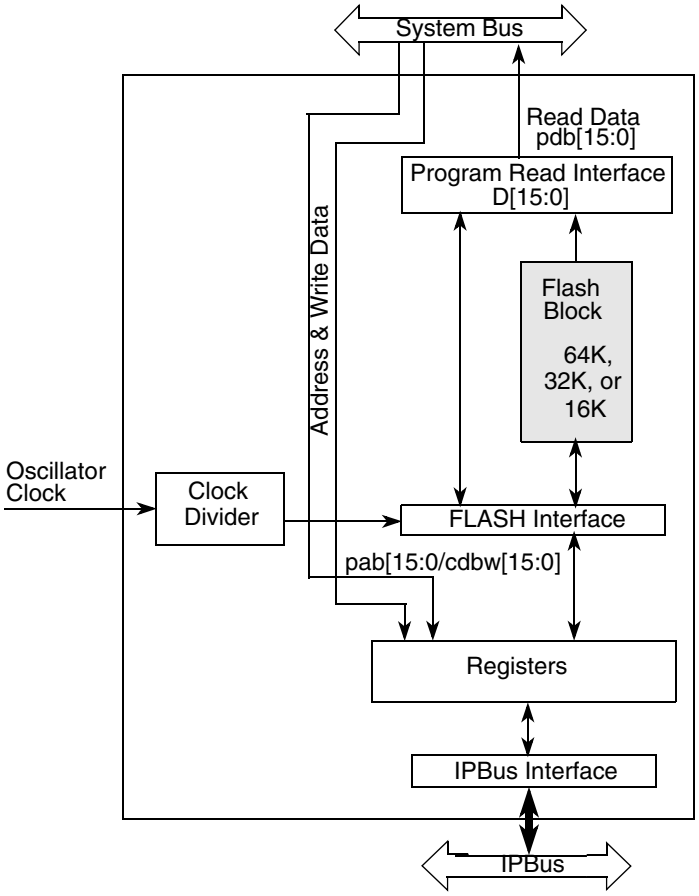


Figure 6-1. Flash Block Diagram

6.4 Memory Map

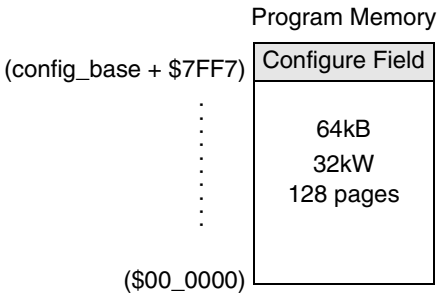


Figure 6-2. Flash Array Memory Map

The configuration field contains data to facilitate both security and protection features. Protection refers to undesired core access while security refers to undesired external access. The configuration field is composed of the top nine words of program Flash memory, containing information determining the module's protection and access restriction scheme out of reset. The protection word in the configuration field is transferred into the protection

(PROT) register at reset. The security words in the configuration field are transferred into the security (SECLO, SECHI) registers upon reset. A description of each value in the configuration field is provided in Table 6-1. For further details refer to the protection and security register descriptions.

Table 6-1. Flash Memory Configuration Field

Address 64k	Description	Word Name
$7FFF	Back door comparison key 3	BACK_KEY_3_VALUE
$7FFE	Back door comparison key 2	BACK_KEY_2_VALUE
$7FFD	Back door comparison key 1	BACK_KEY_1_VALUE
$7FFC	Back door comparison key 0	BACK_KEY_0_VALUE
$7FFB	Not used	Not Used
$7FFA	Protection word (see Section 6.7.4)	PROT_VALUE
$7FF9	Not used	Not Used
$7FF8	Security word upper (see Section 6.7.3)	SECH_VALUE
$7FF7	Security word lower (see Section 6.7.3)	SECL_VALUE

Flash memory also contains a set of control and status registers located data memory space at the FM register base register address. A summary of these registers is provided in Table 6-2.

Table 6-2. Flash Memory Register Address Map

Offset From Register Base Address[1]	Bits 15-8	Bits 7-0
Base + $0	RESERVED	CLKD[2]
Base + $1	CNFG	
Base + $2	RESERVED	
Base + $3	SECHI	
Base + $4	SECLO	
Base + $5 – $F	RESERVED	
Base + $10	PROT	
Base + $11 – $12	RESERVED	
Base + $13	RESERVED	USTAT
Base + $14	RESERVED	CMD
Base + $15 – $17	RESERVED	
Base + $18	DATA	
Base + $19 – $1A	RESERVED	
Base + $1B	OPT1	
Base + $1C	RESERVED	
Base + $1D	TSTSIG	
Base + $1E – $3C	RESERVED	

6.5 Operating Modes

This section details the modes of operation of Flash memory. Table 6-3 provides all of the available user commands.

6.5.1 Read Operation

The FM module provides transparent read access of all memory locations in its Flash array. Each read is achieved without wait states, providing memory access without delays.

6.5.2 Write Operation

There are two types of write access supported by the FM module. 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 FM module's command (CMD) register.

Erased Flash are all ones, so any programming attempt writes only the required zeros to create the desired bit pattern. It is up to the user to verify the erased state of the destination in Flash before programming it.

If the KEYACC bit is set in the configuration (CNFG) register, all writes to the Flash are interpreted as attempts to unlock program Flash by matching the 64-bit key string found in the four word memory. For more information about this write mode, please see Section 6.6.1.

6.5.3 Program and Erase Operation

Both read and write operations are used for the program and erase algorithms described in this section. These algorithms are controlled by the module's state machine.

The Flash command (CMD) register operates as a buffer and a register (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 feature increases the rate at which the FM state machine receives commands, reduces command processing time, and expedites program/erase cycles. Buffer empty as well as command completion are signaled by flags in the Flash status (STAT) register. Interrupts are generated when enabled.

6.5.4 User Mode Commands

Table 6-3 summarizes the valid Flash user commands.

Table 6-3. Flash User Mode Commands

FM Command	Meaning	Description	Array Write Address/Data Usage
$05	Erase Verify	Verify the Flash array is erased. If the array is verified to be blank, the BLANK bit will set in the USTAT register, Figure 6-12, upon command completion.	The address must be any address within the array. Data is ignored.
$06	Calculate Data Signature	Calculate a signature over a user-specified range of words in the Flash array. The resulting signature is returned in the DATA register, Figure 6-17, at the completion of the command. The new signature can be compared with a previously calculated signature (performed by user)[1] to verify program Flash integrity.	The array write address specifies the starting address and data specifies the number of words to calculate the signature over.
$20	Program	Program 16-bit word. A program is only possible when the protection bit for that sector is not set.	The address and data specifies the address and value to program.

Table 6-3. Flash User Mode Commands (Continued)

FM Command	Meaning	Description	Array Write Address/Data Usage
$40	Page Erase	Erase a specific page (sector) of the Flash array. A page erase is only possible 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.
$41	Mass Erase	Erase all Flash memory. A mass erase is only possible when no PROTECT bits are set.	The address must be any address within the array. Data is ignored.
$66	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, Figure 6-14, at the completion of the command. The entire IFR row one is compressed except the last word containing the expected compression result. The IFR Block Signature calculated at the factory is available, for comparison to the new value, in the TSTSIG register. If the value in DATA differs with that found in the TSTSIG 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.

1. PERL script exists which can be used to calculate the signature. See FAQ 25630 on *freescale.com*.

6.5.5 Command Sequence Operation

This section describes how to perform a command sequence. This algorithm involves the use of an array write operation described above, as well as read/writes of FM registers. These algorithms are controlled by a state machine whose time base (FCLK) is derived from the FM system clock using a programmable prescaler and divider controlled by fields in the CLKDIV register. Please see Section 6.5.5.1.

Parameters required for timed events in program and erase algorithms are loaded from the Flash information row (IFR). Buffer empty as well as command complete are signalled by flags in the status register. Interrupts are generated if enabled.

While the algorithm can be used to process commands to completion one at a time, the command register operates as a buffer and a register (two-stage FIFO), so 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 calculate signature commands.

6.5.5.1 Writing the CLKDIV Register

Prior to the execution of any Flash module command, the Flash module clock divider (CLKDIV) register must be initialized. The values of this register determine the speed of the internal Flash clock (FCLK). FCLK must be in the range of 150 kHz ≤ FCLK ≤ 200 kHz for proper operation of the Flash module. (Running FCLK too slowly wears out the module, while running it too fast under programs Flash leads to bit errors.)

Table 6-4 provides CLKDIV register values for various operating calculations. The table shows values when a external clock is used as well as cases where the on-chip relaxation oscillator (ROSC) is used. The OCCS parameters required for each clock setup are shown in the columns for PRECS, PLLCOD, and ZSRC. The CTRL register in OCCS contains PRECS and ZSRC. PLLCOD is located in the DIVBY register in the OCCS peripheral.

The parameter t_{BUS} represents the largest period of the device's system clock, measured in microseconds. If t_{BUS} > 1 µsec, it is not possible to execute Flash module commands. Reading Flash is still possible, but writing (programming new values) or erasing Flash is not possible until $t_{BUS} \leq 1$ µsec.

The DIVLD bit in the CLKDIV register shows if the register has been initialized since the last reset. Applications should check this bit before initiating any Flash module commands.

Table 6-4. Flash Module Clock Parameters for Various Operating Conditions

Input Clock *MSTR_OSC*	Freq (MHz)	Prescaler Clock *PRECS*	PLL Divider *PLLCOD*	Output Clock Source	ZSRC	Sys_Clk (MHz)	t_{BUS} (µsec)	t_{BUS} <=1µs?	PRDIV8	DIV	FCLK
External	1	1	n/a	MSTR_OSC	01	0.5	2	NO	0	5	166.7
External	2	1	n/a	MSTR_OSC	01	1	1	YES	0	10	181.8
External	4	1	n/a	MSTR_OSC	01	2	0.5	YES	0	20	190.5
ROSC	0.4	0	n/a	MSTR_OSC	01	0.2	5	NO	0	2	133.3
ROSC	8	0	n/a	MSTR_OSC	01	4	0.25	YES	0	40	195.1
ROSC	8	0	32	Postscaler	10	1	1	YES	0	41	190.5
ROSC	8	0	16	Postscaler	10	2	0.5	YES	0	40	195.1
ROSC	8	0	8	Postscaler	10	4	0.25	YES	0	40	195.1
ROSC	8	0	4	Postscaler	10	8	0.125	YES	0	40	195.1
ROSC	8	0	2	Postscaler	10	16	0.0625	YES	0	40	195.1
ROSC	8	0	1	Postscaler	10	32	0.03125	YES	0	40	195.1
External	16	1	n/a	MSTR_OSC	01	8	0.125	YES	1	10	181.8
External	32	1	n/a	MSTR_OSC	01	16	0.0625	YES	1	20	190.5
External	64	1	n/a	MSTR_OSC	01	32	0.03125	YES	1	40	195.1

A spreadsheet tool is available in the Freescale FAQs for this family. This spreadsheet, illustrated in Figure 6-3, allows the recalculation of PRDIV8 and DIV values for operating scenarios not included in Figure 6-4. Simply search for FAQ number 25464 on the Freescale website.

Frequency of MSTR_OSC Clock (MHz)	F_{osc}	8	<- Enter
Frequency of System Clock (MHz)	F_{sys}	32	<- Enter
System clock period (µSeconds)	t_{BUS}	0.03125	
Is $t_{BUS} \leq$ = 1 µSecond?	Continue?	**Continue**	
Enable Prescaler Divide By 8	PRDIV8	0	<- Use
Output Stage Frequency After Divider (MHz)	Fadj	8	
Provisional Divider Value	Prov DIV	39	
Provisional Flash Module Clock (kHz)	Prov Fclk	200	
Final Divider Value	DIV	40	<- Use
Final Flash Module Clock (kHz)	F_{clk}	195.122	

Figure 6-3. FM Clock Parameter Tool

6.5.5.2 Command Sequence Protocol

A command state machine supervises command processing. To prepare for a command execution, the CBEIF flag in the USTAT register should be tested, ensuring the address, data, and command buffers are empty. If the CBEIF flag is set, the command sequence can be started.

You must follow the command write sequence below to successfully program Flash.

NOTE:

The command write sequence must execute from RAM.

Intermediate writes to the FM are not permitted between the three steps.

Command write sequences (See Table 6-4):

1. Write the desired value to the address in Flash memory you wish to program.
2. Write the numeric code for the command to the command buffer (CMD) register. Flash module commands are described in Table 6-3.
3. To launch the command, clear the CBEIF flag by writing one. After the CBEIF flag is cleared, the CCIF flag in the STAT register is cleared by hardware, indicating the command was successfully launched. The CBEIF flag is set again indicating the address, data and command buffers are ready for a new Command Write sequence to begin.

The completion of the command operation is indicated by the CCIF flag setting.

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 USTAT register. An erroneous command write sequence will abort, setting the appropriate flag. If set, the ACCERR or PVIOL flags must be cleared before starting another command write sequence.

NOTE:

By writing zero 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 CMD register and before the command is launched. The ACCERR flag is set on aborted commands and must be cleared before a new command is launched.

A flow chart of the entire command sequence is provided in Figure 6-4.

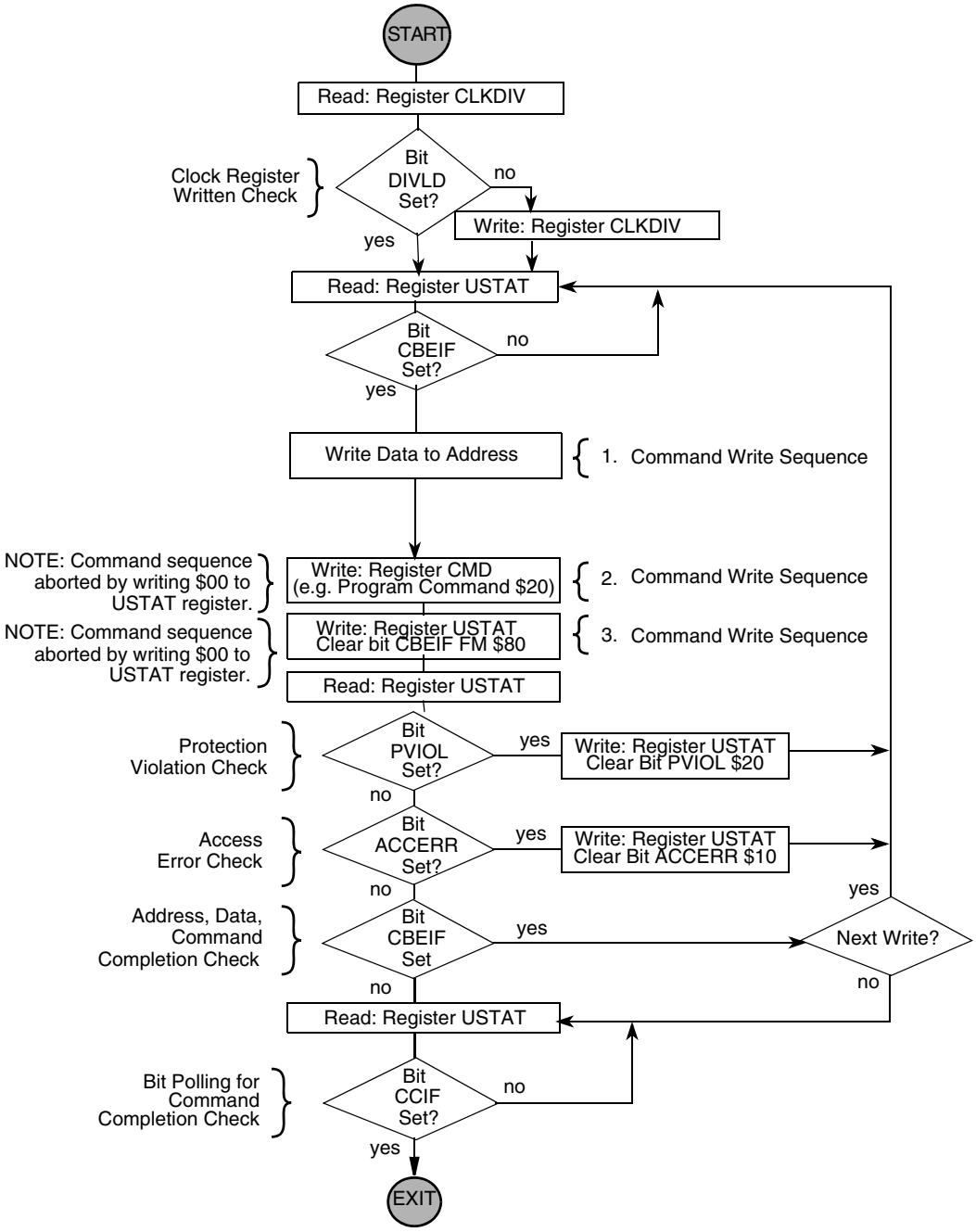


Figure 6-4. Command Sequence Flow Chart[1]

6.5.5.3 Flash User Mode Illegal Operations

The ACCERR flag is set during the command write sequence if any of the following illegal operations are performed. These operations cause the command state machine to immediately abort. Writes to the FM address space refer to writes through the 16-bit controller core buses, not the IP register bus. The following are illegal operations with undetermined results:

1. Code must reside in RAM

- Writing to the Flash address space before initializing CLKD
- Writing to the Flash address space while CBEIF in USTAT is not set
- Writing a second word to the Flash address space
- Writing an invalid user command to the CMD register
- Writing to any Flash register other than the CMD register after writing a word to the Flash address space
- Writing a second command to the CMD register before executing the previously written command
- Writing to any FM register other than the USTAT register (to clear CBEIF) after writing to the CMD register
- The part enters stop or wait modes and a program or erase command is in progress; the command is aborted
- Aborting a command sequence by writing zero to the CBEIF bit in USTAT register after a word write to the Flash address space, or after writing a command to the CMD register, and before the command is launched
- Writing to the array while a calculate signature command is running (either IFR or main array)

The protection violation (PVIOL) flag is set during the command/write sequence after the word write to the Flash address space if any of the following illegal operations are performed. Such operations cause the command sequence to immediately abort:

1. Writing a Flash address to program a protected area
2. Page erase of a protected area
3. Writing a mass erase command to the CMD register while any protection is enabled for that block. Please see Section 6.7.4.

If a Flash block is read during execution of an algorithm on that block (that is, CCIF is low), the read returns non valid data and the ACCERR bit in the USTAT register will not be set.

6.5.5.4 Affects of Wait/Stop Modes

If a command is active (CCIF = 0) when the controller enters wait or stop mode, the command is aborted, and the data being programmed or erased is lost. The high voltage circuitry to the Flash is switched off and a pending command (CBEIF = 0) will not be executed once the controller exits wait or stop mode. The CCIF and ACCERR flags in the USTAT register are set if a command is active when the 16-bit controller enters wait or stop mode.

WARNING:

As active commands are immediately aborted when the 16-bit controller enters wait mode, it is strongly recommended not to execute the wait instruction during program and erase execution.

6.6 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 SECHI register. 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. When 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. Executing 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 content of Flash memory.

6.6.1 Back Door Access

Flash security can be disabled at run-time by following these directions. The KEYEN bit in the SECHI register must be set to enable back door key access.

1. Set the KEYACC bit in the configuration (CNFG) register.
2. Write the correct four word (64-bit) back door comparison key to the FM configuration field. Please see Table 6-1. For example, the addresses for a 64 kB block would be: $7FFC – $7FFF. This operation must be composed of four word writes started with the smallest address. For example, write to address $7FFC, $7FFD, $7FFE and $7FFF in that order for a 64 kB block. 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. In the unprotected state the DSC core has full control of FM.

The value of the Flash security words ($7FF7 – $7FF8), is not changed by the back door method of unsecuring the device. After the next reset sequence, the device is secured again and the same back door key is in effect, unless the configuration field is changed by program or erase. Flash security, defined by the Flash security words in Table 6-7, must be changed directly by reprogramming the Flash security words in memory when the highest sector is unprotected.

The back door method of unsecuring the device has no effect on the program and erase protections defined in the protection (PROT) register.

6.6.2 JTAG Lockout Recovery

To unsecure a secured FM:

- Mass erase the FM via a sequence of JTAG commands, and then on the next reset, the part is unsecured.
- See the device data sheet section on security features.

6.6.3 Erase Verify Check

A secured module can be unsecured by verifying the Flash array is blank. If required, the mass erase command sequence can be executed on the Flash array. The erase verify command sequence must then be executed on the Flash array. The Flash module is unsecured if the erase verify command determines the entire Flash array is blank. After the next reset sequence, the security state of the Flash module is determined again by the lower Flash security word in the configuration field.

6.7 Register Description

Table 6-5. FM Memory Map

Device	Peripheral	Base Address
56F80xx	FM	$00F400

A register address is the sum of a base address and an address offset. The base address is defined at the device level and the address offset is defined at the module level.

Table 6-6 lists the Flash registers in ascending address order, including their acronyms and address offset of each register.

Table 6-6. Flash Registers

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CLKDIV	Clock divider register	Read/Write	Section 6.7.1
Base + $1	CNFG	Configuration register	Read/Write	Section 6.7.2
		Reserved		
Base + $3	SECHI	Security high half register	Read-Only	Section 6.7.3
Base + $4	SECLO	Security low half register	Read-Only	
		Reserved		
Base + $10	PROT	Protection register	Read/Write	Section 6.7.4
		Reserved		
		Reserved		
Base + $13	USTAT	User status register	Read/Write	Section 6.7.5
Base + $14	CMD	Command register	Read/Write	Section 6.7.6
		Reserved		
Base + $18	DATA	Data buffer register	Read-Only	Section 6.7.7
		Reserved		
Base + $1B	OPT1	Optional data 1 register	Read-Only	Section 6.7.8
		Reserved		
Base + $1D	TSTSIG	Test array signature register	Read-Only	Section 6.7.9

There are 12 registers, including two optional registers, in the Flash peripheral summarized in Figure 6-5. Each is detailed in the following sections.

Addr. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CLKDIV	R	0	0	0	0	0	0	0	0	DIVLD	PR DIV8	DIV					
		W																
$1	CNFG	R	0	0	0	0	0	LOCK	0	AEIE	CBEIE	CCIE	KEY ACC	0	0	0	0	0
		W																
$3	SECHI	R	KEY EN	SEC STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
$4	SECLO	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEC	
		W																
$10	PROT	R	PROTECT															
		W																
$13	USTAT	R	0	0	0	0	0	0	0	0	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
		W																
$14	CMD	R	0	0	0	0	0	0	0	0	0	COMMAND						
		W																
$18	DATA	R	DATA															
		W																
$1B	OPT1	R	OPT1															
		W																
$1D	TSTSIG	R	TST_AREA_SIG															
		W																

R 0 Read as 0
W Reserved

Figure 6-5. Flash Register Map Summary

6.7.1 Clock Divider Register (CLKDIV)

The Flash memory clock divider (CLKDIV) register controls the period of the FCLK used for timed events in program and erase algorithms within the Flash interface (FI). While the FI operates at system bus frequency, FCLK must operate within the 150-200 kHz range. FCLK is generated by dividing the oscillator clock (MSTR_OSC in OCCS) by a prescaler and a divider. Please refer to Section 6.5.5.1.

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DIVLD	PRDIV8	DIV					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-6. Clock Divider (CLKDIV) Register

6.7.1.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

6.7.1.2 Clock Divider Loaded (DIVLD)—Bit 7

This is a status-only bit. Writing has no effect.

- 0 = Register has not been written
- 1 = Register has received writing since the last reset

6.7.1.3 Enable Prescaler By 8 (PRDIV8)—Bit 6

This is a read and write bit.

- 0 = Prescaler divides oscillator clock by one
- 1 = Prescaler divides oscillator clock by eight

6.7.1.4 Clock Divider (DIV)—Bits 5–0

The clock divider register bits PRDIV8 and DIV must be set with appropriate values before programming or erasing the FM array. Because FCLK is re-timed into the system clock domain, the values of PRDIV8 and DIV are affected by the system bus frequency as well. Refer to the functional description of writing the CLKDIV register located in Section 6.5.5.1 for the detailed algorithm to determine the settings for DIV and PRDIV8.

NOTE:

Unlike the 56F8300 series, bits 6–0 are not write-protected once written. These bits can be modified even after their initialization.

6.7.2 Configuration Register (CNFG)

The FM Configuration (CNFG) register configures and controls the operation of the FM array.

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	LOCK	0	AEIE	CBEIE	CCIE	KEYACC	0	0	0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-7. Configuration (CNFG) Register

6.7.2.1 Reserved—Bits 15–11

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

6.7.2.2 Write Lock Control (LOCK)—Bit 10

This bit can always be read. Once set, this bit cannot be cleared except by reset. This bit provides additional security for the Flash array by disabling writes to the protection register.

- 0 = The PROT register can be modified by writing
- 1 = The PROT register is write-locked

6.7.2.3 Reserved—Bit 9

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

6.7.2.4 Access Error Interrupt Enable (AEIE)—Bit 8

This read/write bit enables an interrupt in case of an error accessing the Flash.

- 0 = ACCERR interrupts disabled
- 1 = An interrupt is requested whenever the ACCERR flag is set

6.7.2.5 Command Buffer Empty Interrupt Enable (CBEIE)—Bit 7

This read/write bit enables an interrupt in case of an empty command buffer in the Flash.

- 0 = Command buffer empty interrupts disabled
- 1 = An interrupt is requested whenever the CBEIF flag is set

6.7.2.6 Command Complete Interrupt Enable (CCIE)—Bit 6

This read/write bit enables an interrupt in case of all commands being completed in the Flash.

- 0 = Command complete interrupts disabled
- 1 = An interrupt is requested whenever the CCIF flag is set

6.7.2.7 Enable Security Key Writing (KEYACC)—Bit 5

This bit can be read, however, it can receive writing if the KEYEN bit in the SECHI register is set.

- 0 = Flash writes are interpreted as the start of a command sequence, that is program or erase
- 1 = Writes to Flash array are interpreted as keys to open the back door

6.7.2.8 Reserved—Bits 4–0

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

6.7.3 Security Registers (SECHI and SECLO)

The SECHI and SECLO registers store the Flash security word, defined in Table 6-1.

Base + $3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	KEYEN	SECSTAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Write																
Reset	F[1]	F[2]	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset state loaded from Flash array during reset.
2. Reset state determined by security state of module.

Figure 6-8. Security High (SECHI) Register

Base + $4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SECURITY	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Figure 6-9. Security Low (SECLO) Register

Bits 15 and 14 of the SECHI register, illustrated in Figure 6-8, and each of the 16 bits of SECLO register, illustrated in Figure 6-8, can be read; however, neither register can be modified by writing.

The SECLO register is initialized at reset using SECLO_VALUE from the Flash configuration field. Likewise, the SECHI register is initialized at reset using SECHI_VALUE from the Flash configuration field. The Flash configuration field in the Flash array is described in Table 6-1. Value *F* in the reset value shown in Figure 6-8 indicates bits copied directly from the corresponding Flash memory configuration field in the top nine words of FM.

The reset value of SECSTAT is determined by the value loaded into the SECURITY field at reset as described in the following four sections. Flash security can be enabled or disabled during read only run-time, discussed in Section 6.6, and the current state of Flash security determines the state of SECSTAT.

6.7.3.1 Enable Back Door Key to Security (KEYEN)—Bit 15

This is a read-only bit.

- 0 = Back door to Flash is disabled
- 1 = Back door to Flash is enabled

6.7.3.2 Security Status (SECSTAT)—Bit 14

This is a read-only bit.

- 0 = Flash Security is disabled
- 1 = Flash Security is enabled

6.7.3.3 Reserved—Bit 13–0

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

6.7.3.4 Security (SECURITY)—Bits 15–0

This is a read-only register. Value loaded into the security (SECURITY) bit field from the configuration field at reset in turn determines the state of Flash security at reset (SECSTAT). Table 6-7 outlines the single code enabling the security feature in the FM.

Table 6-7. Security States

Security	Description
$2	Flash secured[1] (SECSTAT set at reset)
All other combinations	Flash unsecured (SECSTAT cleared at reset)

Table 6-7. Security States

Security	Description

1. The $E70A value was selected because it represents an illegal instruction on the 56800E core, making it unlikely user compiled code accidentally programmed in the security configuration field location would unintentionally secure the Flash.

WARNING:

To perform product analysis when security is enabled, either security must be disabled by the back door key, or the array must be totally erased either by performing the lockout recovery sequence or by performing a mass erase followed by a read verify.

6.7.4 Protection Register (PROT)

This register defines which Flash pages or sectors are protected against program and erase.

Base + $10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PROTECT															
Write																
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Figure 6-10. Protection (PROT) Register

This register may always be read, but may be modified by writing only when LOCK = 0.

This register value is reset to the PROT_VALUE as defined in the Flash memory configuration field; however, this is only possible if the LOCK bit in CNFG is zero. To change the Flash protection loaded on reset, the sector [15] of program Flash must first be unprotected as just described above, then the protection word in the configuration field at addresses defined in Table 6-1 must be programmed with the desired value.

On part configurations for which the first half of the Flash array is not implemented, bits 0–7 of the PROT register

PROT Bit N = Protection Field for Sector N

PROT Bit 0 → Sector 0

PROT Bit 1 → Sector 1

PROT Bit 15 → Sector 15

are reserved and read as zero. Bits 8–15 continue to define protection status for the remaining eight sectors of the Flash array.

- PROTECT[M] = 0: Array sector M is not protected
- PROTECT[M] = 1: Array sector M is protected

The PROT register controls the protection of 16 sectors in a 64 K byte section of program memory. Figure 6-11 outlines the association between each bit in the PROT register and the corresponding Flash memory sector within the program Flash.

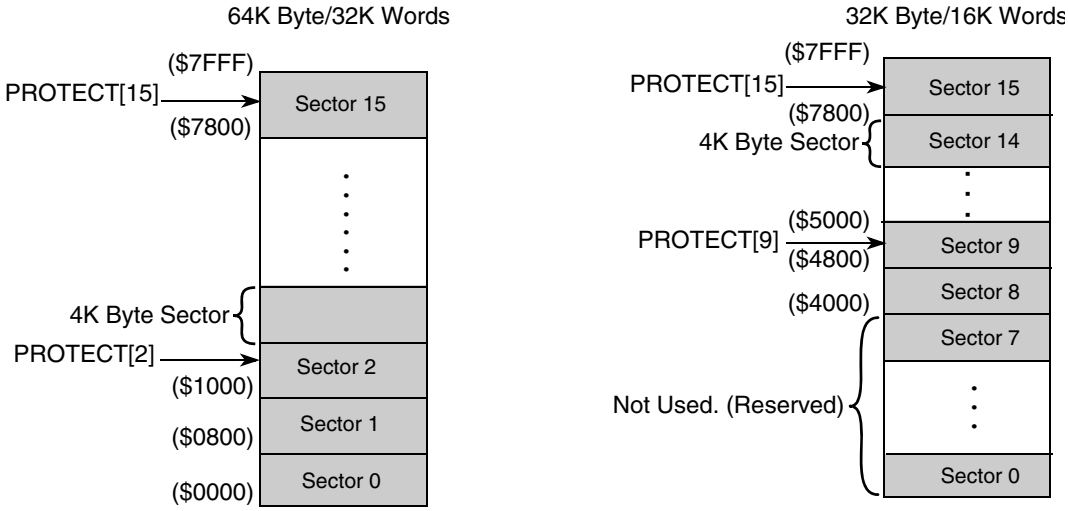


Figure 6-11. Protection Diagram

6.7.5 User Status Register (USTAT)

The USTAT register defines the Flash state machine command status and Flash array access, protection and blank verify status.

NOTE:

Only one bit should be cleared at a time in the USTAT register. This is due to the nature of the state machine clearing the register bits.

Base + $13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
Write																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 6-12. User Status (USTAT) Register

6.7.5.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It cannot be read or written.

6.7.5.2 Command Buffer Empty Interrupt Flag (CBEIF)—Bit 7

The CBEIF flag indicates the address, data, and command buffers are empty, allowing a new command sequence to be started. The CBEIF flag is cleared by writing one. Clearing the flag results in the CMD register being transferred to the internal state machine for launch of a command.

Writing zero has no effect on CBEIF, but it can be used to abort a command sequence. The CBEIF bit can generate an interrupt if the CBEIE bit in the CNFG register is set. While the CBEIF flag is clear the CMD register cannot be modified by writing.

- 0 = Command buffer is full
- 1 = Command buffer is ready to accept a new command

6.7.5.3 Command Complete Interrupt Flag (CCIF)—Bit 6

The CCIF flag indicates there are no other pending commands. The CCIF flag is set and cleared automatically upon start and completion of a command. Writing to CCIF has no effect. The CCIF bit can generate an interrupt if the CCIE bit in the CNFG register is set.

- 0 = Command in progress
- 1 = All commands are completed

6.7.5.4 Protection Violation (PVIOL)—Bit 5

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 one. Writing zero has no effect on PVIOL. While the PVIOL flag is set, it is not possible to launch another command.

- 0 = No failure
- 1 = A protection violation has occurred

6.7.5.5 Access Error (ACCERR)—Bit 4

The ACCERR flag indicates an illegal access to the FM array or registers caused by a bad program or erase sequence. The ACCERR flag is cleared by writing one. Writing zero to ACCERR bit has no effect. While the ACCERR flag is set, it is not possible to launch another command. The ACCERR relates to FM array writes from the 56F800E core buses and will not be set by writing directly to the data and address registers from the IPBuses. Please see Section 6.5.5.3 to set ACCERR flag details.

- 0 = No failure
- 1 = Access error has occurred

6.7.5.6 Flash Block Verified Erased (BLANK)—Bit 2

The BLANK flag indicates an erase verify command (RDARY1) checked the Flash block and found it to be blank. The BLANK flag is cleared by writing one. Writing zero has no effect.

- 0 = If an erase verify command is requested, and the CCIF flag is set, a zero in BLANK indicates the block is not erased.
- 1 = Flash block verifies as erased

6.7.6 Command Register (CMD)

This register defines the Flash Command to be executed.

Base + $14	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	COMMAND						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-13. Command (CMD) Register

6.7.6.1 Reserved—Bits 15–7

This bit field is reserved or not implemented. It is not available to be read or written.

6.7.6.2 Command (COMMAND)—Bits 6–0

Valid user mode commands are shown in Table 6-8. Writing a command in user mode other than those listed in Table 6-8 causes the ACCERR flag in the USTAT register to be set. Please see Table 6-3 for a description of the commands.

Table 6-8. Command User Mode Commands

Command	Description
$05	Erase verify (all ones)
$06	Calculate data signature
$20	Word program
$40	Page erase
$41	Mass erase
$66	Calculate IFR block signature

6.7.7 Data Register (DATA)

Read access is permitted to the DATA register.

Base + $18	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DATA															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-14. Data (DATA) Register

6.7.7.1 Data Buffer (DATA)—Bits 15–0

The results of the calculate data signature and calculate IFR block signature commands are available in this register after execution of these commands.

6.7.8 Option Data 1 (OPT1) Register

The Flash optional data 1 (OPT1) register stores Flash information row data, an output of the module, and is used at the system level.

Base + $1B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	OPT1															
Write																
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Figure 6-15. Option Data 1 (OPT1) Register

6.7.8.1 Information Flash Row Option 1 (OPT1)—Bits 15–0

All bits can be read in user and test modes; however, the OPT1 register cannot be written in any mode. The OPT1 register is loaded at reset with the value IFR_OPT1 from physical address $xx31 within the Flash information row. Function and use of OPT1 bits is defined by system integrator. (56F80xx stores the factory determined ROSC trim value in the OPT1 register.)

6.7.9 Test Array Signature Register (TSTSIG)

The TSTSIG register stores the Flash information block signature, generated by the calculate IFR block signature command during factory test. The value in the TSTSIG register is compared to the result of the calculate IFR block signature throughout the life of the part to confirm data used by the user commands and internal module adjustments has not been compromised.

Base + $1D	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TEST SIGNATURE															
Write																
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Figure 6-16. Test Array Signature (TSTSIG) Register

6.8 Interrupts

The FM module generates an interrupt when all Flash commands are completed, or when the address, data, and command buffers are empty. Interrupts can also be generated when the ACCERR bit is set.

Table 6-9. Flash Memory Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable
Flash command, data and address buffer empty	CBEIF (USTAT)	CBEIE (CNFG)
All commands are completed on Flash	CCIF (USTAT)	CCIE (CNFG)
ACCERR generated	ACCERR(USTAT)	AEIE(CNFG)

NOTE:

Vector addresses and their relative interrupt priority are determined at the 16-bit controller level.

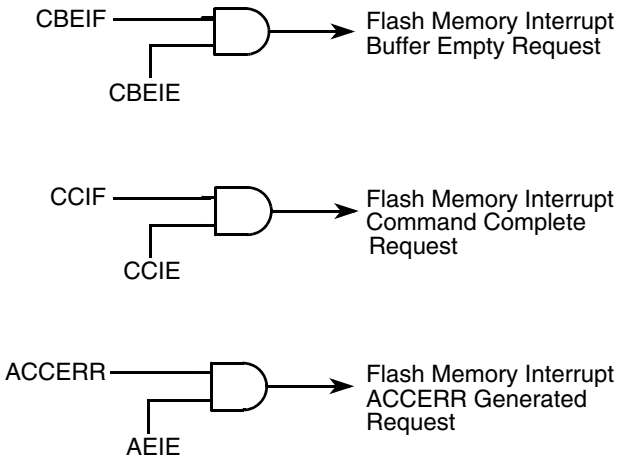


Figure 6-17. Interrupt Implementation

6.9 Resets

The FM module uses the early reset (16 clocks in advance of the core's reset signal) signal supplied by the SIM module to load the reset state of bit fields within the PROT and SECHI registers with data from the FM area. The early reset signal is also used to trigger the read of the security word and the enabling of chip security if it is so configured. Please see Section 6.6 for more details. The Flash array is not accessible for any operations, via the address and data buses, during early reset. If a reset occurs while any command is in progress, that command is immediately aborted. The state of the word being programmed or the page/block being erased is not guaranteed.

Table 6-10. Document Revision History for Chapter 6

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 7
General Purpose Input/Output (GPIO)

7.1 Introduction

The general purpose input/output (GPIO) module allows direct read or write access to pin values, as well as the ability to use a pin as an external interrupt. GPIO pins can also be multiplexed with other peripherals on the chip as well. The *Device Data Sheet* specifies the assigned GPIO ports and its multiplexed pin package.

A GPIO pin may be configured in three ways:

1. As GPIO input with, or without, pullup
2. As GPIO output with push-pull mode or open drain mode
3. As a peripheral pin when multiplexed with another module

GPIOs are placed on the chip in groups of one to 16 bits, called ports, which are designated as port A, B, C, an so on. Please refer to the *Device Data Sheet* for the specific definition of each of the GPIO ports on the chip. For purposes of illustration, a port width of 16 bits is assumed throughout this chapter.

7.2 Features

The GPIO module design includes these characteristics:

- Individual control for each pin to be in either peripheral or GPIO mode
- Individual input/output direction control for each pin in GPIO mode
- Individual pullup enable control for each input pin in either peripheral or GPIO mode
- Individual output push-pull mode or open drain mode control for each output pin in either peripheral or GPIO mode
- Individual output drive strength control for each pin
- Ability to monitor pin logic values even when GPIO are not enabled by using the RDATA register
- Interrupt assert capability
- 5 V tolerant

7.3 Block Diagram

Figure 7-1 illustrates the logic associated with a single multiplexed device pin. Each pin may be configured as either peripheral or GPIO mode. An input edge detection feature and a pullup enable feature are provided independent of the operating mode configuration when the pin is input pin.

When the pin is configured for GPIO mode, the corresponding PE bit in the peripheral enable (PEREN) register must be set to zero. At this time the pin may be configured as either an input or output by setting the corresponding DD bit in the data direction (DDIR) register to zero for input or one for output. When configured as an input, the logic level on the pin can be determined by reading the data (DATA) register. When configured as an output, any data written to the DATA register will be reflected on the output pin.

When the pin is configured for peripheral mode, the corresponding PE bit in the PEREN register will be set to one. In this configuration the corresponding peripheral function will drive the functionality of the pin. For example, the SCI peripheral requires two pins (one transmit and one receive) for correct operation. The two corresponding PE bits must be set to enable these pins for SCI operation. See the specific peripheral section for more details.

As can be seen in the block diagram, the following features are available independent of the peripheral or GPIO mode configuration.

- The edge detection feature is available to provide detection and interrupt notification of rising or falling edge signals. The active state of the input signal is configured in the interrupt edge polarity (IEPOL) register and interrupt notification is enabled in the interrupt enable (IEN) register. The interrupt pending (IPEND) and interrupt edge (IEDGE) registers work in concert to provide detection status and interrupt clearing capability respectively. The interrupt assert (IASSRT) register provides software interrupt generation as well as a way to simulate external interrupts. Please see Section 7.7 for more details.
- While configured as an input, the pins may be internally pulled up by enabling the corresponding bit in the pullup enable (PUPEN) register. The typical value of internal pullup is about 110 KΩ.
- While configured as an output, the pin may be placed in either push/pull or open drain mode through the push/pull output mode control (PPOUTM) register.
- While configured as an output, the output drive capability of the pin may be configured to source either 4 mA or 8 mA through the drive strength control (DRIVE) register.
- The logic level of the pin may be sampled at any time by reading the raw data (RDATA) register. This reading is independent of the input/output configuration of the pin. This feature my be required during the system verification or debugging of the system.
- Capable of accepting 5 V inputs or driving 5 V outputs when configured as open drain mode with external pullups.

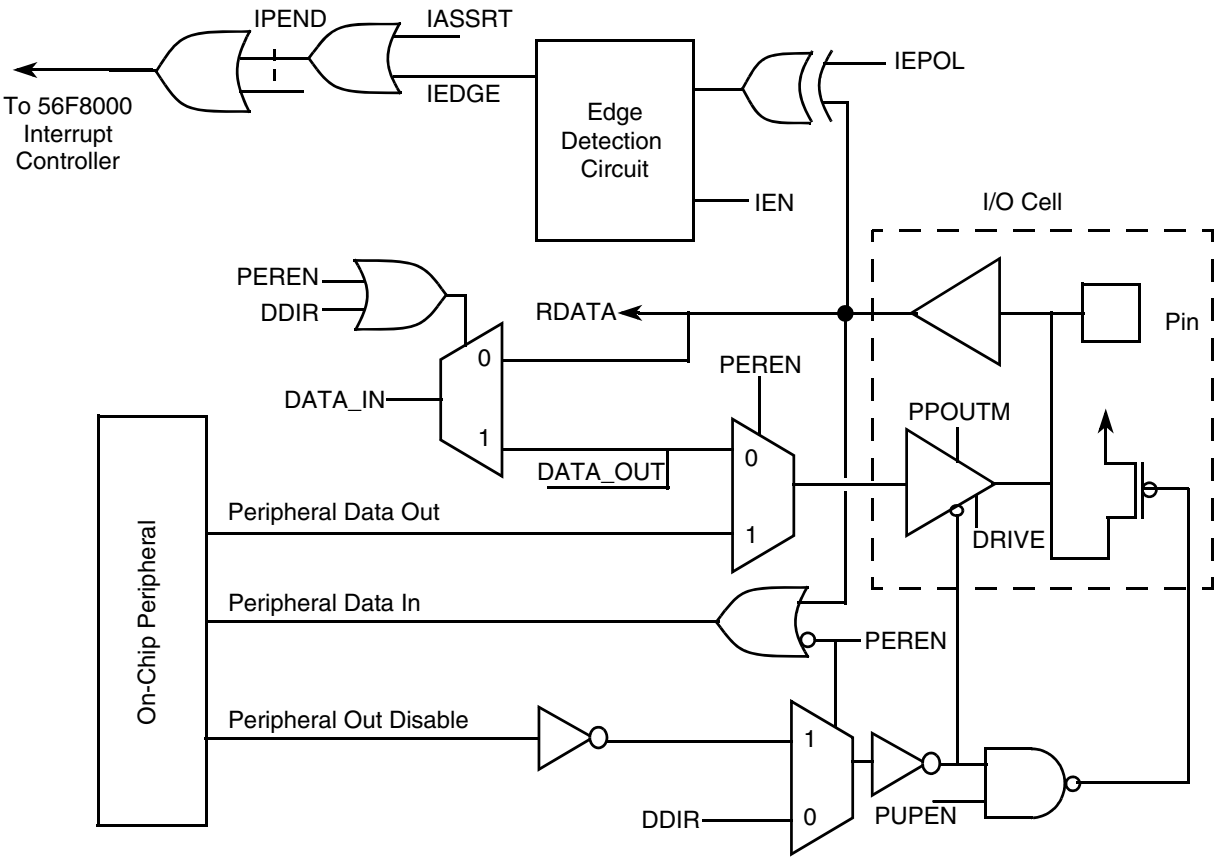


Figure 7-1. Bit-Slice View of the GPIO Logic

7.4 Operating Modes

The GPIO module contains two major modes of operation:

- Peripheral mode—The peripheral module controls the output/input enable and any output data is supplied by the peripheral.
- GPIO mode—In this mode, the GPIO module controls input/output direction of each pin. Any input data can be read from the DATA register. If the pin is configured as input and any data written to the DATA register is reflected on the output pin if it is configured as output.

Pullup enables are controlled by the PUPEN register when a pin is configured as a peripheral input or GPIO input. See Table 7-2 for more detail.

7.5 Register Description

NOTE:

Please refer to the *Device Data Sheet* for the base address of this peripheral.

For illustration purposes, a 16-pin wide GPIO port is assumed for all registers in this chapter. Each register bit performs an identical function for each of the GPIO pins controlled by a specific GPIO port. Besides pin widths, the only other differences between GPIO ports relate to the initial operating conditions at reset. Some GPIO ports are initialized in GPIO mode as default, while others are not. The same is true with pullup resistors. Some may be enabled, others not.

NOTE:

Please be certain to consider the specific chip's availability on GPIO ports. Not all GPIO ports are available on all chips.

A register address is the sum of a base address and an address offset. Please see the *Device Data Sheet* for base addresses. Each GPIO module has 12 registers.

Table 7-1 lists the GPIO registers in ascending address order, including the acronyms and address offset of each register.

Table 7-1. GPIO Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	PUPEN	Pullup enable register	Read/Write	Section 7.5.1
Base + $1	DATA	Data register	Read/Write	Section 7.5.2
Base + $2	DDIR	Data direction register	Read/Write	Section 7.5.3
Base + $3	PEREN	Peripheral enable register	Read/Write	Section 7.5.4
Base + $4	IASSRT	Interrupt assert register	Read/Write	Section 7.5.5
Base + $5	IEN	Interrupt enable register	Read/Write	Section 7.5.6
Base + $6	IPOL	Interrupt edge polarity register	Read/Write	Section 7.5.7
Base + $7	IPEND	Interrupt pending register	Read Only	Section 7.5.8
Base + $8	IEDGE	Interrupt edge sensitive register	Read/Write	Section 7.5.9
Base + $9	PPOUTM	Push-pull output mode register	Read/Write	Section 7.5.10
Base + $A	RDATA	Provides an unclocked version of the data values currently present on each GPIO pin even when not in GPIO mode.	Read-Only	Section 7.5.11
Base + $B	DRIVE	Drive strength register	Read/Write	Section 7.5.12

Each of the registers in the GPIO peripheral is summarized in Figure 7-2 and detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	PUPEN	R	PU															
		W																
$1	DATA	R	D															
		W																
$2	DDIR	R	DD															
		W																
$3	PEREN	R	PE															
		W																
$4	IASSRT	R	IA															
		W																
$5	IEN	R	IEN															
		W																
$6	IPOL	R	IPOL															
		W																
$7	IPEND	R	IPR															
		W																
$8	IEDGE	R	IES															
		W																
$9	PPOUTM	R	OEN															
		W																
$A	RDATA	R	RAWDATA															
		W																
$B	DRIVE	R	DRIVE															
		W																

R	0	Read as 0
W		Reserved

Note: Assumes 16-bit GPIO. This information maybe different depending on the specific part. Please see the device Data Sheet for actual value.

Figure 7-2. GPIO Register Map Summary

7.5.1 Pullup Enable (PUPEN) Register

This read/write register enables and disables the pullup on each GPIO pin. This configuration affects the pin only when it is functioning as an input which may function as an input in both GPIO mode and peripheral mode. Please see Table 7-2 for active conditions.

This pullup is intended only to drive an undriven input pin to a known state. It is characteristically a very weak pullup and it typically will not meet the pullup requirements of an active circuit.

NOTE:

The reset value may be different depending on the specific device. Please see the *Device Data Sheet* for actual value.

Base + $0	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

Figure 7-3. Pullup Enable (PUPEN) Register

- 0 = Pullup is disabled
- 1 = Pullup is enabled (see Section 7-2 for additional conditions)

Table 7-2 provides the state of the pin and PUPEN register as a function of current peripheral state and control register values.

Table 7-2. GPIO Pullup Enable Functionality

PEREN	Pin Operating Mode	PUPEN	DDIR	Pin State	Pin Pullup
0	GPIO	0	0	Input	Disabled
0	GPIO	0	1	Output	Disabled
0	GPIO	1	0	Input	Enabled
0	GPIO	1	1	Output	Disabled
1	Peripheral	x	x	Output	Disabled
1	Peripheral	0	x	Input	Disabled
1	Peripheral	1	x	Input	Enabled

When the peripheral enable (PEREN) register is zero, the pullup enable (PUPEN) register and the data direction (DDIR) register control the pin pullup. When the PEREN register is one, the pin pullup is controlled by the PUPEN and peripheral pin function. The PUPEN value is only recognized when the pin is functioning as an input.

7.5.2 Data (DATA) Register

This read/write register holds data coming either from the pin or the IPBus when pins are configured as GPIO. When pins are configured as GPIO and input, logic level on the pin can be determined by reading the DATA register value. When pins are configured as GPIO and output, value written to the DATA register is reflected on the output pin.

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	D															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-4. Data (DATA) Register

7.5.3 Data Direction (DDIR) Register

When the pin is configured as GPIO mode (PE*n* = 0), this read/write register configures the state of the pin as either an input or output. When DDIR is set to zero, the pin is an input. When DDIR is set to one, the pin is an output.

Base + $2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DD															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-5. Data Direction (DDIR) Register

- 0 = Pin is an input
- 1 = Pin is an output

7.5.4 Peripheral Enable (PEREN) Register

This read/write register determines the GPIO configuration. When PEREN value is one, the peripheral owns the pin. This ownership includes configuring the pin as a required input, or output depending on the status of the peripheral output enable and includes data transfers from the pin to the peripheral. Please see Table 7-2 for additional information. When the PEREN value is zero, the GPIO module owns the pin, controlling the direction of the data flow via the DDIR. Regardless peripheral mode or GPIO mode, input pullup, output drive strength and output push-pull/open drain selection are controlled by the corresponding registers in the GPIO module.

Base + $3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-6. Peripheral Enable (PEREN) Register

NOTE:

The reset value may be different depending upon the specific device. Please see the *Device Data Sheet* for actual value.

- 0 = Pin is for GPIO
- 1 = Pin is for Peripheral

7.5.5 Interrupt Assert (IASSRT) Register

This read/write register is typically used for software testing, but it also provides a software interrupt compatibility. An interrupt is asserted when any IA bit is set to one. The register is cleared by writing zeros. Interrupts will be generated continually until this bit is cleared.

Base + $4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IA															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-7. Interrupt Assert (IASSRT) Register

- 0 = Deassert software interrupt
- 1 = Assert software interrupt

7.5.6 Interrupt Enable (IEN) Register

This read/write register enables or disables the interrupt for each GPIO pin. Set a bit to one to enable interrupts for the associated GPIO pin. Interrupts are recorded in the IPEND register, if edge transition is detected.

Base + $5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IEN															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-8. Interrupt Enable (IEN) Register

- 0 = Edge interrupt is disabled
- 1 = Edge interrupt is enabled

7.5.7 Interrupt Polarity (IPOL) Register

This read/write register controls the edge polarity of any external interrupts enabled by an IEN bit set to one. When this register is set to one, the falling edge causes the interrupt. When this register is set to zero, the rising edge causes the interrupt.

Base + $6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IPOL															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-9. Interrupt Polarity (IPOL) Register

- 0 = Rising edge generates the interrupt
- 1 = Falling edge generates the interrupt

7.5.8 Interrupt Pending (IPEND) Register

This read-only register records any incoming interrupts. This register is read to determine which pin or bit of IASSRT register caused the interrupt. For external interrupts, the IPEND is cleared by writing ones into the IEDGE register. For software interrupts, the IPEND is cleared by writing zeros into the IASSRT register.

Base + $7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IPR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-10. Interrupt Pending (IPEND) Register

- 0 = Interrupt cleared
- 1 = Interrupt pending

7.5.9 Interrupt Edge Sensitive (IEDGE) Register

When an edge is detected by the edge detector circuit, and IEN is set to one, IEDGE records the interrupt. The corresponding bit in the IPEND register is also set. This read/write register clears the corresponding IPR bit field by writing one to the corresponding bit in the IEDGE register. Writing zero to an IEDGE bit is ignored.

Base + $8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IES															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-11. Interrupt Edge Sensitive (IEDGE) Register

- 0 = No edge seen
- 1 = Edge detected

7.5.10 Push/Pull Output Mode Control (PPOUTM) Register

This read/write register explicitly sets each output driver to either push/pull or open drain mode, independent of peripheral or GPIO mode configuration at the pin.

NOTE:

The open drain mode can be used to tri-state any pin on the GPIO port without switching that pin to input mode. This is useful for some applications, including a keypad interface. These bits default to push/pull mode upon reset, giving all GPIO ports the same default functionality in this regard.

Base + $9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	OEN															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 7-12. Push/Pull Output Mode Control (PPOUTM) Register

- 0 = Open drain mode
- 1 = Push/Pull mode

7.5.11 Raw Data (RDATA) Register

This read-only register allows the controller direct access to the logic values on each GPIO pin even when pins are not in the GPIO mode. Values are not locked and are subject to change at any time. The reset state is unknown. It is recommended to read several times to assure a stable value.

Base + $A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RAWDATA															
Write																
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 7-13. Raw Data (RDATA) Register

- 0 = Logic 0 present on GPIO pin
- 1 = Logic 1 present on GPIO pin

7.5.12 Drive Strength Control (DRIVE) Register

This read/write register can be used to explicitly set the drive strength of each output driver, independent of the peripheral or GPIO mode configuration of the pin. This feature provides an additional design variable with the ability to control the effects of electromagnetic interference (EMI) due to digital switching. For capacitive loads, higher drive strengths improve low to high transition rates but increase EMI. Lower drive strengths reduce EMI and circuit board trace width requirements. These bits default to 4 mA upon reset, providing all GPIO ports with the same default functionality.

Base + $B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DRIVE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-14. Drive Strength Control (DRIVE) Register

- 0 = 4 mA drive strength outputs
- 1 = 8 mA drive strength outputs

7.6 Clocks and Resets

The module runs at standard IPBus speeds and will assume reset states as defined in each chip specification. Reset occurs whenever any source of system reset occurs (power-on reset (POR), external reset (RESET), or COP or, software reset).

7.7 Interrupts

The GPIO has two types of interrupts:

1. Software interrupt

   The interrupt assert (IASSRT) register is used to generate the software interrupt. It can be generated by writing ones to IASSRT. The interrupt pending (IPEND) register will record the value of IASSRT. The IPEND register can be cleared by writing zeros into the IASSRT during the IASSRT testing.

NOTE:

When testing the IASSRT, the interrupt edge polarity (IEPOL) register, interrupt edge sensitive (IEDGE) register, and the interrupt enable (IEN) register must be zero to guarantee the interrupt registered in the IPEND is due to IASSRT only.

2. Hardware interrupt from the pin

   When a GPIO pin is used as an external interrupt source, its IEN bit is set to one and the IASSRT register must be set to zero. The IEPOL register must be set to one for an falling edge interrupt and zero 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 IEDGE register and recorded by the IPEND register. Please see Table 7-3. The IPEND is cleared by writing 1s into the IEDGE.

   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 IPEND 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.

Table 7-3. GPIO Interrupt Assert Functionality

IPOLR	Interrupt Asserted	Remark
0	Rising Edge	If the IEN is set to 1, as the pin goes from low to high an interrupt will be recorded by the IPEND register.
1	Falling Edge	If the IEN is set to 1, as the pin goes from high to low an interrupt will be recorded by the IPEND register.

Table 7-4. Document Revision History for Chapter 7

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 8
Pulse Width Modulator (PWM)

8.1 Introduction

This chapter describes the pulse width modulator (PWM) module. The PWM can be configured as three complementary pairs, six independent PWM signals or their combinations, such as one complementary and four independent. Both edge- and center-aligned synchronous pulse width control, from 0 to 100% modulation, are supported.

A 15-bit common PWM counter is applied to all six channels. PWM resolution is one clock period for edge-aligned operation and two clock periods for center-aligned operation. The clock period is dependent on clock source frequency at either system clock or 3× system clock and a programmable prescaler.

When generating complementary PWM signals, the module features automatic deadtime insertion to PWM output pairs. Each PWM output can be controlled by PWM generator, timer, conversion results of ADC, GPIO pins, or software manually and separate top and bottom output polarity control. Asymmetric PWM output is able to change PWM duty cycle alternatively at every half cycle without software involvement.

8.2 Features

- PWM operation clock runs at either system clock or 3× system clock
- 6 PWM signals
  - All independent
  - Complementary pairs
  - Mix independent and complementary
- Features of complementary channel operation
  - Separate deadtime insertions for rising and falling edges
  - Separate top and bottom pulse width correction through software
  - Asymmetric PWM output within center align operation
  - Separate top and bottom polarity control
- Edge- or center-aligned PWM signals
- 15 bits of resolution
- Half-cycle reload capability
- Integral reload rates from 1 to 16
- Individual software controlled PWM output

- Programmable fault protection
- PWM compare output polarity control
- PWM output polarity control
- Push-Pull and open drain modes are available on PWM pins
- External sync control with sync window
- Programmable input filters for the fault signals
- Write-protected registers
- Selectable PWM supply source for each complementary PWM signal pair
  - PWM generator
  - External GPIO pin
  - Internal timer channel
  - ADC conversion result, taking into account values set in ADC high and low limit registers

All three pairs can be driven by any one of the external sources. The following features are disabled when any of external sources is used as PWM source:

- PWM sync is not applicable
- PWM reload registers have no effect

8.3 Block Diagram

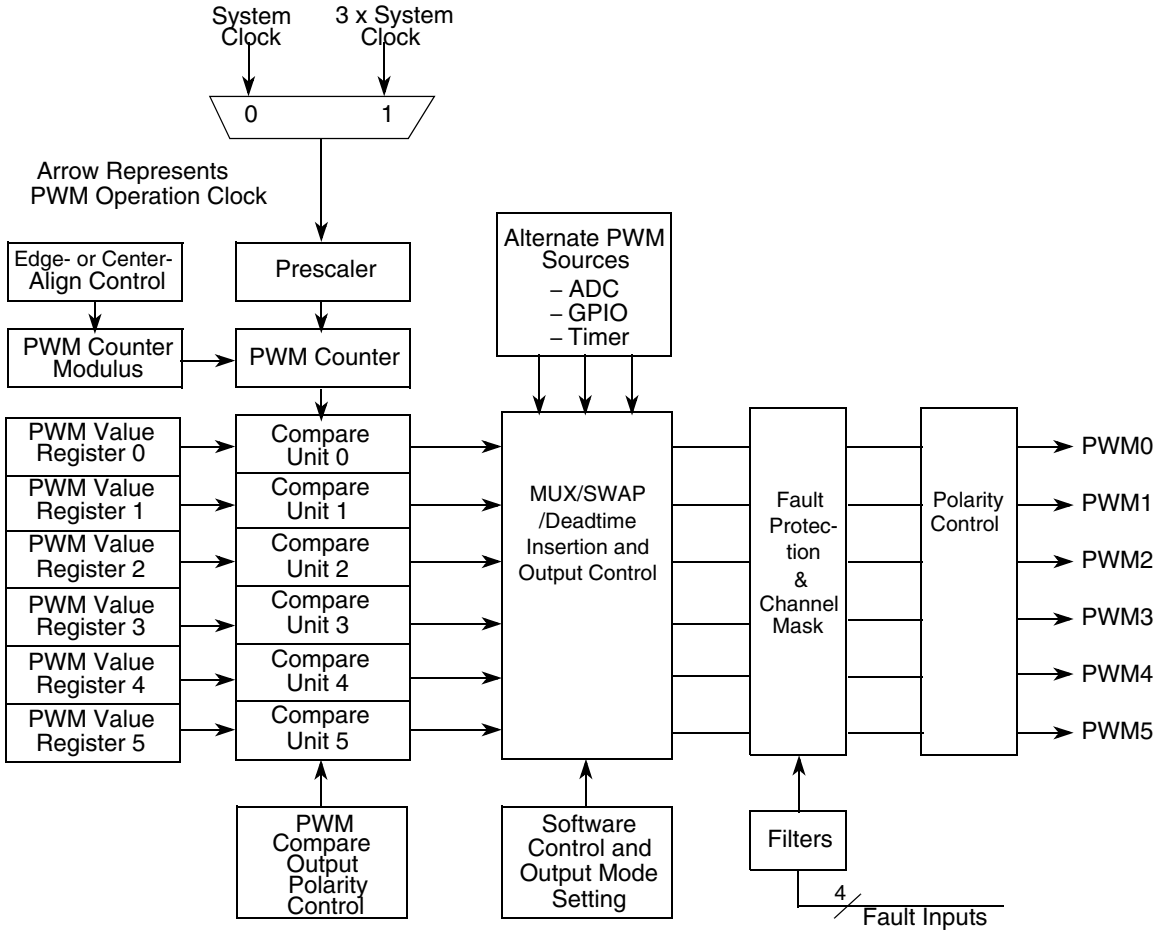


Figure 8-1. PWM Block Diagram

The MUX/Swap block is tightly integrated with the dead time insertion block. This detail is shown in Figure 8-2. SWAP/MASK functionality can be programmed to 56F80xx compatible mode (default, shown as SWAP/MASK0/MASK1), or enhanced (SWAP*n*/MASK0*n*/MASK1*n*). The choice is made using the nBX bit of the CCTRL register. Please see Section 8.7.11.2.

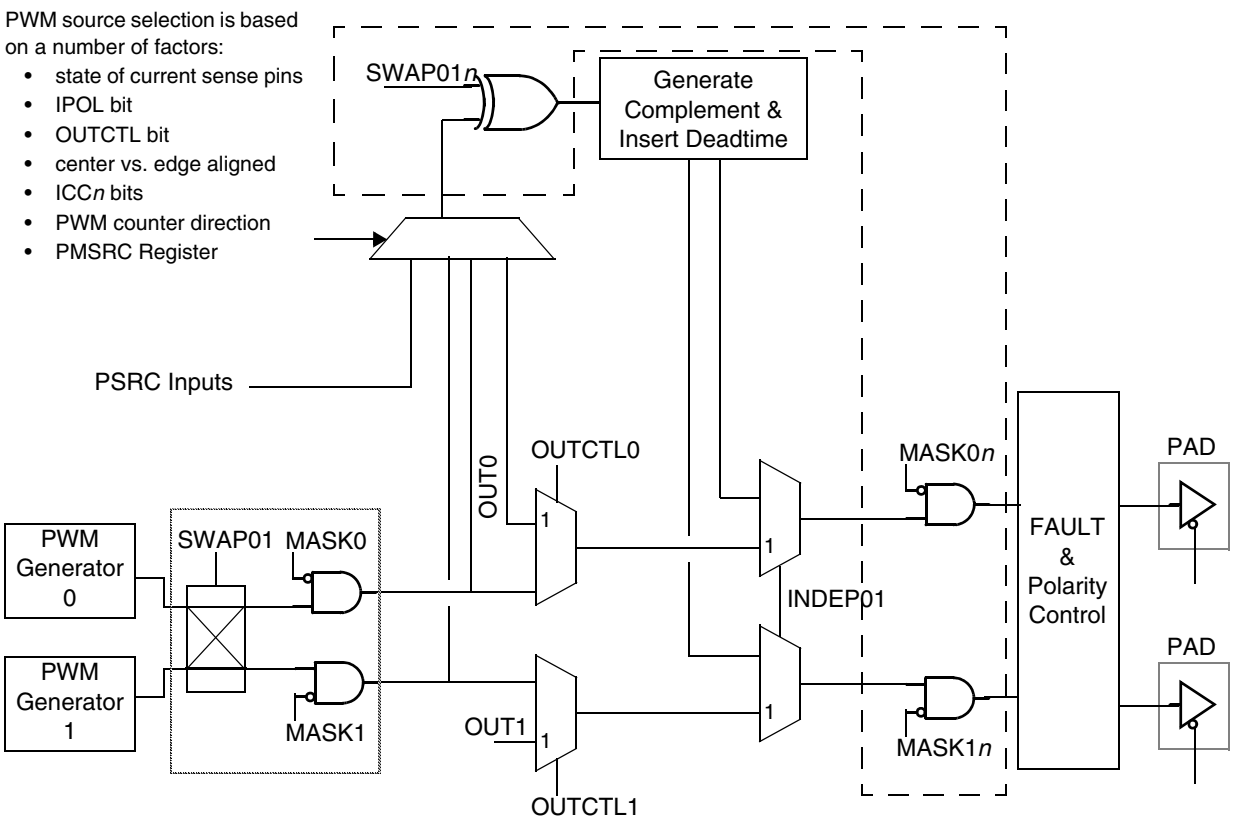


Figure 8-2. Detail of MUX, Swap, and Deadtime Functions

8.4 Functional Description

A block diagram of the PWM is illustrated in Figure 8-1.

8.4.1 Prescaler

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by one, two, four, or eight. The prescaler bits (PRSC[3:0]) in the control (CTRL) register, select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until the LDOK bit is set and a new PWM reload cycle begins.

8.4.2 PWM Generator

The PWM generator contains a 15-bit up/down PWM counter producing output signals with software-selectables:

- Alignment—The logic state of the EDGE bit in the configuration register determines whether the PWM output is edge-aligned or center-aligned
- Period—The value written to the counter modulos (CMOD) register is used to determine the PWM period. The period can also be varied by using the prescaler
  - With edge-aligned output, the modulus is the period of the PWM output in clock cycles
  - With center-aligned output, the modulus is one-half of the PWM output period in clock cycles

- Pulse Width—The number written to the value (VAL0–5) register determines the pulse width duty cycle of the PWM output in clock cycles
  - With edge-aligned output, the pulse width is the value written to the VAL0–5 register
  - With center-aligned output, the pulse width is twice the value written to the VAL0–5 register

Use of the PWM generator is optional for complementary PWM output pairs. The SCTRL register can be used to alternately specify the PSRC inputs as source for the complementary signal.

Each PWM output pair may be controlled individually, or ganged with one or both of the other pairs by specifying the same source for each in the SCTRL.

8.4.3 Alignment

The edge-align (EDGE) bit in the configure (CNFG) register selects either center-aligned or edge-aligned PWM generator outputs.

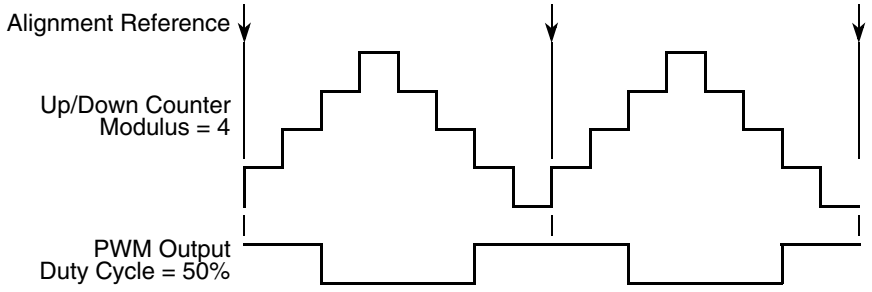


Figure 8-3. Center-Aligned PWM Output

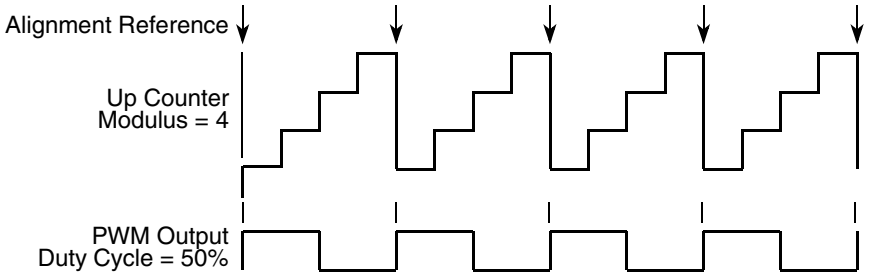


Figure 8-4. Edge-Aligned PWM Output

NOTE:

Because of the equals-comparator architecture of this PWM, the modulus equals zero case is considered illegal. Therefore, the modulus register is not reset, and a modulus value of zero results in waveforms inconsistent with the other modulus waveforms. If a modulus of zero is loaded, the counter continually counts down from $7FFF. This operation was not tested or is it guaranteed. Consider it illegal. However, the deadtime constraints and fault conditions are still guaranteed.

8.4.4 Period

The PWM period is determined by the value written to the counter modulus (CMOD) register.

The PWM counter is an up/down counter in a center-aligned operation. In this mode the PWM highest output resolution is two IPBus clock cycles. See the chip introduction section for its IPBus clock frequency.

PWM period = (PWM modulus) × (PWM clock period) × 2

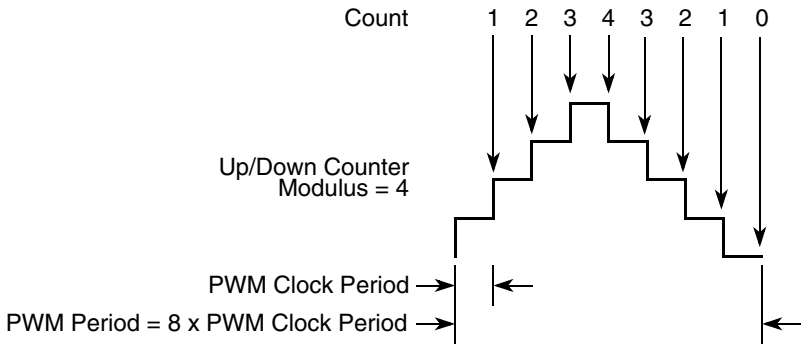


Figure 8-5. Center-Aligned PWM Period

In an edge-aligned operation, the PWM counter is an up-counter. The PWM output resolution is one IPBus clock cycle.

PWM period = PWM modulus × PWM clock period

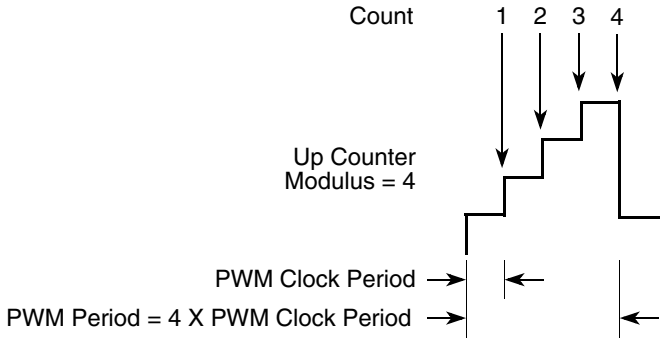


Figure 8-6. Edge-Aligned PWM Period

8.4.5 Duty Cycle

The signed 16-bit number written to the PWM value registers is the pulse width in PWM clock periods of the PWM generator output.

$$\text{Duty cycle} = \frac{\text{PWM value}}{\text{Modulus}} \times 100$$

NOTE:

A PWM value less than, or equal to zero deactivates the PWM output for the entire PWM period. A PWM value greater than, or equal to the modulus activates the PWM output for the entire PWM period.

Table 8-1. PWM Value and Underflow Conditions

PMVAL*n*	Condition	PWM Value Used
$0000–$7FFF	Normal	Value in Registers
$8000–$FFFF	Underflow	$0000

Center-aligned operation is illustrated in Figure 8-7.

PWM pulse width = (PWM value) × (PWM clock period) × 2

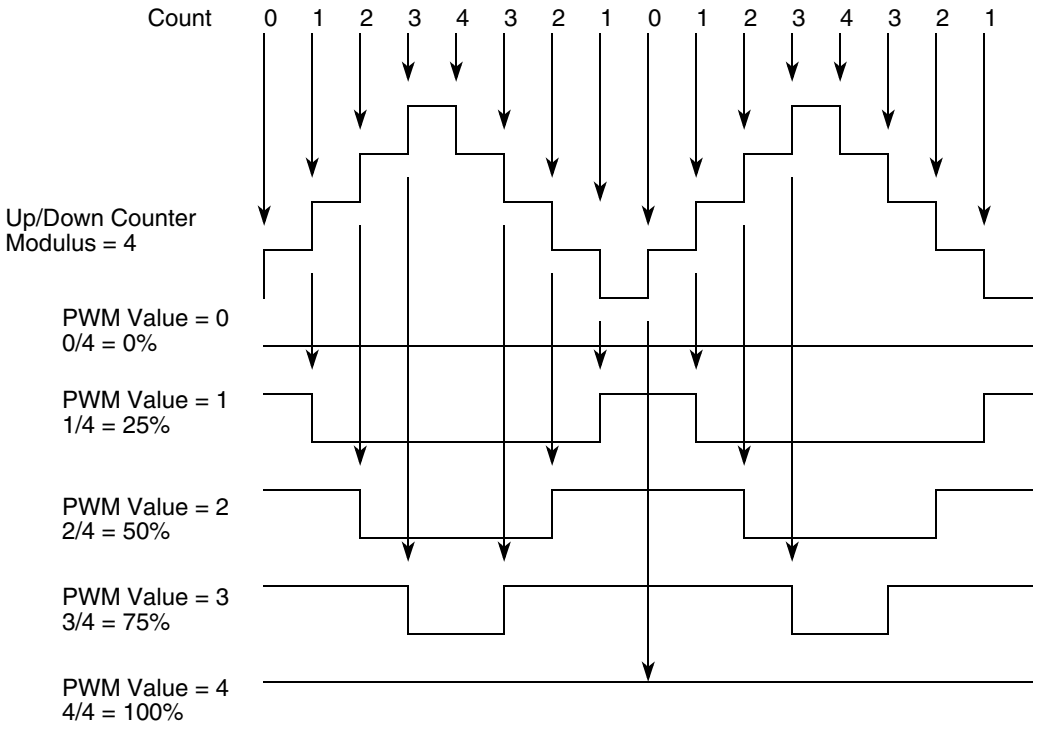


Figure 8-7. Center-Aligned PWM Pulse Width

Edge-aligned operation is illustrated in Figure 8-8.

PWM pulse width = (PWM value) × (PWM clock period)

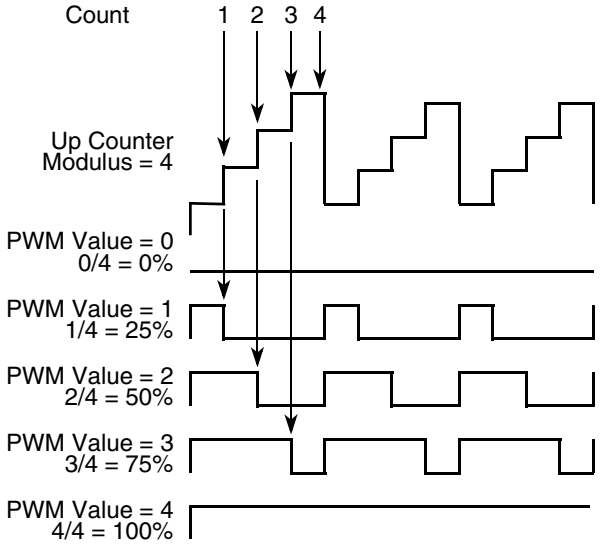


Figure 8-8. Edge-Aligned PWM Pulse Width

8.5 Independent or Complementary Channel Operation

In the configure (CNFG) register, writing logic one to the INDEP*nn* bit configures a pair of the PWM outputs as two independent PWM channels. Each PWM output has its own PWM value register operating independently of the other channels in independent channel operation.

In the CNFG register, writing logic zero to the INDEP*nn* bit configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in Figure 8-9 in complementary channel operation.

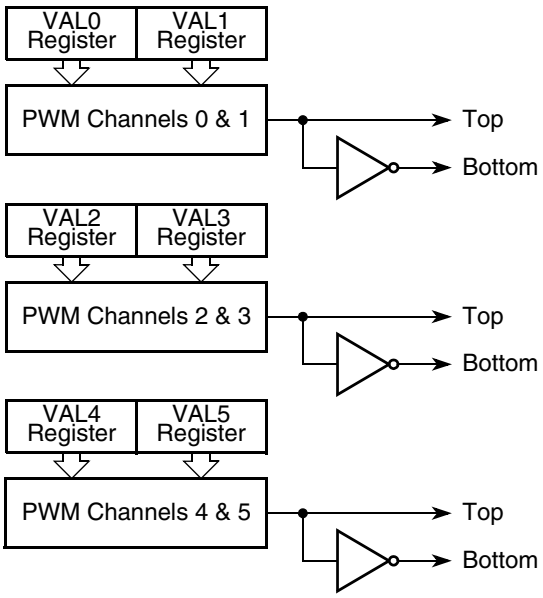


Figure 8-9. Complementary Channel Pairs

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in Figure 8-2.

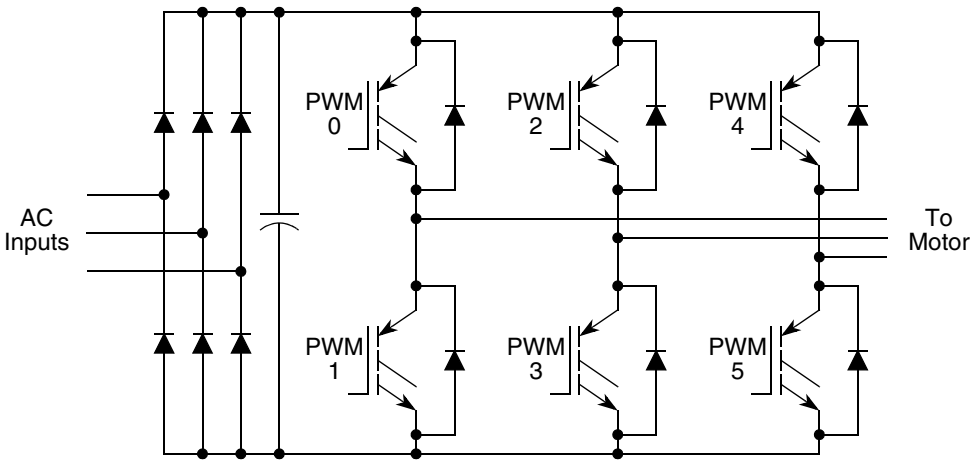


Figure 8-10. Typical 3-Phase AC Motor Drive

In complementary channel operation, there are three additional features:

1. Deadtime insertion
2. Separate top and bottom pulse width correction for distortions caused by deadtime inserted and the motor drive characteristics.
3. Separate top and bottom output polarity control.

8.6 Deadtime Generators

While in the complementary mode, each PWM pair can be used to drive top/bottom transistors, illustrated in Figure 8-11. Ideally, the PWM pairs are an inversion of each other. 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 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.

Deadtime generators automatically insert software-selectable activation delays into each pair of PWM outputs. The deadtime (DTIM0 and DTIM1) registers specify the number of PWM clock cycles to use for deadtime delay. Every time the deadtime generator inputs changes state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

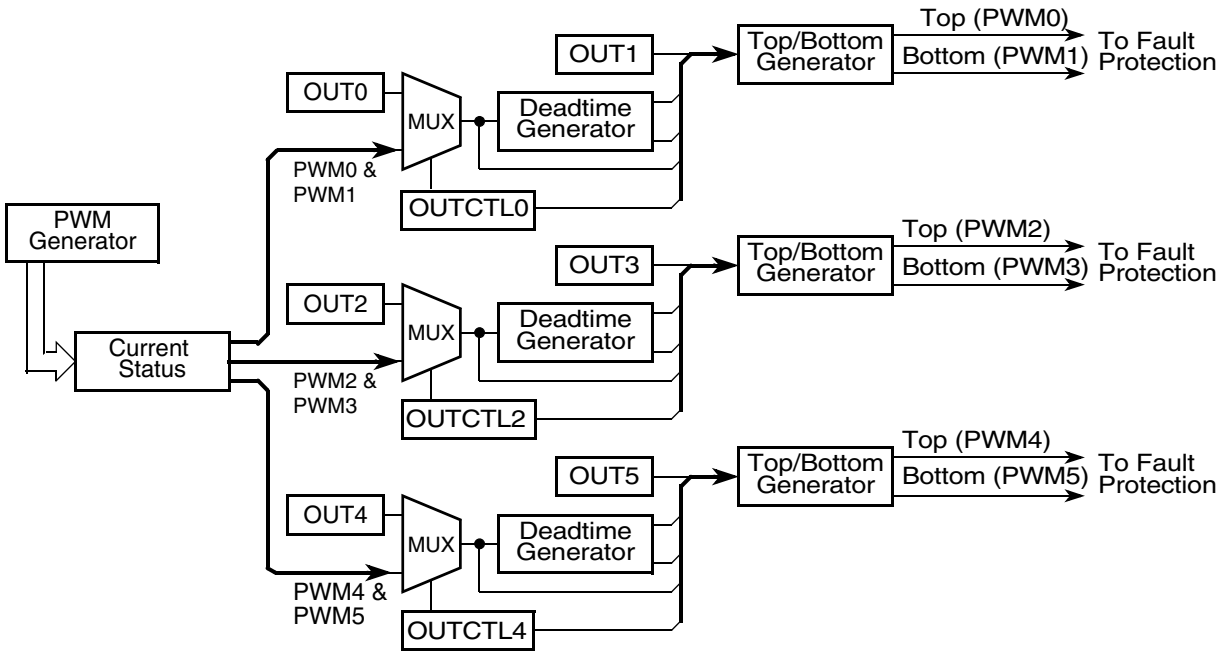


Figure 8-11. Deadtime Generators

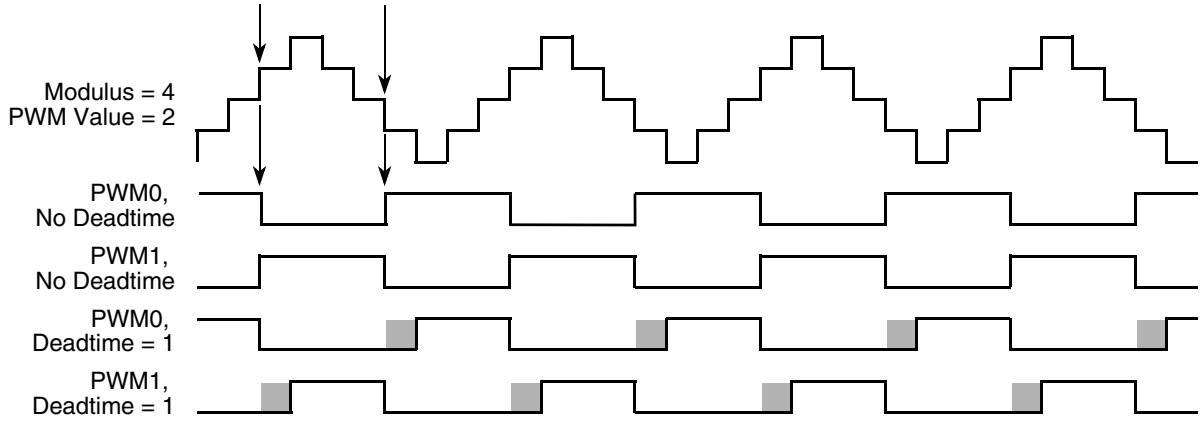


Figure 8-12. Deadtime Insertion, Center Alignment

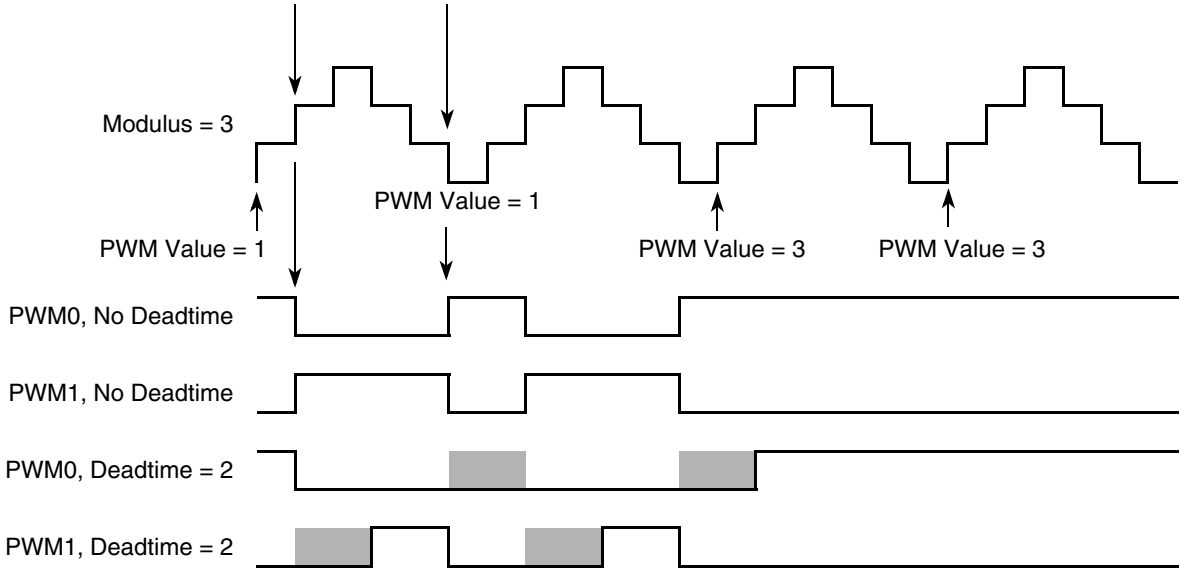


Figure 8-13. Deadtime at Duty Cycle Boundaries

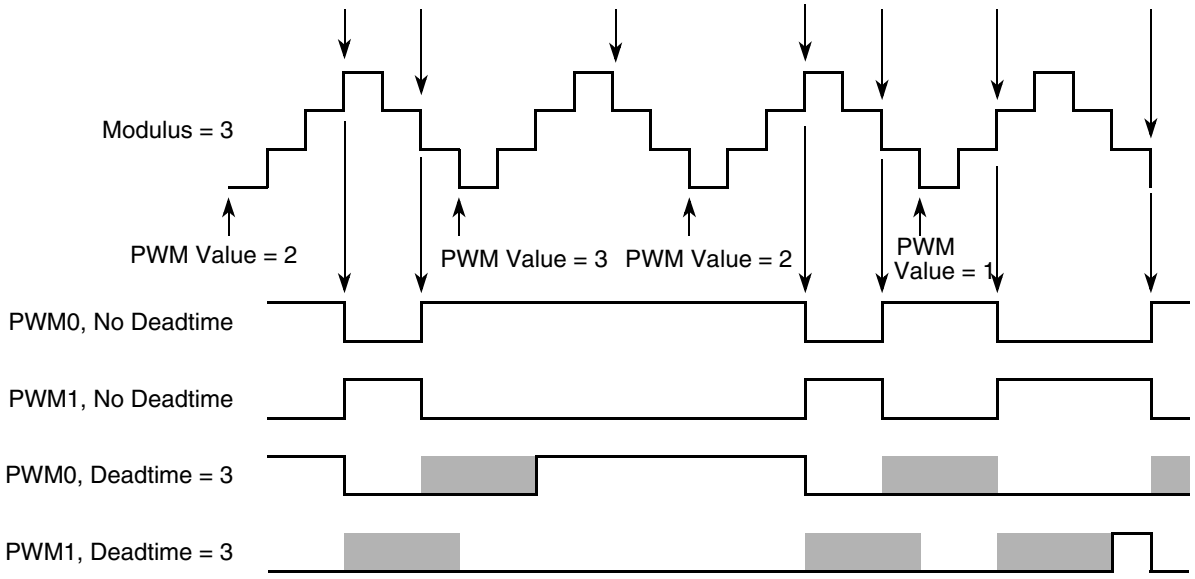


Figure 8-14. Deadtime and Small Pulse Widths

NOTE:

The waveform at the pad is delayed by two IPBus clock cycles for deadtime insertion.

8.6.1 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 Figure 8-15. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

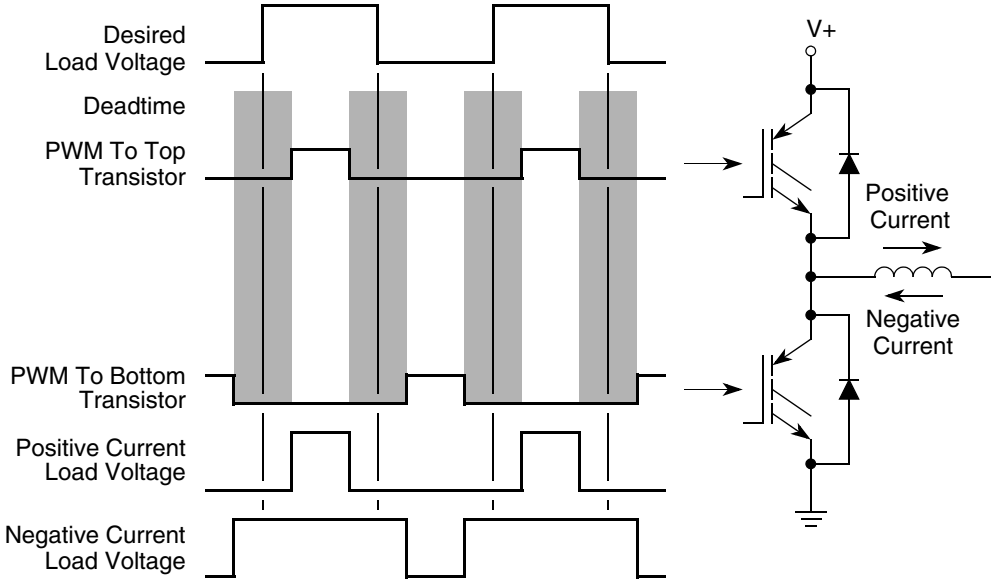


Figure 8-15. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage varying 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 the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output is 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 is effective in controlling the output voltage at any given time. This depends on the direction of the motor current for that pair. See Figure 8-15. 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 an odd-numbered/even numbered PWM register pair. Either the odd or the even VAL0–5 register controls the pulse width at any given time. For a given PWM pair, whether the odd or even VAL0–5 register is active depends on either:

- The state of the odd/even correction bit, IPOL*n*, for that driver
- The direction of PWM counter if ICC bits in ICCTRL register are set.

To correct deadtime distortion, software can decrease or increase the value in the appropriate VAL0–5 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.

8.6.2 Manual Correction

The IPOL0–IPOL2 bits select either the odd or the even PWM value registers to use in the next PWM cycle.

Table 8-2. Top/Bottom Manual Correction

Bit	Logic State	Output Control
IPOL0	0	VAL0 controls PWM0/PWM1 pair
	1	VAL1 controls PWM0/PWM1 pair
IPOL1	0	VAL2 controls PWM2/PWM3 pair
	1	VAL3 controls PWM2/PWM3 pair
IPOL2	0	VAL4 controls PWM4/PWM5 pair
	1	VAL5 controls PWM4/PWM5 pair

NOTE:

IPOL*n* bits are buffered so only one PWM register is used per PWM cycle. If an IPOL*n* bit changes during a PWM period, the new value does not take effect until the next PWM period.

IPOL*n* bits take effect at the end of each PWM cycle regardless of the state of the load okay bit, LDOK.

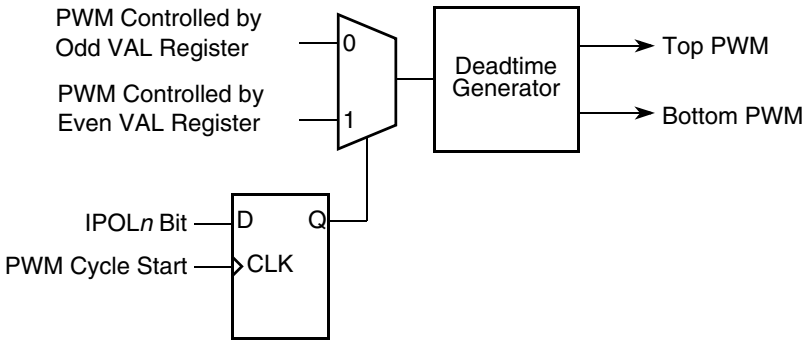


Figure 8-16. Internal Correction Logic

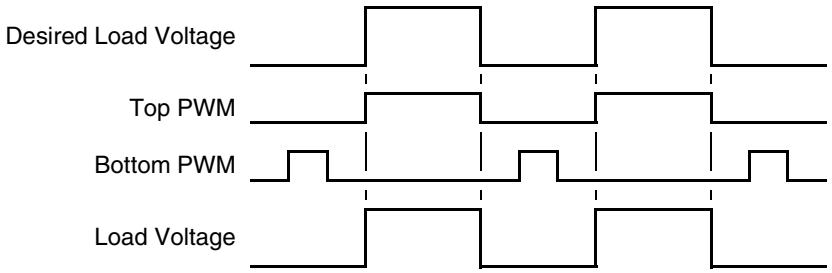


Figure 8-17. Correction with Positive Current

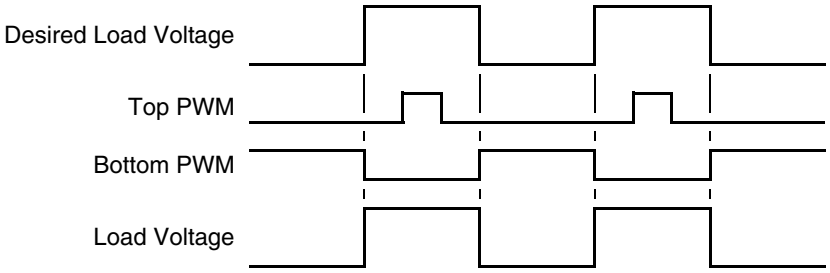


Figure 8-18. Correction with Negative Current

8.6.3 Asymmetric PWM Output

In complementary mode with center aligned operation, the PWM duty cycle is able to change alternatively at every half cycle. The count direction of the PWM counter selects either the odd or even PWM value registers to use in the PWM cycle. For counting up, select even PWM value registers to use in the PWM cycle. For counting down, select odd PWM value registers.

Table 8-3. Top/Bottom Correction Using ICCTRL*n* Bits

Bit	Logic State	Output Control
ICC0	0	IPOL0 controls PWM0/PWM1 pair
	1	PWM count direction controls PWM0/PWM1 pair
ICC1	0	IPOL1 controls PWM2/PWM3 pair
	1	PWM count direction controls PWM2/PWM3 pair
ICC2	0	ISPOL2 controls PWM4/PWM5 pair
	1	PWM count direction controls PWM4/PWM5 pair

NOTE:

If an ICC*n* bit in ICCTRL register changes during a PWM period, the new value does not take effect until the next PWM period.

ICC*n* bits take effect at the end of each PWM cycle regardless of the state of the load okay bit, LDOK.

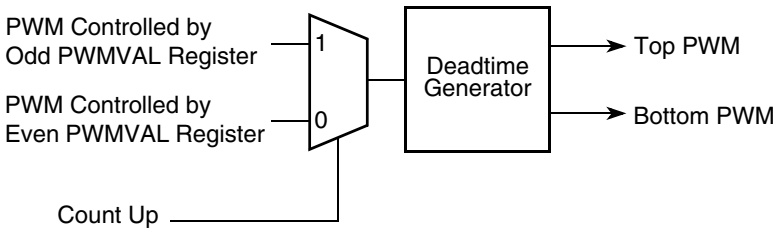


Figure 8-19. Correction Logic with ICC*n* Bits

8.6.4 Output Polarity

Positive polarity means when the PWM is active its output is high. Conversely, negative polarity means when the PWM is active its output is low. Please see Figure 8-20.

Output polarity of the PWMs is determined by two options:

1. TOPNEG controls the polarity of PWM0, PWM2 and PWM4 outputs, typically drives the top transistors of the pair. When TOPNEG is set these outputs are active-low.
2. BOTNEG controls the polarity of PWM1, PWM3 and PWM5 outputs, typically drives the bottom transistors of the pair. When BOTNEG is set these outputs are active-low.

The TOPNEG and BOTNEG control bits are in the configure (CNFG) register.

8.6.5 PWM Generator Loading

8.6.5.1 Load Enable

The load okay (LDOK) bit enables loading the PWM generator with:

- A prescaler divisor—from the PRSC1 and PRSC0 bits in the CTRL register
- A PWM period—from the CMOD registers
- A PWM pulse width—from the VAL0–5 registers

LDOK prevents reloading of these PWM parameters before software is finished calculating them. Setting LDOK allows the prescale bits, CMOD and VAL*n* registers to be loaded into a set of buffers. The loaded buffers are used by the PWM generator at the beginning of the next PWM reload cycle. Set LDOK by reading it when it is logic 0 and then writing logic 1 to it. After loading, LDOK is automatically cleared.

8.6.5.2 Load Frequency

The LDFQ bits in the CTRL register select an integral loading frequency of one to 16-PWM reload opportunities. The LDFQ bits take effect at every PWM reload opportunity, regardless the state of the load okay (LDOK) bit. The HALF bit in the CTRL register controls half-cycle reloads for center-aligned PWMs. If the HALF bit is set, a reload opportunity occurs at the beginning of every PWM cycle and half cycle when the count equals the modulus. If the half bit is not set, a reload opportunity occurs only at the beginning of every cycle. Reload opportunities can only occur at the beginning of a PWM cycle in edge-aligned mode.

NOTE:

Loading a new modulus on a half cycle forces the count to the new modulus value minus one on the next clock cycle. Half cycle reloads are possible only in center-aligned mode. Enabling or disabling half-cycle reloads in edge-aligned mode has no effect on the reload rate

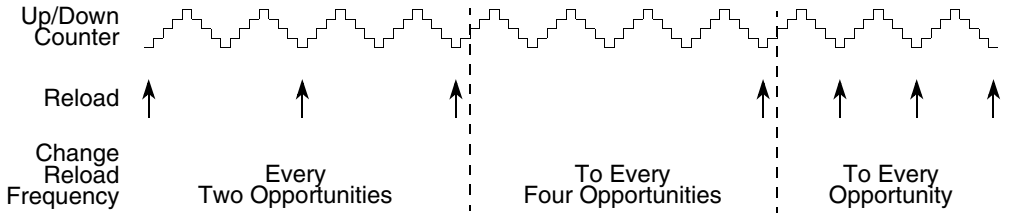


Figure 8-20. Full Cycle Reload Frequency Change

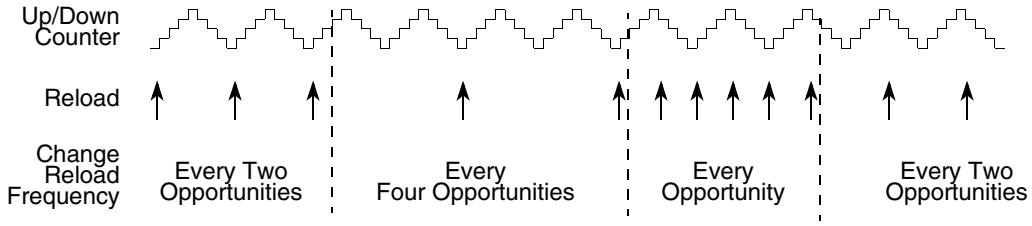


Figure 8-21. Half Cycle Reload Frequency Change

8.6.6 Reload Flag

At every reload opportunity the reload flag (PWMF) bit in the CTRL register is set. Setting PWMF bit in the CTRL register happens even if an actual reload is prevented by the LDOK bit. If the reload interrupt enable (PWMRIE) bit is set, the PWMF flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When PWMRIE bit is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

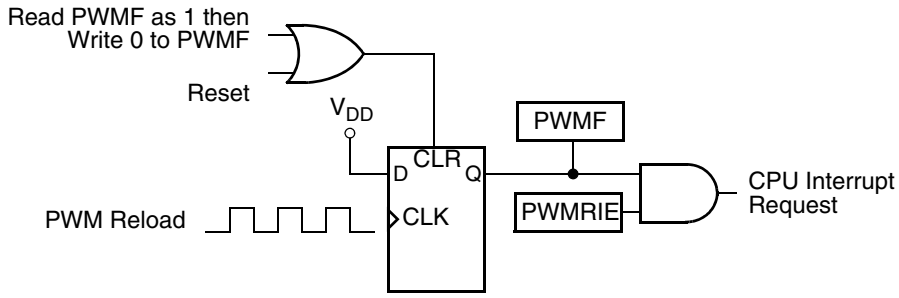


Figure 8-22. PWMF Reload Interrupt Request

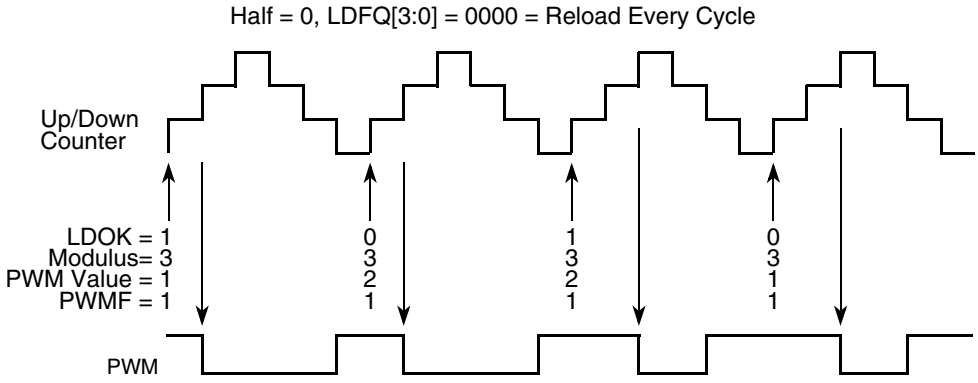


Figure 8-23. Full-Cycle Center-Aligned PWM Value Loading

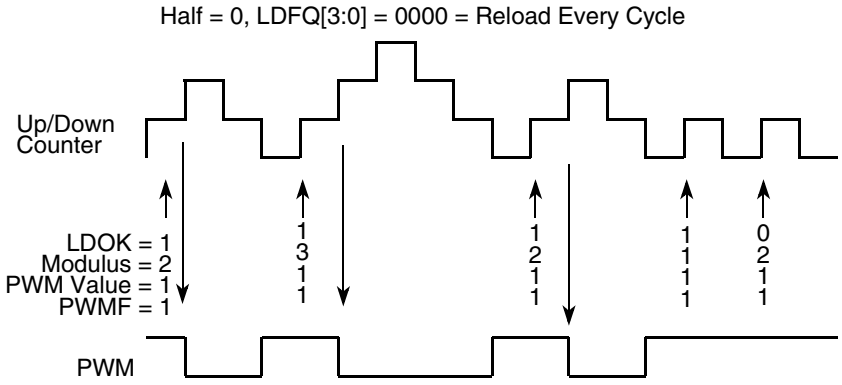


Figure 8-24. Full-Cycle Center-Aligned Modulus Loading

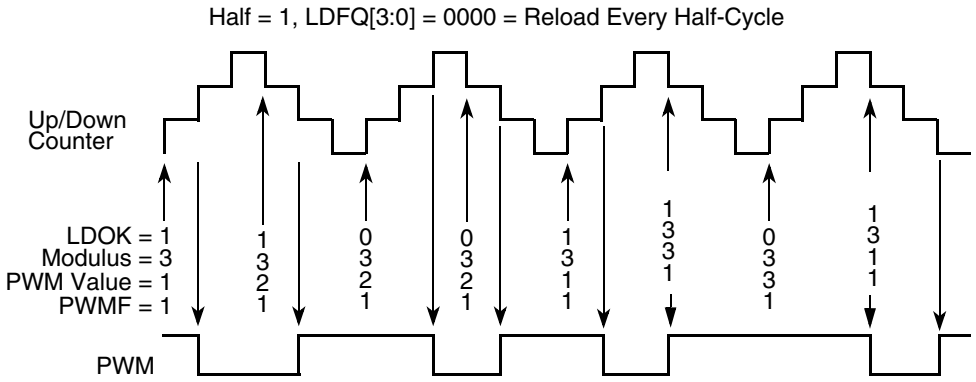


Figure 8-25. Half-Cycle Center-Aligned PWM Value Loading

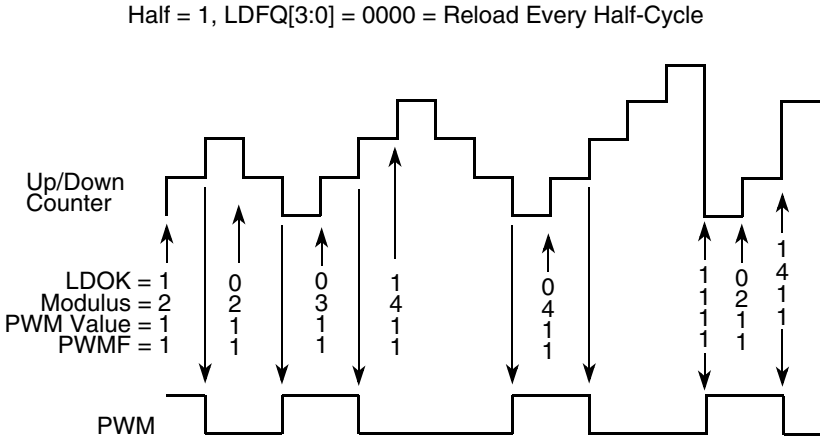


Figure 8-26. Half-Cycle Center-Aligned Modulus Loading

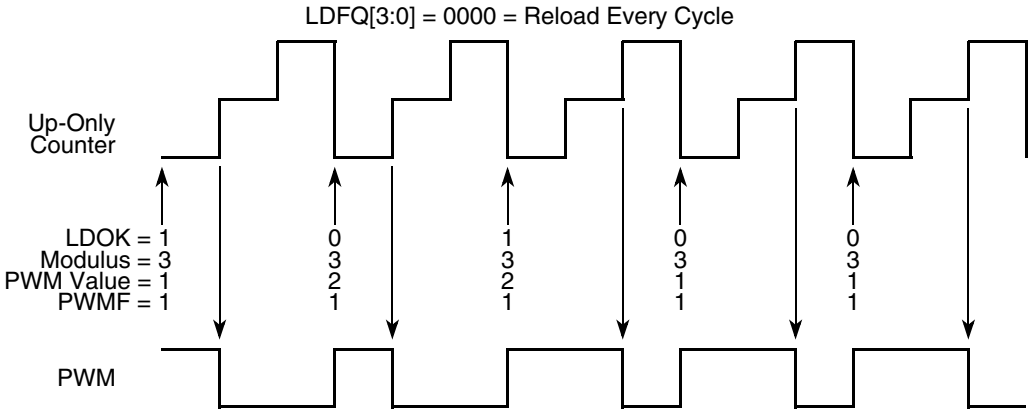


Figure 8-27. Edge-Aligned PWM Value Loading

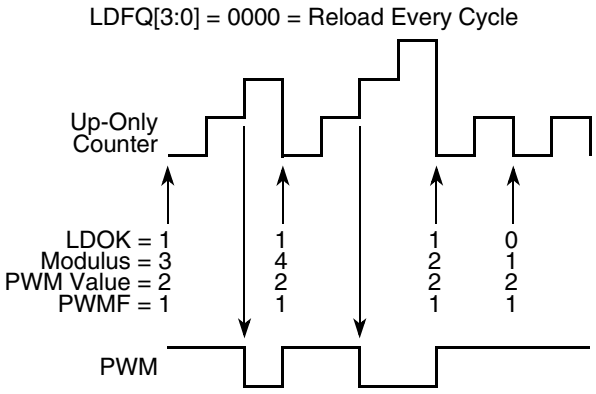


Figure 8-28. Edge-Aligned Modulus Loading

8.6.7 Internal Synchronization Output

The PWM outputs a synchronization pulse connected as an input to the synchronization module, timer A. A high-true pulse occurs for each reload of the PWM regardless of the state of the LDOK bit. When half-cycle reloads are enabled, HALF equals one in the CTRL register, the pulse can occur on the half cycle.

8.6.8 Initialization

Initialize all registers and set the LDOK bit before setting the PWMEN bit. With LDOK set, setting PWMEN bit for the first time after reset immediately loads the PWM generator thereby setting the PWMF flag. PWMF generates a CPU interrupt request if the PWMRIE bit is set. In complementary channel operation with current-status correction selected, value registers one, three, and five control the outputs for the first PWM cycle.

NOTE:

Even if LDOK is not set, setting PWMEN bit also sets the PWMF flag. To prevent a CPU interrupt request, clear the PWMRIE bit before setting PWMEN.

Setting PWMEN for the first time after reset without first setting LDOK loads a prescaler divisor of one, a PWM value of $0000, and an unknown modulus.

The PWM generator uses the last values loaded if PWMEN is cleared and then set while LDOK equals zero.

Initializing the deadtime register, after setting PWMEN bit or OUTCTL*n*, can cause an improper deadtime insertion. However, the deadtime can never be shorter than the specified value.

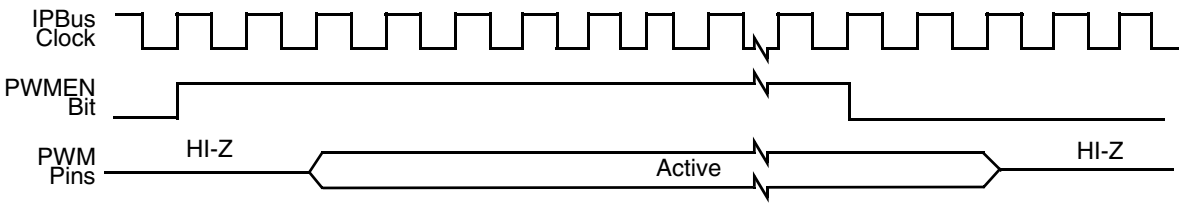


Figure 8-29. PWMEN and PWM Pins in Independent Operation (OUTCTL0–5 = 0)

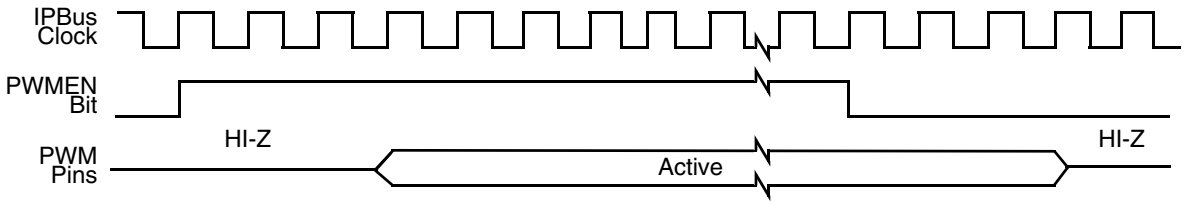


Figure 8-30. PWMEN and PWM Pins in Complementary Operation (OUTCTL0, 2, 4 = 0)

When the PWMEN bit is cleared:

- The PWM*n* pins are tri-stated unless OUTCTL*n* = 1
- The PWM counter is cleared and does not count
- The PWM generator forces its outputs to zero
- The PWMF flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software output control remains active
- Deadtime insertion continues during software output control

8.6.9 Fault Protection

Fault protection can disable any combination of PWM pins. Faults are generated by logic 1 on any of the FAULT pins. Each FAULT pin can be mapped arbitrarily to any of the PWM pins.

When fault protection hardware disables PWM pins, the PWM generator continues to run, only the output pins are deactivated.

The fault decoder disables PWM pins selected by the fault logic and the disable mapping (DISMAP) register. Please see Figure 8-31. Each bank of four bits in the DISMAP register control the mapping for a single PWM pin. Refer to Table 8-1.

The fault protection is enabled even when the PWM is not enabled; therefore, a fault is latched in and must be cleared in order to prevent an interrupt when the PWM is enabled Section 8.7.3.4.

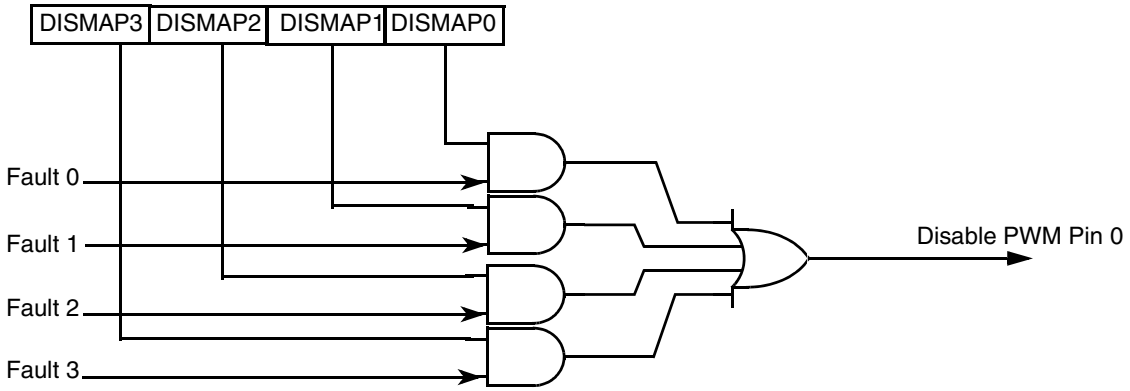


Figure 8-31. Fault Decoder for PWM 0

Table 8-4. Fault Mapping

PWM Pin	Controlling Register Bits
PWM0	DISMAP3–DISMAP0
PWM1	DISMAP7–DISMAP4
PWM2	DISMAP11 –DISMAP8
PWM3	DISMAP15–DISMAP12
PWM4	DISMAP19–DISMAP16
PWM5	DISMAP23–DISMAP20

8.6.10 Fault Pin Filter

Each fault pin has a programmable filter capable of being bypassed. The sampling period of this filter can be adjusted with the FILT_PER field of the fault filter*n* (FFILT*n)* register. The number of consecutive samples required to agree before an input transition is recognized can be adjusted using the FILT_CNT field of the same register. Setting FILT_PER to all zeros disables the input filter for a given FAULT*n* pin.

Upon detecting logic 1 on the filtered FAULT*n* pin (or logic 0 if FPOL*n* is set), the corresponding FAULT*n* pin bit (FPIN*n*) and FAULT*n* pin flag (FFLAG*n*) are set. The FPIN*n* bit remains set as long as the filtered FAULT*n* pin is set. Clear FFLAG*n* by writing logic 1 to the corresponding fault acknowledge (FTACK*n*) bit.

If the interrupt enable (FIE*n*) bit in the is set, the FFLAG*n* flag generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears the FFLAG*n* flag by writing logic 1 to the FTACK*n* bit
- Software clears the FIE*n* bit by writing logic 0 to it
- A reset occurs

8.6.11 Automatic Fault Clearing

Setting a fault mode (FMODE*n*) bit configures faults from the FAULT*n* pin for automatic clearing.

When FMODE*n* is set, disabled PWM pins are enabled when the filtered FAULT*n* pin returns to logic 0 and a new PWM half cycle begins. Please see Figure 8-32. Clearing the FFLAG*n* flag does not affect disabled PWM pins when FMODE*n* is set.

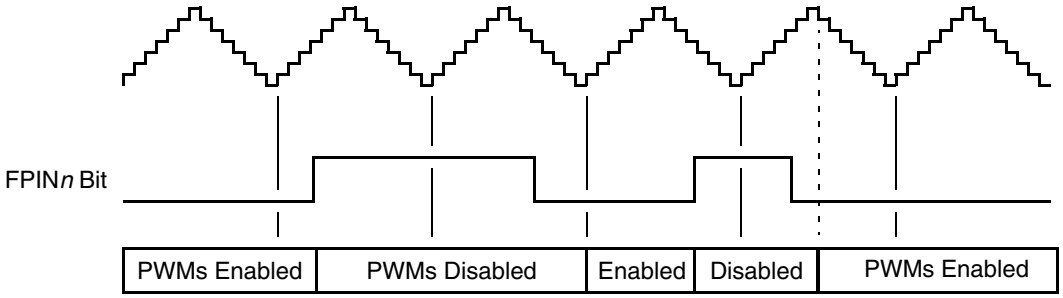


Figure 8-32. Automatic Fault Clearing

8.6.12 Manual Fault Clearing

Clearing a fault mode (FMODE*n*) bit configures faults from the FAULT*n* pin for manual clearing:

- PWM pins disabled by the FAULT0 pin or the FAULT2 pin are enabled when
  - Software clears the corresponding FFLAG*n* flag
  - The pins are enabled when the next PWM half cycle begins regardless of the logic level detected on the filtered fault pin. Please see Figure 8-33.
- PWM pins disabled by the FAULT1 pin or the FAULT3 pin are enabled when
  - Software clears the corresponding FFLAG*n* flag
  - There is a logic 0 on the filtered fault pin at the start of the next PWM half cycle boundary. Please see Figure 8-34.

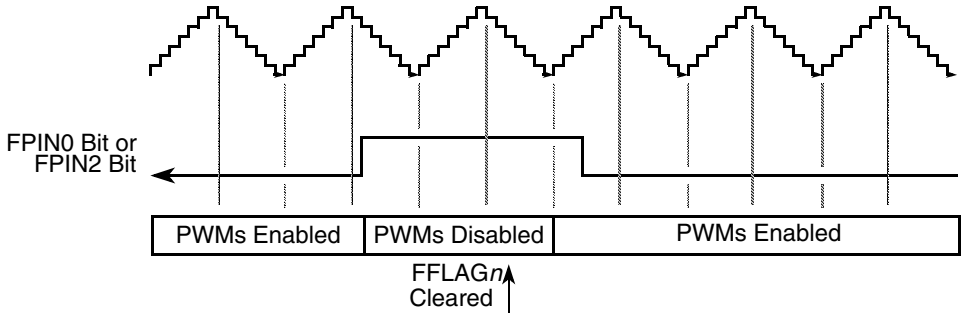


Figure 8-33. Manual Fault Clearing (Example 1)

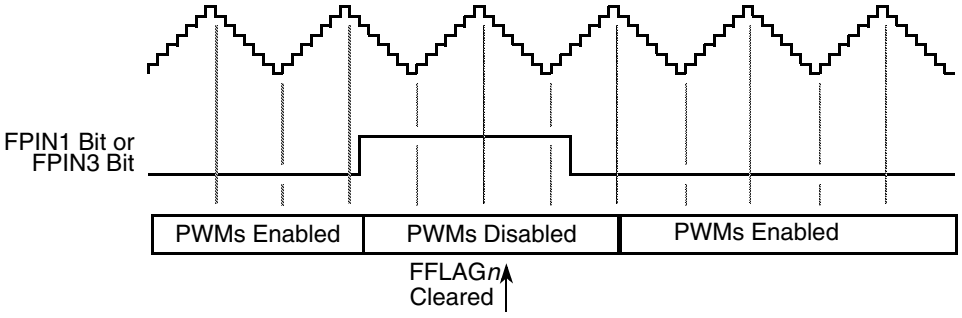


Figure 8-34. Manual Fault Clearing (Example 2)

NOTE:

PWM half-cycle boundaries occur at both the PWM cycle start and when the counter equals the modulus, so in edge-aligned operation full-cycles and half-cycles are equal.

Fault protection also applies during software output control when the OUTCTL*n* bits are set. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, PWMEN equals one. But the OUT*n* bits can control the PWM pins while the PWM generator is off, PWMEN 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.

8.6.13 External Synchronization of PWM Counting

When not being used as a fault input, the FAULT2 pin may be used for external synchronization. If SYNC_OUT_EN is set, a positive pulse is put out on FAULT2 at the start of every cycle. The polarity of this bit can be changed by the FPOL2 bit. This pulse can be used to synchronize other PWMs.

If SYNC_OUT_EN is clear and SYNC_WINDOW has a value other than zero, input synchronization is enabled. FAULT2 is the external sync input and must not be used as a fault input. The DISMAP register should be set so FAULT2 does not disable any of the PWM outputs. Filtering of the external sync input is controlled with the FFILT2 register and polarity is controlled by FPOL2. Upon recognizing an incoming pulse, the PWM counter is reset to zero if the current counter value is within the window defined by SYNC_WINDOW bit field.

8.7 Register Description

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level and the address offset is defined at the module level. PWM uses PWM_BASE plus the given offset in the figures below.

Table 8-5. PWM Memory Map

Device	Peripheral	Base Address
56F80xx	PWM	$00F0C0

Table 8-6 lists the PWM registers in ascending address order, including their acronyms and address offset of each register.

Table 8-6. PWM Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL	Control register	Read/Write	Section 8.7.1
Base + $1	FCTRL	Fault control register	Read/Write	Section 8.7.2
Base + $2	FLTAK	Fault status/acknowledge register	Read/Write	Section 8.7.3
Base + $3	OUT	Output control register	Read/Write	Section 8.7.4
Base + $4	CNTR	Counter register	Read-Only	Section 8.7.5
Base + $5	CMOD	Counter modulo register	Read/Write	Section 8.7.6
Base + $6–$B	VAL0–5	Value 0–5 registers	Read/Write	Section 8.7.7
Base + $C–$D	DTIM0–1	Deadtime 0 –1 registers	Read/Write	Section 8.7.8
Base + $E–$F	DMAP1–2	Disable mapping 1–2 registers	Read/Write	Section 8.7.9
Base + $10	CNFG	Configure register	Read/Write	Section 8.7.10
Base + $11	CCTRL	Channel control register	Read/Write	Section 8.7.11
Base + $12	PORT	Port register	Read/Write	Section 8.7.12
Base + $13	ICCTRL	Internal correction control register	Read/Write	Section 8.7.13
Base + $14	SCTRL	Source control register	Read/Write	Section 8.7.14
Base + $15	SYNC	Synchronization window register	Read-Only	Section 8.7.15
Base + $16–$19	FFILT0–3	Fault filter 0–3 registers	Read/Write	Section 8.7.16

There are 21 registers in the PWM peripheral. Each is summarized in Figure 8-35 and detailed in following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL	R	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC		PWM RIE	PWMF	0	0	LDOK	PWME N
		W																
$1	FCTRL	R	0	0	0	0	FPOL3	FPOL2	FPOL1	FPOL0	FIE3	FMOD E3	FIE2	FMOD E2	FIE1	FMOD E1	FIE0	FMOD E0
		W																
$2	FLTACK	R	FPIN3	FFLAG 3	FPIN2	FFLAG 2	FPIN1	FFLAG 1	FPIN0	FFLAG 0	0	0	0	0	0	0	0	0
		W										FTACK 3		FTACK 2		FTACK 1		FTACK 0
$3	OUT	R	PAD_ EN	0	OUT CTL5	OUT CTL4	OUT CTL3	OUT CTL2	OUT CTL1	OUT CTL0	0	0	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
		W																
$4	CNTR	R	0	CNTR														
		W																
$5	CMOD	R	0	CM														
		W																
$6-$B	VAL0-5	R	PWMVAL															
		W																
$C-	DTIM0	R	0	0	0	0	PWMDT0											
		W																
$D	DTIM1	R	0	0	0	0	PWMDT1											
		W																
$D	DMAP0	R	DISMAP0															
		W																
$E	DMAP1	R	0	0	0	0	0	0	0	0	DISMAP1							
		W																
$10	CNFG	R	0	DBG_ EN	WAIT_ EN	EDG	0	TOP NEG45	TOP NEG 23	TOP NEG 01	0	BOT NEG 45	BOT NEG 23	BOT NEG 01	INDEP 45	INDEP 23	INDEP 01	WP
		W																
$11	CCTRL	R	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP 45	SWP 23	SWP 01
		W																
$12	PORT	R	0	0	0	0	0	0	0	0	0	0	0	0	PORT			
		W																
$13	ICCTRL	R	0	0	0	0	0	0	0	0	0	0	0	0	0	ICC2	ICC1	ICC0
		W																
$14	SCTRL	R	0	0	CINV5	CINV4	CINV3	CINV2	CINV1	CINV0	0	SRC2		0	SRC1		0	SRC0
		W																
$15	SYNC	R	SYNC_ OUT_ EN	SYNC_WINDOW														
		W																
$16	FFILT0	R	0	0	0	0	0	FILT0_CNT			FILT0_PER							
		W																
$17	FFILT1	R	0	0	0	0	0	FILT1_CNT			FILT1_PER							
		W																
$18	FFILT2	R	0	0	0	0	0	FILT2_CNT			FILT2_PER							
		W																
$19	FFILT3	R	0	0	0	0	0	FILT3_CNT			FILT3_PER							
		W																

R	0	Read as 0
W		Reserved

Figure 8-35. PWM Register Map Summary

8.7.1 Control (CTRL) Register

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC		PWMRIE	PWMF	0	0	LDOK	PWMEN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-36. Control (CTRL) Register

8.7.1.1 Load Frequency (LDFQ)—Bits 15–12

These buffered read/write bits select the PWM load frequency according to Table 8-7. Reset clears the LDFQ bits, selecting loading every PWM opportunity. A PWM opportunity is determined by the half bit.

NOTE:

The LDFQ*n* bits take effect when the current load cycle is complete, regardless of the state of the load okay (LDOK) bit. Reading the LDFQ*n* bits reads the buffered values and not necessarily the values currently in effect.

Table 8-7. PWM Reload Frequency

LDFQ[3:0]	PWM Reload Frequency
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

LDFQ[3:0]	PWM Reload Frequency
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

8.7.1.2 Half Cycle Reload (HALF)—Bit 11

This read/write bit enables half-cycle reloads in center-aligned PWM mode. This bit has no effect on edge-aligned PWMs.

- 0 = Half-cycle reloads disabled
- 1 = Half-cycle reloads enabled

8.7.1.3 Current Polarity 2 (IPOL2)—Bit 10

This buffered read/write bit selects the value register for the PWM4 and PWM5 pins in top/bottom software correction. Reset clears IPOL2.

- 0 = Value register four in next PWM cycle
- 1 = Value register five in next PWM cycle

NOTE:

The IPOL*n* bits take effect at the beginning of the next load cycle, regardless of the state of the LDOK bit. Select top/bottom software correction by writing 00 or 01 to the current select bits (ISENS) in the CTRL register. Reading the IPOL*n* bits reads the buffered values and not necessarily the values currently in effect.

8.7.1.4 Current Polarity 1 (IPOL1)—Bit 9

This buffered read/write bit selects the value register for the PWM2 and PWM3 pins in top/bottom software correction. Reset clears IPOL1.

- 0 = Value register two in next PWM cycle
- 1 = Value register three in next PWM cycle

8.7.1.5 Current Polarity 0 (IPOL0)—Bit 8

This buffered read/write bit selects the value register for the PWM0 and PWM1 pins in top/bottom software correction. Reset clears IPOL0.

- 0 = Value register zero in next PWM cycle
- 1 = Value register one in next PWM cycle

8.7.1.6 Prescaler (PRSC)—Bits 7–6

These buffered read/write bits select the PWM clock frequency illustrated in Table 8-8.

Table 8-8. PWM Prescaler

PRSC[1:0]	PWM Clock Frequency
00	f_{IPBus}
01	$f_{IPBus}/2$
10	$f_{IPBus}/4$
11	$f_{IPBus}/8$

NOTE:

Reading the PRSC*n* bits reads the buffered values and not necessarily the values currently in effect. The PRSC*n* bits take effect at the beginning of the next PWM cycle and only when the LDOK bit is set.

8.7.1.7 PWM Reload Interrupt Enable (PWMRIE)—Bit 5

This read/write bit enables the PWMF flag to generate CPU interrupt requests. Reset clears PWMRIE bit.

- 0 = PWMF bit CPU interrupt requests disabled
- 1 = PWMF bit CPU interrupt requests enabled

8.7.1.8 PWM Reload Flag (PWMF)—Bit 4

This read/write flag is set at the beginning of every reload cycle regardless of the state of the LDOK bit. Clear the PWMF bit by reading CTRL register with PWMF bit set before writing logic 0 to the PWMF bit. If another reload occurs before the clearing sequence is complete, writing logic 0 to the PWMF bit has no effect. Reset clears the PWMF bit.

- 0 = No new reload cycle since last PWMF clearing
- 1 = New reload cycle since last PWMF clearing

NOTE:

Clearing PWMF bit clears pending PWMF CPU interrupt requests.

8.7.1.9 Reserved—Bits 3–2

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.1.10 Load Okay (LDOK)—Bit 1

This read/write bit loads the prescaler bits of CTRL register and the entire CMOD and VAL0–5 registers into a set of buffers. The buffered prescaler divisor, PWM counter modulus value, and PWM pulse width take effect at the next PWM reload. Set LDOK by writing logic 1 to it. LDOK is automatically cleared after the new values are loaded, or can be manually cleared before a reload by writing logic 0 to it. Reset clears LDOK.

- 0 = Do not load new modulus, prescaler, and PWM values
- 1 = Load prescaler, modulus, and PWM values

NOTE:

For proper initialization of the LDOK and PWMEN bits, see Section 8.6.8.

8.7.1.11 PWM Enable Bit (PWMEN)—Bit 0

This read/write bit enables the PWM generator and the PWM pins. When PWMEN equals zero, the PWM pins are in their inactive states unless OUTCTL*n* equals one. A reset clears PWMEN.

- 0 = PWM generator and PWM pins disabled unless OUTCTL = 1
- 1 = PWM generator and PWM pins enabled

NOTE:

For proper initialization of the LDOK and PWMEN bits, see Section 8.6.8.

8.7.2 Fault Control (FCTRL) Register

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	FPOL3	FPOL2	FPOL1	FPOL0	FIE3	FMODE3	FIE2	FMODE2	FIE1	FMODE1	FIE0	FMODE0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-37. Fault Control (FCTRL) Register

8.7.2.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.2.2 Fault*n* Polarity Control (FPOL*n*)—Bits 11, 10, 9, 8

These read/write bits control the polarity of the FAULT*n* pin inputs. A reset clears FPOL*n*. FPOL2 is also used to control the polarity of the external sync input and output.

- 0 = Logic 1 on FAULT*n* indicates a fault condition
- 1 = Logic 0 on FAULT*n* indicates a fault condition

8.7.2.3 Fault*n* Interrupt Enable (FIE*n*)—Bits 7, 5, 3, 1

These read/write bits enable CPU interrupt requests generated by the filtered FAULT*n* pin. A reset clears FIE*n*.

- 0 = FAULT*n* CPU interrupt requests disabled
- 1 = FAULT*n* CPU interrupt requests enabled

NOTE:

The fault protection circuit is independent of the FIE*n* bits and is always active. If a fault is detected, the PWM pins are disabled according to the PWM disable mapping register.

8.7.2.4 Fault*n* Clearing Mode (FMODE*n*)—Bits 6, 4, 2, 0

These read/write bits select automatic or manual clearing of FAULT*n* pin faults. A reset clears FMODE*n*.

- 0 = Manual fault clearing of FAULT*n* pin faults
- 1 = Automatic fault clearing of FAULT*n* pin faults

8.7.3 Fault Status & Acknowledge (FLTACK) Register

Base + $2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FPIN3	FFLAG3	FPIN2	FFLAG2	FPIN1	FFLAG1	FPIN0	FFLAG0	0	0	0	0	0	0	0	0
Write										FTACK3		FTACK2		FTACK1		FTACK0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-38. Fault Status and Acknowledge (FLTACK) Register

8.7.3.1 Fault*n* Pin (FPIN*n*)—Bits 15, 13, 11, 9

These read-only bits reflect the current state of the filtered FAULT*n* pins. A reset has no effect on FPIN*n*.

- 0 = Logic 0 on the FAULT*n* pin
- 1 = Logic 1 on the FAULT*n* pin

8.7.3.2 Fault*n* Flag (FFLAG*n*)—Bits 14, 12, 10, 8

These read-only bits are set within two CPU cycles after a rising edge on the filtered FAULT*n* pins. Clear FFLAG*n* by writing logic 1 to the FTACK*n* bit in this register (PMFSA). A reset clears FFLAG*n*.

- 0 = No fault on the FAULT*n* pin
- 1 = Fault on the FAULT*n* pin

8.7.3.3 Reserved Bit—Bit 7

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.3.4 Fault*n* Acknowledge (FTACK*n*)—Bits 6, 4, 2, 0

Writing logic 1 to FTACK*n* clears FFLAG*n*. Writing logic 0 has no effect. Read these bits as zero. Reset clears FTACK*n*. The fault protection is enabled even when the PWM is not enabled; therefore, a fault is latched in, requiring it to be cleared in order to prevent an interrupt when the PWM is enabled.

8.7.4 Output Control (OUT) Register

Base + $3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PAD_ EN	0	OUT CTL5	OUT CTL4	OUT CTL3	OUT CTL2	OUT CTL1	OUT CTL0	0	0	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-39. Output Control (OUT) Register

8.7.4.1 Output Pad Enable (PAD_EN)—Bit 15

The PWM outputs can be enabled or disabled by setting the PAD_EN bit. The power up default has the pads disabled. This bit does not affect the functionality of the PWM, so the PWM module can be energized with the output pads disabled. This enable is to power up with a safe default value for the PWM drivers.

- 0 = Output pads disabled
- 1 = Output pads enabled

8.7.4.2 Reserved—Bit 14

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.4.3 Output Control Enable (OUTCTL5–0)—Bits 13–8

These read/write bits enable software control of their corresponding PWM pin. When OUTCTL*n* is set, the OUT*n* bit activates and deactivates the PWM*n* output or the SRC*n* bits in the SCTRL register are used to select an alternate control of the PWM outputs. A reset clears the OUTCTL bits.

When operating the PWM in complementary mode, these bits must be switched in pairs for proper operation. That is, OUTCTL0 and OUTCTL1 must have the same value; OUTCTL2 and OUTCTL3 must have the same value; and OUTCTL4 and OUTCLT5 must have the same value.

- 0 = Software control disabled (normal PWM operation)
- 1 = Software control enabled

8.7.4.4 Reserved—Bits 7–6

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.4.5 Output Control Bits (OUT5–0)—Bits 5–0

When the corresponding OUTCTL bit is set, these read/write bits control the PWM pins, illustrated in Table 8-9.

Table 8-9. Software Output Control

OUT*n* Bit	Complementary Channel Operation	Independent Channel Operation
OUT0	1—PWM0 is active 0—PWM0 is inactive	1—PWM0 is active 0—PWM0 is inactive
OUT1	1—PWM1 is complement of PWM 0 0—PWM1 is inactive	1—PWM1 is active 0—PWM1 is inactive
OUT2	1—PWM2 is active 0—PWM2 is inactive	1—PWM2 is active 0—PWM2 is inactive
OUT3	1—PWM3 is complement of PWM 2 0—PWM3 is inactive	1—PWM3 is active 0—PWM3 is inactive
OUT4	1—PWM4 is active 0—PWM4 is inactive	1—PWM4 is active 0—PWM4 is inactive
OUT5	1—PWM5 is complement of PWM 4 0—PWM5 is inactive	1—PWM5 is active 0—PWM5 is inactive

NOTE:

OUT1, OUT3, and OUT5 must be set during complementary operation even if the SRC*n* fields of the PSRC register indicate an alternate control signal is being used in place of direct software control.

8.7.5 Counter (CNTR) Register

This read-only register displays the state of the 15-bit PWM counter.

Base + $4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	CNTR														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-40. Counter (CNTR) Register

8.7.5.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.5.2 Counter—Bits 14–0

This read-only field displays the state of the 15-bit PWM counter.

8.7.6 Counter Modulo (CMOD) Register

Base + $5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	CM														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-41. Counter Modulo (CMOD) Register

8.7.6.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.6.2 PWM Counter Modulo (PWMCM)—Bits 14–0

The 15-bit unsigned value written to this buffered, read/write register defines the PWM period in PWM clock periods. Do not write a modulus value of zero to bit 15.

NOTE:

The PWM counter modulo register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins. Reading CMOD reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

8.7.7 Value 0–5 (VAL0–5) Registers

The 16-bit signed value in these buffered, read/write registers defines the pulse width in PWM clock periods for each PWM output channel.

PWM Channel 0 Value Register (PWMVAL0)—Address: PWM_BASE + $6
PWM Channel 1 Value Register (PWMVAL1)—Address: PWM_BASE + $7
PWM Channel 2 Value Register (PWMVAL2)—Address: PWM_BASE + $8
PWM Channel 3 Value Register (PWMVAL3)—Address: PWM_BASE + $9
PWM Channel 4 Value Register (PWMVAL4)—Address: PWM_BASE + $A
PWM Channel 5 Value Register (PWMVAL5)—Address: PWM_BASE + $B

Base + $6 - $B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PWMVAL															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-42. Value 0–5 (VAL0–5) Registers

NOTE:

The value registers are buffered. The value written does not take effect until the LDOK bit is set and the next load cycle begins. Reading VAL*n* reads the value in a buffer and not necessarily the value the PWM generator is currently using.

A value less than, or equal to zero deactivates the output for the entire PWM period. A value greater than, or equal to the modulus, activates the output for the entire period. Please see Table 8-1.

NOTE:

The terms activate and deactivate refer to the high and low logic states of the PWM outputs.

8.7.8 Deadtime 0–1 (DTIM0–1) Registers

Deadtime operation is only applicable to complementary channel operation. The 12-bit values written to these write-protected registers are in terms of PWM clock cycles. Reset sets the PWM deadtime registers to a default value of 0x0FFF, selecting a deadtime of 4096-PWM clock cycles minus one IPBus clock cycle. These registers are write protected after the WP bit in the CNFG register is set. Please refer to Section 8.7.10. Reserved bits 15–12 cannot be modified. They are read as zero.

Deadtime is affected by changes to the prescaler value. The deadtime duration is determined as follows: DT = P × PWMDT – 1, where DT is deadtime, P is the prescaler value, PWMDT is the programmed value of deadtime. For example: if the prescaler is programmed for a divide-by-two and PWMDT is set to five, then P = 2 and the deadtime value is equal to DT = 2 × 5 – 1 = 9 IPBus clock cycles. A special case exists when the P = 1 and DT = PWMDT.

Base + $C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	PWMDT0											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-43. Deadtime 0 (DTIM0) Register

Base + $D	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	PWMDT1											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-44. Deadtime 1 (DTIM1) Register

The PWMDT0 field is used to control the deadtime during zero to one transitions of the PWM output and during one to zero transitions of the complementary output (assuming normal polarity). The PWMDT1 field is used to control the deadtime during one to zero transitions of the primary output and zero to one transitions of the complementary output.

8.7.9 Disable Mapping 1–2 (DISMAP1–2) Registers

These write-protected registers determine which PWM pins are disabled by the fault protection inputs, illustrated in Table 8-4. Reset sets all of the bits used in the PWM disable mapping registers. These registers are write protected after the WP bit in the CNFG register is set. Reserved bits 15–8 in the DISMAP2 register cannot be modified. The bits are read as zero.

Base + $E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DISMAP															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-45. Disable Mapping 1 (DISMAP1) Register

Base + $F	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DISMAP							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-46. Disable Mapping 2 (DISMAP2) Register

8.7.10 Configure (CNFG) Register

This write-protected register contains the configuration bits determining PWM modes of operation detailed below. This register cannot be modified after the WP bit is set.

Base + $10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	DBG_EN	WAIT_EN	EDG	0	TOP NEG 45	TOP NEG 23	TOP NEG 01	0	BOT NEG 45	BOT NEG 23	BOT NEG 01	INDEP 45	INDEP 23	INDEP 01	WP
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-47. Configure (CNFG) Register

8.7.10.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.10.2 Debug Enable (DBG_EN)—Bit 14

When set to one, the PWM continues to run while the chip is in EOnCE debug mode. If the device enters EOnCE mode and this bit is zero, the PWM outputs switch to their inactive state until EOnCE mode is exited. At that point the PWM pins resume operation as programmed in the PWM registers.

For certain types of motors, such as 3-phase AC, it is imperative this bit be left in its default state (in which the PWM is disabled in EOnCE mode). Failure to do so could result in damage to the motor. For other types of motors, such as DC motors, this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates do not occur in debug mode. Any motors requiring such updates should be disabled during EOnCE mode. If in doubt, leave this bit set to zero.

8.7.10.3 WAIT Enable (WAIT_EN)—Bit 13

When set to one, the PWM continues to run while the chip is in wait mode. In this mode, the peripheral clock continues to run; however, the core clock does not. If the device enters wait mode and this bit is zero, the PWM outputs switch to their inactive state until wait mode is exited. At that point the PWM pins resume operation as programmed in the PWM registers.

For certain types of motors, such as 3-phase AC, it is imperative 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. For other types of motors, such as DC motors, this bit might safely be set to one, enabling the PWM in wait mode. The key point is PWM parameter updates do 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.

8.7.10.4 Edge-Aligned or Center-Aligned PWMs (EDG)—Bit 12

This write-protected bit determines whether all PWM channels use edge-aligned or center-aligned waveforms.

- 0 = Center-aligned PWMs
- 1 = Edge-aligned PWMs

8.7.10.5 Reserved—Bit 11

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.10.6 Top-Side Polarity Bit (TOPNEG)—Bits 10–8

This write-protected bit determines the polarity for the top-side PWMs.

- 0 = Positive top-side polarity
- 1 = Negative top-side polarity

NOTE:

Each pair of PWM channels can be configured: channel zero to one, channel two to three, and channel four to five.

8.7.10.7 Reserved—Bit 7

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.10.8 Bottom-Side Polarity Bit (BOTNEG)—Bits 6–4

This write-protected bit determines the polarity for the bottom-side PWMs.

- 0 = Positive bottom-side polarity
- 1 = Negative bottom-side polarity

NOTE:

Each pair of PWM channels can be configured: channel zero to one, channel two to three, and channel four to five.

8.7.10.9 Independent or Complimentary Pair Operation (INDEP)—Bits 3–1

This write-protected bit determines if the motor control PWM channels are independent PWMs or complementary PWM pairs.

- 0 = Complementary PWM pair
- 1 = Independent PWMs

NOTE:

Each pair of PWM channels can be configured: channel zero to one, channel two to three, and channel four to five.

8.7.10.10 Write Protect (WP)—Bit 0

This write-protected bit enables write protection to be used for all write-protected registers. While clear, WP allows write-protected registers to be written. When set, WP prevents any further writes to write-protected registers. Once set, WP can be cleared only by a reset. Write-protected registers include SCTRL, DISMAP1-2, DTIM0 and DTIM1, CNFG, SYNC, FFILT*n*, and the ENHA bit in the CCTRL. The VLMODE, SWP0, SWP1 and SWP2 bits in the CCTRL register are protected when the ENHA bit is set to zero in the CCTRL register. ENHA is in turn protected by setting WP in the CNFG register.

- 0 = Write-protected registers can be modified by writing
- 1 = Write-protected registers are read-only

NOTE:

The write to CNFG register setting the WP bit is the last write accepted to the register until the part is reset.

8.7.11 Channel Control (CCTRL) Register

This write-protected register contains the configuration bits determining PWM modes of operation as detailed below. The ENHA bit cannot be modified after the WP bit in the CNFG register is set. ENHA bit in turn provides protection for the nBX, VLMODE, SWP45, SWP23 and SWP01 bits. The mask bits are not write protected.

Base + $11	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP 45	SWP 23	SWP 01
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-48. Channel Control (CCTRL) Register

8.7.11.1 Enable Hardware Acceleration (ENHA)—Bit 15

This bit enables writing to the nBX, VLMODE, SWP45, SWP23, and SWP01 bits. The bit is write-protected by the CNFG register's WP bit.

- 0 = Disable writing to nBX, VLMODE, SWP45, SWP23, and SWP01 bits
- 1 = Enable writing to nBX, VLMODE, SWP45, SWP23, and SWP01 bits

8.7.11.2 56F80x Compatibility (nBX)—Bit 14

This bit is used to enable/disable improved SWAP and MASK operations. In one case, SWAP/MASK operates identical to the 56F80x version of this module. In the other case, these functions were moved to an improved location in the PWM data flow. If the latter is chosen, the 56F80x compatible features are not supported. See Figure 8-2 for details.

- 0 = SWAP and MASK provide 56F80x compatible operation
- 1 = SWAP*n* and MASK*n* provide new functionality as shown in Figure 8-2.

This bit is write-protected when ENHA bit is zero.

NOTE:

This bit must be set to zero in order to use SWAP in INDEPENDENT mode.

8.7.11.3 Mask 5–0 (MSK5–0)—Bits 13–8

These six bits determine the mask for each of the PWM logical channels.

- 0 = Unmasked
- 1 = Masked, channel set to a value of zero percent duty cycle

The SWAP and MASK functions have two different modes of operation controlled by the nBX bit in the CCTRL register. Figure 8-49 is somewhat of a simplification. See Figure 8-2 and Section 8.7.11.2 for details.

8.7.11.4 Reserved—Bits 7–6

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.11.5 Value Register Load Mode (VLMODE)—Bits 5–4

These two bits determine the way the value registers are being loaded.

- 00 = Each value register is accessed independently
- 01 = Writing to value register zero also writes to value registers one to five
- 10 = Writing to value register zero also writes to value registers one to three
- 11 = Reserved

These bits are write protected when ENHA bit is zero.

8.7.11.6 Swap45 (SWAP45)—Bit 2

The SWAP and MASK functions have two different modes of operation controlled by the nBX bit in the CCTRL register. Figure 8-49 is somewhat of a simplification. See Figure 8-2 and Section 8.7.11.2 for details.

- 0 = No swap
- 1 = Channel four and channel five are swapped

This bit is write protected when ENHA bit is zero.

8.7.11.7 Swap23 (SWAP23)—Bit 1

The SWAP and MASK functions have two different modes of operation controlled by the nBX bit in the CCTRL register. Figure 8-49 is somewhat of a simplification. See Figure 8-2 and Section 8.7.11.2 for details.

- 0 = No swap
- 1 = Channel two and channel three are swapped

This bit is write protected when ENHA bit is zero.

8.7.11.8 Swap01—Bit 0

The SWAP and MASK functions have two different modes of operation controlled by the nBX bit in the CCTRL register. Figure 8-49 is somewhat of a simplification. See Figure 8-2 and Section 8.7.11.2 for details.

- 0 = No swap
- 1 = Channel zero and one are swapped

This bit is write protected when ENHA bit is zero.

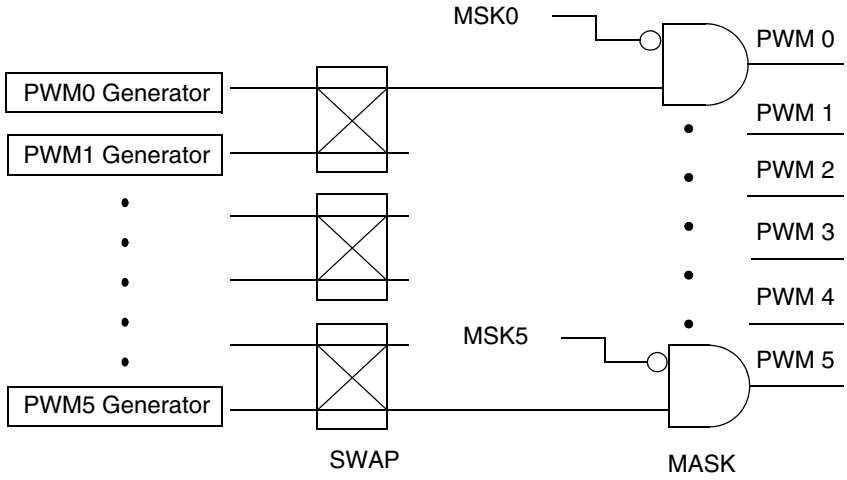


Figure 8-49. Channel Swapping

8.7.12 Port (PORT) Register

This register contains values of the four fault inputs, bits three, two, one, and zero. This is a read-only register, therefore any writes to the register do not affect it. This register may be read while the PWM is active.

Base + $12	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	PORT			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	U	U	U	U

Figure 8-50. Port (PORT) Register

8.7.13 Internal Correction Control (ICCTRL) Register

These bits only apply in center aligned operation during complementary mode.

Setting the ICCTRL register bits ICC*n*, for a PWM pair, selects the even PWM value registers to use when counting up and the odd PWM value registers when counting down. Please see Figure 8-19.

Base + $13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	ICC2	ICC1	ICC0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-51. Internal Correction Control (ICCTRL) Register

8.7.13.1 Reserved—Bits 15–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.13.2 Internal Current Control 2 (ICC2)—Bit 2

This bit controls PWM4/PWM5 pair.

- 0 = Use VAL4 register at all times.
- 1 = Use VAL4 register when the PWM counter is counting up. Use VAL5 register when counting down.

8.7.13.3 Internal Current Control 1 (ICC1)—Bit 1

This bit controls PWM2/PWM3 pair.

- 0 = Use VAL2 register at all times.
- 1 = Use VAL2 register when the PWM counter is counting up. Use VAL3 register when counting down.

8.7.13.4 Internal Current Control 0 (ICC0)—Bit 0

This bit controls PWM0/PWM1 pair.

- 0 = Use VAL0 register at all times.
- 1 = Use VAL0 register when the PWM counter is counting up. Use VAL1 register when counting down.

8.7.14 Source Control (SCTRL) Register

This register contains the control bits used to determine the signals to be applied as the source signals for the complementary PWM outputs. This register is affected by the WP bit in the CNFG register. It can only be written when that bit is clear.

Base + $14	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	CINV5	CINV4	CINV3	CINV2	CINV1	CINV0	0	SRC2		0	SRC1		0	SRC0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-52. Source Control (SCTRL) Register

8.7.14.1 Reserved—Bits 15–14

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.14.2 Compare Invert 5 (CINV5)—Bit 13

This bit controls the polarity of PWM compare output five. Please see the output operations in Figure 8-7 and Figure 8-8.

- 0 = PWM output five is high when CNTR register is less than VAL5 register
- 1 = PWM output five is high when CNTR register is greater than VAL5 register

8.7.14.3 Compare Invert 4 (CINV4)—Bit 12

This bit controls the polarity of PWM compare output four. Please see the output operations in Figure 8-7 and Figure 8-8.

- 0 = PWM output four is high when CNTR register is less than VAL4 register
- 1 = PWM output four is high when CNTR register is greater than VAL4 register

8.7.14.4 Compare Invert 3 (CINV3)—Bit 11

This bit controls the polarity of PWM compare output three. Please see the output operations in Figure 8-7 and Figure 8-8.

- 0 = PWM output three is high when CNTR register is less than VAL3 register
- 1 = PWM output three is high when CNTR register is greater than VAL3 register

8.7.14.5 Compare Invert 2 (CINV2)—Bit 10

This bit controls the polarity of PWM compare output two. Please see the output operations in Figure 8-7 and Figure 8-8.

- 0 = PWM output two is high when CNTR register is less than VAL2 register
- 1 = PWM output two is high when CNTR register is greater than VAL2 register

8.7.14.6 Compare Invert 1 (CINV1)—Bit 9

This bit controls the polarity of PWM compare output one. Please see the output operations in Figure 8-7 and Figure 8-8.

- 0 = PWM output one is high when CNTR register is less than VAL1 register
- 1 = PWM output one is high when CNTR register is greater than VAL1 register

8.7.14.7 Compare Invert 0 (CINV0)—Bit 8

This bit controls the polarity of PWM compare output zero. Please see the output operations in Figure 8-7 and Figure 8-8.

- 0 = PWM output zero is high when CNTR register is less than VAL0 register
- 1 = PWM output zero is high when CNTR register is greater than VAL0 register

8.7.14.8 Reserved—Bit 7

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.14.9 Source 2 (SRC2)—Bits 6–5

This field controls the PWM5/PWM4 pair. Make sure OUTCTL4 and OUTCTL5 (bits 12 and 13 of the OUT register) are set when using these bits.

- 00 = Use PWM generator as PWM source (operation is consistent with 56F80x and 56F83xx devices).
- 01 = Use PSRC2 input as PWM source.
- The specific signal driving this input is based on muxing controlled by the SIM IPS register. This information can be found in the device data sheet.
- 1x = Use the value selected in SCTRL0 as the PWM source.

8.7.14.10 Reserved—Bit 4

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.14.11 Source 1 (SRC1)—Bits 3–2

This field controls the PWM2/PWM3 pair. Make sure OUTCTL2 and OUTCTL3 (bits 10 and 11 of the OUT register) are set when using these bits.

- 00 = Use PWM generator as PWM source (operation is consistent with 56F80x and 56F83xx devices)
- 01 = Use PSRC1 input as PWM source
- The specific signal driving this input is based on muxing controlled by the SIM IPS register. This information can be found in the device data sheet
- 1x = Use the value selected in SRC0 as the PWM source

8.7.14.12 Reserved—Bit 1

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.14.13 Source 0 (SRC0)—Bit 0

This bit controls the PWM0/PWM1 pair. Make sure OUTCTL0 and OUTCTL1 (bits 8 and 9 of the OUT register) are set when using these bits.

- 0 = Use PWM generator as PWM source (operation is consistent with 56F80x and 56F83xx devices)
- 1 = Use PSRC0 input as PWM source
- The specific signal driving this input is based on muxing controlled by the SIM IPS register. This information can be found in the device data sheet.

8.7.15 Synchronization Window (SYNC) Register

This register is used to define the window of time during which the external sync can reset the PWM counter. This register is affected by the WP bit in the CNFG register. It can only be written when that bit is clear.

Base + $15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SYNC_OUT_EN	SYNC_WINDOW														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-53. Synchronization Window (SYNC) Register

8.7.15.1 Synchronization Output Enable (SYNC_OUT_EN)—Bit 15

This bit controls the enable for the synchronization output. Do not set this bit when the FAULT2 pin is to be used as a fault input.

- 0 = Synchronization output pulse disabled
- 1 = Synchronization output pulse enabled on FAULT2 pin

8.7.15.2 Synchronization Window (SYNC_WINDOW)—Bits 14–0

This field defines the window of opportunity for the external sync signal to reset the counter. For center aligned operation (EDG = 0) if the value of the counter is less than the value of SYNC_WINDOW bit field, the external synchronization is enabled. This means, for a SYNC_WINDOW bit field value of zero, external synchronization can never take place, that is it is disabled. For a SYNC_WINDOW bit field value of 0x7FFF, the external sync can occur at any time, that is always enabled. For edge aligned operation (EDG = 1) if the value of the counter is less than the value of the SYNC_WINDOW bit field or if the difference between the value of the counter and the counter modulo value is less than the value of SYNC_WINDOW, then external synchronization is enabled.

SYNC_WINDOW bit field should be set to zero when SYNC_OUT_EN is set to one. This enables the sync output on the FAULT2 pin and disable the sync input on that same pin.

8.7.16 Fault Filter 0–3 (FFILT0–3) Registers

These registers are used to program the characteristics of the filters for the fault inputs. These registers are affected by the WP bit in the CNFG register. They can only be written when WP bit is clear.

Base + $16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	FILT0_CNT			FILT0_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-54. Fault 0 Filter (FFILT0) Register

Base + $17	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	FILT1_CNT			FILT1_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-55. Fault 1 Filter (FFILT1) Register

Base + $18	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	FILT2_CNT			FILT2_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-56. Fault 2 Filter (FFILT2) Register

Base + $19	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	FILT3_CNT			FILT3_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-57. Fault 3 Filter (FFILT3) Register

8.7.16.1 Reserved—Bits 15–11

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

8.7.16.2 Input Filter Sample Count (FILT*n*_CNT)—Bits 10–8

These bits represent the number of consecutive samples required to agree prior to the input filter accepting an input transition. A value of 0x0 represents three samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency as described in Section 8.7.16.4.

8.7.16.3 Input Filter Sample Period (FILT*n*_PER)—Bits 7–0

These bits represent the sampling period, in IPBus clock cycles, of the fault input signals. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), the input filter is bypassed. The value of FILT_PER affects the input latency as described in Section 8.7.16.4.

8.7.16.4 Input Filter Considerations

The FILT_PER value should be set such so the sampling period is larger than the period of the expected noise. This way a noise spike only corrupts one sample. The FILT_CNT 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 FILT_CNT + 3 power.

The values of FILT_PER and FILT_CNT must also be traded off 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) × FILT_PER + 2) IPBus clock periods. Even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also ensuring fault response if the PWM module loses its clock. The latency induced by the filter is seen in the time to set the FFLAG and FPIN bits of the FLTACK register.

8.8 Clocks

The system IPBus clock is the only clock required by this module. It determines, along with the PWM prescaler, the amount of time allocated for a single PWM bit value.

8.9 Interrupts

Five PWM sources can generate CPU interrupt requests:

- Reload flag (PWMF)—PWMF is set at the beginning of every reload cycle. The reload interrupt enable bit (PWMRIE) enables PWMF to generate CPU interrupt requests. PWMF and PWMRIE are in CTRL register.
- Fault flags (FFLAG0–FFLAG3—The FFLAG*n* bit is set when logic 1 occurs on the FAULT*n* pin. The fault pin interrupt enable bits (FIE0–FIE3) enable the FFLAG*n* flags to generate CPU interrupt requests. FFLAG0–FFLAG3 are in the FLTACK register. FIE0–FIE3 are in the FCTRL register.

8.10 Resets

All PWM registers are reset to their default values upon any system reset.

Table 8-10. Document Revision History for Chapter 8

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 9
Multi-Scalable Controller Area Network (MSCAN)

9.1 Introduction

The controller area network (CAN) definition is based on the SCAN12 definition, the specific implementation of the CAN concept is 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. A recommendation to fully understand the CAN specification is to first read the Bosch specification to become familiar with the terms and concepts contained within this document.

Though not exclusively intended for automotive applications, CAN protocol is designed to meet specific requirements of a vehicle serial data bus, that is real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

CAN uses an advanced buffer arrangement resulting in predictable real-time behavior and simplified application software.

9.2 Features

The basic CAN qualities are:

- Implementation of the CAN protocol–Version 2.0 A/B
- Standard and extended data frames
- 0-to-8 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
- Programmable loopback mode supports self-test operation

1. Depending on the actual bit timing and the clock jitter of the PLL.

- 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, such as warning, error passive, bus-off
- Programmable CAN 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
  - CAN enable
- Global initialization of configuration registers

9.3 Block Diagram

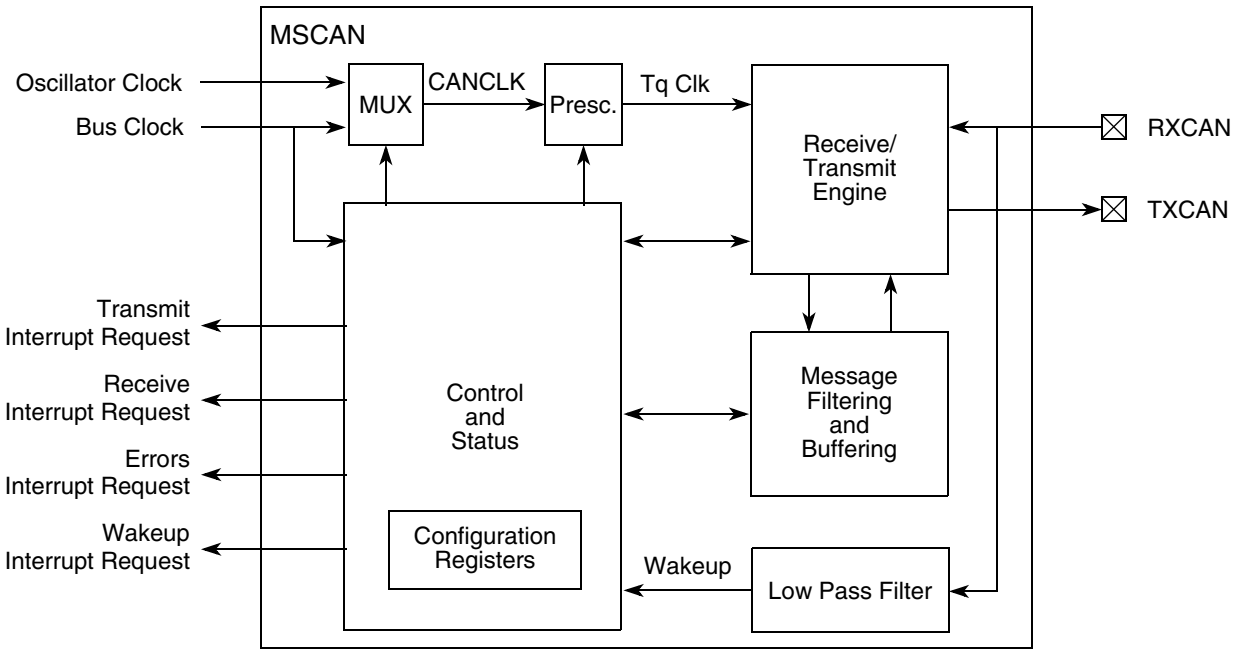


Figure 9-1. CAN Block Diagram

9.4 Typical CAN System

A typical CAN system with MSCAN is illustrated in Figure 9-2. Each CAN station is physically connected to the CAN bus lines through a transceiver device. The transceiver is capable of driving the large current required for the CAN bus; it has current protection against defective CAN or defective stations.

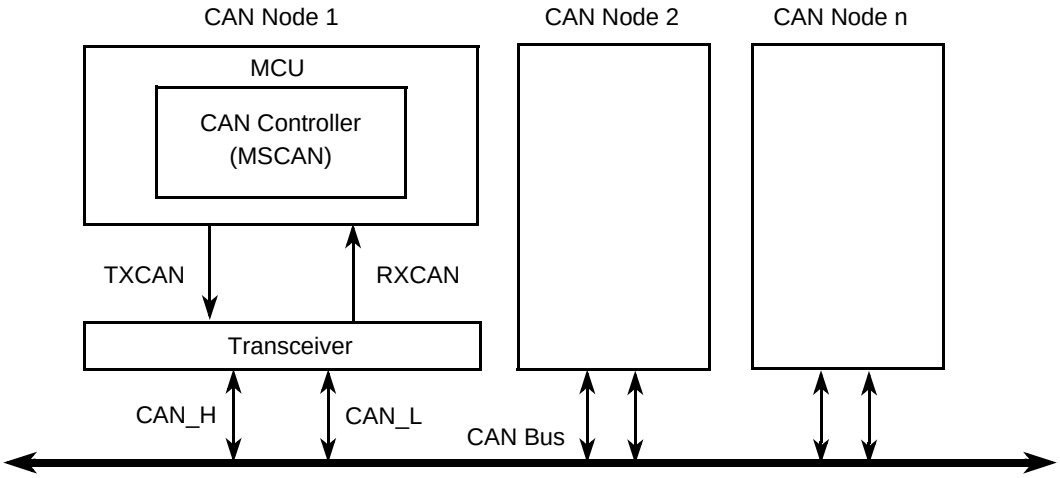


Figure 9-2. Typical CAN System

9.5 Functional Description

This section provides a complete functional description of the MSCAN. It describes each of the features and modes listed in the introduction.

9.5.1 Message Storage

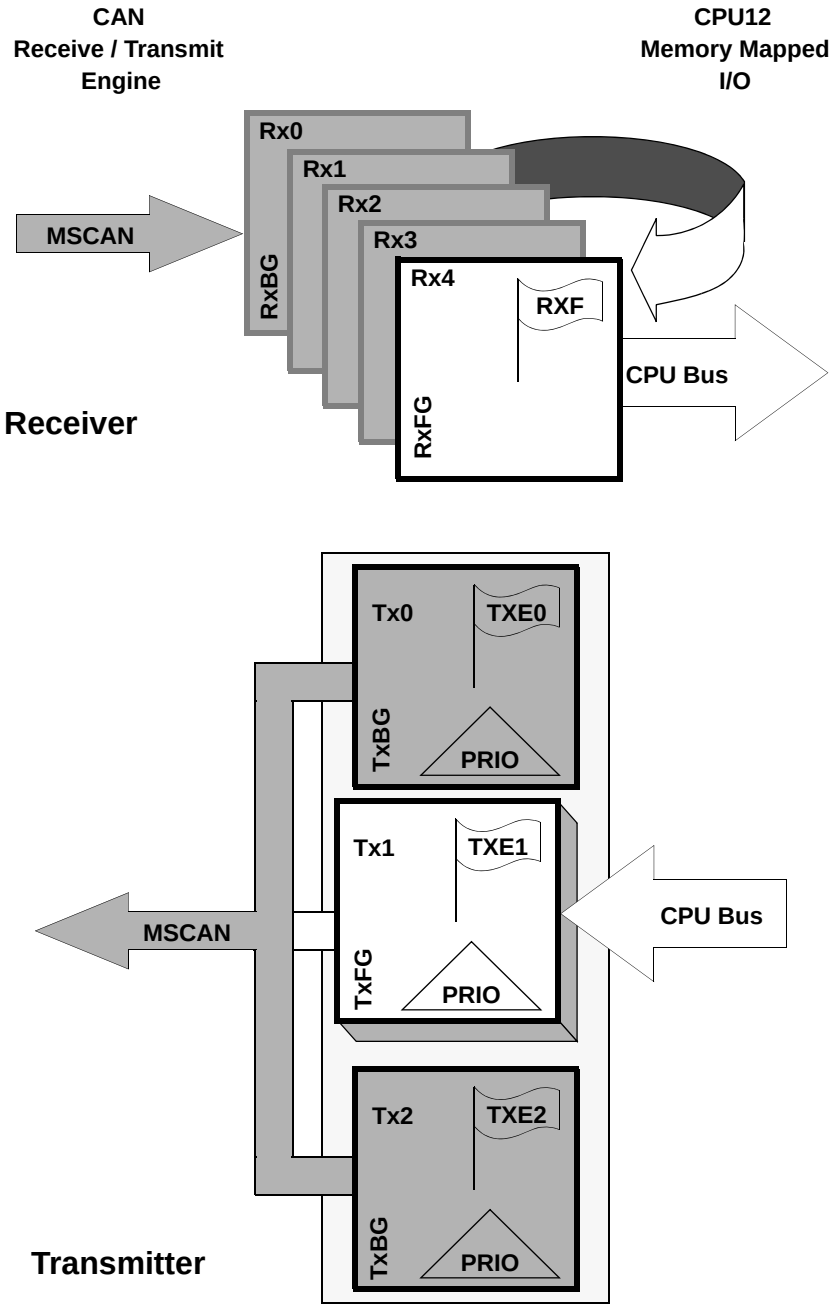


Figure 9-3. User Model for Message Buffer Organization

CAN facilitates a sophisticated message storage system addressing the requirements of a broad range of network applications.

9.5.1.1 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 only release the CAN bus in case of lost arbitration.
- The internal message queued within any CAN node is organized such, 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 the CPU to react with short latencies to the transmit interrupt.

A double buffer scheme disconnects the reloading of the transmit buffer from the actual message sending, therefore, reducing the response requirements of the CPU. Problems can arise if a sent message is completed while the CPU re-loads the second buffer. There would be no buffer ready for transmission, releasing the CAN bus.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances.

The second requirement calls for some sort of internal prioritization. The CAN implements the above two requirements with three transmit buffers and a local priority concept described in Section 9.5.1.2.

9.5.1.2 Transmit Structures

The CAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be created in advance. The three buffers are arranged as shown in Figure 9-3.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers. Please see Section 9.9.18. Additionally, Section 9.9.30 contains an 8-bit local priority (PRIO) field. The remaining two bytes are stamp message time if required. Please see Section 9.9.31.

To transmit a message, the CPU must identify an available transmit buffer, indicated by a set transmitter buffer empty (TXE*n*) bit; see Section 9.9.7. If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the transmit buffer selection (TBSEL) register. Please refer to Section 9.9.11. This makes the respective buffer accessible within the transmit buffer (TXFG) registers address space; see Section 9.9.18. The algorithmic feature associated with the TBSEL register simplifies the transmit buffer selection. Additionally, this scheme makes the handler software simpler because only one address area is applicable for the transmit process, minimizing the required address space.

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 TXE bit.

The CAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE bit. A transmit interrupt discussed in Section 9.10.2 is generated[1] when TXE*n* is set, It can be used to drive the application software to reload the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the CAN 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 (PRIO) field. The application software programs this field when the message is created. The local priority reflects the preference 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 CAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXE*n* also.

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 already in transmission cannot be aborted, it is necessary to request an interrupt by setting the corresponding abort request (ABTRQ) bit. Please see Section 9.9.9. The request is granted by CAN if possible, by:

1. Setting the corresponding abort acknowledge (ABTAK) bit in the transmitter message abort acknowledge (TAAK) register
2. Setting the associated transmit buffer empty (TXE) bit to release the buffer
3. Generating a transmit interrupt using the transmit interrupt handler software capable to determine from the setting of the ABTAK bit whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0)

9.5.1.3 Receive Structures

Received messages are stored in a five-stage input FIFO. The five message buffers are alternately mapped into a single memory area, illustrated in Figure 9-3. The background receive buffer (RXBG) is exclusively associated with the CAN, but the foreground receive buffer (RXFG) is addressable by the CPU, illustrated in Figure 9-3. 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 and discussed in Section 9.9.18.

The receiver full (RXF) bit, discussed in Section 9.9.5, signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this bit is set.

On reception, each message is checked to see whether it passes the filter. This is discussed in Section 9.5.2 and is simultaneously written into the active background receiver buffer (RXBG). After successful reception of a valid message, the CAN shifts the content of RXBG into the receiver FIFO[1], sets the RXF bit, and generates a receive interrupt to the CPU[2]. Please see Section 9.10.3. The receive handler must read the received message from the RXFG. The receive handler then resets the RXF bit, acknowledging the interrupt before it releases 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 CAN receives an invalid message in its RXBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be overwritten by the next message. The buffer will then not be shifted into the FIFO.

When the CAN module is transmitting, it 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 CAN treats its own messages exactly like all other incoming messages. Please refer to Section 9.9.2. The CAN receives its own transmitted messages in the event it loses arbitration. If arbitration is lost, the CAN must be prepared to become a receiver.

1. Only if the RXF bit is not set.
2. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

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 CANBUS with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled. Please also see Section 9.10.5. The CAN remains able to transmit messages while the receiver FIFO is being filled, discarding all incoming messages. As soon as a receive buffer in the FIFO is available again, new valid messages are accepted.

9.5.2 Identifier Acceptance Filter

The CAN identifier acceptance registers, see additional discussion in Section 9.9.12, 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 CAN identifier mask registers. Section 9.9.17 provides additional discussion.

A filter hit is indicated to the application software by a set receive buffer full (RXF = 1) bit and three bits in the IDAC register. Please refer to Section 9.9.16. These identifier hit bits (IDHIT[2:0]) clearly identify the filter section causing the acceptance. They simplify the task of application software 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 was introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes. See Bosch CAN 2.0 A/B protocol specification for further details:

- 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 2.0 A/B messages[1]. This mode implements two filters for a full length CAN 2.0 B compliant extended identifier. Figure 9-4 illustrates how the first 32-bit filter bank (IDAR0–IDAR3, IDMR0–IDMR3) produces a filter zero hit. Similarly, the second filter bank (IDAR4–IDAR7, IDMR4–IDMR7) produces a filter one hit.
- Four identifier acceptance filters, each to be applied to:
  - a) the 14 most significant bits (MSBs) of the extended identifier plus the SRR and IDE bits of CAN 2.0 B messages, or
  - b) the 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0 A/B messages. Figure 9-5 illustrates how the first 32-bit filter bank (IDAR0–IDA3, IDMR0–IDMR3) produces filter zero and one hits. Similarly, the second filter bank (IDAR4–IDAR7, IDMR4–IDMR7) produces filter two and three 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 2.0 A/B compliant standard identifier, or a CAN 2.0 B compliant extended identifier. Figure 9-6 delineates how the first 32-bit filter bank (IDAR0–IDAR3, IDMR0–IDMR3) produces filter zero to three hits. Similarly, the second filter bank (IDAR4–IDAR7, IDMR4–IDMR7) produces filter four to seven hits.
- Closed filter. No CAN message is copied into the foreground buffer RXFG, and the RXF bit is never set.

1. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters for standard identifiers

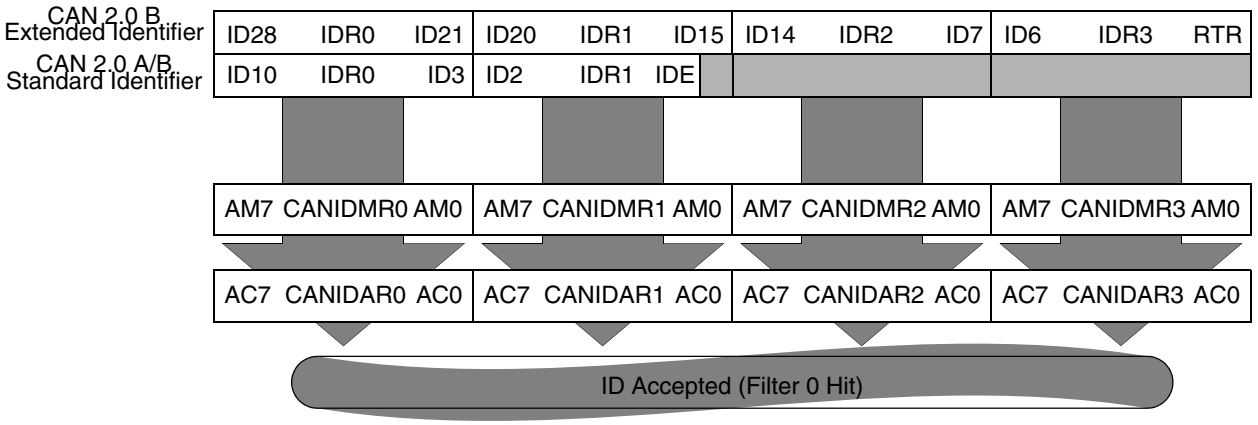


Figure 9-4. 32-Bit Maskable Identifier Acceptance Filter

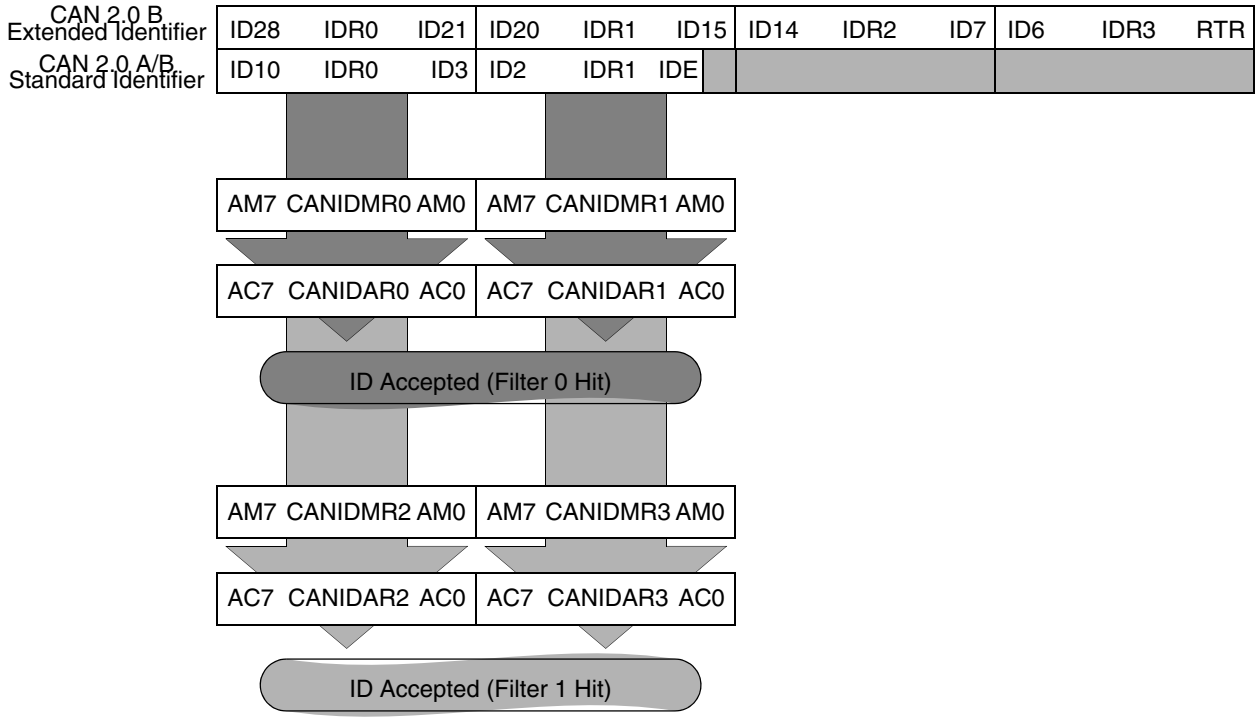


Figure 9-5. 16-Bit Maskable Identifier Acceptance Filters

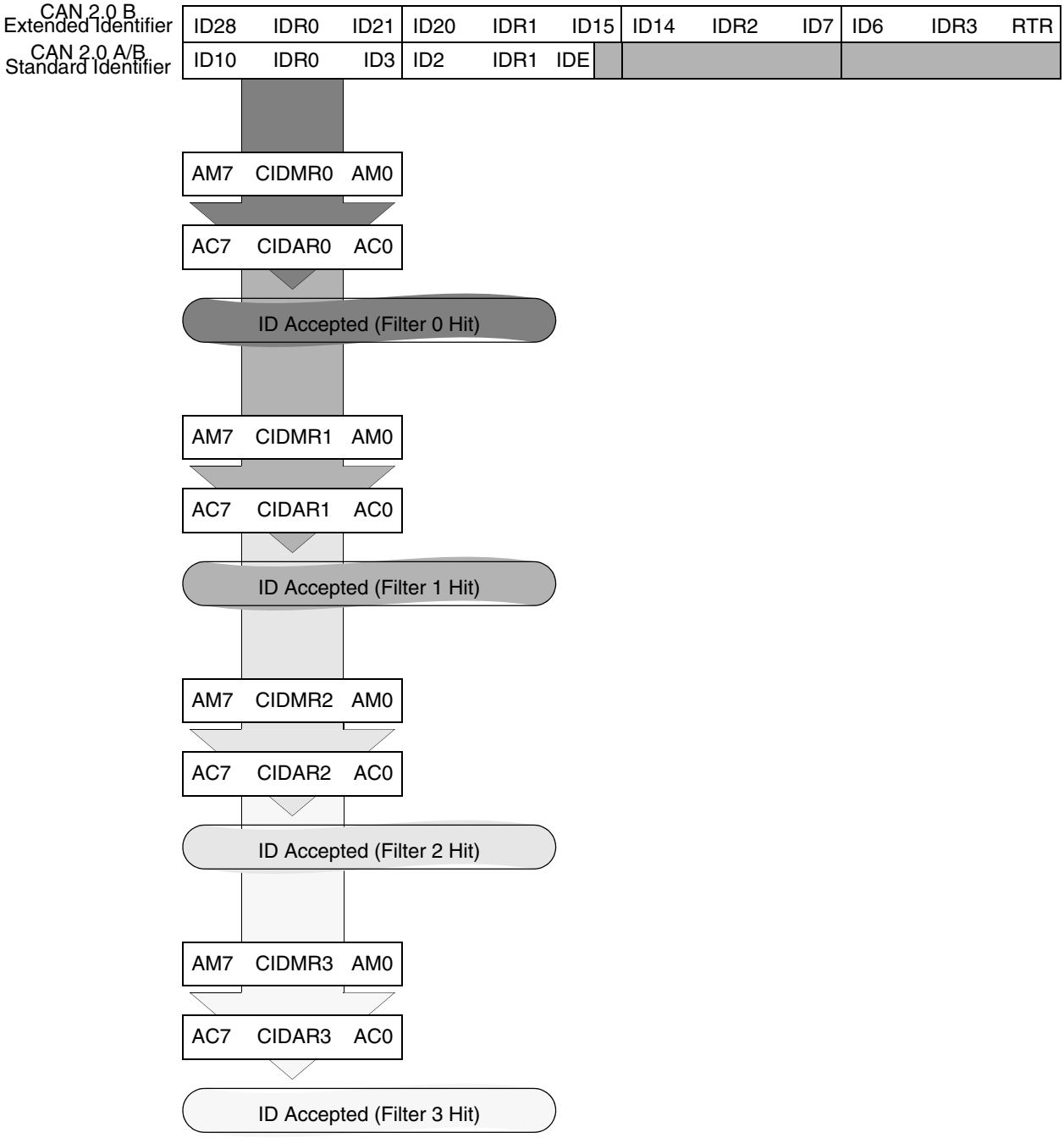


Figure 9-6. 8-Bit Maskable Identifier Acceptance Filters

9.5.2.1 Protocol Violation Protection

The MSCAN protects 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 configuration controlling registers of the CAN cannot be modified while the MSCAN is on-line. The CAN has to be in initialization mode. The corresponding INITRQ/INITAK handshake bits in the CTRL0/CTRL1 registers (see Section 9.9.1) serve as a lock to protect the following registers:

— CAN control 1 (CTRL1) register
— CAN bus timing registers 0 and 1 (BTR0, BTR1)
— CAN identifier acceptance control (IDAC) register
— CAN identifier acceptance registers (IDAR0–IDAR7)
— CAN identifier mask registers (IDMR0–IDMR7)

- The TXCAN pin is immediately forced to a recessive state when the CAN goes into either the power down or initialization modes. Please see Section 9.7.6.6 and Section 9.7.6.5.
- The CAN enable bit (CANE) is writable only once in normal system operation modes, providing further protection against inadvertently disabling the CAN.

9.5.2.2 Clock System

Figure 9-7 shows the structure of the CAN clock generation circuitry.

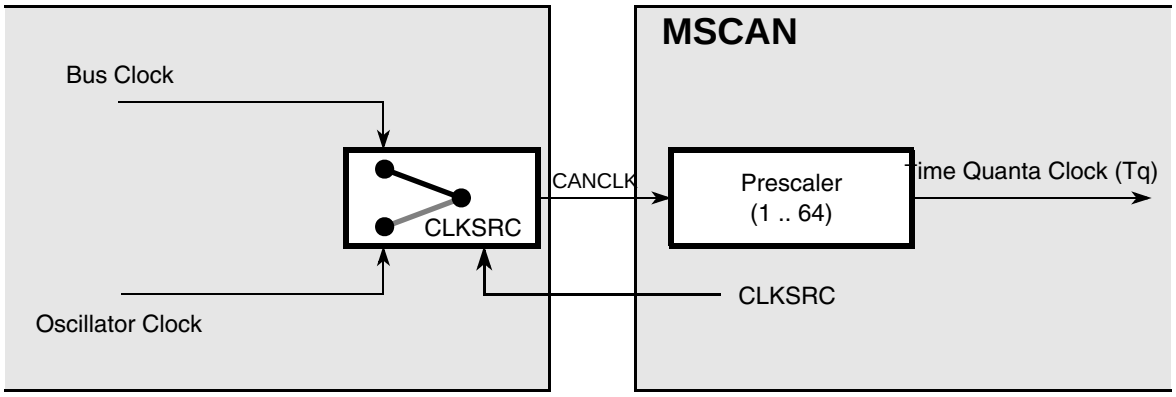


Figure 9-7. MSCAN Clocking Scheme

The clock source (CLKSRC) bit in the CTRL1 register defines whether the internal CLK is connected to the output of a crystal oscillator (oscillator clock) or to the bus clock. Please see Section 9.9.1.

The clock source has to be chosen to meet the tight oscillator tolerance requirements (up to 0.4 %) of the CAN protocol. 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 (T_q) clock from CLK. A time quantum is the atomic unit of time handled by the CAN.

$$f_{Tq} = \frac{f_{CLK}}{(\text{Prescaler Value})} \qquad \textit{Eqn. 9-1}$$

A bit time is subdivided into three segments as described in the Bosch CAN specification. Please see Figure 9-8:

- 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 four to 16 times 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 two to eight times quanta long.

Eqn. 9-2

$$Bit\ Rate = \frac{f_{Tq}}{(Number\ of\ Time\ Quantra)}$$

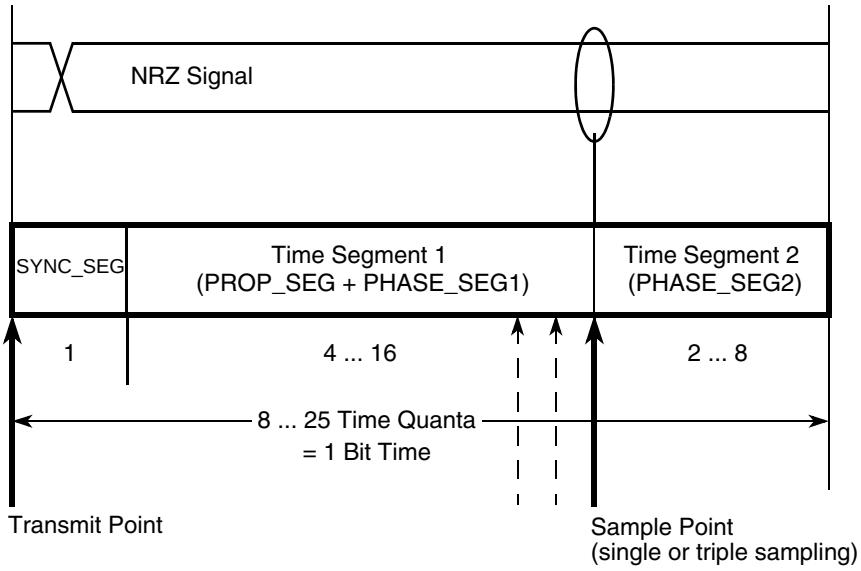


Figure 9-8. Segments Within the Bit Time

Table 9-1. 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 can be programmed in a range of one to four times quanta by setting the SJW parameter. See the Bosch CAN specification for details.

The SYNC_SEG, TSEG1, TSEG2, and SJW parameters are set by programming the CAN bus timing registers (BTR0, BTR1). Please refer to Section 9.9.3 and Section 9.9.4.

Table 9-2 provides an overview of the CAN compliant segment settings and the related parameter values.

NOTE:

It is necessary to ensure the bit time settings are in compliance with the CAN standard.

Table 9-2. 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
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

9.5.3 Timer Link

The CAN generates an internal time stamp whenever a valid frame is received or transmitted and the TIME bit is enabled. Because the CAN specification defines a frame to be valid if no errors occur before the end of frame (EOF) field is transmitted successfully, the actual value of an internal timer is written at EOF to the appropriate time stamp position within the transmit buffer. For receive frames, the time stamp is written to the receive buffer.

9.6 Initialization/Application Information

9.6.1 CAN Initialization

The procedure to initially start-up the CAN 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, writable in initialization mode, needs to be changed only when the CAN module is in normal mode:

1. Bring the module into sleep mode by setting SLPRQ and await 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

9.6.2 Bus-Off Recovery

The bus-off recovery is manually configured. The bus-off state can either be left automatically or on user request.

For reasons of backwards compatibility, the CAN defaults to automatic recovery after reset. In this case, the CAN becomes 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 CAN is configured for user request, BORM set in Section 9.9.2, 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 Section 9.9.13 is cleared by the user

These two events may occur in any order.

9.7 Operation Modes

The following modes of operation are specific to the CAN. See Section 9.5 for details.

9.7.1 Normal Modes

The CAN module behaves as described within this specification in all normal system operation modes.

9.7.2 Special Modes

The CAN module behaves as described within this specification in all special system operation modes.

9.7.3 Emulation Modes

In all emulation modes, the CAN module behaves just like normal system operation modes described within this document.

9.7.4 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. Further, it cannot start a transmision. If the MAC sub-layer is required to send a *dominant* bit (ACK bit, overload bit, or active error bit), the bit is rerouted internally so the MAC sub-layer monitors this *dominant* bit, although the CAN bus may remain in recessive state externally.

9.7.5 Security Modes

The CAN module has no security features.

9.7.6 Low-Power Options

If the CAN is disabled (CANE = 0), its clocks are stopped to save power.

If the CAN is enabled (CANE = 1), the CAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down modes. 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.

Table 9-3 summarizes the combinations of CAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

For all modes, a CAN wakeup interrupt can occur only if the CAN is in sleep mode (SLPRQ = 1 and SLPAK = 1), wakeup functionality is enabled (WUPE = 1), and the wakeup interrupt is enabled (WUPIE = 1).

Table 9-3. CPU vs. CAN 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		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			CSWAI = X SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = X SLPAK = X

1. X means don't care.

9.7.6.1 Operation in Run Mode

As shown in Table 9-3 above, only CAN sleep mode is available as low power option when the CPU is in run mode.

9.7.6.2 Operation in Wait Mode

The WAI instruction puts the MCU in a low power consumption standby 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 CAN restarts its internal controllers and enters normal mode again.

While the CPU is in wait mode, the CAN can be operated in normal mode and generate interrupts (registers can be accessed via background debug mode). The CAN can also operate in any of the low-power modes depending on the values of the SLPRQ/SLPAK and CSWAI bits illustrated in Table 9-3.

9.7.6.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption standby mode. In stop mode, the CAN is set in power down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits Table 9-3.

9.7.6.4 CAN Sleep Mode

The CPU can request the CAN to enter this low power mode by asserting the SLPRQ bit in the CTRL0 register. The time when the CAN enters sleep mode depends on a fixed synchronization delay and its current activity:

- If there are one or more message buffers scheduled for transmission (TXE*n* = 0), the CAN continues to transmit until all transmit message buffers are empty (TXE*n* = 1, transmitted successfully or aborted) and then goes into sleep mode
- If the CAN is receiving, it continues to receive and goes into sleep mode as soon as the CAN bus next becomes idle
- If the CAN is neither transmitting nor receiving, it immediately goes into sleep mode

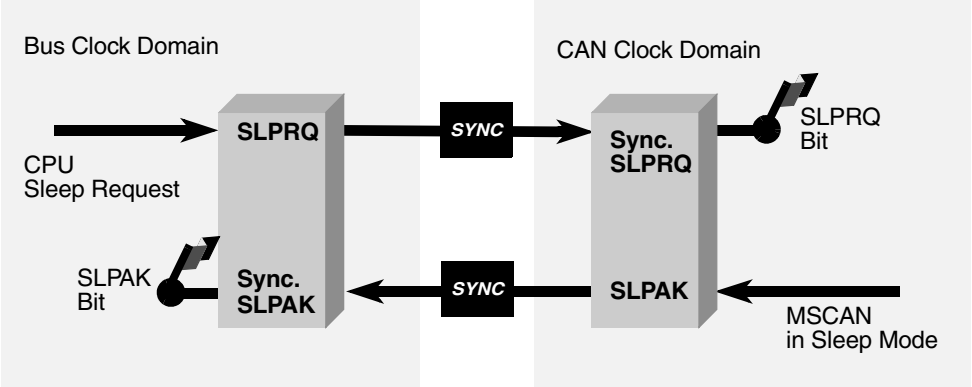


Figure 9-9. Sleep Request/Acknowledge Cycle

NOTE:

The application software must avoid setting up a transmission (by clearing one or more TXE*n* bit(s)) and immediately request sleep mode (by setting SLPRQ). Whether the CAN 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. See Figure 9-9. 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 CAN stops its internal clocks. However, clocks allowing register accesses from the CPU side continue to run.

If the CAN 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 TXE bits. No message abort takes place while in sleep mode.

If the WUPE bit in CTRL0 is not asserted, the CAN masks any activity it detects on CAN. The RXCAN pin is therefore, held internally in a recessive state. This locks the CAN in sleep mode, illustrated in Figure 9-10. For sleep mode to take effect, the WUPE bit must be set before entering sleep mode.

The CAN is able to leave sleep mode (wakeup) 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) are active.

After wakeup, the CAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, if the CAN is awakened by a CAN frame, this frame is not received.

The receive message buffers (RXFG and RXBG) contain messages only if the messages were received before sleep mode was entered. All pending actions are executed upon wakeup; copying of RXBG into RXFG, message aborts and message transmissions. If the CAN remains in bus-off state after sleep mode was exited, it continues counting the 128 occurrences of 11 consecutive recessive bits.

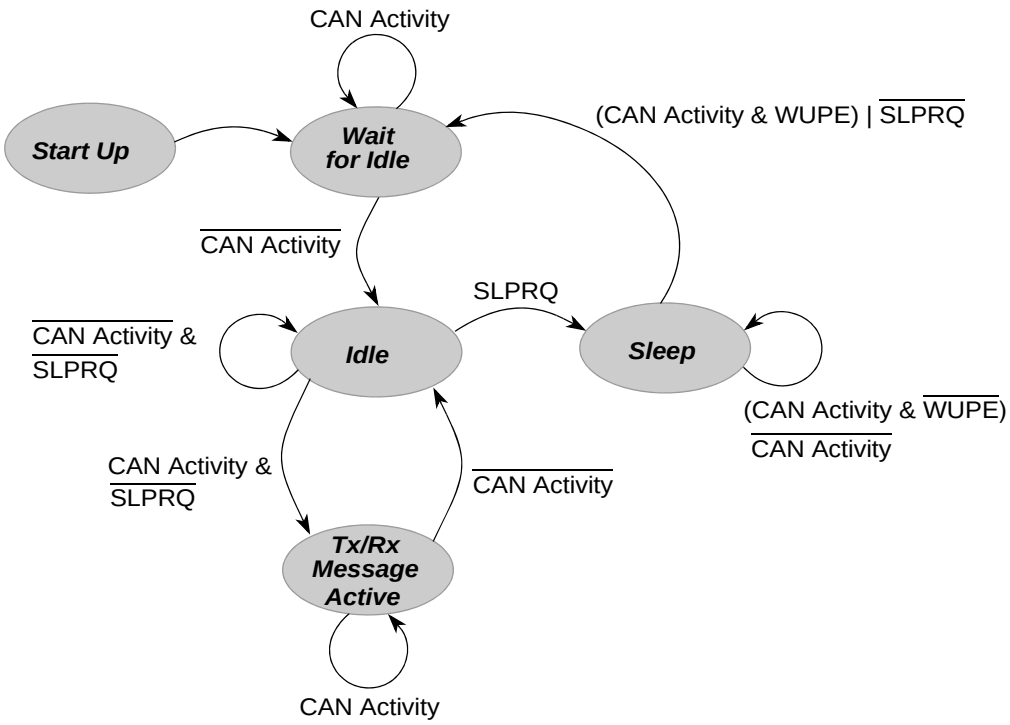


Figure 9-10. Simplified State Transitions for Entering/Leaving Sleep Mode

9.7.6.5 CAN 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 CAN immediately drives the TXCAN pin into a recessive state.

NOTE:

Ensure the CAN is not active when initialization mode is entered. The recommended procedure is to bring the CAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before setting the INITRQ bit in the CTRL0 register. Otherwise, the abort of an on-going message can cause an error condition, impacting other CAN

bus devices.

In initialization mode, the CAN is stopped. However, interface registers remain accessible. This mode is used to reset the CTRL0, RFLG, RIER, TFLG, TIER, TARQ, TAAK, and TBSEL registers to their default values. In addition, the CAN enables the configuration of the BTR0, BTR1 bit timing registers; IDAC; and the IDAR, IDMR message filters. See Section 9.9.1 for a detailed description of the Initialization mode.

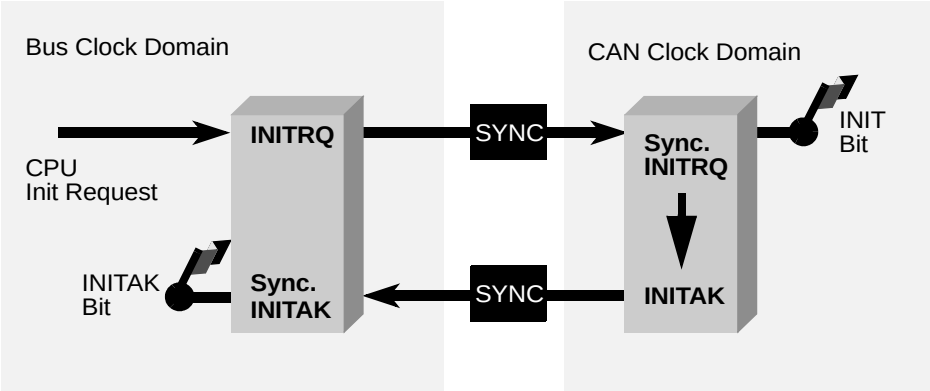


Figure 9-11. Initialization Request/Acknowledge Cycle

Due to independent clock domains within the CAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay, illustrated above in Figure 9-11.

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 CAN are in Initialization mode, the INITAK bit 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) are active.

9.7.6.6 CAN Power Down Mode

The CAN is in power down mode (Table 9-3) 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 CAN 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 CAN immediately drives the TXCAN pin into a recessive state.

NOTE:

Ensure the MCAN is not active when power down mode is entered. The recommended procedure is to bring the CAN 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 CAN 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.

9.7.6.7 Programmable Wakeup Function

The CAN can be programmed to wakeup the CAN as soon as CAN bus activity is detected. Please see control bit WUPE in Section 9.9.1. 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. Please see control bit WUPM in Section 9.9.2.

This feature can be used to protect the CAN from wakeup due to short glitches on the CAN bus lines. Such glitches can result from—for example—electromagnetic interference within noisy environments.

9.8 External Signal Descriptions

The CAN uses two external pins:

- Receiver input (RXCAN)
- Transmitter output (TXCAN)–this terminal represents the logic level on the CAN bus:
  - 0 = Dominant state
  - 1 = Recessive state

9.9 Register Description

This section details all of the registers in the CAN module. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams in bit order. All bits of all registers in this module are completely synchronous to internal clocks during a register read. 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. The address offset is defined at the module level

The CAN occupies 64 bytes in the memory space. The base address of the CAN module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset.

Table 9-5 lists the CAN registers in ascending address order, including their acronyms and address offset of each register.

Table 9-4. MSCAN Memory Map

Device	Peripheral	Base Address
56F80xx	MSCAN	$00F800

Table 9-5. CAN Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL0	Control 0 register	Read/Write[1]	Section 9.9.1
Base + $1	CTRL1	Control 1 register	Read/Write[1]	Section 9.9.2
Base + $2	BTR0	Bus timing 0 register	Read/Write	Section 9.9.3

Table 9-5. CAN Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $3	BTR1	Bus timing 1 register	Read/Write	Section 9.9.4
Base + $4	RFLG	Receiver flag register	Read/Write[1]	Section 9.9.5
Base + $5	RIER	Receiver interrupt enable register	Read/Write	Section 9.9.6
Base + $6	TFLG	Transmitter flag register	Read/Write[1]	Section 9.9.7
Base + $7	TIER	Transmitter interrupt enable register	Read/Write	Section 9.9.8
Base + $8	TARQ	Transmitter message abort request register	Read/Write[1]	Section 9.9.9
Base + $9	TAAK	Transmitter message abort acknowledge register	Read-Only	Section 9.9.10
Base + $A	TBSEL	Transmit buffer selection register	Read/Write[1]	Section 9.9.11
Base + $B	IDAC	Identifier acceptance control register	Read/Write[1]	Section 9.9.12
Base + $C		Reserved		
Base + $D	MISC	Miscellaneous register	Read/Write[1]	Section 9.9.13
Base + $E	RXERR	Receiver error counter register	Read-Only	Section 9.9.14
Base + $F	TXERR	Transmit error counter register	Read-Only	Section 9.9.15
Base + $10 – $13	IDAR0–3	Identifier acceptance 0–3 registers	Read/Write	Section 9.9.16
Base + $14 – $11	IDMR0–3	Identifier mask 0–3 registers	Read/Write	Section 9.9.17
Base + $18 – $1B	IDAR4-7	Identifier acceptance 4–7 registers	Read/Write	Section 9.9.16
Base + $1C – $1F	IDMR4–7	Identifier acceptance 4–7 registers	Read/Write	Section 9.9.17
Base + $20 – $2F	RXFG	Foreground receive buffer registers	Read-Only[2]as x	Section 9.9.18
Base + $30 – $3F	TXFG	Foreground transmit buffer registers	Read[2]/Write as x	Section 9.9.18

1. Refer to detailed register description for write access restrictions on per bit basis.
2. Reserved bits and unused bits within the TX and RX buffer (TXFG, RXFG) will be read as "x" because of RAM-based implementation.

The registers in the CAN peripheral are summarized in Figure 9-12. Each of the registers is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL0	R									RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
		W																
$1	CTRL1	R									CANE	CLK SRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
		W																
$2	BTR0	R									SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		W																
$3	BTR1	R									SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		W																
$4	RFLG	R									WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
		W																
$5	RIER	R									WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
		W																
$6	TFLG	R									0	0	0	0	0	TXE2	TXE1	TXE0
		W																
$7	TIER	R									0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
		W																
$8	TARQ	R									0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
		W																
$9	TAAK	R									0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
		W																
$A	TBSEL	R									0	0	0	0	0	TX2	TX1	TX0
		W																
$B	IDAC	R									0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
		W																
$C	Reserved	R																
		W																
$D	MISC	R									0	0	0	0	0	0	0	BO HOLD
		W																
$E	RXERR	R									RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		W																
$F	TXERR	R									TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		W																
$10 - $13	IDAR0-3	R									AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W																
$14 - $17	IDMR0-3	R									AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W																
$18 - $1B	IDAR4-7	R									AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W																
$1C - $F	IDMR4-7	R									AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W																
$20 - $2F	RXFG*n*	R	Please refer to Table 9-12 and Figure 9-1															
		W																
$30 - $3F	TXFG*n*	R	Please refer to Table 9-12 and Figure 9-1															
		W																

R	0	Read as 0
W		Reserved

Figure 9-12. CAN Register Map Summary

9.9.1 Control 0 (CTRL0) Register

This register is read at anytime. Write anytime when it is out of initialization mode. Exceptions are read-only RXACT, SYNCH, RXFRM (set by module only), and INITRQ (also without write restriction in initialization mode).

Base + $0	7	6	5	4	3	2	1	0
Read	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-13. Control 0 (CTRL0) Register

NOTE:

The CTRL0 register, except bits 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).

9.9.1.1 Received Frame Flag (RXFRM)—Bit 7

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 being set, it remains set until cleared by software or reset. Clearing is achieved by writing one. Writing zero is ignored. This bit is not valid in loopback mode. CAN must be in normal mode for this bit to be set.

- 0 = No valid message was received since last clearing this bit
- 1 = A valid message was received since last clearing of this bit

9.9.1.2 Receiver Active Status (RXACT)—Bit 6

This read-only bit indicates the CAN is receiving a message. The bit is controlled by the receiver front end. This bit is not valid in loopback mode.

- 0 = CAN is transmitting or idle[1]
- 1 = CAN is receiving a message, including when arbitration is lost[1]

9.9.1.3 CAN Stops in Wait Mode (CSWAI)—Bit 5

Enabling this bit allows for lower power consumption in wait mode by disabling all the clocks at the CPU bus interface to the CAN module. In order to protect from accidentally violating the CAN protocol, the TXCAN terminal is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode. Please see Section 9.7.6.2 and Section 9.7.6.3.

- 0 = The module is not affected during wait mode
- 1 = The module ceases to be clocked during wait mode

9.9.1.4 Synchronized Status (SYNCH)—Bit 4

This read-only bit indicates if CAN is synchronized to the CAN bus and is able to participate in the communication process. It is set and cleared by the CAN.

- 0 = CAN is not synchronized to the CAN bus
- 1 = CAN is synchronized to the CAN bus

1. See the Bosch CAN 2.0 A/B specification for a detailed definition of transmitter and receiver states.

9.9.1.5 Timer Enable (TIME)—Bit 3

This bit activates an internal 16-bit wide free running timer clocked by the bit clock rate. If the timer is enabled, a 16-bit time stamp is assigned to each transmitted/received message within the active TX/RX buffer. As soon as a message is acknowledged on the CAN bus, the time stamp is written to the highest bytes (0x000E, 0x000F) in the appropriate buffer. Please see Section 9.9.18. The internal timer is reset (all bits set to zero) when disabled. This bit is held low in initialization mode.

- 0 = Disable internal CAN timer
- 1 = Enable internal CAN timer

9.9.1.6 Wakeup Enable (WUPE)—Bit 2

This configuration bit allows the CAN to restart from sleep mode when traffic on CAN is detected. Please see Section 9.7.6.4. If the WUPE bit in CTRL0 is not asserted, the MSCAN masks any activity it detects on CAN. The RXCAN bit is therefore held internally in a recessive state. This locks the MSCAN in sleep mode, illustrated in Figure 9-10. For sleep mode to take effect, the WUPE bit must be set before entering sleep mode.

The CPU has to be certain the WUPE bit in the CTRL0 register and WUPIE bit in the RIER register are enabled, if the recovery mechanism from stop or wait is required. Please see Section 9.9.6.

- 0 = wakeup disabled – The CAN ignores traffic on CAN
- 1 = wakeup enabled – The CAN is able to restart

9.9.1.7 Sleep Mode Request (SLPRQ)—Bit 1

This bit requests the CAN to enter sleep mode, an internal power saving mode. Please see Section 9.7.6.4. 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. For further information see Section 9.9.2. Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the CAN detects activity on the CAN bus, clearing SLPRQ itself. The CPU cannot clear SLPRQ before the CAN has entered sleep mode (SLPRQ = 1 and SLPAK = 1). The SLPR bit cannot be set while the WUPIF bit is set.

- 0 = Running – The CAN functions normally
- 1 = Sleep mode request – The CAN enters sleep mode when CAN bus idle

9.9.1.8 Initialization Mode Request (INITRQ)—Bit 0

When this bit is set by the CPU, the CAN skips to initialization mode. Please refer to Section 9.7.6.5. 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. See Section 9.9.2.

The following registers enter their hard reset state and restore their default values: CTRL0[1], RFLG[2], RIER[3], TFLG, TIER, TARQ, TAAK, and TBSEL.

Registers CTRL1, BTR0, BTR1, IDAC, IDAR0-7, and IDMR0-7 can only be written by the CPU when the CAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode.

1. Not including WUPE, INITRQ, and SLPRQ
2. TSTAT1 and TSTAT0 are not affected by initialization mode
3. RSTAT1 and RSTAT0 are not affected by initialization mode

When this bit is cleared by the CPU, the CAN restarts and then attempts to synchronize to the CAN bus. If the CAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the CAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits.

Writing to otherbits in CTRL0, RFLG, RIER, TFLG, or TIER registers must be completed only after initialization mode is exited (INITRQ = 0 and INITAK = 0).

The CPU cannot clear INITRQ before the CAN enters initialization mode (INITRQ = 1 and INITAK = 1). In order to protect from accidentally violating the CAN protocol, the TXCAN terminal is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the CAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before requesting initialization mode.

- 0 = Normal operation
- 1 = CAN in initialization mode

9.9.2 Control 1 (CTRL1) Register

The CTRL1 register provides various control bits and handshake status information of the CAN module. It is read anytime while modification by writing is possible anytime when INITRQ = 1 and INITAK = 1, except CANE, which is write once in normal and anytime in special system operation modes when the CAN is in initialization mode (INITRQ = 1 and INITAK = 1).

Base + $1	7	6	5	4	3	2	1	0
Read	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
Write								
Reset	0	0	0	1	0	0	0	1

Figure 9-14. Control 1 (CTRL1) Register

9.9.2.1 Enable (CANE)—Bit 7

- 0 = CAN module is disabled
- 1 = CAN module is enabled

9.9.2.2 Clock Source (CLKSRC)—Bit 6

This bit defines the clock source for the CAN module only for systems with a clock generation module; Section 9.5.2.2, and Figure 9-7.

- 0 = CAN clock source is the oscillator clock
- 1 = CAN clock source is the bus clock

9.9.2.3 Loop Back Self Test Mode (LOOPB)—Bit 5

When this bit is set, the CAN performs an internal loop back, used for self test operation. The bit stream output of the transmitter is fed back to the receiver internally. The RXCAN input terminal is ignored and the TXCAN output goes to the recessive state (logic 1). The CAN behaves as it does normally when transmitting, treating its own transmitted message as a message received from a remote node. In this state, the CAN 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

9.9.2.4 Listen-Only Mode (LISTEN)—Bit 4

This bit configures the CAN as a CAN bus monitor. When LISTEN bit is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out. Further information is available in Section 9.7.4. Additionally, the error counters are frozen. Listen-only mode supports applications requiring hot plugging, or throughput analysis. The CAN is unable to transmit any messages when listen-only mode is active.

- 0 = Normal operation
- 1 = Listen-only mode activated

9.9.2.5 Bus-Off Recovery Mode (BORM)—Bit 3

This bits configures the bus-off state recovery mode of the CAN. Refer to Section 9.6.2 for details.

- 0 = Automatic bus-off recovery (see Bosch CAN 2.0 A/B protocol specification)
- 1 = Bus-off recovery upon user request

9.9.2.6 Wakeup Mode (WUPM)—Bit 2

If WUPE bit in the CTRL0 register is enabled, it defines whether the integrated low-pass filter is applied to protect the CAN from spurious wakeup. See Section 9.7.6.4.

- 0 = CAN wakes up the CPU on any dominant level on the CAN bus
- 1 = CAN wakes up the CPU only in case of a dominant pulse on the CAN bus having a length of T_{WUP}

9.9.2.7 Sleep Mode Acknowledge (SLPAK)—Bit 1

This bit indicates whether the CAN module has entered sleep mode. Section 9.7.6.4 provides additional information. SLPAK is used as a handshake bit for the SLPRQ sleep mode request. Sleep mode is active when SLPRQ = 1 and SLPAK = 1. Depending on the setting of WUPE, the CAN will clear the bit if it detects activity on the CAN bus while in sleep mode.

- 0 = Running – The CAN operates normally
- 1 = Sleep mode active – The CAN has entered sleep mode

9.9.2.8 Initialization Mode Acknowledge (INITAK)—Bit 0

This bit indicates if the CAN module is in initialization mode. Section 9.7.6.5 provides additional data. INITAK is used as a handshake bit for the INITRQ Initialization mode request. Initialization mode is active when INITRQ = 1 and INITAK = 1. The registers CTRL1, BTR0, BTR1, IDAC, IDAR0–IDAR7, and IDMR0–IDMR7 can be written only by the CPU when the CAN is in initialization mode.

- 0 = Running – The CAN operates normally
- 1 = Initialization mode active – The CAN has entered initialization mode

9.9.3 Bus Timing 0 (BTR0) Register

The BTR0 register configures various CAN bus timing parameters of the CAN module. It is read anytime. Write anytime in Initialization mode (INITRQ = 1 and INITAK = 1).

Base + $2	7	6	5	4	3	2	1	0
Read	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-15. Bus Timing 0 (BTR0) Register

9.9.3.1 Synchronization Jump Width 1 and 0 (SJW1–0)—Bits 7–6

The synchronization jump width defines the maximum number of time quanta (T_q) clock cycles a bit can be shortened or lengthened to achieve resynchronization to data transitions on the CAN bus. Table 9-6 provides additional information.

Table 9-6. Synchronization Jump Width

SJW1	SJW0	Synchronization Jump Width
0	0	1 T_q clock cycle
0	1	2 T_q clock cycles
1	0	3 T_q clock cycles
1	1	4 T_q clock cycles

9.9.3.2 Baud Rate Prescaler (BRP5–0)—Bits 5–0

These bits determine the time quanta (T_q) clock used to build-up the timing. Please see Table 9-7.

Table 9-7. 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

9.9.4 Bus Timing 1 (BTR1) Register

The BTR1 register configures various CAN bus timing parameters of the CAN module. It is read anytime register and can receive writing anytime in initialization mode (INITRQ = 1 and INITAK = 1).

Base + $3	7	6	5	4	3	2	1	0
Read Write	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
Reset	0	0	0	0	0	0	0	0

Figure 9-16. Bus Timing 1 (BTR1) Register

9.9.4.1 Sampling (SAMP)—Bit 7

This bit determines the number of CAN bus samples taken per bit time. If SAMP = 0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP = 1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended only one sample is taken per bit time (SAMP = 0).

- 0 = One sample per bit
- 1 = Three samples per bit (In this case, PHASE_SEG1 must be at least two times quanta (T_q)).

9.9.4.2 Time Segment 2 (TSEG2)—Bits 6–4

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. Please see Figure 9-8. Time segment 2 (TSEG2) values are programmable, illustrated in Table 9-8.

9.9.4.3 Time Segment 1 (TSEG1)—Bits 3–0

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. Please see Figure 9-8. Time segment 1 (TSEG1) values are programmable, illustrated in Table 9-9.

Table 9-8. Time Segment 2 Values

TSEG22	TSEG21	TSEG20	Time Segment 2
0	0	0	1 T_q clock cycle[1]
0	0	1	2 T_q clock cycles
:	:	:	:
1	1	0	7 T_q clock cycles
1	1	1	8 $T_q$clock cycles

1. This setting is not valid. Please refer to Table 9-2 for valid settings.

Table 9-9. Time Segment 1 Values

TSEG13	TSEG12	TSEG11	TSEG10	Time segment 1
0	0	0	0	1 T_q clock cycle[1]
0	0	0	1	2 T_q clock cycles[1]
0	0	1	0	3 T_q clock cycles[1]
0	0	1	1	4 T_q clock cycles
:	:	:	:	:
1	1	1	0	15 T_q clock cycles
1	1	1	1	16 T_q clock cycles

1. This setting is not valid. Please refer to Table 9-2 for valid settings.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (T_q) clock cycles per bit, provided in Table 9-8 and Table 9-9.

$$\text{Bit Time} = \frac{(\text{Prescaler Value})}{f_{CANCLK}} \times (1 + \text{Time Segment1} + \text{Time Segment2})$$

9.9.5 Receiver Flag (RFLG) Register

A bit can be cleared only by software by writing one to the corresponding bit position when the condition causing the setting is no longer valid. Every bit has an associated interrupt enable bit in the RIER register.

This bit can be read at anytime It is able to receive writing anytime it is out of initialization mode, except RSTAT[1:0] and TSTAT[1:0] bits. These bits are read-only; write one clears bit; write zero is ignored.

Base + $4	7	6	5	4	3	2	1	0
Read	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-17. Receiver Flag (RFLG) Register

NOTE:

The RFLG register is held in the reset state[1] when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register receives writing when the initialization mode is exited (INITRQ = 0 and INITAK = 0).

9.9.5.1 Wakeup Interrupt Flag (WUPIF)—Bit 7

If the CAN detects CAN bus activity while in sleep mode and WUPE = 1 bit in CTRL0 register, the module will set WUPIF. See both Section 9.7.6.4 and Section 9.9.1 for additional information. If not masked, a wakeup interrupt is pending while this bit is set.

1. The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

- 0 = No wakeup activity observed while in sleep mode
- 1 = SCAN detected activity on the CAN bus and requested wakeup

9.9.5.2 CAN Status Change Interrupt Flag (CSCIF)—Bit 6

This bit is set when the CAN 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, split into separate sections for TEC/REC, informs the system on the actual CAN bus status. Additional data is available in Section 9.9.6. If not masked, an error interrupt is pending while this bit is set. CSCIF bit provides a blocking interrupt. That block guarantees 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 bit is asserted, causing an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF bit interrupt is cleared again.

- 0 = No change in CAN bus status occurred since last interrupt
- 1 = CAN changed current CAN bus status

9.9.5.3 Receiver Status Bits (RSTAT)—Bits 5–4

The values of the error counters control the actual CAN bus status of the CAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the SCAN. The coding for the bits RSTAT1, RSTAT0 is:

- 00 = RXOK: 0 ≤ receive error counter ≤ 96
- 01 = RXWRN: 96 < receive error counter ≤ 127
- 10 = RXERR: 127 < receive error counter
- 11 = Bus-Off[1]: transmit error counter > 255

9.9.5.4 Transmitter Status Bits (TSTAT)—Bits 3–2

The values of the error counters control the actual CAN bus status of the CAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the CAN. The coding for the bits TSTAT1, TSTAT0 is:

- 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

9.9.5.5 Overrun Interrupt Flag (OVRIF)—Bit 1

This bit is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this bit is set.

- 0 = No data overrun condition
- 1 = A data overrun detected

1. 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[1:0] coding in this register.

9.9.5.6 Receive Buffer Full Flag (RXF)—Bit 0

RXF is set by the CAN when a new message is shifted in the receiver FIFO. This bit 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 reads that message from the RXFG buffer in the receiver FIFO, the RXF [1] bit must be cleared to release the buffer. A set RXF bit prohibits shifting of the next FIFO entry into the foreground buffer (RXFG). If not masked, a receive interrupt is pending while this bit is set.

- 0 = No new message available within the RXFG
- 1 = The receiver FIFO is not empty. A new message is available in the RXFG

9.9.6 Receiver Interrupt Enable (RIER) Register

This register contains the interrupt enable bits for the interrupt bits described in the RFLG register. It is read at anytime and can be modified by writing when not in initialization mode.

Base + $5	7	6	5	4	3	2	1	0
Read	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-18. Receiver Interrupt Enable (RIER) Register

NOTE:

The 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).

9.9.6.1 Wakeup Interrupt Enable (WUPIE)—Bit 7

Both the WUPIE and WUPE bits must be enabled if the recovery mechanism from stop or wait is required. Please refer to Section 9.9.1.

- 0 = No interrupt request is generated from this event
- 1 = A wakeup event causes a wakeup interrupt request

9.9.6.2 CAN Status Change Interrupt Enable (CSCIE)—Bit 6

- 0 = No interrupt request is generated from this event
- 1 = A CAN status change event causes an error interrupt request

9.9.6.3 Receiver Status Change Enable (RSTATE)—Bits 5–4

These RSTATE enable bits control the sensitivity level in the receiver state changes are causing CSCIF interrupts. independent of the chosen sensitivity level the RSTATE bits continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending.

1. To ensure data integrity, do not read the receive buffer registers while the RXF bit is cleared. For MCUs with dual CPUs, reading the receive buffer registers while the RXF bit is cleared may result in a CPU fault condition.

- 00 = Do not generate any CSCIF interrupt caused by receiver state changes.
- 01 = Generate CSCIF interrupt only if the receiver enters or leaves bus-off state. Discard other receiver state changes for generating CSCIF interrupt.
- 10 = Generate CSCIF interrupt only if the receiver enters or leaves RXERR or bus-off state. Discard other receiver state changes for generating CSCIF interrupt.

  Bus-off state is defined by the CAN standard (see Bosch CAN 2.0 A/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 RXSTAT[1:0] bits define an additional bus-off state for the receiver. Please see Section 9.9.5.
- 11 = Generate CSCIF interrupt on all state changes.

9.9.6.4 Transmitter Status Change Enable (TSTATE)—Bits 3–2

These TSTATE enable bits control the sensitivity level in the transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTATE bits 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.

9.9.6.5 Overrun Interrupt Enable (OVRIE)—Bit 1

- 0 = No interrupt request is generated from this event
- 1 = An overrun event causes an error interrupt request

9.9.6.6 Receiver Full Interrupt Enable (RXFIE)—Bit 0

- 0 = No interrupt request is generated from this event
- 1 = A receive buffer full (successful message reception) event causes a receiver interrupt request

9.9.7 Transmitter Flag (TFLG) Register

The transmit buffer empty bits each have an associated interrupt enable bit in the TIER. It is read at anytime and can be modified by writing anytime for TXE*n* bits when not in initialization mode.Write one clears the bits, while writing zero is ignored.

Base + $6	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	TXE2	TXE1	TXE0
Write								
Reset	0	0	0	0	0	1	1	1

Figure 9-19. Transmitter Flag (TFLG) Register

NOTE:

The 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).

9.9.7.1 Reserved—Bits 7–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.7.2 Transmitter Buffer Empty (TXE)—Bits 2–0

This bit field indicates the associated transmit message buffer is empty and thus not scheduled for transmission. The CPU must clear the bit after a message is set up in the transmit buffer and is due for transmission. The CAN sets the bit after the message is sent successfully. The bit is also set by the CAN when the transmission request is successfully aborted due to a pending abort request. More data is available in Section 9.9.9. If not masked, a transmit interrupt is pending while this bit is set.

Clearing a TXE*n* bit also clears the corresponding ABTAK*n*. Additional data is available in Section 9.9.9. When a TXE*n* bit is set, the corresponding ABTRQ*n* bit is cleared.

When listen mode is active, the TXE*n* bits cannot be cleared and no transmission is started. Please see Section 9.9.2.

Read and write accesses to the transmit buffer is blocked if the corresponding TXE*n* bit is cleared (TXE*n* = 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)

9.9.8 Transmitter Interrupt Enable Register (TIER)

This register contains the interrupt enable bits for the transmit buffer empty Interrupt bits. It is read at anytime and receives writing anytime when not in initialization mode.

Base + $7	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-20. Transmitter Interrupt Enable Register (TIER)

NOTE:

The TIER register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register receives writing when not in initialization mode (INITRQ = 0 and INITAK = 0).

9.9.8.1 Reserved—Bits 7–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.8.2 Transmitter Empty Interrupt Enable (TEIE)—Bits 2–0

- 0 = An interrupt request is not generated from this event.
- 1 = A transmitter empty (transmit buffer available for transmission) event causes a transmitter empty interrupt request.

9.9.9 Transmitter Message Abort Request (TARQ) Register

The TARQ register allows abort request of queued messages, described below. It is read and can receive writing at anytime when not in Initialization mode.

Base + $8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-21. Transmitter Message Abort Request (TARQ) Register

NOTE:

The CANTARQ 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).

9.9.9.1 Reserved—Bits 7–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.9.2 Abort Request (ABTRQ)—Bits 2–0

The CPU sets the ABTRQ*n* bit to request a scheduled message buffer (TXE*n* = 0) be aborted. The CAN grants the request if the message is not already in transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE (see Section 9.9.7) and abort acknowledge (ABTAK) bits are set and a transmit interrupt occurs if enabled. The CPU cannot reset ABTRQ*n*. ABTRQ*n* is reset whenever the associated TXE bit is set. Please see Section 9.9.10.

- 0 = No abort request
- 1 = Abort request pending

9.9.10 Transmitter Message Abort Acknowledge (TAAK) Register

The TAAK register indicates the successful abort of a queued message, if requested by the appropriate bits in the TARQ register. It is read at anytime and can not be modified by writing for ABTAK*n* bits.

Base + $9	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-22. Transmitter Message Abort Acknowledge (TAAK) Register

NOTE:

The TAAK register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1).

9.9.10.1 Reserved—Bits 7–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.10.2 Abort Acknowledge (ABTAK)—Bits 2–0

This bit acknowledges a message was aborted due to a pending abort request from the CPU. After a particular message buffer is flagged empty, this bit can be used by the application software to identify whether the message was aborted successfully or was sent anyway. The ABTAK*n* bit is cleared whenever the corresponding TXE bit is cleared.

- 0 = The message was not aborted
- 1 = The message was aborted

9.9.11 Transmit Buffer Selection (TBSEL) Register

The TBSEL register allows the selection of the actual transmit message buffer, accessible in the TXFG register space. It is read at the lowest ordered bit set to one. All other bits are read as zero. Write to this bit at anytime when not in initialization mode.

Base + $A	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	TX2	TX1	TX0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-23. Transmit Buffer Selection (TBSEL) Register

NOTE:

The 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).

9.9.11.1 Reserved—Bits 7–3

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.11.2 Transmit Buffer Select (TX)—Bits 2–0

The lowest numbered bit places the respective transmit buffer in the TXFG register space, that is, 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 are blocked if the corresponding TXE*n* bit is cleared and the buffer is scheduled for transmission. Please see Section 9.9.7.

- 0 = The associated message buffer is deselected
- 1 = The associated message buffer is selected, if lowest numbered bit

A brief programming example of the TBSEL register usage is provided here:

To get the next available transmit buffer, application software must read the TFLG register and write this value back into the TBSEL register. In this example TX buffers TX1 and TX2 are available. The value read from TFLG is therefore $06. When writing this value back to TBSEL, the TX buffer (TX1) is selected in the TXFG register blocks because the lowest numbered bit set to one is at bit position one. Reading back this value out of CANTBSEL results in $02, because only the lowest numbered bit position set to one is presented. This mechanism eases the application software the selection of the next available TX buffer.

If all transmit message buffers are deselected, accesses are not allowed to the TXFG register block.

9.9.12 Identifier Acceptance Control (IDAC) Register

The IDAC register is used for identifier acceptance control, described below. It is read at anytime and can receive writing at anytime in initialization mode (INITRQ = 1 and INITAK = 1), except bits IDHIT*n*, which are read-only.

Base + $B	7	6	5	4	3	2	1	0
Read	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-24. Identifier Acceptance Control (IDAC) Register

9.9.12.1 Reserved—Bits 7–6

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.12.2 Identifier Acceptance Mode (IDAM)—Bits 5–4

The CPU sets these bits to define the identifier acceptance filter organization. Please see Section 9.5.2 while Table 9-10 summarizes the different settings. In filter closed mode, no message is accepted, therefore the foreground buffer is never reloaded.

Table 9-10. Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Two 32-bit acceptance filters
0	1	Four 16-bit acceptance filters
1	0	Eight 8-bit acceptance filters
1	1	Filter closed

9.9.12.3 Reserved—Bit 3

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.12.4 Identifier Acceptance Hit Indicator (IDHIT)—Bits 2–0

The CAN sets these bits to indicate an identifier acceptance hit. Please see Section 9.5.2 while Table 9-11 summarizes the different settings.

Table 9-11. Identifier Acceptance Hit Indication

IDHIT2	IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	0	Filter 0 hit
0	0	1	Filter 1 hit
0	1	0	Filter 2 hit
0	1	1	Filter 3 hit
1	0	0	Filter 4 hit
1	0	1	Filter 5 hit
1	1	0	Filter 6 hit
1	1	1	Filter 7 hit

The IDHIT*n* indicators are always related to the message in the foreground buffer (RXFG). When a message gets shifted into the foreground buffer of the receiver FIFO the indicators are updated as well.

9.9.13 Miscellaneous (MISC) Register

This register provides additional features. This is a read/write register. Writing one clears bits; writing zero is ignored.

Base + $D	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	BOHOLD
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-25. Miscellaneous (MISC) Register

9.9.13.1 Reserved—Bits 7–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

9.9.13.2 Bus-off State Hold Until User Request (BOHOLD)—Bit 0

If BORM bit in CTRL1 register is set, the BHOLD bit indicates whether the module has entered the bus-off state. Please refer to Section 9.9.2 for details concerning the BORM bit. Clearing this bit requests the recovery from bus-off. Refer to Section 9.6.2 for details.

- 0 = Module is not bus-off or recovery has been requested by user in bus-off state
- 1 = Module is bus-off and holds this state until user request

9.9.14 Receive Error Counter (RXERR) Register

This register reflects the status of the CAN receive error counter. This register is read-only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1) and cannot be modified by writing.

Base + $E	7	6	5	4	3	2	1	0
Read	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-26. Receive Error Counter (RXERR) Register

NOTE:

Reading this register when in any other mode other than sleep or initialization modes may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition. Writing to this register when in special modes can alter the CAN functionality.

9.9.15 Transmit Error Counter (TXERR) Register

This register reflects the status of the CAN transmit error counter. This register is read-only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1). It cannot be modified by writing.

NOTE:

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition. Writing to this register when in special modes can alter the CAN functionality.

Base + $F	7	6	5	4	3	2	1	0
Read	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-27. Transmit Error Counter (TXERR) Register

9.9.16 Identifier Acceptance (IDAR0–7) Registers

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 CAN acceptance registers are applied on the IDR0–IDR3 registers of incoming messages in a bit by bit manner. Please see Section 9.9.19 and Section 9.5.2.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (IDAR0/1, IDMR0/1) are applied.

This register is read at anytime and receives modification by writing anytime in Initialization mode (INITRQ=1 and INITAK=1).

CAN Identifier Acceptance Register 0 (IDAR0) — Address: BASE + $10
CAN Identifier Acceptance Register 1 (IDAR1) — Address: BASE + $11
CAN Identifier Acceptance Register 2 (IDAR2) — Address: BASE + $12
CAN Identifier Acceptance Register 3 (IDAR3) — Address: BASE + $13

Base + $10 - $13	7	6	5	4	3	2	1	0
Read	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-28. First Bank Identifier Acceptance Registers 0–3 (IDAR0–3)

CAN Identifier Acceptance Register 4 (IDAR4) — Address: BASE + $18
CAN Identifier Acceptance Register 5 (IDAR5) — Address: BASE + $19
CAN Identifier Acceptance Register 6 (IDAR6) — Address: BASE + $1A
CAN Identifier Acceptance Register 7 (IDAR7) — Address: BASE + $1B

Base + $18 - $1B	7	6	5	4	3	2	1	0
Read	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-29. Second Bank Identifier Acceptance Registers 4–7 (IDAR4–7)

9.9.16.1 Acceptance Code (AC)—Bits 7–0

AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier (IDR*n*) register of the receive message buffer, are compared. The result of this comparison is then masked with the corresponding identifier mask register.

9.9.17 Identifier Mask Registers 0–7 (IDMR0–7)

The identifier mask register (IDMR) 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 IDMR1 and 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 IDMR1, IDMR3, IDMR5, and IDMR7 to don't care. It is read anytime and modified by writing anytime in initialization mode (INITRQ = 1 and INITAK = 1).

CAN Identifier Mask Register 0 (IDMR0) — Address: Base + $14
CAN Identifier Mask Register 1 (IDMR1) — Address: Base + $15
CAN Identifier Mask Register 2 (IDMR2) — Address: Base + $16
CAN Identifier Mask Register 3 (IDMR3) — Address: Base + $17

Base + $14 - $17	7	6	5	4	3	2	1	0
Read	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-30. First Bank Identifier Mask Registers 0–7 (IDMR0–3)

CAN Identifier Mask Register 4 (IDMR4) — Address: Base + $1C
CAN Identifier Mask Register 5 (IDMR5) — Address: Base + $1D
CAN Identifier Mask Register 6 (IDMR6) — Address: Base + $1E
CAN Identifier Mask Register 7 (IDMR7) — Address: Base + $1F

Base + $1C - $F	7	6	5	4	3	2	1	0
Read	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Write								
Reset	0	0	0	0	0	0	0	0

Figure 9-31. Second Bank Identifier Mask Registers 4–7 (IDMR4–7)

9.9.17.1 Acceptance Mask (IDMR)—Bits 7–0

If a particular bit in this register is cleared, this indicates the corresponding bit in the identifier acceptance register (IDAR) 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 the state of the corresponding bit in the IDAR 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

9.9.18 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 CAN stores a special 16-bit time stamp 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 in CTRL0 register is set. Please see Section 9.9.1.

The time stamp register is written by the CAN. The CPU can only read these registers.

Table 9-12. Message Buffer Organization

Offset Address	Register
0x00X0	Identifier Register 0
0x00X1	Identifier Register 1
0x00X2	Identifier Register 2
0x00X3	Identifier Register 3
0x00X4	Data Segment Register 0
0x00X5	Data Segment Register 1
0x00X6	Data Segment Register 2
0x00X7	Data Segment Register 3
0x00X8	Data Segment Register 4
0x00X9	Data Segment Register 5
0x00XA	Data Segment Register 6
0x00XB	Data Segment Register 7
0x00XC	Data Length Register
0x00XD	Transmit Buffer Priority Register[1]
0x00XE	Time Stamp Register (High Byte)[2]
0x00XF	Time Stamp Register (Low Byte)[2]

1. Not applicable for receive buffers
2. Read-only for CPU

Figure 9-32 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR is illustrated in Figure 9-32.

All bits of the receive and transmit buffers are *x* out of reset because of RAM-based implementation[1]. All reserved or unused bits of the receive and transmit buffers always read *x*.

1. Exception: The transmit priority registers are 0 out of reset.

Register Name		Bit 7	6	5	4	3	2	1	0
0x00X0 IDR0	R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
0x00X1 IDR1	R W	ID2	ID1	ID0	RTR	IDE (=0)			
0x00X2 IDR2	R W								
0x00X3 IDR3	R W								
0x00X4 DSR0	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X5 DSR1	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X6 DSR2	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X7 DSR3	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X8 DSR4	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X9 DSR5	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XA DSR6	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XB DSR7	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XC DLR	R W					DLC3	DLC2	DLC1	DLC0

= Unused, always read x

Figure 9-32. Receive/Transmit Message Buffer — Standard Identifier Mapping

Read transmit buffers anytime when TXE*n* bit is set and the corresponding transmit buffer is selected in TBSEL for receive buffers, only when RXF bit is set. Please see Section 9.9.5, Section 9.9.7 and Section 9.9.11.

Write to transmit buffers anytime when TXE*n* bit is set and the corresponding transmit buffer is selected in TBSEL. Please see Section 9.9.7 and Section 9.9.11. This register is not implemented for receive buffers. Reset is undefined (0x00XX) due to RAM-based implementation.

Register Name		Bit 7	6	5	4	3	2	1	0
0x00X0 IDR0	R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	W								
0x00X1 IDR1	R	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
	W								
0x00X2 IDR2	R	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
	W								
0x00X3 IDR3	R	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
	W								
0x00X4 DSR0	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X5 DSR1	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X6 DSR2	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X7 DSR3	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X8 DSR4	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X9 DSR5	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00XA DSR6	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00XB DSR7	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00XC DLR	R					DLC3	DLC2	DLC1	DLC0
	W								

= Unused, always read x

Figure 9-33. Receive/Transmit Message Buffer — Extended Identifier Mapping

9.9.19 Extended Identifier Registers (IDR0–IDR3)

The identifier registers for an extended format identifier consists of a total of 32 bits; ID[28:0], SRR, IDE, and RTR bits. The identifier registers for a standard format identifier consists of a total of 13 bits; ID[10:0], RTR, and IDE bits.

9.9.20 Extended Identifier Register 0 (IDR0)

Base + $X0	7	6	5	4	3	2	1	0
Read	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-34. Identifier Register 0 (IDR0)–Extended Identifier Mapping

9.9.20.1 Extended Format Identifier (ID)—Bits 7–0

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.

9.9.21 Identifier Register 1 (IDR1)

Base + $X1	7	6	5	4	3	2	1	0
Read	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-35. Identifier Register 1 (IDR1)–Extended Identifier Mapping

9.9.21.1 Extended Format Identifier (ID)—Bits 7–5

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.

9.9.21.2 Substitute Remote Request (SRR)—Bit 4

This fixed recessive bit is used only in extended format. It must be set to one for transmission buffers and is stored as received on the CAN bus for receive buffers.

9.9.21.3 ID Extended (IDE)—Bit 3

This bit indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the bit is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the bit indicates to the CAN what type of identifier to send.

- 0 = Standard format (11-bit)
- 1 = Extended format (29-bit)

9.9.21.4 Extended Format Identifier (ID)—Bits 2–0

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.

9.9.22 Identifier Register 2 (IDR2)

Base + $X2	7	6	5	4	3	2	1	0
Read	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-36. Identifier Register 2 (IDR2)–Extended Identifier Mapping

9.9.22.1 Extended Format Identifier (ID)—Bits 7–0

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.

9.9.23 Identifier Register 3 (IDR3)

Base + $X3	7	6	5	4	3	2	1	0
Read	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-37. Identifier Register 3 (IDR3)–Extended Identifier Mapping

9.9.23.1 Extended Format Identifier (ID)—Bits 7–1

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.

9.9.23.2 Remote Transmission Request (RTR)—Bit 0

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

9.9.24 Standard Identifier Register 0 (IDR0)

Base + $X0	7	6	5	4	3	2	1	0
Read	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-38. Standard Identifier Register 0

9.9.24.1 Standard Format Identifier (ID)—Bits 7–0

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. See also ID bits in Section 9.9.25.

9.9.25 Standard Identifier Register 1 (IDR1)

Base + $X1	7	6	5	4	3	2	1	0
Read	ID2	ID1	ID0	RTR	IDE (=0)			
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-39. Standard Identifier Register 1

9.9.25.1 Standard Format Identifier (ID)—Bits 7–5

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.

9.9.25.2 Remote Transmission Request (RTR)—Bit 4

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

9.9.25.3 ID Extended (IDE)—Bit 3

This bit indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the bit is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the bit indicates to the CAN what type of identifier to send.

- 0 = Standard format (11-bit)
- 1 = Extended format (29-bit)

9.9.25.4 Reserved—Bits 2–0

This bit field is reserved or not implemented. It cannot be read nor modified by writing.

9.9.26 Standard Identifier Register 2 (IDR2)

Base + $X2	7	6	5	4	3	2	1	0
Read								
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-40. Standard Identifier Register 2 (IDR2)

9.9.27 Standard Identifier Register 3 (IDR3)

Base + $X3	7	6	5	4	3	2	1	0
Read								
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-41. Standard Identifier Register 3 (IDR3)

9.9.28 Data Segment Registers 0–7 (DSR0–7)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

CAN Identifier Mask Register 4 (DSR0) — Address: Base + $X4
CAN Identifier Mask Register 5 (DSR1) — Address: Base + $X5
CAN Identifier Mask Register 6 (DSR2) — Address: Base + $X6
CAN Identifier Mask Register 7 (DSR3) — Address: Base + $X7
CAN Identifier Mask Register 8 (DSR4) — Address: Base + $X8
CAN Identifier Mask Register 9 (DSR5) — Address: Base + $X9
CAN Identifier Mask Register A (DSR6) — Address: Base + $XA
CAN Identifier Mask Register B (DSR7) — Address: Base + $XB

Base + $X4 - $XB	7	6	5	4	3	2	1	0
Read	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-42. Data Segment Registers (DSR0–DSR7)—Extended Identifier Mapping

9.9.29 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

Base + $XC	7	6	5	4	3	2	1	0
Read					DLC3	DLC2	DLC1	DLC0
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-43. Data Length Register (DLR)–Extended Identifier Mapping

9.9.29.1 Reserved—Bits 7–4

This bit field is reserved or not implemented. It cannot be read nor modified by writing.

9.9.29.2 Data Length Code (DLC)—Bits 3–0

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. Table 9-13 shows the effect of setting the DLC bits.

Table 9-13. Data Length Codes

Data Length Code				Data Byte Count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

9.9.30 Transmit Buffer Priority Register (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 TXE*n* bit 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.

Base + $XD	7	6	5	4	3	2	1	0
Read	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-44. Transmit Buffer Priority Register (TBPR)

This is a read anytime register when TXE*n* bit is set and the corresponding transmit buffer is selected in TBSEL. Please see Section 9.9.7 and Section 9.9.11.

This is a write anytime register when TXE*n* bit is set (see Section 9.9.7) and the corresponding transmit buffer is selected in TBSEL. Please see Section 9.9.7 and Section 9.9.11.

9.9.31 Time Stamp Register (TSRH–TSRL)

If the TIME bit is enabled, the CAN writes a special time stamp to the respective registers in the active transmit or receive buffer as soon as a message is acknowledged on the CAN bus. Please refer to Section 9.9.1. The time stamp is written on the bit sample point for the recessive bit of the ACK delimiter in the CAN frame. In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer is flagged empty.

The timer value, used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the CAN. The timer is reset, that is all bits set to zero during Initialization mode. The CPU can only read the Time Stamp Registers (TSR).

Base + $XE	7	6	5	4	3	2	1	0
Read	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-45. Time Stamp Register–High Byte (TSRH)

Base + $XF	7	6	5	4	3	2	1	0
Read	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
Write								
Reset	x	x	x	x	x	x	x	x

Figure 9-46. Time Stamp Register–Low Byte (TSRL)

These registers are read at anytime when TXE*n* bit is set and the corresponding transmit buffer is selected in TBSEL. Please refer to Section 9.9.7 and Section 9.9.11. Writing to these registers is not implemented.

9.10 Interrupts

This section describes all interrupts originated by the CAN. It documents the enable bits and generated bits. Each interrupt is listed and described separately.

9.10.1 Description of Interrupt Operation

The CAN supports four interrupt vectors (see Table 9-14), any of which can be individually masked. For details see Section 9.9.6, to Section 9.9.9.

NOTE:

The dedicated interrupt vector addresses are defined in Section 9.10 and Section 9.11.

Table 9-14. Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
Wakeup interrupt (WUPIF)	I bit	RIER (WUPIE)
Error interrupts interrupt (CSCIF, OVRIF)	I bit	RIER (CSCIE, OVRIE)
Receive interrupt (RXF)	I bit	RIER (RXFIE)
Transmit interrupts (TXE[2:0])	I bit	TIER (TXEIE[2:0])

9.10.2 Transmit Interrupt

At least one of the three transmit buffers is empty, or not scheduled, and can be loaded to schedule a message for transmission. The TXE*n* bit of the empty message buffer is set.

9.10.3 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 bit is set. If there are multiple messages in the receiver FIFO, the RXF bit is set as soon as the next message is shifted to the foreground buffer.

9.10.4 Wakeup Interrupt

A wakeup interrupt is generated if activity on the CAN bus occurs during CAN internal sleep mode. WUPE bit in CTRL0 register must be enabled. Please see Section 9.9.1.

9.10.5 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurrs. Section 9.9.5 indicates one of the following conditions:

- Overrun — An overrun condition of the receiver FIFO, described in Section 9.5.1.3, occurred
- CAN Status Change — Actual value of the transmit and receive error counters control the CAN bus state of the CAN. As soon as the error counters skip into a critical range (TX/RX-warning, TX/RX-error, bus-off) the CAN flags an error condition. The status change, causing the error condition, is indicated by the TSTAT and RSTAT bits. Please see Section 9.9.5 and Section 9.9.6.

9.10.6 Interrupt Acknowledge

Interrupts are directly associated with one or more status bits in either the Section 9.9.5 or the Section 9.9.7. Interrupts are pending as long as one of the corresponding bits is set. The bits in RFLG and TFLG must be reset within the interrupt handler to handshake the interrupt. The bits are reset by writing one to the corresponding bit position. A bit cannot be cleared if the respective condition prevails.

NOTE:

It must be guaranteed the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt bits. These instructions may cause accidental clearing of interrupt bits, set after entering the current interrupt service routine.

9.10.7 Recovery from Stop or Wait

The CAN can recover from stop or wait via the wakeup interrupt. This interrupt can only occur if the CAN 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).

9.11 Resets

The reset state of each individual bit is listed in Section 9.9, detailing all the registers and their bit fields.

Table 9-15. Document Revision History for Chapter 9

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 10
Joint Test Action Group Port (JTAG)

10.1 Introduction

Because this device also has a 56800E core containing its own test access port (TAP), or *Core TAP*, a TAP linking module (TLM) it is included to manage the TAP access. Normal operation of this part will use the chip TAP as the master TAP controller, thereby disabling the 56800E TAP (Core TAP) controller. This chapter discusses the master TAP only.

10.2 Features

TAP characteristics include:

- Provide a means of accessing the EOnCE module controller and circuits to control the 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 bypass register
- Enable/disable pullup devices on peripheral boundary scan pins

10.3 Block Diagram

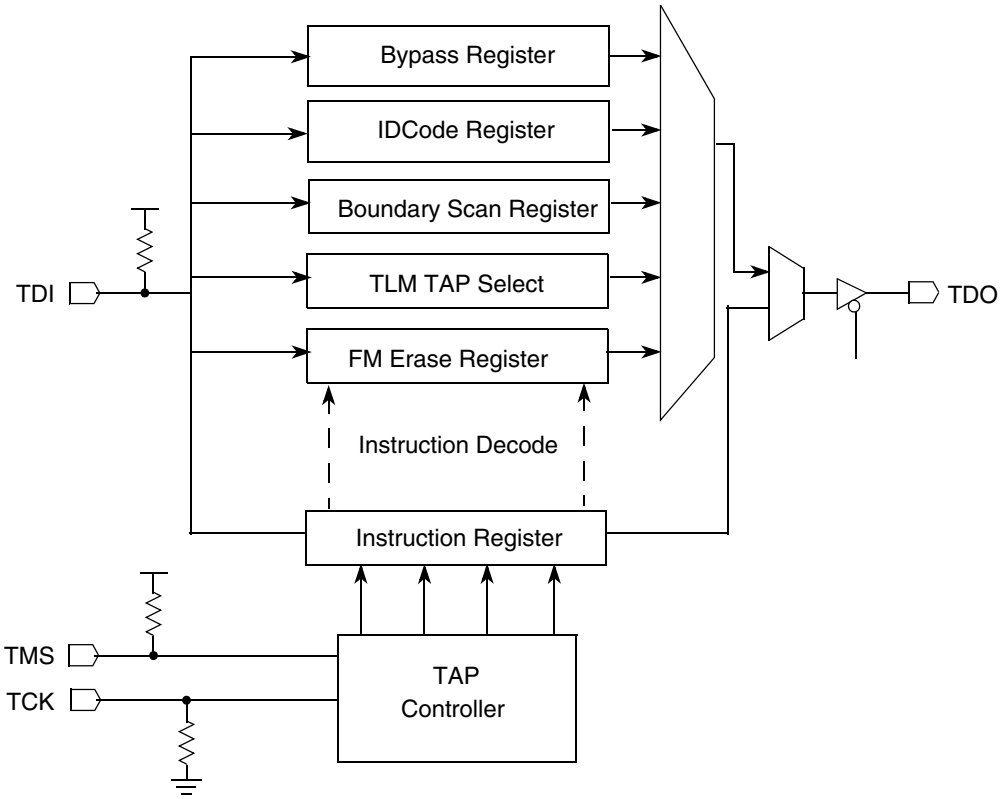


Figure 10-1. JTAG Block Diagram

10.4 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.

10.4.1 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 10-1. 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 *56800E Data Sheet*.

10.4.2 Master TAP Instructions

The eight-bit master TAP instruction register is in support of all JTAG functions. It is described in Table 10-1. This register includes all IEEE 1149.1 required instructions plus several additional instruction registers accommodating TAP selection and Flash lockout recovery.

Table 10-1. Master TAP Instructions Opcode

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

10.4.2.1 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 10-2. This instruction enhances test efficiency by shortening the overall path between TDI and TDO when no test operation of a component is required.

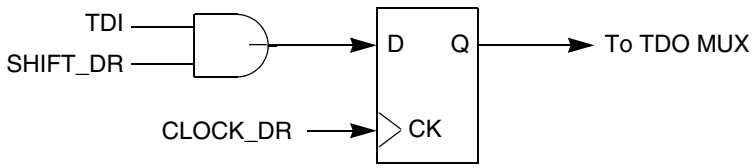


Figure 10-2. Bypass Register Diagram

10.4.2.2 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.

10.4.2.3 TLMSEL

The TLMSEL instruction is an implementation specific JTAG instruction, disabling the master TAP and enabling the TAP linking module, or TLM. The TLM then selects the 56800E core TAP or the master TAP as the enabled TAP.

10.4.2.4 FLASH ERASE

The Flash erase instruction is an implementation specific JTAG instruction that mass erases and unsecures the Flash memory.

10.5 TAP Controller

The TAP controller is a synchronous 16-bit finite state machine illustrated in Figure 10-3. 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.

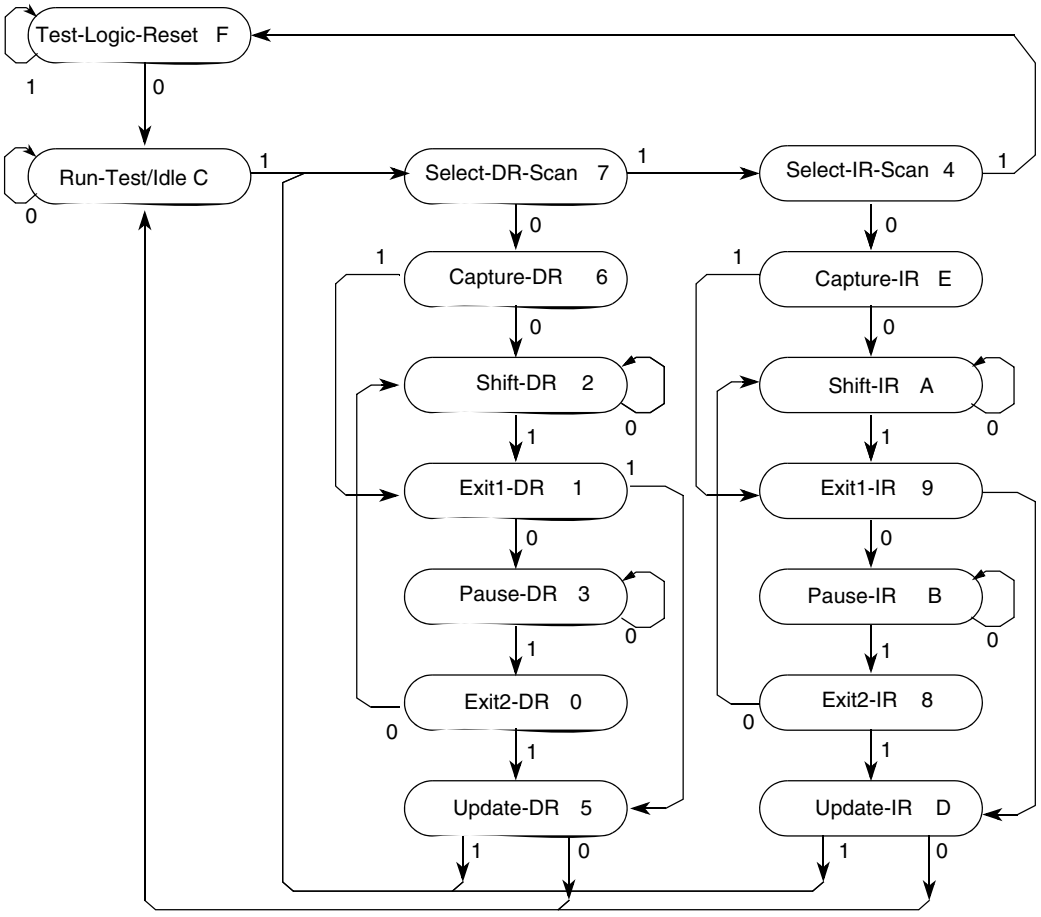


Figure 10-3. TAP Controller State Diagram

10.5.1 Operation

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 Figure 10-3.

10.5.1.1 Test Logic Reset (pstate = F)

During test-logic-reset, all JTAG test logic is disabled so the chip 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.

10.5.1.2 Run-Test-Idle (pstate = C)

Run-test-idle is a controller state between scan operations. Once 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.

10.5.1.3 Select Data Register (pstate = 7)

The select-data register state is a temporary state. In this state, 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 data register is initiated. If TMS is held high and a rising edge of TCK occurs, the controller moves to the select-IR state.

10.5.1.4 Select Instruction Register (pstate = 4)

The select-instruction register state is a temporary state. In this state, 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.

10.5.1.5 Capture Data Register (pstate = 6)

In this controller state, data may be parallel loaded 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.

10.5.1.6 Shift Data Register (pstate = 2)

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 one is applied to TMS and a positive edge of TCK occurs, the controller will move to the exit1-DR state.

10.5.1.7 Exit1 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 Uupdate-DR state. This terminates the scanning process.

10.5.1.8 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.

10.5.1.9 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.

10.5.1.10 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 is used to apply 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.

10.5.1.11 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.

10.5.1.12 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.

10.5.1.13 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.

10.5.1.14 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.

10.5.1.15 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.

10.5.1.16 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 into the instruction latch. It becomes the current instruction. On a rising edge of TCK, the controller advances to the select-IR state if TMS is held high or the run-test-idle state if TMS is held low.

10.6 Register Description

JTAG has no memory mapped registers.

10.7 Pin Description

The signal summaries for the JTAG are located in Table 10-2.

Table 10-2. JTAG Pin Description

Pin Name	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 pull down non-disabled resistor.
TDI	Test data input—This input pin provides a serial input data stream to all TAP controllers and the TLM. It is sampled on the rising edge of TCK. TDI has an on-chip pullup resistor which can be disabled through the PUPEN register in the GPIO module.
TMS	Test mode select input—This input pin is used to sequence all TAP Controllers and TLM state machine. It is sampled on the rising edge of TCK. TMS has an on-chip pull up resistor which can be disabled through the PUPEN register in the GPIO module.
TDO	Test data output—This three-stated 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. TDO has an on-chip pull up resistor which can be disabled through the PUPEN register in the GPIO module.

10.8 Clocks

10.8.1 TCK

This is the sole clock used by the master TAP module. If the EOnCE module is not being accessed using the master or 56800E core TAP controllers, the maximum TCK frequency is one-quarter the maximum frequency for the 56800E core. When accessing the EOnCE module through the 56800E core TAP controller, the maximum frequency for TCK is one eighth the maximum frequency for the 56800E core.

Table 10-3. Clock Summary

Clock	Priority	Source	Characteristics
TCK	1	External	This user provided clock shifts data and controls the state machine.

10.9 Interrupts

This module has no interrupt capabilities.

Table 10-4. Document Revision History for Chapter 10

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 11
Power Supervisor (PS)

11.1 Introduction

This chapter details the on-chip power supervisor (PS) module. Its function ensures the chip is operated only within required voltage ranges while assisting in a orderly shutdown of the chip in the event the power supply is interrupted, or there is a brownout, or a voltage sag.

11.2 Features

The power supervisor (PS) monitors on-chip voltages (digital 3.3 V and 2.5 V rails), providing the following qualities:

- Power-on reset (POR)
  - Holds device in reset until:
    - V_{DD} core voltage exceeds 1.8 V
    - Regulator voltages have risen above LVI thresholds (2.2 V and 2.7 V)
- Core low voltage interrupt (LVI22)
  - Generated when the 2.5 V rail drops below 2.2 V
- I/O low voltage interrupt (LVI27)
  - Generated when the 3.3 V rail drops below 2.7 V

Working under the assumption power supply voltages move relatively slowly with respect to the system clocks, low voltage interrupts should provide ample warning of impending problems as well as time to accomplish an orderly shutdown of the chip be accomplished.

11.3 Block Diagram

The basic power-on reset (POR) and low voltage detect module is illustrated in Figure 11-1. The POR circuit is designed to assert the internal reset from V_{DD} = 0 V to V_{DD} = 1.8 V. $\overline{POR}$ switching high indicates the 3.3 V and 2.5 V rails are above their minimum thresholds. POR is asserted when the core voltage drops below 1.8 V.

The deglitch blocks are essentially strings of four flops in series. The outputs of all four must indicate an active interrupt before the output switches to the active state. This is intended to prevent the LVI circuitry from responding to momentary glitches brought about as a result of normal operation. Once set, the sticky status bits LVIS27S, LVIS22S, and LVI must be explicitly reset by writing one.

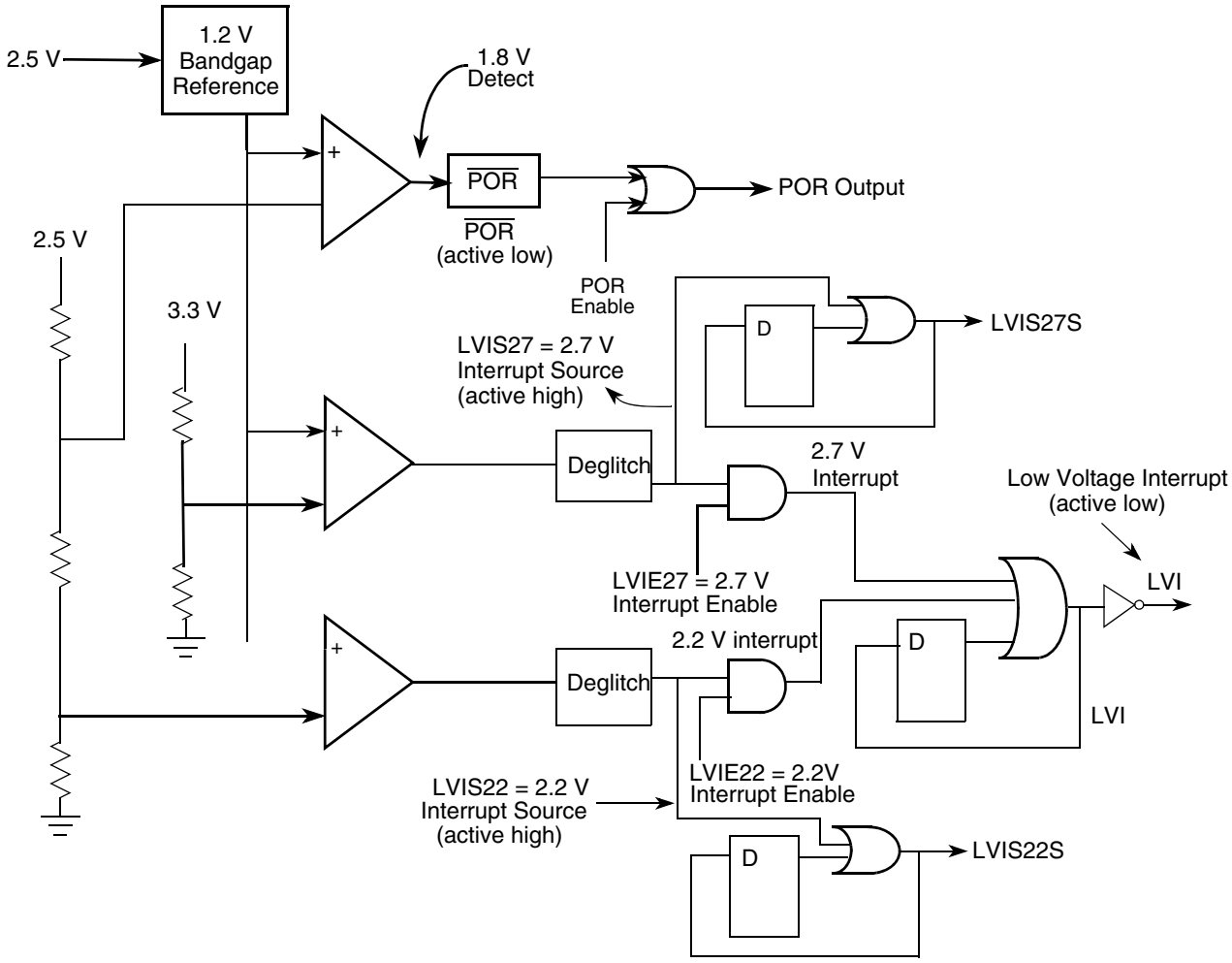


Figure 11-1. PS Block Diagram

11.4 Functional Description

The power-on reset ($\overline{\text{POR}}$) signal is always enabled and the low voltage interrupts are disabled out of reset. As the device powers up, the voltage regulator circuit provides 3.3 V for the I/O circuitry and 2.5 V for the core. When the core voltage exceeds 1.8 V and the LVI (2.2 V and 2.7 V) thresholds are exceeded, the device is released from reset 64 clock cycles later, illustrated in Figure 11-2. At this point, the LVIs should be enabled and the PLL used to increase the system clock frequency.

For most initializations, waiting for the PLL to lock represents the largest component in the start-up time line. Hence, it is prudent to check the LVI status bits again just after PLL lock is achieved.

Figure 11-2 illustrates operation of the $\overline{\text{POR}}$ versus low voltage detect circuits. The LVI should be explicitly cleared, and disabled by the interrupt service routine responsible for shutting down the part when a low voltage is detected.

Low voltage interrupts are masked upon $\overline{\text{POR}}$. They must be explicitly enabled and disabled thereafter. Low voltage status bits in the power supervisor status (STAT) register are always active, independent of whether low voltage interrupts are enabled. LVIs include roughly 50-100 mV of hysteresis.

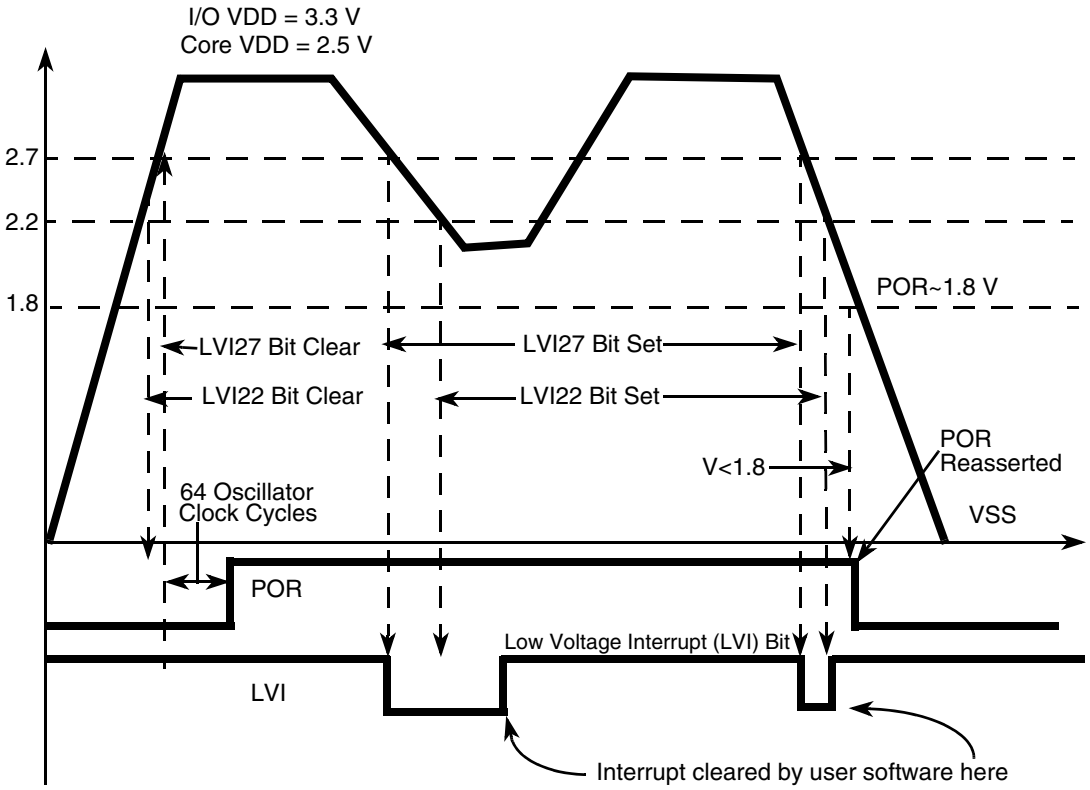


Figure 11-2. $\overline{\text{POR}}$ Vs. Low-Voltage Interrupts

11.5 Register Description

Table 11-1. PS Memory Map

Device	Peripheral	Address
56F80xx	PS	$00F140

A register address is the sum of a base address and an address offset. The base address is defined at the device level and the address offset is defined at the module level. The low power voltage supervisor (PS) module has two registers.

Table 11-2. Power Supervisor Register Summary

Register Address	Register Acronym	Register Name	Access Type	Chapter Location
Base + $0	CTRL	Control Register	Read/Write	Section 11.5.1
Base + $1	STAT	Status Register	Read/Write	Section 11.5.2

Bit fields of each of the two PS registers are illustrated in Table 11-3 and Table 11-4. Details of each follow.

11.5.1 Control (CTRL) Register

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LVIE27	LVIE22
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-3. Control (CTRL) Register

11.5.1.1 Reserved—Bits 15–2

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

11.5.1.2 2.7 V Low Voltage Interrupt Enable (LVIE27)—Bit 1

- 0 = Interrupt is disabled
- 1 = Interrupt is enabled

11.5.1.3 2.2 V Low Voltage Interrupt Enable (LVIE22)—Bit 0

- 0 = Interrupt is disabled
- 1 = Interrupt is enabled

11.5.2 Status (STAT) Register

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	LVI	LVIS27S	LVIS22S	LVIS27	LVIS22
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-4. Status (STAT) Register

11.5.2.1 Reserved—Bits 15–5

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

11.5.2.2 Low Voltage Interrupt (LVI)—Bit 4

LVI is a sticky status bit derived from the conditions where either of the 2.2 or 2.7 LVI thresholds have been met and associated interrupts are enabled.

LVI = (LVIS27 AND LVIE27) or (LVIS22 AND LVIE22)

This bit may be cleared by writing one to it. Writing zero has no effect. It may be set again in the very next clock cycle after being cleared if the voltages are still below the thresholds.

- 0 = Interrupt not asserted
- 1 = Interrupt asserted

11.5.2.3 Sticky 2.7 V Low Voltage Interrupt Source (LVIS27S)—Bit 3

When this bit is set, it must be cleared explicitly by writing one to it. Writing zero has no effect.

- 0 = 2.7 V interrupt threshold not passed
- 1 = 3.3 V supply dropped below the 2.7 V interrupt threshold

Bit LVIS27S may be set again in the very next clock cycle after being cleared if the condition causing the supply voltage to drop still persists.

11.5.2.4 Sticky 2.2 V Low Voltage Interrupt Source (LVIS22S)—Bit 2

When this bit is set, it must be cleared explicitly by writing one to it. Writing zero has no effect.

- 0 = 2.2 V interrupt threshold not passed
- 1 = 2.5 V supply dropped below the 2.2 V interrupt threshold

Bit LVIS22S may be set again in the very next clock cycle after being cleared if the condition causing the supply voltage to drop still persists.

11.5.2.5 Non-Sticky 2.7 V Low Voltage Interrupt Source (LVIS27)—Bit 1

This read-only bit resets itself if the supply voltage raises above the comparator threshold point. This bit cannot be modified.

- 0 = 2.7 V interrupt threshold not passed
- 1 = 3.3 V supply dropped below the 2.7 V interrupt threshold

11.5.2.6 Non-Sticky 2.2 V Low Voltage Interrupt Source (LVIS22)—Bit 0

This read-only bit resets itself if the supply voltage raises above the comparator threshold point. This bit cannot be modified.

- 0 = 2.2 V interrupt threshold not passed
- 1 = 2.5 V supply is below the 2.2 V interrupt threshold

11.6 Suggestions for LVI Interrupt Service Routines

The presence of one or more LVI bits high in the power supervisor status (STAT) register indicates normal operation of the chip is no longer possible. If the 2.7 V interrupt occurred, the operation of the chip I/O is questionable. Peripherals operates correctly but output signal levels will not meet requirements. If the 2.2 V interrupt occurred, the core’s operation at high speed is questionable.

Suggested Handling of the LVI Interrupt:

```
While (LVIS22S or LVIS27S) bit is set
{
    If (LVIS22S bit is set) then
    {
        Clear LVIS22S bit
        Reduce the core operating frequency to 8 MHz
        Shutdown ALL peripherals in an orderly manner.
        While (LVIS22S bit is set) then
        {
        Clear LVIS22S bit
        Delay four clock cycles
        }
        Restore Core Operating Frequency
    }
    If (LVIS27S bit is set) then
    {
        Clear LVIS27S bit
        Shutdown I/O peripherals in an orderly manner
        While ((LVIS27S bit is set) and (LVIS22S bit is not set)) then
        {
        Clear LVIS27S bit
        Delay four clock cycles
        }
    }
}
Restore ALL peripherals to operation
Return from ISR
```

Table 11-3. Document Revision History for Chapter 11

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 12
Queued Serial Communications Interface (QSCI)

12.1 Introduction

This chapter describes the queued serial communications interface (QSCI) module. The module allows asynchronous serial communications with peripheral devices and other controllers.

12.2 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- or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter CPU interrupt requests
- Programmable polarity for transmitter and receiver
- Two receiver wakeup methods:
  - Idle line
  - Address mark
- Interrupt-driven operation with seven flags:
  - Transmitter empty
  - Transmitter idle
  - Receiver full
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
  - LIN sync error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

12.3 Block Diagram

The block diagram of the design is located in Figure 12-1.

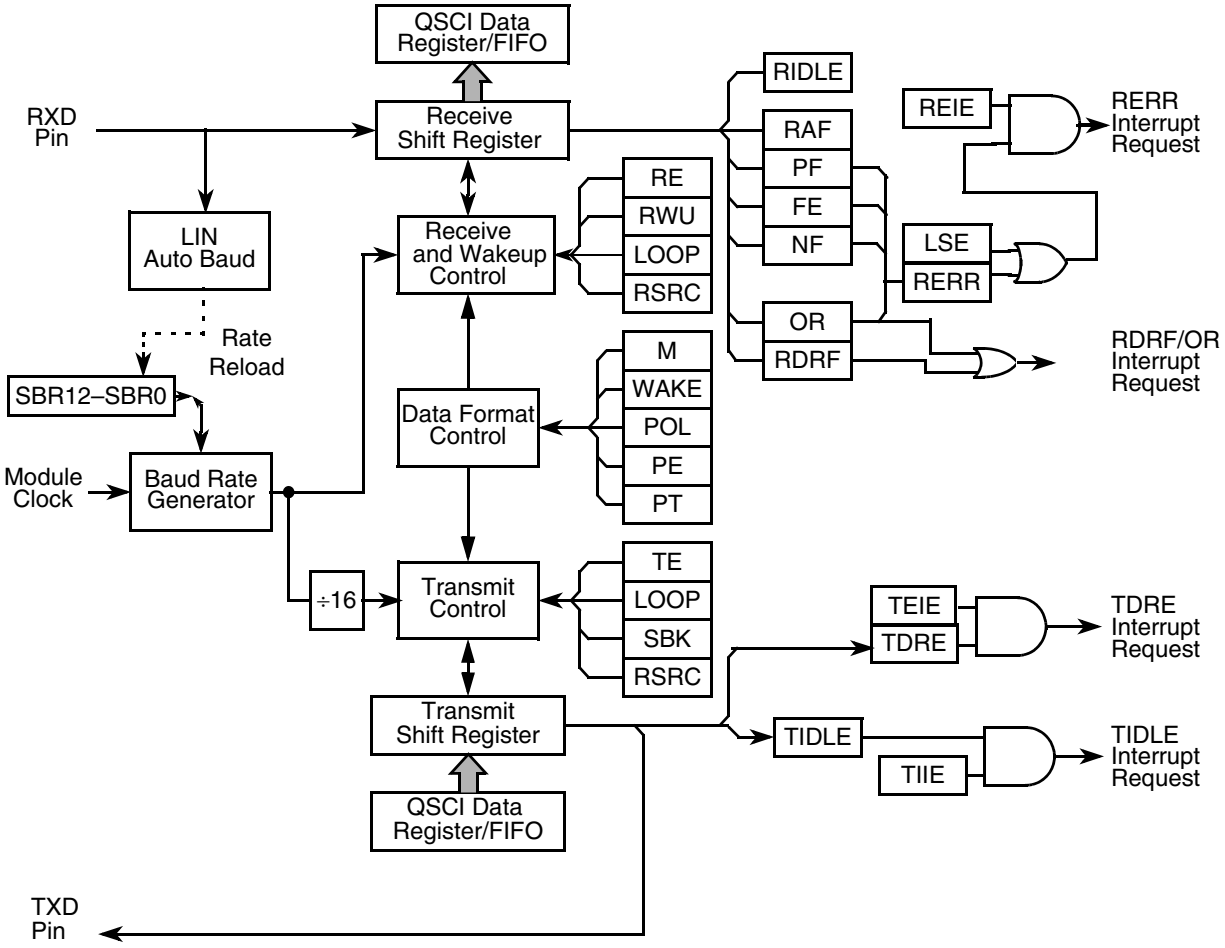


Figure 12-1. QSCI Block Diagram

12.4 Functional Description

Figure 12-1 explains the QSCI module structure. The QSCI allows full duplex, asynchronous, non-return-to-zero (NRZ) serial communication between the controller and remote devices, including other controllers. The QSCI transmitter and receiver operate independently although they use the same baud rate generator. The CPU monitors the status of the QSCI, writes the data to be transmitted, and processes received data.

When initializing the QSCI, be certain to set the proper peripheral enable bits in the general purpose input/output (GPIO) registers as well as any pull-up enables.

To initialize the QSCI for full duplex operation:

- Initialize software flags
- Set each bit of the CTRL1 register as 0:

  LOOP = 0, SWAI = 0, RSRC = 0, M = 0, WAKE = 0, POL = 0, PE = 0, PT = 0, TEIE = 0
  TITE = 0, RFIE = 0, REIE = 0, TE = 0, RE = 0, RWU = 0, and SBK = 0

- If the LIN slave mode is desired, enable that bit in the CTRL2 register
- Set the prescaler bits
- Enable transmitter and receiver

12.4.1 Data Frame Format

The QSCI uses the standard NRZ mark/space data frame format illustrated in Figure 12-2.

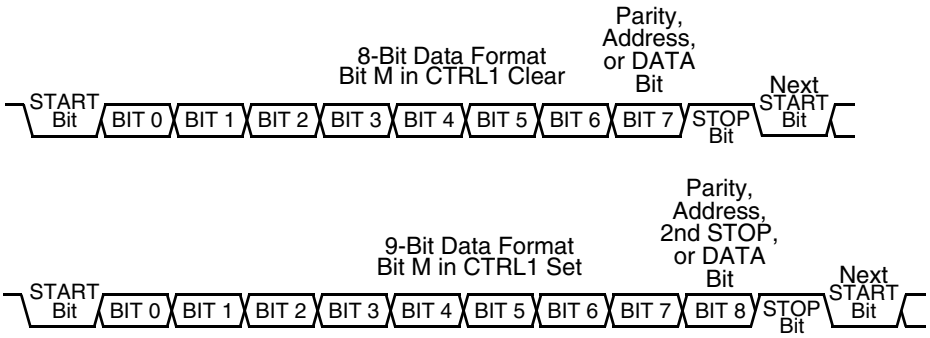


Figure 12-2. QSCI Data Frame Formats

Each data character is contained in a frame including a START bit, eight or nine data bits, and a STOP bit. Clearing the MODE (M) bit in the CTRL1 register configures the QSCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Formats are provided in Table 12-1.

Table 12-1. 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. Please see Section 12.4.4.6

Setting the M bit configures the QSCI for 9-bit data characters. A frame with nine data bits has a total of 11 bits. Formats are provided in Table 12-2.

Table 12-2. 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. Please see **Section 12.4.4.6.**

12.4.2 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 SBR and FRAC_SBR bits determines the module clock divisor. The SBR and FRAC_SBR bits are in the QSCI baud rate (RATE) register. The baud rate clock is synchronized with the IPBus 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:

1. The integer division of the module clock may not give the exact target frequency.
2. Synchronization with the bus clock can cause phase shift.

Table 12-3 list some examples of achieving target baud rates with a module clock frequency of 32 MHz.

Table 12-3. Example Baud Rates (Module Clock = 32MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
1.375	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
104125	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 module clock rate divided by 16. System overhead may preclude processing the data at this speed.

12.4.3 Transmitter

Figure 12-3 illustrates the QSCI transmitter block diagram. Detailed discussion of the transmitter is in the following sections.

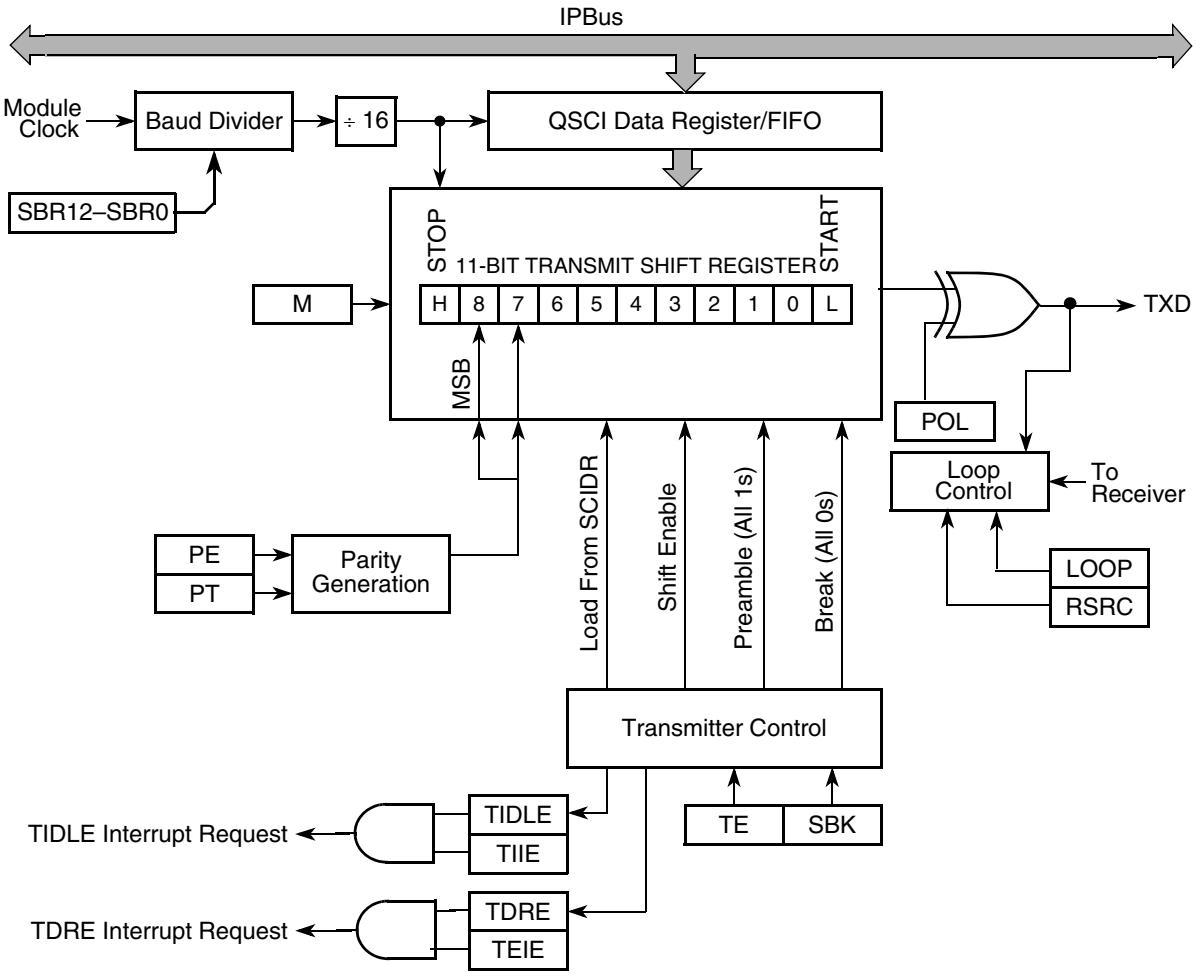


Figure 12-3. QSCI Transmitter Block Diagram

12.4.3.1 Character Length

The QSCI transmitter can accommodate either 8- or 9-bit data characters. The state of the M bit in the control 1 (CTRL1) register determines the length of data characters.

12.4.3.2 Character Transmission

During a QSCI transmission, the transmit shift register shifts a frame out to the TXD pin. The data (DATA) register is the write-only buffer between the internal data bus and the transmit shift register.

To initiate a QSCI transmission:

1. Enable the transmitter by writing a logic 1 to the transmitter enable (TE) bit in the QEMI control 1 (CTRL1) register.
2. Wait for the transmit data register empty (TDRE) flag to be set.
3. Clear the TDRE flag, by first reading the QSCI status (STAT) register, and then writing to the QSCI data (DATA) register.
4. Repeat step 2 and 3 for each subsequent transmission.

Writing the TE bit from 0 to 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic automatically transfers the data from the DATA register into the transmit shift register. A logic 0 START bit automatically goes into the least significant bit (LSB) position of the transmit shift register. A logic 1 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 (TDRE) flag in the STAT register becomes set when the DATA register transfers a character to the transmit shift register leaving the TXFIFO word count (TFCNT) at the TXWM value. The TDRE flag indicates the DATA register can accept new transmit data from the internal data bus. When the FIFO is enabled, the DATA register can have room to accept new data even if TDRE is not set. If the transmitter empty interrupt (TEIE) bit in the control (CTRL1) register is also set, the TDRE flag generates a transmitter empty interrupt request.

When the transmit shift register is not transmitting a frame and TE = 1, the TXD pin goes to the idle condition, logic 1. If, at any time, software clears the TE bit in the control (CTRL1) register, the transmitter relinquishes control of the port I/O pin upon completion of the current transmission, causing the TXD pin to go into a HighZ state.

If software clears TE while a transmission is in progress (TIDLE = 0), the frame in the transmit shift register continues to shift out. Then transmission stops even if there is data pending in the QSCI DATA register. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles having 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 the TDRE flag to go high while TFWM = 00, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing, and then set the TE bit.
4. Write the first character of the second message to DATA register.

12.4.3.3 Break Characters

Writing logic 1 to the send break (SBK) bit in the CTRL1 register loads the transmit shift register with a break character. A break character contains all logic 0s without START, STOP, or PARITY bits. Break character length depends on the MODE (M) bit in the CTRL1 register. As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the transmit shift register completes transmitting the last break character subsequently transmitting at least one logic 1. The automatic logic 1 at the end of the last break character guarantees the recognition of the START bit of the next frame.

The QSCI recognizes a break character when a start bit is followed by eight or nine logic 0 data bits and a logic 0 where the STOP bit should be. Receiving a break character has these effects on QSCI registers:

- Sets the framing error (FE) flag
- Sets the receive data register full (RDRF) flag, if RFCNT >RFWM.
- Clears the data (DATA) register
- May set the overrun (OR) flag, noise flag (NF), parity error (PE) flag, or the receiver active flag (RAF). Please see STAT register in Section 12.6.4.

12.4.3.4 Preambles

A preamble contains all logic 1s with no START, STOP, or PARITY bit. Preamble length depends on the M bit in the CTRL1 register. The preamble is a synchronizing mechanism initiating the first transmission begun after modifying the TE bit from 0 to 1.

If the TE bit is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues a preamble to be sent after the frame currently being transmitted.

NOTE:

Toggle the TE bit for a queued preamble when the TDRE flag becomes set and immediately before writing the next character to the DATA register.

When queueing a preamble, return the TE bit to logic 1 before the STOP bit of the current frame shifts out to the TXD pin. Setting TE after the STOP bit appears on TXD causes data previously written to the DATA register to be lost.

12.4.4 Receiver

Figure 12-4 illustrates the QSCI receiver block diagram. Detailed discussion of the receiver function is in the following paragraphs.

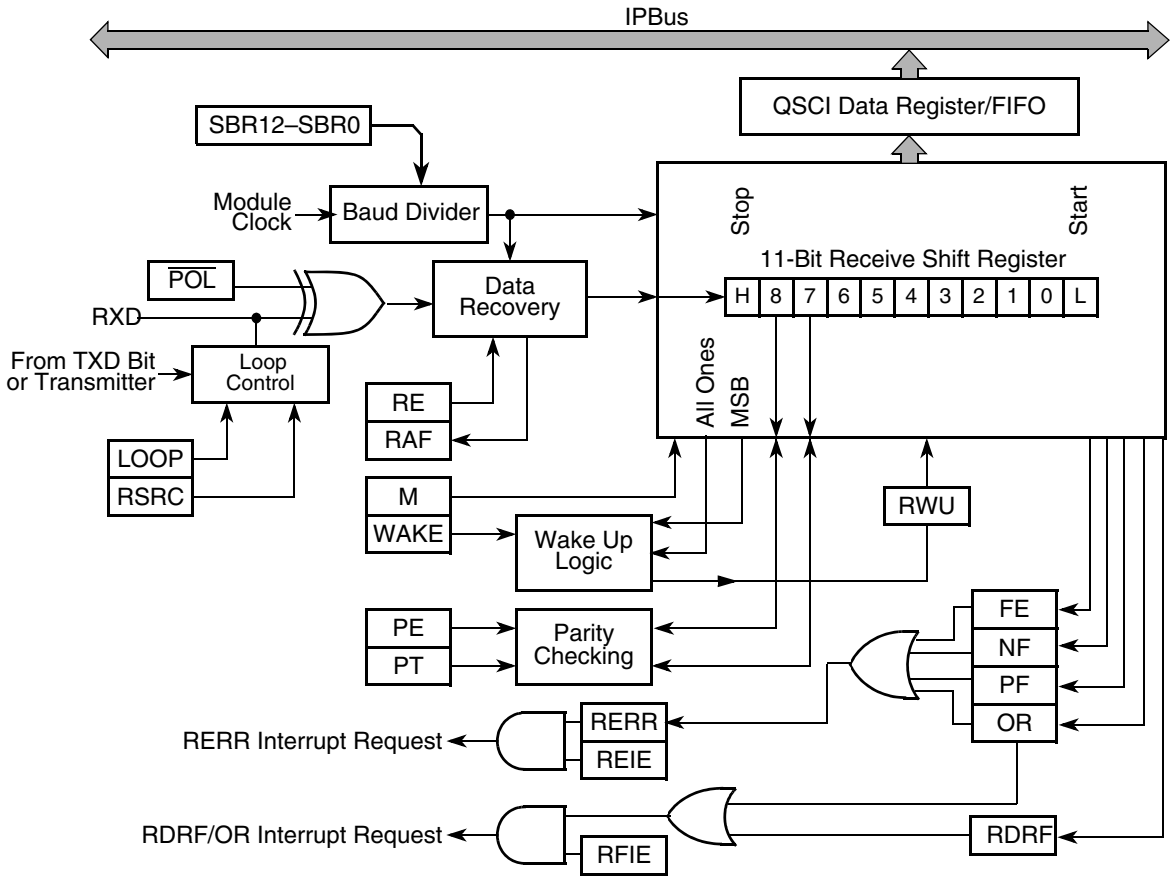


Figure 12-4. QSCI Receiver Block Diagram

12.4.4.1 Character Length

The QSCI receiver can accommodate either 8- or 9-bit data characters. The state of the M bit in the CTRL1 register determines the length of data characters.

12.4.4.2 Character Reception

During a QSCI reception, the receive shift register moves a frame in from the RXD pin. The 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 FE, NF, PF, and LSE status flags transfer to the DATA register. The receive data register full (RDRF) flag in the STAT register is set when the RX FIFO word count is above the watermark, indicating a received character can be read. When the FIFO is enabled, readable data words can be received even if RDRF is not set. If the receiver full interrupt enable (RFIE) bit in the CTRL1 register is also set, the RDRF flag generates an RDRF interrupt request.

The FE, NF, PF, and LSE bits of the STAT are the flags associated with the current character to be read from the receive data register/FIFO.

12.4.4.3 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 the baud rate mismatch, the RT clock, illustrated in Figure 12-5, is resynchronized:

- After every START bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

To locate the START bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible START bit occurs, the RT clock begins to count to 16.

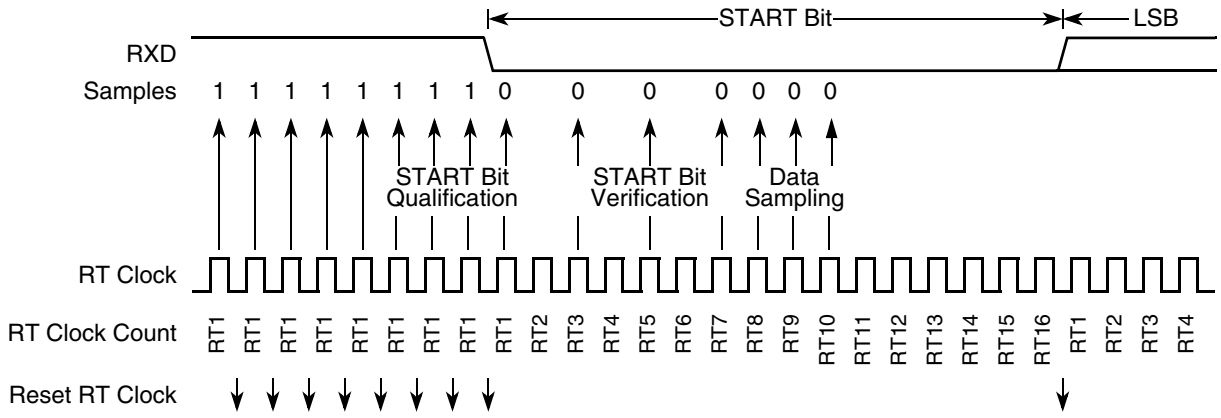


Figure 12-5. Receiver Data Sampling

To verify the START bit and detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 12-4 summarizes the results of the START bit verification samples.

Table 12-4. 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, data recovery logic takes samples at RT8, RT9, and RT10. Table 12-5 summarizes the results of DATA bit samples.

Table 12-5. 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 1s following a successful START bit verification, the noise flag (NF) is set and the receiver assumes the bit is a START bit (logic 0).

To verify a STOP bit and to detect noise, data recovery logic takes samples at RT8, RT9, and RT10. Table 12-6 summarizes the results of the STOP bit samples.

Table 12-6. 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 Figure 12-6 the verification samples RT3 and RT5 determine 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.

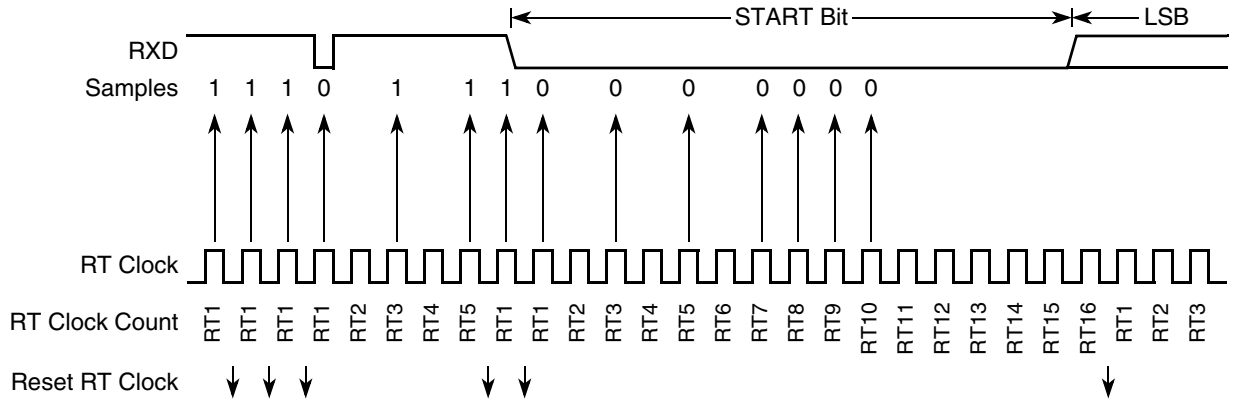


Figure 12-6. Start Bit Search Example 1

In Figure 12-7 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.

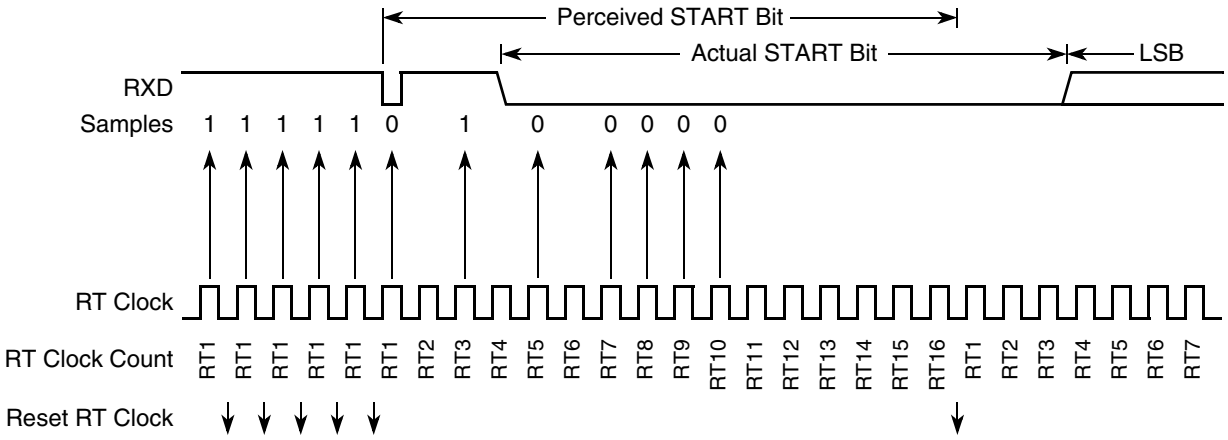


Figure 12-7. Start Bit Search Example 2

In Figure 12-8 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.

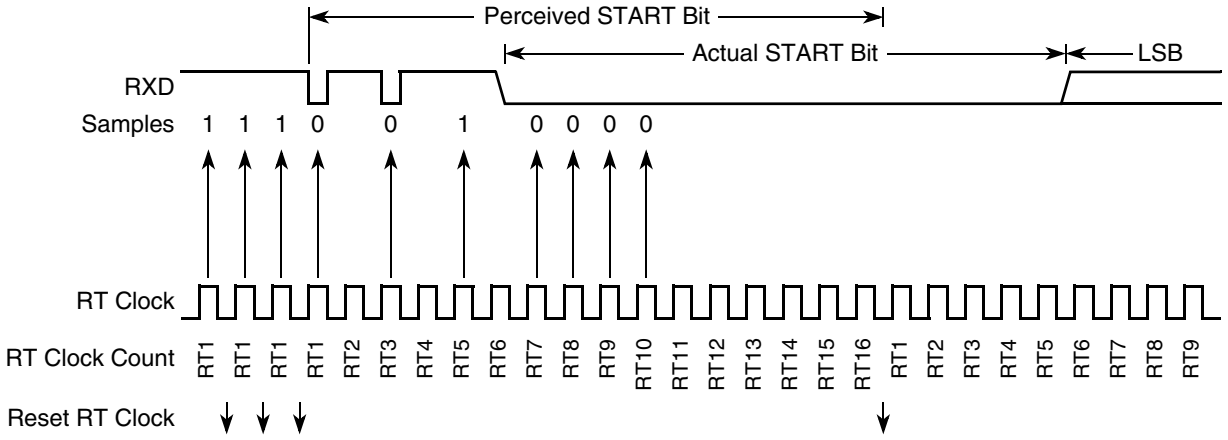


Figure 12-8. Start Bit Search Example 3

Figure 12-9 illustrates 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.

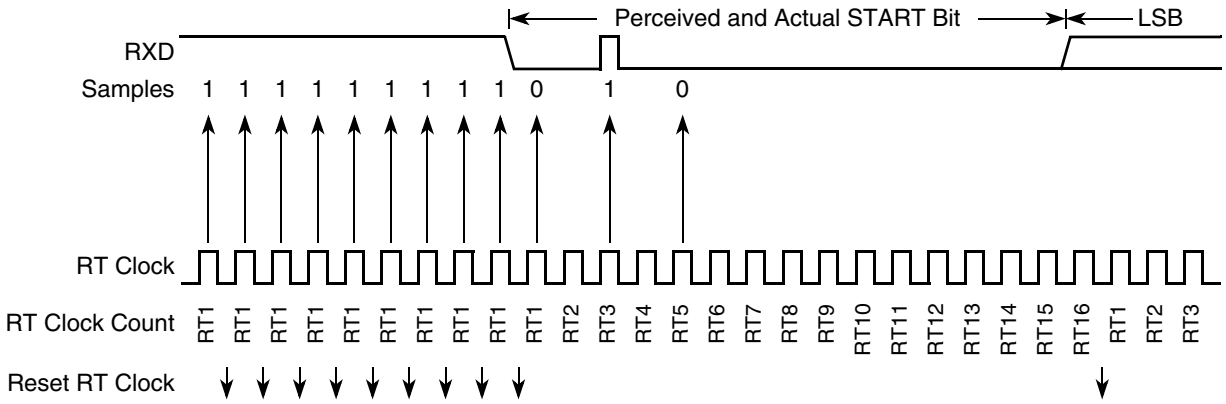


Figure 12-9. Start Bit Search Example 4

Figure 12-10 delineates a burst of noise near the beginning of the start bit, resetting the RT clock. The sample after the reset is low but is not preceded by three high samples qualifying 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.

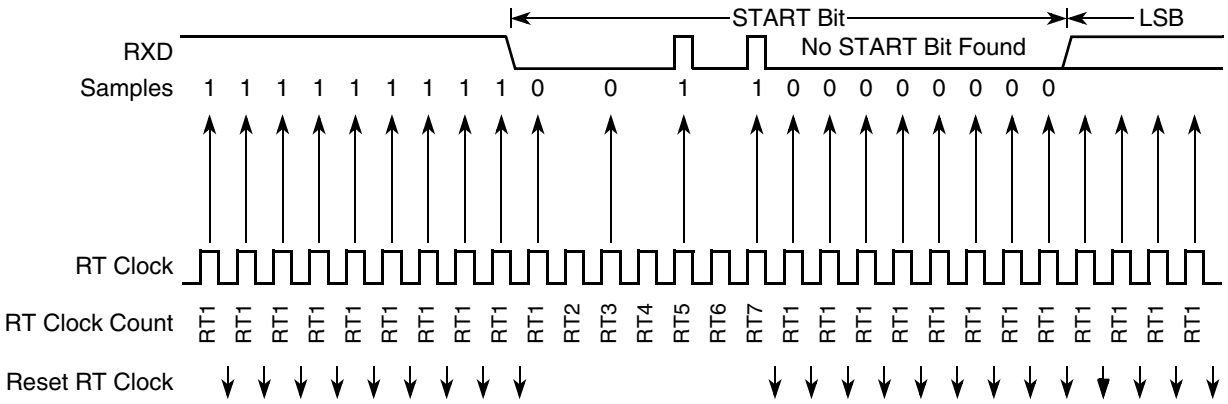


Figure 12-10. Start Bit Search Example 5

In Figure 12-11 a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but it does not reset the RT clock. In START bits only, the RT8, RT9, and RT10 data samples are ignored.

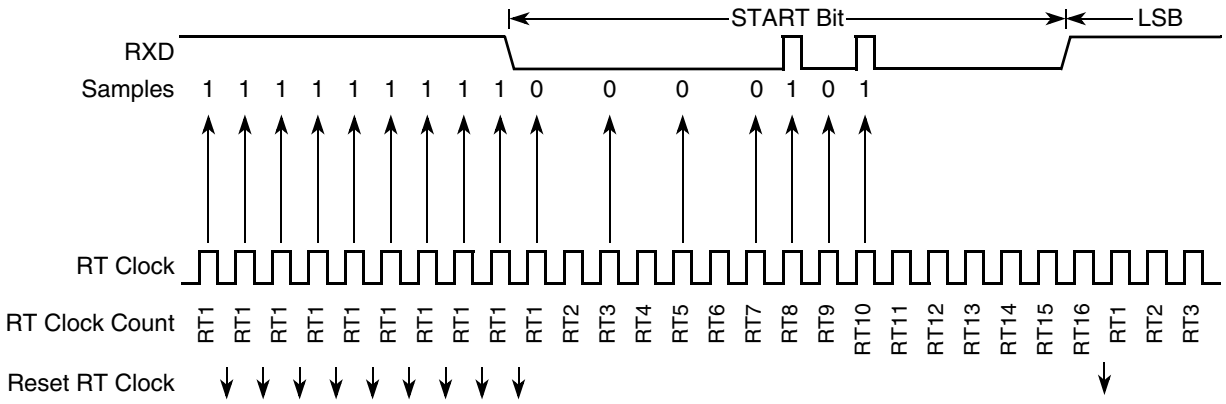


Figure 12-11. Start Bit Search Example 6

12.4.4.4 Framing Errors

If the data recovery logic does not detect a logic 1 where the STOP bit should be in an incoming frame, it sets the framing error (FE) flag in STAT register. A break character also sets the FE flag because a break character has no STOP bit. The FE flag is set concurrently with the RDRF flag.

12.4.4.5 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 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.

12.4.4.5.1 Slow Data Tolerance

Figure 12-12 illustrates how much a slow received frame can be misaligned without causing a noise or framing error. The slow STOP bit begins at RT8 instead of RT1, but it arrives in time for the STOP bit data samples at RT8, RT9, and RT10.

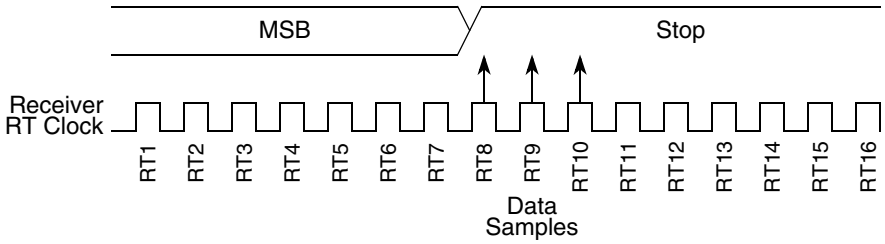


Figure 12-12. Slow Data

For an 8-bit data character, data sampling of the STOP bit takes the receiver:

9-bit × 16 RT cycles + 10 RT cycles = 154 RT cycles

With the misaligned character, illustrated in Figure 12-12**,** the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

9-bit × 16 RT cycles + 3 RT cycles = 147 RT cycles

The maximum percentage difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

$$\left|\frac{154 - 147}{154}\right| \times 100 = 4.54\%$$

For a 9-bit data character, data sampling of the STOP bit takes the receiver:

10-bit × 16 RT cycles + 10 RT cycles = 170 RT cycles

With the misaligned character illustrated in Figure 12-12**,** the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

10-bit × 16 RT cycles + 3 RT cycles = 163 RT cycles

With the misaligned character delineated in Figure 12-12, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles

The maximum percentage difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$\left|\frac{170 - 163}{170}\right| \times 100 = 4.12\%$$

12.4.4.5.2 Fast Data Tolerance

Figure 12-13 illustrates 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 it is still sampled at RT8, RT9, and RT10.

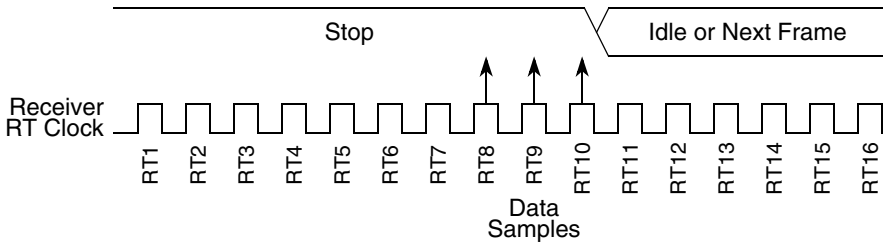


Figure 12-13. Fast Data

For an 8-bit data character, data sampling of the STOP bit takes the receiver:

9-bit × 16 RT cycles + 10 RT cycles = 154 RT cycles

With the misaligned character, illustrated in Figure 12-13, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

10-bit × 16 RT cycles = 160 RT cycles

The maximum percentage difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.90\%$$

For a 9-bit (all 0s or 1s) data character, data sampling of the STOP bit takes the receiver:

10-bit × 16 RT cycles + 10 RT cycles = 170 RT cycles

With the misaligned character, illustrated in Figure 12-13, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

11-bit × 16 RT cycles = 176 RT cycles

The maximum percentage difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$\left|\frac{170 - 176}{170}\right| \times 100 = 3.53\%$$

12.4.4.6 Receiver Wakeup

In order for the QSCI to ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup (RWU) bit in the CTRL1 register puts the receiver into a standby state while 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.

The WAKE bit in the CTRL1 register determines how the QSCI is brought out of the standby state to process an incoming message. The WAKE bit enables either idle line wakeup or address mark wakeup:

- Idle input line wakeup (WAKE = 0)—In this wakeup method, an idle condition on the RXD pin clears the RWU bit, waking up the QSCI. The initial frame, or frames of every message contain addressing information. All receivers evaluate the addressing information. Receivers of the message then process the following frames. Any receiver a message does not address can set its RWU bit, returning to the standby state. The RWU bit remains set and the receiver remains on standby until another preamble appears on the RXD pin.

Idle line wakeup require messages be separated by at least one preamble and no message contains preambles.

The preamble waking a receiver does not set the receiver idle bit (RIDLE), nor the receive data register full (RDRF) flag.

- Address mark wakeup (WAKE = 1)—In this wakeup method, a logic 1 in the MSB position of a frame clears the RWU bit, awakening the QSCI. The logic 1 in the MSB position marks a frame as an address frame containing addressing information. All receivers evaluate the addressing information. Receivers of the message then process the following frames. Any receiver a message does not address can set its RWU bit, returning to the standby state. The RWU bit remains set and the receiver remains on standby until another address frame appears on the RXD pin.

The logic 1 MSB of an address frame clears the receiver's RWU bit before the STOP bit is received and sets the RDRF flag.

Address mark wakeup allows messages to contain preambles but requires the MSB be reserved for use in address frames.

NOTE:

With the WAKE bit clear, setting the RWU bit after the RXD pin is idle can cause the receiver to wakeup immediately.

12.5 Operating Modes

12.5.1 Single-Wire Operation

Normally, the QSCI uses two pins for transmitting and receiving. In single-wire operation, the RXD pin is disconnected from the QSCI, making it available as a general-purpose I/O (GPIO) pin. The QSCI uses the TXD pin for both receiving and transmitting. Please see Figure 12-14.

Setting the TE bit in the CTRL1 register configures TXD as the output for transmitted data. Clearing the TE bit configures TXD as the input for received data.

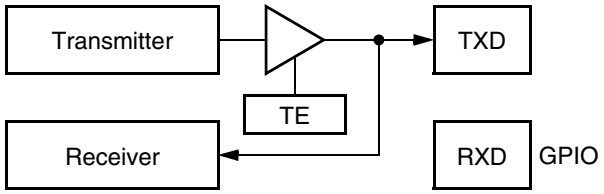


Figure 12-14. Single-Wire Operation (LOOP = 1, RSRC = 1)

Enable single-wire operation by setting the LOOP bit and the receiver source (RSRC) bit in the CTRL1 register. Setting the LOOP bit disables the path from the RXD pin to the receiver. Setting the RSRC bit connects the receiver input to the output of the TXD pin driver.

12.5.2 Loop Operation

In loop operation, the transmitter output goes to the receiver input. The RXD pin is disconnected from the QSCI and is available as a GPIO pin. Please see Figure 12-15.

Setting the TE bit in the CTRL1 register connects the transmitter output to the TXD pin. Clearing the TE bit disconnects the transmitter output from the TXD pin.

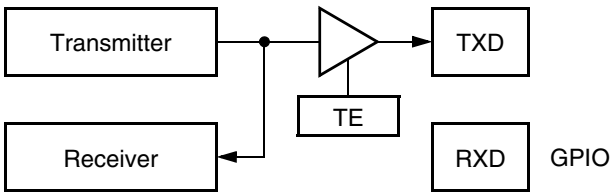


Figure 12-15. Loop Operation (LOOP = 1, RSRC = 0)

Enable loop operation by setting the LOOP bit and clearing the RSRC bit in the CTRL1 register. Setting the LOOP bit disables the path from the RXD pin to the receiver. Clearing the RSRC bit connects the transmitter output to the receiver input. To enable loop operation both the transmitter enable (TE) and receiver enable (RE) bits in the CTRL1 register must be set (TE = 1 and RE = 1).

12.5.3 LIN Slave Operation

LIN slave operation occurs when the LIN MODE bit is set in CTRL2 register. The receiver searches for a break character (a start bit, eight bits of data all 0, and a 0 in the STOP bit location). Once a break character is detected (11 consecutive samples of logic 0), the next field to be received is the synch field. The synch field is a word with 0×55 data which produces an alternating 0 and 1 pattern. The receiver detects the falling edge at the beginning of

the START bit and begins counting system clocks until the falling edge at the beginning of data bit seven is detected at which time it stops counting. This count is divided by eight (for the passed 8-bit periods) and further divided by 16 to provide new SBR and FRAC_SBR values. If the data value of the synch field is 0 × 55, this new SBR and FRAC_SBR value are placed in the baud rate register and the slave is considered synced to the master. Further data words will be received properly. If the data value of the synch field is not 0 × 55, the LIN sync error (LSE) bit is set in the STAT register and subsequent received data bytes should be ignored. The receiver resumes searching for a break character as above.

In order for the break character to be successfully detected, the initial baud rate for this slave device must be within 15% of the nominal baud rate of the LIN master device.

12.5.4 Low-Power Options

12.5.4.1 Run Mode

Clearing the transmitter enable (TE) or receiver enable (RE) bits in the CTRL1 register reduces power consumption in run mode. QSCI registers are still accessible when TE or RE is cleared, but clocks to the core of the QSCI are disabled.

12.5.4.2 Wait Mode

QSCI operation in wait mode depends on the state of the SWAI bit in the CTRL1 register.

- If SWAI is clear, the QSCI operates normally when the CPU is in wait mode.
- If SWAI is set, QSCI clock generation ceases and the QSCI module enters a power-conservation state when the CPU is in wait mode. In this condition, QSCI registers are not accessible. Setting SWAI does not affect the states of the receiver enable (RE) nor the transmitter enable (TE) bits.

  If 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 CPU out of wait mode. Exiting wait mode by reset aborts any transmission or reception in progress, resetting the QSCI.

12.5.4.3 Stop Mode

The QSCI operation in stop mode depends on the state of the QSCI_SD bit field in the SIM stop disable register.

- If QSCI_SD is clear, the QSCI is inactive in stop mode for reduced power consumption. The STOP instruction does not affect the QSCI register states. QSCI 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 QSCI.
- If QSCI_SD is set, the QSCI operates normally in stop mode.

12.6 Register Description

A register address is the sum of a base address and an address offset. The base address is defined at the device level and the address offset is defined at the module level.

Table 12-7. Memory Map

Device	Peripheral	Base Address
56F80xx	QSCI0	$00F200
	QSCI1	$00F210

Table 12-8 lists the QSCI registers in ascending address order, including their acronyms and address offset of each register.

Table 12-8. QSCI Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	RATE	Baud rate register	Read/Write	Section 12.6.1
Base + $1	CTRL1	Control 1 register	Read/Write	Section 12.6.2
Base + $2	CTRL2	Control 2 register	Read/Write	Section 12.6.3
Base + $3	STAT	Status register	Read-Only	Section 12.6.4
Base + $4	DATA	Data register	Read/Write	Section 12.6.5

There are five registers in the QSCI peripheral summarized in Figure 12-16**.** Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	RATE	R	SBR													FRAC_SBR		
		W																
$1	CTRL1	R	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
		W																
$2	CTRL2	R	TFCNT			TFWM		RFCNT			RFWM		FIFO_EN	0	LIN MODE	0	0	0
		W																
$3	STAT	R	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0	0	0	0	LSE	0	0	RAF
		W																
$4	DATA	R	0	0	0	0	0	0	0	RECEIVE DATA								
		W								TRANSMIT DATA								

R	0	Read as 0
W		Reserved

Figure 12-16. QSCI Register Map Summary

12.6.1 Baud Rate (RATE) Register

This register can be read and written at any time.

Base + $0	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

Figure 12-17. Baud Rate (RATE) Register

12.6.1.1 QSCI Baud Rate (SBR)—Bits 15–3

The SBR value is from 1 to 8191.

The count in this register determines the baud rate of the QSCI. The formula for calculating baud rate is:

$$\text{QSCI Baud Rate} = \frac{\text{QSCI Module Clock}}{16 \times (\text{SBR} + (\text{FRAC\_SBR}/8))}$$

NOTE:

The baud rate generator is disabled until the TE or the RE bits are set for the first time after reset. The baud rate generator is disabled when SBR = 0.

12.6.1.2 Fractional QSCI Baud Rate (FRAC_SBR)—Bits 2–0

This field combines with SBR field (Section 12.6.1.1) to form the divider, determining the baud rate of the QSCI. SBR represents the integer portion of the baud rate divider while FRAC_SBR represents the fractional portion.

NOTE:

If the LIN mode bit of CTRL2 register is set, the value of this register is automatically adjusted to match the data rate of the LIN master deice. Reading this register yields the auto baud value set.

12.6.2 Control 1 (CTRL1) Register

The control 1 (CTRL1) register can be read and written at anytime.

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-18. Control 1 (CTRL1) Register

12.6.2.1 Loop Select (LOOP)—Bit 15

This bit enables loop operation. Please see Section 12.5.2. The loop operation disconnects the RXD pin from the QSCI 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, only requiring one or the other to be enabled. Please see Section 12.5.1 and Table 12-9.

- 0 = Normal operation enabled
- 1 = Loop operation enabled

The receiver input is determine by the RSRC bit. The transmitter output is controlled by the TE bit. If the TE bit is set and LOOP = 1, the transmitter output appears on the TXD pin. If the TE bit is clear and LOOP = 1, the TXD pin is high-impedance.

Table 12-9. Loop Functions

Loop	RSRC	Function
0	x	Normal operation
1	0	Loop mode with internal TXD fed back to RXD
1	1	Single-wire mode with TXD output fed back to RXD

12.6.2.2 Stop in Wait Mode (SWAI)—Bit 14

The SWAI bit disables the QSCI in the wait mode. Please see Section 12.5.4.2.

- 0 = QSCI enabled in wait mode
- 1 = QSCI disabled in wait mode

12.6.2.3 Receiver Source (RSRC)—Bit 13

When LOOP = 1, the RSRC bit determines the internal feedback path for the receiver, indicated in Table 12-9.

- 0 = Receiver input internally connected to transmitter output
- 1 = Receiver input connected to TXD pin

12.6.2.4 Data Format Mode (M)—Bit 12

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

12.6.2.5 Wakeup Condition (WAKE)—Bit 11

This bit determines which condition wakes up the QSCI.

- 0 = Idle line wakeup
- 1 = Address mark wakeup (a logic 1 in the MSB position of a receive data character)

12.6.2.6 Polarity (POL)—Bit 10

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.

- 0 = Doesn’t invert transmit and receive data bits (normal mode)
- 1 = Invert transmit and receive data bits (inverted mode)

NOTE:

It is recommended the POL bit be toggled only when both TE and RE = 0.

12.6.2.7 Parity Enable (PE)—Bit 9

This bit enables the parity function. When enabled, the parity function replaces the MSB of the data character with a parity bit.

- 0 = Parity function disabled
- 1 = Parity function enabled

12.6.2.8 Parity Type (PT)—Bit 8

This bit determines whether the QSCI generates and checks for even or odd parity of the data bits. With even parity, an even number of ones clear the PARITY bit while an odd number of ones, sets the PARITY bit. However, with odd parity, an odd number of ones clear the PARITY bit while an even number of ones, sets the PARITY bit.

- 0 = Even parity
- 1 = Odd parity

12.6.2.9 Transmitter Empty Interrupt Enable (TEIE)—Bit 7

This bit enables the TDRE bit to generate interrupt requests.

- 0 = TDRE interrupt requests disabled
- 1 = TDRE interrupt requests enabled

12.6.2.10 Transmitter Idle Interrupt Enable (TIIE)—Bit 6

This bit enables the TIDLE bit to generate interrupt requests.

- 0 = TIDLE interrupt requests disabled
- 1 = TIDLE interrupt requests enabled

12.6.2.11 Receiver Full Interrupt Enable (RFIE)—Bit 5

This bit enables the RDRF bit or the OR bit to generate interrupt requests.

- 0 = RDRF and OR interrupt requests disabled
- 1 = RDRF and OR interrupt requests enabled

12.6.2.12 Receive Error Interrupt Enable (REIE)—Bit 4

This bit enables the receive error (RE) bits (NF, PF, FE, LSE, and OR) to generate interrupt requests. The status bits can be checked during the error interrupt process.

- 0 = Error interrupt requests disabled
- 1 = Error interrupt requests enabled

12.6.2.13 Transmitter Enable (TE)—Bit 3

This bit enables the QSCI transmitter and configures the TXD pin as the QSCI transmitter output. The TE bit can be used to queue an idle preamble.

- 0 = Transmitter disabled
- 1 = Transmitter enabled

12.6.2.14 Receiver Enable (RE)—Bit 2

This bit enables the QSCI receiver.

- 0 = Receiver disabled
- 1 = Receiver enabled

12.6.2.15 Receiver Wakeup (RWU)—Bit 1

This bit enables the wakeup function, inhibiting further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU. Please refer to Section 12.4.4.6 for a description of receive wakeup.

- 0 = Normal operation
- 1 = Standby state

12.6.2.16 Send Break (SBK)—Bit 0

Toggling SBK sends one break character (10 or 11 logic 0s) as long as SBK is set, the transmitter sends uninterrupted break characters.

- 0 = No break characters
- 1 = Transmit break characters

12.6.3 Control 2 (CTRL2) Register

This register contains selected bits capable of being read and written at any time.

Base + $2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TFCNT			TFWM		RFCNT			RFWM		FIFO_EN	0	LIN MODE	0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-19. Control 2 (CTRL2) Register

12.6.3.1 Transmit FIFO Count (TFCNT)—Bits 15–13

These read-only bits exhibit how many words are used in the TX FIFO. Writes to DATA will cause TFCNT to increment. A decrease takes place as words are pulled for transmission. Attempts to write new data to DATA are ignored when TFCNT indicates FIFO is full (four words).

12.6.3.2 Transmit FIFO Empty Watermark (TFWM)—Bits 12–11

These bits set the TX FIFO word count level, thereby setting the TDRE flag. If FIFO_EN is clear (FIFO disabled), this field is forced to 00 and TDRE is set when there is no word in the transmit buffer.

12.6.3.3 Receive FIFO Count (RFCNT)—Bits 10–8

Table 12-10. TFWM Encoding

TFWM[1:0]	TDRE is Set When...
00	0 words are in the FIFO
01	1 or less words are in the FIFO
10	2 or less words are in the FIFO
11	3 or less words are in the FIFO

These read-only bits illustrate how many words are used in the RX FIFO. Received words cause RFCNT to increment. RFCNT values decrease as words are read from DATA. There is one word-time to read DATA between when the RDRF status bit is set (interrupt asserted) due to the RX FIFO being full and when an overflow condition will be flagged.

12.6.3.4 Receive FIFO Full Watermark (RFWM)—Bits 7–6

These bits set the RX FIFO word count level, thereby setting the RDRF flag. If FIFO_EN is clear (FIFO disabled), then this field is forced to 00 and RDRF is set if there is a word in the Receive Buffer.

Table 12-11. RFWM Encoding

RFWM[1:0]	RDRF is Set When...
00	at least 1 word is in the FIFO
01	at least 2 words are in the FIFO
10	at least 3 words are in the FIFO
11	at least 4 words are in the FIFO

12.6.3.5 Reserved—Bit 4

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

12.6.3.6 Local Interconnect Network Mode (LIN MODE)—Bit 3

This bit should only be used in local interconnect network (LIN) applications.

- 0 = The LIN auto baud feature is disabled and the SBR register will maintain whatever value the processor writes to it.
- 1 = Enable a search for the break followed by sync char (0x55) from the master LIN device. When the break is detected the following sync character will be used to measure the baud rate of the transmitting master and the SBR register will be automatically reloaded with the value needed to match that baud rate.

NOTE:

During initialization the SBR register should be loaded to a value that is within 15% of the actual master data rate, otherwise 0x00 data may be misinterpreted as a break.

If the first character following a break is not the LIN sync character (0x55) the SBR will not be adjusted.

12.6.3.7 Reserved—Bits 2–0

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

12.6.4 Status (STAT) Register

This register can be read at anytime; however, it cannot be modified by writing. Writes clear flags.

Base + $3	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	0	0	LSE	0	0	RAF
Write																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-20. Status (STAT) Register

12.6.4.1 Transmit Data Register Empty Flag (TDRE)—Bit 15

This bit is set when the transmit shift register receives a character from the DATA register. Clear TDRE by reading the STAT register, then write to the DATA register.

- 0 = TX FIFO word count is above watermark
- 1 = TXFIFO word count is at, or below watermark

12.6.4.2 Transmitter Idle Flag (TIDLE)—Bit 14

This bit is set when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TIDLE is set, the TXD pin becomes idle (logic 1). Clear TIDLE by reading the STAT register with TIDLE set and then writing to the 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

12.6.4.3 Receive Data Register Full Flag (RDRF)—Bit 13

RDRF is set when the RX FIFO word count (RFCNT) rises above the watermark (RXWM). Clear RDRF by reading the status (STAT) register with RDRF set and then reading the QSCI data register until RFCNT is no longer above RFWM. If FIFO_EN is set, then you can clear RDRF by reading the QSCI data register without first reading STAT with RDRF set.

- 0 = RX FIFO word count is at, or below watermark
- 1 = RX FIFO word count is above watermark

NOTE:

The RDRF flag may be erased when a breakpoint is reached when using the MetroWerks debugger. If a memory window, including the QSCI registers, is open when a breakpoint is reached, these memory addresses will be read to update the memory window. When the RDRF bit is set at this time, these reads satisfy the requirements for clearing RDRF in that the STAT register is read with RDRF set, resulting in the DATA register to be read, causing RDRF to clear.

12.6.4.4 Receiver Idle Line Flag (RIDLE)—Bit 12

This bit is set when ten consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. The RIDLE flag is cleared by reading the STAT register with RIDLE set followed by reading the DATA register. Once the RIDLE flag is cleared, a valid frame must be received before an idle condition can set the RIDLE flag.

- 0 = Receiver input is either active now or was never active since the RIDLE flag was last cleared by reset
- 1 = Receiver input is now idle (after receiving a valid frame)

NOTE:

When the receiver wakeup (RWU) bit is set, an idle line condition does not set the RIDLE flag.

12.6.4.5 Overrun (OR)—Bit 11

This bit is set when software fails to read the 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 DATA/FIFO register is not affected. Clear OR by reading the STAT register with OR set and then write the STAT register with any value.

- 0 = No overrun
- 1 = Overrun

12.6.4.6 Noise Flag (NF)—Bit 10

This bit is set when the QSCI detects noise on the receiver input. The NF bit is set during the same cycle as the RDRF flag, but it is not set in the case of an overrun. Clear NF by reading the STAT register, and then write the STAT register with any value.

- 0 = No noise
- 1 = Noise

12.6.4.7 Framing Error (FE)—Bit 9

This bit is set when a logic 0 is accepted as the STOP bit. The FE bit is set during the same cycle as the RDRF flag, but it is not set in the case of an overrun. Clear FE by reading the STAT register with FE set and then write the STAT register with any value.

- 0 = No framing error
- 1 = Framing error

12.6.4.8 Parity Error Flag (PF)—Bit 8

This bit is set when the parity enable (PE) bit is set and the parity of the received data does not match its parity bit. Clear PF by reading the STAT register, and then write the STAT register with any value.

- 0 = No parity error
- 1 = Parity error

12.6.4.9 Reserved—Bits 7–4

These bits are reserved or not implemented. They are read as zero and cannot be modified by writing.

12.6.4.10 Local Interconnect Network Sync Error (LSE)—Bit 3

This bit is only active when LIN mode is set. When LSE is set, an RERR interrupt will occur if REIE is set.

LSE is set when a LIN sync search detects a non-sync character (anything other than 0x55). Having this bit set indicates either 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 the STAT register with LSE set and then writing the STAT register with any value.

- 0 = No error occurred since the LIN mode was enabled or the bit was last cleared
- 1 = A sync error prevented loading of the SBR with a revised value after the break was detected

12.6.4.11 Reserved—Bits 2–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

12.6.4.12 Receiver Active Flag (RAF)—Bit 0

This bit is set when the receiver detects a logic 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

12.6.5 Data (DATA) Register

The DATA register can be read and modified at any time. Reading accesses the receive data register. Writing to the register accesses the transmit data register.

Base + $4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	RECEIVE DATA								
Write								TRANSMIT DATA								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-21. Data (DATA) Register

12.6.5.1 Reserved—Bits 15–9

These bits are reserved or not implemented. They are read as zero and cannot be modified by writing.

12.6.5.2 Receive/Transmit Data—Bits 8–0

Reading these bits accesses the receive data. Writing to these bits loads the transmit data.

When the TDRE bit is set, writes to the transmit data field are ignored unless the STAT register is read while TDRE is set. When FIFO_EN is set it is possible to write to the DATA register without first reading STAT with TDRE set.

If the RDRF bit is set, reads from the receive data field are ignored unless the STAT register is read while RDRF is set. When FIFO_EN is set it is possible to read from the DATA register without first reading STAT with RDRF set.

12.7 Clocks

All timing is derived from the IPBus clock, the main clock for this module. See Section 12.4.2 for a description of how the data rate is determined.

12.8 Interrupts

12.8.1 Description of Interrupt Operation

Figure 12-22. QSCI Interrupt Summary

Interrupt	Source	Interrupt Vector	Description
TDRE	Transmitter	QSCI_Vector_Base + $0	Transmit data register empty interrupt
TIDLE		QSCI_Vector_Base + $1	Transmit idle interrupt
RERR	Receive	QSCI_Vector_Base + $3	Receive error (FE, NF, PF, LSE, or OR) interrupt
RDRF/OR		QSCI_Vector_Base + $4	Receive data register full/overrun interrupt

Figure 12-23. QSCI Interrupt Sources

Flag	Source	Local Enable	Description
TDRE	Transmitter	TEIE	Transmit data register empty interrupt
TIDLE		TIIE	Transmit idle interrupt
RDRF	Receiver	RFIE	Receive data register full/overrun interrupt
OR			
FE		REIE	Receive error (FE, NF, PF, LSE, or OR) interrupt
PE			
NF			
OR			
LSE			

12.8.1.1 Transmitter Empty Interrupt

This interrupt is enabled by setting the TEIE bit of the CTRL1 register. When this interrupt is enabled an interrupt is generated when data is transferred from the DATA register to the transmit shift register. The interrupt service routine should read the STAT register, verifying the TDRE bit is set, and then write the next data to be transmitted to the DATA register, clearing the TDRE bit.

12.8.1.2 Transmitter Idle Interrupt

This interrupt is enabled by setting the TIIE bit of the CTRL1 register. This interrupt indicates the TDRE flag is set and the transmitter is no longer sending data, preamble, or break characters. The interrupt service routine should read the STAT register, verifying the TIDLE bit is set and then initiate the preamble, break, or write a data character to the DATA register. Any of these actions clears the TIDLE bit because the transmitter is busy.

12.8.1.3 Receiver Full Interrupt

This interrupt is enabled by setting the RFIE bit of the CTRL1 register. This interrupt indicates receive data is available in the DATA register. The interrupt service routine should read the STAT register, verifying the RDRF bit is set and then read the data from the STAT register, clearing the RDRF bit.

12.8.1.4 Receive Error Interrupt

This interrupt is enabled by setting the REIE bit of the CTRL1 register. This interrupt indicates any of the listed errors were detected by the receiver.

- Noise flag (NF) set
- Parity error flag (PF) set
- Framing error (FE) flag set
- Overrun (OR) flag set
- LIN synchronization error (LSE) flag set

The interrupt service routine should read the STAT register to determine which of the error flags is set. The error flag is cleared by writing anything to the STAT register. The appropriate action should then be taken by the software to handle the error condition.

12.8.1.5 Recover From Wait or Stop Mode

Any enabled QSCI interrupt request can bring the CPU out of wait or stop mode.

12.9 Resets

Any system reset completely resets the QSCI.

Table 12-12. Document Revision History for Chapter 12

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 13
Queued Serial Peripheral Interface (QSPI)

13.1 Introduction

This chapter describes the queued serial peripheral interface (QSPI) module. The module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other 16-bit controllers. Software can poll the QSPI status bits or QSPI operation can be interrupt driven. The block contains six, 16-bit memory mapped registers for control parameters, status, and data transfer.

13.2 Features

Characteristics of the QSPI module include:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four-word deep FIFOs available on transmit and receive buffers
- Programmable length transmissions (two to 16 bits)
- Programmable transmit and receive shift order (MSB as first bit transmitted)
- Nine master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency < bus frequency ÷ 2
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts
  - — QSPI receiver full/error, supporting three interrupt bits:
    - – Receiver full
    - – Mode fault error bit
    - – Overflow error bit
  - — SPTE (QSPI transmitter empty)
- Wired OR mode functionality enabling connection to multiple QSPIs

NOTE:

Throughout this chapter, there are references to the QSPI module clock. The QSPI module clock is the IPBus clock. It is equal to the system clock.

13.3 Block Diagram

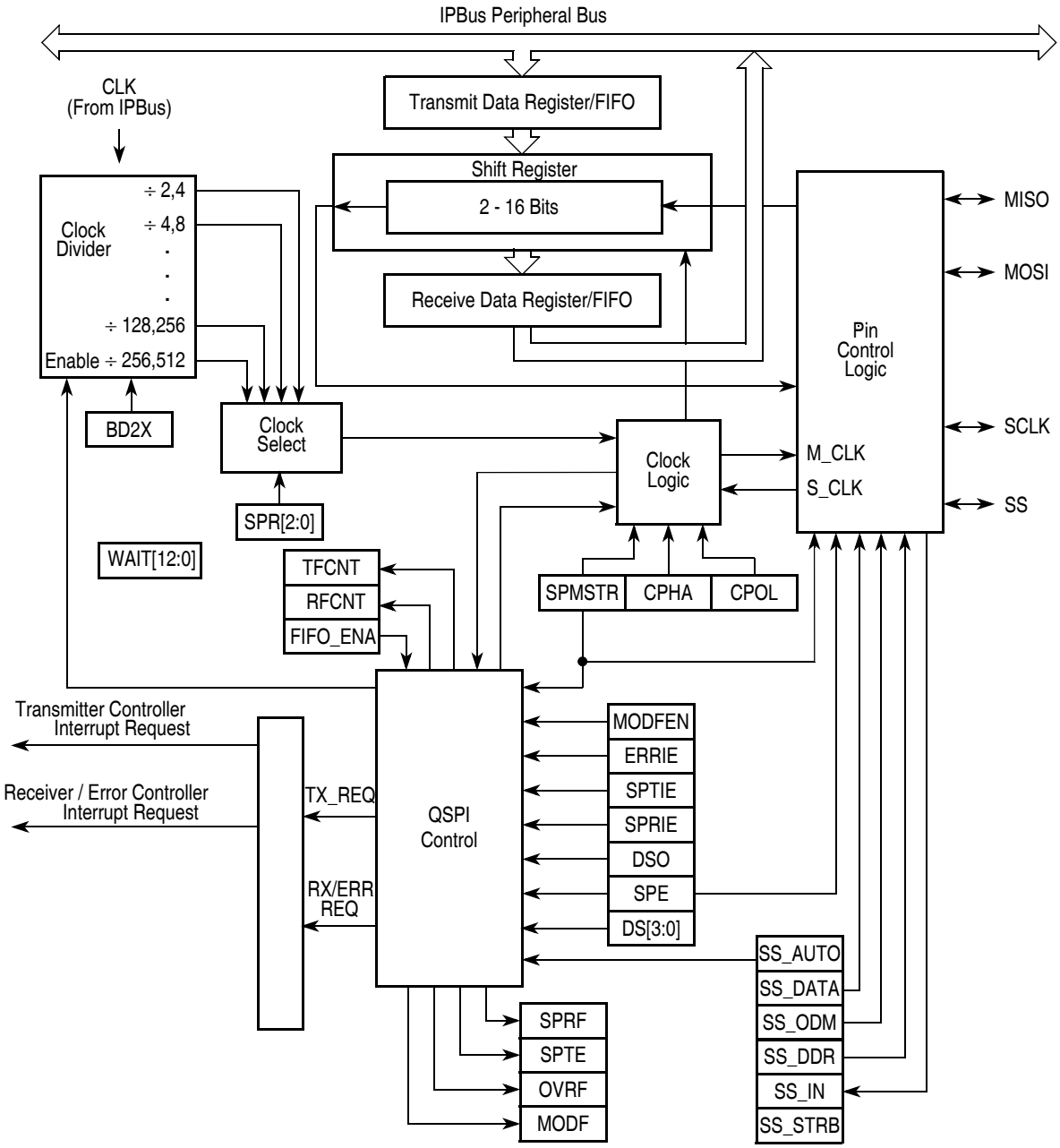


Figure 13-1. QSPI Block Diagram

13.4 Operating Modes

The QSPI has two operating modes:

- Master
- Slave

QSPI operates in master mode when the QSPI master bit (SPMSTR) is set.

NOTE:

Configure the QSPI module as master or slave before enabling the QSPI. Enable the master QSPI before enabling the slave QSPI. Disable the slave QSPI before disabling the master QSPI.

An operating mode is selected by the SPMSTR bit in the QSPI status and control (CTRL) register as follows:

- SPMSTR = 0 (slave mode)
- SPMSTR = 1 (master mode)

Configuration steps for the typical QSPI slave mode:

- Set each of the QSPI status and control (SCTRL) register bits as required for the application to match the settings of the master mode device (SPE = 0)
- Set the data size and control (DSCTRL) register as required for the application
- Enable the device set SPE = 1
- Store the empty character in the data transmit (DXMIT) register

Configuration steps for typical QSPI master mode:

- Set the QSPI status and control (SCTRL) register bits as required by the application and possibly limited by the characteristics of the slave device (SPE = 0)
- Set the data size and control (DSCTRL) register bits as required by the application
- Initialize the software pointers and buffers
- Enable the device set SPE = 1

13.4.1 Master Mode

Only a master QSPI module can initiate transmissions. With the QSPI enabled, software begins the transmission from the master QSPI module by writing to the data transmit (DXMIT) register. If the shift register is empty, the data immediately transfers to the shift register, setting the QSPI transmitter empty (SPTE) bit. Data begins shifting out on the master out/slave in (MOSI) pin under the control of the serial clock (SCLK).

The SPR2, SPR1, SPR0, and (BD2x) bits in the status and control (SCTRL) register control the baud rate generator, thereby determining 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 data shifts out on the MOSI pin of the master, external data shifts in from the slave on the master in/slave out (MISO) pin. The transmission ends when the receiver full (SPRF) bit in the SCTRL register becomes set. At the same time the SPRF becomes set, the data from the slave transfers to the data receive (DRCV) register. In a normal operation SPRF signals the end of a transmission. Software clears the SPRF by reading the DRCV. Writing to the DXMIT clears the SPTE bit.

Figure 13-2 is an example configuration for a full-duplex master/slave configuration. Having the slave select ($\overline{SS}$) bit of the master MCU held high is only necessary if MODFEN = 1. Tying the slave controller $\overline{SS}$ bit to ground should only be executed if CPHA = 1.

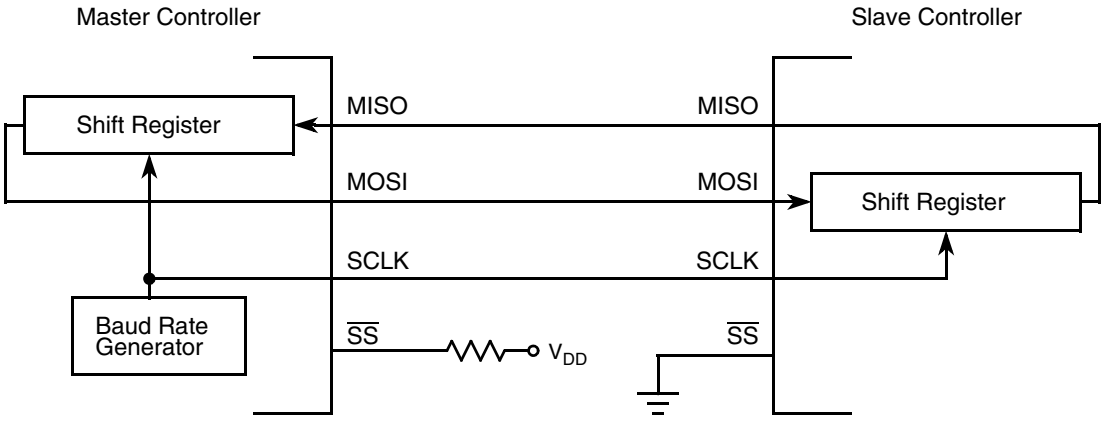


Figure 13-2. Full Duplex Master/Slave Connections

13.4.2 Slave Mode

The QSPI operates in slave mode when the SPMSTR bit is clear. While in slave mode, the SCLK pin acts as an input for the serial clock from the master controller. Before a data transmission occurs, the $\overline{SS}$ pin of the slave QSPI must be a logic 0. $\overline{SS}$ must remain low until the transmission is complete, or a mode fault error occurs.

NOTE:

The QSPI must be enabled (SPE = 1) for slave transmissions to be received.

Data in the slave's transmitter shift register is unaffected by SCLK transitions in the event the slave QSPI is deselected ($\overline{SS}$ = 1).

In a slave QSPI module, data enters the shift register under the control of the serial clock (SCLK) from the master QSPI module. After the last bit of a data transmission enters the shift register of a slave QSPI, the data transfers to the DRCV register and the receiver full (SPRF) bit in the SCTRL register is set. If the receive interrupt enable (SPRIE) bit in the SCTRL register is set, a receive Interrupt is also generated. To prevent an overflow condition, slave software must read the DRCV register before the last bit of the next full length data transmission enters the shift register.

The maximum frequency of the SCLK for the QSPI configured as a slave is less than half of the bus clock frequency. The frequency of the SCLK for a QSPI configured as a slave does not have to have to correspond to any QSPI baud rate as defined by the SPR bits.

When the master QSPI starts a transmission, the data in the slave DXMIT register is loaded to the shift register and begins shifting out on the MISO pin. The slave can reload its DXMIT register with new data for the next transmission by writing to its DXMIT register after the first bit is shifted out (SPTE = 1). The slave must write to its DXMIT at least one bus cycle before the master starts the next transmission. Otherwise, the data last transmitted is re-loaded into the slave shift register and shifts out on the MISO pin again. Data written to the slave DXMIT register during a transmission remains in a buffer until the end of the transmission.

When the clock phase (CPHA) bit is set, the first edge of SCLK starts a transmission. When CPHA is cleared the falling edge of $\overline{SS}$ starts a transmission.

NOTE:

The master QSPI's SCLK must be in the proper idle state (depends on setting of CPOL bit) before the slave is enabled otherwise it may appear as a clock edge. This preserves the proper SCLK, MISO, MOSI timing relationships.

13.4.3 Wired-OR Mode

Wired-OR functionality is provided to permit the connection of multiple QSPIs. Figure 13-3 illustrates a single master device controlling multiple slave QSPIs. When the WOM bit is set, the outputs switch from conventional complementary CMOS output to open drain outputs. This allows the internal pull-up resistor bring the line high, and whichever QSPI drives the line pulls it low as needed.

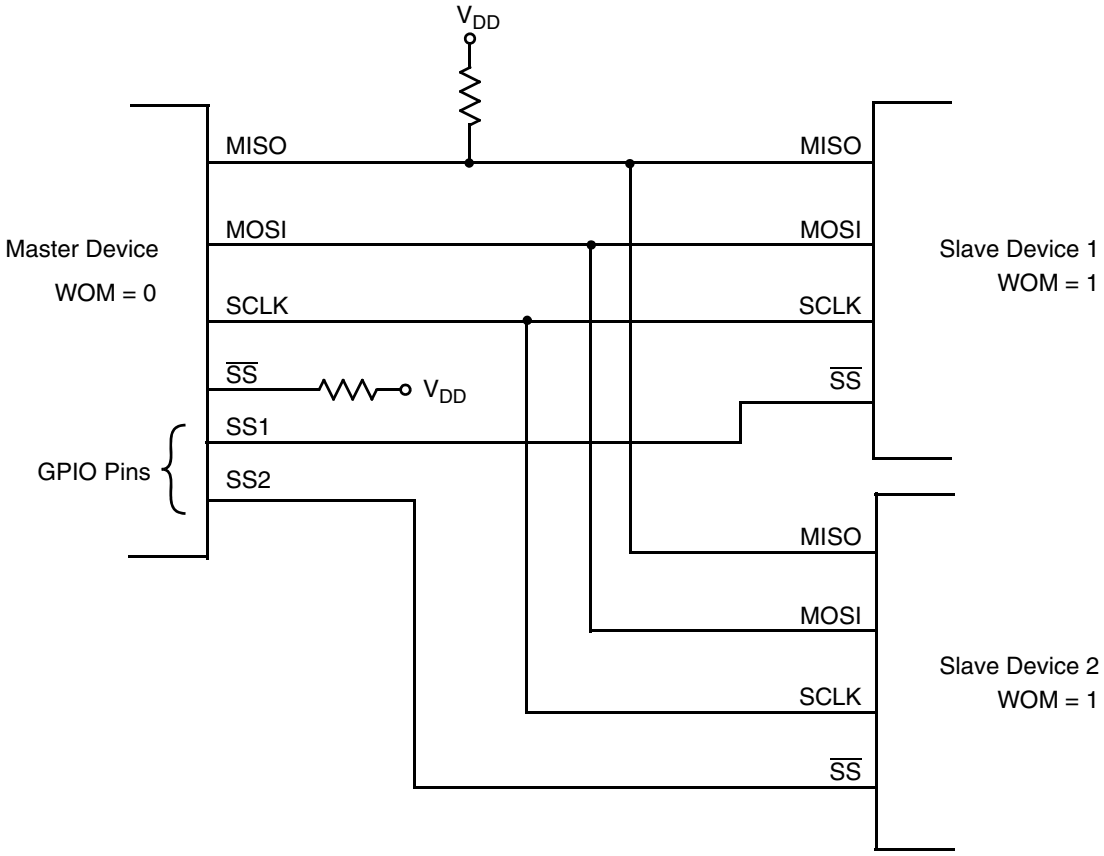


Figure 13-3. Master with Two Slaves

13.5 Pin Descriptions

There are four external QSPI pins. Each is summarized in Table 13-1.

Table 13-1. Summary External I/O Signals

Signal Name	Description	Direction
MISO	Master-in slave-out pad pin	Bidirectional
MOSI	Master-out slave-in pad pin	Bidirectional
SCLK	Serial clock pad pin	Bidirectional
$\overline{SS}$	Slave select pad pin (active low)	Bidirectional

13.5.1 Master In/Slave Out (MISO)

MISO is one of the two QSPI module pins dedicated to transmit serial data. In full duplex operation, the MISO pin of the master QSPI module is connected to the MISO pin of the slave QSPI module. The master QSPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin. The slave QSPI simultaneously transmits data on its MISO pin and receives data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the QSPI is configured as a slave. The QSPI is configured as a slave when the SPMSTR bit, discussed in Section 13.9.1, is logic 0 and its $\overline{SS}$ pin is logic 0. To support a multiple slave system, logic 1 on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

13.5.2 Master Out/Slave In (MOSI)

MOSI is the other QSPI module pin dedicated to transmit serial data. In full duplex operation, the MOSI pin of the master QSPI module is connected to the MOSI pin of the slave QSPI module. The master QSPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin. The slave QSPI simultaneously receives data from its MOSI pin and transmits data on its MISO pin.

13.5.3 Serial Clock (SCLK)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SCLK pin is the clock output. In a slave controller, the SCLK pin is the clock input. In full duplex operation, the master and slave MCU exchange data in the same number of clock cycles as the number of bits of transmitted data.

13.5.4 Slave Select ($\overline{SS}$)

The $\overline{SS}$ pin has various functions depending on the current state of the QSPI. For a QSPI configured as a slave, the $\overline{SS}$ is used to enable a slave. For clock phase (CPHA = 0), the $\overline{SS}$ in is used to define the start of a transmission. $\overline{SS}$ must be toggled high and low between each full length data word. However, the $\overline{SS}$ can remain low between data words for the CPHA = 1 format, illustrated in Figure 13-5.

When a QSPI 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:

Logic 1 voltage on the $\overline{SS}$ pin of a slave QSPI puts the MISO pin in a high-impedance state. The slave QSPI ignores all incoming SCLK clocks, even if it was already in the middle of a transmission. A mode fault error occurs if the $\overline{SS}$ pin changes state during a transmission and MODFEN = 1.

When a QSPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF bit to prevent multiple masters from driving MOSI and SCLK. For the state of the $\overline{SS}$ pin to set the MODF bit, the MODFEN bit in the SCLK register must be set.

If the system has only one master, then the master's $\overline{SS}$, when configured as a GPIO, can be used to select a slave.

Table 13-2. QSPI I/O Configuration

SPE	SPMSTR	MODFEN	QSPI Configuration	State of $\overline{SS}$ Logic
0	x	x	Not Enabled	$\overline{SS}$ ignored by QSPI
1	0	x	Slave	Input-only to QSPI

Table 13-2. QSPI I/O Configuration

SPE	SPMSTR	MODFEN	QSPI Configuration	State of $\overline{SS}$ Logic
1	1	0	Master without MODF	$\overline{SS}$ input ignored by QSPI. $\overline{SS}$ output may be activated under software or hardware control to select slave devices.
1	1	1	Master with MODF	Input-only to QSPI

x = Don't care

13.6 Transmission Formats

During a QSPI transmission, 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 QSPI device; slave devices not selected do not interfere with QSPI bus activities. On a master QSPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

13.6.1 Data Transmission Length

The QSPI can support data lengths from two to 16 bits. This can be configured in the data size and control (DSCTRL) register. When the data length is less than 16 bits, the DRCV register pads the upper bits with zeros.

NOTE:

Data is lost if the data length is not the same for both master and slave devices.

13.6.2 Data Shift Ordering

The QSPI 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 which bit is transmitted or received first, the data shall always be written to the DXMIT register and read from the DRCV register with the LSB in bit zero and the MSB in the correct position, depending on the data transmission size.

13.6.3 Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SCLK) phase and polarity using two bits in the SCTRL register. The clock polarity (CPOL) is specified by the control bit. This specification selects an active high or low clock but has no other significant effect on the transmission format.

The clock phase (CPHA) control bit selects one of two fundamentally different transmission formats. The clock phase and polarity should be identical for the master QSPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

NOTE:

Before writing to either the CPOL or CPHA bits, disable the QSPI by clearing the QSPI enable (SPE) bit.

13.6.4 Transmission Format When CPHA = 0

Figure 13-4 exhibits a QSPI transmission with CPHA as logic 0.

NOTE:

The figure should not be used as a replacement for data sheet parametric information.

Two waveforms for the SCLK are shown:

- CPOL = 0
- CPOL = 1

The waveforms 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/LSB 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 transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each full length data word transmission, as depicted in Figure 13-5.

NOTE:

Figure 13-4 assumes 16-bit data lengths and the MSB shifted out first.

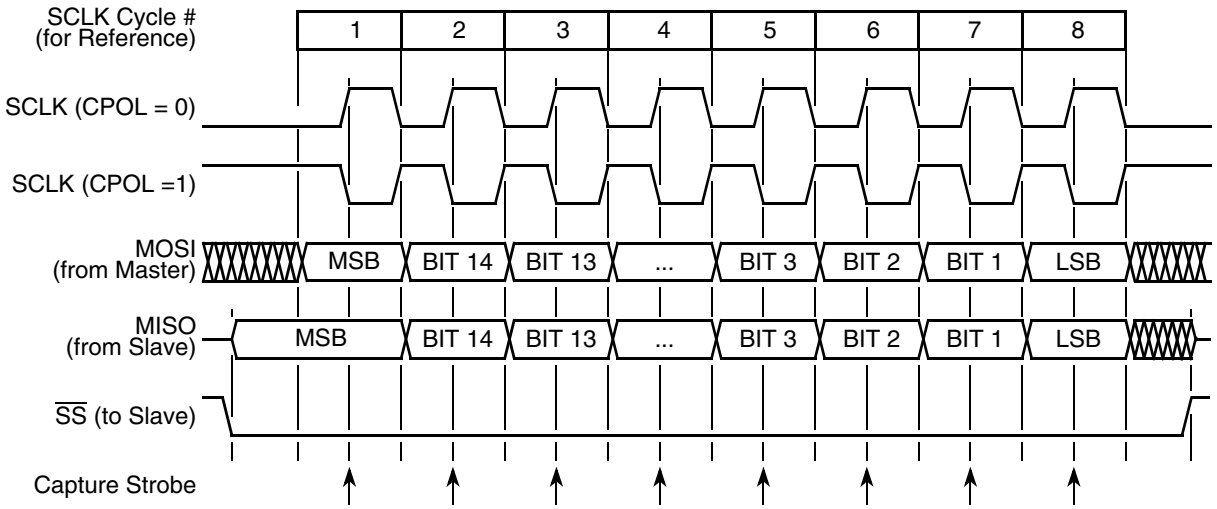


Figure 13-4. Transmission Format (CPHA = 0)

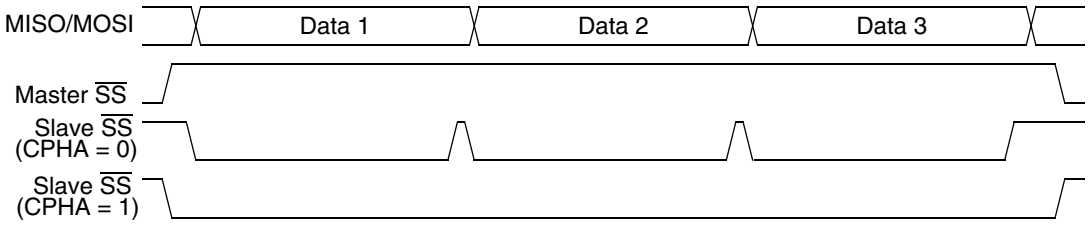


Figure 13-5. CPHA/$\overline{SS}$ Timing

When CPHA = 0 for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transmission. This causes the QSPI to leave its idle state and begin driving the MISO pin with the first bit of its data. Once the transmission begins, no new data is allowed into the shift register from the DXMIT register. Therefore, the DXMIT of the slave must be loaded with transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the DXMIT register and transferred to the shift register after the current transmission. Also for correct operation of the slave, SPE must be active before the negative edge of $\overline{SS}$ to correctly send/receive the first word. The slave QSPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so only the selected slave drives to the master.

When CPHA = 0 for a master, normal operation would begin by the master initializing the $\overline{SS}$ pin of the slave high. A transfer would then begin by the master setting the $\overline{SS}$ pin of the slave low and then writing the SPDTR register. After completion of a data transfer, the $\overline{SS}$ pin would be put back into the high state by the master device. While MODFEN = 1, the $\overline{SS}$ pin of the master must be high or a mode fault error occurs. If MODFEN = 0, the state of the $\overline{SS}$ pin is ignored.

13.6.5 Transmission Format When CPHA = 1

A QSPI transmission is shown in Figure 13-6 where CPHA is logic 1.

NOTE:

The figure should not be used as a replacement for data sheet parametric information.

Two waveforms are shown for SCLK: one for CPOL = 0 and another for CPOL = 1. The waveform diagrams 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 transmission. This causes the QSPI to leave its idle state and begin driving the MISO pin with the first bit of its data. Once transmission begins, no new data is allowed into the shift register from the DXMIT. Therefore, the DXMIT 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 DXMIT register and transferred to the shift register after the current transmission. The slave QSPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so 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. Therefore, the slave uses the first SCLK edge as a start transmission signal. 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 having only one master and only one slave driving the MISO data line.

NOTE:

Figure 13-6 assumes 16-bit data lengths and the MSB shifted out first.

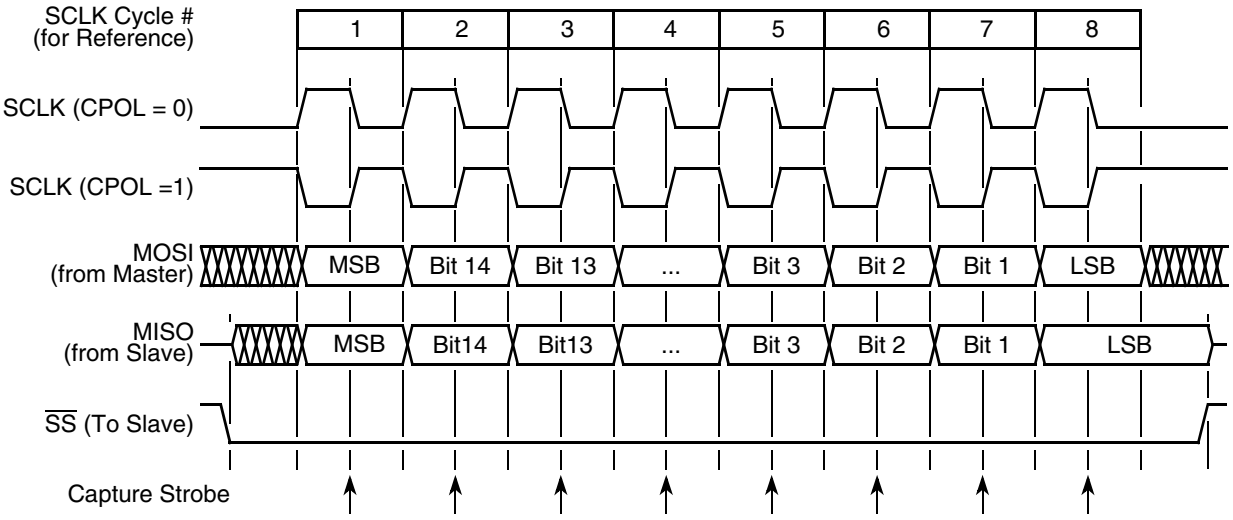


Figure 13-6. Transmission Format (CPHA = 1)

13.6.6 Transmission Initiation Latency

When the QSPI is configured as a master (SPMSTR = 1), writing to the DXMIT register begins a transmission. CPHA has no effect on the delay to the start of the transmission, 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 QSPI clock rate, selected by SPR[2:0], affects the delay from the write to DXMIT register and the start of the QSPI transmission. The internal QSPI baud clock in the master is a derivative of the internal clock. To conserve power, it is enabled only after the SPE and SPMSTR bits are set and there is a new word written to the DXMIT. If the DXMIT has no new word when the current transaction completes the internal baud clock is stopped. The initiation delay is a single QSPI bit time, illustrated in Figure 13-7. That is, the delay is four controller bus cycles for DIV4, eight controller bus cycles for DIV8, 16-controller bus cycles for DIV16, 32-controller bus cycles for DIV32, and so on.

NOTE:

Figure 13-7 assumes 16-bit data lengths and the MSB shifted out first.

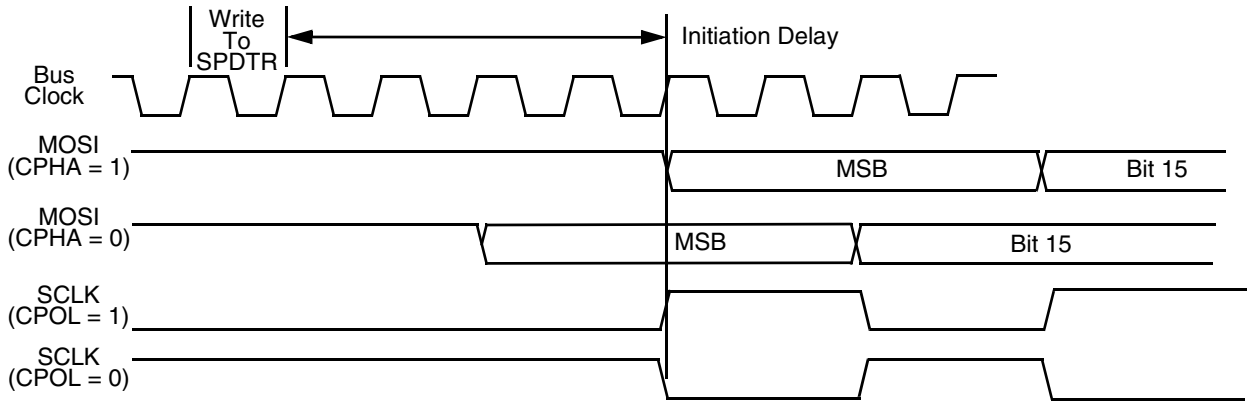


Figure 13-7. Transmission Start Delay (Master)

13.6.7 $\overline{SS}$ Hardware Generated Timing in Master Mode

If the SS_STRB bit is set in master mode the QSPI generates a word strobe pulse on $\overline{SS}$ for a slave device. Please see Figure 13-8 and Figure 13-9.

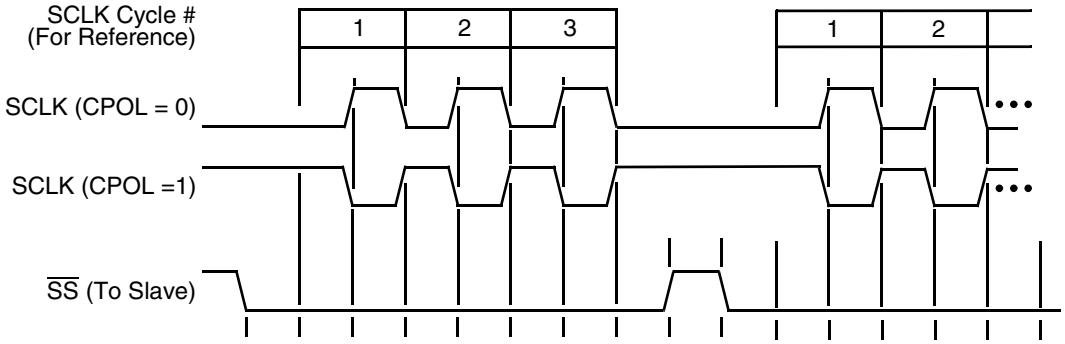


Figure 13-8. $\overline{SS}$ Strobe Timing (CPHA = 0)

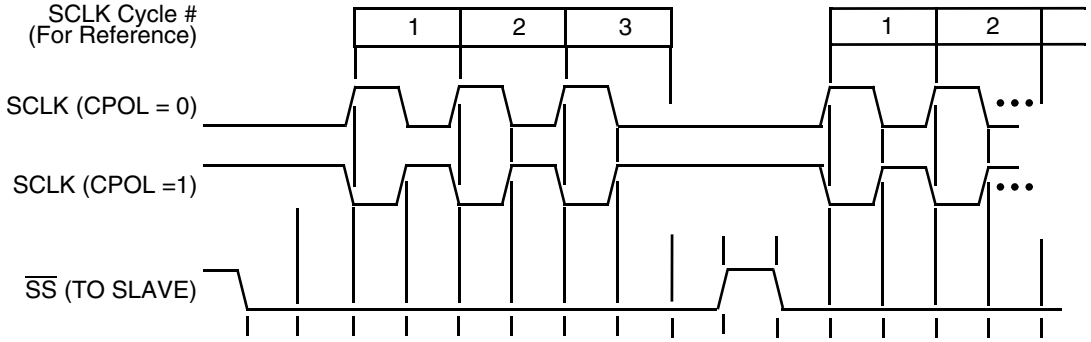


Figure 13-9. $\overline{SS}$ Strobe Timing (CPHA = 1)

If the SS_AUTO bit is set in master mode the QSPI 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. Please see Figure 13-10.

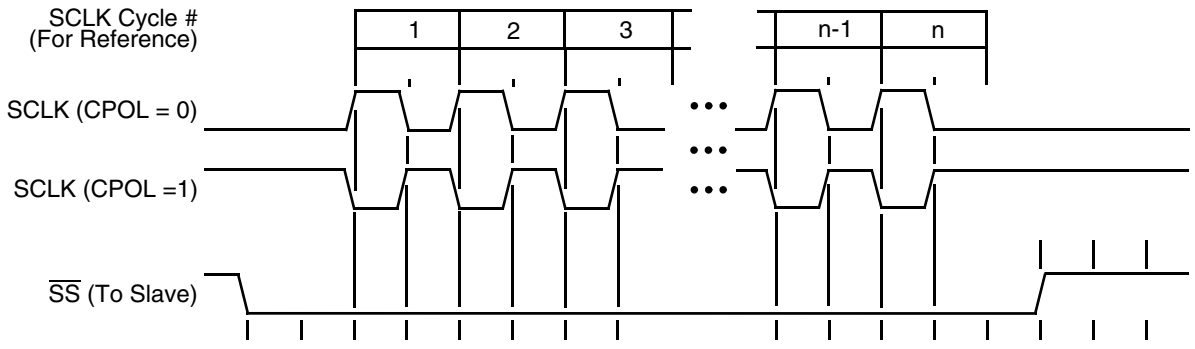


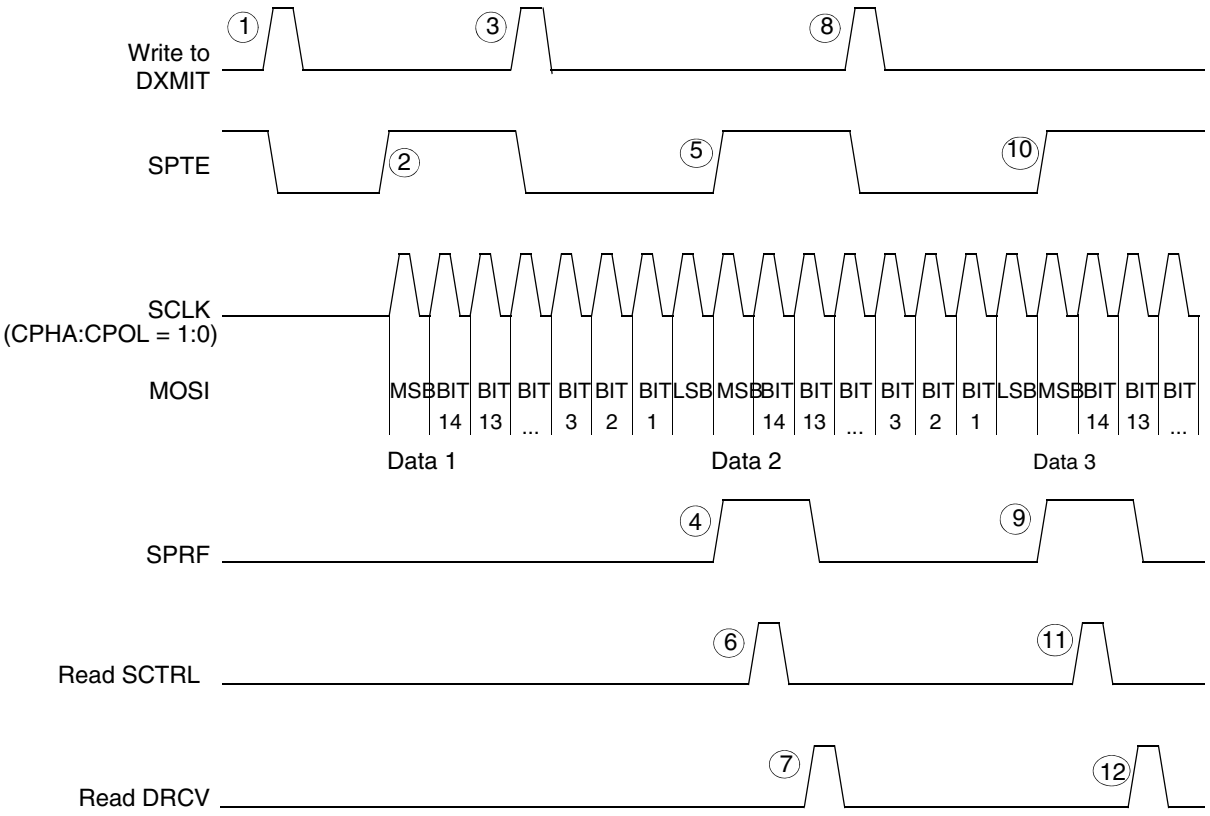
Figure 13-10. $\overline{SS}$ Auto Timing (CPHA = 1)

13.7 Transmission Data

The double-buffered DXMIT register allows data to be queued and transmitted. For a QSPI configured as a master, the queued data is transmitted immediately after the previous transmission is completed. The SPTE bit indicates when the transmit data buffer is ready to accept new data. Write to the DXMIT register only when the SPTE bit is high. Figure 13-11 illustrates the timing associated with doing back-to-back transmissions with the QSPI (SCLK has CPHA = 1; CPOL = 0).

NOTE:

Figure 13-11 assumes 16-bit data lengths and the MSB shifted out first.



1. MCU writes DATA1 to DXMIT, clearing the SPTE bit.
2. DATA1 transfers from transmit data register to shift register, setting SPTE bit.
3. MCU writes DATA2 to DXMIT, queueing DATA2 and clearing SPTE bit.
4. First incoming word transfers from the shift register to the DRCV register, setting the SPRF bit.
5. DATA2 transfers from the DXMIT register to shift register, setting the SPTE bit.
6. MCU reads SCTRL with the SPRF bit set.
7. MCU reads DRCV, clearing SPRF bit.
8. MCU writes DATA3 to DXMIT, queueing DATA3 and clearing the SPTE bit.
9. Second incoming data transfers from shift register to DRCV register setting the SPRF bit.
10. DATA3 Transfers from the DXMIT register to the shift register, setting the SPTE bit.
11. MCU reads SCTRL with SPRF bit.
12. MCU reads DRCV, clearing the SPRF bit.

Figure 13-11. SPRF/SPTE Interrupt Timing

The transmit data buffer permits back-to-back transmissions without the slave precisely timing its writes between transmissions as is necessary in a system with a single data buffer. Also, in slave mode, if no new data is written to DXMIT the last value contained in DXMIT is retransmitted if the external master starts a new transaction.

For an idle master without data loaded into its transmit buffer and no word is currently being transmitted, the SPTE bit is set again no more than two bus cycles after DXMIT is written. This allows the user to queue up at most a 32-bit value to send. For a QSPI operating in slave mode, the load of the shift register is controlled by the external master and back-to-back writes to the DXMIT register are not possible. The SPTE bit indicates when the next write can occur.

13.8 Error Conditions

The following bits signal QSPI error conditions:

- Overflow (OVRF)—Failing to read the DRCV register before the next full length data enters the shift register sets the OVRF bit. The new data does not transfer to the DRCV register, and the unread data word can still be read. OVRF is in the SCTRL register.
- Mode fault error (MODF)—The MODF bit indicates the voltage on the slave select pin ($\overline{SS}$) is inconsistent with QSPI mode. MODF is in the SCTRL register.

13.8.1 Overflow Error

The overflow flag (OVRF) becomes set if last bit of a current transmission is received and the DRCV register still has unread data from a previous transmission. If an overflow occurs, all data received after the overflow, and before the OVRF bit is cleared, does not transfer to the DRCV register. It does not set the QSPI receiver full (SPRF) bit. The unread data transferred to the DRCV register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow bit by reading the SCTRL register, and then read the DRCV register.

OVRF generates a receiver/error interrupt request if the error interrupt enable (ERRIE) bit is also set. The ERRIE bit enables both the OVRF and MODF interrupts. If OVRF is the only error interrupt of interest, the MODF interrupt can be disabled by setting MODEFEN low. Because SPRF and OVRF share a single interrupt signal, the interrupt service routine must be coded to check the OVRF bit for an error condition.

If the QSPI receiver full/error (SPRF) bit interrupt is enabled and the overflow flag (OVRF) interrupt is not set, watch for an overflow condition. Figure 13-12 explains how it is possible to miss an overflow. The first element of the same figure illustrates how it is possible to read the SCTRL and DRCV registers to clear the QSPI receiver full (SPRF) bit without problems. However, as illustrated by the second transmission example, the overflow flag (OVRF) bit can be set between the time SCTRL and DRCV registers are read.

In this case, an overflow can easily be missed. Since no more QSPI receiver full (SPRF) bit interrupts can be generated until this overflow flag (OVRF) is serviced, it is not obvious data is being lost as more transmissions are completed. To prevent this loss, either enable the overflow flag (OVRF) interrupt or take another read of the SCTRL register following the read of the DRCV register. This read ensures the overflow flag (OVRF) was not set before the receiver full (SPRF) bit was cleared and future transmissions can set the SPRF bit.

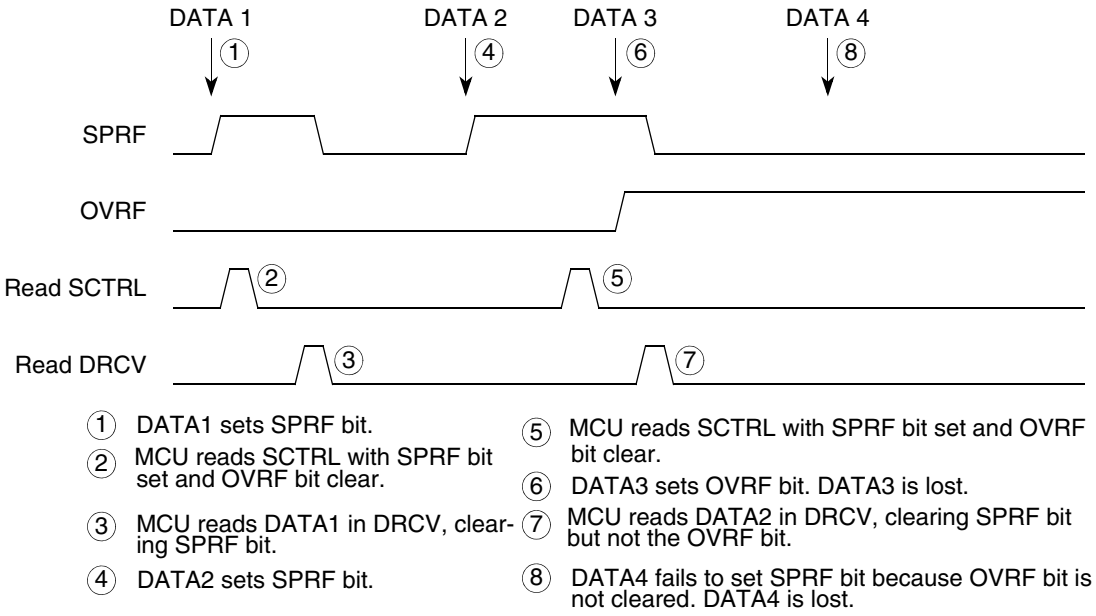


Figure 13-12. Missed Read of Overflow Condition

Figure 13-13 illustrates the described process. Generally, to avoid a second SCTRL register read, enable the overflow flag (OVRF) by setting the error interrupt enable (ERRIE) bit.

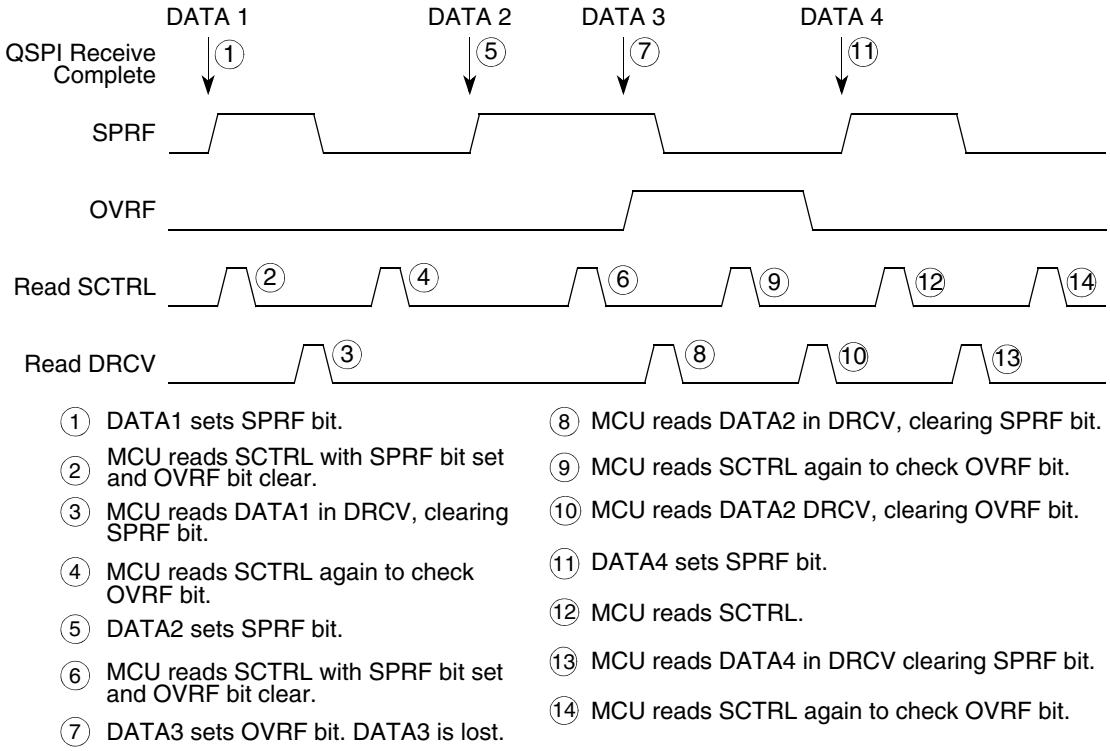


Figure 13-13. Clearing SPRF When OVRF Interrupt Is Not Enabled

13.8.2 Mode Fault Error

Setting the SPMSTR bit selects master mode, configuring the SCLK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode, configuring the SCLK and MOSI pins as inputs and the MISO pin as an output. The mode fault (MODF) bit becomes set any time the state of the $\overline{SS}$ pin is inconsistent with the mode selected by SPMSTR provided MODFEN = 1. To prevent QSPI pin contention and damage to the MCU, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave QSPI goes high during a transmission
- The $\overline{SS}$ pin of a master QSPI goes low at any time

To set the MODF bit, the mode fault error enable (MODFEN) bit must be set. Clearing the MODFEN bit does not clear the MODF bit, but it does prevent the MODF from being set again after the MODF is cleared.

MODF generates a receiver full/error controller interrupt request if the error interrupt enable (ERRIE) bit is also set. It is not possible to enable MODF independently of OVRF to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set and only OVRF interrupts are observed.

13.8.2.1 Master Mode Fault

In a master QSPI with mode fault enable (MODFEN) bit set, mode fault (MODF) bit is set if $\overline{SS}$ goes to logic 0. A mode fault in a master QSPI causes the following events to occur:

- If ERRIE = 1, the QSPI generates a QSPI receiver full/error controller interrupt request
- The SPE bit is cleared (QSPI disabled)
- The SPTE bit is set
- The QSPI state counter is cleared

In a master QSPI, the MODF bit does not clear until the $\overline{SS}$ pin is at logic 1, or the QSPI is configured as a slave.

To clear the MODF bit, write one to the MODF bit field in the SCTRL register. The mode bit does not clear if the triggering condition continues ($\overline{SS}$ = 0). The clearing mechanism must occur with no MODF condition existing or the bit is not cleared.

NOTE:

Setting the MODF bit 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 QSPI is a master and when it is a slave.

13.8.2.2 Slave Mode Fault

When configured as a slave (SPMSTR = 0), the MODF bit is set if the $\overline{SS}$ goes high during a transmission. The MODF bit generates a QSPI receiver/error controller interrupt request if the error interrupt enable (ERRIE) bit is set. The MODF bit does not clear the QSPI enable (SPE) bit or reset the QSPI in any way. Software can abort the QSPI transmission by clearing the SPE bit of the slave.

When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SCLK goes back to its idle level following the shift of the last data bit. A MODF occurs if a slave is selected ($\overline{SS}$ is at logic 0) and later unselected ($\overline{SS}$ is at logic 1) even if no SCLK is sent to that slave. This happens because $\overline{SS}$ at logic 0 indicates the start of the transmission (MISO driven out with the value of MSB).

When CPHA = 1, the transmission begins when the SCLK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SCLK returns to its idle level following the shift of the last data bit. When CPHA =1, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.

NOTE:

Logic 1 voltage on the $\overline{SS}$ pin of a slave QSPI puts the MISO pin in a high impedance state. Also, the slave QSPI ignores all incoming SCLK clocks, even if it was already in the middle of a transmission.

In a slave QSPI, if the MODF bit is not cleared by writing one to the MODF bit, the QSPI Receiver full/error interrupt continues to fire. In this case, the interrupt caused by the MODF bit can be cleared by disabling the EERIE or MODFEN bits (if set), or by disabling the QSPI. Disabling the QSPI using the QSPI enable (SPE) bit causes a partial reset of the QSPI and may cause the loss of a message currently being received or transmitted.

13.9 Register Description

A register address is the sum of a base address and an address offset. The base address is defined at the device level and the address offset is defined at the module level.

The QSPI peripheral has six registers to control and monitor QSPI operation. These registers should be accessed only with word accesses. Accesses other than word lengths result in undefined results. The QSPI bit in the SIM module's SIM_PCE register must be one before the QSPI registers can be changed.

Table 13-3. QSPI Memory Map

Device	Peripheral	Base Address
56F80xx	QSPI0	$00F220
	QSPI1	$00F230

Table 13-4 lists the QSPI registers in ascending address order, including their acronyms and address offset of each register.

Table 13-4. QSPI Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	SCTRL	Status and control register	Read/Write	Section 13.9.1
Base + $1	DSCTRL	Data size and control register	Read/Write	Section 13.9.2
Base + $2	DRCV	Data receive register	Read-Only	Section 13.9.3
Base + $3	DXMIT	Data transmit register	Write-Only	Section 13.9.4
Base + $4	FIFO	FIFO control register	Read/Write	Section 13.9.5
Base + $5	DELAY	Word delay register	Read/Write	Section 13.9.6

The six registers in the QSPI peripheral are summarized in Table 13-4. Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	SCTRL	R	SPR			DSO	ERRIE	MOD FEN	SPRIE	SPM STR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
		W																
$1	DSCTRL	R	WOM	0	0	BD2X	SS_IN	SS_ DATA	SS_ ODM	SS_ AUTO	SS_ DDR	SS_ STRB	SS_ OVER	0	DS3	DS2	DS1	DS0
		W																
$2	DRCV	R	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
		W																
$3	DXMIT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
$4	FIFO	R	0	TFCNT			0	RFCNT			0	TFWM		0	RFMW		0	FIFO_ ENA
		W																
$5	DELAY	R	0	0	0	WAIT												
		W																

R	0	Read as 0
W		Reserved

Figure 13-14. QSPI Register Map Summary

13.9.1 Status and Control (SCTRL) Register

Base + $0	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

Figure 13-15. Status and Control (SCTRL) Register

NOTE:

Using BFCLR or BFSET instructions on the SCTRL register can cause unintended side effects on the status bits. This is due to these instructions being a read/write modify instruction.

13.9.1.1 QSPI Baud Rate Select (SPR)—Bits 15–13

While in the master mode, these read/write bits select one of eight baud rates depicted in Table 13-5. SPR bit field has no effect in slave mode. Reset sets SPR bits to $3. Use the formula below to calculate the QSPI baud rate.

$$\text{Baud Rate} = \frac{\text{Module Clock}}{\text{Baud Rate Divisor}}$$

Table 13-5. QSPI Master Baud Rate Selection

SPR[2:0]	Baud Rate Divisor (BD)	
	BD2X=0	BD2X=1
000	2	4
001	4	8
010	8	16
011	16	32
100	32	64
101	64	128
110	128	256
111	256	512

NOTE:

The maximum data transmission rate for the QSPI is typically limited by the bandwidth of the I/O drivers on the chip. Typical limits are provided in Table 13-6. These apply to both master and slave modes. The SPR field needs to be set to keep the module within these ranges.

Table 13-6. Transmission Baud Rate Limitations Due to I/O as a Function of Technology

	Normal	Wired-OR
TSMC 0.25	40MHz	10MHz
TSMC 0.18	40MHz	10MHz

13.9.1.2 Data Shift Order (DSO)—Bit 12

This read/write bit determines whether the MSB or LSB is transmitted or received first. Both master and slave QSPI modules must transmit and receive the same length packets. Regardless how this bit is set, when reading from the data receive (DRCV) register, or writing to the data transmit (DXMIT) register, the LSB is always be at bit location zero and the MSB is at the correct bit position. If the data length is less than 16 bits, the data is zero padded on the upper bits.

- 0 = MSB transmitted first (MSB->LSB)
- 1 = LSB transmitted first (LSB->MSB)

13.9.1.3 Error Interrupt Enable (ERRIE)—Bit 11

This read/write bit enables the MODF, if MODFEN is also set, and OVRF bits to generate controller interrupt requests. Reset clears the ERRIE bit.

- 0 = MODF and OVRF cannot generate controller interrupt requests
- 1 = MODF and OVRF can generate controller interrupt requests

13.9.1.4 Mode Fault Enable (MODFEN)—Bit 10

When set to one, this read/write bit allows the MODF bit to be set. If the MODF bit is set, clearing the MODFEN does not clear the MODF bit.

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled QSPI configured as a master. If configured as a master and MODFEN = 1, a transmission in progress stops if $\overline{SS}$ goes low.

For an enabled QSPI configured as a slave, having MODFEN low only prevents the MODF bit from being set. It does not affect any other part of QSPI operation.

13.9.1.5 QSPI Receiver Interrupt Enable (SPRIE)—Bit 9

This read/write bit enables interrupt requests generated by the QSPI receiver full (SPRF) bit or the receive FIFO watermark register.

- 0 = SPRF interrupt requests disabled
- 1 = SPRF interrupt requests enabled

13.9.1.6 QSPI Master (SPMSTR)—Bit 8

This read/write bit selects master or slave modes of operation.

- 0 = Slave mode
- 1 = Master mode

13.9.1.7 Clock Polarity (CPOL)—Bit 7

This read/write bit determines the logic state of the SCLK pin between transmissions. To transmit data between QSPI modules, the QSPI modules must have identical CPOL values. Please see Figure 13-4 and Figure 13-6.

- 0 = SCLK idle state between transmission
- 1 = SCLK idle state between transmission

13.9.1.8 Clock Phase (CPHA)—Bit 6

This read/write bit controls the timing relationship between the serial clock and QSPI data. Please see Figure 13-4 and Figure 13-6. To transmit data between QSPI modules, the QSPI modules must have identical CPHA values. When CPHA = 0, the $\overline{SS}$ pin of the Slave QSPI module must be set to logic 1 between data transmissions, illustrated in Figure 13-5.

13.9.1.9 QSPI Enable (SPE)—Bit 5

This read/write bit enables the QSPI module. Clearing SPE causes a partial reset of the QSPI. When setting/clearing this bit, no other bits in the SCTRL register should be changed. Change any other bits in a separate write statement. In master mode the SPE bit can be cleared by a mode fault condition. Failure to follow this statement may result in spurious clocks to the slave.

- 0 = QSPI module disabled
- 1 = QSPI module enabled

13.9.1.10 QSPI Transmit Interrupt Enable (SPTIE)—Bit 4

This read/write bit enables interrupt requests generated by the transmitter empty (SPTE) bit, or the transmit FIFO watermark register.

- 0 = SPTE interrupt requests disabled

- 1 = SPTE interrupt requests enabled

13.9.1.11 QSPI Receiver Full (SPRF)—Bit 3

This is a read-only status bit. The way SPRF is detected depends on the setting of FIFO enable (FIFO_EN). SPRF generates an interrupt request if the receiver interrupt enable (SPRIE) bit in the SCTRL register is set. This bit is automatically cleared by reading the DRCV register.

- 0 = Data receive (DRCV) register has no new data, or the receive data FIFO level (RFCNT) is less than the receive data FIFO watermark (RFWM). If using the FIFO, read the RFCNT bit field to determine the number of valid words available.
- 1 = Data receive (DRCV) register has new data, or the receive data FIFO (RFCNT) level is greater than, or equal to the receive data FIFO watermark (RFWM).

13.9.1.12 Overflow (OVRF)—Bit 2

This is a read-only status bit. It is set if software does not read the data in the DRCV register before the next full data enters the shift register. In an overflow condition, the data already in the DRCV register is unaffected, and the data shifted in last is lost. Clear the overflow (OVRF) bit by reading the SCTRL register and then reading the DRCV register. OVRF generates a receiver/error interrupt if the error interrupt enable (EERIE) bit is set. See Section 13.8.1 for more details.

- 0 = No overflow
- 1 = Overflow

13.9.1.13 Mode Fault (MODF)—Bit 1

This read-only status bit is capable of being cleared. It is set in a slave QSPI if the $\overline{SS}$ pin goes high during a transmission with the MODFEN bit set. In a master QSPI, the MODF bit is set if the $\overline{SS}$ pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing one to the MODF bit when it is set. MODF generates a receive/error interrupt if the EERIE bit is set. Please see Section 13.8.2 for more details.

- 0 = $\overline{SS}$ pin at appropriate logic level
- 1 = $\overline{SS}$ pin at inappropriate logic level

13.9.1.14 QSPI Transmitter Empty (SPTE)—Bit 0

This is a read-only status bit. The way SPTE is detected depends on the setting of FIFO enable (FIFO_EN). SPTE generates an interrupt request if the transmit interrupt enable (SPTIE) bit in the SCTRL register is set. SPTE is automatically cleared by writing to the DXMIT register.

- 0 = DXMIT register was not read by the shift register or the transmit FIFO level (TFCNT) is greater than the TxFIFO watermark (TFWM). If using the FIFO, read the TFCNT bit field to determine how many words can be written safely.
- 1 = DXMIT register was read by the shift register or the transmit FIFO level (TFCNT) is less than, or equal to the TxFIFO watermark (TFWM).

NOTE:

Do not write to the DXMIT register unless the SPTE bit is high or TFCNT is less than $4 otherwise data may be lost.

13.9.2 Data Size and Control (DSCTRL) Register

To utilize the $\overline{SS}$ control functions in master mode the appropriate bit must be set in GPIO_x_PER register to enable peripheral control of the $\overline{SS}$ pin.

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WOM	0	0	BD2X	SS_IN	SS_ DATA	SS_ ODM	SS_ AUTO	SS_ DDR	SS_ STRB	SS_ OVER	0	DS3	DS2	DS1	DS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 13-16. Data Size and Control (DSCTRL) Register

13.9.2.1 Wired-OR Mode (WOM)—Bit 15

This control bit selects the nature of the QSPI pins. When the WOM bit is set, the QSPI pins are configured as open-drain drivers. When the WOM bit is cleared, the QSPI pins are configured as push-pull drivers.

- 0 = Wired OR mode disabled
- 1 = Wired OR mode enabled

13.9.2.2 Reserved—Bits 13 and 14

13.9.2.3 Baud Divisor Times 2 (BD2X)—Bit 12

When set, the baud rate divisor bit is multiplied by two. Please see Table 13-5.

13.9.2.4 Slave Select Input (SS_IN)—Bit 11

This read-only bit shows the current state of the $\overline{SS}$ pin in all modes.

13.9.2.5 Slave Select Data (SS_DATA)—Bit 10

This read/write bit is the value to drive on the $\overline{SS}$ pin. This bit is disabled when SS_AUTO = 1 or SS_STRB = 1.

- 0 = $\overline{SS}$ pin is driven low if SS_DDR=1
- 1 = $\overline{SS}$ pin is driven high if SS_DDR=1

13.9.2.6 Slave Select Open Drain Mode (SS_ODM)—Bit 9

This read/write bit enables open drain mode on the $\overline{SS}$ pin in master mode.

- 0 = $\overline{SS}$ is configured for high and low drive. This mode is generally used in single master systems.
- 1 = $\overline{SS}$ is configured as an open drain pin (only drives low output level). This mode is useful for multiple master systems.

13.9.2.7 Slave Select Auto (SS_AUTO)—Bit 8

This read/write bit enables hardware control of the $\overline{SS}$ pin in master mode. The legacy design requires software to control the $\overline{SS}$ output pin.

The initial falling edge of $\overline{SS}$ is generated and $\overline{SS}$ is held low until the TX buffer, or the FIFO is empty. This bit may be used alone or in combination with the SS_STRB in order to generate the required $\overline{SS}$ signal.

- 0 = $\overline{SS}$ output signal is software generated by directly manipulating the various bits in this register (SS_DATA) or the GPIO registers (compatible with legacy QSPI software).
- 1 = $\overline{SS}$ output signal is hardware generated to create the initial falling edge and final rising edge. The idle state of the $\overline{SS}$ is high.

13.9.2.8 Slave Select Data Direction (SS_DDR)—Bit 7

This read/write bit controls input/output mode on the $\overline{SS}$ pin in master mode.

- 0 = $\overline{SS}$ is a configured as an input pin. Use this setting in slave or master modes with MODFEN = 1.
- 1 = $\overline{SS}$ is configured as an output pin. Use this setting in master mode with MODFEN = 0.

13.9.2.9 Slave Select Strobe Mode (SS_STRB)—Bit 6

This read/write bit enables hardware pulse of the $\overline{SS}$ pin in master mode between words. This bit may be used alone or in combination with the SS_AUTO to generate the required $\overline{SS}$ signal. Pulses are generated between words irrespective of the setting of CPHA.

- 0 = No $\overline{SS}$ pulse between words
- 1 = $\overline{SS}$ output signal is pulsed high between words. This adds 1.5 baud clocks to the total word period. The idle state of the $\overline{SS}$ is low unless SSB_AUTO is high and then the idle state is high.

13.9.2.10 Slave Select Override (SS_ORR)—Bit 5

This read/write bit overrides the internal $\overline{SS}$ signal input from the I/O pad and replaces it with a level equal to the setting of the SPMSTR bit. This allows the QSPI to function in slave mode, CPHA = 1, without committing 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, therefore this bit should not be used in a multi-master system.

- 0 = $\overline{SS}$ internal module input is selected to be connected to a GPIO pin
- 1 = $\overline{SS}$ internal module input is selected to be equal to SPMSTR

13.9.2.11 Reserved—Bit 4

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

13.9.2.12 Data Size (DS)—Bits 3–0

This read/write register determines the data length for each transmission. The master and slave must transfer the same size data on each transmission. A new value only takes effect at the time the QSPI is enabled (SPE bit in SCTRL set from zero to one). In order to have a new value take effect, disable then re-enable the QSPI with the new value in the register. Please see Table 13-7 for detailed transmission data size.

Table 13-7. Data Size

DS[3:0]	Size of Transmission
$0	Not Allowed
$1	2 Bits
$2	3 Bits

Table 13-7. Data Size (Continued)

$3	4 Bits
$4	5 Bits
$5	6 Bits
$6	7 Bits
$7	8 Bits
$8	9 Bits
$9	10 Bits
$A	11 Bits
$B	12 Bits
$C	13 Bits
$D	14 Bits
$E	15 Bits
$F	16 Bits

13.9.3 Data Receive (DRCV) Register

This read-only register provides the last data word received after a complete transmission. When SPRF bit in the SCTRL register is set, new data can be read from this register.

Base + $2	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

Figure 13-17. Data Receive (DRCV) Register

13.9.4 Data Transmit (DXMIT) Register

This write-only register holds data to be transmitted. When the transmitter empty (SPTE) bit is set, new data should be written to this register. If new data is not written while in the master mode, a new transaction is not initiated until this register is written. In slave mode, the old data re-transmits if DXMIT is not rewritten before the next transmission begins. All data should be written with the LSB at bit zero. This register can only be written when the QSPI is enabled, SPE = 1.

Base + $3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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

Figure 13-18. Data Transmit (DXMIT) Register

13.9.5 FIFO Control (FIFO) Register

This register is used for FIFO control status.

Base + $4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	TFCNT			0	RFCNT			0	TRWM		0	RFWM		0	FIFO_ ENA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Figure 13-19. FIFO Control (FIFO) Register

13.9.5.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

13.9.5.2 Transmit FIFO Level (TFCNT)—Bit 14-12

This read-only bit field indicates the number of words are used in the Tx FIFO. Writes to the DXMIT causes TFCNT to increase, as words are pulled for transmission TFCNT is decreased. Attempts to write new data to DXMIT register are ignored when TFCNT indicates the FIFO is full. If FIFO_ENA is set to zero, the TFCNT bit always indicates FIFO empty. If master mode is enabled, transmission continues until the FIFO is empty, even if SPE is set to zero.

Table 13-8. QSPI TX FIFO Level Decode

TFCNT[2-0]	TX FIFO Status
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

13.9.5.3 Reserved—Bit 11

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

13.9.5.4 Receive Data FIFO Level (RFCNT)—Bits 10–8

This read-only bit field indicates the number of words used in the RX FIFO. As words are received RFCNT increases, as words are read from the DRCV register the value of RFCNT are decreased. There is one word time to read DRCV register between when the SPRF status bit is set (interrupt asserted) and when a overflow condition is flagged. If FIFO_ENA is set to zero the RFCNT always indicates FIFO is empty.

Table 13-9. QSPI Rx FIFO Level Decode

RFCNT[2-0]	RX FIFO Status
000	RX FIFO empty
001	One word used in RXFIFO
010	Two words used in RX FIFO
011	Three words used in RX FIFO
100	RX FIFO full (If enabled receiver full interrupt asserted)

13.9.5.5 Reserved—Bit 7

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

13.9.5.6 Transmit Data FIFO Watermark (TFWM)—Bits 6–5

These read/write bits determine the number of words required to 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 under running 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 QSPI 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 DXMIT register, or the value of TFWM reduced.

Table 13-10. QSPI Tx FIFO Watermark Decode

TFWM[1-0]	Transmit Interrupt Active When:
00	TX FIFO is empty
01	TX FIFO has one or less words available
10	TX FIFO has two or less words available
11	TX FIFO has three or less words available

13.9.5.7 Reserved—Bit 4

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

13.9.5.8 Receive Data FIFO Watermark (RFWM)—Bits 3–2

These read/write bits determine the number of words required to be used in the RXFIFO 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, only one QSPI 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 bit, words must be read from DRCV register, or the value of RFWM must be increased.

Table 13-11. QSPI Rx FIFO Watermark Decode

RFWM[1-0]	Receive Interrupt Active When:
00	RX FIFO has at least one word used
01	RX FIFO has at least two words used
10	RX FIFO has at least three words used
11	RX FIFO if full

13.9.5.9 Reserved—Bit 1

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

13.9.5.10 FIFO Enable (FIFO_ENA)—Bit 0

This read/write bit enables TX and RX FIFO's mode.

- 0 = FIFOs are disabled and reset. Operation is compatible with legacy SPI.
- 1 = FIFOs are enabled and retain their status even if SPE is set to zero

13.9.6 Word Delay (DELAY) Register

This register controls the delay between words.

Base + $5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	WAIT												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-20. Word Delay (DELAY) Register

13.9.6.1 Reserved—Bits 15–13

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

13.9.6.2 Wait Delay (WAIT)—Bits 12–0

This bit field controls the time between data transactions in master mode. The value is the number of peripheral bus clocks to delay between words.

13.10 Interrupts

Four QSPI status bits can be enabled to generate controller interrupt requests.

Table 13-12. QSPI Interrupts Sources

Bit	Interrupt Enabled By	Description
SPTE (Transmitter Empty)	QSPI transmitter interrupt enable (SPTIE = 1, SPE = 1)	The QSPI transmitter interrupt enable (SPTIE) bit enables the SPTE bit, or TFWM to generate transmitter interrupt requests provided the QSPI is enabled (SPE = 1). The SPTE bit becomes set every time data transfers from the DXMIT to the shift register and there is no more new data available in the TX queue. The clearing mechanism for the SPTE bit is to write to the DXMIT register.
SPRF (Receiver Full)	QSPI receiver interrupt enable (SPRIE = 1)	The QSPI receiver interrupt enable (SPRIE) bit enables the SPRF bit, or RFWM to generate receiver interrupt requests. The SPRF is set every time data transfers from the shift register to the DRCV register and there is no more room available in the RX queue to receive new data. The clearing mechanism for the SPRF bit is to read the DRCV register.
OVRF (Overflow)	QSPI error interrupt enable (ERRIE = 1)	The error interrupt enable (ERRIE) bit enables both the MODF and OVRF bits to generate a receiver/error interrupt request.
MODF (Mode Fault)	QSPI error interrupt enable (ERRIE = 1) mode fault enable (MODFEN = 1)	The mode fault enable (MODFEN) bit enables the mode fault (MODF) bit to generate the receive/error interrupt request. The MODFEN bit can prevent the MODF bit from being set so only the OVRF bit is enabled by the ERRIE bit to generate receiver/error controller interrupt requests.

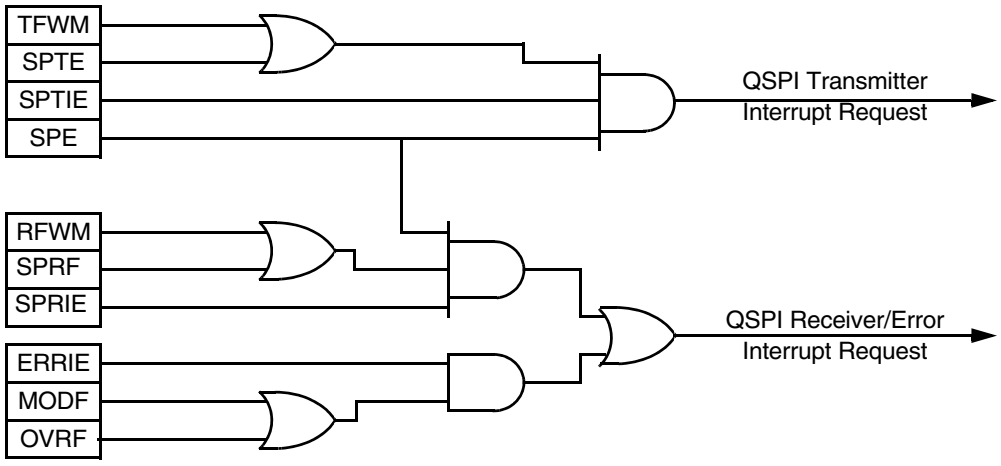


Figure 13-21. QSPI Interrupt Request Generation

13.11 QSPI Reset

Any system reset completely resets the QSPI. Partial resets occur whenever the QSPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

1. The SPTE bit 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 QSPI state counter is cleared, making it ready for a new complete transaction.
5. All the QSPI port logic is disabled.

Items four and five occur after item two when in slave mode, or after item three when in master mode. The following items are reset only by a system reset:

- The SPDTR and SPDRR registers
- All control bits in the SPSCR register (MODFEN, ERRIE, and SPR2:0)
- The status bits SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, it is possible to 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 bits, it is possible to service these interrupts after the QSPI is disabled. It is possible to disable the QSPI by writing zero to the SPE bit. The QSPI is also be disabled by a mode fault occurring in a QSPI configured as a master.

Table 13-13. Document Revision History for Chapter 13

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 14
Quad-Timer (TMR)

14.1 Introduction

The quad timer (TMR) module contains four identical counter/timer groups. Each 16-bit counter/timer group contains a:

- Prescaler
- Counter
- Load register
- Hold register
- Capture register
- Two compare registers
- One status and control register
- One control register

All of the registers are read/write capable.

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 is reached. The hold register captures the counter's value when other registers are being read. This feature supports the reading of cascaded counters. The capture register enables an external signal to take a snapshot of the counter's current value.

COMP1 and COMP2 registers provide 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 COMP1 or COMP2 registers from CMPLD1 and CMPLD2 when enabled.

The prescaler provides different time bases useful for clocking the counter/timer. The counter provides the ability to count internal or external events. Input pins can be shared within a timer module (set of four timer/counters).

14.2 Features

- Four 16-bit counters/timers
- Capable of counting up and down
- Counters will cascade
- Count modulo can be programmed
- Maximum count rate equals system clock ÷2 for external clocks
- Maximum count rate equals system clock, or 3× system clock for internal clocks[1]
- Will count once or repeatedly
- Counters can be preloaded
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Programmable operation during debug mode
- Programmable input filter
- Counting start can be synchronized across counters

14.3 Block Diagram

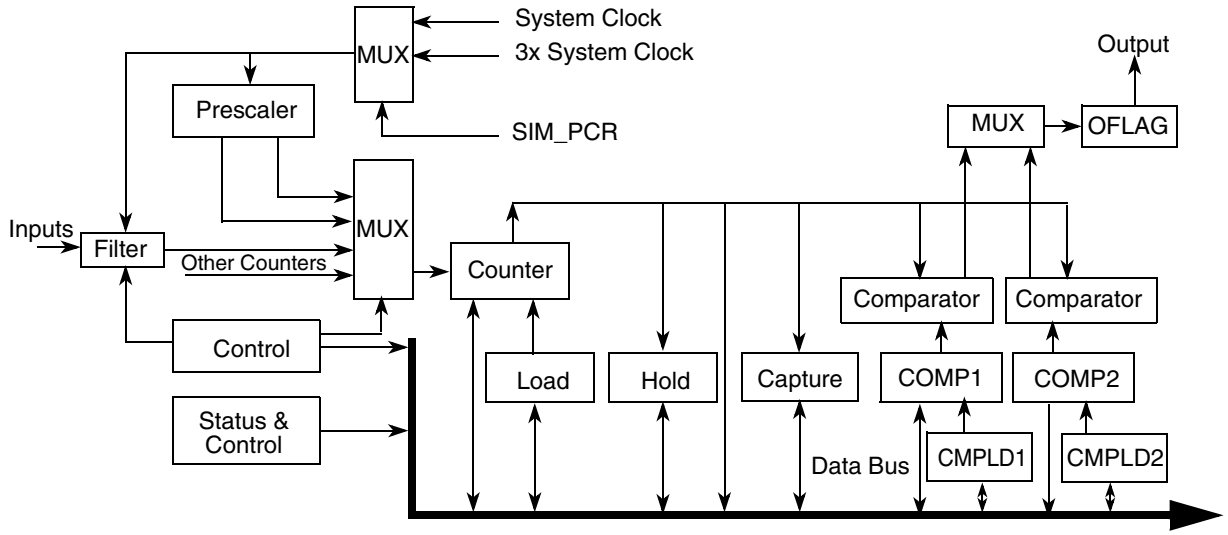


Figure 14-1. TMR Module Block Diagram

1. Timer clock selection determined by TCR bit in SIM_PCR register.

14.4 Functional Description

The counter/timer has two basic modes of operation:

1. Count internal or external events
2. Count an internal clock source while an external input signal is asserted, or an external event has occurred, thereby timing the width of the external input signal, or the time between external events.

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 also count up and down using dual inputs in a count with direction format. The counter's terminal count value (modulo) is programmable. The value 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 before immediately re-initializing, or it can count through the compare value until the count rolls over to zero.

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

- Generate count commands
- Generate timer commands
- Trigger current counter value to be captured
- Generate interrupt requests

The polarity of the external inputs can be selected. 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 shared with an external input signal.

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.

14.4.1 Compare Registers Usage

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability.

- COMP1 register = used when the counter is counting up
- COMP2 register = used when the counter is counting down

Alternating compare mode (OM = 100) is the only exception. (See below.)

The COMP1 register should be set to the desired maximum count value, or 0xFFFF, indicating the maximum unsigned value prior to roll-over. The COMP2 register should be set to the minimum count value, or 0x0000, indicating the minimum unsigned value prior to roll under.

If output mode is set to 100, the OFLAG toggles while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the ON time. The COMP1 register defines the OFF time. The variable frequency PWM mode is defined for positive counting only.

NOTE:

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it counts to 0xFFFF or 0x0000, roll-over, then begin counting toward the new value.

The check is: Count = COMP*n*, *not* Count > COMP1 or Count < COMP2.

The use of the CMPLD1 and CMPLD2 registers to preload compare values helps to minimize this problem.

14.4.2 Comparator Load Registers

The CMPLD1, CMPLD2, and CSCTRL offers 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, it is recommended to use the method described in this section.

The purpose of the comparator load feature is to allow quicker updating of the compare registers. In prior families, 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 is the possibility the counter may have already counted past the new compare value by the time the compare register is 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, compare registers are updated in hardware in the same manner the counter register is re-initialized to the value stored in the load register. The comparator load feature allows calculation of new compare values to be stored in 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 a software requirement to accomplish this.

The compare load feature is intended to be used in a variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG while COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the sum of COMP1 and COMP2 values and the frequency of the primary clock source. Please see Figure 14-2.

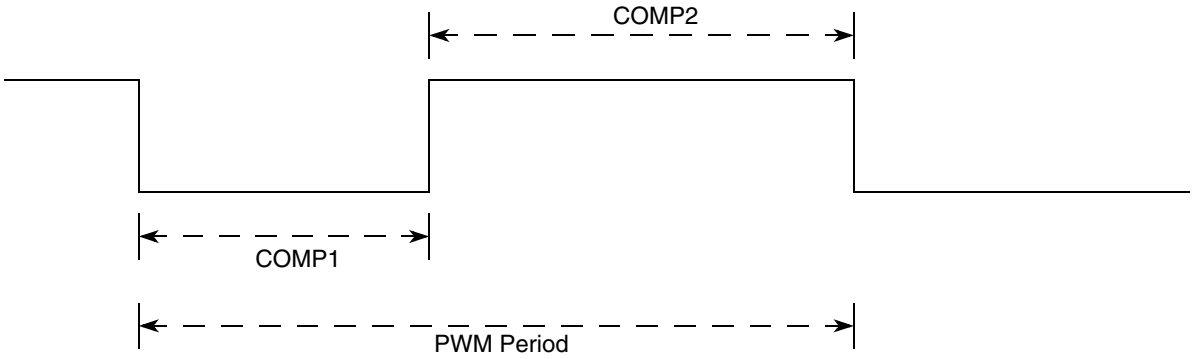


Figure 14-2. Variable PWM Waveform

If there is a desire to update the duty cycle or period of the above waveform, it is necessary to update the COMP1 and COMP2 values using the compare load feature.

14.4.3 Capture Register Usage

The capture (CAPT) register stores a copy of the counter's value when an input edge (positive, negative, or both) is detected on the secondary count source. Once a capture event occurs, no further updating of the capture register occurs until the input edge flag (IEF) is cleared by writing zero to it.

14.5 Operating Modes

The various counting modes are detailed in the following subsections.

Selected external count signals are sampled at the TMR's base clock rate (peripheral clock), 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 clock rate. These signals are all at the same clock rate.

If a counter is programmed to count to a specific value and then stop, count mode in the timer control (CTRL) register is cleared when the count terminates.

14.5.1 Stop Mode

When the count mode field is set to 000 in the control register (see Section 14.6.7.1), the counter is inert. No counting occurs.

14.5.2 Count Mode

When the count mode field is set to 001 (see Section 14.6.7.1), the counter counts 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 the input polarity select (IPS) bit (see Section 14.6.8.7), the negative edge of the selected external input signal is counted.

Example:

- Set the control (CTRL) register PCS bits to desired input
- Clear the status and control (SCTRL) register
- Set the counter (CNTR) register to 0x0000
- Set the load (LOAD) register to 0x0000
- Set the control (CTRL) register CM bits to 001

In this example, the timer counts events from primary source and measure the counter register for every successful event detection.

Example:

- Set the control (CTRL) register PCS bits to 1000
- Clear the status and control (SCTRL) register
- Set the load (LOAD) register to 0x0000
- Set the counter (CNTR) register to 0x0000
- Set the compare 1 (COMP1) register to 25000
- Set the comparator load 1 (CMPLD1) register to 25000
- Set the comparator status and control (CSCTRL) register TCF1EN bit to 1
- Set the comparator status and control (CSCTRL) register CL1 bit to 1
- Set the control (CTRL) register CM bits to 001

14.5.3 Edge Count Mode

When the count mode field is set to 010 (see Section 14.6.7.1), the counter counts 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:

- Set the control (CTRL) register PCS bits to desired input
- Clear the status and control (SCTRL) register
- Set the counter (CNTR) register to 0x0000
- Set the load (LOAD) register to 0x0000
- Set the control (CTRL) register CM bits to 010

14.5.4 Gated Count Mode

When the count mode field is set to 011 (see Section 14.6.7.1), the counter counts 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 the Input polarity select (IPS) bit, the counter counts while the selected secondary input is low.

Example:

- Set the control (CTRL) register PCS bits to the desired input
- Set the control (CTRL) register SCS bits to the desired input
- Clear the status and control (SCTRL) register
- Set the counter (CNTR) register to 0x0000
- Set the load (LOAD) register to 0x0000
- Set the control (CTRL) register CM bits to 011

14.5.5 Quadrature Count Mode

When the count mode field is set to 100 (see Section 14.6.7.1), the counter decodes 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, 90° out-of-phase. The decoding of quadrature signal provides both count and direction information. A timing diagram illustrating the basic operation of a quadrature incremental position encoder is provided in Figure 14-3.

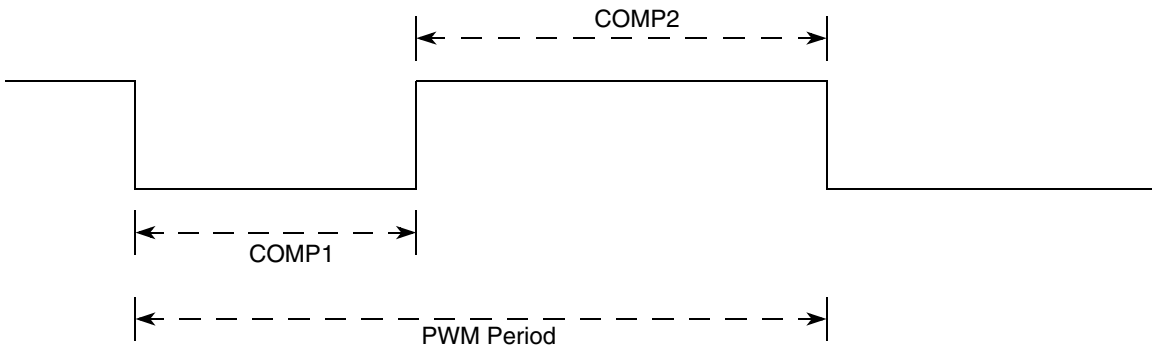


Figure 14-3. Quadrature Incremental Position Encoder

Example:

- Set the control (CTRL) register PCS bits to 0000
- Set the control (CTRL) register SCS bits to 01
- Clear the status and control (SCTRL) register
- Set the counter (CNTR) register to 0x0000
- Set the load (LOAD) register to 0x0000
- Set the compare 1 (COMP1) register to 0xFFFF
- Set the compare 2 (COMP2) register to 0x0000
- Clear the comparator status and control (CSCTRL) register
- Set the control (CTRL) register CM bits to 100

14.5.6 Signed Count Mode

When the count mode field is set to 101 (see Section 14.6.7.1), the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

Example:

- Set the control (CTRL) register PCS bits to the desired input
- Set the control (CTRL) register SCS bits to the desired input
- Clear the status and control (SCTRL) register
- Set the counter (CNTR) register to 0x0000
- Set the load (LOAD) register to 0x0000
- Set the control (CTRL) register CM bits to 101

14.5.7 Triggered Count Mode

When the count mode field is set to 110 (see Section 14.6.7.1), the counter begins counting the primary clock source after a positive transition (negative edge if IPS = 1) of the secondary input occurs. The counting continues 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 stops. Subsequent secondary input transitions continues to cause the counting to restart and stop until a compare event occurs.

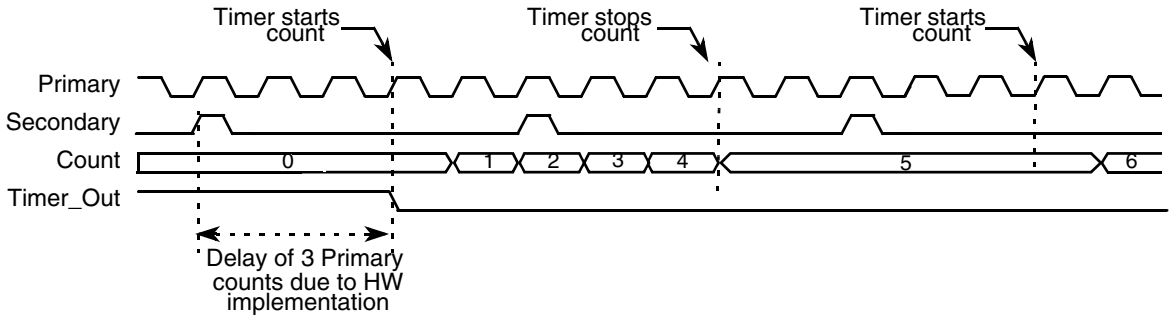


Figure 14-4. Triggered Count Mode (Length = 0)

Example:

- Set the control (CTRL) register PCS bits to the desired input
- Set the control (CTRL) register SCS bits to the desired input
- Clear the status and control (SCTRL) register
- Set the counter (CNTR) register to 0x0000
- Set the load (LOAD) register to 0x0000
- Set the compare 1 (COMP1) register to 0x0012
- Clear the comparator status and control (CSCTRL) register
- Set the control (CTRL) register CM bits to 110

14.5.8 One-Shot Mode

This is a sub-mode of triggered count mode when count mode field is set to 110 (see Section 14.6.7.1) while:

- LENGTH is set (reinitialize on compare)
- The OFLAG output mode is set to 101 (set OFLAG on compare, clear OFLAG on secondary input)

In the above state, the counter works in a one-shot mode. An external event causes the counter to count. When terminal count is reached, the OFLAG output is asserted. This delayed output assertion can be used to provide timing delays. When used with timer C2 or C3, this one-shot mode may be used to delay the ADC acquisition of new samples until a specified period of time has passed since a pulse width modulated (PWM) module reload.

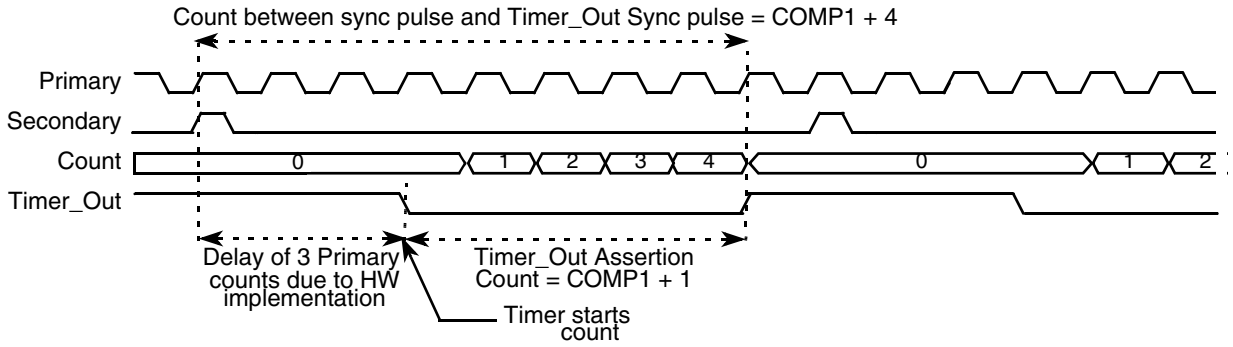


Figure 14-5. One-Shot Mode (LOAD = 0, COMP1 = 4)

Example:

- Set the control (CTRL) register PCS bits to the desired input
- Set the control (CTRL) register SCS bits to the desired input
- Set the control (CTRL) register LENGTH bit to 1
- Set the control (CTRL) register OM bits to 101
- Clear the status and control (SCTRL) register
- Set the counter (CNTR) register to 0x0000
- Set the load (LOAD) register to 0x0000
- Set the compare 1 (COMP1) register to the desired delay
- Clear the comparator status and control (CSCTRL) register
- Set the control (CTRL) register CM bits to 110

14.5.9 Cascaded Count Mode

When the count mode field is set to 111 (see Section 14.6.7.1), the counter's input is connected to the output of another selected counter. The counter counts up and down as compare events occur in the selected source counter. This cascaded, or daisy-chained mode, enables multiple counters to be cascaded in order to yield longer counter lengths. When operating in cascaded count 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 are incremented. If the selected source counter is counting down and it experiences a compare event, the counter is decremented. Up to four counters may be cascaded to create a 64-bit wide synchronous counter. Whenever any counter is read within a timer module, all of the timers' 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. Cascaded count 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 transitions a clock later than a purely synchronous design.

Example:

- Set Timer2 control (CTRL) register PCS bits to the desired input
- Set Timer2 control (CTRL) register LENGTH bits to 01
- Set Timer3 control (CTRL) register PCS bits to 110
- Set Timer3 control (CTRL) register CM bits to 111
- Set Timer3 control (CTRL) register LENGTH bits to 01
- Clear status and control (SCTRL) register for both timers
- Clear the counter (CNTR) and load (LOAD) registers from both timers
- Set Timer3 comparator 1 (COMP1) register to 30000
- Set Timer3 comparator load 1 (CMPLD1) register to 30000
- Set Timer2 comparator 1 (COMP1) register to 32000
- Set Timer2 comparator load 1 (CMPLD1) register to 32000
- Set Timer3 comparator status and control (SCTRL) register to 0x0041
- Set Timer2 comparator status and control (SCTRL) register to 0x0001
- Set Timer2 control (CTRL) register CM bits to 001

14.5.10 Pulse Output Mode

The counter outputs a stream of pulses with the same frequency of the selected clock source (cannot be IPBus clock divided by 1) if the counter is setup for:

- Count mode (CM = 001) (count rising edges of primary source)
- The OFLAG output mode is set to 111 (gated clock output)
- The count once bit is set (count until compare and then stop)

The number of output pulses is equal to the compare value minus the initial value. This mode is useful for driving step motor systems.

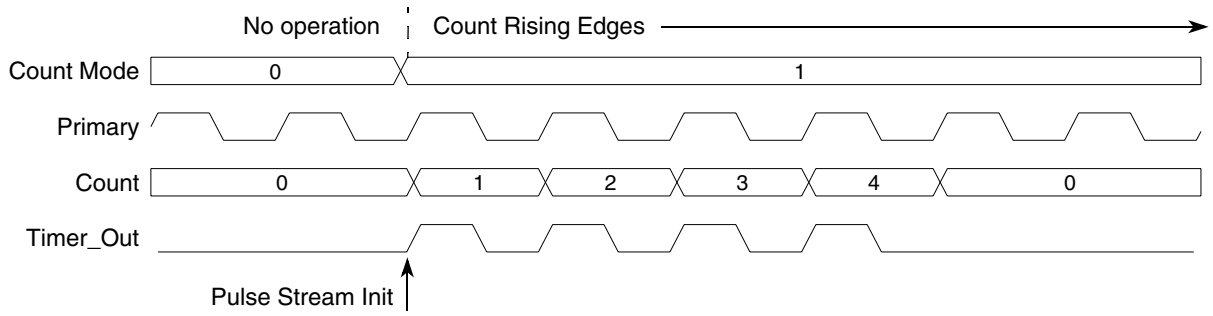


Figure 14-6. Pulse Output Mode (LOAD = 0, COMP1 = 4)

14.5.11 Fixed Frequency PWM Mode

The counter output yields a PWM signal with a frequency equal to the count clock frequency, divided by 65,536. It has a pulse width duty cycle equal to the compare value divided by 65,536, if the counter is setup for:

- Count mode (CM = 01) (count rising edges of primary source)
- Count through roll-over (LENGTH = 0)
- Continuous count (ONCE = 0)
- OFLAG output mode is 110 (set on compare, cleared on counter roll-over)
- Output polarity inverted (OPS = 1)

This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

14.5.12 Variable Frequency PWM Mode

If the counter is setup for:

- Continuous count (ONCE = 0)
- Count mode (CM = 001), count until compare (LENGTH = 1)
- OFLAG output mode is 100 (toggle OFLAG and alternate compare registers)

The counter output then yields a 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 because they permit time to calculate values for the next PWM cycle while the PWM current cycle is underway.

To setup 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, the CTRL register is suggested to be updated last because the counter starts counting if the count mode is changed to any other value other than three bits with a value of 000, assuming the primary count source is already active.

The following sections detail the timer settings required to implement variable frequency PWM operation.

14.5.12.1 Timer Control Register (CTRL)

For complete register details, please refer to Section 14.6.7.

- CM = 001 (count rising edges of primary source)
- PCS = 1000 (IPBus clock for best granularity for waveform timing)
- SCS = Any (ignored in this mode)
- ONCE = 0 (want to count repeatedly)
- LENGTH = 1 (want to count until compare value is reached and re-initialize counter register)
- DIR = Any (make a choice. The compare register values need to be chosen carefully to account for things like roll-under, etc.)
- CoINIT = 0 (can be set if this function is required)
- OM = 00 (toggle OFLAG output using alternating compare registers)

14.5.12.2 Timer Status and Control Register (SCTRL)

For complete register details, please refer to Section 14.6.8.

- OEN = 1 (output enable to allow OFLAG output to be put on an external pin. Set this bit as required)
- OPS = Any (make a choice)

Make certain the rest of the bits are cleared for this register. Interrupts in the comparator status and control (CSCTRL) register are enabled instead of this register.

14.5.12.3 Comparator Status and Control Register (CSCTRL)

For complete register details, please refer to Section 14.6.11.

- TCF2EN = 1 (allows interrupt to be issued when TCF2 is set)
- TCF1EN = 0 (does not allow interrupt to be issued when TCF1 is set)
- TCF1 = 0 (clear timer Compare Flag 1. This is set when CNTR register equals COMP1 register value and OFLAG is low)
- TCF2 = 0 (clear timer compare flag 2. This is set when CNTR register equals COMP2 register value and OFLAG is high)
- CL1[1:0] = 10 (load compare1 register with CMPLD1 register when TCF2 is asserted)
- CL2[1:0] = 01 (load compare2 register with CMPLD2 register when TCF1 is asserted)

14.5.12.4 Interrupt Service Routines

To service the TCF2 interrupts generated by the timer, the interrupt controller must be configured, enabling the interrupts for the particular timer being used. Additionally, it is necessary to write an interrupt service routine to do the following at a minimum:

- Clear the TCF2 and TCF1 flags
- Calculate and write new values for both CMPLD1 and CMPLD2

14.5.12.5 Timing

Figure 14-7 contains the timing to use the compare preload feature. The compare preload cycle begins with a compare event on COMP2, causing TCF2 to be asserted. COMP1 is loaded with the value in the CMPLD1, one IPBus clock later. Additionally, an interrupt is asserted by the timer and the interrupt service routine is executed. During this time both comparator load registers are updated with new values. When TCF1 is asserted, COMP2 is loaded with the value in CMPLD2. On the subsequent TCF2 event, COMP1 is loaded with the value in CMPLD1. The cycle starts over again as an interrupt is asserted and the interrupt service routine clears TCF1 and TCF2, calculating new values for CMPLD1 and CMPLD2.

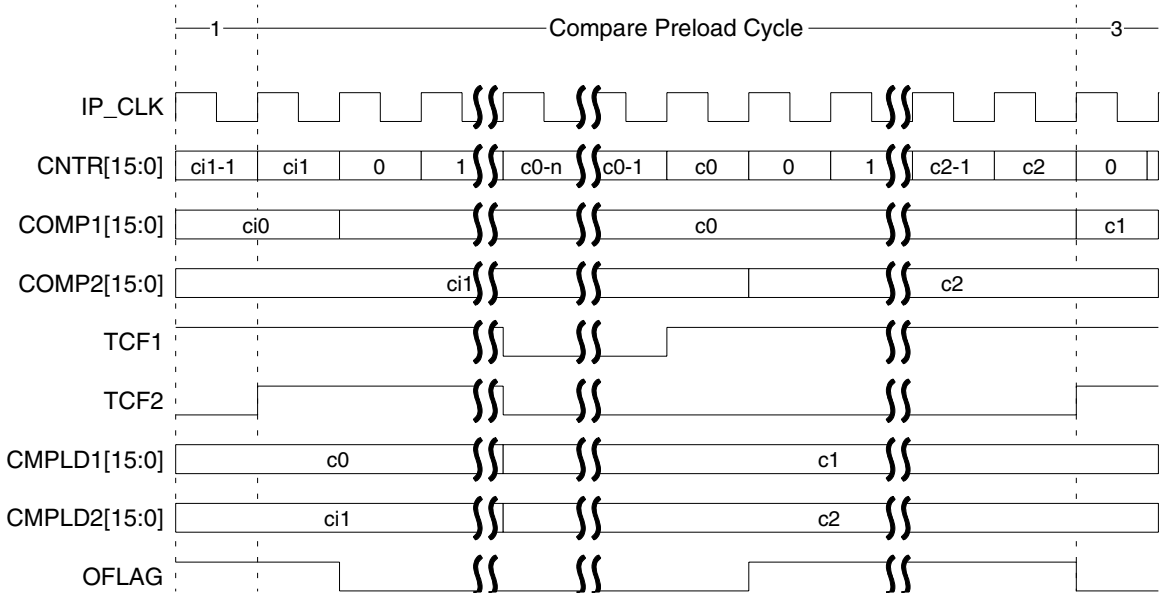


Figure 14-7. Compare Preload Timing

14.6 Register Description

A register address is the sum of a base address and an address offset. The base address is defined at the device level and the address offset is defined at the module level. Table 14-1 lists the TMR registers in ascending address, including their acronyms and address offset of each of the 44 registers. Make certain to check which timer channels have external I/O.

Device	Peripheral	Base Address
56F80xx	TMRA0	$00F000
	TMRA1	$00F010
	TMRA2	$00F020
	TMRA3	$00F030
	TMRB0	$00F040
	TMRB1	$00F050
	TMRB2	$00F060
	TMRB3	$00F070

Table 14-1. TMR Register Summary

Register Address Offsets	Register Acronym[1]	Register Name	Access Type	Chapter Location
Base + $0	COMP1	Compare 1 register	Read/Write	Section 14.6.1
Base + $1	COMP2	Compare 2 register	Read/Write	Section 14.6.2
Base + $2	CAPT	Capture register	Read/Write	Section 14.6.3
Base + $3	LOAD	Load register	Read/Write	Section 14.6.4
Base + $4	HOLD	Hold register	Read/Write	Section 14.6.5
Base + $5	CNTR	Counter register	Read/Write	Section 14.6.6
Base + $6	CTRL	Control registers	Read/Write	Section 14.6.7
Base + $7	SCTRL	Status and control register	Read/Write	Section 14.6.8
Base + $8	CMPLD1	Comparator load 1 register	Read/Write	Section 14.6.9
Base + $9	CMPLD2	Comparator load 2 register	Read/Write	Section 14.6.10
Base + $A	CSCTRL	Comparator status and control register	Read/Write	Section 14.6.11
Base + $B	FILT	Input filter register	Read/Write	Section 14.6.12
Base + $F[2]	ENBL	Channel enable register	Read/Write	Section 14.6.13

1. In the device's Data Sheet Timer acronyms are made globally unique with the addition of a prefix of the form TMR*x n,* where *x* = A, B, C . . . to identify multiple Quad Timers and *n* = 0, 1, 2, 3 . . . to identify individual timers within a Quad Timer.
2. Channel Enable register is only available for TIMERA0 and TIMERB0.

Bits of each of the 13 registers are illustrated in Figure 14-8. Details of each follow.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	COMP1	R	COMPARISON 1															
		W																
$1	COMP2	R	COMPARISON 2															
		W																
$2	CAPT	R	CAPTURE															
		W																
$3	LOAD	R	LOAD															
		W																
$4	HOLD	R	HOLD															
		W																
$5	CNTR	R	COUNTER															
		W																
$6	CTRL	R	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
		W																
$7	SCTRL	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE MODE		MSTR	EEOF	VAL	0	OPS	OEN
		W														FORCE		
$8	CMPLD1	R	COMPARATOR LOAD 1															
		W																
$9	CMPLD2	R	COMPARATOR LOAD 2															
		W																
$A	CSCTRL	R	DBG_EN		0	0	0	0	0	0	TCF2 EN	TCF1 EN	TCF2	TCF1	CL2		CL1	
		W																
$B	FILT	R	0	0	0	0	0	FILT_CNT			FILT_PER							
		W																
$F[1]	ENBL	R	0	0	0	0	0	0	0	0	0	0	0	0	ENBL			
		W																

R	0	Read as 0
W		Reserved

1. Channel Enable Register is only available for TIMERA0 and TIMERB0.

Figure 14-8. TMR Register Map Summary

14.6.1 Compare 1 (COMP1) Register

This read/write timer compare 1 (COMP1) register stores the value used for comparison with counter value.

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON 1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-9. Compare 1 (COMP1) Register

14.6.2 Compare 2 (COMP2) Register

This read/write timer compare 2 (COMP2) register stores the value used for comparison with counter value.

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON 2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-10. Compare 2 (COMP2) Register

14.6.3 Capture (CAPT) Register

This read/write timer capture (CAPT) register stores the values captured from the counters. Please see Section 14.6.8.9 for information explaining which inputs generate a capture of the counter's value.

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

Figure 14-11. Capture (CAPT) Register

14.6.4 Load (LOAD) Register

This read/write timer load (LOAD) register stores the value used to load the counter.

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

Figure 14-12. Load (LOAD) Register

14.6.5 Hold (HOLD) Register

This read/write register stores the channel's value whenever any timer counter (CNTR) register is read.

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

Figure 14-13. Hold (HOLD) Register

14.6.6 Counter (CNTR) Register

This is a read/write timer counter (CNTR) register.

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

Figure 14-14. Counter (CNTR) Register

14.6.7 Control (CTRL) Register

This is the timer control (CTRL) register. For details regarding the timer settings required to implement variable frequency PWM operation, please refer to Section 14.5.12.1.

Base + $6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-15. Control (CTRL) Register

14.6.7.1 Count Mode (CM)—Bits 15–13

These bits control the basic counting behavior of the counter.

- 000 = Stop mode, no operation
- 001 = Counting mode, count rising/falling edges of primary source[1]
  - Rising edges if IPS = 0 (see Section 14.6.8.7)
  - Falling edges if IPS = 1 (see Section 14.6.8.7)
- 010 = Edge count mode, count rising and falling edges of primary source
- 011 = Gated count mode, count rising edges of primary source while secondary input high active
- 100 = Quadrature count mode, uses primary and secondary sources
- 101 = Signed count mode, count edges of primary source
  - IPS = 0 then secondary Input = 0; count up on rising edges of primary input
  - IPS = 0 then secondary Input = 1; count down on rising edges of primary input
  - IPS = 1 then secondary Input = 0; count up on falling edges of primary input
  - IPS = 1 then secondary Input = 1; count down on falling edges of primary input
- 110 = Triggered count mode, edge of secondary source triggers primary count until compare
- 111 = Cascaded counter mode (up/down)[2]

1. If Primary count source is IPBus clock, divide by one, then only rising edges are counted regardless of IPS value.
2. Primary count source must be set to one of the counter outputs.

14.6.7.2 Primary Count Source (PCS)—Bits 12–9

These bits select the primary count source.

- 0000 = Counter 0 pin (T0)
- 0001 = Counter 1 pin (T1)
- 0010 = Counter 2 pin (T2)
- 0011 = Counter 3 pin (T3)
- 0100 = Counter 0 OFLAG
- 0101 = Counter 1 OFLAG
- 0110 = Counter 2 OFLAG
- 0111 = Counter 3 OFLAG3
- 1000 = Prescaler (system clock or 3× system clock[1] divide by 1)
- 1001 = Prescaler (system clock or 3× system clock[2] divide by 2)
- 1010 = Prescaler (system clock or 3× system clock[1] divide by 4)
- 1011 = Prescaler (system clock or 3× system clock[1] divide by 8)
- 1100 = Prescaler (system clock or 3× system clock[1] divide by 16)
- 1101 = Prescaler (system clock or 3× system clock[1] divide by 32)
- 1110 = Prescaler (system clock or 3× system clock[1] divide by 64)
- 1111 = Prescaler (system clock or 3× system clock[1] divide by 128)

NOTE:

A timer selecting its own output for input is not a legal choice. The result is no counting.

14.6.7.3 Secondary Count Source (SCS)—Bits 8–7

These bits identify the external input pin to be used as a count command. The selected input can trigger the timer to capture the current value of the CNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the IPS bit of the SCTRL register.

- 00 = Counter 0 pin
- 01 = Counter 1 pin
- 10 = Counter 2 pin
- 11 = Counter 3 pin

1. 3× system clock must be selected via SIM_PCR register.
2. 3× system clock must be selected via SIM_PCR register.

14.6.7.4 Count Once (ONCE)—Bit 6

This bit selects continuous or one-shot counting modes.

- 0 = Count repeatedly
- 1 = Count until compare and then stop. If counting up, successful compare occurs when the counter reaches a COMP1 value. If counting down, successful compare occurs when the counter reaches a COMP2 value. When the compare occurs, the timer module changes its Count Mode to Stop Mode (CM=0).

14.6.7.5 Count Length (LENGTH)—Bit 5

This bit determines whether the counter counts to the compare value and then reinitializes itself to the value specified in the load register, or the counter continues counting past the compare value, to the binary roll-over.

- 0 = Roll-over
- 1 = Count till compare, then reinitialize. If counting up, successful compare occurs when the counter reaches COMP1 value. If counting down, successful compare occurs when the counter reaches COMP2 value.

When output mode 100 is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until COMP1 value is reached, re-initializes, then counts until COMP2 value is reached, re-initializes, then counts until COMP1 value is reached, etc.

14.6.7.6 Count Direction (DIR)—Bit 4

This bit selects either the normal count direction up, or the reverse down direction.

- 0 = Count up
- 1 = Count down

14.6.7.7 Co-Channel Initialization (CoINIT)—Bit 3

This bit enables another counter/timer within the same module to force the reinitialization of this counter/timer when it has an active compare event. The master channel forcing reinitialization has MSTR bit set. Please see Section 14.6.8.10.

- 0 = Co-channel counter/timers cannot force a reinitialization of this counter/timer
- 1 = Co-channel counter/timers can force a reinitialization of this counter/timer

14.6.7.8 Output Mode (OM)—Bits 2–0

This bit field determines the mode of operation for the OFLAG output signal.

- 000 = Assert OFLAG 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[1]
- 101 = Set OFLAG on compare, cleared on secondary source input edge
- 110 = Set OFLAG on compare, cleared on counter roll-over
- 111 = Enable gated clock output while counter is active

NOTE:

Unexpected results may occur if OM field is set to use alternating compare registers (mode 100) and the ONCE bit are set.

14.6.8 Status and Control (SCTRL) Register

This is the timer status/control (SCTRL) register. For details regarding the timer settings required to implement variable frequency PWM operation, please refer to Section 14.5.12.2.

Base + $7	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	0	0	0	0	0	0	0	0	0

Figure 14-16. Status and Control (SCTRL) Register

14.6.8.1 Timer Compare Flag (TCF)—Bit 15

This bit is set when a successful compare occurs. Clear the bit by writing zero to it.

TCF asserts every time there is a compare event (either when counter = COMP1 or counter = COMP2).

14.6.8.2 Timer Compare Flag Interrupt Enable (TCFIE)—Bit 14

When set, the timer compare interrupt is enabled.

14.6.8.3 Timer Overflow Flag (TOF)—Bit 13

This bit is set when the counter rolls over its maximum or minimum value $FFFF or $0000, depending on count direction. Clear the bit by writing zero to it.

1. When using output mode 100, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, when the output mode is 100, the counter calculates until COMP1 value is reached., reinitializes, then counts until COMP2 value is reached, reinitializes, then counts until COMP1 value is reached, and so on.

14.6.8.4 Timer Overflow Flag Interrupt Enable (TOFIE)—Bit 12

When set, this bit enables interrupts when the TOF bit is set.

14.6.8.5 Input Edge Flag (IEF)—Bit 11

This bit is set when a positive input transition occurs on an input selected as a secondary count source while the counter is enabled. Clear the bit by writing zero to it.

NOTE:

Setting the IPS bit enables the detection of negative input edge transitions detection. Also, the control register's secondary count source (SCS) determines which external input pin is monitored by the detection circuitry.

14.6.8.6 Input Edge Flag Interrupt Enable (IEFIE)—Bit 10

When set, this bit enables interrupts if the IEF bit is set.

14.6.8.7 Input Polarity Select (IPS)—Bit 9

When set, this bit inverts the polarity of both the primary and secondary inputs.

14.6.8.8 External Input Signal (INPUT)—Bit 8

This read-only bit reflects the current state of the external input pin selected via the secondary count source after application of the IPS bit and filtering.

14.6.8.9 Input Capture Mode (CAPTURE MODE)—Bits 7–6

This bit field specifies the operation of the CAPT register as well as the operation of the input edge flag (IEF). The input source is the secondary count source (SCS).

Capture Mode	IPS	Action
00	x	Capture function is disabled
01	0	Load capture register on rising edge of input
	1	Load capture register on falling edge of input
10	0	Load capture register on falling edge of input
	1	Load capture register on rising edge of input
11	x	Load capture register on both edges of input

14.6.8.10 Master Mode (MSTR)—Bit 5

When set, this bit enables the compare register function output to be broadcasted to the other counter/timers in the module. This signal then can be used to reinitialize the other counters and/or force their OFLAG signal outputs. Other timer channels within the quad timer accept the reinitialization signal when their CoINIT bit is set. Please see Section 14.6.7.7.

14.6.8.11 Enable External OFLAG Force (EEOF)—Bit 4

When set, this bit enables the compare register from another counter/timer within the same module to force the state of this counter's OFLAG output signal.

14.6.8.12 Forced OFLAG Value (VAL)—Bit 3

This bit determines the value of the OFLAG output signal when a software triggered FORCE command occurs, that is when FORCE = 1, discussed in Section 14.6.8.13.

14.6.8.13 Force OFLAG Output (FORCE)—Bit 2

This write-only bit forces the current value of the VAL bit to be written to the OFLAG output. This bit is always read as zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if the counter is disabled. Setting this bit while the counter is enabled may yield unpredictable results.

14.6.8.14 Output Polarity Select (OPS)—Bit 1

This bit determines the polarity of the OFLAG output signal.

- 0 = True polarity
- 1 = Inverted polarity

14.6.8.15 Output Enable (OEN)—Bit 0

When set, this bit determines the direction of the external pin.

- 0 = The external pin is configured as an input.
- 1 = OFLAG output signal is driven on the external pin. The polarity of the signal is determined by the OPS bit.

14.6.9 Comparator Load 1 (CMPLD1) Register

This read/write register makes up the comparator 1 preload value for the CMPLD1 register of the corresponding channel in a timer module. Please see Section 14.4.2 for additional information. Please see Section 14.6.11.8 for information explaining how to control comparator loading.

Base + $8	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

Figure 14-17. Comparator Load 1 (CMPLD1) Register

14.6.10 Comparator Load 2 (CMPLD2) Register

This read/write register makes up the comparator 2 preload value for the CMPLD2 register of the corresponding channel in a timer module. Please see Section 14.4.2 for additional information. Please see Section 14.6.11.7 for information explaining how to control the loading of comparator 2.

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

Figure 14-18. Comparator Load 2 (CMPLD2) Register

14.6.11 Comparator Status/Control (CSCTRL) Register

Base + $A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DBG_EN		0	0	0	0	0	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

Figure 14-19. Comparator Status/Control (CSCTRL) Register

14.6.11.1 Debug Actions Enable (DBG_EN)—Bits 15-14

This bit field allows the TMR module to perform certain actions in response to the chip entering debug mode.

Table 14-2. DBG_EN Values

Value	Meaning
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 the OPS bit).
11	Both halt counter and force output to 0 during debug mode.

14.6.11.2 Reserved—Bits 13–8

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

14.6.11.3 Timer Compare 2 Interrupt Enable (TCF2EN)—Bit 7

An interrupt is issued when both this bit and the TCF2 bit are set.

14.6.11.4 Timer Compare 1 Interrupt Enable (TCF1EN)—Bit 6

An interrupt is issued when both this bit and the TCF1 bit are set.

14.6.11.5 Timer Compare Flag 2 (TCF2)—Bit 5

When set, this bit indicates a successful comparison of the timer and COMP2 register occurred. This bit is sticky, and remains set until explicitly cleared by writing zero to this bit location.

14.6.11.6 Timer Compare Flag 1 (TCF1)—Bit 4

When set, this bit indicates a successful comparison of the timer and COMP1 register occurred. This bit is sticky, and remains set until explicitly cleared by writing zero to this bit location.

14.6.11.7 Compare Load Control 2 (CL2)—Bit 3–2

These bits control when COMP2 is preloaded with the value from CMPLD2.

Table 14-3. Values for Compare Preload Control 2

Value	Meaning
00	Never preload
01	Load upon successful compare with the value in COMP1, that is when TCF1 is set
10	Load upon successful compare with the value in COMP2, that is when TCF2 is set
11	Reserved

14.6.11.8 Compare Load Control 1 (CL1)—Bit 1–0

These bits control when COMP1 is preloaded with the value from CMPLD1.

Table 14-4. Values for Compare Preload Control 1

Value	Meaning
00	Never preload
01	Load upon successful compare with the value in COMP1, that is when TCF1 is set
10	Load upon successful compare with the value in COMP2, that is when TCF2 is set
11	Reserved

14.6.12 Input Filter (FILT) Register

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

Figure 14-20. Input Filter (FILT) Register

14.6.12.1 Reserved—Bits 15–11

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

14.6.12.2 Input Filter Sample Count (FILT_CNT)—Bits 10–8

This bit field represents the number of consecutive samples required to agree prior to the input filter accepting an input transition. A value of 0x0 represents three samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency as described in Section 14.6.12.4.

14.6.12.3 Input Filter Sample Period (FILT_PER)—Bits 7–0

This bit field represents the sampling period (in system clock or 3xsystem clock[1] cycles) of the TMR input signals. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), the input filter is bypassed. The value of FILT_PER affects the input latency, described in Section 14.6.12.4.

14.6.12.4 Input Filter Considerations

The FILT_PER value should be set, assuring the sampling period is larger than the period of the expected noise. In this manner, a noise spike only corrupts one sample. The FILT_CNT 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 FILT_CNT + 3 power.

The values of FILT_PER and FILT_CNT must also be traded off 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) system clock[1] periods.

14.6.13 Channel Enable (ENBL) Register

Base + $F	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	ENABLE			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 14-21. Channel Enable (ENBL) Register

NOTE:

This register is only available for TIMERA0 and TIMERB0.

14.6.13.1 Reserved—Bits 15–4

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

14.6.13.2 Timer Channel Enable (ENBL)—Bits 3-0

If the prescaler is being used, this bit field enables it and the counter in each channel. Multiple ENBL bits can be concurrently set to synchronize the start of separate counters. If an ENBL bit is set, the corresponding channel starts the counter as soon as the COUNT MODE field has a value other than zero. When an ENBL bit is clear, the corresponding counter maintains its current value.

- 0 = Timer channel is disabled
- 1 = Timer channel is enabled (default)

1. 3x system clock must be selected via SIM PCR register.

14.7 Clocks

The Timer operates from the IPBus clock or high speed IPBus clock.

14.8 Interrupts

Each of the four timers in a timer module can generate an interrupt, serviced by the interrupt service routine (ISR) identified by the interrupt vector table. Please see the chip's data sheet for more information on the vector table.The ISR has to check the SCTRL register and the CSCTRL register for which of the five interrupt flag bits are high.

Table 14-5 describes each of these flag bits.

Table 14-5. Timer Interrupt Flags

Acronym	Name	Located In		Location
		SCTRL	CSCTRL	
TCF	Timer compare flag	x	—	Section 14.6.8.1
TOF	Timer overflow flag	x	—	Section 14.6.8.3
IEF	Input edge flag	x	—	Section 14.6.8.5
TCF1	Timer compare flag 1	—	x	Section 14.6.11.6
TCF2	Timer compare flag 2	—	x	Section 14.6.11.5

The ISR should reset each set flag bit by writing zero to the bit.

14.8.1 Description of Interrupt Operation

14.8.1.1 Timer Compare Interrupts

These interrupts are generated when a successful compare occurs between a counter and it's compare registers and while the timer compare flag interrupt enable (TCFIE) is set in the SCTRL register. These interrupts are cleared by writing a zero to the TCF bit in the appropriate SCTRL register.

When a TCFIE bit is set in the SCTRL register and the CMPLD registers are available, one of the following two interrupts is also asserted.

14.8.1.2 Timer Compare 1 Interrupts

These interrupts are generated when a successful compare occurs between a counter and its COMP1 register while the timer compare flag 1 interrupt enable (TCF1EN) bit is set in the CSCTRL register. These interrupts are cleared by writing zero to the TCF1 bit in the appropriate CSCTRL register.

14.8.1.3 Timer Compare 2 Interrupts

These interrupts are generated when a successful compare occurs between a counter and its COMP2 register while the timer compare 2 interrupt enable (TCF2EN) bit is set in the CSCTRL register. These interrupts are cleared by writing zero to the TCF1 bit in the appropriate CSCTRL register.

14.8.1.4 Timer Overflow Interrupts

These interrupts are generated when a counter rolls over its maximum value while the timer overflow flag interrupt enable (TOFIE) bit is set in the SCTRL register. These interrupts are cleared by writing zero to the TOF bit of the appropriate SCTR register.

14.8.1.5 Timer Input Edge Interrupts

These interrupts are generated by a transition of the input signal, either positive or negative, depending on IPS setting and while the input edge flag interrupt enable (IEFIE) bit is set in the SCTRL register. These interrupts are cleared by writing zero to the IEF bit of the appropriate SCTRL register.

14.9 Resets

The TMR module can only be reset b y the chip wide $\overline{\text{RST}}$ signal. This forces all registers to their reset state and clears the OFLAG signal. The counter is turned off until the settings in the CTRL register are changed.

Table 14-6. Document Revision History for Chapter 14

Version History	Description of Change
Rev. 3	• Added revision history table. • Added the following sentence to Section 14.6.7.4 (CTRL[ONCE]=1): "*When the compare occurs, the timer module changes its Count Mode to Stop Mode (CM=0).*"

Chapter 15
Voltage Regulator (VREG)

15.1 Introduction

This module provides an on-chip mechanism to regulate an external 3.3 V supply down to 2.5 V levels for use with the internal core logic. It also provides a precision voltage to the on-chip relaxation oscillator and PLL using a small regulator. The PLL enjoys additional noise immunity because of the small regulator. On-board regulators are stable and have sufficient capacity and dynamic response allowing the host chip to operate correctly at all times and under all combinations of specified loads, frequencies, voltages, temperatures, and process variations.

15.2 Features

Qualities of the linear voltage regulator contain:

- Provide a 2.5 V ±10% accuracy
- Provide a maximum current of 125 mA for the large regulator
- Provide a maximum current of a 4 mA for the small regulator

15.3 Block Diagram

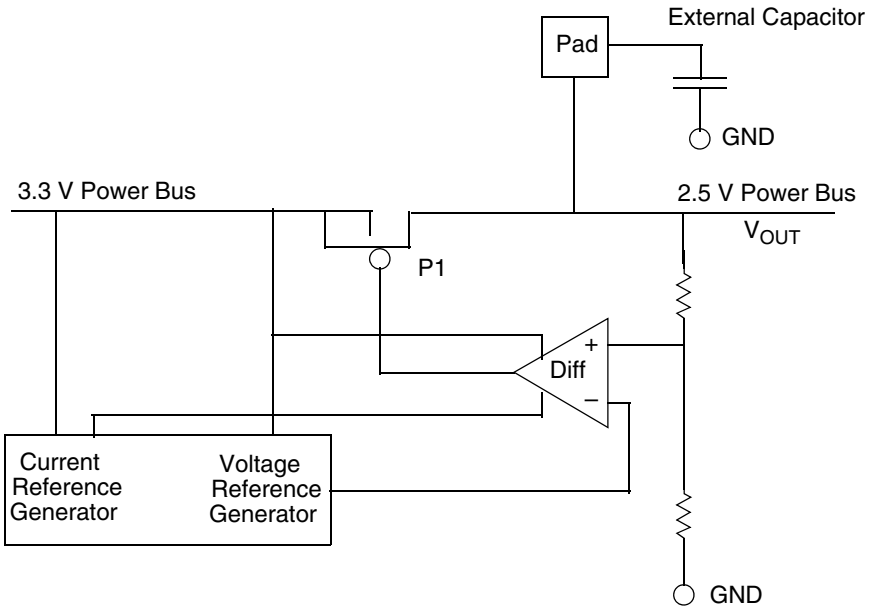


Figure 15-1. Large Voltage Regulator Block Diagram

15.4 Functional Description

Internal regulators (typically one for logic, and one for analog) are added to regulate down from 3.3 V to 2.5 V. The large core voltage regulator look like the circuit illustrated in Figure 15-1. Small voltage regulator do not have a pad.

The large regulator, illustrated in Figure 15-1, is a linear series-pass low drop-out regulator. It uses a very large p-channel MOSFET as the pass device, and it uses an equally sized overload protection device. It takes an input of 3.3V ±10% and provides a regulated 2.5 V output at a maximum current level of 125 mA.

The large regulator, illustrated in Figure 15-1, requires external 4.4 µF capacitors to be tied to the V_{CAP} pin(s) to help maintain stability. For optimal transient performance, the 4.4 µF filter cap should be low equivalent series resistance (ESR) multi-layer ceramic chip (MLCC) caps and should be placed as close as possible to the chip's V_{CAP} pin(s). Using electrolytic capacitors is discouraged because of their high ESR. Using high ESR capacitors causes the regulator to have poor transient load regulation performance. Filter capacitors can be used, since they improve the transient load regulation performance. However, any filter capacitor used must be at least 4.4 µF.

15.5 Operating Modes

This is an analog part excited by the introduction of the input voltage. Its modes of operation are ON and standby capability.

15.6 Memory Map

This device has no memory mapped registers except the SIM power control (SIM_PWR) register provided below.

15.7 Register Description

15.7.1 SIM Power Control Register (SIM_PWR)

This register controls the standby mode of the large regulator. The large regulator derives the core digital logic power supply from the I/O power supply. In some circumstances the large regulator may be placed in a reduced power standby mode without interfering with part operation. In this mode voltage will be regulated down to 2.5 V, but the current output will be reduced. Please refer to the overview of power down modes and the overview of clock generation for more information on the use of large regulator standby.

Base + $8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LRSTDBY	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-2. SIM Power Control (SIM_PWR) Register

15.7.1.1 Large Regulator Standby Mode (LRSTDBY)—Bits 1–0

- 00 = Large regulator is in normal mode
- 01 = Large regulator is in standby (reduced power) mode
- 10 = Large regulator is in normal mode and the LRSTDBY bit field is write-protected until the next reset
- 11 = Large regulator is in standby mode and the LRSTDBY bit field is write-protected until the next reset

15.8 Pin Descriptions

Table 15-1. Signal Properties

Name	I/O Type	Function	Reset State	Notes
V_{SS}	DC Input	Ground	N/A	—
V_{DD}	DC Source	Input voltage supply	N/A	—
V_{CAP}	—	Capacitor	N/A	Not included on the smaller regulators

15.8.1 Input Voltage (V_{DD})

V_{DD} is the input voltage of 3.3 V typically required by the regulator to convert to 2.5 V.

15.8.2 Capacitor Pin(s) (V_{CAP})

4.4 µF capacitor must be connected to the V_{CAP} pin(s) on the larger regulator for proper operation.

NOTE:

This pin is intended to be a static DC signal from power up to shut down. Do not toggled this pin for power savings during operation.

15.9 Clocks

There are no clocks used by this module.

15.10 Resets

There are no resets for this device.

15.11 Interrupts

There are no interrupts generated by this module.

Table 15-2. Document Revision History for Chapter 15

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 16
Programmable Interval Timer (PIT)

16.1 Introduction

The programmable interval timer (PIT) module contains a 16-bit up counter, a modulo register, and a control register. The modulo and control registers are read/writable. The counter is read only.

The modulo register is loaded with a value to count to and the prescaler is set to determine the counting rate. When enabled, the counter counts up to the modulo value and sets a flag (and an interrupt request if enabled), resets to $0000, and resumes counting.

16.2 Features

The PIT module design includes these distinctive characteristics:

- 16-bit counter/timer
- Programmable count modulo
- Slave mode allows synchronization of multiple PIT count enables

16.3 Block Diagram

The block diagram of the PIT is shown in Figure 16-1.

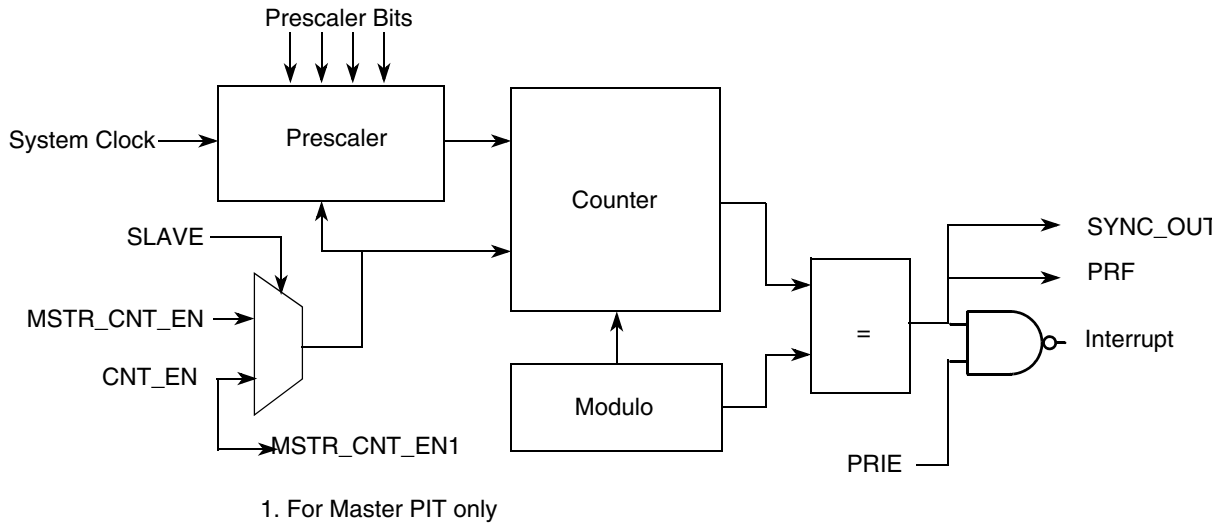


Figure 16-1. PIT Block Diagram

16.4 Functional Description

The purpose of the PIT is to create a repeated interrupt request at a programmable time interval. The periodic rate is determined based on the system clock, the prescaler value, and the modulo value as shown in the following equation:

Interrupt Rate = System Clock Rate ÷ ((2^prescaler) × Modulo Value)

When using the PIT, set the prescaler and modulo values prior to setting CNT_EN. Changing prescaler or modulo settings with CNT_EN set may cause unexpected operation.

The roll-over flag is set upon rolling over the modulo value back to $0000. See Figure 16-2 for an example of PRF timing using a prescaler of $2 (system clock divided by 4).

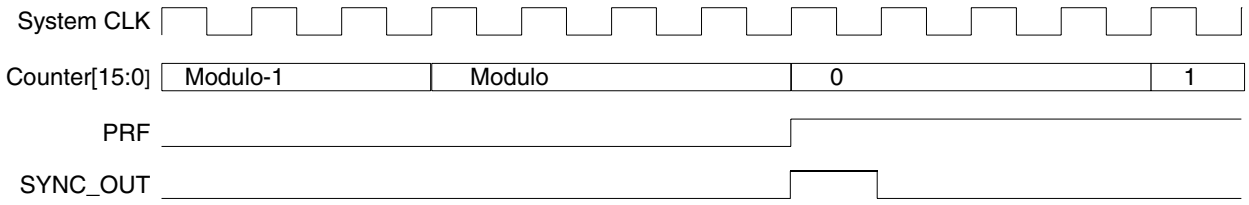


Figure 16-2. Example of Timing

The roll-over flag can be cleared by writing zero to the PRF bit position or by clearing CNT_EN.

16.5 Operation Modes

16.5.1 Slave Mode

When using slave mode to synchronize several PIT modules only the CNT_EN signal is shared not the clocking for the counters. This means each slave PIT must have their PRESCALER field and MOD register programmed prior to setting CNT_EN in the master PIT. The following figure illustrates the connection between the master PIT (PIT0) and the possible slave PITs (PIT1 - PITx).

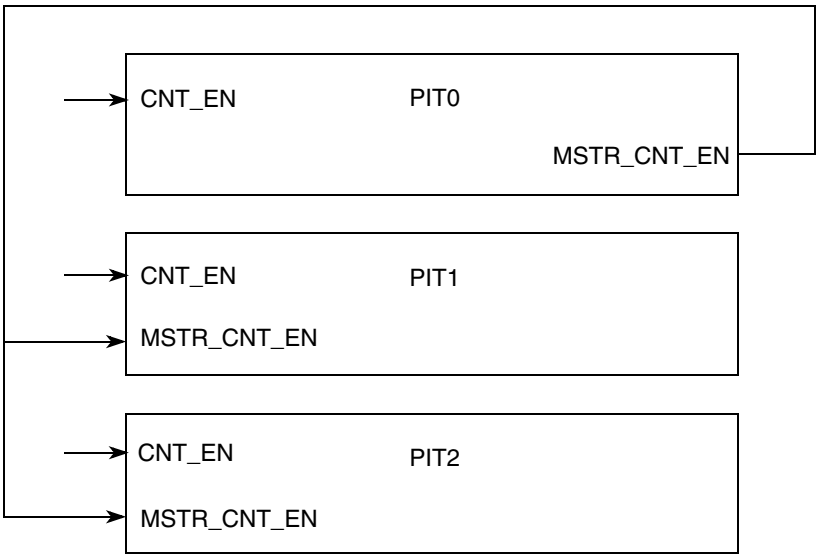


Figure 16-3. CNT_EN Connection Between Multiple PITs

16.5.2 Low Power Modes

16.5.2.1 Wait Mode

If the CNT_EN bit is set prior to entering wait mode, then the PIT continues to count. It can wake up the chip by asserting its interrupt upon reaching the modulo value.

16.5.2.2 Stop Mode

Stop mode operation depends on whether the system integration module (SIM) is set to allow the PIT to be clocked in stop mode. See SIM SD1 register. If not, the PIT counter does not operate during stop mode, but retains its current settings. If CNT_EN is set, then the counter resumes counting upon exit of stop mode assuming the exit is not caused by a reset. If the PIT does receive clocks while the chip is in stop mode, operation continues normally.

16.5.2.3 Debug Mode

If the CNT_EN bit is set prior to the chip entering debug mode, the PIT continues to count during debug mode.

16.6 Signal Description

The PIT module has no external signals. The PIT interfaces to the peripheral bus.

16.6.1 PIT Interrupt ($\overline{\text{PIT\_INT}}$)

This bit is set when the PRF flag is generated and the interrupt enable PRIE is also set.

16.6.2 Count Enable (COUNT_EN)

This output reflects the value of the CNT_EN bit.

16.6.3 Sync Output (SYNC_OUT)

This output is set for a single system clock cycle when the CNTR value rolls over from the MOD value. This signal is typically used as a trigger for DACs or ADCs.

16.6.4 Master Count Enable (MSTR_CNT_EN)

This input is used when the SLAVE bit is set instead of using the CNT_EN bit.

16.7 Register Description

The address of a register is the sum of a base address and an address offset. The base address is defined at the MCU level and the address offset is defined at the module level. The base address given for each register is PIT_BASE.

Table 16-1. PIT Memory Map

Device	Peripheral	Base Address
56F80xx	PIT0	$00F190
	PIT1	$00F1A0
	PIT2	$00F1B0

Table 16-2 lists the PIT registers in ascending address order, including their acronyms and address offset of each register.

Table 16-2. PIT Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL	Control register	Read/Write	Section 16.7.1
Base + $1	MOD	Modulo register	Read/Write	Section 16.7.2
Base + $2	CNTR	Counter register	Read-Only	Section 16.7.3

There are three registers in the PIT peripheral summarized in Figure 16-4**.** Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL	R	SLAVE	0	0	0	0	0	0	0	0	PRESCALER				PRF	PRIE	CNT_EN
		W																
$1	MOD	R	MODULO_VALUE															
		W																
$2	CNTR	R	COUNTER_VALUE															
		W																

R	0	Read as 0
W		Reserved

Figure 16-4. PIT Register Map Summary

16.7.1 Control (CTRL) Register

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SLAVE	0	0	0	0	0	0	0	0	PRESCALER				PRF	PRIE	CNT_EN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-5. Control (CTRL) Register

16.7.1.1 Slave (SLAVE)—Bit 15

This field is used to place this PIT module in slave mode. This means the CNT_EN field is ignored and instead this PIT uses the master count enable signal broadcast from the PIT0 module. This bit allows synchronization of the counts across multiple PIT modules. This bit is only useful in designs with multiple PIT modules. Setting this bit in the (master) PIT0 module has no effect as its own CNT_EN field is also the master count enable.

- 0 = CNT_EN from this PIT is used to control operation (default)
- 1 = CNT_EN from master PIT is used to control operation

16.7.1.2 Reserved—Bits 14–7

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

16.7.1.3 Prescaler (PRESCALER)—Bits 6–3

This field is used to select the presaging of the system clock to determine the counting rate of the PIT.

Table 16-3. Prescaler Values

Value	PIT Clock Rate
0000	System Clock
0001	System Clock Divided By 2
0010	System Clock Divided By 4
0011	System Clock Divided By 8
0100	System Clock Divided By 16
0101	System Clock Divided By 32
0110	System Clock Divided By 64
0111	System Clock Divided By 128
1000	System Clock Divided By 256
1001	System Clock Divided By 512
1010	System Clock Divided By 1024
1011	System Clock Divided By 2048
1100	System Clock Divided By 4096
1101	System Clock Divided By 8192
1110	System Clock Divided By 16384
1111	System Clock Divided By 32768

NOTE:

Changing prescaler value with CNT_EN set may cause unexpected operation.

16.7.1.4 PIT Roll-Over Flag (PRF)—Bit 2

This bit is set when the counter rolls over to $0000 after matching the value in the MOD register. This bit is cleared by reading the CTRL register with PRF set and then writing a zero to this bit position. This bit can also be cleared by setting the CNT_EN bit to zero. Writing one to the PRF bit position has no effect.

- 0 = PIT counter has not rolled over the modulo value (default)
- 1 = PIT counter has reached the modulo value

16.7.1.5 PIT Roll-Over Interrupt Enable (PRIE)—Bit 1

This bit enables the PIT roll-over interrupt when the PRF bit becomes set.

- 0 = PIT roll-over interrupt disabled (default)
- 1 = PIT roll-over interrupt enabled

16.7.1.6 Count Enable (CNT_EN)—Bit 0

This bit enables the PIT prescaler and counter. When this bit is clear the counter remains at, or returns to a $0000 value. The PRF bit is also reset when CNT_EN is clear. This field is ignored when the SLAVE bit is set and the count enable signal from the master PIT is used instead.

- 0 = PIT counter reset (default)
- 1 = PIT counter active

16.7.2 Modulo (MOD) Register

This read/write register stores the modulo value for the PIT counter. When the PIT counter rolls over to $0000 from this value, the PRF bit becomes set and the PIT counter resumes counting from $0000.

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	MODULO_VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-6. Modulo (MOD) Register

Note: Changing prescaler value with CNT_EN set may cause unexpected operation.

16.7.3 Counter (CNTR) Register

This read-only register contains the current value of the PIT counter. Clearing CNT_EN resets the counter to $0000. When the PIT counter rolls over the modulo value, the PRF bit becomes set and the PIT counter resumes counting from $0000.

Base + $2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNTER_VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-7. Counter (CNTR) Register

16.8 Clocks

The PIT only receives the system clock. This clock must be enabled via SIM PCE1 register.

16.9 Interrupts

The PIT module can generate a single interrupt when the counter rolls over to $0000 from the modulo value.

16.10 Reset

The PIT module can only be reset by the $\overline{\text{RST}}$ signal. This forces all registers to their reset state, clearing the PRF signal if it is asserted. The counter is turned off until the settings in the CTRL register are changed.

Table 16-4. Document Revision History for Chapter 16

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 17
Digital-to-Analog Converter (DAC)

17.1 Introduction

This chapter discusses the operation, programming model, operating modes, and initialization of the digital-to-analog converter (DAC). The DAC accepts a 12-bit digital signal and creates an analog output varying from ~V_{SSA} to ~V_{DDA}. The DAC module consists of a conversion unit, an output amplifier, and the associated digital control blocks. The DAC output can be used internally as an input into peripherals such as the analog comparator peripheral and optionally, depending on the specific chip implementation, to an output pin. The DAC peripheral is optimized to be both low power and compact.

17.2 Features

DAC features include:

- 12-bit resolution
- Two microsecond conversion rate
- Power down mode
- Output can be routed to internal comparator, ADC, or optionally off chip
- DAC can drive 3 KΩ, 400 pF load
- Choice of asynchronous or synchronous updates
  - Sync input can be connected to on chip PITs, TMRs, PWM reload flag, or GPIO inputs
- Automatic mode allows the DAC to generate its own output waveforms including square, triangle, and sawtooth waveforms
- Automatic mode allows programmable period, update rate, and range

17.3 Block Diagram

The digital-to-analog converter (DAC) configuration is provided in the following block diagram.

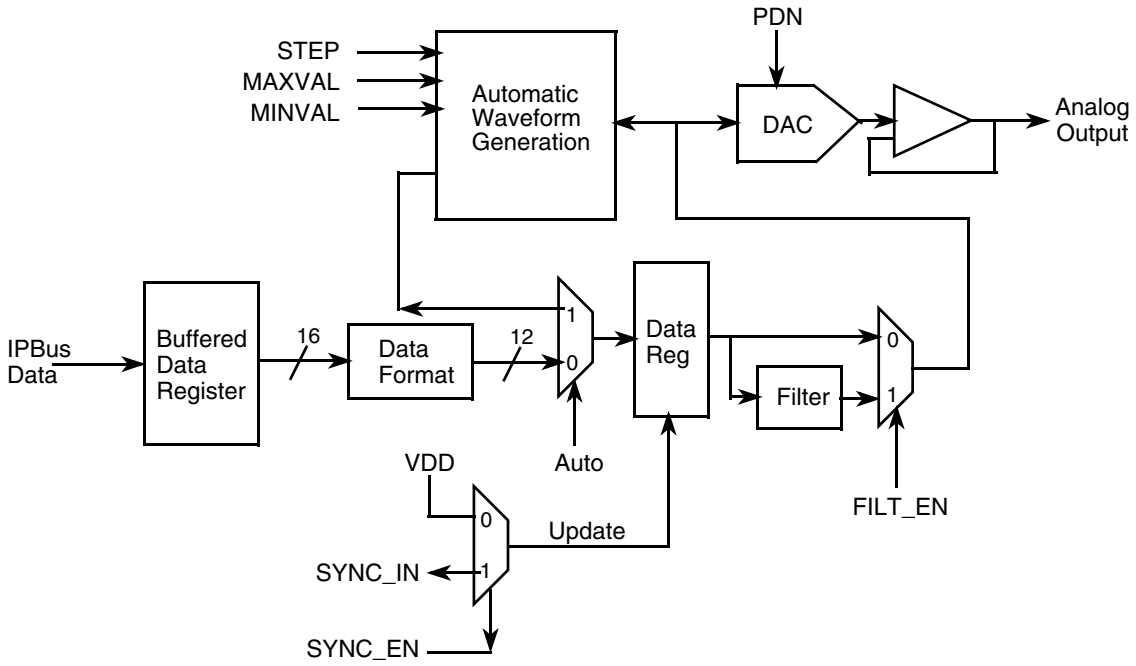


Figure 17-1. DAC Block Diagram

17.4 Functional Description

17.4.1 Normal Mode

The DAC receives data words via a memory mapped register on the IPBus (DATA). This digital word is applied to the DAC inputs based on the SYNC_EN bit. The analog DAC will generate an analog representation of the digital word in less than two microseconds.

17.4.2 Automatic Mode

This mode allows the automatic generation of triangle, sawtooth, 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. When the SYNC_IN input indicates it is time to update the data presented to the DAC, the STEP value is added to/subtracted from the current DATA value and DATA is updated. If UP bit is set, then STEP is added to DATA each update until maximum value (MAXVAL) is reached. At this point the generator begins subtracting STEP from DATA if DOWN is set (down counting enabled) or reload minimum value (MINVAL) if DOWN is clear (no down counting). Similarly, upon DATA reaching MINVAL while counting down, the generator resumes up counting if UP bit is set, or reload MAXVAL if UP is clear (up counting disabled). The initial direction of the count is dependent on which bit (UP or DOWN) is set last. The following figures illustrate several examples of automatically generated waveforms. The waveforms shown are ideal. Actual waveforms are limited by the slew rate of the DAC output.

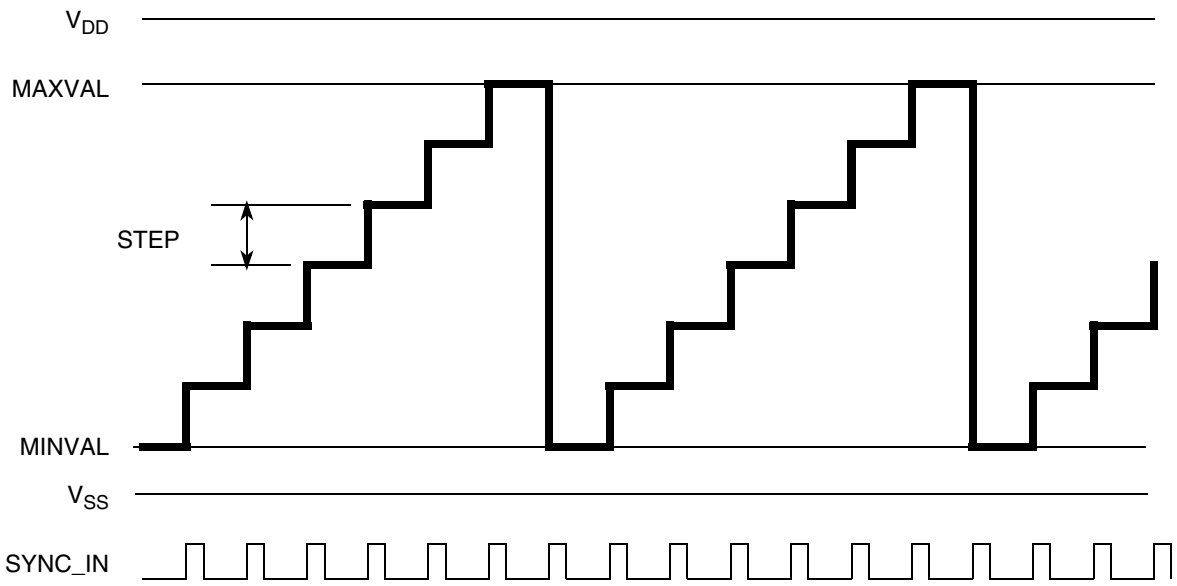


Figure 17-2. Sawtooth Waveform Example with UP = 1 and DOWN = 0

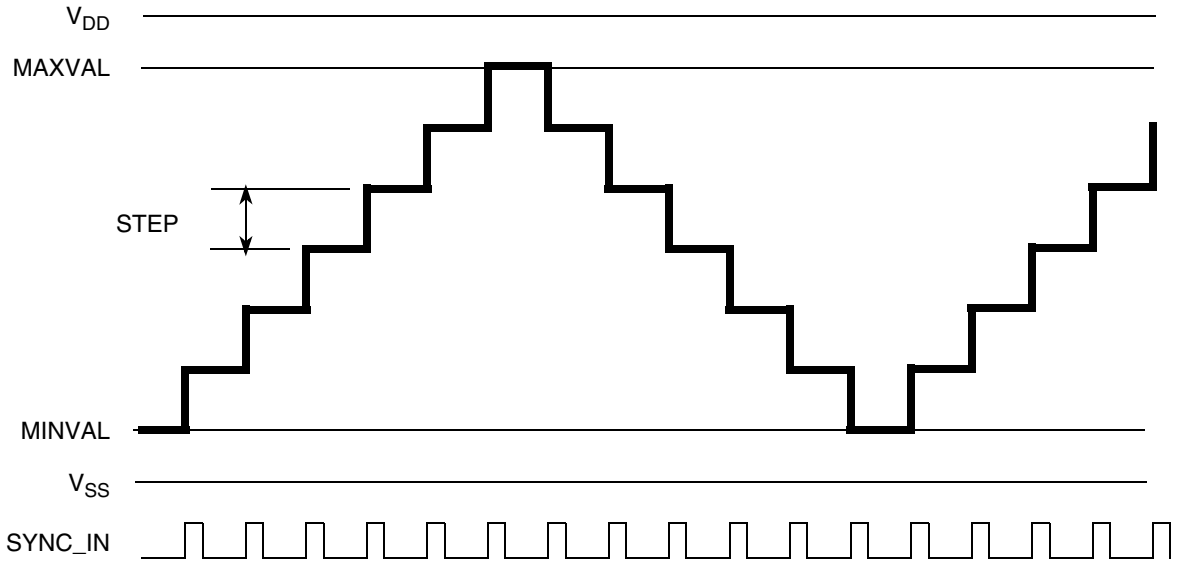


Figure 17-3. Triangle Waveform Example with UP = 1 and DOWN = 1

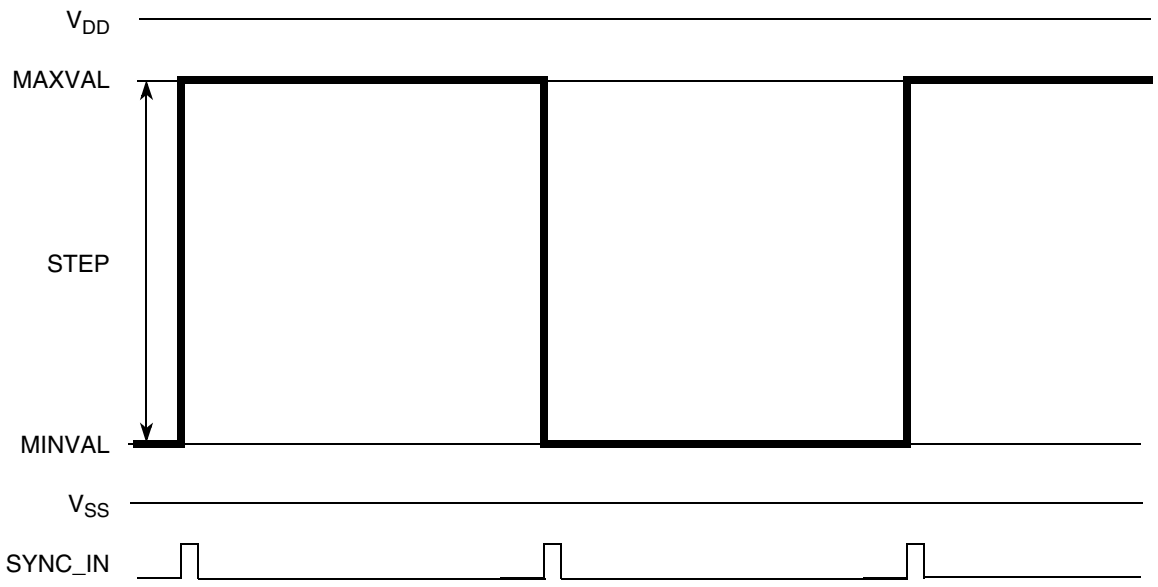


Figure 17-4. Square Wave Waveform Example with UP = 1 and DOWN = 0

The above examples show the waveform period is a function of the difference between MAXVAL and MINVAL, the STEP size, and the update rate as shown below.

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

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.

17.4.3 Programming Example

Assume the desire to create a waveform to go down from 3.0 V to 1.5 V in 1 millisecond. First, calculate MAXVAL and MINVAL. Based on each DAC least significant bit (LSB) representing 0.806 mV the results are MAXVAL = $E8A and MINVAL = $745. This represents a difference of 1861 LSBs to be accomplished in one millisecond. A one-millisecond time period means the DAC can safely be updated 500 times since the DAC has a two-microsecond conversion time. To go from MAXVAL to MINVAL in 500 steps requires STEP to be $004 after rounding the result of 1861/500. To go from MAXVAL to MINVAL with a STEP size of $004 requires 465 steps. To keep the waveform period of one millisecond using 465 updates calls for an update period of 465 kHz. Assuming operation is using a 32 MHz system clock, a timer module can be programmed to pulse the SYNC_IN input every 32000000/465000 = 69 counts.

When the timer module is programmed, along with the MAXVAL, MINVAL, and STEP registers, the DATA register should be written with a value equal to MAXVAL (because only a count down is occurring) as a starting point. When writing to DATA, STEP, MAXVAL, and MINVAL be certain FORMAT is the desired value and the data values are properly justified to match FORMAT. SYNC_EN, DOWN, and AUTO should be set. UP and PDN should be cleared. FILT_CNT and FILT_EN can be set to suppress glitches on the output. The desired waveform begins within 12 microseconds from the clearing of PDN and continues until PDN is set or the timer is stopped.

17.4.4 Sources of Waveform Distortion

17.4.4.1 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. These glitches can be avoided by setting FILT_EN and a suitable value for FILT_CNT in the CTRL register. This causes the DAC to hold its current output for a number of clock cycles equal to FILT_CNT while the switching glitches settle out. After the filter time is satisfied, the output smoothly slews to the new value.

17.4.4.2 Slew Effects

The example waveforms are ideal waveforms and show transitions as step functions. In reality the DAC output has a finite slew rate designed to round 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).

17.4.4.3 Clipping Effects (AUTO Mode Only)

One form of clipping can occur during automatic waveform generation when the difference between MAXVAL and MINVAL isn't a (near) even multiple of the STEP value. This results in a less than whole step as the waveform approaches MAXVAL and MINVAL. An example is drawn below.

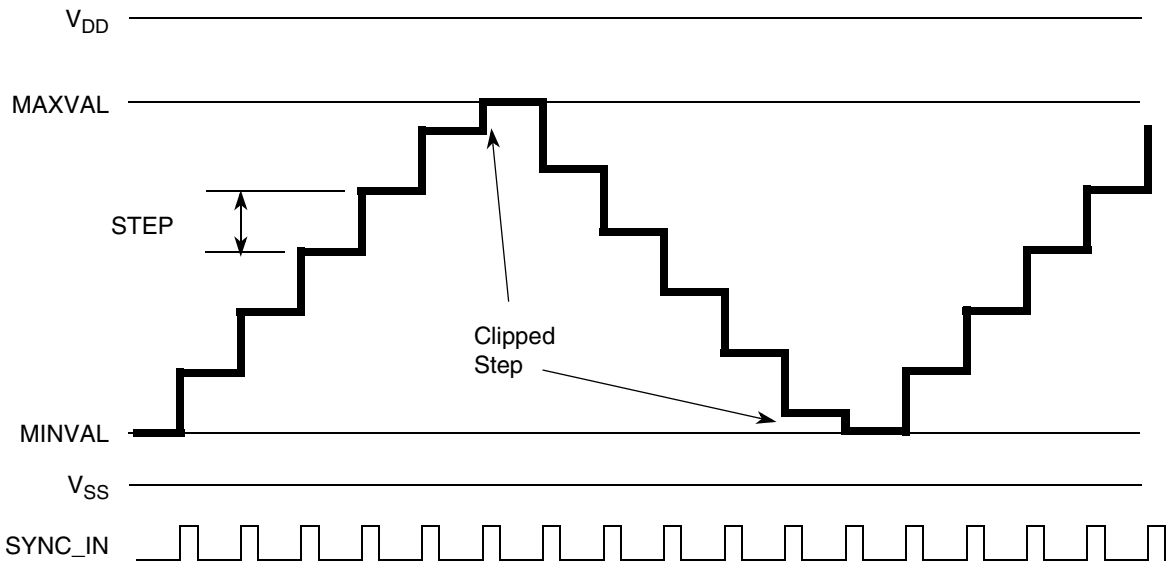


Figure 17-5. Triangle Waveform Example with Clipping

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

17.4.5 External Signal Descriptions

Table 17-1. External Signal Properties

Name	I/O Type	Function	Reset State
DAC_OUT	Output	Analog output	High impedance, pulled low
SYNC_IN	Input	Trigger an update	Driven low

17.4.5.1 Analog Output Pin (DAC_OUT)

Each DAC has a single current mode analog output pin. Digital words to be converted are 12 bits long with an LSB representing 0.806 mV. Analog output ranges from ~V_{SSA}+40 mV to ~V_{DDA}-40 mV (actual output range can be found in the chip data sheet) and can drive a 3 KΩ load. Routing of the DAC output to a chip pad or an internal comparator is a function of chip integration. Consult the chip system specification to determine routing control and options.

17.4.5.2 Trigger Input (SYNC_IN)

Trigger input controls when the buffered data is presented to the input of the DAC. This input must be high for at least one IPBus clock cycle; and must be low for at least one IPBus clock cycle. The update will occur on the rising edge. Typically, this input is connected to a timer output. This direct connection eliminates the jitter caused by variable latency in the servicing of the timer interrupt requests. The CPU must still update the buffered data (DATA) register prior to the next SYNC_IN rising edge, or the old buffered data will be reused.

The SIM module controls the muxing of various signals to the SYNC_IN input. The actual signals available varies for each chip and is detailed in the chip data sheet.

17.5 Register Description

The registers below are named using the prefix DAC. In the actual design, when multiple converters are used, they would be prefixed more specifically with DAC0 or DAC1. 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. The address offset is defined at the module level.

Table 17-2. DAC Memory Map

Device	Peripheral	Base Address
56F80xx	DAC0	$00F1C0
	DAC1	$00F1D0

Table 17-3 lists the DAC registers in ascending address order, including their acronyms and address offset of each register.

Table 17-3. DAC Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL	Control register	Read/Write	Section 17.5.1
Base + $1	DATA	Buffered data register	Read/Write	Section 17.5.2
Base + $2	STEP	Step size register	Read/Write	Section 17.5.3
Base + $3	MINVAL	Minimum value register	Read/Write	Section 17.5.4
Base + $4	MAXVAL	Maximum value register	Read/Write	Section 17.5.5

There are five registers in the DAC peripheral summarized in Figure 17-6. Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL	R	FILT_CNT			FILT_EN	0	0	0	0	0	0	UP	DOWN	AUTO	SYNC_EN	FORMAT	PDN
		W																
$1	DATA	R	0	0	0	0	DATA											
		W																
		R	DATA												0	0	0	0
		W																
$2	STEP	R	0	0	0	0	STEP											
		W																
		R	STEP												0	0	0	0
		W																
$3	MINVAL	R	0	0	0	0	MINVAL											
		W																
		R	MINVAL												0	0	0	0
		W																
$4	MAXVAL	R	0	0	0	0	MAXVAL											
		W																
		R	MAXVAL												0	0	0	0
		W																

R 0 Read as 0
W Reserved

Figure 17-6. DAC Register Map Summary

17.5.1 Control (CTRL) Register

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

Figure 17-7. Control (CTRL) Register

17.5.1.1 Glitch Filter Count (FILT_CNT)—Bits 15–13

This bit field represents the number of IPBus clock cycles the DAC output is held unchanged after new data is presented to the analog DACs inputs. Approximately 240 nsec is required for worst case settling of the DAC output; therefore, a value of seven should be used for 32 MHZ operation.

NOTE:

When using the glitch filter, be sure the filter count is less than the update count otherwise the DAC output will not be updated.

17.5.1.2 Glitch Filter Enable (FILT_EN)—Bit 12

This bit enables the glitch suppression filter. The glitch suppression filter introduces a latency equivalent to FILT_CNT IPBus clock cycles for DAC updates.

- 0 = Filter disabled
- 1 = Filter enabled

17.5.1.3 Reserved—Bits 11–6

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

17.5.1.4 Enable Up Counting (UP)—Bit 5

This bit enables counting up in automatic mode. Please see Section 17.4.2 for a description of how this affects automatic waveform generation.

- 0 = Up counting disabled
- 1 = Up counting enabled

17.5.1.5 Enable Down Counting (DOWN)—Bit 4

This bit enables counting down in automatic mode. Please see Section 17.4.2 for a description of how this affects automatic waveform generation.

- 0 = Down counting disabled
- 1 = Down counting enabled

17.5.1.6 Automatic Mode (AUTO)—Bit 3

This bit enables automatic waveform generation mode. In automatic mode an external source (typically a TMR or PIT) driving SYNC_IN determines the data update rate while the STEP, MINVAL, and MAXVAL registers and the UP and DOWN bits are used to shape the waveform. Please see Section 17.4.2 for a full description of automatic waveform generation. If SYNC_EN is not set when using this mode, then the data for the analog DAC will be updated every clock cycle, even though the maximum guaranteed update is 2 µsec.

- 0 = Normal mode, automatic waveform generation disabled
- 1 = Automatic waveform generation enabled

17.5.1.7 Synchronization Enable (SYNC_EN)—Bit 2

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 SYNC_EN bit is clear, then asynchronous mode is indicated and data written to the DATA reg will be presented to the analog DAC input in the following IPBus clock cycle.

- 0 = Asynchronous mode; data written to DATA register is presented to DAC inputs on next clock cycle
- 1 = Synchronous mode; rising edge of SYNC_IN updates data presented to DAC input

17.5.1.8 Data Format (FORMAT)—Bit 1

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 four unused bits are ignored.

- 0 = Data words are right justified (default)
- 1 = Data words are left justified

17.5.1.9 Power Down (PDN)—Bit 0

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 PDN the analog DAC will output the value currently presented to its inputs. The analog block requires 12 µsec to recover from the power down state before proper operation is guaranteed.

- 0 = DAC is operational
- 1 = DAC is powered down (default)

17.5.2 Buffered Data (DATA) Register

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

Figure 17-8. Buffered Data (DATA) Register with FORMAT = 0

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

Figure 17-9. Buffered Data (DATA) Register with FORMAT = 1

17.5.2.1 Reserved—Bits 15–12 or 3–0

These bit fields are reserved or not implemented. They are read as zero and cannot be modified by writing.

17.5.2.2 Digital Analog Converter Data (DATA)—Bits 11–0 or 15–4

When the DAC is in operational mode (PDN = 0), the digital data contained in this register is presented to the analog DAC upon the rising edge of the SYNC_IN signal (or at the next clock cycle if SYNC_EN is clear) and converted to analog and output by the DAC. The data in this register can be updated at any rate, but the DAC is only guaranteed to operate at a maximum update rate of 0.5 MHz.

17.5.3 Step Size (STEP) Register

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

Figure 17-10. Step Size (STEP) Register with FORMAT = 0

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

Figure 17-11. Step Size (STEP) Register with FORMAT = 1

17.5.3.1 Reserved—Bits 15–12 or 3–0

These bit fields are reserved or not implemented. They are read as zero and cannot be modified by writing.

17.5.3.2 Step Size (STEP)—Bits 11–0 or 15–4

When the DAC is in automatic mode (AUTO = 1), the step size contained in this register is added to, or subtracted from, the current value creating the next value presented to the DAC inputs. This register is not used during normal mode operation.

17.5.4 Minimum Value (MINVAL) Register

Base + $3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	MINVAL											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-12. Minimum Value (MINVAL) Register with FORMAT = 0

Base + $3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	MINVAL												0	0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-13. Minimum Value (MINVAL) Register with FORMAT = 1

17.5.4.1 Reserved—Bits 15–12 or 3–0

These bit fields are reserved or not implemented. They are read as zero and cannot be modified by writing.

17.5.4.2 Minimum Value (MINVAL)—Bits 11–0 or 15–4

When the DAC is in automatic mode (AUTO = 1), the minimum value contained in this register acts as the lower range limit during automatic waveform generation. Please see Section 17.4.2 for a description of how this value is used in automatic waveform generation. This register is not used during normal mode operation. Please refer to the chip data sheet for limitations on the low end voltage output of the DAC.

17.5.5 Maximum Value (MAXVAL) Register

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

Figure 17-14. Maximum Value (MAXVAL) Register with FORMAT = 0

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

Figure 17-15. Maximum Value (MAXVAL) Register with FORMAT = 1

17.5.5.1 Reserved—Bits 15–12 or 3–0

These bit fields are reserved or not implemented. They are read as one and cannot be modified by writing.

17.5.5.2 Maximum Value (MAXVAL)—Bits 11–0 or 15–4

When the DAC is in automatic mode (AUTO = 1), the maximum value contained in this register acts as the upper range limit during automatic waveform generation. Please see Section 17.4.2 for a description of how this value is used in automatic waveform generation. This register is not used during normal mode operation. Please refer to the chip data sheet for limitations on the high end voltage output of the DAC.

17.6 Clocks

The DAC has a single clock input.

Clock Input	Source	Characteristics
IP Clock	SIM	Maximum Rate is 32MHz

17.7 Interrupts

The DAC module does not generate interrupts.

17.8 Resets

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

Reset	Source	Characteristics
$\overline{\text{RST}}$	SIM	

Table 17-4. Document Revision History for Chapter 17

Version History	Description of Change
Rev. 3	Added revision history table.

Chapter 18
Comparator (CMP)

18.1 Introduction

The comparator (CMP) includes a digital control interface and an analog comparator module. The analog comparator also includes a five-input analog switching matrix with the continuous-time differential-input analog comparator function. The five analog inputs include three GPIO (referred to as CIN1, CIN2, and CIN3), a DAC input, and an import input from another comparator module. The switching matrix supports the independent connection of the analog inputs to the + and – input of the analog comparator and to the comparator's export output. A bidirectional interface between two comparators is formed by connecting the export output of each comparator to the import input of the other. This provides each comparator the option to perform an analog compare using the analog inputs of the other comparator.

18.2 Features

CMP characteristics include:

- Continuous-time differential-input analog comparator
- Internal switching matrix supports the independent connection of the analog inputs to the + and – input of the analog comparator and to the comparator's export output
- Input sources include five analog inputs include three GPIO (referred to as CIN1, CIN2, and CIN3), a DAC output, and an import input from another comparator module.
- Analog comparator polarity control (optional inversion)
- Programmable low-pass filter
- Power saving mode
- Falling and rising comparator edge detection with compare interrupt on rising and/or falling comparator edge
- Output can be used to control timer inputs, PWM faults, PWM control, external pin output, or as a interrupt source

18.3 Block Diagram

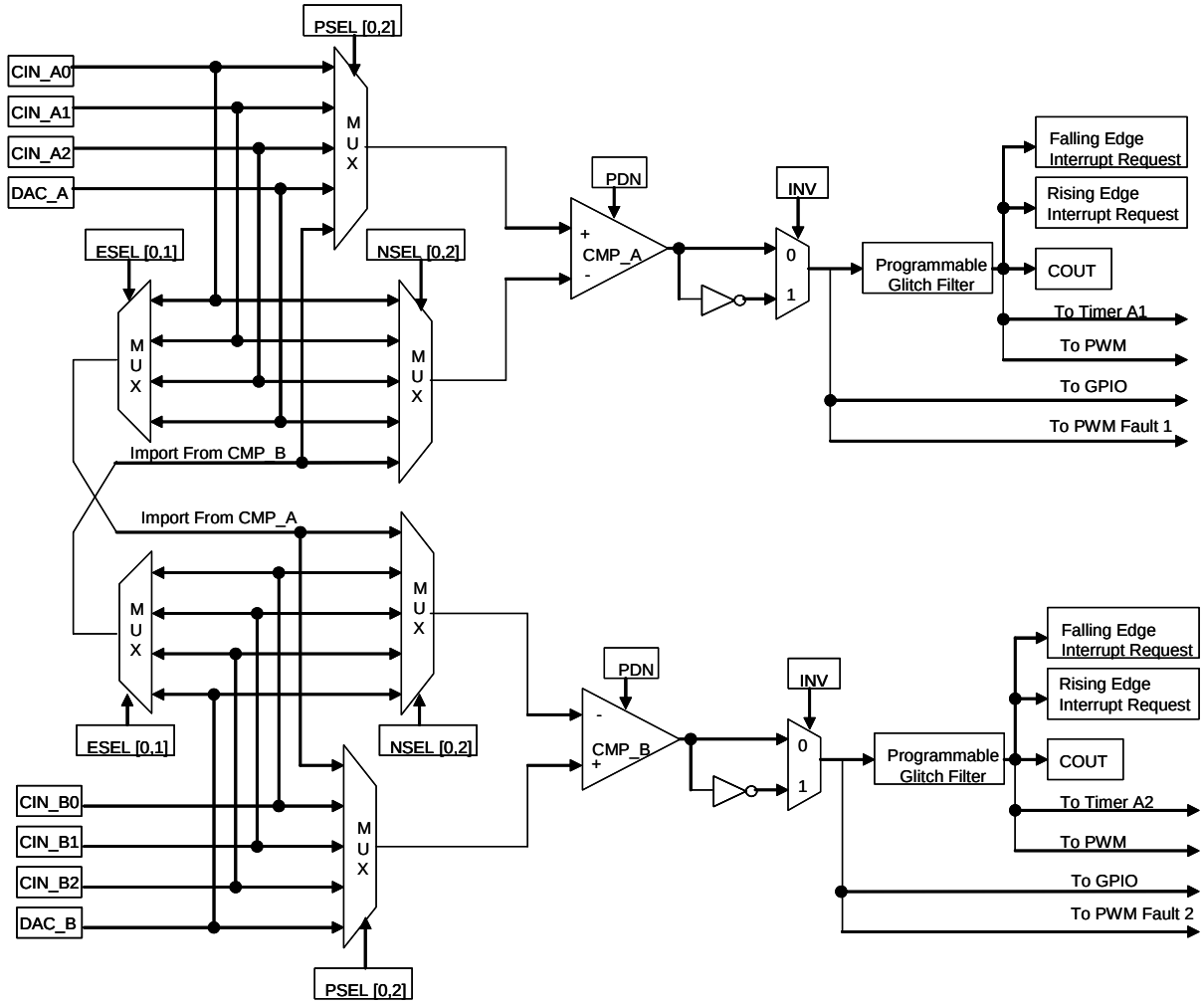


Figure 18-1. CMP/Comparator Integration Block Diagram

18.4 Functional Description

This section provides additional details of the comparator functionality and operation.

18.4.1 Uses of Comparator Output

The output of the differential comparator goes through an optional inversion stage to generate comparator output A (COUTA), the asynchronous comparator output. COUTA is then synchronized to the peripheral bus clock and passed through an optional low-pass digital filter to generate comparator output (COUT), the synchronized comparator output. COUT and COUTA comparator output pad signal because it is necessary for the implementation of an external hysteresis circuit the comparator output propagates through combinational logic only on the way to the pad. COUTA is also used to feed a PWM fault input signal since PWM faults must also propagate through combinational logic only so they can reach the PWM during a system failure.

COUT is used to feed a TMR or to feed PWM_GPIO inputs (these are the PWM input channels) because these potentially benefit from synchronization and noise filtering provided by the COUT output.

The distribution of COUT and COUTA signals within a part is part-specific and is addressed in the system specification.

18.4.2 Handling of Ambiguous Control Inputs

If either positive comparator input source select (PSEL) or negative comparator input source select (NSEL) is set to a reserved value or the two bit fields PSEL and NSEL are equal, then the NOSTATE signal is asserted high. This causes selection of V_{SS_A} as the positive input to the differential comparator and V_{DD_A} as the negative input to the differential comparator. This, in turn forces the output of the differential comparator (CMP) to remain zero whenever invalid/ambiguous source selections are made.

18.4.3 Hysteresis

The following diagram illustrates implementation of an external hysteresis resistor bridge between the asynchronous comparator output and the positive (+) input of the comparator. Since positive feedback is required, invert control (INV) bit 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.

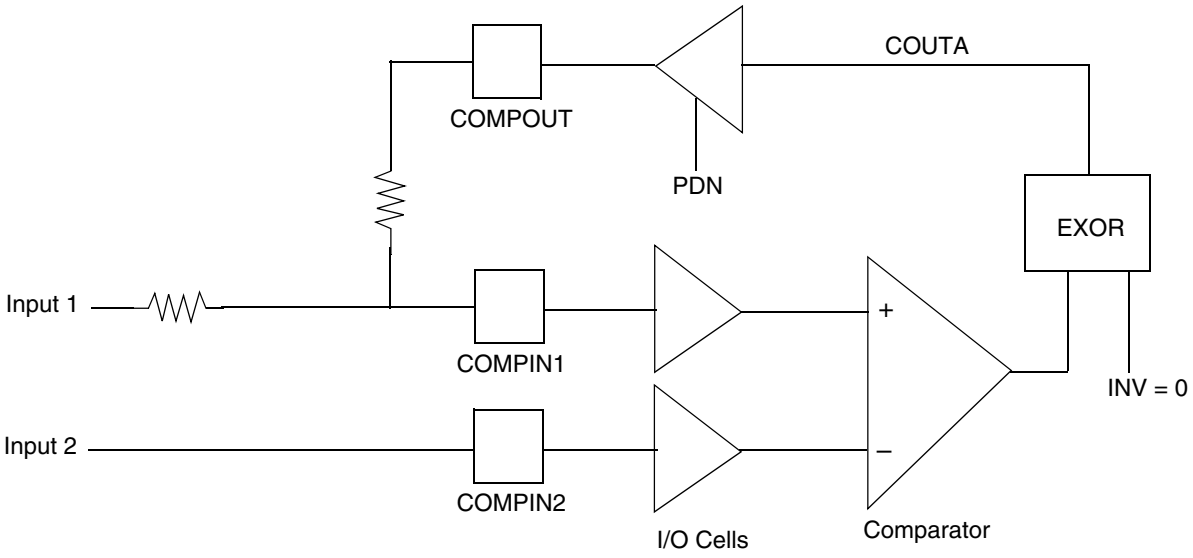


Figure 18-2. External Hysteresis Resistor Bridge

The differential comparator does not have internal hysteresis for two reasons. First, hysteresis can be controlled more accurately using an external resistor bridge. Secondly, any chip-to-chip variation in the internal hysteresis would degrade the apparent accuracy of the DAC when used as an input to 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 COMPOUT.

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

18.4.4 Power Down

Asserting the power down (PDN) control eliminates bias current in the comparator, the differential comparator output CMP is driven low while PDN is set to one.

The PDN output of the CMP module is also used to tristate the COMPOUT external I/O driver while the comparator is in power down mode and the I/O cell is configured for COMPOUT functionality. This will avoid forcing an unnecessary current through an external hysteresis resistor bridge, if one is present.

Though not required, it is recommended to set PDN = 1 while the comparator is not in use. This saves power and reduces noise.

18.4.5 Startup and Operation

A typical startup sequence follows:

- If intending to compare using the DAC output or the IMPORT signal provided from another comparator, configure those modules appropriately.
- Set FILT_PER, FILT_CNT, INV, ESEL, PSEL, and NSEL to desired values and set PDN = 0 to power up the comparator.
- If appropriate for application, wait for the delay required to power up the used analog modules and stabilize COUT and then clear any startup related edge detections by writing 1 to both RC and FC.

The time required to stabilize COUT is the largest power-on delay of the components (DACs and/or comparators) involved plus the largest propagation delay from a selected analog source through the differential comparator and through the filter. Power-on and propagation delay of the comparators and DACs are available from data sheets. Filter delay is specified in the next section.

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 RC/FC status bits to reflect an input change or a configuration change to one of the components involved in the data path.

18.4.6 Low Pass Filter

The low pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT are provided as module outputs and are used for different purposes within the part as described in Section 18.4.1.

The filter always performs synchronization and edge detection. Filtering is enabled by setting FILT_PER to a non-zero value. When enabled, the filter takes samples of COUTA every FILT_PER system cycles. The output state of the filter changes when FILT_CNT + 3 consecutive samples all agree the output value changed. Setting FILT_PER to 0 disables the filter and eliminates switching current associated with the filtering process.

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.

The FILT_PER value should be set so the sampling period is just larger than the period of the expected noise. This way a noise spike only corrupts one sample. The FILT_CNT 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 FILT_CNT + 3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. When the filter is disabled, the latency between COUTA and COUT is two IPBus clock periods. In the absence of noise, the additional latency of the filter does not exceed ((FILT_CNT + 3) x FILT_PER) + 2 IPBus clock periods. However, this latency is restarted each time an actual output transition is masked by noise. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the FILT_CNT + 3 power.

18.5 Register Description

The registers below are associated with a single instance of the CMP module and are named using the prefix CMP. A one-letter suffix is used such as CMPA or CMPB, to distinguish between multiple CMP modules integrated on the same chip.

Table 18-1. CMP Memory Map

Device	Peripheral	Base Address
56F80xx	CMPA	$00F1E0
	CMPB	$00F1F0

Table 18-2 lists the CMP registers in ascending address order, including their acronyms and address offset of each register.

Table 18-2. CMP Register Summary

Address Offset	Acronym	Register Name	Access Type	Location
Base + $0	CTRL	Control register	Read/Write	Section 18.5.2
Base + $1	STAT	Status register	Read/Write	Section 18.5.3
Base + $2	FILT	Filter register	Read/Write	Section 18.5.3

There are three registers in the CMP peripheral summarized in Figure 18-3. Each is detailed in the following sections.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL	R	RCIE	FCIE	0	0	ESEL		0	NSEL			0	PSEL			INV	PDN
		W																
$1	STAT	R	RC	FC	0	0	0	0	0	0	0	0	0	0	0	0	0	COUT
		W																
$2	FILT	R	0	0	0	0	0	FILT_CNT			FILT_PER							
		W																

R 0 Read as 0
W Reserved

Figure 18-3. CMP Register Map Summary

18.5.1 Control (CTRL) Register

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RCIE	FCIE	0	0	ESEL		0	NSEL			0	PSEL			INV	PDN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 18-4. Control (CTRL) Register

18.5.1.1 Rising Compare Interrupt Enable (RCIE)—Bit 15

This bit enables assertion of a compare interrupt when the rising compare (RC) status bit is high.

- 0 = RC has no effect on the compare interrupt (default)
- 1 = Compare interrupt asserts while RC is high

18.5.1.2 Falling Compare Interrupt Enable (FCIE)—Bit 14

This bit enables assertion of a compare interrupt when the falling compare (FC) status bit is high.

- 0 = FC has no affect on the compare interrupt (default)
- 1 = Compare interrupt will assert while FC is high

18.5.1.3 Reserved—Bits 13–12

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

18.5.1.4 Export Source Select (ESEL)—Bits 11–10

This bit selects analog input source connected to export output.

- 0 = CIN1 input (default)
- 1 = CIN2 input
- 2 = CIN3 input
- 3 = DAC input

18.5.1.5 Reserved—Bit 9

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

18.5.1.6 Negative Comparator Input Source Select (NSEL)—Bits 8–6

This bit field selects the analog input source connected to the negative (–) input of the differential comparator. Setting NSEL to a reserved value connects the negative input of the differential comparator to V_{SS_A}. Setting PSEL and NSEL to the same value treats both as if set to reserved values.

- 0 = CIN1 input (default)
- 1 = CIN2 input
- 2 = CIN3 input
- 3 = DAC input
- 4 = IMPORT input
- 5-7 = Reserved (V_{SS_A})

18.5.1.7 Reserved—Bit 5

This bit is reserved or not implemented. It is read as zero and cannot be modified by writing.

18.5.1.8 Positive Comparator Input Source Select (PSEL)—Bits 4–2

This bit selects the analog input source connected to the positive (+) input of the differential comparator. Setting PSEL to a reserved value connects the positive input of the differential comparator to V_{DD_A}. Setting PSEL and NSEL to the same value treats both as if set to reserved values.

- 0 = CIN1 input (default)
- 1 = CIN2 input
- 2 = CIN3 input
- 3 = DAC input
- 4 = IMPORT input
- 5–7 = Reserved (V_{DD_A})

18.5.1.9 Invert Control (INV)—Bit 1

Enables optional inversion of the differential comparator output CMP. By default (not inverted) the comparator outputs one when the positive (+) input voltage exceeds the negative (–) input voltage.

- 0 = Comparator output is not inverted (default)
- 1 = Comparator output is inverted

18.5.1.10 Power Down (PDN)—Bit 0

This bit is used to power down the differential comparator when it is not in use. The power down recovery time is provided under electrical specifications. This is the time required to recover from the power down state before proper operation is guaranteed. The output of the differential comparator CMP is zero while PDN is one.

- 0 = Differential comparator is operational
- 1 = Differential comparator is powered down and output CMP is set to zero (default)

18.5.2 Status (STAT) Register

Base + $1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RC	FC	0	0	0	0	0	0	0	0	0	0	0	0	0	COUT
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-5. Status (STAT) Register

18.5.2.1 Rising Comparator Output Edge Indicator (RC)—Bit 15

This bit asserts high upon the detection of a rising edge on COUT. COUT is the output of the differential comparator after optional inversion, synchronization, and optional low-pass filter. Writing one clears the RC bit while writing zero to the RC bit is ignored.

- 0 = Rising edge not detected on COUT since last time RC was cleared (default)
- 1 = Rising edge detected on COUT

18.5.2.2 Falling Comparator Output Edge Indicator (FC)—Bit 14

This bit asserts high upon the detection of a falling edge on COUT. COUT is the output of the differential comparator after optional inversion, synchronization, and optional low-pass filter. Writing one clears the FC bit while writing zero to the FC bit is ignored.

- 0 = Falling edge not detected on COUT since last time FC was cleared (default)
- 1 = Falling edge detected on COUT

18.5.2.3 Reserved—Bits 13–1

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

18.5.2.4 Synchronized and Filtered Compare Output (COUT)—Bit 0

This bit provides read access to the COUT signal. COUT is the output of the differential comparator after optional inversion, synchronization, and optional low-pass filter.

18.5.3 Filter (FILT) Register

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

Figure 18-6. Filter (FILT) Register

18.5.3.1 Reserved—Bits 15–11

This bit field is reserved or not implemented. It is read as zero and cannot be modified by writing.

18.5.3.2 Filter Sample Count (FILT_CNT)—Bits 10–8

This bit field represents the number of consecutive samples requiring agreement prior to the comparator output filter accepting a new output state. A value of 0x0 represents three samples. A value of 0x7 represents ten samples. Filter programming and latency details are described in Section 18.4.6.

18.5.3.3 Filter Sample Period (FILT_PER)—Bits 7–0

This bit field specifies the sampling period, in peripheral clock cycles, of the comparator output filter. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details are described in Section 18.4.6.

18.6 Clocks

The CMP has a single clock input.

Table 18-3. Clock Summary

Clock Input	Source	Characteristics
clk_ipb	SIM	Standard peripheral clock

18.7 Interrupts

Interrupts generated by the CMP module and their relationship to module status indicators are shown in the following table. Please see Table 18-4.

Table 18-4. Interrupt Summary

Core Interrupt	Status (Polling) Bit	Description
Compare Interrupt	RC	Compare interrupt asserts while RC is 1 and RCIE is 1
	FC	Compare interrupt asserts while FC is 1 and FCIE is 1

18.7.1 Description of Interrupt Operation

18.7.1.1 Compare Interrupt

The compare interrupt can be asserted upon the detection of a rising edge, a falling edge, or any edge on COUT. COUT is the output of the differential comparator after optional inversion, synchronization, and low-pass filter. The edge sensitivity of interrupt generation is determined by whether RCIE, FCIE, or both are set to one.

More specifically, the compare interrupt output $\overline{\text{CINT}}$ will assert low when RCIE and RC are both high, or when both FCIE and FC are both high. The RC status bit asserts high upon the detection of a rising edge at COUT and remains high until RC is written to one. Similarly the FC status bit asserts high upon the occurrence of a falling edge at COUT and remains high until FC is written to one. RCIE and FCIE affect only the assertion of the interrupt signal and have no affect on the functionality of the RC and FC status bits.

The compare interrupt functions in the 56800E stop mode if and only if the option is provided to keep the peripheral clock to the CMP module operating in stop mode. This feature, if available, is implemented in the SIM module.

18.8 Resets

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

Table 18-5. Reset Summary

Reset	Source	Characteristics
$\overline{\text{rst\_periph}}$	SIM	Standard peripheral reset

Table 18-6. Document Revision History for Chapter 18

Version History	Description of Change
Rev. 3	Added revision history table.

Appendix A
Definitions, Acronyms, and Abbreviations

ACIM	A/C induction motors
Active State	A scan is in process using either or both converters
ACK	Acknowledge of CAN message
A/**D**	Analog-to-digital
ADC	Analog-to-digital converter
ADC Clock	The ADC blocks run at a slower speed than the rest of the system. A separate ADC clock is derived from the system IPBus clock based upon the divisor specified in the ADC control 2 register (CTRL2).
Array Write	A write from 56800E to the FLASH memory array space. Array writes provide data via the write data bus and address via the program address bus. The address and data are registered as part of the command sequence prior to accessing the FLASH arrays. BDCbrush DC motor
BLDC	Brushless DC motor
BOTNEG	Bottom-side PWM polarity bit
BSDL	Boundary scan description language
BSEL	MSCAN peripheral transmit buffer selection register
BSR	JTAG peripheral boundary scan register
BTR0	MSCAN peripheral bus timing 0 register
BTR1	MSCAN peripheral bus timing 1 register
Bus Clock	CPU bus related clock
CAN	Controller area network
CAN Bus	CAN protocol related serial bus
CAN Clock	CAN protocol related clock
CAL	ADC peripheral calibration register

CAPT	TMR peripheral capture register
CCTRL	PWM peripheral channel control register
Channel	An ADC module has 15 analog inputs and contains two channels, or ADC converters. Inputs ANA0-7 are processed by converter A only and inputs ANB0-7 are processed by converters B only.
CLKCHK	OCCS peripheral external clock check register
CLIST1	ADC peripheral channel list 1 register
CLIST2	ADC peripheral channel list 2 register
CLIST3	ADC peripheral channel list 3 register
CLIST4	ADC peripheral channel list 4 register
CLRACT	I^2C peripheral clear activity interrupt register
CLRGC	I^2C peripheral clear general call register
CLRINT	I^2C peripheral clear interrupt register
CLRRDREQ	I^2C peripheral clear read request register
CLRRXOVR	I^2C peripheral clear receive over interrupt register
CLRRXUND	I^2C peripheral clear receive under interrupt register
CLRSTDET	I^2C peripheral clear start detect interrupt register
CLRSTPDET	I^2C peripheral clear stop detect interrupt register
CLRTXABRT	I^2C peripheral clear transmit abort interrupt register
CLRTXDONE	I^2C peripheral clear transmit done interrupt register
CLRTXOVR	I^2C peripheral clear transmit over interrupt register
CID	JTAG peripheral chip identification register
CLKO	Clock Output terminal
CLKDIV	FM peripheral clock divider register
CMD	FM peripheral Command register
CMOS	Complementary metal oxide semiconductor, (a form of digital logic) characterized by low power consumption, wide power supply range, and high noise immunity.)
CMOD	PWM peripheral counter modulo register
CMPLD1	TMR peripheral comparator load 1 register
CMPLD2	TMR peripheral comparator load 2 register
Codec	Coder/decoder
Command	A three-step digital signal controller (DSC) instruction sequence to program or erase sequence the FLASH.

Converter	Refers to one of the two analog-to-digital converters within the ADC module.
CNFG	FM and PWM peripherals configuration registers
CNTR	COP, PIT, PWM, and TMR peripherals counter registers
Command Sequence	A three-step controller instruction sequence to program or erase the FLASH
COP	Computer operating properly
COMP1	TMR peripheral compare 1 register
COMP2	TMR peripheral compare 2 register
CPHA	Clock phase
CPOL	Clock polarity
CPU Bus	CPU related read/write data bus
CRC	Cyclic redundancy code
CSCTRL	TMR peripheral comparator status and control register
CTRL	CMP, COP, DAC, I^2C, OCCS, PS, PIT, PWM, and TMR peripherals control registers
CTRL0	MSCAN peripheral control 0 register
CTRL1	ADC, MSCAN, and QSCI peripherals control 1 registers
CTRL2	ADC and QSCI peripherals control 2 register
DAC	Digital-to-analog converter peripheral
DATA	DAC, I^2C, FM, GPIO, and QSCI peripherals data registers
DB	OCCS peripheral divide-by register
DDIR	GPIO peripheral data direction register
DELAY	QSPI peripheral word delay register
Differential	Differential samples convert the differential voltage on a pair of inputs.
Differential Pair	Differential measurement is limited to the input pairs such as ANA0-1, ANA2-3, ANB0-1, and ANB2-3. When configured for differential measurement, the pair is referred to be the number of the even input. For example, to obtain a differential measurement on pair ANA0-1, enable differential measurement on the ANA0-1 pair, and specify a sample on input ANA0 or ANA1.
DIVBY	OCCS peripheral divide-by register
DMAP1-2	PWM peripheral disable mapping 1-2 registers
DRCV	QSPI peripheral data receive register
DRIVE	GPIO peripheral drive strength control register
DSCTRL	QSPI peripheral data size and control register
DTIM0-1	PWM peripheral deadtime 0-1 registers

DXMIT	QSPI peripheral data transmit register
ENBL	TMR peripheral channel enable and I^2C enable registers
EOF	MSCAN end of frame
EOnCE	Enhanced on-chip emulation (unit)
ESR	Equivalent series resistance
EXTBOOT	External boot
FAULT	Fault input to PWM peripheral
FCTRL	PWM peripheral fault control register
FFLIT	PWM peripheral fault filter 0-3 registers
FIFO	QSPI peripheral FIFO control register, MSCAN first-in-first-out memory, and I^2C first-in-first-out memory
FILT	TMR peripheral input filter and CMP peripheral filter registers
FLASH Erase Page	Eight rows of either 128 or 64 bytes (1k or 512 bytes) depending on the block chosen. This will be the result in either 1k or 512 bytes for a block being erased due to a page erase command.
FLASH Module	Includes bus interface, command control, and the FLASH physical block
FLASH Physical Block	Four different hard blocks from TSMC are available though not all have been used as Block a part of a valid chip configuration. 1. 128K bytes organized as 64K by16-bit words constructed with a 128 byte wide row 2. 64K bytes organized as 32K by 16-bit words constructed with a 64 byte wide row 3. 32K bytes organized as 16K by 16-bit works constructed with a 64 byte wide row 4. 16K bytes organized as 8K by 16-bit words constructed with a 64 byte wide row Each block includes high voltage generation and parametric test features.
FLASH User Mode	FLASH module operations defined for user mode
FLTACK	PWM peripheral fault status/acknowledge register
FOSC	Oscillator frequency
FREF	Reference frequency
FSHCNT	I^2C peripheral fast speed clock high count register
FSLCNT	I^2C peripheral fast speed clock low count register
FSTAT	PWM peripheral fault control register
GPIO	General purpose input/output
Harvard Architecture	This is a microprocessor architecture using separate buses for program and data. This architecture is typically used on digital signal controller (DSC) to optimize the

	data throughput.
HILIM0-7	ADC peripheral high limit 0-7 registers
HOLD	TMR peripheral hold register
I^{2}C	Inter-integrated circuit interface
I^{2}C Arbitration	This predefined procedure authorizes only one master at a time to take control of the bus.
I^{2}C Master	This is the device initializing a transfer (START command), generates the clock (SCL) signal, and terminates the transfer (STOP command). A master can be either a transmitter or a receiver.
I^{2}C Multi-Master	This is the ability for more than one master to co-exist on the bus at the same time without collision or data loss.
I^{2}C Receiver	This device receives data from the bus. A receiver can either be a device receiving data on its own request (a master receiver), or in response to a request from the master (a slave master).
I^{2}C Slave	This device is addressed by the master. A slave can be either receiver or transmitter.
I^{2}C Synchronization	This predefined procedure synchronizes the clock signal provided by two or more masters.
I^{2}C Transmitter	This device sends data to the bus. A transmitter can either be a device initiating data transmission to the bus (a master transmitter), or in response to a request from the master to send data to the bus (a slave transmitter).
IASSRT	GPIO peripheral interrupt assert register
ic_clk	Front end clock of the I^{2}C peripheral
ICCTRL	PWM peripheral Internal correction control register
IDAC	MSCAN peripheral identifier acceptance control register
IDAR0-7	MSCAN peripheral identifier acceptance 0-7 registers
Idle State	A scan is not in process using either converter
IDMR0-7	MSCAN peripheral identifier mask 0-7 registers
IEDGE	GPIO peripheral interrupt edge sensitive register
IENBL	I^{2}C peripheral interrupt enable register
IEN	GPIO peripheral interrupt enable register
IEPOL	GPIO peripheral interrupted polarity register
IFS	MSCAN inter-frame sequence
IPEND	GPIO peripheral interrupt pending register
IPOL	GPIO peripheral interrupt polarity register

I/O	Input/output
IPBus	Intellectual properties bus. This Freescale proprietary bus architecture accommodates various intellectual properties operating at slower speed than the system bus.
IPBus Clock	System peripheral clock
ICCTRL	PWM peripheral internal correction control register
IRQ	Interrupt request
ISTAT	I^2C peripheral interrupt status register
ITCN	Interrupt controller (see device data sheet for complete information)
JTAG	Joint test action group
LIMSTAT	ADC peripheral limit status register
LOAD	TMR peripheral load register
LOLIM0-7	ADC peripheral low limit 0-7 registers
LSB	Least significant bit
LSH_ID	Least significant half of JTAG_ID
LVI	Low voltage interrupt
MAXVAL	DAC peripheral maximum value register
MEMSEL	FM peripheral memory select register
MHz	Megahertz
MINVAL	DAC peripheral minimum value register
MISO	Master in/slave out
MOD	PIT peripheral Modulo register
MOSI	Master out/slave in
MSB	Most significant bit
MSH_ID	Most significant half of JTAG ID
MISC	MSCAN peripheral miscellaneous register
MISR	A multiple input signature analyzer register is an output response analyzer implemented using a linear feedback shift register.
MUX	Multiplexer
Multiple Input Analyzer	An output response analyzer implemented using a linear feedback shift register; signature (MISR).
Non-Simultan-Scan	The converter A and B scans occur in parallel but asynchronously. The A and B Parallelscans may be of different lengths.
OCCS	On-chip clock synthesis peripheral

OCTRL	OCCS peripheral oscillator control register
OFFST0-7	ADC peripheral offset 0-7 registers
OnCE	On-chip emulation (unit)
OPT1	FM peripheral optional data 1 register
Oscillator Clock	Direct clock from external oscillator
OUT	PWM peripheral output control register
PAB	Program address bus
Parallel Scan	A type of scan consisting of at most eight converter A samples and eight converter B samples. The converter A and B scans are performed in parallel.
pclk	Register interface clock of the I^2C peripheral
PEREN	GPIO peripheral enable register
PFLASH	FM peripheral program FLASH
PLL	Phase locked loop
POL	Polarity
POR	Power-on reset
PORT	PWM peripheral port register
PPOUTM	GPIO peripheral push-pull output mode control register
PRAM	Program RAM
PROT	FM and OCCS peripherals protection registers
PUPEN	GPIO peripheral pullup enable register
PWM	Pulse width modulator
PWR	ADC peripheral power control register
QSCI	Queued serial communications interface
QSPI	Queued serial peripheral interface
RDATA	GPIO peripheral raw data register
RAM	Random access memory
RATE	QSCI peripheral baud rate register
RDY	ADC peripheral conversion ready register
RFLG	MSCAN peripheral receiver flag register
RIER	MSCAN peripheral receiver interrupt enable register
RISTAT	I^2C peripheral raw interrupt status register
RSLT0-7	ADC peripheral result 0-7 registers

RSLT8-15	ADC peripheral result 8-15 registers
ROM	Read only memory
RXERR	MSCAN receiver error counter register
RXFG	MSCAN peripheral foreground receive buffer registers
RXFLR	I^2C peripheral receive FIFO level register
RXFT	I^2C peripheral receive FIFO threshold register
Sample	Refers to a single analog-to-digital conversion
SAR	I^2C peripheral slave address register
Scan	Performing a scan refers to performing a series of analog-to-digital conversions on an ordered list of samples. Define which input is sampled at each step. Each sample may be single ended or differential. The same input may be sampled at more than one step.
SCL	I^2C clock signal line (serial clock)
SCTRL	QSPI status and control, PWM source Control, and TMR status and control peripheral registers
SDA	I^2C data signal line (serial data)
SDIS	ADC peripheral sample disable register
SECHI	FM peripheral security high half register
SECLO	FM peripheral security low half and optional security low half register
Sequential Scan	A type of scan consisting of at most 16 samples where the samples are taken on at a time. Each sample may refer to any of the 16 analog inputs.
S/H	Sample and hold
SHUTDN	OCCS peripheral shutdown register
SIM	System integration module
SIM_PWR	VREG peripheral SIM power control register
Simultaneous Parallel Scan	Converters A and B scans occur both in parallel and simultaneously. The A and B scans are always of the same length and measurements occur as simultaneous pairs, one from converter A and one from converter B.
Single Ended	Single ended samples convert a single voltage on a user-specified input
SOF	Start of frame
SOC	System-on-chip
SRM	Switched reluctance motor
SSHCNT	I^2C peripheral standard speed clock high count register
SSLCNT	I^2C peripheral standard speed clock low count register

STAT	ADC, CMP, I^2C, OCCS, PS, and QSCI peripherals status registers
STEP	DAC peripheral step size register
SYNC	PWM peripheral synchronization window register
System Clock	A free running clock generated by the PLL and oscillators. In reality, each system component will receive a unique version of this clock generated by the clock generation module and controlled by the system integration module (SIM).
TAAK	MSCAN peripheral transmitter message abort acknowledge register
TAP	Test access port
TAR	I^2C peripheral target address register
TARQ	MSCAN peripheral transmitter message abort request register
TCK	TAP clock
TDI	TAP data in
TDO	TAP data out
TFLG	MSCAN peripheral transmitter flag register
TIER	MSCAN peripheral transmitter interrupt enable register
TLM	TAP linking module
TOUT	COP peripheral timeout register
TSMC IP	Taiwan semiconductor manufacturing company intellectual property. In the FLASH module, the TSMC IPs are the 128k, 64k, 16k, and 8k byte FLASH hard blocks.
TSTSIG	FM peripheral test array signature register
TXABRTSRC	I^2C peripheral transmit abort source register
TXERR	MSCAN peripheral transmit error counter register
TXFG	MSCAN peripheral foreground transmit buffer registers
TXFLR	I^2C peripheral transmit FIFO level register
TXFT	I^2C peripheral transmit FIFO threshold register
USTAT	FM peripheral user status register
VAL0-5	PWM peripheral value 0-5 registers
VCO	Voltage controlled oscillator
V_{DD}	Power
V_{DDA}	Analog power
V_{REF}	Voltage reference
VRM	Variable reluctance motor

V_{SS}	Ground
V_{SSA}	Analog ground
XRAM	Data RAM
ZXCTRL	ADC peripheral zero crossing control register

Appendix B
Programmer Sheets

B.1 Introduction

The following pages provide a set of reference tables and programming sheets intended to simplify programming the 56F8000 Family. The programming sheets provide room to add the value of each bit and the hexadecimal value for each register. These pages may be photocopied.

For complete instruction set details, refer to Chapter 4 of the *DSP56800 Reference Manual*.

B.2 Legacy and New Acronym Cross Reference

Register acronyms are revised from previous device peripheral manuals to provide a cleaner register description. A cross reference to legacy and revised acronyms are provided in Table B-1.

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 1 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Analog-to-Digital Converter (ADC) Module							
Control 1 Register	CTRL1	ADCR1	ADC_CTRL1	ADC_ADCR1	ADC_ADCR1	0xF080	
Control 2 Register	CTRL2	ADCR2	ADC_CTRL2	ADC_ADCR2	ADC_ADCR2	0xF081	
Zero Crossing Control Register	ZXCTRL	ADZCC	ADC_ZXCTRL	ADC_ADZCC	ADC_ADZCC	0xF082	
Channel List 1 Register	CLIST1	ADLST1	ADC_CLIST1	ADC_ADLST1	ADC_ADLST1	0xF083	
Channel List 2 Register	CLIST2	ADLST2	ADC_CLIST2	ADC_ADLST2	ADC_ADLST2	0xF084	
Channel List 3 Register	CLIST3		ADC_CLIST3	ADC_ADCLST3	ADC_ADCLST3	0xF085	
Channel List 4 Register	CLIST4		ADC_CLIST4	ADC_ADCLST4	ADC_ADCLST4	0xF086	
Sample Disable Register	SDIS	ADSDIS	ADC_SDIS	ADC_ADSDIS	ADC_ADSDIS	0xF087	
Status Register	STAT	ADSTAT	ADC_STAT	ADC_ADSTAT	ADC_ADSTAT	0xF088	
Conversion Ready Register	RDY		ADC_CNRDY	ADC_ADCNRDY	ADC_ADCNRDY	0xF089	
Limit Status Register	LIMSTAT	ADLSTAT	ADC_LIMSTAT	ADC_ADLSTAT	ADC_ADLSTAT	0xF08A	
Zero Crossing Status Register	ZXSTAT	ADZCSTAT	ADC_ZXSTAT	ADC_ADZCSTAT	ADC_ADZCSTAT	0xF08B	
Result 0-7 Registers	RSLT0-7	ADRSLT0-7	ADC_RSLT0-7	ADC_ADRSLT0-7	ADC_ADRSLT0-7	0xF08C	0XF093
Result 8-15 Registers	RSLT8-15		ADC_RSLT8-15	ADC_ADRSLT8-15	ADC_ADRSLT8-15	0xF094	0XF09B
Low Limit 0-7 Registers	LOLIM0-7	ADLLMT0-7	ADC_LOLIM0-7	ADC_ADLLMT0-7	ADC_ADLLMT0-7	0XF09C	0XF0A3
High Limit 0-7 Registers	HILIM0-7	ADHLMT0-7	ADC_HILIM0-7	ADC_ADHLMT0-7	ADC_ADHLMT0-7	0XF0A4	0XF0AB
Offset 0-7 Registers	OFFST0-7	ADOFS0-7	ADC_OFFST0-7	ADC_ADOFS0-7	ADC_ADOFS0-7	0XF0AC	0XF0B3
Power Control Register	PWR	ADPOWER	ADC_PWR	ADC_ADPOWER	ADC_ADPOWER	0XF0B4	
Calibration Register	CAL		ADC_CAL	ADC_ADCAL	ADC_ADCAL	0XF0B5	
Computer Operating Properly (COP) Module							
Control Register	CTRL	COPCTL	COP_CTRL	COPCTL	COPCTL	0XF120	
Timeout Register	TOUT	COPTO	COP_TOUT	COPTO	COPTO	0XF121	
Counter Register	CNTR	COPCTR	COP_CNTR	COPCTR	COPCTR	0XF122	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 2 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Inter-Integrated Circuit Interface (I^2C) Module							
Control Register	CTRL	IBCR	I2C_CTRL	I2C_IBCR	I2C_IBCR	0xF280	
Target Address Register	TAR		I2C_TAR	I2CTAR	I2C_TAR	0xF282	
Slave Address Register	SAR		I2C_SAR	I2CSAR	I2C_SAR	0xF242	
Data Buffer & Command Register	DATA		I2C_DATA	I2C_DATACMD	I2C_DATACMD	0xF288	
Standard Speed Clock SCL High Count Register	SSHCNT		I2C_SS_SCL_HCNT	I2C_SS_SCLHCNT	I2C_SS_SCLHCNT	0xF28A	
Standard Speed Clock SCL Low Count Register	SSLCNT		I2C_SS_SCL_LCNT	I2C_SS_SCLLCNT	I2C_SS_SCLLCNT	0xF28C	
Fast Speed Clock SCL High Count Register	FSHCNT		I2C_FS_SCL_HCNT	I2C_FS_SCLHCNT	I2C_FS_SCLHCNT	0xF28E	
Fast Speed Clock SCL Low Count Register	FSLCNT		I2C_FS_SCL_LCNT	I2C_FS_SCLLCNT	I2C_FS_SCLLCNT	0xF290	
Interrupt Status Register	ISTAT		I2C_INTR_STAT	I2C_INTRSTAT	I2C_INTRSTAT	0xF296	
Interrupt Mask Register	IENBL		I2C_INTR_MASK	I2C_INTRMASK	I2C_INTRMASK	0xF298	
Raw Interrupt Status Register	RISTAT		I2C_RAW_INTR_ STAT	I2C_RAW_INTRSTAT	I2C_RAW_INTRSTAT	0xF29A	
Receive FIFO Threshold Level Register	RXFT		I2C_RXTL		I2C_RXTL	0xF29C	
Transmit FIFO Threshold Level Register	TXFT		I2C_TXTL		I2C_TXTL	0xF29E	
Clear Combined & Individual Interrupts Register	CLRINT		I2C_CLRINTR		I2C_CLRINTR	0xF2A0	
Clear Receive Under Interrupt Register	CLRRXUND		I2C_CLR_RXUNDER		I2C_CLR_RXUNDER	0xF2A2	
Clear Receive Over Interrupt Register	CLRRXOVR		I2C_CLROVER		I2C_CLROVER	0xF2A4	
Clear Transmit Over Register	CLRTXOVR		I2C_CLR_TXOVER		I2C_CLR_TXOVER	0xF2A6	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 3 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Clear Read Required Interrupt Register	CLRRDREQ		I2C_CLR_RDREQ		I2C_CLR_RDREQ	0xF2A8	
Clear Transmit Abort Interrupt Register	CLRTXABRT		I2C_CLR_TXABRT		I2C_CLR_TXABRT	0xF2AA	
Clear Transmit Done Interrupt Register	CLRTXDONE		I2C_CLR_RXDONE		I2C_CLR_RXDONE	0xF2AC	
Clear Activity Interrupt Register	CLRACT		I2C_CLRACTIVITY		I2C_CLRACTIVITY	0xF2AE	
Clear Stop Detect Interrupt Register	CLRSTPDET		I2C_CLR_STOPDET		I2C_CLR_STOPDET	0xF2B0	
Clear Start Detect Interrupt Register	CLRSTDET		I2C_CLR_STAR_DET		I2C_CLR_STAR_DET	0xF2B2	
Clear General Call Interrupt Register	CLRGC		I2C_CLR_GENCALL		I2C_CLR_GENCALL	0xF2B4	
Enable Register	ENBL		I2C_ENABLE		I2C_ENABLE	0xF2B6	
Status Register	STAT		I2C_STAT		I2C_STAT	0xF2B8	
Transmit FIFO Level Register	TXFLR		I2C_TXFLR		I2C_TXFLR	0xF2BA	
Receive FIFO Level Register	RXFLR		I2C_RXFLR		I2C_RXFLR	0xF2BC	
Transmit Abort Source Register	TXABRTSRC		I2C_TX_ABRTSRC		I2C_TX_ABRTSRC	0xF2C0	
On-Clock Chip Synthesis (OCCS) Module							
Control Register	CTRL	PLLCR	OCCS_CTRL	PLLCR	PLLCR	0xF130	
Divide-By Register	DIVBY	PLLDB	OCCS_DIVBY	PLLDB	PLLDB	0xF131	
Status Register	STAT	PLLSR	OCCS_STAT	PLLSR	PLLSR	0xF132	
Oscillator Control Register	OCTRL	OSCTL	OCCS_OCTRL	OSCTL	OSCTL	0xF135	
Clock Check Register	CLKCHK		OCCS_CLCHK	PLLCLCHK	OCCS_CLCHK	0xF136	
Protection Register	PROT		OCCS_PROT	PLLPROT	OCCS_PROT	0xF137	
Clock Divider Register	CLKDIV	FMCLKD	FM_CLKDIV	FMCLKD	FMCLKD	0xF400	
Configuration Register	CNFG	FMCR	FM_CNFG	FMCR	FMCR	0xF401	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 4 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Security High Half Register	SECHI	FMSECH	FM_SECHI	FMSECH	FMSECH	0xF403	
Security Low Half Register	SECLO	FMSECL	FM_SECLO	FMSECL	FMSECL	0xF404	
Protection Register	PROT	FMPROT	FM_PROT	FMPROT	FMPROT	0xF410	
User Status Register	USTAT	FMUSTAT	FM_USTAT	FMUSTAT	FMUSTAT	0xF413	
Command Register	CMD	FMCMD	FM_CMD	FMCMD	FMCMD	0xF414	
Data Buffer Register	DATA	FMDATA	FM_DATA	FMDATA	FMDATA	0xF418	
Info Optional Data 1 Register	OPT1	FMOPT1	FM_OPT1	FMOPT1	FMOPT1	0xF41**B**	
Test Array Signature Register	TSTSIG	FMTST_SIG	FM_TSTSIG	FMTST_SIG	FMTST_SIG	0xF41D	
General Purpose Input/Output (GPIO) Module							
					x = A (***n***=0) B (***n***=1) C (***n***=2) D (***n***=3)		
Pull-Up Enable Register	PUPEN	PUR	GPIO***x***_PUPEN	GPIO***x***_PUR	GPIO_***x***_PUR	0xF1***n***0	
Data Register	DATA	DR	GPIO***x***_DATA	GPIO***x***_DR	GPIO_***x***_DR	0xF1***n***1	
Data Direction Register	DDIR	DDR	GPIO***x***_DDIR	GPIO***x***_DDR	GPIO_***x***_DDR	0xF1***n***2	
Peripheral Enable Register	PEREN	PER	GPIO***x***_PEREN	GPIO***x***_PER	GPIO_***x***_PER	0xF1***n***3	
Interrupt Assert Register	IASSRT	IAR	GPIO***x***_IASSRT	GPIO***x***_IAR	GPIO_***x***_IAR	0xF1***n***4	
Interrupt Enable Register	IEN	IENR	GPIO***x***_IEN	GPIO***x***_IENR	GPIO_***x***_IENR	0xF1***n***5	
Interrupt Polarity Register	IPOL	IPOLR	GPIO***x***_IPOL	GPIO***x***_IPOLR	GPIO_***x***_IPOLR	0xF1***n***6	
Interrupt Pending Register	IPEND	IPR	GPIO***x***_IPEND	GPIO***x***_IPR	GPIO_***x***_IPR	0xF1***n***7	
Interrupt Edge-Sensitive Register	IEDGE	IESR	GPIO***x***_IEDGE	GPIO***x***_IESR	GPIO_***x***_IESR	0xF1***n***8	
Push-Pull Mode Registers	PPOUTM	PPMODE	GPIOx_PPOUTM	GPIO***x***_PPMODE	GPIO_***x***_PPMODE	0xF1***n***9	
Raw Data Input Register	RDATA	RAWDATA	GPIO***x***_RDATA	GPIOx_RAWDATA	GPIO_***x***_RAWDATA	0xF1***n***A	
Output Drive Strength Register	DRIVE	DRIVE	GPIO***x***_DRIVE	GPIO***x***_DRIVE	GPIO_***x***_DRIVE	0xF1***n***B	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 5 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Pulse Width Modulator (PWM) Module							
Control Register	CTRL	PMCTL	PWM_CTRL	PWM_PMCTL	PWM_PMCTL	0xF0C0	
Fault Control Register	FCTRL	PMFCTL	PWM_FCTRL	PWM_PMFCTL	PWM_PMFCTL	0xF0C1	
Fault Status/Acknowledge Regis.	FLTACK	PMFSA	PWM_FLTACK	PWM_PMFSA	PWM_PMFSA	0xF0C2	
Output Control Register	OUT	PMOUT	PWM_OUT	PWM_PMOUT	PWM_PMOUT	0xF0C3	
Counter Register	CNTR	PMCNT	PWM_CNTR	PWM_PMCNT	PWM_PMCNT	0xF0C4	
Counter Modulo Register	CMOD	MCM	PWM_CMOD	PWM_MCM	PWM_MCM	0xF0C5	
Value 0-5 Registers	VAL0-5	PMVAL0-5	PWM_VAL0-5	PWM_PMVAL0-5	PWM_PMVAL0-5	0xF0C6	0xF0CB
Deadtime 0-1 Registers	DTIM0-1	PMDEADTM0-1	PWM_DTIM0-1	PWM_PMDEADTM0-1	PWM_PMDEADTM0-1	0xF0CC	0xF0CD
Disable Mapping 1-2 Registers	DMAP1-2	PMDISMAP1-2	PWM_DMAP1-2	PWM_PMDISMAP1-2	PWM_PMDISMAP1-2	0xF0CE	0xF0CF
Configure Register	CNFG	PMCFG	PWM_CNFG	PWM_PMCFG	PWM_PMCFG	0xF0D0	
Channel Control Register	CCTRL	PMCCR	PWM_CCTRL	PWM_PMCCR	PWM_PMCCR	0xF0D1	
Port Register	PORT	PMPORT	PWM_PORT	PWM_PMPORT	PWM_PMPORT	0xF0D2	
Internal Correction Control Register	ICCTRL	PMICCR	PWM_ICCTRL	PWM_PMICCR	PWM_PMICCR	0xF0D3	
Source Control Register	SCTRL	PMSRC	PWM_SCTRL	PWM_PMSRC	PWM_PMSRC	0xF0D4	
Synchronization Window Register	SYNC		PWM_SYNC	PWM_SYNC	PWM_SYNC	0xF0D5	
Fault Filter 0-3 Register	FFILT0-3		PWM_FFILT0-3	PWM_FFILT0-3	PWM_FFILT0-3	0xF0D6	0xF0D9
Multi-Scalable Controller Area Network (MSCAN) Module							
Control 0 Register	CTRL0		CAN_CTRL0		CANCTRL0	0XF800	
Control 1 Register	CTRL1		CAN_CTRL1		CANCTRL1	0XF801	
Bus Timing 0 Register	BTR0		CAN_BTR0		CANBTR0	0XF802	
Bus Timing 1 Register	BTR1		CAN_BTR1		CANBTR1	0XF803	
Receive Flag Register	RFLG		CAN_RFLG		CANRFLG	0XF804	
Receiver Interrupt Enable Register	RIER		CAN_RIER		CANRIER	0XF805	
Transmitter Flag Register	TFLG		CAN_TFLG		CANTFLG	0XF806	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 6 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Transmitter Interrupt Enable Register.	TIER		CAN_TIER		CANTIER	0XF807	
Transmitter Msg Abort Request Register	TARQ		CAN_TARQ		CANTARQ	0XF808	
Transmitter Message Abort Acknowledge Register	TAAK		CAN_TAAK		CANTAAK	0XF809	
Transmitter FIFO Selection Register	TBSEL		CAN_TBSEL		CANTBSEL	0XF80A	
Identifier Acceptance Control Register	IDAC		CAN_IDAC		CANIDAC	0XF80B	
Miscellaneous Register	MISC		CAN_MISC		CANMISC	0XF80D	
Receive Error Register	RXERR		CAN_RXERR		CANRXERR	0XF80E	
Transmit Error Register	TXERR		CAN_TXERR		CANTXERR	0XF80F	
Identfier Acceptance 0-3 Registers	IDAR0-3		CAN_IDAR0-3		CANIDAR0-3	0xF810	0xF813
Identifier Mask 0-3 Registers	IDMR0-3		CAN_IDMR0-3		CANIDMR0-3	0xF814	0xF817
Identifier Acceptance 4-7 Register	IDAR4-7		CAN_IDAR4-7		CANIDAR4-7	0xF818	0xF81B
Identifier Mask 4-7 Registers	IDMR4-7		CAN_IDMR4-7		CANIDMR4-7	0xF81C	0xF81F
Foreground Receive FIFO Register	RXFG		CAN_RXFG		CANRXFG	0xF82F	0xF820
Foreground Transmit FIFO Register	TXFG		CAN_TXFG		CANTXFG	0xF830	0xF83F
Power Supervisor (PS) Module							
Control Register	CTRL	LVICONTROL	PS_CTRL	LVICONTROL	LVICTRL	0xF140	
Status Register	STAT	LVISTATUS	PS_STAT	LVISTATUS	LVISR	0xF141	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 7 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Queued Serial Communications Interface (QSCI) Module							
					n = 0, 1		
Baud Rate Register	RATE		QSCI_RATE		QSCI_SCIBR	0xF2***n***0	
Control 1 Register	CTRL1		QSCI_CTRL1		QSCI_SCICR	0xF2***n***1	
Control 2 Register	CTRL2		QSCI_CTRL2		QSCI_SCICR2	0xF2***n***2	
Status Register	STAT		QSCI_STAT		QSCI_SCISR	0xF2***n***3	
Data Register	DATA		QSCI_DATA		QSCI_SCIDR	0xF2***n***4	
Queued Serial Peirpheral Interface (QSPI) Module							
Status and Control Register	SCTRL		QSPI_SCTRL		QSPI_SPSCR	0xF2***n***0	
Data Size and Control Register	DSCTRL		QSPI_DSCTRL		QSPI_SPDSR	0xF2***n***1	
Data Receive Register	DRCV		QSPI_DRCV		QSPI_SPDRR	0xF2***n***2	
Data Transmit Register	DXMIT		QSPI_DXMIT		QSPI_SPDTR	0xF2***n***3	
FIFO Control Register	FIFO		QSPI_FIFO		QSPI_SPFIFO	0xF2***n***4	
Word Delay Register	DELAY		QSPI_WAIT		QSPI_SPWAIT	0xF2***n***5	
Quad-Timer (TMR) Module							
					n = 0, 1, 2, 3		
Compare 1 Register	COMP1	TMRCMP1	TMR***n***_COMP1	TMR***n***_CMP1	TMR***n***_CMP1	0xF0***n***0	
Compare 2 Register	COMP2	TMRCMP2	TMR***n***_COMP2	TMR***n***_CMP2	TMR***n***_CMP2	0xF0***n***1	
Capture Register	CAPT	TMRCAP	TMR***n***_CAPT	TMR***n***_CAP	TMR***n***_CAP	0xF0***n***2	
Load Register	LOAD	TMRLOAD	TMR***n***_LOAD	TMR***n***_LOAD	TMR***n***_LOAD	0xF0***n***3	
Hold Register	HOLD	TMRHOLD	TMR***n***_HOLD	TMR***n***_HOLD	TMR***n***_HOLD	0xF0***n***4	
Counter Register	CNTR	TMRCNTR	TMR***n***_CNTR	TMR***n***_CNTR	TMR***n***_CNTR	0xF0***n***5	
Control Register	CTRL	TMRCTRL	TMR***n***_CTRL	TMR***n***_CTRL	TMR***n***_CTRL	0xF0***n***6	
Status and Control Register	SCTRL	TMRSCR	TMR***n***_SCTRL	TMR***n***_SCR	TMR***n***_SCR	0xF0***n***7	
Comparator Load 1 Register	CMPLD1	TMRCMPLD1	TMR***n***_CMPLD1	TMR***n***_CMPLD1	TMR***n***_CMPLD1	0xF0***n***8	
Comparator Load 2 Register	CMPLD2	TMRCMPLD2	TMR***n***_CMPLD2	TMR***n***_CMPLD2	TMR***n***_CMPLD2	0xF0***n***9	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 8 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Comparator Status/Control Register	CSCTRL	TMRCOMSCR	TMR*n*_CSCTRL	TMR*n*_COMSCR	TMR*n*_COMSCR	0xF0*n*A	
Input Filter Register	FILT		TMR*n*_FILT	TMR*n*_FILT	TMR*n*_FILT	0xF0*n*B	
Enable Register	ENBL		TMR*n*_ENBL	TMR*n*_ENBL	TMR*n*_ENBL	0xF0*n*F	
Voltage Regulator (VREG) Module							
See SIM section on the next page							
Programmable Interval Timer (PIT) Module							
					n = 0, 1, 2		
Control Register	CTRL		PIT*n*_CTRL	PITCTRL0-2	PIT*n*_CTRL	0xF1*n*0	
Modulo Register	MOD		PIT*n*_MOD	PITMOD0-2	PIT*n*_MOD	0xF1*n*1	
Counter Register	CNTR		PIT*n*_CNTR	PITCNTR0-2	PIT*n*_CNTR	0xF1*n*2	
Digital-to-Analog Converter (DAC) Module							
					n = 0, 1		
Control Register	CTRL		DAC*n*_CTRL	DACCTRL0-2	DAC*n*_CTRL	0xF1*n*0	
Data Register	DATA		DAC*n*_DATA	DACDATA0-2	DAC*n*_DATA	0xF1*n*1	
Step Register	STEP		DAC*n*_STEP	DACSTEP0-2	DAC*n*_STEP	0xF1*n*2	
Minimum Value Register	MINVAL		DAC*n*_MINVAL	DACMINVAL0-2	DAC*n*_MINVAL	0xF1*n*3	
Maximum Value Register	MAXVAL		DAC*n*_MAXVAL	DACMAXVAL0-2	DAC*n*_MAXVAL	0xF1*n*4	
Comparator (CMP) Module							
					A x = E ***B x*** = F		
Control Register	CTRL		CMP_CTRL	CMP*x*_CTRL	CMP*x*_CTRL	0xF1*x*0	
Status Register	STAT		CMP_STAT	CMP*x*_STAT	CMP*x*_STAT	0xF1*x*1	
Filter Register	FILT		CMP_FILT	CMP*x*_FILT	CMP*x*_FILT	0xF1*x*2	

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 9 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Interrupt Controller (ITCN) Module							
Interrupt Priority 0-4 Registers	N/A	N/A	ITCN_IPR0-4	ITCN_IPR0-4	INTC_IPR0-4	0XF060	0XF064
Vector Base Address Register	N/A	N/A	ITCN_VBA	ITCN_VBA	INTC_VBA	0XF065	
Fast Interrupt Match 0 Register	N/A	N/A	ITCN_FIM0	ITCN_FIM0	INTC_FIM0	0XF066	
Fast Interrupt Vector Address Low 0	N/A	N/A	ITCN_FIVAL0	ITCN_FIVAL0	INTC_FIVAL0	0XF067	
Fast Interrupt Vector Address High 0	N/A	N/A	ITCN_FIVAH0	ITCN_FIVAH0	INTC_FIVAH0	0XF068	
Fast Interrupt Match 1 Register	N/A	N/A	ITCN_FIM1	ITCN_FIM1	INTC_FIM1	0xF069	
Fast Interrupt Vector Address Low 1	N/A	N/A	ITCN_FIVAL1	ITCN_FIVAL1	INTC_FIVAL1	0xF06A	
Fast Interrupt Vector Address High 1	N/A	N/A	ITCN_FIVAH1	ITCN_FIVAH1	INTC_FIVAH1	0xF06B	
Interrupt Pending 0 Register	N/A	N/A	ITCN_IRQP0	ITCN_IRQP0	INTC_IRQP0	0xF06C	
Interrupt Pending 1 Register	N/A	N/A	ITCN_IRQP1	ITCN_IRQP1	INTC_IRQP1	0xF06D	
Interrupt Pending 2 Register	N/A	N/A	ITCN_IRQP2	ITCN_IRQP2	INTC_IRQP2	0xF06E	
System Integration Module (SIM)							
Control Register	N/A	N/A	SIM_CTRL	SIM_CONTROL	SIM_CONTROL	0xF100	
Reset Status Register	N/A	N/A	SIM_RSTAT	SIM_RSTSTS	SIM_RSTSTS	0xF101	
Software Control 0-3 Registers	N/A	N/A	SIM_SWC0-3	SIM_SCR0-3	SIM_SCR0-3	0xF102	0xF105
Most Significant Half JTAG ID	N/A	N/A	SIM_MSHID	SIM_MSH_ID	SIM_MSH_ID	0xF106	
Least Significant Half JTAG ID	N/A	N/A	SIM_LSHID	SIM_LSH_ID	SIM_LSH_ID	0xF107	
Power Control Register	N/A	N/A	SIM_PWR	SIM_POWER		0xF108	
Clock Out Select Register	N/A	N/A	SIM_CLKOUT	SIM_CLKOSR	SIM_CLKOSR	0xF10A	
Peripheral Clock Rate Register	N/A	N/A	SIM_PCR	SIM_PCR	SIM_PCR	0xF10B	
Peripheral Clock Enable 0-1 Register	N/A	N/A	SIM_PCE0-1	SIM_PCE0-1	SIM_PCE0-1	0xF10C	0xF10D

Table B-1. Cross-Reference to Legacy and Revised Acronyms (Sheet 10 of 10)

Register Name	Peripheral Reference Manual		Data Sheet		Processor Expert Acronym	Memory Address	
	New Acronym	Legacy Acronym	New Acronym	Legacy Acronym		Start	End
Peripheral Stop Disable 0-1 Register	N/A	N/A	SIM_SD0-1	SIM_SD0-1	SIM_SD0-1	0xF10E	0xF10F
I/O Short Address Location High Register	N/A	N/A	SIM_ISALH	SIM_ISALH	SIM_ISALH	0xF110	
I/O Short Address Location Low Register	N/A	N/A	SIM_ISALL	SIM_ISALL	SIM_ISALL	0xF111	
Protection Register	N/A	N/A	SIM_PROT	SIM_PROT	SIM_PROT	0xF112	
GPIOA Peripheral Select 0 Register	N/A	N/A	SIM_GPISA0	SIM_GPISA0	SIM_GPISA0	0xF113	
GPIOA Peripheral Select 0 Register	N/A	N/A	SIM_GPSA1	SIM_GPSA1	SIM_GPSA1	0xF114	
GPIOB Peripheral Select 0 Register	N/A	N/A	SIM_GPSB0	SIM_GPSB0	SIM_GPSB0	0xF115	
GPIOB Peripheral Select 1 Register	N/A	N/A	SIM_GPSB1	SIM_GPSB1	SIM_GPSB1	0xF116	
GPIO Perip. Select Register for GPIO C & D	N/A	N/A	SIM_GPSCD	SIM_GPSCD	SIM_GPSCD	0xF117	
Internal Peripheral. Select Register for PWM	N/A	N/A	SIM_ISPWM	SIM_ISPWM	SIM_ISPWM	0xF118	
Internal Peripheral Select Register for DAC	N/A	N/A	SIM_IPSDAC	SIM_IPSDAC	SIM_IPSDAC	0xF119	
Internal Peripheral Select Registerfor TMRA	N/A	N/A	SIM_IPSTMRA	SIM_IPSTMRA	SIM_IPSTMRA	0xF11A	

B.3 Programmer Sheets

The following pages provide programmer sheets summarizing functions of the bits in various registers in the 56F8000 family found in this manual. The programmer sheets provide room to write in the value of each bit and the hexadecimal value for each register. Programmers may photocopy these sheets.

The programmer sheets are arranged in the same order as the sections in this document. Table B-2 lists the programmer sheets by module, the registers in each module, and the pages in this appendix where the programmer sheets are located.

NOTE:

Reserved bits in all registers should only be set to zero unless otherwise stated.

Please see the applicable *Device Data Sheet* or *Core Manual* for register information about references to *Data Sht.* and *Core Man.* in the following table. However, interrupt Controller (ITCN) and System Integration Module (SIM) register are included in this addendum for convenience.

Table B-2. List of Programmer Sheets (Sheet 1 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Enhanced On Chip Emulation (EOnCE)	**EOnCE_BASE: 56F80xx = $FFFFFB - $FFFF8A**	
EOnCE_		
Transmit Register Upper Word/Receive Register Upper Word	OTX1 / ORX1	Core Man
Transmit Register/Receive Register	OTX / ORX (32 Bits)	Core Man
Transmit and Receive Status and Control Register	OTXRXSR (8 Bits)	Core Man
Core Lock/Unlock Status Register	OCLSR (8 Bits)	Core Man
Control Register	OCR	Core Man
Instruction Step Counter	OXCNTR (24 Bits)	Core Man
Status Register	OSR	Core Man
Peripheral Base Address Register	OBASE (8 Bits)	Core Man
Trace Buffer Control Register	OTBCR	Core Man
Trace Buffer Pointer Register	OTBPR (8 Bits)	Core Man
Trace Buffer Register Stages	OTB (21-24 Bits/Stage)	Core Man
Breakpoint Unit [0] Control Register	OBCR (24 Bits)	Core Man
Breakpoint 1 Unit [0] Address Register	OBAR1 (24 Bits)	Core Man
Breakpoint 2 Unit [0] Address Register	OBAR2 (32 Bits)	Core Man
Breakpoint 1 Unit [0] Mask Register	OBMSK (32 Bits)	Core Man
Breakpoint [0] Unit Counter	OBCNTR	Core Man
External Signal Control Register	OESCR	Core Man
Interrupt Control (ITCN)	**ITCN_BASE: 56F80xx = $00F0E0**	
ITCN_		
Interrupt Priority 0 Register	IPR0	Data Sht
Interrupt Priority 1 Register	IPR1	Data Sht
Interrupt Priority 2 Register	IPR2	Data Sht
Interrupt Priority 3 Register	IPR3	Data Sht
Interrupt Priority 4 Register	IPR4	Data Sht

Table B-2. List of Programmer Sheets (Sheet 2 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Interrupt Priority 5 Register	IPR5	Data Sht
Interrupt Priority 6 Register	IPR6	Data Sht
Vector Base Address Register	VBA	Data Sht
Fast Interrupt Match 0 Register	FIM0	Data Sht
Fast Interrupt 0 Vector Address Low Register	FIVAL0	Data Sht
Fast Interrupt 0 Vector Address High Register	FIVAH0	Data Sht
Fast Interrupt 1 Match Register	FIM1	Data Sht
Fast Interrupt 1 Vector Address Low Register	FIVAL1	Data Sht
Fast Interrupt 1 Vector Address High Register	FIVAH1	Data Sht
IRQ Pending 0 Register	IRQP0	Data Sht
IRQ Pending 1 Register	IRQP1	Data Sht
IRQ Pending 2 Register	IRQP2	Data Sht
IRQ Pending 3 Register	IRQP3	Data Sht
Interrupt Control Register	ICTRL	Data Sht

System Integration Module (SIM)	**SIM_BASE: 56F80xx = $00F100**	
	SIM_	
Control Register	CTRL	Data Sht
Reset Status Register	RSTAT	Data Sht
Software Control 0-3 Registers	SWC0-3	Data Sht
Most Significant Half of JTAG ID Register	MSHID	Data Sht
Least Significant Half of JTAG ID Register	LSHID	Data Sht
Power Control Register	PWR	Data Sht
Clock Output Select Register	CLKOUT	Data Sht
Peripheral Clock Rate Register	PCR	Data Sht
Peripheral Clock Enable 0 Register	PCE0	Data Sht
Peripheral Clock Enable 1 Register	PCE1	Data Sht
Stop Disable 0 Register	SD0	Data Sht
Stop Disable 1 Register	SD1	Data Sht
I/O Short Address Location High Register	IOSAHI	Data Sht
I/O Short Address Location Low Register	IOSALO	Data Sht
Protection Register	PROT	Data Sht
GPIO Peripheral Select 0 for GPIOA0 Register	GPSA0	Data Sht
GPIO Peripheral Select 1 for GPIOA1 Register	GPSA1	Data Sht
GPIO Peripheral Select 0 for GPIOB0 Register	GPSB0	Data Sht
GPIO Peripheral Select 1 for GPIOB1 Register	GPSB1	Data Sht
GPIO Peripheral Select for GPIOC and GPIOD Register	GPSCD	Data Sht
Internal Peripheral Select for PWM Register	IPSPWM	Data Sht
Internal Peripheral Select for DAC Register	IPSDAC	Data Sht
Internal Peripheral Select for TMRA Register	IPSTRMA	Data Sht

Table B-2. List of Programmer Sheets (Sheet 3 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Analog to Digital Converter (ADC)	**ADC_BASE: 56F80xx = $00F080**	
ADC_		
Control 1 Register	CTRL1	B-21
Control 2 Register	CTRL2	B-23
Zero Crossing Control Register	ZXCTRL	B-24
Channel List 1 Register	CLST1	B-25
Channel List 2 Register	CLST2	B-26
Channel List 3 Register	CLST3	B-27
Channel List 4 Register	CLST4	B-28
Sample Disable Register	SDIS	B-29
Status Register	STAT	B-30
Conversion Ready Register	RDY	B-32
Limit Status Register	LIMSTAT	B-33
Zero Crossing Status Register	ZXSTAT	B-34
Result 0-7 Registers	RSLT0–7	B-35
Result 8-15 Registers	RSLT8–15	B-36
Low Limit 0–7 Register s	LOLIM0–7	B-37
High Limit 0–7 Registers	HILIM0–7	B-37
Offset 0–7 Registers	OFFST0–7	B-38
Power Control Register	PWR	B-39
Calibration Register	CAL	B-41

Computer Operating Properly (COP)	**COP_BASE: 56F80xx = $00F120**	
COP_		
Control Register	CTRL	B-43
Timeout Register	TOUT	B-45
Counter Register	CTNR	B-46

Inter-Integrated Circuit (I^2C)	**I^2C_BASE: 56F80xx = $00F280**	
I^2C_		
Control Register	CTRL	B-47
Target Address Register	TAR	B-48
Slave Address Register	SAR	B-49
Data Buffer and Command Register	DATA	B-50
Standard Speed I^2C Clock SCL High Count Register	SSHCNT	B-51
Standard Speed I^2C Clock SCL Low Count Register	SSLCNT	B-52
Fast Speed Clock I^2C SCL High Count Register	FSHCNT	B-53
Fast Speed Clock I^2C SCL Low Count Register	FSLCNT	B-54
Interrupt Status Register	ISTAT	B-55
Interrupt Mask Register	IMASK	B-61
Raw Interrupt Status Register	RISTAT	B-64
Receive FIFO Threshold Register	RXFT	B-64
Transmit FIFO Threshold Register	TXFT	B-65
Clear Combined and Individual Interrupt Register	CLRINT	B-66

Table B-2. List of Programmer Sheets (Sheet 4 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Clear Receive Under Interrupt Register	CLRRXUND	B-67
Clear Receive Over Interrupt Register	CLRRXOVR	B-68
Clear Transmit Over Interrupt Register	CLRTXOVR	B-69
Clear Read Required Interrupt Register	CLRRDREQ	B-70
Clear Transmit Abort Interrupt Register	CLRTXABRT	B-71
Clear Receive Done Interrupt Register	CLRRXDONE	B-72
Clear Activity Interrupt Register	CLRACT	B-73
Clear Stop Detect Interrupt Register	CLRSTPDET	B-74
Clear Start Detect Interrupt Register	CLRSTDET	B-75
Clear General Call Interrupt Register	CLRGC	B-76
Enable Register	ENBL	B-77
Status Register	STAT	B-78
Transmit FIFO Level Register	TXFLR	B-80
Receive FIFO Level Register	RXFLR	B-81
Transmit Abort Source Register	TXABRTSRC	B-82

On Chip Clock Synthesis (OCCS)	**OCCS_BASE: 56F80xx = $00F130**	
OCCS_		
Control Register	CTRL	B-84
Divide-By Register	DIVBY	B-86
Status Register	STAT	B-87
Oscillator Control Register	OCTRL	B-89
External Clock Check Register	CLKCHK	B-91
Protection Register	PROT	B-92

Flash Module (FM)	**FM_BASE: 56F80xx = $00F400**	
FM_		
Clock Divider Register	CLKDIV	B-93
Configuration Register	CNFG	B-94
Security High Register	SECHI	B-95
Security Low Register	SECLO	B-95
Protection Register	PROT	B-96
User Status Register	USTAT	B-97
Command Register	CMD	B-99
Data Register	DATA	B-100
Optional Data 1 Register	OPT1	B-101
Test Array Signature Register	TSTSIG	B-102

Table B-2. List of Programmer Sheets (Sheet 5 of 9)

Register Type/Name	Base Address/ New Acronym	Location
General Purpose Input/Output (GPIO)	**GPIOA_BASE: 56F80xx = See Data Sheet for Address**	
GPIO_		
Pull-Up Enable Register	PUPEN	B-103
Data Register	DATA	B-104
Data Direction Register	DDIR	B-105
Peripheral Enable Register	PEREN	B-106
Interrupt Assert Register	IASSRT	B-107
Interrupt Enable Register	IEN	B-108
Interrupt Polarity Register	IPOL	B-109
Interrupt Pending Register	IPEND	B-110
Interrupt Edge Sensitive Register	IEDGE	B-111
Push/Pull Output Mode Control Register	PPOUTM	B-112
Raw Data Register	RDATA	B-113
Drive Strength Control Register	DRIVE	B-114

Pulse Width Module (PWM)	**PWM_BASE: 56F80xx = $00F0C0**	
PWM_		
Control Register	CTRL	B-115
Fault Control Register	FCTRL	B-117
Fault Status and Acknowledge Register	FLTACK	B-118
Output Control Register	OUT	B-119
Counter Register	CNTR	B-120
Counter Modulo Register	CMOD	B-121
Value Registers	VAL0-5	B-122
Deadtime 0-1 Registers	DTIM0-1	B-123
Disable Mapping 1-2 Register	DMAP1-2	B-124
Configuration Register	CNFG	B-126
Channel Control Register	CCTRL	B-128
Port Register	PORT	B-129
Internal Correction Control Register	ICCTRL	B-130
Source Control Register	SCTRL	B-131
Synchronization Window Register	SYNC	B-134
Fault Filter 0-3 Registers	FFILT0-3	B-135

Multi-Scalable Controller Area Network (MSCAN)	**MSCAN_BASE: 56F80xx = $00F800**	
MSCAN_		
Control 0 Register	CTRL0	B-136
Control 1 Register	CTRL1	B-139
Bus Timing 0 Register	BTR0	B-141
Bus Timing 1 Register	BTR1	B-142
Receiver Flag Register	RFLG	B-144
Receiver Interrupt Enable Register	RIER	B-146
Transmitter Flag Register	TFLG	B-147
Transmitter Interrupt Enable Register	TIER	B-148
Transmitter Message Abort Request Register	TARQ	B-149

Table B-2. List of Programmer Sheets (Sheet 6 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Transmitter Message Abort Acknowledge Register	TAAK	B-150
Transmitter Buffer Selection Register	TBSEL	B-151
Identifier Acceptance Control Register	IDAC	B-152
Miscellaneous Register	MISC	B-153
Receive and Transmit Error Counter Registers	RXERR - TXERR	B-154
Identifier Acceptance 0-7 Registers	IDAR0-7	B-155
Identifier Mask 0-7 Registers	IDMR0-7	B-159
Extended Identifier 0-3 Registers	IDR0-3	B-160
Standard Identifier 0 Register	IDR0	B-161
Standard Identifier 1 Register	IDR1	B-162
Data Segment 0-7 Registers	DSR0-7	B-163
Data Length Register	DLR	B-164
Transmit Buffer Priority Register	TBPR	B-165
Time Stamp High and Low Registers	TSRH - TSRL	B-166
Power Supervisor (PS)	**PS_BASE: 56F80xx = $00F140**	
PS_		
Control Register	CTRL	B-167
Status Register	STAT	B-168
Queued Serial Communication Interface (QSCI)	**QSCI0_BASE: 56F80xx = $00F200** **QSCI1_BASE: 56F80xx = $00F210**	
QSCI_		
Baud Rate Register	RATE	B-169
Control 1 Register	CTRL1	B-170
Control 2 Register	CTRL2	B-174
Status Register	STAT	B-175
Data Register	DATA	B-178
Queued Serial Peripheral Interface (QSPI)	**QSPI0 _BASE: 56F80xx = $00F220** **QSPI1 _BASE: 56F80xx = $00F230**	
QSPI_		
Status and Control Register	SCTRL	B-182
Data Size and Control Register	DSCTRL	B-185
Data Receive Register	DRCV	B-186
Data Transmit Register	DXMIT	B-187
FIFO Control Register	FIFO	B-189
Word Delay Register	DELAY	B-190

Table B-2. List of Programmer Sheets (Sheet 7 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Quad Timer (TMR)	**TMRA0_BASE: 56F80xx = $00F000** **TMRA1_BASE: 56F80xx = $00F010** **TMRA2_BASE: 56F80xx = $00F020** **TMRA3_BASE: 56F80xx = $00F030** **TMRB0_BASE: 56F80xx = $00F040** **TMRB1_BASE: 56F80xx = $00F050** **TMRB2_BASE: 56F80xx = $00F060** **TMRB3_BASE: 56F80xx = $00F070**	
TMR_		
Compare 1 Register	COMP1	B-191
Compare 2 Register	COMP2	B-192
Capture Register	CAPT	B-193
Load Register	LOAD	B-194
Hold Register	HOLD	B-195
Counter Register	CNTR	B-196
Control Registers	CTRL	B-197
Status and Control Register	SCTRL	B-201
Comparator Load 1 Register	CMPLD1	B-203
Comparator Load 2 Register	CMPLD2	B-204
Comparator Status and Control Register	CSCTRL	B-205
Input Filter Register	FILT	B-206
Channel Enable Register	ENBL	B-207

Voltage Regulator (VREG)	**VREG_BASE: 56F80xx = $00F140**	
VREG_		
Power Control Register	SIM_PWR	B-208

Programmer Interval Timer (PIT)	**PIT0_BASE: 56F80xx = $00F190** **PIT1_BASE: 56F80xx = $00F1A0** **PIT2_BASE: 56F80xx = $00F1B0**	
PIT_		
Control Register	CTRL	B-209
Modulo Register	MOD	B-210
Counter Register	CNTR	B-211

Digital-to-Analog Converter (DAC)	**DAC0_BASE: 56F80xx = $00F1C0** **DAC1_BASE: 56F80xx = $00F1D0**	
DAC_		
Control Register	CTRL	B-212
Buffered Data Register	DATA	B-214
Step Size Register	STEP	B-215
Minimum Value Register	MINVAL	B-216
Maximum Value Register	MAXVAL	B-217

Table B-2. List of Programmer Sheets (Sheet 8 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Comparator (CMP)	**CMPA_BASE: 56F80xx = $00F1E0** **CMPB_BASE: 56F80xx = $00F1F0**	
CMP_		
Control Register	CTRL	B-219
Status Register	STAT	B-220
Filter Register	FILT	B-221

Interrupt Controller (ITCN)	**ITCN_BASE: 56F80xx = $00F0E0**	
ITCN_		
Interrupt Priority 0 Register	IPR0	B-222
Interrupt Priority 1 Register	IPR1	B-224
Interrupt Priority 2 Register	IPR2	B-226
Interrupt Priority 3 Register	IPR3	B-228
Interrupt Priority 4 Register	IPR4	B-230
Interrupt Priority 5 Register	IPR5	B-232
Interrupt Priority 6 Register	IPR6	B-235
Vector Base Address Register	VBA	B-236
Fast Interrupt Match 0 Register	FIM0	B-237
Fast Interrupt 0 Vector Address Low Register	FIVAL0	B-238
Fast Interrupt 0 Vector Address High Register	FIVAH0	B-239
Fast Interrupt 1 Match Register	FIM1	B-240
Fast Interrupt 1 Vector Address Low Register	FIVAL1	B-241
Fast Interrupt 1 Vector Address High Register	FIVAH1	B-242
IRQ Pending 0 Register	IRQP0	B-243
IRQ Pending 1 Register	IRQP1	B-244
IRQ Pending 2 Register	IRQP2	B-245
IRQ Pending 3 Register	IRQP3	B-246
Interrupt Control Register	ICTRL	B-247

System Integrated Module (SIM)	**SIM_BASE: 56F80xx = $00F100**	
SIM_		
Control Register	CTRL	B-248
Reset Status Register	RSTAT	B-249
Software Control 0-3 Registers	SWC0-3	B-250
Most Significant Half of JTAG ID Register	MSHID	B-251
Least Significant Half of JTAG ID Register	LSHID	B-252
Power Control Register	PWR	B-253
Clock Output Select Register	CLKOUT	B-254
Peripheral Clock Rate Register	PCR	B-255
Peripheral Clock Enable 0 Register	PCE0	B-256
Peripheral Clock Enable 1 Register	PCE1	B-258
Stop Disable 0 Register	SD0	B-260
Stop Disable 1 Register	SD1	B-262
I/O Short Address Location High Register	IOSAHI	B-264
I/O Short Address Location Low Register	IOSALO	B-265

Table B-2. List of Programmer Sheets (Sheet 9 of 9)

Register Type/Name	Base Address/ New Acronym	Location
Protection Register	PROT	B-266
GPIO Peripheral Select 0 for GPIOA0 Register	GPSA0	B-267
GPIO Peripheral Select 1 for GPIOA1 Register	GPSA1	B-268
GPIO Peripheral Select 0 for GPIOB0 Register	GPSB0	B-270
GPIO Peripheral Select 1 for GPIOB1 Register	GPSB1	B-272
GPIO Peripheral Select for GPIOC and GPIOD Register	GPSCD	B-273
Internal Peripheral Select for PWM Register	IPSPWM	B-275
Internal Peripheral Select for DAC Register	IPSDAC	B-276
Internal Peripheral Select for TMRA Register	IPSTRMA	B-277

Application: _____ Date: _____

Programmer: _____

ADC

Control 1 (CTRL1) Register

Please see the following page for continuation of this register

Bits	Name	Description
14	STOP0	**Stop**
		When selected, the current conversion process is stopped. Any further SYNC0 input pulses or modifications to the START0 bit are ignored until the STOP0 bit is cleared. After the ADC is in Stop mode, the results registers can be modified by the processor. This is not the same as Stop mode for the whole chip.
		0 Normal operation
		1 Stop mode
13	START0	**Start**
		A scan is started by writing 1 to the START0 bit. This is a *write-only* bit. Writing 1 to the START0 bit again will be ignored until the end of the current scan. 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	**Synchronization**
		A conversion can be initiated by asserting a positive edge on the SYNC0 input. Any subsequent SYNC0 input pulses while the scan remains in process are ignored. Refer to the functional description of power modes for further details.
		0 Scan is initiated by a write to START0 bit only
		1 Use the SYNC0 input pulse bit to initiate a scan
11	EOSIE0	**End of Scan Interrupt Enable**
		This bit enables an EOSI0interrupt 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, configured by the ZXCTRL register.
		0 Interrupt disabled
		1 Interrupt enabled

Control 1 (CTRL1) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	STOP0	0	SYNC0	EOSIE0	ZCIE	LLMTIE	HLMTIE	CHNCFG_L				0	SMODE		
	Write			START0													
	Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Reserved Bits

ADC

Control 1 (CTRL1) Register Continued

Bits	Name	Description
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 the Low Limit (LOLIMn) register, bits LLMT[11:0], 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 the High Limit (HILIMn) register, bits HLMT[11:0], before the Offset register value is subtracted.
		0 Interrupt disabled
		1 Interrupt enabled
7 - 4	CHNCFG_L	**Channel Configure Low**
		The inputs can be configured for either single-ended or differential conversions.
		xxx1 Inputs AN0–AN1 configured as differential inputs (AN0 is + and AN1 is -)
		xxx0 Inputs AN0–AN1 configured as single ended inputs
		xx1x Inputs AN2–AN3 configured as differential inputs (AN2 is + and AN3 is -)
		xx0x Inputs AN2–AN3 configured as singled ended inputs
		x1xx Inputs AN4–AN5 configured as differential inputs (AN4 is + and AN5 is -)
		x0xx Inputs AN4–AN5 configured as singled ended inputs
		1xxx Inputs AN6–AN7 configured as differential inputs (AN6 is + and AN7 is -)
		0xxx Inputs AN6–AN7 configured as singled ended inputs
2 - 0	SMODE	**Scan Mode Control**
		SMODE controls the scan mode of the ADC module. (See **Section 2.7.1.10** for details.)

Control 1 (CTRL1) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	STOP0	0	SYNC0	EOSIE0	ZCIE	LLMTIE	HLMTIE	CHNCFG_L				0	SMODE		
	Write			START0													
	Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Control 2 (CTRL2) Register Under Sequential Scan Modes

Bits	Name	Description
14	STOP1	**Stop1**
		During Parallel Scan modes when SIMULT=0, setting this bit stops parallel scans in the B converter, preventing new ones from starting.
		0 Normal operation
		1 Stop issued command
13	START1	**Start Conversion One**
		During Parallel Scan modes when SIMULT=0, a B converter parallel scan is started by writing one to START1 bit.
		0 No action
		1 Start a B converter parallel scan
12	SYNC1	**Synchronization Enable One**
		During Parallel Scan modes when SIMULT=0, setting SYNC1 to one permits a B converter parallel scan to be initiated by asserting the SYNC1 input for at least one ADC clock cycle.
		0 B converter parallel scan is initiated by a write to START1 bit only
		1 Use a SYNC1 input pulse or START1 bit to initiate a B converter parallel scan
11	EOSIE1	**End of Scan Interrupt One Enable**
		During Parallel Scan modes when SIMULT=0, this bit enables an EOSI1 interrupt to be generated upon completion of a B converter parallel scan.
		0 Interrupt disabled
		1 Interrupt enabled
9 - 6	CHNCHG_H	**Channel Configure High**
		The bits configure the analog inputs for either single ended or differential conversions. (See **Table 2-6**.)
5	SIMULT	**Simultaneous Mode**
		This bit only affects parallel scan modes.
		0 Parallel scans achieved independently
		1 Parallel scans achieved simultaneously (default)
4 - 0	DIV	**Clock Divisor Select**
		The divider circuit generates the ADC clock by dividing the system clock by 2× (DIV[4:0]+1). A DIV value must be chosen so the ADC clock does not exceed 5.33MHz. **Table 2-7** shows ADC clock frequency based on the value of DIV for these various OCCS configurations.

Control 2 (CTRL2) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	STOP1	0	SYNC1	EOSIE1	0	CHANGFG_H				SIMULT	DIV				
	Write			START1													
	Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	0

Reserved Bits

ADC

Zero Crossing Control (ZXCTRL) Register

Bits	Name	Description	
15 - 0	ZCE*n*	**Zero Crossing Enable *n***	
		The ADC Zero Crossing Control (ZXCTRL) register provides the ability to monitor the selected channels and determine the direction of zero crossing triggering the optional interrupt. Zero crossing logic monitors only the sign change between current and previous sample. ZCE0 bit monitors the sample stored in RSLT0, ZCE1 bit monitors RSLT1, ZCE7 bit monitors RSLT7. When the Zero Crossing is disabled for a selected result register, sign changes are not monitored or updated in the ZXSTAT register.	
		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

Zero Crossing Control (ZXCTRL) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

ADC

Channel List1 (CLST1) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
15 - 12	SAMPLE3	**Sample3**
		The Channel List (CLIST*n*) registers contain an ordered list of the channels to be converted when the next scan is initiated. If all samples are enabled, in the SDIS register, a sequential scan of inputs proceeds in order of: SAMPLE0 through SAMPLE15. If one of the parallel sampling modes is selected instead, the converter A sampling order is SAMPLE0-3 and the converter B sampling order is SAMPLE4-7, and then SAMPLE8-11 in parallel with SAMPLE12-15. In Sequential Conversion mode full functionality, offset subtraction and high/low limit checks, is only available on the first eight conversion slots, SAMPLE0-7. In Parallel Conversion mode full functionality is only available on the first four conversion slots of each channel, SAMPLE0-3 for converter A and SAMPLE4-7 for converter B.
11 - 8	SAMPLE2	**Sample2**
		(The above data is applicable to each of the Channel List Registers.)
7 - 4	SAMPLE1	**Sample1**
		(The above data is applicable to each of the Channel List Registers.)
3 - 0	SAMPLE0	**Sample0**
		(The above data is applicable to each of the Channel List Registers.)

Channel List 1 (CLST1) Register Base + $3	Bits	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

Channel List 2 (CLST2) Registers Continued

Please see the following page for continuation of this register

Bits	Name	Description
15 - 12	SAMPLE7	**Sample7**
		The Channel List (CLIST*n*) registers contain an ordered list of the channels to be converted when the next scan is initiated. If all samples are enabled, in the SDIS register, a sequential scan of inputs proceeds in order of: SAMPLE0 through SAMPLE15. If one of the parallel sampling modes is selected instead, the converter A sampling order is SAMPLE0-3 and the converter B sampling order is SAMPLE4-7, and then SAMPLE8-11 in parallel with SAMPLE12-15. In Sequential Conversion mode full functionality, offset subtraction and high/low limit checks, is only available on the first eight conversion slots, SAMPLE0-7. In Parallel Conversion mode full functionality is only available on the first four conversion slots of each channel, SAMPLE0-3 for converter A and SAMPLE4-7 for converter B.
11 - 8	SAMPLE6	**Sample6**
		(The above data is applicable to each of the Channel List Registers.)
7 - 4	SAMPLE5	**Sample5**
		(The above data is applicable to each of the Channel List Registers.)
3 - 0	SAMPLE4	**Sample4**
		(The above data is applicable to each of the Channel List Registers.)

Channel List 2 (CLST2) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SAMPLE7				SAMPLE6				SAMPLE5				SAMPLE4			
	Write																
	Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

Application: _____ Date: _____

Programmer: _____

ADC

Channel List 3 (CLST3) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
15 - 12	SAMPLE11	**Sample11**
		The Channel List (CLIST*n*) registers contain an ordered list of the channels to be converted when the next scan is initiated. If all samples are enabled, in the SDIS register, a sequential scan of inputs proceeds in order of: SAMPLE0 through SAMPLE15. If one of the parallel sampling modes is selected instead, the converter A sampling order is SAMPLE0-3 and the converter B sampling order is SAMPLE4-7, and then SAMPLE8-11 in parallel with SAMPLE12-15. In Sequential Conversion mode full functionality, offset subtraction and high/low limit checks, is only available on the first eight conversion slots, SAMPLE0-7. In Parallel Conversion mode full functionality is only available on the first four conversion slots of each channel, SAMPLE0-3 for converter A and SAMPLE4-7 for converter B.
11 - 8	SAMPLE10	**Sample10**
		(The above data is applicable to each of the Channel List Registers.)
7 - 4	SAMPLE9	**Sample9**
		(The above data is applicable to each of the Channel List Registers.)
3 - 0	SAMPLE8	**Sample8**
		(The above data is applicable to each of the Channel List Registers.)

Channel List 3 (CLST3) Register Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	SAMPLE11				SAMPLE10				SAMPLE9				SAMPLE8			
	Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

Application: _____ Date: _____

Programmer: _____

ADC

Channel List 4 (CLST4) Register Continued

Bits	Name	Description
15 - 12	SAMPLE15	**Sample15**
		The Channel List (CLIST*n*) registers contain an ordered list of the channels to be converted when the next scan is initiated. If all samples are enabled, in the SDIS register, a sequential scan of inputs proceeds in order of: SAMPLE0 through SAMPLE15. If one of the parallel sampling modes is selected instead, the converter A sampling order is SAMPLE0-3 and the converter B sampling order is SAMPLE4-7, and then SAMPLE8-11 in parallel with SAMPLE12-15. In Sequential Conversion mode full functionality, offset subtraction and high/low limit checks, is only available on the first eight conversion slots, SAMPLE0-7. In Parallel Conversion mode full functionality is only available on the first four conversion slots of each channel, SAMPLE0-3 for converter A and SAMPLE4-7 for converter B.
11 - 8	SAMPLE14	**Sample14**
		(The above data is applicable to each of the Channel List Registers.)
7 - 4	SAMPLE13	**Sample13**
		(The above data is applicable to each of the Channel List Registers.)
3 - 0	SAMPLE12	**Sample12**
		(The above data is applicable to each of the Channel List Registers.)

Channel List 4 (CLST4) Register Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SAMPLE15				SAMPLE14				SAMPLE13				SAMPLE12			
	Write																
	Reset	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0

Application: _____ Date: _____

Programmer: _____

ADC

Sample Disable (SDIS) Register

Bits	Name	Description	
15 - 0	**DS*n***	**Disable Sample7-0**	
		The respective SAMPLE*n* can be enabled or disabled where *n* = 0–7.	
		0	Enable SAMPLE*n*
		1	Disable SAMPLE*n* and all subsequent samples. Which samples are actually disabled depends on the conversion mode, sequential/parallel, and the value of SIMULT.

Sample Disable (SDIS) Register Base + $7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DS15	DS14	DS13	DS12	DS11	DS10	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
	Write																
	Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

ADC

Status (STAT) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	CIP0	**Conversion in Progress 0**
		This bit indicates when a scan is in progress.
		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 1**
		This bit indicates when a scan is in progress.
		0 Idle state
		1 A scan cycle is in progress. The ADC will ignore all sync pulses or start commands.
12	EOSI1	**End of Scan Interrupt 1**
		This bit indicates if a scan of analog inputs has been completed since the last read of the STAT register, or since a reset. The EOSI1 bit is cleared by writing one to it. This bit cannot be set by software.
		0 Scan is not completed; no end of scan IRQ pending
		1 Scan cycle is completed; end of scan IRQ pending
11	EOSI0	**End of Scan Interrupt 0**
		This bit indicates if a scan of analog inputs has been completed since the last read of the STAT register, or since a reset. The EOSI0 bit is cleared by writing one to it. This bit cannot be set by software. EOSI0 is the preferred bit to poll for scan completion if interrupts are not enabled.
		0 Scan is not completed; no end of scan IRQ pending
		1 Scan cycle is completed; end of scan IRQ pending
10	ZCI	**Zero Crossing Interrupt**
		If the respective Offset (OFFSTn) register is configured by having a value greater than 0000h, zero crossing checking is enabled. If the OFFSTn register is programmed with 7FF8h, the result will always be less than, or equal to zero. The ZCI bit is cleared by writing one to all active ZCSn bits.
		0 No ZCI interrupt request
		1 Zero crossing encountered; IRQ pending if ZCIE is set.

Status (STAT) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMT	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Status Register (STAT) Continued

Bits	Name	Description
9	LLMTI	**Low Limit Interrupt**
		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. The LLMTI bit is cleared by writing one to all active LLS*n* bits.
		0 No low limit interrupt request
		1 Low limit exceeded; IRQ pending if 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. The HLMTI bit is cleared by writing one to all active HLS*n* bits.
		0 No high limit interrupt request
		1 High limit exceeded; IRQ pending if HLMTIE is set
7 - 0	RDY*n*	**Ready Sample7-0**
		These bits indicate samples seven through zero are ready to read. These bits are cleared after a read from the respective results register. The RDY*n* bits are set as the individual channel conversions are completed. If polling the RDYn bits to determine if a particular sample is completed care should be taken not to start a new scan until all enabled samples are done.
		0 Sample not ready or was read
		1 Sample ready to read

Status (STAT) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMT	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Conversion Ready (RDY) Register

Bits	Name	Description
15 - 0	RDY*n*	**Ready bits**
		This register provides the current status of the ADC conversions. RDY*n* bit are cleared by reading their corresponding result registers (RSLT). Bits 7-0 are equivalent to the RDY bits in the STAT register. These bits indicate samples 15 through zero are ready to be read. These bits are cleared after a read from the respective RSLT*n* register. The RDY*n* bits are set as the individual channel conversions are completed. If polling the RDY*n* bits to determine if a particular sample is completed care should be taken not to start an new scan until all enabled samples are done.
	0	Sample not ready or was read
	1	Sample ready to read

Conversion Ready (RDY) Register Base + $9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RDY15	RDY14	RDY13	RDY12	RDY11	RDY10	RDY9	RDY8	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Limit Status (LIMSTAT) Register

Bits	Name	Description
15 - 8	HLS*n*	**High Limit Status*n***
		The Limit Status register latches in the result of the comparison between the result of the sample in the RSLT*n* register and the respective Limit register, HILIM*n* and LOLIM*n*.
7 - 0	LLS*n*	**Low Limit Status*n***
		The Limit Status register latches in the result of the comparison between the result of the sample and the respective limit register, HILIM*n* and LOLIM*n*.

Limit Status (LIMSTAT) Register Base + $A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC

Zero Crossing Status (ZXSTAT) Register

Bits	Name	Description
7 - 0	ZCS*n*	**Zero Crossing Status**
		The zero crossing condition is determined by examining the ADC value after it is adjusted by the offset for the RSLT register. Each bit of the register is cleared by writing one to the register bit.
	0	A sign change did not occur in a comparing the current RSLT*n* value and the previous RSLT*n* value, --OR-- Zero crossing control is disabled for sample *n* in the ZXCTRL register.
	1	In a comparison between the current channeln result and the previous channeln result, a sign change occurred as defined in the ZXCTRL register.

Zero Crossing Status (ZXSTAT) Register Base + $B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Result 0-7 (RSLT0–7) Registers

Bits	Name	Description
15	SEXT	**Sign Extend**
		SEXT is the sign-extend bit of the result. When the SEXT bit is set to one it implies a negative result. Set to zero, the implication is a positive result. If only positive results are required, the respective OFFST*n* register must be set to a value of zero.
14 - 3	RSLT*n*	**Digital Result of the Conversion**
		RSLT can be interpreted as either a signed integer or a signed fixed point fractional number. As a fixed point number, the RSLT can be used directly. As a signed integer, one has the option to right shift with sign extend (ASR) three places to fit it into the range [0,4095], Or one can accept the number as presented in the register, knowing there are missing codes because the lower three LSBs are always zero. Negative results (SEXT = 1) are always presented in twos complement format. If it is a requirement of an application, the Result registers always be positive, the Offset register must always be set to zero.
14 - 3	TEST_DATA	**Test Data**
		When the ADC is stopped or in power down mode this field can be written by accessing the register in the memory map. Please see **Section 2.7.10** more information.

Result 0-7 (RSLT0-7) Registers Base + $C - $13	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SEXT	RSLT												0	0	0
	Write		TEST_DATA														
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

Result 8-15 (RSLT8–15) Registers

Bits	Name	Description
14 - 3	RSLT*n*	**Digital Result of the Conversion**
		These eight RSLT registers contain the converted results from a scan. The SAMPLE8 result is loaded into RSLT8, SAMPLE9 result in RSLT9, and so on. In a Parallel Scan mode, the first channel pair designated by SAMPLE8 and SAMPLE12 in register CLST3-4 are stored in RSLT8 and RSLT12, respectively. **Note:** When writing to this register, only the RSLT portion of the value written is used.
14 - 3	TEST_DATA*n*	**Test Data**
		When the ADC is stopped or in power down mode this field can be written by accessing the register in the memory map. Please see **Section 2.7.11** more information.

Result 8-15 (RSLT8-15) Registers Base + $14 - $1B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	RSLT												0	0	0
	Write		TEST_DATA														
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Low and High Limit (LOLIM0–7 & HILIM0-7) Registers

Bits	Name	Description
14 - 3	LLMT	**Low Limit**
		Each ADC sample is compared against the values in the Limit Registers. The comparison is based upon the raw conversion value before the offset correction is applied. Refer to **Figure 2-7**. ADC Limit Registers (LOLIM*n* and HILIM*n*) correspond to Results (RSLT*n*) registers. 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 0x7FF8 for the high limit and 0x0000 for the low limit. At reset, limit checking is disabled.
1 4- 3	HLMT	**High Limit**
		Each ADC sample is compared against the values in the Limit Registers. The comparison is based upon the raw conversion value before the offset correction is applied. Refer to **Figure 2-7**. ADC Limit Registers (LOLIM*n* and HILIM*n*) correspond to Results (RSLT*n*) registers. 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 0x7FF8 for the high limit and 0x0000 for the low limit. At reset, limit checking is disabled.

Low Limit 0-7 (LOLIM0-7) Registers Base + $1C - $23	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	LLMT												0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

High Limit 0-7 (HILIM0-7) Registers Base + $24 - $2B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	HLMT												0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

Offset (OFFST0–7) Registers

Bits	Name	Description
14 - 3	OFFSET	**Offset**
		Value of the Offset (OFFST*n*) register is used to correct the ADC result before it is stored in the RSLT*n* registers. The offset value is subtracted from the ADC result. In order to obtain unsigned results, the respective OFFST*n* register should be programmed with a value of $0000, thus giving a result range of $0000 to $7FF8.

Offset 0-7 (OFFST0-7) Registers Base + $2C - $33	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Power Control (PWR) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	ASB	**Auto Standby**
		The ASB bit selects Auto Standby mode. ASB is ignored if 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 a delay of 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.
		0 Auto standby mode disabled
		1 Auto standby mode enabled
12	PSTS2	**Voltage Reference Power Status 2**
		PSTS2 is a *read-only* bit. It simply reflects whether the voltage reference circuit is currently enabled.
		0 Voltage reference circuit is currently powered up
		1 Voltage reference circuit is currently powered down
11	PSTS1	**Converter B Power Status 1**
		PSTS1 is a *read-only* bit. It is asserted immediately following a write of one to PD1. It is deasserted PUDELAY ADC clock cycles after writing zero to PD1 if APD is 0. This bit can be read as a status bit to determine when the ADC is ready for operation. During Auto power down mode, this bit indicates the current powered state of Converter B.
		0 Converter B is currently powered up
		1 Converter B is currently powered down
10	PSTS0	**Converter A Power Status 0**
		PSTS0 is a *read-only* bit. It is asserted immediately following a write of one to PD0. It is deasserted PUDELAY ADC clock cycles after writing zero to PD0 if APD is 0. This bit can be read as a status bit to determine when the ADC is ready for operation. During Auto power down mode, this bit indicates the current powered state of Converter A.
		0 Converter A is currently powered up
		1 Converter A is currently powered down

Power Control (PWR) Register Base + $34	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ASB	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						APD	PD2	PD1	PD0
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1

Reserved Bits

ADC

Power Control (PWR) Register Continued

Bits	Name	Description
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 PD0 or PD1 to 0) before allowing a scan to start. It also determines the number of ADC clocks of delay provided in auto power down (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. (See **Section 2.7.14.6** for more details.)
3	APD	**Auto Power Down**
		Auto power down mode powers down converters when not in use for a scan. APD takes precedence over ASB. When a scan is started in APD mode, a delay of PUDELAY ADC clock cycles is imposed during which the needed converter(s), if idle, are powered up. (See **Section 2.7.14.7** for more details.)
		0 Auto power down mode is not active
		1 Auto power down mode is active
2	PD2	**Power Down Control for Voltage Reference Circuit 2**
		This bit controls the power down of the ADC's voltage reference current. The voltage reference circuit is shared by both converters. When PD2=1 the voltage reference will be activated whenever PD1 or PD0 are powered up. (See **Section 2.7.14.8** for more details.)
		0 Manually Power Up voltage reference circuit
		1 Power down voltage reference circuit is controlled by PD0 and PD1 (default)
1	PD1	**Manual Power Down for Converter B**
		This bit forces ADC Converter B to power down.
		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.
		0 Power Up ADC Converter A
		1 Power Down ADC Converter A

Power Control (PWR) Register Base + $34	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ASB	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						APD	PD2	PD1	PD0
	Write																
	Reset	0	0	0	1	1	1	0	0	1	1	0	1	0	1	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

Calibration (CAL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	SEL_VREFH_1	**Select V_{REFH} Source 1**
		This bit selects the source of the V_{REFH} reference for all conversions in converter 1.
		0 Internal V_{DDA}
		1 ANB2
14	SEL_VREFLO_1	**Select V_{REFLO} Source 1**
		This bit selects the source of the V_{REFLO} reference for all conversions in converter 1.
		0 Internal V_{SSA}
		1 ANB3
13	SEL_VREFH_0	**Select V_{REFH} Source 0**
		This bit selects the source of the V_{REFH} reference for all conversions in converter 0.
		0 Internal V_{DDA}
		1 ANA2
12	SEL_VREFLO_0	**Select V_{REFLO} Source 0**
		This bit selects the source of the V_{REFLO} reference for all conversions in converter 0.
		0 Internal V_{SSA}
		1 ANA3
3	SEL_TEST_1	**Select Analog BIST Mode Converter1**
		This bit forces the analog input select mux on ANB7 to always be in Pass-Through mode. In this mode additional capacitance from either the internal node or added as an external capacitor can be used to filter the DAC1 output. SEL_DAC1 must have been previously set to one before SEL_TEST_1 can be set. Clearing SEL_DAC1 also clears SEL_TEST_1.
		0 Normal operation
		1 ANB7 input mux is always enabled.

Calibration (CAL) Register Base + $35	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SEL_ VREFH_1	SEL_ VREFLO_1	SEL_ VREFH_0	SEL_ VREFLO_0	0	0	0	0	0	0	0	0	SEL_ TEST_1	SEL_ TEST_0	SEL_ DAC1	SEL_ DAC0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

Calibration (CAL) Register Continued

Bits	Name	Description
2	SEL_TEST_0	**Select Analog BIST Mode Converter 0**
		This bit forces the analog input select mux on ANA7 to always be in Pass-Through mode. In this mode additional capacitance from either the internal node or added as an external capacitor can be used to filter the DAC0 output. SEL_DAC0 must have been previously set to one before SEL_TEST_0 can be set. Clearing SEL_DAC0 also clears SEL_TEST_0.
		0 Normal Operation
		1 ANA7 input mux is always enabled
1	SEL_DAC1	**Select Digital-to-Analog Converter1 (DAC) Alternate Source**
		This bit selects the source of the ADCB7 input as being either the input pin GPIOC15 or DAC1 output.
		0 Normal Operation (GPIOC15)
		1 ANA7 input is replaced with DAC1 output
0	SEL_DAC0	**Select Digital-to-Analog Converter0 (DAC) Alternate Source**
		This bit selects the source of the ADCA7 input as being either the input pin GPIOC11 or DAC0 output.
		0 Normal Operation (GPIOC11)
		1 ANA7 input is replaced with DAC0 output

Calibration (CAL) Register Base + $35	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SEL_ VREFH_1	SEL_ VREFLO_1	SEL_ VREFH_0	SEL_ VREFLO_0	0	0	0	0	0	0	0	0	SEL_ TEST_1	SEL_ TEST_0	SEL_ DAC1	SEL_ DAC0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

COP

Control (CTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
6	CLKSEL	**Clock Source Select**
		This bit selects the clock source for the counter. Some safety applications require the Watchdog counter to use a different clock source than the system clock. This bit can only be changed when CWP bit is set to zero. It is recommended this bit only be changed when CEN bit is clear. This bit can be changed with CEN set if the MSTR_OSC is coming from the Relaxation Oscillator (i.e., MSTR_OSC and RLX_OSC are the same signal).
		0 MSTR_OSC clocks the counter by default
		1 RLX_OSC clocks the counter
5	CLOREN	**COP Loss of Reference Enable**
		This bit enables the operation of the Loss of Reference counter. This bit can only be changed when CWP bit is set to zero.
		0 COP counter is disabled
		1 COP Loss of Reference counter is enabled by default
4	BYPS	**Bypass Master OSC**
		This bit is intended for factory use only. Setting this bit allows testing of the COP to be speeded up by routing the IPBus clock to the counter instead of the prescaled MSTR_OSC. This bit should not be set during normal operation of the chip. If this bit is used it should only be changed while CEN bit is zero.
		0 Counter uses MSTR_OSC by default
		1 Counter uses IPBus clock
3	CSEN	**COP Stop Mode Enable**
		This bit controls the operation of the counter in Stop mode. This bit can only be changed when CWP bit is set to zero.
		0 COP counter stops in Stop mode by default
		1 COP counter runs in Stop mode if CEN is set to 1

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	CLKSEL	CLOREN	BYPS	CSEN	CWEN	CEN	CWP
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

COP

Control (CTRL) Register Continued

Bits	Name	Description	
2	CWEN	**COP Wait Mode Enable**	
		This bit controls the operation of the counter in Wait mode. This bit can only be changed when CWP bit is set to zero.	
		0	COP counter stops in Wait mode by 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 counter. This bit can only be changed when CWP bit is set to zero. This bit always reads as zero when the chip is in Debug mode.	
		0	COP counter is disabled
		1	COP counter is enabled by default
0	CWN	**COP Write Protect**	
		This bit controls the write protection feature of the Control (CTRL) register and the Time-out (TOUT) register. Once set, this bit can only be cleared by resetting the module.	
		0	CTRL and TOUT can be read and written by default
		1	CTRL and TOUT are *read-only*

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	CLKSEL	CLOREN	BYPS	CSEN	CWEN	CEN	CWP
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

COP

Time-Out (TOUT) Register

Bits	Name	Description
15 - 0	TIMEOUT	**Time-Out Period**
		The value in this register determines the time-out period of the COP counter. TIMEOUT should be written before enabling COP. Once COP is enabled, the recommended procedure for changing TIMEOUT is to disable the COP, write to the TOUT register, then re-enable the COP. This procedure ensures that the new TIMEOUT is loaded into the counter. Alternatively, the 16-bit controller can write to the TOUT register prior to writing the proper service patterns to the CNTR register, thereby causing the counter to reload with the new TIMEOUT value. The COP Counter is not reset by a write to the TOUT register. Changing TIMEOUT while the COP is enabled results in a time-out period differing from the expected value. These bits can be changed only when CWP is set to zero.

Time-Out (TOUT) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TIMEOUT															
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Application: _____ Date: _____

_____ Programmer: _____

COP

Counter (CNTR) Register

Bits	Name	Description
15 - 0	COUNT	**Count *(Read-Only)***
		This is the current value of the COP counter as it counts down from the time-out value to 0. A reset is issued when this count reaches zero.
15 - 0	SERVICE	**Service (*Write-Only)***
		When enabled, the COP requires a service sequence be performed periodically in order to clear its counter and prevent a reset from being issued. This routine consists of writing $5555 to the CNTR register followed by writing $AAAA before the time-out period expires. The writes to the CNTR register 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.

Counter (CNTR) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COUNT															
	Write	SERVICE															
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Application: _____ Date: _____

Programmer: _____

I²C

Control (CTRL) Register

Bits	Name	Description
6	SLVDIS	**Slave Disable**
		This bit controls whether I²C has its slave disabled. If this bit is set, the slave is disabled) I²C module functions only as a master and does not perform any action requiring a slave.
		0 Slave is enabled
		1 Slave is disabled
5	RSTEN	**Repeated START Enable**
		This bit determines whether repeated START conditions may be sent when acting as a master. Some older slaves do not support handling repeated START conditions; however, repeated START conditions are used in several I²C module operations. (See **Section 4.6.1.3** for more details.)
		0 Disable
		1 Enable
4	ADDRMST	**Address Mode Master**
		This bit is a *read-only* copy of the Address Master on the Target Address Register (TAR).
		0 7-bit addressing
		1 10-bit addressing
3	ADDRSLV	**Address Mode Slave**
		When acting as a slave, this bit controls whether the I²C module responds to 7- or 10-bit addresses.
		0 7-bit addressing (See **Section 4.6.1.5** for more detail.)
		1 10-bit addressing (See **Section 4.6.1.5** for more detail.)
2 - 1	SPD	**Speed**
		This bit field controls the speed the I²C module operates; its setting is relevant only if one is operating the I²C module in Master mode. Hardware protects against illegal values being programmed by software. This register should be programmed only with a value in the range of one to two; otherwise hardware updates this register with the value of two.
		0 Standard mode (0-100k/bs)
		1 Fast mode (0-400k/bs)
0	MSTEN	**Master Enable**
		This bit controls whether the I²C module master is enabled.
		0 Master disabled
		1 Master enabled

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	8	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	SLVDIS	RSTEN	ADDRMST	ADDRSLV	SPD		MSTEN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

Reserved Bits

I²C

Target Address Register (TAR)

Bits	Name	Description
12	ADDRMST	**Address Mode Master**
		This bit controls whether the I²C module begins its transfers in 7- or 10-bit addressing mode when acting as a master.
		0 — 7-bit addressing
		1 — 10-bit addressing
11	SPCL	**Special**
		This bit indicates whether software performs a General Call or START byte command.
		0 — Ignore bit 10 (GC/STRT) and use TAR normally
		1 — Perform special I²C command as specified in GC/STRT bit
10	GCSTRT	**General Call or START**
		If the SPCL bit is set to one, it indicates whether a General Call or START byte command is to be performed by the I²C module. When issuing a General Call, only writes may be performed to the TXFIFO bit. Attempting to issue a read command results in setting TXABRT bit on the RISTAT register. The I²C module remains in General Call mode until the SPCL bit value is cleared, or until the GC/STRT bit value is set.
		0 — General Call address
		1 — START byte
9 - 0	TA	**Target Address**
		This is the target address for any master transaction. This bit field is ignored when transmitting a General Call. If TA and SA bits are the same, loopback exists but the FIFOs are shared between master and slave, therefore full loopback is not feasible. Only one direction (simplex) Loopback mode is supported, consisting of Master mode transmit and Slave mode receive.

Target Address Register (TAR) Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	ADDRMST	SPCL	GCSTRT	TA									
	Write																
	Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Slave Address Register (SAR)

Bits	Name	Description
9 - 0	SA	**Slave Address**
		The SA bit field holds the slave address when the I²C is operating as a slave. For 7-bit addressing only the SA bit field is used. This register can be written only when the I²C interface is disabled. Writes at other times have no effect. The programmed value cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7F. These addresses should be obtained from Philips as part of the licensing agreement. Philips will assign a slave address. The correct operation of the device is not guaranteed if registers SAR or TAR are programmed to a reserved value.

Slave Address Register (SAR) Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	SA									
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

I²C

Data Buffer and Command (DATA) Register

Bits	Name	Description	
8	CMD	**Command**	
		When writing to the DATA register, this bit controls whether a read or a write is performed.	
		0	Write
		1	Read
7 - 0	DAT	**Data**	
		This register contains the data to be transmitted or received on the I²C bus. If writing a read command to this register, the DAT bit field is ignored by the I²C module. However, when this register is read, this bit field returns the value of data received on the I²C module interface.	

Data Buffer & Command (DATA) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	CMD	DAT							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Standard Speed I²C Clock SCL High Count (SSHCNT) Register

Bits	Name	Description
15 - 0	SSHCNT	**Standard Speed SCL High Count**
		To ensure proper I/O timing, this register must be set before any standard speed I²C bus transaction can take place. This register sets the SCL clock high-period count for standard speed. Table 4-4 provides recommended SSHCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I²C interface is disabled corresponding to the ENBL register being set to zero. Writes at other times have no effect. The minimum valid value is six; hardware prevents values less than six being written. If the CPU attempts to write a value less than six, a value of six results.

Standard Speed Clock High Count (SSHCNT) Reg. Base + $A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SSHCNT															
	Write																
	Reset	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0

Reserved Bits

I²C

Standard Speed Clock SCL Low Count (SSLCNT) Register

Bits	Name	Description
15 - 0	SSLCNT	**Standard Speed SCL Low Count**
		To ensure proper I/O timing, this register must be set before any standard speed I²C bus transaction can take place. This register sets the SCL clock low period count for standard speed. Table 4-5 provides recommended SSLCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I²C interface is disabled, corresponding to the ENBL register being set to zero. Writes at other times have no effect. The minimum valid value is eight; hardware prevents values less than eight being written. If the CPU attempts to write a value less than eight, a value of eight results.

Standard Speed Low Count (SSLCNT) Reg. Base + $C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SSLCNT															
	Write																
	Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Fast Speed Clock SCL High Count (FSHCNT) Register

Bits	Name	Description
15 - 0	FSHCNT	**Fast Speed High Count**
		To ensure proper I/O timing, this register must be set before any fast speed I²C bus transaction can take place. This register sets the SCL clock high-period count for fast speed. Table 4-7 provides recommended FSHCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I²C interface is disabled, corresponding to the ENBL register being set to zero. Writes at other times have no effect. The minimum valid value is six; hardware prevents values less than six being written. If the CPU attempts to write a value less than eight, a value of eight results.

Fast Speed High Count (FSHCNT) Register Base + $E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FSHCNT															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Reserved Bits

I²C

Fast Speed Clock SCL Low Count (FSLCNT) Register

Bits	Name	Description
15 - 0	F_LCNT	**Fast Speed Low Count**
		To ensure proper I/O timing, this register must be set before any fast speed I²C bus transaction can take place. This register sets the SCL clock low period count for fast speed. Table 4-7 provides recommended FSLCNT values. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I²C interface is disabled, corresponding with the ENBL register being set to zero. Writes at other times have no effect. The minimum valid value is eight; hardware prevents values less than eight being written. If the CPU attempts to write a value less than eight, a value of eight results.

Fast Speed Low Count (FSLCNT) Register Base + $10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FSLCNT															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Interrupt Status (ISTAT) Register

Please see the following page for continuation of this register

Bits	Name	Description
11	GC	**General Call**
		Set only when a General Call address is received and it is acknowledged. It remains set until it is cleared either by disabling the I²C module or when the CPU reads bit zero of the CLRGC register. I²C module stores the received data in the RX buffer.
10	STDET	**Start Detect**
		Indicates whether a START or RESTART condition occurred on the I²C interface regardless whether I²C module is operating in Slave or Master mode.
9	STPDET	**Stop Detect**
		Indicates whether a STOP condition occurred on the I²C interface regardless whether I²C module is operating in Slave or Master mode.
8	ACT	**Activity**
		This bit captures I²C module activity and remains set until it is cleared. There are four way to clear the bit. 1. Disable the I²C module 2. Read the CLRACT register 3. Read the CLRINTR register 4. System reset Once this bit is set, it remains set unless one of the above four methods is used to clear it. Even if the I²C module is idle, this bit remains set until cleared, indicating there was activity on the bus.
7	TXDONE	**Transmit Done**
		When the I²C module is acting as a slave-transmitter, this bit is set to one if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating the transmission is completed.

Interrupt Status (ISTAT) Register Base + $16	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	STDET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Interrupt Status (ISTAT) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
6	TXABRT	**Transmit Abort**
		This bit is set to one when the I²C module acting as a master is unable to complete a command the processor sent. (Please refer to Section 4.6.9.7 for conditions to set TXABRT.)
5	RDREQ	**Read Required**
		This bit is set to one when I²C module is acting as a slave and another I²C master is attempting to read data from I²C module. The I²C module holds the I²C bus in a wait state (SCL=0) until this interrupt is serviced, meaning the slave was addressed by a remote master asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the DATA register. This bit is set to zero just after the required data is written to the DATA register.
4	TXEMPTY	**Transmit Empty**
		This bit is set to one when the transmit buffer is at, or below the threshold value set in the Transmit FIFO Threshold (TXFT) register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the ENABLE bit in the ENBL register is zero, the TXFT register is flushed and held in reset. There the TXFT register looks like it has no data within it, so this bit is set to one provided there is activity in the master or slave state machines. When there is no longer activity with ENABLE=0, the TXEMPTY bit is set to zero.

Interrupt Status (ISTAT) Register Base + $16	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	STDET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Interrupt Status (ISTAT) Register Continued

Bits	Name	Description
3	TXOVR	**Transmit Over**
		This bit is set during transmit if the transmit buffer is filled to transmit buffer depth and the processor attempts to issue another I²C command by writing to the DATA register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when the ENABLE bit goes to zero, this interrupt is cleared.
2	RXFULL	**Receive Full**
		This bit is set when the receive buffer reaches, or goes above the receive threshold level in the Receive FIFO Threshold (RXFT) register. It is automatically cleared by hardware when the buffer level goes below the threshold. If the module is disabled (ENABLE=0), the RXFIFO is flushed and held in reset; therefore the RXFIFO is not full. So this bit is cleared once the ENABLE bit in the ENBL register is programmed with a zero, regardless of the continuing activity.
1	RXOVR	**Receive Over**
		This bit is set if the receive buffer is completely filled to receive buffer depth and an additional byte is received from an external I²C device. The ²C module acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (ENABLE=0), this bit keeps its level until the master or slave state machines go into idle, and when the ENABLE bit in the ENBL register goes to zero, this interrupt is cleared.
0	RXUND	**Receive Under**
		This bit is set if the processor attempts to read the receive buffer when it is empty by reading from the DATA register. If the module is disabled (ENABLE=0), this bit keeps its level until the master or slave state machines go into idle and when ENABLE bit in the ENBL register goes to zero, this interrupt is cleared.

Interrupt Status (ISTAT) Register Base + $16	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	STDET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Interrupt Enable (IENBL) Register

Please see the following page for continuation of this register

Bits	Name	Description
11	GC	**General Call**
		Set only when a General Call address is received and it is acknowledged. It remains set until it is cleared either by disabling the I²C module or when the CPU reads bit zero of the CLRGC register. I²C module stores the received data in the RX buffer.
10	STDET	**Start Detect**
		Indicates whether a START or RESTART condition occurred on the I²C interface regardless whether I²C module is operating in Slave or Master mode.
9	STPDET	**Stop Detect**
		Indicates whether a STOP condition occurred on the I²C interface regardless whether I²C module is operating in Slave or Master mode.
8	ACT	**Activity**
		This bit captures I²C module activity and remains set until it is cleared. There are four way to clear the bit. 1. Disable the I²C module 2. Read the CLRACT register 3. Read the CLRINTR register 4. System reset Once this bit is set, it remains set unless one of the above four methods is used to clear it. Even if the I²C module is idle, this bit remains set until cleared, indicating there was activity on the bus.
7	TXDONE	**Transmit Done**
		When the I²C module is acting as a slave-transmitter, this bit is set to one if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating the transmission is completed.

Interrupt Enable (IENBL) Register Base + $18	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Interrupt Enable (IENBL) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
6	TXABRT	**Transmit Abort**
		This bit is set to one when the I²C module acting as a master is unable to complete a command the processor sent. (Please refer to Section 4.6.9.7 for conditions to set TXABRT.)
5	RDREQ	**Read Required**
		This bit is set to one when I²C module is acting as a slave and another I²C master is attempting to read data from I²C module. The I²C module holds the I²C bus in a wait state (SCL=0) until this interrupt is serviced, meaning the slave was addressed by a remote master asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the DATA register. This bit is set to zero just after the required data is written to the DATA register.
4	TXEMPTY	**Transmit Empty**
		This bit is set to one when the transmit buffer is at, or below the threshold value set in the Transmit FIFO Threshold (TXFT) register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the ENABLE bit in the ENBL register is zero, the TXFT register is flushed and held in reset. There the TXFT register looks like it has no data within it, so this bit is set to one provided there is activity in the master or slave state machines. When there is no longer activity with ENABLE=0, the TXEMPTY bit is set to zero.
3	TXOVR	**Transmit Over**
		This bit is set during transmit if the transmit buffer is filled to transmit buffer depth and the processor attempts to issue another I²C command by writing to the DATA register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when the ENABLE bit goes to zero, this interrupt is cleared.

Interrupt Mask (IENBL) Register Base + $18	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Interrupt Enable (IENBL) Register Continued

Bits	Name	Description
2	RXFULL	**Receive Full**
		This bit is set when the receive buffer reaches, or goes above the receive threshold level in the Receive FIFO Threshold (RXFT) register. It is automatically cleared by hardware when the buffer level goes below the threshold. If the module is disabled (ENABLE=0), the RXFIFO is flushed and held in reset; therefore the RXFIFO is not full. So this bit is cleared once the ENABLE bit in the ENBL register is programmed with a zero, regardless of the continuing activity.
1	RXOVR	**Receive Over**
		This bit is set if the receive buffer is completely filled to receive buffer depth and an additional byte is received from an external I²C device. The I2C module acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (ENABLE=0), this bit keeps its level until the master or slave state machines go into idle, and when the ENABLE bit in the ENBL register goes to zero, this interrupt is cleared.
0	RXUND	**Receive Under**
		This bit is set if the processor attempts to read the receive buffer when it is empty by reading from the DATA register. If the module is disabled (ENABLE=0), this bit keeps its level until the master or slave state machines go into idle and when ENABLE bit in the ENBL register goes to zero, this interrupt is cleared.

Interrupt Mask (IMASK) Register Base + $18	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Raw Interrupt Status (RISTAT) Register

Please see the following page for continuation of this register

Unlike the ISTAT register, these bits are not masked, so they always show the true status of the I²C module.

Bits	Name	Description
11	GC	**General Call**
		This bit is set only when a General Call (GC) address is received and it is acknowledged. It remains set until it is cleared either by a CPU read of the Clear General Call Interrupt (CLRGC) register, or a CPU read of the Clear Combined Individual Interrupts (CLRINT) register.
10	STDET	**Start Detect**
		This bit indicates a START or repeated START condition occurred on the I²C interface regardless whether the I²C module is operating in Slave or Master mode. It remains set until it is cleared either by a CPU read of the Clear Start Detect Interrupt (CLRSTDET) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINT) register.
9	STPDET	**Stop Detect**
		This bit indicates a STOP condition occurred on the I²C interface regardless whether the I²C module is operating in Slave or Master mode. It remains set until it is cleared either by a CPU read of the Clear Stop Detect Interrupt (CLRSTPDET) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINT) register.
8	ACT	**Activity**
		This bit is set by hardware and cleared by software. It is set to one when the bus transitions from an idle state to a busy state. It remains set until it is cleared either by a CPU read of the Clear Activity Interrupt (CLRACT) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINT) register.
7	TXDONE	**Transmit Done**
		When the I²C module is acting as a slave-transmitter, this bit is set to one if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating the transmission is done. It remains set until it is cleared either by a CPU read of the Clear Receive Done Interrupt (CLRRXDONE) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINT) register.

Raw Interrupt Status (RISTAT) Register Base + $1A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Raw Interrupt Status (RISTAT) Register Continued

Please see the following page for continuation of this register

Unlike the ISTAT register, these bits are not masked, so they always show the true status of the I²C module.

Bits	Name	Description
6	TXABRT	**Transmit Abort**
		This bit is set to one when the I²C module is unable to complete a command the processor requested. It remains set until it is cleared either by a CPU read of the Clear Transmit Abort Interrupt (CLRTXABRT) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINT) register. (See **Section 4.6.11.7** for conditions to set TXABRT.)
5	RDREQ	**Read Request**
		This bit is set to one when a remote I²C master is attempting to read data from the I²C module. The I²C module operates as a slave-receiver. The I²C module holds the I²C bus in a wait state (SCL=0) until the processor writes data to the TXFIFO. The processor services the read request by writing data to the DATA register. It remains set until it is cleared either by a CPU read of the Clear Read Request (CLRRDREQ) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINT) register.
4	TXEMPTY	**Transmit Empty**
		This bit is set to one when the transmit buffer is at, or below the threshold value set in the Transmit FIFO Threshold (TXFT) register. It is automatically cleared by hardware when the buffer level goes above the threshold.
3	TXOVR	**Transmit Over**
		Set during transmit if the transmit buffer is filled to transmit buffer depth and the processor attempts to issue another I²C command by writing to the DATA register. It remains set until it is cleared either by a CPU read of the Clear Transmit Over (CLRTXOVR) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINT) register.

Raw Interrupt Status (RISTAT) Register Base + $1A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Raw Interrupt Status (RISTAT) Register Continued

Unlike the ISTAT register, these bits are not masked, so they always show the true status of the I²C module.

Bits	Name	Description
2	RXFULL	**Receive Full**
		This bit is set when the receive buffer reaches, or goes above the RXTL in the Receive FIFO Threshold (RXFT) register. It is automatically cleared by hardware when the buffer level goes below the threshold.
1	RXOVR	**Receive Over**
		This bit is set if the receive buffer is completely filled to receive buffer depth and an additional data byte is received from a remote I²C device. The I²C module acknowledges this additional data byte, but any data bytes received after the FIFO is full are lost. It remains set until it is cleared either by a CPU read of the Clear Receive Over Interrupt (CLRRXOVR) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINTR) register.
0	RXUND	**Receive Under**
		This bit is set if the processor attempts to read the receive buffer when it is empty by reading from the DATA register. It remains set until it is cleared either by a CPU read of the Clear Receive Under Interrupt (CLRRXUND) register, or a CPU read of the Clear Combined and Individual Interrupts (CLRINTR) register.

Raw Interrupt Status (RISTAT) Register Base + $1A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	GC	ST DET	STP DET	ACT	TX DONE	TX ABRT	RD REQ	TX EMPTY	TX OVR	RX FULL	RX OVR	RX UND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Receive FIFO Threshed (RXFT) Register

Bits	Name	Description
7 - 0	RXTL	**Receive FIFO Threshold Level**
		Receive FIFO Threshold Level controls the level of entries (or above) triggering the RXFULL bit in the RISTAT register. The valid range is 0-255, with the additional restriction hardware does not allow this threshold level to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set is the maximum depth of the buffer minus one. A value of zero sets the threshold for one entry, and a value of 255 sets the threshold for 256 entries.

Receive FIFO Threshold (RXFT) Register Base + $1C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXTL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Transmit FIFO Threshold (TXFT) Register

Bits	Name	Description
7 - 0	TXTL	**Transmit FIFO Threshold Level**
		Transmit FIFO Threshold Level controls the level of entries (or below) triggering the TXEMPTY bit in the RISTAT register. The valid range is 0-255, with the additional restriction it may not be set to a threshold level larger than the depth of the buffer. If an attempt is made to do that, the actual value set is the maximum depth of the buffer minus one. A value of zero sets the threshold for zero entries, and a value of 255 sets the threshold for 255 entries.

Transmit FIFO Threshold (TXFT) Register Base + $1E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXTL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Clear Combined & Individual Interrupt (CLRINT) Register

Bits	Name	Description
0	INT	**Clear Interrupt**
		This *read-only* bit is read to clear the combined interrupt, all individual interrupts, and the TXABRTSRC register. This bit clears only software interrupts capable of being cleared. Refer to bit nine on the TXABRTSRC register for an exception to clearing the TXABRTSRC register.

Clear Interrupt (CLRINT) Register Base + $20	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Clear Receive Under Interrupt (CLRRXUND) Register

Bits	Name	Description
0	RXUND	**Clear Receive Under**
		This *read-only* bit is read to clear the RXUNDER interrupt bit on the RISTAT register.

Clear Receive Under Interrupt (CLRRXUND) Base + $22	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXUND
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

I²C

Clear Receive Over Interrupt (CLRRXOVR) Register

Bits	Name	Description
0	RXOVR	**Clear Receive Over**
		This *read-only* bit is read to clear the RXOVER interrupt bit on the RISTAT register.

Clear Receive Over Interrupt (CLRRXOVR) Base + $24	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RXOVR
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Clear Transmit Over Interrupt (CLRTXOVR) Register

Bits	Name	Description
0	TXOVR	**Clear Transmit Over**
		This *read-only* bit is read to clear the TXOVER interrupt bit on the RISTAT register.

Clear Transmit Over Interrupt (CLRRXOVR) Base + $26	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXOVR
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

I²C

Clear Read Required Interrupt (CLRRDREQ) Register

Bits	Name	Description
0	RDREQ	**Read Required**
		This *read-only* bit is read to clear the RDREQ interrupt bit on the RISTAT register.

Clear Read Req'd. Interrupt (CLRRDREQ) Base + $28	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RDREQ
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Clear Transmit Abort Interrupt (CLRTXABRT) Register

Bits	Name	Description
0	TXABRT	**Clear Transmit Abort Interrupt**
		This *read-only* bit is read to clear the TXABRT interrupt bit on the RISTAT and TXABRTSRC registers. Refer to bit nine on the TXABRTSRC register for an exception to clearing that register.

Clear Transmit Abort Interrupt (CLRTXABRT) Base + $2A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXABRT
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Clear Transmit Done Interrupt (CLRTXDONE) Register

Bits	Name	Description
0	TXDONE	**Clear Transmit Done Interrupt**
		This *read-only* bit is read to clear the RXDONE interrupt bit on the RISTAT register.

Clear Receive Done Interrupt (CLRTXDONE) Base + $2C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXDONE
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Clear Activity Interrupt (CLRACT) Register

Bits	Name	Description
0	ACT	**Clear Activity Interrupt**
		This *read-only* bit is read to clear the ACT interrupt bit on the RISTAT register if the I²C is no longer active. If the I²C module is still active on the bus, the ACT interrupt bit remains set.

Clear Activity Interrupt (CLRACT) Reg. Base + $2E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ACT
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Clear Stop Detect Interrupt (CLRSTPDET) Register

Bits	Name	Description
0	STPDET	**Clear Stop Detect Interrupt**
		This *read-only* bit is read to clear the STP_DET interrupt bit on the RISTAT register.

Clear Stop Detect Interrupt (CLRSTPDET) Base + $30	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STPDET
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Clear Start Detect Interrupt (CLRSTDET) Register

Bits	Name	Description
0	STDET	**Clear Start Detect Interrupt**
		This *read-only* bit is read to clear the STRT_DET interrupt bit on the RISTAT register.

Clear Start Detect Interrupt (CLRSTDET) Base + $32	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STDET
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

I²C

Clear General Call Interrupt (CLRGC) Register

Bits	Name	Description
0	GC	**Clear General Call Interrupt**
		This *read-only* bit is read to clear the GC interrupt bit on the RISTAT register.

Clear General Call Interrupt (CLRGC) Base + $34	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GC
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Enable (ENBL) Register

Bits	Name	Description
0	EN	**Enable**
		This bit controls when the I²C module is enabled.
		0 Disables I²C module
		1 Enables I²C module
		Software can disable the ²C module while it is active. The ACT interrupt bit on the RISTAT register can be polled to determine if the I²C module is active. When I²C module is disabled, the following occurs: • TXFIFO and RXFIFO get flushed • Interrupt bits in the RISTAT register are cleared • Status bits in the ISTAT register remain active until I²C module goes into idle state. If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the I²C module stops the current transfer at the end of the current byte and does not acknowledge the transfer.

Enable (ENBL) Register Base + $36	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Status (STAT) Register

Please see the following page for continuation of this register

Bits	Name	Description
6	SLVACT	**Slave FSM Activity Status**
		This bit is set when the Slave Finite State Machine (FSM) is not in the idle state.
	0	Slave FSM is in idle state so the slave part of I²C module is not active
	1	Slave FSM is not in idle state so the slave part of I²C module is active
5	MSTACT	**Master FSM Activity Status**
		This bit is set when the Master Finite State Machine (FSM) is not in the idle state.
	0	Master FSM is in idle state so the master part of I²C module is not active
	1	Master FSM is not in idle state so the master part of I²C module is active
4	RFF	**Receive FIFO Completely Full**
		This bit is set when the receive FIFO is completely full. It is cleared when the receive FIFO contains one or more empty location.
	0	Receive FIFO is not full
	1	Receive FIFO is full
3	RFNE	**Receive FIFO Not Empty**
		This bit is set when the receive FIFO contains one or more entries, and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO.
	0	Receive FIFO is empty
	1	Receive FIFO is not empty

Status (STAT) Register Base + $38	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	SLVACT	MSTACT	RFF	RFNE	TFE	TFNF	ACT
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Status (STAT) Register Continued

Bits	Name	Description
2	TFE	**Transmit FIFO Completely Empty**
		This bit is set when the transmit FIFO is completely empty. It is cleared when it contains one or more valid entries.
		0 Transmit FIFO is not empty
		1 Transmit FIFO is empty
1	TFNF	**Transmit FIFO Not Full**
		This bit is set when the transmit FIFO contains one or more empty locations. It is cleared when the FIFO is full. This bit does not request an interrupt.
		0 Transmit FIFO is full
		1 Transmit FIFO is not full
0	ACT	**Activity Status**
		I²C Activity Status

Status (STAT) Register Base + $38	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	SLVACT	MSTACT	RFF	RFNE	TFE	TFNF	ACT
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Reserved Bits

I²C

Transmit FIFO Level Register (TXFLR)

Bits	Name	Description
2 - 0	TXFL	**Transmit FIFO Level**
		This bit field contains the number of valid data entries in the TXFIFO.

Transmit FIFO Level (TXFLR) Register Base + $3A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	TXFL		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Receive FIFO Level Register (RXFLR)

Bits	Name	Description
2 - 0	RXFL	**Receive FIFO Level**
		This bit field contains the number of valid data entries in the RXFIFO.

Receive FIFO Level (RXFLR) Register Base + $3C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	RXFL		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

I²C

Transmit Abort Source (TXABRTSRC) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	SLVRD	**Abort Slave Read in Transmit**
		The I²C was expecting the processor to service a read request by writing a write command to the DATA register (DATA[8]=0), but the processor wrote a read command to the DATA register (DATA[8]=1).
14	SLVAL	**Slave Arbitration Lost**
		Slave lost arbitration while transmitting data to a remote master. Bit 12 in the TXABRTSRC register is set at the same time.
13	SLVFLSH	**Abort Slave Slush Transmit FIFO**
		Slave received a read command but data still exists in the TXFIFO, so the slave issues a TXABRT interrupt to flush old data in TXFIFO.
12	AL	**Arbitration Lost**
		This bit indicates an arbitration loss occurred. If bits 12 and 14 are set, the slave loses arbitration. However, if bit 12 is set and bit 14 is clear, the master loses arbitration.
11	MSTDIS	**Abort Master Disabled**
		The processor attempts a Master mode command, but the Master mode is disabled.
10	RNORST	**Abort 10B Read No Repeated Start**
		The repeated START is disabled (RSTRT_EN bit (CTRL[5]=0) and the processor attempts a read command in Master 10-bit addressing mode.
9	SNORST	**Abort Start Byte No Repeated Start**
		The repeated START is disabled (RSTRT_EN) bit (CTRL[5]=0) and the processor attempts to generate a START byte on the bus in Master mode. To clear bit nine, the source of the Abort Start Byte No Restart (SNORSTRT) bit in this register must be fixed first; repeated START must be enabled (CTRL[5]=1), the SPCL bit must be cleared (TAR[11]=0), or the GC/START bit must be cleared (TAR[10]=0). Once the source of the SNORSTRT bit is fixed, this bit can be cleared in the same manner as other bits in this register. However, if the source of the SNORSTRT bit is not fixed before attempting to clear this bit, bit nine clears for one cycle and then becomes reasserted.
7	SACKDET	**Abort Start Byte Acknowledge Detect**
		The I²C sent a START byte in Master mode, and the START byte was acknowledged.

Transmit Abort Source (TXABRTSRC) Base + $4D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SLVRD	SLVAL	SLV FLSH	AL	MST DIS	RNOR ST	SNOR ST	0	SACK DET	0	GC READ	GC NACK	TD NACK	AD2_ NACK	AD1_ NACK	AD7_ NACK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

I²C

Transmit Abort Source (TXABRTSRC) Register Continued

Bits	Name	Description
5	GCREAD	**Abort General Call Read**
		The TAR register was configured for General Call (TAR[11]=1 and TAR[10]=0), but the processor wrote a read command to the DATA register (DATA[8]=1).
4	GCNACK	**Abort General Call No Acknowledge**
		The I²C attempted a General Call transfer in Master mode, but none of the bus slaves acknowledged the General Call address.
3	TDNACK	**Abort Transmit Data No Acknowledge**
		This is a Master mode only bit. Master received an acknowledgement for the address, but did not receive an acknowledgement on one of the transmitted data bytes.
2	AD2NACK	**Abort 10 Address2 No Acknowledge**
		The I²C attempted a 10-bit addressing transfer in Master mode, but none of the bus slaves acknowledged the second address byte of the 10-bit address.
1	AD1NACK	**Abort 10 Address1 No Acknowledge**
		The I²C attempted a 10-bit addressing transfer in Master mode, but none of the bus slaves acknowledged the first address byte of the 10-bit address.
0	AD7NACK	**Abort 7B Address No Acknowledge**
		The I²C attempted a 7-bit addressing transfer in Master mode, but none of the bus slaves acknowledged the address byte of the 7-bit address.

Transmit Abort Source (TXABRTSRC) Base + $4D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SLVRD	SLVAL	SLV FLSH	AL	MST DIS	RNOR ST	SNOR ST	0	SACK DET	0	GC READ	GC NACK	TD NACK	AD2_ NACK	AD1_ NACK	AD7_ NACK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

Control (CTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15 - 14	PLLIE1	**PLL Interrupt Enable 1**
		An optional interrupt can be generated when the PLL Lock (LCK1) status bit in the PLL Status (STAT) register 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 (LCK0) status bit in the PLL Status (STAT) register 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**
		Loss of the reference clock circuit monitors the output of the selected clock source. In the event of reference clock loss, an interrupt can be generated.
		0 Interrupt disabled
		1 Interrupt enabled
7	LCKON	**Lock Detector On**
		0 Lock detector disabled
		1 Lock detector enabled

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	0	0	PLLPD	0	PRECS	ZSRC	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

Control (CTRL) Register Continued

Bits	Name	Description
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, preventing loss of reference clock to the core.
		0 PLL enabled
		1 PLL powered down
2	PRECS	**Prescaler Clock Select**
		This bit is used to select between the external clock source or the internal relaxation oscillator. **Note:** This bit should not be set unless the external reference (CLKIN) is enabled.
		0 Relaxation Oscillator selected (reset value)
		1 External reference selected (either crystal oscillator, or external clock source)
1 - 0	ZSRC	**Clock Source**
		The Clock Source (ZSRC) determines the SYS_CLK_×2 source to the SIM module, generating divided down versions of this signal for use by memories and the IPBus. ZSRC is automatically set to one during Stop mode, or when PLLPD is set, preventing loss of the reference clock to the core. For the 278 family parts, ZSRC may have the following values.
		00 Reserved
		01 MSTR_OSC
		10 Postscaler output
		11 Reserved

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	PRECS	ZSRC	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

Divide-By (DIVBY) Register

Bits	Name	Description
15 - 12	LORTP	**Loss of Reference Clock Timer Period**
		These bits control the amount of time required for the loss of reference clock interrupt to be generated. This failure detection time is LORTP × 10 × PLL-output-clock-time-period. The Loss of Reference Clock Detector block counts FOUT/2 clocks continuously, illustrated in **Figure 5-1** of this manual. The MSTR_OSC clock input resets this counter. If the counter ever exceeds LORTP × 10 the loss of reference clock interrupt is generated.
10 - 8	PLLCOD	**PLL Clock Out Divide or Postscaler**
		The PLL output clock can be divided down by a 2-bit postscaler, defined in **Figure 5-1** in this manual. The output of the postscaler is a selectable clock source for the core as determined by the ZSRC bit in the CTRL register. Legal combinations of PLL settings may be found in **Figure 5-2** also in this manual.
	000	Divide-by 1
	001	Divide-by 2
	010	Divide-by 4
	011	Divide-by 8
	100	Divide-by 16
	101	Divide-by 32
	11x	Divide-by 32

Divide-By (DIVBY) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LORTP				0	PLLCOD			0	0	0	0	0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

Status (STAT) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	LOLI1	**Loss of Lock Interrupt**
		LOLI1 displays the status of the lock detector state from LCK1 circuit. The interrupt is cleared by writing one to LOLI1. This bit will not be set by the hardware if the corresponding CTRL register PLLIE1 bit is cleared or set to zero.
		0 PLL locked
		1 PLL unlocked
14	LOLI0	**Loss of Lock Interrupt 0**
		LOCI shows the status of the reference clock detection circuit. This bit will not set by the hardware if the CTRL register PLLIE0 bit is cleared or set to zero.
		0 PLL locked
		1 PLL unlocked
13	LOCI	**Loss of Reference Clock Interrupt**
		LOCI shows the status of the reference clock detection circuit.
		0 Oscillator clock normal
		1 Loss of oscillator clock detected

Status (STAT) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLLPDN	0	0	ZSRCS	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

Status (STAT) Register Continued

Bits	Name	Description	
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 register.	
		0	PLL not powered down
		1	PLL powered down
1 - 0	ZSRCS	**Clock Source Status**	
		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

Status (STAT) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLLPDN	0	0	ZSRCS	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

Oscillator Control (OCTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	ROPD	**Relaxation Oscillator Power Down**
		This bit powers down the relaxation oscillator. The relaxation oscillator may powered down 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 was changed to the external source by setting the PRECS bit in CTRL register.
		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: Relaxation oscillator output frequency is 8MHz
		1 Standby mode: Relaxation escalator 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 controls the power usage of the crystal oscillator. It is reset to the high power state, allowing either a crystal or resonator to be used.
		0 High power mode: This mode is required when a resonator is used.
		1 Low power mode: This is the desired mode when a crystal is used.

Oscillator Control (OCTRL) Register Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ROPD	ROSB	COHL	CLK_MODE	EXT_SEL		TRIM									
	Write																
	Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1

Reserved Bits

OCCS

Oscillator Control (OCTRL) Register Continued

Bits	Name	Description
12	CLK_MODE	**Crystal Oscillator Clock Mode**
		This bit controls the crystal/resonator clock selection. When Direct Clock mode is selected, this bit also turns off the Crystal Oscillator for power savings.
		0 — Crystal oscillator enabled
		1 — Direct clock mode: Setting this bit shuts down the crystal oscillator and allows an external clock source on the XTAL pin to drive the clock input to the chip directly.
11 - 10	EXT_SEL	**External Clock in Select**
		These bits selects the source of the external clock input.
		00 — Select primary external clock input (GPIOB6)
		01 — Select alternate external clock input (GPIOB5)
		10 — Select Crystal Oscillator (XTAL)
		11 — Select Crystal Oscillator
9 - 0	TRIM	**Internal Relaxation Oscillator Trim**
		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 $200, centering the range of possible adjustment. (A factory calibrated value is available in the HFM_IFROPT_1 register.)

Oscillator Control (OCTRL) Register Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ROPD	ROSB	COHL	CLK_MODE	EXT_SEL		TRIM									
	Write																
	Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

External Clock Check (CLKCHK) Register

Bits	Name	Description
15	CHK_EN	**Check Enable**
		This bit starts and stops the clock checking function. Allow enough time after the CLK_CHK 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 its current state. Reading a low after a check has been started indicates the check operation is complete and the final values are valid in the REF_CNT and TARGET_CNT registers.
		1 Writing one clears the REF_CNT and TARGET_CNT bit fields and begins the clock checking function. The CHK_EN bit remains high while the operation is in progress.
14 - 8	REF_CNT	**Reference Count**
		This bit field provides the number of counted ROSC clock cycles. At the end of a clock check operation this count reads as all lows. (The reference counter has counted through its whole range and rolled over to zero.)
7 - 0	TARGET_CNT	**Target Count**
		This bit field provides the number of counted external clock cycles.

External Clock Check (CLKCHK) Register Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CHK_EN	REF_CNT							TARGET_CNT							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

OCCS

Protection (PROT) Register

Bits	Name	Description	
5 - 4	**FRQEP**	**Frequency Enable Protection**	
		Enables write protection of the PLLCOD and ZSRCS bits	
		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 bits	
		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 and LORTP bits. By write protecting these bits (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

Protection (PROT) Register Base + $7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	FRQEP		OSCEP		PLLEP	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

FLASH

Clock Divider (CLKDIV) Register

Bits	Name	Description
7	DIVLD	**Clock Divider Loaded**
		This is a *status-only* bit. Writing has no effect.
		0 Register has not been written
		1 Register has received writing since the last reset
6	PRDIV8	**Enable Prescaler By 8**
		This is a read and write bit.
		0 Prescaler divides oscillator clock by 1
		1 Prescaler divides oscillator clock by 8
5 - 0	DIV	**Clock Divider**
		The Clock Divider register bits PRDIV8 and DIV must be set with appropriate values before programming or erasing the FM array. Because FCLK is re-timed into the system clock domain, the values of PRDIV8 and DIV are affected by the system bus frequency as well. Refer to the Functional Description of Writing the CLKDIV register located in **Section 6.5.5.1** for the detailed algorithm for determining the settings for DIV and PRDIV8.

Clock Divider (CLKDIV) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	DIVLD	PRDIV8	DIV					
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

FLASH

Configuration (CNFG) Register

Bits	Name	Description
10	LOCK	**Write Lock Control**
		This bit can always be read. Once set, it can't be cleared except by reset. This bit provides additional security for the Flash array by disabling writes to the protection register.
		0 The PROT register can be modified by writing
		1 The PROT register is write-locked
8	AEIE	**Access Error Interrupt Enable**
		This read/write bit enables an interrupt in case of an error accessing the Flash.
		0 ACCERR interrupt s disabled
		1 An interrupt is requested whenever the ACCERR flag is set
7	CBEIE	**Command Buffer Empty Interrupt Enable**
		This read/write bit enables an interrupt in case of an empty command buffer in the Flash.
		0 Command Buffer Empty interrupts disabled
		1 An interrupt is requested whenever the CBEIF flag is set
6	CCIE	**Command Complete Interrupt Enable**
		This read/write bit enables an interrupt in case of all commands being completed in the Flash.
		0 Command complete interrupts disabled
		1 An interrupt is requested whenever the CCIF flag is set
5	KEYACC	**Enable Security Key Writing**
		This bit can be read; however, it can receive writing if the KEYEN bit in the SECHI register is set.
		0 Flash writes are interpreted as the start of a program or erase sequence, e.g. program or erase
		1 Writes to Flash array are interpreted as keys to open the back door

Configuration (CNFG) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	LOCK	0	AEIE	CBEIE	CCIE	KEYACC	0	0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

FLASH

Security (SECHI and SECLO) Registers

Bits	Name	Description
15	KEYEN	**Enable Back Door Key to Security**
		This bit is *read-only*.
		0 Back door to Flash is disabled
		1 Back door to Flash is enabled
14	SECSTAT	**Security Status**
		This bit is *read-only*.
		0 Flash Security is disabled
		1 Flash Security is enabled

Security High (SECHI) Register Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	KEYEN	SECSTAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Write																
	Reset	F[1]	F[2]	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset state loaded from Flash array during reset.
2. Reset state determined by security state of module.

Bits	Name	Description
15	SEC	**Security**
		This *read-only* register. Value loaded into The Security (SEC) bits from the configuration field at reset in turn determines the state of Flash Security at reset (SECSTAT). **Table 6-7** in this manual outlines the single code enabling the security feature in the Flash Memory.

Security Low (SECLO) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEC	
	Write																
	Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

FLASH

Protection (PROT) Register

Bits	Name	Description
15 - 0	PROTECT	**Protection Register**
		This register may always be read, but may be modified by writing only when LOCK=0. This register value is reset to the PROT_VALUE in the PROT register; however, this is only possible if the LOCK bit in CNFG is zero. To change the Flash protection loaded on reset, the sector [15] of program Flash must first be unprotected as just described above, then the protection word in the Configuration Field at addresses defined in **Table 6-1** in this manual must be programmed with the desired value.

Protection (PROT) Register Base + $10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PROTECT															
	Write																
	Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Application: _____ Date: _____

Programmer: _____

FLASH

User Status (USTAT) Register

Please see the following page for continuation of this register

Bits	Name	Description	
7	CBEIF	**Command Buffer Empty Interrupt Flag**	
		The CBEIF flag indicates the address, data, and command buffers are empty allowing a new command sequence to be started. The CBEIF flag is cleared by writing one. Clearing the flag results in the CMD, DATA, and ADDR registers being transferred to the internal state machine for launch of a command. Writing zero has no effect on CBEIF, but it can be used to abort a command sequence. The CBEIF bit can generate an interrupt if the CBEIE bit in the CNFG register is set. While the CBEIF flag is clear the CMD, DATA, and ADDR registers cannot be modified by writing.	
		0	Buffers are full
		1	Buffers are ready to accept a new command
6	CCIF	**Command Complete Interrupt Flag**	
		The CCIF flag indicates there are no other pending commands. The CCIF flag is set and cleared automatically upon start and completion of a command. Writing to CCIF has no effect. The CCIF bit can generate an interrupt if the CCIE bit in the CNFG 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 one. Writing zero has no effect on PVIOL. While the PVIOL flag is set, it is not possible to launch another command.	
		0	No Failure
		1	A protection violation has occurred

User Status (USTAT) Register Base + $13	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

FLASH

User Status (USTAT) Register Continued

Bits	Name	Description
4	ACCERR	**Access Error**
		The ACCERR flag indicates an illegal access to the FM array or registers caused by a bad program or erase sequence. The ACCERR flag is cleared by writing one. Writing zero to ACCERR bit has no effect. While the ACCERR flag is set, it is not possible to launch another command. The ACCERR relates to FM array writes from the 56F800E core buses and will not be set by writing directly to the data and address registers from the IPBuses. Please see **Section 6.5.5.3** in this manual to set ACCERR flag details.
		0 No failure
		1 Access error has occurred
2	BLANK	**Flash Blank Verified Erased**
		The BLANK flag indicates an Erase Verify command (RDARY1) checked the Flash block and found it to be blank. The BLANK flag is cleared by writing one. Writing zero has no effect.
		0 If an Erase Verify command is requested, and the CCIF flag is set, a zero in BLANK indicates the block is not erased.
		1 Flash block verifies as erased

User Status (USTAT) Register Base + $13	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

FLASH

Command (CMD) Register

Bits	Name	Description
6 - 0	COMMAND	**Command**
		Valid User mode commands are shown in **Table 6-7** in this manual. Writing a command in User mode other than those listed in **Table 6-7** will cause the ACCERR flag in the USTAT register to set

Command (CMD) Register Base + $14	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	COMMAND						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

FLASH

Data (DATA) Register

Bits	Name	Description
15 - 0	DATA	**Data Buffer**
		The results of the Calculate Data Signature and Calculate IFR Block Signature commands are available in this register after execution of these commands.

Data (DATA) Register Base + $18	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DATA															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

FLASH

Option Data 1 (OPT1) Register

Bits	Name	Description
15 - 0	OPT1	**Information Flash Row Option 1**
		All bits can be read in user and test modes; however, the OPT1 register cannot be written in any mode. The OPT1 register is loaded at reset with the value ifr_opt1 from physical address $xx31 within the Flash Information Row. Function and use of OPT1 bits is defined by system integrator. (56F80xx stores the factory determined ROSC trim value in the OPT1 register.)

Option Data 1 (OPT1) Register Base + $1B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	OPT1															
	Write																
	Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

FLASH

Test Array Signature (TSTSIG) Register

Bits	Name	Description
15 - 0	TEST SIGNATURE	**Test Signature**
		The TSTSIG register stores the Flash Information Block Signature, generated by the Calculate IFR Block Signature command during factory test. The value in the TSTSIG register is compared to the result of the Calculate IFR Block Signature throughout the life of the part to confirm data used by the user commands and internal module adjustments has not been compromised.

Test Array Signature (TSTSIG) Base + $1D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TEST SIGNATURE															
	Write																
	Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1. Reset state loaded from Flash array during reset.

Reserved Bits

Application: _____ Date: _____

Programmer: _____

GPIO

Pull-Up Enable (PUPEN) Register

Bits	Name	Description	
15 - 0	PU	**Pull-Up Enable**	
		Table 7-2 provides the state of the pin and Pull-Up register as a function of current peripheral output state and control register values.	
		0	Pull-Up is disabled
		1	Pull-Up is enabled (See Table 7-2 for additional conditions.)

Pull-Up Enable (PUPEN) Register Base + $0	Bits	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

Application: _____ Date: _____

_____ Programmer: _____

GPIO

Data (DATA) Register

Bits	Name	Description
15 - 0	D	Data
		This read/write register holds data coming either from the pin or the IPBus when pins are configured as GPIO. When pins are configured as GPIO and input, logic level on the pin can be determined by reading the DATA register value. When pins are configured as GPIO and output, value written to the DATA register is seen at the pin.

Data (DATA) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	D															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

GPIO

Data Direction (DDIR) Register

Bits	Name	Description
15 - 0	DD	**Data Direction**
		When the pin is configured as GPIO mode (PE*n*=0), this read/write register configures the state of the pin as either an input or output. When DDIR is set to zero, the pin is an input. When DDIR is set to one, the pin is an output.

Data Direction (DDIR) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DD															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIO

Peripheral Enable (PEREN) Register

Bits	Name	Description	
15 - 0	PE	**Peripheral Enable**	
		This read/write register determines the GPIO configuration. When PEREN value is one, the peripheral owns the pin. This ownership includes configuring the pin as a required input, or output depending on the status of the peripheral output enable and includes data transfers from the pin to the peripheral. Please see **Table 7-2** for additional information. When the PEREN value is zero, the GPIO module owns the pin, controlling the direction of the data flow via the DDIR.	
		0	Pin is for GPIO
		1	Pin is for Peripheral

Peripheral Enable (PEREN) Register Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

GPIO

Interrupt Assert (IASSRT) Register

Bits	Name	Description	
15 - 0	IA	**Interrupt Assert**	
		This read/write register is used for software testing only. An interrupt is asserted when the IASSRT is set to one. The register is cleared by writing zeros. Keep in mind interrupts will be generated continually until this bit is cleared.	
		0	Deassert software interrupt
		1	Assert software interrupt

Interrupt Assert (IASSRT) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	IA															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

_____ Programmer: _____

GPIO

Interrupt Enable (IEN) Register

Bits	Name	Description	
15 - 0	**IEN**	**Interrupt Enable**	
		This read/write register enables or disables the edge detection for any incoming interrupt from the pin. Set this register to one for interrupt detection. The interrupts are recorded in the IPEND register.	
		0	Edge interrupt is disabled
		1	Edge interrupt is enabled

Interrupt Enable (IEN) Register Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	IEN															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

GPIO

Interrupt Polarity (IPOL) Register

Bits	Name	Description
15 - 0	IPOL	**Interrupt Polarity**
		This read/write register controls the edge polarity of any external interrupts enabled by an IEN bit set to one. When this register is set to one, the falling edge causes the interrupt. When this register is set to zero, the rising edge causes the interrupt.
		0 Rising edge generates the interrupt
		1 Falling edge generates the interrupt

Interrupt Polarity (IPOL) Register Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	IPOL															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

_____ Programmer: _____

GPIO

Interrupt Pending (IPEND) Register

Bits	Name	Description	
15 - 0	IPR	**Interrupt Pending**	
		This *read-only* register records any incoming interrupts. This register is read to determine which pin or bit of IASSRT register caused the interrupt. For external interrupts, the IPEND is cleared by writing ones into the IEDGE register. For software interrupts, the IPEND is cleared by writing zeros into the IASSRT register.	
		0	Interrupt cleared
		1	Interrupt pending

Interrupt Pending (IPEND) Register Base + $7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	IPR															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

GPIO

Interrupt Edge Sensitive (IEDGE) Register

Bits	Name	Description	
15 - 0	IES	**Interrupt Edge**	
		When an edge is detected by the edge detector circuit, and IEN is set to one, IEDGE records the interrupt. Please see **Figure 7-5**. The corresponding bit in the IPEND register is also set. This read/write register clears the corresponding IPR bit field by writing one to the corresponding bit in the IEDGE register. Writing zero to an IEDGE bit is ignored.	
		0	No edge seen
		1	Edge detected

Interrupt Edge Sensitive (IEDGE) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	IES															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

_____ Programmer: _____

GPIO

Push/Pull Output Mode Control (PPOUTM) Register

Bits	Name	Description
15 - 0	OEN	**Push/Pull Output Mode**
		This read/write register explicitly sets each output driver to either Push/Pull or Open Drain mode. The Open Drain mode can be used to tri-state any pin on the GPIO port without switching that pin to input mode. This is useful for some applications, including a keypad interface. These bits default to Push/Pull mode upon reset, giving all GPIO ports the same default functionality in this regard.
		0 Open drain mode
		1 Push/Pull mode

Push/Pull Output Mode Control (PPOUTM) Register Base + $9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	OEN															
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Application: _____ Date: _____

Programmer: _____

GPIO

Raw Data (RDATA) Register

Bits	Name	Description	
15 - 0	RAWDATA	**Raw Data**	
		This *read-only* register allows the controller direct access to the logic values on each GPIO pin even when pins are not in the GPIO mode. Values are not clocked and are subject to change at any time. The reset state is unknown. It is recommended to read several times to assure a stable value.	
		0	Logic 0 present on GPIO pin
		1	Logic 1 present on GPIO pin

Raw Data (RDATA) Register Base + $A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RAWDATA															
	Write																
	Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Reserved Bits

GPIO

Drive Strength Control (DRIVE) Register

Bits	Name	Description	
15 - 0	DRIVE	Drive	
		This read/write register can be used to explicitly set the drive strength of each output driver, independent of the Peripheral or GPIO mode configuration of the pin. This feature provides an additional design variable with the ability to control the effects of Electromagnetic Interference (EMI) due to digital switching. For capacitive loads, higher drive strengths improve low to high transition rates but increase EMI. Lower drive strengths reduce EMI and circuit board trace width requirements. These bits default to 4mA upon reset, providing all GPIO ports with the same default functionality.	
		0	4mA drive strength outputs
		1	8mA drive strength outputs

Drive Strength Control (DRIVE) Register Base + $B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DRIVE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

PWM

Control (CTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15 - 12	LDFQ	**Load Frequency**
		These buffered read/write bits select the PWM load frequency according to Table 8-8. Reset clears the LDFQ bits, selecting loading every PWM opportunity. The occurrence of a PWM opportunity is determined by the half bit.
11	HALF	**Half Cycle Reload**
		This read/write bit enables half cycle reloads in Center-Aligned PWM mode. This bit has no effect on Edge-Aligned PWMs.
		0 Half cycle reloads disabled
		1 Half cycle reloads enabled
10	IPOL2	**Current Polarity 2**
		This buffered read/write bit selects the PWM value register for the PWM4 and PWM5 pins in top/bottom manual deadtime correction.
		0 VAL4 register in next PWM cycle
		1 VAL5 register in next PWM cycle
9	IPOL1	**Current Polarity 1**
		This buffered read/write bit selects the PWM value register for the PWM2 and PWM3 pins in top/bottom manual deadtime correction.
		0 VAL2 register in next PWM cycle
		1 VAL3 register in next PWM cycle
8	IPOL0	**Current Polarity 0**
		This buffered read/write bit selects the PWM value register for the PWM0 and PWM1 pins in top/bottom manual deadtime correction.
		0 VAL0 register in next PWM cycle
		1 VAL1 register in next PWM cycle
7 - 6	PRSC	**Prescaler**
		These buffered read/write bits select the PWM clock frequency illustrated in Table 8-8 in this manual.

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC		PWMRIE	PWMF	0	0	LDOK	PWMEN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM

Control (CTRL) Register Continued

Bits	Name	Description
5	PWMRIE	**PWM Reload Interrupt Enable**
		This read/write bit enables the PWMF flag to generate interrupt requests. Reset clears PWMRIE bit.
		0 PWMF bit CPU interrupt request disabled
		1 PWMF bit CPU interrupt request enabled
4	PWMF	**PWM Reload Flag**
		This read/write flag is set at the beginning of every reload cycle regardless of the state of the LDOK bit. Clear PWMF bit by reading CTRL register with PWMF bit set before writing a Logic 0 to the PWMF bit. If another reload occurs before the clearing sequence is complete, writing Logic 0 to PWMF bit has no effect. Reset clears the PWMF bit.
		0 No new reload cycle since last PWMF clearing
		1 New reload cycle since last PWMF clearing
1	LDOK	**Load Okay**
		This read/write bit loads the prescaler bits of CTRL register and the entire CMOD and VALn registers into a set of buffers. The buffered prescaler divisor, PWM counter modulus value, and PWM pulse width take effect at the next PWM reload. Set LDOK by writing a Logic 1 to it. LDOK is automatically cleared after the new values are loaded, or can be manually cleared before a reload by writing a Logic 0 to it. Reset clears LDOK.
		0 Do not load new modulus, prescaler, and PWM values
		1 Load prescaler, modulus, and PWM values
0	PWMEN	**PWM Enable**
		This read/write bit enables the PWM generator and the PWM pins. When PWMEN=0, the PWM pins are in their inactive states unless OUTCTL*n*=1. A reset clears PWMEN.
		0 PWM generator and PWM pins disabled unless OUTCTL*n*=1
		1 PWM generator and PWM pins enabled

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC		PWMRIE	PWMF	0	0	LDOK	PWMEN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

PWM

Fault Control (FCTRL) Register

Bits	Name	Description
11, 10, 9, 8	**FPOL*n***	**Fault*n* Polarity Control**
		These read/write bits control the polarity of the FAULT*n* pin inputs. A reset clears FPOL*n*. FPOL2 is also used to control the polarity of the external sync input and output.
		0 Logic 1 on FAULT*n* indicates a fault condition
		1 Logic 0 on FAULT*n* indicates a fault condition
7, 5, 3, 1	**FIE*n***	**Fault*n* Interrupt Enable**
		This read/write bit enables CPU interrupt requests generated by the filtered FAULT*n* pin. A reset clears FIE*n*.
		0 FAULT*n* CPU interrupt request disabled
		1 FAULT*n* CPU interrupt request enabled
6, 4, 2, 0	**FMODE*n***	**FAULT*n* Clearing Mode**
		These read/write bits select automatic or manual clearing of FAULT*n* pin faults. A reset clears FMODE*n*.
		0 Manual fault clearing of FAULT*n* pin faults
		1 Automatic fault clearing of FAULT*n* pin faults

Fault Control (FCTRL) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	FPOL3	FPOL2	FPOL1	FPOL0	FIE3	FMODE3	FIE2	FMODE2	FIE1	FMODE1	FIE0	FMODE0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Fault Status and Acknowledge (FLTACK) Register

Bits	Name	Description
15,13, 11, 9	**FPIN*n***	**FAULT*n* Pin**
		These *read-only* bits reflect the current state of the filtered FAULT*n* bit. A reset has no effect on FPIN*n*.
		0 Logic 0 on the FAULT*n* bit
		1 Logic 1 on the FAULT*n* bit
14, 12, 10, 8	**FFLAG*n***	**FAULT*n* Pin Flag**
		These *read-only* bits are set within two CPU cycles after a rising edge on the filtered FAULT*n* pins. Clear FFLAG*n* by writing Logic 1 to the FTACK*n* bit in this register (PMFSA). A reset clears FFLAG*n*.
		0 No fault on the FAULT*n* bit
		1 Fault on the FAULT*n* bit
6, 4, 2, 0	**FTACK*n***	**FAULT*n* Pin Acknowledge**
		Writing Logic 1 to FTACK*n* clears FFLAG*n*. Writing Logic 0 has no effect. Read these bits as zero. Reset clears FTACK*n*. The fault protection is enabled even when the PWM is not enabled; therefore, a fault is latched in, requiring it to be cleared in order to prevent an interrupt when the PWM is enabled.

Fault Status & Acknowledge (FLTACK) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FPIN3	FFLAG3	FPIN2	FFLAG2	FPIN1	FFLAG1	FPIN0	FFLAG0	0	0	0	0	0	0	0	0
	Write										FTACK3		FTACK2		FTACK1		FTACK0
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Output Control (OUT) Register

Bits	Name	Description
15	PAD_EN	**Output Pad Enable**
		The PWM*n* outputs can be enabled or disabled by setting the PAD_EN bit. The power up default has the pads disabled. This bit does not affect the functionality of the PWM, so the PWM module can be energized with the output pads disabled. This enable is to power up with a safe default value for the PWM drivers.
		0 Output pads disabled
		1 Output pads enabled
13 - 8	OUTCTL5-0	**Output Control Enable**
		These read/write bits enable software control of their corresponding PWM pin. When OUTCTL*n* is set, the OUT*n* bit activates and deactivates the PWMn output or the SRC*n* bits in the SCTRL register are used to select an alternate control of the PWM outputs. A reset clears the OUTCTL bits. When operating the PWM in complementary mode, these bits must be switched in pairs for proper operation. That is, OUTCTL0 and OUTCTL1 must have the same value; OUTCTL2 and OUTCTL3 must have the same value; and OUTCTL4 and OUTCLT5 must have the same value.
		0 Software control disabled (normal PWM operation)
		1 Software control enabled
5 - 0	OUT5-0	**Output Control**
		When the corresponding OUTCTL bit is set, these read/write bits control the PWM pins, illustrated in **Table 8-9**.

Output Control (OUT) Register Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PAD_EN	0	OUT CTL5	OUT CTL4	OUT CTL3	OUT CTL2	OUT CTL1	OUT CTL0	0	0	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Counter (CNTR) Register

Bits	Name	Description
14 - 0	CNTR	**Counter**
		This *read-only* bit field displays the state of the 15-bit PWM counter.

Counter (CNTR) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	CNTR														
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Counter Modulo (CMOD) Register

Bits	Name	Description
14 - 0	CM	**Counter Modulo**
		The 15-bit unsigned value written to this buffered, read/write register defines the PWM period in PWM clock periods. Do not write a modulus value of zero to bit 15. The PWM counter modulo register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins. Reading CMOD reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

Counter Modulo (CMOD) Register Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	CM														
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Value (VAL0-5) Registers

Bits	Name	Description
15 - 0	PWMVAL	Value
		The 16-bit signed value in these buffered, read/write registers defines the pulse width in PWM clock periods for each PWM Output Channel. The Value registers are buffered. The value written does not take effect until the LDOK bit is set and the next load cycle begins. Reading VAL*n* reads the value in a buffer and not necessarily the value the PWM generator is currently using. A value less than, or equal to zero deactivates the output for the entire PWM period. A value greater than, or equal to the modulus, activates the output for the entire period. Please see **Table 8-1**. The terms activate and deactivate refer to the high and low logic states of the PWM outputs.

Value (VAL0-5) Registers Base + $6 - $B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PWMVAL															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

PWM

Deadtime 0-1 (DTIM0-1) Registers

Bits	Name	Description
11 - 0	PWMDT0	Deadtime 0
11 - 0	PWMDT1	Deadtime 1
		Deadtime operation is only applicable to complementary channel operation. The 12-bit values written to these write-protected registers are in terms of PWM clock cycles. Reset sets the PWM deadtime registers to a default value of 0×0FFF, selecting a deadtime of 4096-PWM clock cycles minus one IPBus clock cycle. These registers are write protected after the WP bit in the CNFG register is set. Please refer to **Section 8.7.10**. Reserved bits 15–12 cannot be modified. They are read as zero. Deadtime is affected by changes to the prescaler value. The deadtime duration is determined as follows: DT = P × PWMDT – 1, where DT is deadtime, P is the prescaler value, PWMDT is the programmed value of deadtime. For example: if the prescaler is programmed for a divide-by-two and PWMDT is set to five, then P = 2 and the deadtime value is equal to DT = 2 × 5 – 1 = 9 IPBus clock cycles. A special case exists when the P = 1, DT = PWMDT. The PWMDT0 field is used to control the deadtime during zero to one transitions of the PWM output and during one to zero transitions of the complementary output (assuming normal polarity). The PWMDT1 field is used to control the deadtime during one to zero transitions of the primary output and zero to one transitions of the complementary output.

Deadtime (DTIM0) Register Base + $C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	PWMDT0											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Deadtime (DTIM1) Register Base + $D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	PWMDT1											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Disable Mapping 1-2 (DMAP1-2) Registers

Bits	Name	Description
15 - 0	DISMAP1	**Disable Mapping 1**
23 - 16	DISMAP2	**Disable Mapping 2**
		These *write-protected* registers determine which PWM pins are disabled by the fault protection inputs, provided in **Table 8-4** in this manual. Reset sets all of the bits used in the DMAP1-2 registers. These registers are write-protected after the WP bit in the CNFG register is set. Reserved bits 15-8 in the DMAP2 register cannot be modified. The bits are read as zero.

Disable Mapping (DMAP1) Register Base + $E

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	DISMAP1[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Disable Mapping (DMAP2) Register Base + $F

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DISMAP2[23:16]							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Configure (CNFG) Register

Please see the following page for continuation of this register

Bits	Name	Description
14	DBG_EN	**Debug Enable**
		When set to one, the PWM will continue to run while the chip is in EOnCE Debug mode. If the device enters the EOnCE mode and this bit is zero, the PWM outputs will be switched to their inactive state until the EOnCE mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.
13	WAIT_EN	**Wait Enable**
		When set to one, the PWM will continue to run while the chip is in the Wait mode. In this mode, the peripheral clock continues to run; however, the core clock does not. If the device enters the Wait mode and this bit is zero, the PWM outputs switch to their inactive state until the Wait mode is exited. At the point the PWM pins resume operation as programmed in the PWM registers.
12	EDG	**Edge-Aligned or Center-Aligned PWMs**
		This *write-protected* bit determines whether all PWM channels will use Edge-Aligned or Center-Aligned waveforms.
		0 Center-Aligned PWMs
		1 Edge-Aligned PWMs
10 - 8	TOPNEG	**Top-Side PWM Polarity**
		This *write-protected* bit determines the polarity for the top-side PWMs.
		0 Positive top-side polarity
		1 Negative top-side polarity

Configure (CNFG) Register Base + $10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	DBG_EN	WAIT_EN	EDG	0	TOP NEG45	TOP NEG23	TOP NEG01	0	BOT NEG45	BOT NEG 23	BOT NEG01	INDEP 45	INDEP 23	INDEP 01	WP
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

PWM

Configure (CNFG) Register Continued

Bits	Name	Description
6 - 4	BOTNEG	**Bottom-Side PWM Polarity**
		This *write-protected* bit determines the polarity for the bottom-side PWMs. Each pair of PWM channels can be configured: channel zero to one, channel two to three, and channel four to five.
		0 Positive bottom-side polarity
		1 Negative bottom-side polarity
3 - 1	INDEP	**Independent or Complement Pair Operation**
		This *write-protected* bit determines if the PWM channels will be independent PWMs or complementary PWM pairs. Each pair of PWM channels can be configured: Channel 0-1, Channel 2-3, and Channel 4-5.
		0 Complementary PWM pair
		1 Independent PWMs
0	WP	**Write Protect**
		This write-protected bit enables write protection to be used for all write-protected registers. While clear, WP allows write-protected registers to be written. When set, WP prevents any further writes to write-protected registers. Once set, WP can be cleared only by a reset. Write-protected registers include SCTRL, DISMAP1-2, DTIM0 and DTIM1, CNFG, SYNC, FFILT*n*, and the ENHA bit in the CCTRL. The VLMODE, SWP0, SWP1 and SWP2 bits in the CCTRL register are protected when the ENHA bit is set to zero in the CCTRL register. ENHA is in turn protected by setting WP in the CNFG register. The write to CNFG setting the WP bit is the last write accepted to the register until reset.
		0 Write-protected registers may be modified by writing
		1 Write-protected register are *read-only*

Configure (CNFG) Register Base + $10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	DBG_EN	WAIT_EN	EDG	0	TOP NEG45	TOP NEG23	TOP NEG01	0	BOT NEG45	BOT NEG 23	BOT NEG01	INDEP 45	INDEP 23	INDEP 01	WP
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Channel Control (CCTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	ENHA	**Enable Hardware Acceleration**
		This bit enables writing to the nBX, VLMODE, SWP45, SWP23, and SWP01 bits. The bit is write-protectable by the WP bit in the CNFG register.
		0 Disable writing to nBX, VLMODE, SWP45, SWP23, and SWP01 bits
		1 Enable writing to nBX, VLMODE, SWP45, SWP23, and SWP01 bits
14	nBX	**56F80x Compatibility**
		This bit is used to enable/disable improved SWAP and MASK operations. If it is cleared, SWAP/MASK operates identical to the 56F80x version of this module. See 56F80x User's Manual for details. If is set, SWAP and MASK operations are described in this manual which are not compatible features with 56F80x and are not supported. This bit is write-protected when ENHA is zero.
		0 SWAP and MASK provide 56F80x compatible operation
		1 SWAP*n* and MASK*n* provide new functionality described in this manual and illustrated in **Figure 8-2**
13 - 8	MSK5-0	**Mask 5-0**
		These six bits determine the mask for each of the PWM logical channels.
		0 Unmasked
		1 Masked, channel set to a value of zero percent duty cycle
5 - 4	VLMODE	**Value Register Load Mode**
		These two bits determine the way the Value registers are being loaded.
		00 Each value register is accessed independently
		01 Writing to VAL0 register also writes to registers VAL1-5
		10 Writing to VAL0 register also writes to registers VAL1-3
		11 Reserved

Channel Control (CCTRL) Register Base + $11	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP45	SWP23	SWP01
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

PWM

Channel Control (CCTRL) Register Continued

Bits	Name	Description
2	SWP45	**Swap45**
		The SWAP and MASK functions have two different modes of operation controlled by the nBX bit in the CCTRL register. **Figure 8-49** is somewhat of a simplification. See **Figure 8-2** and **Section 8.7.11.2** for details.
	0	No swap
	1	Channels four and five are swapped
1	SWP23	**Swap23**
		The SWAP and MASK functions have two different modes of operation controlled by the nBX bit in the CCTRL register. **Figure 8-49** is somewhat of a simplification. See **Figure 8-2** and **Section 8.7.11.2** for details.
	0	No swap
	1	Channels two and three are swapped
0	SWP01	**Swap01**
		The SWAP and MASK functions have two different modes of operation controlled by the nBX bit in the CCTRL register. **Figure 8-49** is somewhat of a simplification. See **Figure 8-2** and **Section 8.7.11.2** for details.
	0	No swap
	1	Channels zero and one are swapped

Channel Control (CCTRL) Register Base + $11	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP45	SWP23	SWP01
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Port (PORT) Register

Bits	Name	Description
3 - 0	PORT	Port
		This register contains values of the four fault inputs, bits three, two, one, and zero. This is a *read-only* register, therefore any writes to the register do not affect it. This register may be read while the PWM is active.

Port (PORT) Register Base + $12	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	PORT			
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	U	U	U	U

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Internal Correction Control (ICCTRL) Register

Bits	Name	Description
2	ICC2	**Internal Current Control 2**
		This bit controls the PWM4/PWM5 pair.
		0 Use VAL4 register at all times
		1 Use VAL4 register when the PWM counter is counting up. Use VAL5 register when counting down.
1	ICC1	**Internal Current Control 1**
		This bit controls the PWM2/PWM3 pair.
		0 Use VAL*2* register at all times
		1 Use VAL2 register when the PWM counter is counting up. Use VAL3 register when counting down.
0	ICC0	**Internal Current Control 0**
		This bit controls the PWM0/PWM1 pair.
		0 Use VAL0 register at all times
		1 Use VAL0 register when the PWM counter is counting up. Use VAL1 register when counting down.

Internal Correction Control (ICCTRL) Register Base + $13	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	ICC2	ICC1	ICC0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Source Control (SCTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
13	CINV5	**Compare Invert 5**
		This bit controls the polarity of PWM compare output five. Please see the output operations in **Figure 8-7** and **Figure 8-8** in this manual.
	0	PWM output five is high when the CNTR register is less than VAL5 register
	1	PWM output five is high when the CNTR register is greater than VAL5 register
12	CINV4	**Compare Invert 4**
		This bit controls the polarity of PWM compare output four. Please see the output operations in **Figure 8-7** and **Figure 8-8** in this manual.
	0	PWM output four is high when the CNTR register is less than VAL4 register
	1	PWM output four is high when the CNTR register is greater than VAL4 register
11	CINV3	**Compare Invert 3**
		This bit controls the polarity of PWM compare output three. Please see the output operations in **Figure 8-7** and **Figure 8-8** in this manual.
	0	PWM output three is high when the CNTR register is less than VAL3 register
	1	PWM output three is high when the CNTR register is greater than VAL3 register
10	CINV2	**Compare Invert 2**
		This bit controls the polarity of PWM compare output two. Please see the output operations in **Figure 8-7** and **Figure 8-8** in this manual.
	0	PWM output two is high when the CNTR register is less than VAL2 register
	1	PWM output two is high when the CNTR register is greater than VAL2 register

Source Control (SCTRL) Register Base + $14	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	CINV5	CINV4	CINV3	CINV2	CINV1	CINV0	0	SRC2		0	SRC1		0	SRC0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

PWM

Source Control (SCTRL) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
9	CINV1	**Compare Invert 1**
		This bit controls the polarity of PWM compare output one. Please see the output operations in **Figure 8-7** and **Figure 8-8** in this manual.
		0 PWM output one is high when the CNTR register is less than VAL1 register
		1 PWM output one is high when the CNTR register is greater than VAL1register
8	CINV0	**Compare Invert 0**
		This bit controls the polarity of PWM compare output zero. Please see the output operations in **Figure 8-7** and **Figure 8-8** in this manual.
		0 PWM output zero is high when the CNTR register is less than VAL0 register
		1 PWM output zero is high when the CNTR register is greater than VAL0 register
6 - 5	SRC2	**Source 2**
		This field controls the PWM5/PWM4 pair. Be certain OUTCTL4 and OUTCTL5 (12 and 13) bits of the PWM OUT register are set when using these bits.
		00 Use PWM generator as PWM source (operation is consistent with 56F80x and 56F83xx devices)
		01 Use PSRC2 input as PWM source. The specific signal driving this input is based on muxing controlled by the SIM IPS register. This information can be found in the device data sheet.
		1x Use the value selected in SCR0 as the PWM source

Source Control (SCTRL) Register Base + $14	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	CINV5	CINV4	CINV3	CINV2	CINV1	CINV0	0	SRC2		0	SRC1		0	SRC0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Source Control (SCTRL) Register Continued

Bits	Name	Description	
3 - 2	**SRC1**	**Source 1**	
		This field controls the PWM2/PWM3 pair. Be certain OUTCTL2 and OUTCTL3 (10 and 11) bits of the PWM OUT register are set when using these bits.	
		00	Use PWM generator as PWM source (operation is consistent with 56F80x and 56F83xx devices)
		01	Use PSRC2 input as PWM source. The specific signal driving this input is based on muxing controlled by the SIM IPS register. This information can be found in the device data sheet.
		1x	Use the value selected in SCR0 as the PWM source
0	**SRC0**	**Source 0**	
		This field controls the PWM0/PWM1 pair. Be certain OUTCTL0 and OUTCTL1 (8 and 9) bits of the PWM OUT register are set when using these bits.	
		00	Use PWM generator as PWM source (operation is consistent with 56F80x and 56F83xx devices)
		01	Use PSRC2 input as PWM source. The specific signal driving this input is based on muxing controlled by the SIM IPS register. This information can be found in the device data sheet.
		1x	Use the value selected in SCR0 as the PWM source

Source Control (SCTRL) Register Base + $14	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	CINV5	CINV4	CINV3	CINV2	CINV1	CINV0	0	SRC2		0	SRC1		0	SRC0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Synchronization Window (SYNC) Register

Bits	Name	Description
15	SYNC_OUT_EN	**Synchronization Output Enable**
		This bit controls the enable for the synchronization output. Do not set this bit when the FAULT2 pin is to be used as a fault input.
		0 Synchronization output pulse disabled
		1 Synchronization output pulse enabled on FAULT2 pin
14 - 0	SYNC_WINDOW	**Synchronization Window**
		This field defines the window of opportunity for the external sync signal to reset the counter. For center aligned operation (EDG=0) if the value of the counter is less than the value of SYNC_WINDOW bit field, the external synchronization is enabled. This means, for a SYNC_WINDOW bit field value of zero, external synchronization can never take place (i.e., it is disabled). For a SYNC_WINDOW bit field value of 0x7FFF, the external sync can occur at any time (i.e., always enabled). For edge aligned operation (EDG=1) if the value of the counter is less than the value of the SYNC_WINDOW bit field or if the difference between the value of the counter and the counter modulo value is less than the value of SYNC_WINDOW, then external synchronization is enabled. SYNC_WINDOW bit field should be set to zero when SYNC_OUT_EN is set to one. This enables the sync output on the FAULT2 pin and disable the sync input on that same pin.

Synchronization Window (SYNC) Register Base + $15	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SYNC_OUT_EN	SYNC_WINDOW														
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PWM

Fault Filter 0-3 (FFILT0-3) Register

Bits	Name	Description
10 - 8	FILT*n*_CNT	**Input Filter Sample Count**
		These bits represent the number of consecutive samples required to agree prior to the input filter accepting an input transition. A value of 0x0 represents three samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency as described in **Section 8.7.16.4**.
7 - 0	FILT*n*_PER	**Input Filter Sample Period**
		These bits represent the sampling period (in IPBus clock cycles) of the fault input signals. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), the input filter is bypassed. The value of FILT_PER affects the input latency as described in **Section 8.7.16.4**.

Fault Filter 0-3 (FFILT0-3) Registers Base + $16 - $19	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	FILT*n*_CNT			FILT*n*_PER							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Control 0 (CTRL0) Register

Please see the following page for continuation of this register

Bits	Name	Description
7	FXFRM	**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 being set, it remains set until cleared by software or reset. Clearing is achieved by writing one. Writing zero is ignored. This bit is not valid in Loopback mode. CAN must be in Normal mode for this bit to be set.
		0 No valid message was received since last clearing this bit
		1 A valid message was received since last clearing this bit
6	RXACT	**Receive Active Status**
		This *read-only* bit indicates the CAN is receiving a message. The bit is controlled by the receiver front end. This bit is not valid in Loopback mode.
		0 SCAN is transmitting or idle (See the Bosch CAN 2.0A/B specification for detailed definition.)
		1 SCAN is receiving a message, including when arbitration is lost. (See Bosch CAN 2.0 A/B as above.)
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 CAN module. In order to protect from accidentally violating the CAN protocol, the TXCAN terminal is immediately forced to a recessive state when the CPU enters wait (CSWAI=1) or Stop mode. Please see **Section 9.7.6.2** and **Section 9.7.6.3**.
		0 The module is not affected during Wait mode
		1 The module is synchronized to the CAN bus
4	SYNCH	**Synchronized Status**
		This *read-only* bit indicates if CAN is synchronized to the CAN bus and is able to participate in the communication process. It is set and cleared by the CAN.
		0 SCAN is not synchronized to the CAN bus
		1 SCAN is synchronized to the CAN bus

Control 0 (CTRL0) Register Base + $0	Bits	7	6	5	4	3	2	1	0
	Read	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
	Write								
	Reset	0	0	0	0	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Control 0 (CTRL0) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
3	TIME	**Timer Enable**
		This bit activates an internal 16-bit wide free running timer clocked by the bit clock rate. If the timer is enabled, a 16-bit time stamp is assigned to each transmitted/received message within the active TX/RX buffer{. As soon as a message is acknowledged on the CAN bus, the time stamp is written to the highest bytes (0x000E, 0x000F) in the appropriate buffer. Please see **Section 9.9.18**. The internal timer is reset (all bits set to zero) when disabled. This bit is held low in Initialization mode.
		0 Disable internal SCAN timer
		1 Enable internal SCAN timer
2	WUPE	**Wake-Up Enable**
		This configuration bit allows the CAN to restart from Sleep mode when traffic on CAN is detected. Please see **Section 9.7.6.4**. If the WUPE bit in CTRL0 is not asserted, the MSCAN masks any activity it detects on CAN. The RXCAN bit is therefore held internally in a recessive state. This locks the MSCAN in Sleep mode, illustrated in **Figure 9-10**. For Sleep mode to take effect, the WUPE bit must be set before entering Sleep mode. The CPU has to be certain the WUPE bit in the CTRL0 register and WUPIE bit in the RIER register are enabled, if the recovery mechanism from stop or wait is required. Please see **Section 9.9.6**.
		0 Wake-up disabled – The SCAN ignores traffic on CAN
		1 Wake-up enabled – The SCAN is able to restart
1	SLPRQ	**Sleep Mode Request**
		This bit requests the CAN to enter Sleep mode, an internal power saving mode. Please see **Section 9.7.6.4**. 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. For further information see **Section 9.9.2**. Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the CAN detects activity on the CAN bus, clearing SLPRQ itself. The CPU cannot clear SLPRQ before the CAN has entered sleep mode (SLPRQ=1 and SLPAK=1). The SLPR bit cannot be set while the WUPIF bit is set.
		0 Running – The SCAN functions normally
		1 Sleep mode request – The SCAN enters Sleep mode when CAN bus idle

Control 0 (CTRL0) Register Base + $0	Bits	7	6	5	4	3	2	1	0
	Read	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
	Write								
	Reset	0	0	0	0	0	0	0	1

Reserved Bits

MSCAN

Control 0 (CTRL0) Register Continued

Bits	Name	Description
0	INITRQ	**Initialization Mode Request** When this bit is set by the CPU, the CAN skips to Initialization mode. Please refer to **Section 9.7.6.5**. 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. See **Section 9.9.2**. The following registers enter their hard reset state and restore their default values: CTRL0[1], RFLG[2], RIER[3], TFLG, TIER, TARQ, TAAK, and TBSEL. Registers CTRL1, BTR0, BTR1, IDAC, IDAR0-7, and IDMR0-7 can only be written by the CPU when the CAN is in Initialization mode (INITRQ=1 and INITAK=1). The values of the error counters are not affected by Initialization mode. When this bit is cleared by the CPU, the CAN restarts and then attempts to synchronize to the CAN bus. If the CAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the CAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits. Writing to other bits in CTRL0, RFLG, RIER, TFLG, or TIER registers must be completed only after Initialization mode is exited (INITRQ=0 and INITAK=0). The CPU cannot clear INITRQ before the CAN enters Initialization mode (INITRQ=1 and INITAK=1). In order to protect from accidentally violating the CAN protocol, the TXCAN terminal is immediately forced to a recessive state when the Initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the CAN into Sleep mode (SLPRQ=1 and SLPAK=1) before requesting Initialization mode. 0 Normal operation 1 SCAN in Initialization mode

1. Not including WUPE, INITRQ, and SLPRQ
2. TSTAT1 and TSTAT0 are not affected by initialization mode
3. RSTAT1 and RSTAT0 are not affected by initialization mode

Control 0 (CTRL0) Register Base + $0	Bits	7	6	5	4	3	2	1	0
	Read	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
	Write								
	Reset	0	0	0	0	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Control 1 (CTRL1) Register

Please see the following page for continuation of this register

Bits	Name	Description
7	CANE	**Enable**
		0 SCAN module is disabled
		1 SCAN in Initialization mode
6	CLKSRC	**SCAN Clock Source**
		This bit defines the clock source for the CAN module (only for systems with a clock generation module; Section 9.5.2.2, and Figure 9-7).
		0 SCAN clock source is the oscillator clock
		1 SCAN clock source is the bus clock
5	LOOPB	**Loop Back Self Test Mode**
		When this bit is set, the CAN performs an internal loop back, used for self test operation. The bit stream output of the transmitter is fed back to the receiver internally. The RXCAN input terminal is ignored and the TXCAN output goes to the recessive state (Logic 1). The CAN behaves as it does normally when transmitting, treating its own transmitted message as a message received from a remote node. In this state, the CAN 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 CAN as a CAN bus monitor. When LISTEN bit is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out. Further information is available in Section 9.7.4. Additionally, the error counters are frozen. Listen-Only mode supports applications requiring hot plugging, or throughput analysis. The CAN is unable to transmit any messages when Listen-Only mode is active.
		0 Normal operation
		1 Listen-Only mode activated

Control 1 (CTRL1) Register Base + $1	Bits	7	6	5	4	3	2	1	0
	Read	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
	Write								
	Reset	0	0	0	1	0	0	0	1

Reserved Bits

MSCAN

Control 1 (CTRL1) Register Continued

Bits	Name	Description
3	BORM	**Bus-Off Recovery Mode**
		This bits configures the Bus-Off state Recovery mode of the CAN. Refer to Section 9.6.2 for details.
		0 Automatic Bus-Off recovery mode (see Bosch CAN 2.9A/B protocol specification)
		1 Bus-Off recovery upon user request
2	WUPM	**Wake-Up Mode**
		If WUPE bit in the CTRL0 register is enabled, it defines whether the integrated low-pass filter is applied to protect the CAN from spurious wake-up. See Section 9.7.6.4.
		0 SCAN wakes up the CPU on any to dominate level on the CAN bus
		1 SCAN wakes up the CPU only in case of a dominate pulse on the CAN bus having a length of T_{WUP}
1	SLPAK	**Sleep Mode Acknowledge**
		This bit indicates whether the CAN module has entered Sleep mode. Section 9.7.6.4 provides additional information. SLPAK is used as a handshake bit for the SLPRQ Sleep mode request. Sleep mode is active when SLPRQ=1 and SLPAK=1. Depending on the setting of WUPE, the CAN will clear the bit if it detects activity on the CAN bus while in Sleep mode.
		0 Running – The SCAN operates normally
		1 Sleep mode active – The SCAN entered Sleep mode
0	INITAK	**Initialization Mode Acknowledge**
		This bit indicates if the CAN module is in Initialization mode. Section 9.7.6.5 provides additional data. INITAK is used as a handshake bit for the INITRQ Initialization mode request. Initialization mode is active when INITRQ=1 and INITAK=1. The registers CTRL1, BTR0, BTR1, IDAC, IDAR0–IDAR7, and IDMR0–IDMR7 can be written only by the CPU when the CAN is in Initialization mode.
		0 Running – The SCAN operates normally
		1 Initialization mode active qwP The SCAN entered Initialization mode

Control 1 (CTRL1) Register Base + $1	Bits	7	6	5	4	3	2	1	0
	Read	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
	Write								
	Reset	0	0	0	1	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Bus Timing 0 (BTR0) Register

Bits	Name	Description
7 - 6	SJW1-0	**Synchronization Jump Width 1-0**
		The synchronization jump width defines the maximum number of Time Quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve re synchronization to data transitions on the CAN bus. **Table 9-6** provides additional information.
5 - 0	BRP5-0	**Baud Rate Prescaler 5-0**
		These bits determine the time quanta (Tq) clock used to build-up the timing. Please see **Table 9-7**.

Bus Timing 0 (BTR0) Register Base + $2	Bits	7	6	5	4	3	2	1	0
	Read	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
	Write								
	Reset	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

MSCAN

Bus Timing 1 (BTR1) Register

Bits	Name	Description
7	SAMP	**Sampling**
		This bit determines the number of CAN bus samples taken per bit time. If SAMP=0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP=1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended only one sample is taken per bit time (SAMP=0).
		0 One sample per bit
		1 Three samples per bit (in this case, PHSE-SEG1 must be at least two times quanta (T_Q))
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. Please see **Figure 9-8**. Time Segment 2 (TSEG2) values are programmable, illustrated in **Table 9-8**.
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. Please see **Figure 9-8**. Time Segment 1 (TSEG1) values are programmable, illustrated in **Table 9-8**.

Bus Timing 1 (BTR1) Register Base + $3	Bits	7	6	5	4	3	2	1	0
	Read	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
	Write								
	Reset	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

MSCAN

Receiver Flag (RFLG) Register

Please see the following page for continuation of this register

Bits	Name	Description
7	WUPIF	**Wake-Up Interrupt Flag**
		If the CAN detects CAN bus activity while in Sleep mode and WUPE=1 bit in CTRL0 register, the module will set WUPIF. See both Section 9.7.6.4 and Section 9.9.1 for additional information. If not masked, a wake-up interrupt is pending while this bit is set.
		0 No wake-up activity observed while in Sleep mode
		1 SCAN detected activity on the CAN bus and requested wake-up
6	CSCIF	**CAN Status Change Interrupt Flag**
		This bit is set when the CAN 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, split into separate sections for TEC/REC, informs the system on the actual CAN bus status. Additional data is available in Section 9.9.6. If not masked, an error interrupt is pending while this bit is set. CSCIF bit provides a blocking interrupt. That block guarantees 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 bit is asserted, causing an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF bit interrupt is cleared again.
		0 No change in CAN bus status occurred since last interrupt
		1 SCAN changed current CAN bus status
5 - 4	RSTAT	**Receiver Status**
		The values of the error counters control the actual CAN bus status of the CAN. As soon as the Status Change Interrupt Flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the SCAN. The coding for the bits RSTAT1, RSTAT0 is:
		00 RXOK: 0 ≤ receive error counter ≤ 96
		01 RXWRN: 96 < receive error counter ≤ 127
		10 RXERR: 127 < receive error counter
		11 Bus-Off[1]: transmit error counter > 255

1. 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[1:0] coding in this register.

Receiver Flag (RFLG) Register Base + $4	Bits	7	6	5	4	3	2	1	0
	Read	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

MSCAN

Receiver Flag (RFLG) Register Continued

Bits	Name	Description
3 - 2	TSTAT	**Transmitter Status**
		The values of the error counters control the actual CAN bus status of the CAN. As soon as the Status Change Interrupt Flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the CAN. The coding for the bits TSTAT1, TSTAT0 is:
		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 bit is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this bit is set.
		0 No data overrun condition
		1 A data overrun detected
0	RXF	**Receive Buffer Full Flag**
		RXF is set by the CAN when a new message is shifted in the receiver FIFO. This bit 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 reads that message from the RXFG buffer in the receiver FIFO, the RXF bit must be cleared to release the buffer. A set RXF bit prohibits shifting of the next FIFO entry into the foreground buffer (RXFG). If not masked, a receive interrupt is pending while this bit is set.
		0 No new message available within the RxFG
		1 The receiver FIFO is not empty. A new message is available in the RxFG

Receiver Flag (RFLG) Register Base + $4	Bits	7	6	5	4	3	2	1	0
	Read	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Receiver Interrupt Enable Register (RIER)

Please see the following page for continuation of this register

Bits	Name	Description	
7	WUPIE	**Wake-Up Interrupt Enable**	
		Both the WUPIE and WUPE bits must be enabled if the recovery mechanism from Stop or Wait is required. Please refer to **Section 9.9.1**.	
		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**	
		These RSTATE enable bits control the sensitivity level in the receiver state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level the RSTATE bits continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending.	
		00	Do not generate any CSCIF interrupt caused by receiver state changes
		01	Generate CSCIF interrupt only if the receiver enters or leaves *bus-off* state. Discard other receiver state changes for generating CSCIF interrupt.
		10	Generate CSCIF interrupt only if the receiver enters or leaves RXERR or bus-off state. Discard other receiver state changes for generating CSCIF interrupt.
		11	Generate CSCIF interrupt on all state changes

Receiver Interrupt Enable (RIER) Register Base + $5	Bits	7	6	5	4	3	2	1	0
	Read	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
	Write								
	Reset	0	0	0	0	0	0	0	0

MSCAN

Receiver Interrupt Enable Register (RIER) Continued

Bits	Name	Description	
3 - 2	TSTATE	**Transmitter Status Change Enable**	
		These TSTATE enable bits control the sensitivity level in the transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTATE bits 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

Receiver Interrupt Enable Register (RIER) Base + $5	Bits	7	6	5	4	3	2	1	0
	Read	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
	Write								
	Reset	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

MSCAN

Transmitter Flag (TFLG) Register

Bits	Name	Description	
2 - 0	TXE	**Transmitter Buffer Empty**	
		This bit field indicates the associated transmit message buffer is empty and thus not scheduled for transmission. The CPU must clear the bit after a message is set up in the transmit buffer and is due for transmission. The CAN sets the bit after the message is sent successfully. The bit is also set by the CAN when the transmission request is successfully aborted due to a pending abort request. More data is available in **Section 9.9.9**. If not masked, a transmit interrupt is pending while this bit is set. Clearing a TXE*n* bit also clears the corresponding ABTAKn. Additional data is available in **Section 9.9.9**. When a TXE*n* bit is set, the corresponding ABTRQ*n* bit is cleared. When Listen mode is active, the TXE*n* bits cannot be cleared and no transmission is started. Please see **Section 9.9.2**. Read and write accesses to the transmit buffer is blocked if the corresponding TXE*n* bit is cleared (TXE*n*=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)

Transmitter Flag (TFLG) Register Base + $6	Bits	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	TXE2	TXE1	TXE0
	Write								
	Reset	0	0	0	0	0	1	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Transmitter Interrupt Enable Register (TIER)

Bits	Name	Description	
2 - 0	TEIE	**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.

Transmitter Interrupt Enable Register (TIER) Base + $7	Bits	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Transmitter Message Abort Request (TARQ) Register

Bits	Name	Description	
2 - 0	ABTRQ	**Abort Request**	
		The CPU sets the ABTRQ*n* bit to request a scheduled message buffer (TXE*n*=0) be aborted. The CAN grants the request if the message is not already in transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE (see **Section 9.9.7**) and Abort Acknowledge (ABTAK) bits are set and a transmit interrupt occurs if enabled. The CPU cannot reset ABTRQ*n*. ABTRQ*n* is reset whenever the associated TXE bit is set. Please see **Section 9.9.10**.	
		0	No abort request
		1	Abort request pending

Transmitter Msg. Abort Request (TARQ) Register Base + $8	Bits	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

MSCAN

Transmitter Message Abort Acknowledge (TAAK) Register

Bits	Name	Description
2 - 0	ABTAK	**Abort Acknowledge**
		This bit acknowledges a message was aborted due to a pending abort request from the CPU. After a particular message buffer is flagged empty, this bit can be used by the application software to identify whether the message was aborted successfully or was sent anyway. The ABTAK*n* bit is cleared whenever the corresponding TXE bit is cleared.
		0 The message was not aborted
		1 The message was aborted

Transmitter Msg. Abort Acknowledge (TAAK) Register Base + $9	Bits	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Transmit Buffer Selection (TBSEL) Register

Bits	Name	Description	
2 - 0	TX	**Transmit Buffer Select**	
		The lowest numbered bit places the respective transmit buffer in the 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 are blocked if the corresponding TXE*n* bit is cleared and the buffer is scheduled for transmission. Please see **Section 9.9.7**. A brief programming example of the TBSEL register usage is provided here: To get the next available transmit buffer, application software must read the TFLG register and write this value back into the TBSEL register. In this example TX buffers TX1 and TX2 are available. The value read from TFLG is therefore $06. When writing this value back to TBSEL, the TX buffer (TX1) is selected in the TXFG register blocks because the lowest numbered bit set to one is at bit position one. Reading back this value out of CANTBSEL results in $02, because only the lowest numbered bit position set to one is presented. This mechanism eases the application software the selection of the next available TX buffer. If all transmit message buffers are deselected, accesses are not allowed to the TXFG register block.	
		0	The associated message buffer is deselected
		1	The associated message buffer is selected, if lowest numbered bit

Transmit Buffer Selection (TBSEL) Register Base + $A	Bits	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	TX2	TX1	TX0
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

MSCAN

Identifier Acceptance Control (IDAC) Register

Bits	Name	Description
5 - 4	IDAM	**Identifier Acceptance Mode**
		The CPU sets these bits to define the identifier acceptance filter organization. **Section 9.5.2** provides additional information while **Table 9-10** summarizes the different settings. In Filter Closed mode, no message is accepted, therefore the foreground buffer is never reloaded.
2 - 0	IDHIT	**Identifier Acceptance Hit Indicator**
		The CAN sets these bits to indicate an identifier acceptance hit. Please see **Section 9.5.2** while **Table 9-11** summarizes the different settings.

Identifier Acceptance Ctrl. (TBSEL) Register Base + $B	Bits	7	6	5	4	3	2	1	0
	Read	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Miscellaneous (MISC) Register

Bits	Name	Description
0	BOHOLD	**Bus-Off State Hold Until User Request**
		If BORM bit in CTRL1 register is set, the BHOLD bit indicates whether the module has entered the bus-off state. Please refer to **Section 9.9.2** for details concerning the BORM bit. Clearing this bit requests the recovery from bus-off. See **Section 9.6.2** for details.
		0 Module is not bus-off or recovery has been requested by user in bus-off state
		1 Module bus-off and holds this state until user request

Miscellaneous (MISC) Register Base + $D	Bits	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	BOHOLD
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

MSCAN

Receive (RXERR) and Transmit Error Counter (TXERR) Registers

Bits	Name	Description
7 - 0	RXERR	**Receive Error**
		This register reflects the status of the CAN Receive Error Counter. This register is *read-only* when in Sleep mode (SLPRQ=1 and SLPAK=1) or Initialization mode (INITRQ=1 and INITAK=1) and cannot be modified by writing. Reading this register when in any other mode other than Sleep or Initialization modes may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition. Writing to this register when in special modes can alter the CAN functionality.

Receiver Error Counter (RXERR) Register Base + $E	Bits	7	6	5	4	3	2	1	0
	Read	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
	Write								
	Reset	0	0	0	0	0	0	0	0

Bits	Name	Description
7 - 0	TXERR	**Transmit Error**
		This register reflects the status of the CAN Transmit Error Counter. This register is *read-only* when in Sleep mode (SLPRQ=1 and SLPAK=1) or Initialization mode (INITRQ=1 and INITAK=1). It cannot be modified by writing. Reading this register when in any other mode other than Sleep or Initialization modes may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition. Writing to this register when in special modes can alter the CAN functionality.

Transmit Error Counter (TXERR) Register Base + $F	Bits	7	6	5	4	3	2	1	0
	Read	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
	Write								
	Reset	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Identifier Acceptance Registers (IDAR0-7)

Bits	Name	Description
7 - 0	AC*n*	**Acceptance Code 7-0**
		AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related Identifier Register (IDR*n*) of the receive message buffer, are compared. The result of this comparison is then masked with the corresponding Identifier Mask register.

2nd Bank Identifier Acceptance Registers (IDAR4-7) Base + $18 - $1B	Bits	7	6	5	4	3	2	1	0
	Read	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	Write								
	Reset	0	0	0	0	0	0	0	0

Bits	Name	Description
7 - 0	AC*n*	**Acceptance Code 7-0**
		AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related Identifier Register (IDR*n*) of the receive message buffer, are compared. The result of this comparison is then masked with the corresponding Identifier Mask register.

1st Bank Identifier Acceptance Registers (IDAR0-3) Base + $10 - $13	Bits	7	6	5	4	3	2	1	0
	Read	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	Write								
	Reset	0	0	0	0	0	0	0	0

MSCAN

Identifier Mask 0-7 Registers (IDMR0-7)

Bits	Name	Description
7 - 0	AM*n*	**Acceptance Mask 7-0**
		If a particular bit in this register is cleared, this indicates the corresponding bit in the Identifier Acceptance Register (IDAR) 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 the state of the corresponding bit in the IDAR 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

1st Bank Identifier Mask 0-3 Registers (IDMR0-3) Base + $14 - $17	Bits	7	6	5	4	3	2	1	0
	Read	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	Write								
	Reset	0	0	0	0	0	0	0	0

Bits	Name	Description
7 - 0	AM*n*	**Acceptance Mask 7-0**
		If a particular bit in this register is cleared, this indicates the corresponding bit in the Identifier Acceptance Register (IDAR) 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 the state of the corresponding bit in the IDAR 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

2nd Bank Identifier Mask 4-7 Registers (IDMR4-7) Base + $1C - $1F	Bits	7	6	5	4	3	2	1	0
	Read	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	Write								
	Reset	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

MSCAN

Extended Identifier Registers (IDR0-3)

Please see the following page for continuation of this register

Bits	Name	Description
7 - 0	ID*n*	**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.

Extended Identifier Register 0 (IDR0-3) Base + $0	Bits	7	6	5	4	3	2	1	0
	Read	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	Write								
	Reset	x	x	x	x	x	x	x	x

MSCAN

Extended Identifier Registers (IDR0-3) Continued

Please see the following page for continuation of this register

Bits	Name	Description
7 - 5	ID*n*	**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	**Substitute Remote Request**
		This fixed recessive bit is used only in extended format. It must be set to one for transmission buffers and is stored as received on the CAN bus for receive buffers.
3	IDE	**ID Extended**
		This bit indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the bit is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the bit indicates to the CAN what type of identifier to send.
		0 Standard format (11-bit)
		1 Extended format (29-bit)
2 - 0	ID*n*	**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.

Extended Identifier Register 1 (IDR0-3) Base + $1	Bits	7	6	5	4	3	2	1	0
	Read	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
	Write								
	Reset	x	x	x	x	x	x	x	x

Application: _____ Date: _____

Programmer: _____

MSCAN

Extended Identifier Registers (IDR0-3) Continued

Please see the following page for continuation of this register

Bits	Name	Description
7 - 0	ID*n*	**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.

Extended Identifier Register 2 (IDR0-3) Base + $2	Bits	7	6	5	4	3	2	1	0
	Read	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
	Write								
	Reset	x	x	x	x	x	x	x	x

MSCAN

Extended Identifier Registers (IDR0-3) Continued

Bits	Name	Description
7 - 1	ID*n*	**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

Extended Identifier Register 3 (IDR0-3) Base + $3	Bits	7	6	5	4	3	2	1	0
	Read	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
	Write								
	Reset	x	x	x	x	x	x	x	x

Application: _____ Date: _____

Programmer: _____

MSCAN

Standard Identifier 0 (IDR0) Register

Bits	Name	Description
7 - 0	ID*n*	**Identifier**
		The identifiers consist of 11 bits (ID[10:0]) for the extended format. ID11 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. See also ID bits in **Section 9.9.25**.

Standard Identifier Register 0 (IDR0) Base + $0	Bits	7	6	5	4	3	2	1	0
	Read	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
	Write								
	Reset	x	x	x	x	x	x	x	x

Application: _____ Date: _____

Programmer: _____

MSCAN

Standard Identifier Register 1 (IDR1)

Bits	Name	Description
7 - 5	**ID*n***	**Identifier**
		The identifiers consist of 11 bits (ID[10:0]) for the extended format. ID11 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. See also ID bits in **Section 9.9.25**.
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**	**ID Extended**
		This bit indicates whether the extended or standard identifier format is applied in this buffer. In the case of a Receive Buffer, the bit is set as received and indicates to the CPU how to process the Buffer Identifier registers. In the case of a Transmit buffer, the bit indicates to the CAN what type of identifier to send.
		0 Standard format (11-bit)
		1 Extended format (29-bit)

Standard Identifier Register 1 (IDR1) Base + $1	Bits	7	6	5	4	3	2	1	0
	Read	ID2	ID1	ID0	RTR	IDE (=0)			
	Write								
	Reset	x	x	x	x	x	x	x	x

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Data Segment Registers 0-7 (DSR0-7)

Bits	Name	Description
7 - 0	DB*n*	**Data Bit**
		The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

Data Segment Registers 0-7 (DSR0-7) Base + $23 - $2A	Bits	7	6	5	4	3	2	1	0
	Read	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	Write								
	Reset	x	x	x	x	x	x	x	x

MSCAN

Data Length Register (DLR)

Bits	Name	Description
3 - 0	DLC*n*	**Data Length Code**
		These bits contain 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. **Table 9-13** shows the effect of setting the DLC bits.

Data Length Register (DLR) Base + $C	Bits	7	6	5	4	3	2	1	0
	Read					DLC3	DLC2	DLC1	DLC0
	Write								
	Reset	x	x	x	x	x	x	x	x

Reserved Bits

Application: _____ Date: _____

Programmer: _____

MSCAN

Transmit Buffer Priority (TBPR) Register

Bits	Name	Description
7 - 0	PRIO*n*	**Priority**
		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 TXE*n* bit 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 TXE*n* bit is set and the corresponding transmit buffer is selected in TBSEL. Please see **Section 9.9.7** and **Section 9.9.11**. This is a write anytime register when TXE*n* bit is set (see **Section 9.9.7**) and the corresponding transmit buffer is selected in TBSEL. Please see **Section 9.9.7** and **Section 9.9.11**.

Transmit Buffer Priority (TBPR) Register Base + $D	Bits	7	6	5	4	3	2	1	0
	Read	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
	Write								
	Reset	x	x	x	x	x	x	x	x

Application: _____ Date: _____

Programmer: _____

MSCAN

Time Stamp Register (TSRH-TSRL)

Bits	Name	Description
7 - 0	TSR*n*	**Time Stamp Register**
		If the TIME bit is enabled, the CAN writes a special time stamp to the respective registers in the active transmit or receive buffer as soon as a message is acknowledged on the CAN bus. Please refer to **Section 9.9.1**. The time stamp is written on the bit sample point for the recessive bit of the ACK delimiter in the CAN frame. In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer is flagged empty. The timer value, used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the CAN. The timer is reset, that is all bits set to zero during Initialization mode. The CPU can only read the Time Stamp Registers (TSR). These registers are read at anytime when TXE*n* bit is set and the corresponding transmit buffer is selected in TBSEL. Please refer to **Section 9.9.7** and **Section 9.9.11**. Writing to these registers is not implemented.

Time Stamp Register High (TSRH) Base + $E	Bits	7	6	5	4	3	2	1	0
	Read	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
	Write								
	Reset	x	x	x	x	x	x	x	x

Time Stamp Register Low (TSRL) Base + $F	Bits	7	6	5	4	3	2	1	0
	Read	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
	Write								
	Reset	x	x	x	x	x	x	x	x

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PS

Control (CTRL) Register

Bits	Name	Description	
1	LVIE27	**2.7V Low Voltage Interrupt Enable**	
		0	Interrupt is disabled
		1	Interrupt is enabled
0	LVIE22	**22V Low Voltage Interrupt Enable**	
		0	Interrupt is disabled
		1	Interrupt is enabled

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LVIE27	LVIE22
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

PS

Status (STAT) Register

Bits	Name	Description
4	LVI	**Low Voltage Interrupt**
		LVI is a sticky status bit derived from the conditions where either of the 2.2 or 2.7 LVI thresholds have been met and associated interrupts are enabled. This bit may be cleared by writing one to it. Writing zero has no effect. It may be set again in the very next clock cycle after being cleared if the voltages are still below the thresholds.
		0 Interrupt not asserted
		1 Interrupt asserted
3	LVIS27S	**Sticky 2.7V Low Voltage Interrupt Source**
		When this bit is set it must be cleared explicitly by writing one to it. Writing zero has no effect. Bit LVIS27S may be set again in the very next clock cycle after being cleared if the condition causing the supply voltage to drop still persists.
		0 2.7V interrupt threshold has not been passed
		1 3.3V supply has dropped below the 2.7V interrupt threshold
2	LVIS22S	**Sticky 2.2V Low Voltage Interrupt Source**
		When this bit is set, it must be cleared explicitly by writing one to it. Writing zero has no effect.
		0 2.2V interrupt threshold has not been passed
		1 2.5V supply dropped below the 2.2V interrupt threshold
1	LVIS27	**Non-Sticky 2.7V Low Voltage Interrupt Source**
		This *read-only* bit resets itself if the supply voltage raises above the comparator threshold point. This bit cannot be modified.
		0 2.7V interrupt threshold has not been passed
		1 3.3V supply has dropped below the 2.7V interrupt threshold
0	LVIS22	**Non-Sticky 2.2V Low Voltage Interrupt Source**
		This *read-only* bit will reset itself if the supply voltage raises above the comparator threshold point. This bit cannot be modified.
		0 2.2V interrupt threshold has not been passed
		1 2.5V supply is below the 2.2V interrupt threshold

Status (STAT) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	LVI	LVIS27S	LVIS22S	LVIS27	LVIS22
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Baud Rate (RATE) Register

Bits	Name	Description
15 - 3	SBR	**Serial Baud Rate**
		The SBR value is from 1 to 8191.
2 - 0	FRAC_SBR	**Fractional Serial Baud Rate**
		This field combines with SBR field (Section 12.6.1.1) to form the divider, determining the baud rate of the QSCI. SBR represents the integer portion of the baud rate divider while FRAC_SBR represents the fractional portion. If the LIN Mode bit of CTRL2 register is set, the value of this register is automatically adjusted to match the data rate of the LIN master deice. Reading this register yields the auto-baud value set.

Baud Rate (RATE) Register Base + $0	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Control 1 (CTRL1) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	LOOP	**Loop Select**
		This bit enables loop operation. Please see **Section 12.5.2**. The loop operation disconnects the RXD pin from the QSCI 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 only requires one or the other to be enabled. Please see **Section 12.5.1** and **Table 12-9**. The receiver input is determine by the RSRC bit. The transmitter output is controlled by the TE bit. If the TE bit is set and LOOP=1, the transmitter output appears on the TXD pin. If the TE bit is clear and LOOP=1, the TXD pin is high-impedance.
		0 Normal operation enabled
		1 Loop operation enabled
14	SWAI	**Stop in Wait Mode**
		The SWAI bit disables the QSCI in the Wait mode. Please see **Section 12.5.4.2**
		0 QSCI enabled in Wait mode
		1 QSCI disabled in Wait mode
13	RSRC	**Receiver Source**
		When LOOP=1, RSRC bit determines the internal feedback path for the receiver, indicated in **Table 12-9**.
		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

Control 1 (CTRL1) Register Base + $1	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Control 1 (CTRL1) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
11	WAKE	**Wake-Up Condition**
		This bit determines which condition wakes up the QSCI.
		0 Idle Line Wake-Up
		1 Address Mark Wake-Up (a Logic 1 in the MSB position of a receive data character)
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.
		0 Doesn't invert transmit and receive data bits (Normal mode)
		1 Invert transmit and receive data bit (Inverted mode)
9	PE	**Parity Enable**
		This bit enables the parity function. When enabled, the parity function replaces the MSB 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 QSCI generates and checks for even or odd parity of the data bits. With *even parity,* an *even* number of *ones clear* the PARITY bit while an *odd* number of *ones, sets* the PARITY bit. However, with *odd parity,* an *odd* number of *ones clear* the PARITY bit while an *even* number of *ones, sets* the PARITY bit.
		0 Even parity
		1 Odd parity

Control 1 (CTRL1) Register Base + $1	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Control 1 (CTRL1) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
7	TEIE	**Transmitter Empty Interrupt Enable**
		This bit enables the TDRE bit to generate interrupt requests.
		0 TDRE interrupt requests disabled
		1 TDRE interrupt requests enabled
6	TIIE	**Transmitter Idle Interrupt Enable**
		This bit enables the TIDLE bit to generate interrupt requests
		0 TIDLE interrupt requests disabled
		1 TIDLE interrupt requests enabled
5	RFIE	**Receiver Full Interrupt Enable**
		This bit enables the RDRF bit or the OR bit to generate interrupt requests.
		0 RDRF and OR interrupt requests disabled
		1 RDRF and OR interrupt requests enabled
4	REIE	**Receive Error Interrupt Enable**
		This bit enables the Receive Error (RE) bits (NF, PF, LSE, and OR) to generate interrupt requests. The status bits can be checked during the error interrupt process.
		0 Error interrupt requests disabled
		1 Error interrupt requests enabled

Control 1 (CTRL1) Register Base + $1	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Control 1 (CTRL1) Register Continued

Bits	Name	Description
3	TE	**Transmitter Enable**
		This bit enables the QSCI transmitter and configures the TXD pin as the QSCI transmitter output. The TE bit can be used to queue an idle preamble.
		0 Transmitter disabled
		1 Transmitter enabled
2	RE	**Receiver Enable**
		This bit enables the QSCI Receiver.
		0 Receiver disabled
		1 Receiver enabled
1	RWU	**Receiver Wake-Up**
		This bit enables the wake-up function, inhibiting further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU. Please refer to **Section 12.4.4.6** for a description of Receive Wake-Up.
		0 Normal operation
		1 Standby state
0	SBK	**Send Break**
		Toggling SBK sends one break character (10 or 11 Logic 0s) as long as SBK is set, the transmitter sends uninterrupted break characters.
		0 No break characters
		1 Transmit break characters

Control 1 (CTRL1) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

_____ Programmer: _____

QSCI

Control 2 (CTRL2) Register

Bits	Name	Description
15 - 13	TFCNT	**Transmit FIFO Count**
		These *read-only* bits exhibit how many words are used in the TX FIFO. Writes to DATA will cause TFCNT to increment. A decrease takes place as words are pulled for transmission. Attempts to write new data to DATA are ignored when TFCNT indicates FIFO is full (four words).
12 - 11	TFWM	**Transmit FIFO Empty Watermark**
		These bits set the TX FIFO word count level at which the TDRE flag is set. If FIFO_EN is clear (FIFO disabled), this field is forced to 00 and TDRE is set when there is no word in the transmit buffer.
10 - 8	RFCNT	**Receiver FIFO Count**
		These *read-only* bits illustrate how many words are used in the RX FIFO. Received words cause RFCNT to increment. RFCNT values decrease as words are read from DATA. There is one word-time to read DATA between when the RDRF status bit is set (interrupt asserted) due to the RX FIFO being full and when an overflow condition will be flagged.
7 - 6	RFWM	**Receive FIFO Full Watermark**
		These bits set the RX FIFO word count level at which the RDRF flag is set. If FIFO_EN is clear (FIFO disabled), then this field is forced to 00 and RDRF is set if there is a word in the Receive Buffer.
3	LIN MODE	**Local Interconnect Network Mode**
		This bit should only be used in Local Interconnect Network (LIN) applications**.**
	0	The LIN auto baud feature is disabled and the SBR register will maintain whatever value the processor writes to it.
	1	Enable a search for the break followed by sync char (0x55) from the master LIN device. When the break is detected the following sync character will be used to measure the baud rate of the transmitting master and the SBR register will be automatically reloaded with the value needed to *match* that baud rate.

Control 2 (CTRL2) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TFCNT			TFWM		RFCNT			RFWM		FIFO_EN	0	LIN MODE	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Status (STAT) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	TDRE	**Transmit Data Register Empty Flag**
		This bit is set when the Transmit Shift register receives a character from the DATA register. Clear TDRE by reading the STAT register, then write to the DATA register.
		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 TDRE bit is set and no data, preamble, or break character is being transmitted. When TIDLE is set, the TXD pin becomes idle (Logic 1). Clear TIDLE by reading the STAT register with TIDLE set and then writing to the 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**
		RDRF is set when the RX FIFO word count (RFCNT) rises above the watermark (RXWM). Clear RDRF by reading the Status (STAT) register with RDRF set and then reading the EMI data register until RFCNT is no longer above RFWM. If FIFO_EN or RDE is set, then you can clear RDRF by reading the SCI data register without first reading STAT with RDRF set.
		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 ten consecutive Logic 1s (if M=0) or 11 consecutive Logic 1s (if M=1) appear on the receiver input. The RIDLE flag is cleared by reading the STAT register with RIDLE set followed by reading the DATA register. Once the RIDLE flag is cleared, a valid frame must be received before an idle condition can set the RIDLE flag.
		0 Receiver input is either active now or was never active since the RIDLE bit was last cleared by reset
		1 Receiver input is now idle (after receiving a valid frame)

Status (STAT) Register Base + $3	Bits	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	0	0	LSE	0	0	RAF
	Write																
	Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Status (STAT) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
11	OR	**Overrun**
		This bit is set when software fails to read the 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 DATA /FIFO register is not affected. Clear OR by reading the STAT register with OR set and then write the STAT register with any value.
		0 No overrun
		1 Overrun
10	NF	**Noise Flag**
		This bit is set when the QSCI detects noise on the receiver input. The NF bit is set during the same cycle as the RDRF flag, but it is not set in the case of an overrun. Clear NF by reading the STAT register, and then write the STAT register with any value.
		0 No noise
		1 Noise
9	FE	**Framing Error**
		This bit is set when a Logic 0 is accepted as the STOP bit. The FE bit is set during the same cycle as the RDRF flag, but it is not set in the case of an overrun. Clear FE by reading the STAT register with FE set and then write the STAT register with any value.
		0 No framing error
		1 Framing error
8	PF	**Parity Error Flag**
		This bit is set when the Parity Enable (PE) bit is set and the parity of the received data does not match its parity bit. Clear PF by reading the STAT register, and then write the STAT register with any value.
		0 No parity error
		1 Parity error

Status (STAT) Register Base + $3	Bits	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	0	0	LSE	0	0	RAF
	Write																
	Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSCI

Status (STAT) Register Continued

Bits	Name	Description
3	**LSE**	**Local Interconnect Network Sync Error**
		This bit is only active when LIN MODE is set. When LSE is set, an RERR interrupt will occur if REIE is set. LSE is set when a LIN sync search detects a non-sync character (anything other than 0x55). Having this bit set indicates either a protocol error was detected from the Master LIN device or there is a gross mis-match in data rates. This bit is cleared by reading the STAT register with LSE set and then writing the STAT register with any value.
		0 No error occurred since the LIN MODE was enabled or the bit was last cleared.
		1 A sync error prevented loading of the SBR with a revised value after the break was detected.
0	**RAF**	**Receiver Active Flag**
		This bit is set when the receiver detects a Logic 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

Status (STAT) Register Base + $3	Bits	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	0	0	LSE	0	0	RAF
	Write																
	Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

QSCI

Data (DATA) Register

Bits	Name	Description
8 - 0	RECEIVE/TRANSMIT	**Receive/Transmit Data**
		Reading these bits accesses the receive data. Writing to these bits loads the transmit data. When the TDRE bit is set, writes to the Transmit Data field are ignored unless the STAT register is read while TDRE is set. When FIFO_EN or TDE is set it is possible to write to the DATA register without first reading STAT with TDRE set. If the RDRF bit is set, reads from the Receive Data field are ignored unless the STAT register is read while RDRF is set. When FIFO_EN or RDE is set it is possible to read from the DATA register without first reading STAT with RDRF set.

Data (DATA) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	RECEIVE DATA								
	Write								TRANSMIT DATA								
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Status and Control (SCTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15 - 13	SPR	**SPI Baud Rate Select**
		While in the Master mode, these read/write bits select one of eight baud rates depicted in **Table 13-5**. SPR bit field has no effect in Slave mode. Reset sets SPR bits to $3. Use the formula below to calculate the QSPI baud rate.
12	DSO	**Data Shift Order**
		This read/write bit determines whether the MSB or LSB is transmitted or received first. Both Master and Slave QSPI modules must transmit and receive the same length packets. Regardless how this bit is set, when reading from the Data Receive (DRCV) register, or writing to the Data Transmit (DXMIT) register, the LSB is always be at bit location zero and the MSB is at the correct bit position. If the data length is less than 16 bits, the data is 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 controller interrupt requests. Reset clears the ERRIE bit.
		0 MODF and OVRF cannot generate controller interrupt requests
		1 MODF and OVRF can generate controller interrupt requests
10	MODFEN	**Mode Fault Enable**
		When set to one, this read/write bit allows the MODF bit to be set. If the MODF bit is set, clearing the MODFEN does not clear the MODF bit. If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled QSPI configured as a master. If configured as a master and MODFEN=1, a transmission in progress stops if $\overline{SS}$ goes low. For an enabled QSPI configured as a slave, having MODFEN low only prevents the MODF bit from being set. It does not affect any other part of QSPI operation.
9	SPRIE	**QSPI Receiver Interrupt Enable**
		This read/write bit enables interrupt requests generated by the QSPI Receiver Full (SPRF) bit or the receive FIFO watermark register.
		0 SPRF interrupt requests disabled
		1 SPRF interrupt requests enabled

Status and Control (SCTRL) Register Base + $0	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Status and Control (SCTRL) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
8	SPMSTR	**QSPI Master**
		This read/write bit selects Master or Slave modes of operation.
		0 Slave mode
		1 Master mode
7	CPOL	**Clock Polarity**
		This read/write bit determines the logic state of the SCLK pin between transmissions. To transmit data between QSPI modules, the QSPI modules must have identical CPOL values. Please see **Figure 13-4** and **Figure 13-6**.
		0 SCLK idle state between transmission
		1 SCLK idle state between transmission
6	CPHA	**Clock Phase**
		This read/write bit controls the timing relationship between the serial clock and QSPI data. Please see **Figure 13-4** and **Figure 13-5**, both in this manual. To transmit data between QSPI modules, the SPI modules must have identical CPHA values. When CPHA=0, the $\overline{SS}$ pin of the Slave QSPI module must be set to Logic 1 between data transmissions, illustrated in **Figure 13-4** in this manual.
5	SPE	**QSPI Enable**
		This read/write bit enables the QSPI module. Clearing SPE causes a partial reset of the QSPI. When setting/clearing this bit, no other bits in the SCTRL register should be changed. In Master mode the SPE bit can be cleared by a mode fault condition. Failure to follow this statement may result in spurious clocks.
		0 QSPI module disabled
		1 QSPI module enabled
4	SPTIE	**QSPI Transmit Interrupt Enable**
		This read/write bit enables interrupt requests generated by the Transmitter Empty (SPTE) bit, or the transmit FIFO watermark register.
		0 SPTE interrupt requests disabled
		1 SPTE interrupt requests enabled

Status and Control (SCTRL) Register Base + $0	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Status and Control (SCTRL) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
3	SPRF	**SPI Receiver Full**
		This is a *read-only* status bit. The way SPRF is detected depends on the setting of FIFO enable (FIFO_EN). SPRF generates an interrupt request if the Receiver Interrupt Enable (SPRIE) bit in the SCTRL register is set. This bit is automatically cleared by reading the DRCV register.
	0	Data Receive (DRCV) register has no new data, or the Receive Data FIFO Level (RFCNT) is less than the Receive Data FIFO watermark (RFWM). (If using the FIFO, read the RFCNT bit field to determine the number of valid words available.)
	1	Data Receive (DRCV) register has new data, or the Receive Data FIFO (RFCNT) level is greater than, or equal to the Receive Data FIFO watermark (RFWM).
2	OVRF	**Overflow**
		This is a *read-only* status bit. It is set if software does not read the data in the DRCV register before the next full data enters the Shift register. In an overflow condition, the data already in the DRCV register is unaffected, and the data shifted in last is lost. Clear the Overflow (OVRF) bit by reading the SCTRL register and then reading the DRCV register. OVRF generates a Receiver/Error interrupt if the Error Interrupt Enable (EERIE) bit is set. See **Section 13.8.1** for more details.
	0	No overflow
	1	Overflow
1	MODF	**Mode Fault**
		This *read-only* status bit is capable of being cleared. It is set in a slave QSPI if the $\overline{SS}$ pin goes high during a transmission with the MODFEN bit set. In a master QSPI, the MODF bit is set if the $\overline{SS}$ pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing one to the MODF bit when it is set. MODF generates a Receive/Error interrupt if the EERIE bit is set. Please see **Section 13.8.2** for more details.
	0	$\overline{SS}$ pin at appropriate Logic level
	1	$\overline{SS}$ pin at inappropriate Logic level

Status and Control (SCTRL) Register Base + $0	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Status and Control (SCTRL) Register Continued

Bits	Name	Description	
0	SPTE	**QSPI Transmitter Empty**	
		This is a *read-only* status bit. The way SPTE is detected depends on the setting of FIFO Enable (FIFO_EN). SPTE generates an interrupt request if the Transmit Interrupt Enable (SPTIE) bit in the SCTRL register is set. SPTE is automatically cleared by writing to the DXMIT register.	
		0	DXMIT register was not read by the Shift register or the transmit FIFO level (TFCNT) is greater than the TxFIFO watermark (TFWM). (If using the FIFO, read the TFCNT bit field to determine how many words can be written safely.)
		1	DXMIT register was read by the Shift register or the transmit FIFO level (TFCNT) is less than, or equal to the TxFIFO watermark (TFWM).

Status and Control (SCTRL) Register Base + $0	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Data Size and Control (DSCTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	WOM	**Wired OR Mode**
		This control bit selects the nature of the QSPI pins. When the WOM bit is set, the QSPI pins are configured as open-drain drivers. When the WOM bit is cleared, the QSPI pins are configured as push-pull drivers.
		0 Wired OR mode disabled
		1 Wired OR mode enabled
14	TDMAEN	**Transmit DMA Enable**
		This read/write bit enables DMA control for transmit data. This function is only supported on devices including DMA functionality. In devices not supporting DMA, it is a *read-only* bit and is always read as zero.
13	RDMAEN	**Receive DMA Enable**
		This read/write bit enables DMA control for receive data. This function is only supported on devices including DMA functionality. In devices not supporting DMA, it is a *read-only* bit as is always read as zero.
12	BD2X	**Baud Divisor x 2**
		When set, the Baud Rate Divisor bit is multiplied by two. Please see **Table 13-5**.
11	$\overline{SS}$_IN	**Slave Select Input**
		This *read-only* bit shows the current state of the $\overline{SS}$ pin in all modes.
10	$\overline{SS}$_DATA	**Slave Select Data**
		This read/write bit is the value to drive on the $\overline{SS}$ pin. This bit is disabled when $\overline{SS}$_AUTO=1 or $\overline{SS}$_STRB=1.
		0 $\overline{SS}$ pin is driven low if $\overline{SS}$ _DDR=1
		1 $\overline{SS}$ pin is driven high if $\overline{SS}$ _DDR=1

Data Size and Control (DSCTRL) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	WOM	TDMAEN	RDMAEN	BD2X	$\overline{SS}$_IN	$\overline{SS}$_DATA	$\overline{SS}$_ODM	$\overline{SS}$_AUTO	$\overline{SS}$_DDR	$\overline{SS}$_STRB	$\overline{SS}$_OVER	0	DS3	DS2	DS1	DS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

QSPI

Data Size and Control (DSCTRL) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
9	SS_ODM	**Slave Select Open Drain Mode**
		This read/write bit enables Open Drain mode on the SS pin in Master mode.
	0	SS is configured for high and low drive. This mode is generally used in single master systems.
	1	SS is configured as an open drain pin (only drives low output level). This mode is useful for multiple master systems.
8	SS_AUTO	Slave Select Auto
		This read/write bit enables hardware control of the SS pin in master mode. (The legacy design requires software to control the SS output pin.) The initial falling edge of SS is generated and SS is held low until the TX buffer, or the FIFO is empty. This bit may be used alone or in combination with the SS_STRB in order to generate the required SS signal.
	0	SS output signal is software generated by directly manipulating the various bits in this register (SS_DATA) or the GPIO registers (compatible with legacy QSPI software).
	1	SS output signal is hardware generated to create the initial falling edge and final rising edge. The idle state of the SS is high.
7	SS_DDR	Slave Select Data Direction
		This read/write bit controls input/output mode on the SS pin in master mode.
	0	SS is a configured as an input pin. Use this setting in Slave or Master modes with MODFEN=1.
	1	SS output signal is pulsed high between words. This adds 1.5 baud clocks to the total word period. The idle state of the SS is low unless SS_AUTO is high and then the idle state is high.
6	SS_STRB	Slave Select Strobe Mode
		This read/write bit enables hardware pulse of the SS pin in Master mode between words. This bit may be used alone or in combination with the SS_AUTO to generate the required SS signal. Pulses are generated between words irrespective of the setting of CPHA.
	0	No SS pulse between words
	1	SS output signal is pulsed high between words. This adds 1.5 baud clocks to the total word period. The idle state of the SS is low unless SS_AUTO is high and then the idle state is high.

Data Size and Control (DSCTRL) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	WOM	TDMA EN	RDMA EN	BD2X	SS_IN	SS_ DATA	SS_ ODM	SS_ AUTO	SS_ DDR	SS_ STRB	SS_ OVER	0	DS3	DS2	DS1	DS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Data Size and Control (DSCTRL) Register Continued

Bits	Name	Description
5	$\overline{SS}$_ORR	**Slave Select Override**
		This read/write bit overrides the internal $\overline{SS}$ signal input from the I/O pad and replaces it with a level equal to the setting of the SPMSTR bit. This allows the QSPI to function in Slave mode, CPHA=1, without committing 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, therefore this bit should not be used in a multi-master system.
		0 $\overline{SS}$ internal module input is selected to be connected to a GPIO pin
		1 $\overline{SS}$ internal module input is selected to be equal to SPMSTR
3 - 0	DS*n*	**Data Size**
		This read/write register determines the data length for each transmission. The master and slave must transfer the same size data on each transmission. A new value only takes effect at the time the QSPI is enabled (SPE bit in SCTRL set from zero to one). In order to have a new value take effect, disable then re-enable the QSPI with the new value in the register. Please see **Table 13-7** for detailed transmission data size.

Data Size and Control Register (DSCTRL) Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	WOM	TDMA EN	RDMA EN	BD2X	$\overline{SS}$_IN	$\overline{SS}$_ DATA	$\overline{SS}$_ ODM	$\overline{SS}$_ AUTO	$\overline{SS}$_ DDR	$\overline{SS}$_ STRB	$\overline{SS}$_ OVER	0	DS3	DS2	DS1	DS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Data Receive (DRCV) Register

Bits	Name	Description
15 - 0	R*n*	**Receive *n***
		This *read-only* register provides the last data word received after a complete transmission. When SPRF bit in the SCTRL register is set, new data can be read from this register.

Data Receive (DRCV) Register Base + $2	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

Data Transmit (DXMIT) Register

Bits	Name	Description
15 - 0	T*n*	**Transmit *n***
		This *write-only* register holds data to be transmitted. When the Transmitter Empty (SPTE) bit is set, new data should be written to this register. If new data is not written while in the Master mode, a new transaction is not initiated until this register is written. In Slave mode, the old data re-transmits if DXMIT is not rewritten before the next transmission begins. All data should be written with the LSB at bit zero. This register can only be written when the QSPI is enabled, SPE=1.

Data Transmit (DXMIT) Register Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	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

Reserved Bits

QSPI

FIFO Control (FIFO) Register

Please see the following page for continuation of this register

Bits	Name	Description
14 - 12	TFCNT	**Transmit FIFO Level**
		This *read-only* bit field indicates the number of words are used in the Tx FIFO. Writes to the DXMIT causes TFCNT to increase, as words are pulled for transmission TFCNT is decreased. Attempts to write new data to DXMIT register are ignored when TFCNT indicates the FIFO is full. If FIFO_ENA is set to zero, the TFCNT bit always indicates FIFO empty. If Master mode is enabled, transmission continues until the FIFO is empty, even if SPE is set to zero.
10 - 8	RFCNT	**Receive Data FIFO Level**
		This *read-only* bit field indicates the number of words used in the RX FIFO. As words are received RFCNT increases, as words are read from the DRCV register the value of RFCNT are decreased. There is one word time to read DRCV register between when the SPRF status bit is set (interrupt asserted) and when a overflow condition is flagged. If FIFO_ENA is set to zero the RFCNT always indicates FIFO is empty.
6 - 5	TFWM	**Transmit Data FIFO Watermark**
		These read/write bits determine the number of words required to 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 under running 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 QSPI word time in interrupt service latency is allowed before an under run 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 DXMIT register, or the value of TFWM reduced.

FIFO Control (FIFO) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	TFCNT			0	RFCNT			0	TRWM		0	RFWM		0	FIFO_ENA
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

QSPI

FIFO Control (FIFO) Register Continued

Bits	Name	Description
3 - 2	RFWM	**Receive Data FIFO Watermark**
		These read/write bits determine the number of words required to be used in the RXFIFO 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, only one QSPI 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 bit, words must be read from DRCV register, or the value of RFWM must be increased.
0	FIFO_ENA	**FIFO Enable**
		This read/write bit enables TX and RX FIFO's mode.
		0 FIFOs are disabled and reset. Operation is compatible with legacy SPI.
		1 FIFOs are enabled and retain their status even if SPE is set to zero

FIFO Control (FIFO) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	TFCNT			0	RFCNT			0	TRWM		0	RFWM		0	FIFO_ENA
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

QSPI

Word Delay (DELAY) Register

Bits	Name	Description
12 - 0	WAIT	**Wait Delay**
		This bit field controls the time between data transactions in Master mode. The value is the number of Peripheral Bus Clocks to delay between words.

Word Delay (DELAY) Register Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	WAIT												
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

TMR

Compare 1 (COMP1) Register

Bits	Name	Description
15 - 0	COMPARISON1	**Comparison 1**
		This read/write Timer Compare 1 (COMP1) register stores the value used for comparison with counter value.

Compare 1 (COMP1) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARISON 1															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR

Compare 2 (COMP2) Register

Bits	Name	Description
15 - 0	COMPARISON 2	**Comparison 2**
		This read/write Timer Compare 2 (COMP2) register stores the value used for comparison with counter value.

Compare 2 (COMP2) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARISON 2															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Capture (CAPT) Register

Bits	Name	Description
15 - 0	CAPTURE	Capture
		This read/write Timer Capture (CAPT) register stores the values captured from the counters. Please see Section 14.6.8.9 for information on which inputs generate a capture of the counter's value.

Capture (CAPT) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CAPTURE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Load (LOAD) Register

Bits	Name	Description
15 - 0	LOAD	**Load**
		This read/write Timer Load (LOAD) register stores the values used to load the counter.

Load (LOAD) Register Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOAD															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Hold (HOLD) Register

Bits	Name	Description
15 - 0	HOLD	**Hold**
		This read/write register stores the channel's value whenever any Timer Counter (CNTR) register is read.

Hold (HOLD) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HOLD															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Counter (CNTR) Register

Bits	Name	Description
15 - 0	COUNTER	Counter
		This is a read/write Timer Counter (CNTR) register.

Counter (CNTR) Register Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COUNTER															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Control (CTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15 - 13	CM	**Count Mode**
		This bit field controls the basic counting behavior of the counter. See **Section 14.5.12.1** for more details.
		000 Stop mode, no operation
		001 Counting mode, count rising edges of Primary Source[1] –Rising edges if IPS=0 (See **Section 14.6.8.7**) –Falling edges if IPS=1 (See **Section 14.6.8.7**)
		010 Edge Count mode, count rising and falling edges of Primary Source
		011 Gated Count mode, count rising edges of Primary Source while secondary input high active
		100 Quadrature Count mode, uses Primary and Secondary Sources
		101 Signed Count mode, count edge of Primary Source –IPS=0 then Secondary Input=0; Count up on rising edges of Primary Input –IPS=0 then Secondary Input=1; Count down on rising edges of Primary Input –IPS=1 then Secondary Input=0; Count up on falling edges of Primary Input –IPS=1 then Secondary Input=1; Count down on falling edges of Primary Input
		110 Triggered Count mode, edge of secondary source triggers primary count until compare
		111 Cascaded Counter mode (up/down)[2]

1. If Primary Count Source is IPBus clock, divide by one. Only rising edges are counted regardless of IPS value.
2. Primary Count Source must be set to one of the counter outputs.

Control (CTRL) Register Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Control (CTRL) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description	
12 - 9	PCS	**Primary Count Source**	
		This bit field selects the primary count source.	
		0000	Counter 0 input pin (T0)
		0001	Counter 1 input pin (T1)
		0010	Counter 2 input pin (T2)
		0011	Counter 3 input pin (T3)
		0100	Counter 0 output (OFLAG0)
		0101	Counter 1 output (OFLAG1)
		0110	Counter 2 output (OFLAG2)
		0111	Counter 3 output (OFLAG3)
		1000	Prescaler (System Clock or 3x System Clock[1] divide by 1)
		1001	Prescaler (System Clock or 3x System Clock[1] divide by 2)
		1010	Prescaler (System Clock or 3x System Clock[1] divide by 4)
		1011	Prescaler (System Clock or 3x System Clock[1] divide by 8)
		1100	Prescaler (System Clock or 3x System Clock[1] divide by 16)
		1101	Prescaler (System Clock or 3x System Clock[1] divide by 32)
		1110	Prescaler (System Clock or 3x System Clock[1] divide by 64)
		1111	Prescaler (System Clock or 3x System Clock[1] divide by 128)

1. 3x System Clock must be selected via SIM_PCR register.

Control (CTRL) Register Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Control (CTRL) Register Continued

Please see the following page for continuation of this register

Bits	Name	Description
8 - 7	SCS	**Secondary Count Source**
		This bit field identifies the external input pin to be used as a count command. The selected input can trigger the timer to capture the current value of the CNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the IPS bit of the SCTRL register.
		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 modes.
		0 Count repeatedly
		1 Count until compare and then stop. If *counting up*, successful compare occurs when the counter reaches a COMP1 value. If *counting down*, successful compare occurs when the counter reaches a COMP2 value.
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 register, or the counter continues counting past the compare value, to the binary roll-over.
		0 Roll-over
		1 Count till compare, then re initialize. If counting up, successful compare occurs when the counter reaches a COMP1 value. If counting down, successful compare occurs when the counter reaches a COMP2 value.
4	DIR	**Count Direction**
		This bit selects either the normal count direction *up*, or the reverse *down* direction.
		0 Count up
		1 Count down

Control (CTRL) Register Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR

Control (CTRL) Register Continued

Bits	Name	Description
3	CO INIT	**Co-Channel Initialization**
		This bit enables another counter/timer within the same module to force the re initialization of this counter/timer when it has an active compare event. The Master channel forcing re initialization has MSTR bit set. Please see **Section 14.6.8.10** in this manual.
		0 Co-Channel counter/timers cannot force a re initialization of this counter/timer
		1 Co-Channel counter/timers can force a re initialization of this counter/timer
2 - 0	OM	**Output Mode**
		This bit field determines the mode of operation for the OFLAG output signal. Unexpected results may occur if OM field is set to use alternating Compare registers (Mode 100) and the ONCE bit are set.
		000 Assert OFLAG 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 OFLAG on compare, cleared on secondary source input edge
		110 Set OFLAG on compare, cleared on counter roll-over
		111 Enable Gated Clock output while counter is active

Control (CTRL) Register Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Status and Control (SCTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	TCF	**Timer Compare Flag**
		This bit is set when a successful compare occurs. Clear the bit by writing zero to it. TCF asserts every time there is a compare event (either when counter=COMP1 or counter=COMP2).
14	TCFIE	**Timer Compare Flag Interrupt Enable**
		When set, the timer compare interrupt is enabled.
13	TOF	**Timer Overflow Flag**
		This bit is set when the counter rolls over its maximum or minimum value $FFFF or $0000, depending on count direction. Clear the bit by writing zero to it.
12	TOFIE	**Timer Overflow Flag Interrupt Enable**
		When set, this bit enables interrupts when the TOF bit is set.
11	IEF	**Input Edge Flag**
		This bit is set when a transition occurs on an input selected as a secondary count source while the counter is enabled. Clear the bit by writing zero to it. Setting the IPS bit enables the detection of negative input edge transitions detection. Also, the Control register's Secondary Count Source (SCS) determines which external input pin is monitored by the detection circuitry.
10	IEFIE	**Input Edge Flag Interrupt Enable**
		When set, this bit enables interrupts if the IEF bit is set.
9	IPS	**Input Polarity Select**
		When set, this bit inverts the polarity of both the primary and secondary inputs.
8	INPUT	**External Input Signal**
		The *read-only* bit reflects the current state of the external input pin selected via the secondary count source after application of the IPS bit and filtering.

Status and Control (SCTRL) Register Base + $7	Bits	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	0	0	0	0	0	0	0	0	0

Reserved Bits

TMR

Status and Control (SCTRL) Register Continued

Bits	Name	Description
7 - 6	CAPTURE MODE	**Input Capture Mode**
		This bit field specifies the operation of the CAPT register as well as the operation of the Input Edge Flag (IEF). The input source is the secondary count source.
5	MSTR	**Master Mode**
		When set, this bit enables the Compare register function output to be broadcasted to the other counter/timers in the module. This signal then can be used to re initialize the other counters and/or force their OFLAG signal outputs. Other timer channels within the Quad Timer accept the re initialization signal when their COINIT bit is set. Please see **Section 14.6.7.7** in this manual.
4	EEOF	**Enable External OFLAG Force**
		When set, this bit enables the Compare register 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 a software triggered FORCE command occurs, i.e. when FORCE=1, discussed in **Section 14.6.8.13** in this manual.
2	FORCE	**Force OFLAG Output**
		This *write-only* bit forces the current value of the VAL bit to be written to the OFLAG output. This bit is always read as zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit 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**
		When set, this bit determines the direction of the external pin.
	0	The external pin is configured as an input
	1	OFLAG output signal will be driven on the external pin. The polarity of the signal will be determined by the OPS bit.

Status and Control (SCTRL) Register Base + $7	Bits	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	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

TMR

Comparator Load 1 (CMPLD1) Register

Bits	Name	Description
15 - 0	COMPARATOR LOAD 1	**Comparator Load 1**
		This read/write register makes up the Comparator 1 preload value for the CMPLD1 register of the corresponding channel in a timer module. Please see **Section 14.4.2** in this manual for additional information. For details regarding the timer settings required to implement variable frequency PWM operation, please refer to **Section 14.5.12.1** in this manual.

Comparator Load 1 (CMPLD1) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARATOR LOAD 1															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR

Comparator Load 2 (CMPLD2) Register

Bits	Name	Description
15 - 0	COMPARATOR LOAD 2	**Comparator Load 2**
		This read/write register makes up the Comparator 2 preload value for the CMPLD2 register of the corresponding channel in a timer module. Please see **Section 14.4.2** in this manual for additional information. For details regarding how to control the loading of Comparator 2, please refer to **Section 14.6.11.6** in this manual.

Comparator Load 2 (CMPLD2) Register Base + $9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARATOR LOAD 2															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

TMR

Comparator Status and Control (CSCTRL) Register

Bits	Name	Description
15 - 14	DBG_EN	**Debug Actions Enable**
		This bit field allows the TMR module to perform certain actions in response to the chip entering Debug mode.
7	TCF2EN	**Timer Compare 2 Interrupt Enable**
		An interrupt is issued when both this bit and the TCF2 bit are set.
6	TDF1EN	**Timer Compare 1 Interrupt Enable**
		An interrupt is issued when both this bit and the TDF1 bit are set.
5	TCF2	**Timer Compare Flag 2**
		When set, this bit indicates a successful comparison of the timer and COMP2 register occurred. This bit is sticky, and will remain set until explicitly cleared by writing zero to this bit location.
4	TCF1	**Timer Compare Flag 1**
		When set, this bit indicates a successful comparison of the timer and COMP1 register occurred. This bit is sticky, and will remain set until explicitly cleared by writing zero to this bit location.
3 - 2	CL2	**Compare Load Control 2**
		This bit field controls when CMP2 is preloaded with the value from CMPLD2.
1 - 0	CL1	**Compare Load Control 1**
		This bit field control when COMP1 is preloaded with the value from CMPLD1.

Comparator Status & Control (CSCTRL) Register Base + $A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DBG_EN		0	0	0	0	0	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

Reserved Bits

TMR

Input Filter (FILT) Register

Bits	Name	Description
10 - 8	FILT_CNT	**Input Filter Sample Count**
		This bit field represents the number of consecutive samples required to agree prior to the input filter accepting an input transition. A value of 0x0 represents three samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency as described in **Section 14.6.12.4**.
7 - 0	FILT_PER	**Input Filter Sample Period**
		This bit field represents the sampling period (in System Clock or 3xSystem Clock[1] cycles) of the TMR input signals. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), the input filter is bypassed. The value of FILT_PER affects the input latency, described in **Section 14.6.12.4**.

1. 3x system clock must be selected via SIM PCR register.

Input Filter (FILT) Register Base + $B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	FILT_CNT			FILT_PER							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

TMR

Channel Enable (ENBL) Register

Bits	Name	Description	
3 - 0	ENBL	**Timer Channel Enable**	
		If the prescaler is being used, this bit field enables it and the counter in each channel. Multiple ENBL bits can be concurrently set to synchronize the start of separate counters. If an ENBL bit is set, the corresponding channel starts the counter as soon as the COUNT MODE field has a value other than zero. When an ENBL bit is clear, the corresponding counter maintains its current value.	
		0	Timer channel is disabled
		1	Timer channel is enabled (default)

Channel Enable (ENBL) Register Base + $F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	ENABLE			
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

VREG

SIM Power Control (SIM_PWR) Register

Bits	Name	Description	
1 - 0	LRSTDBY	**Large Regulator Standby Mode**	
		00	Large regulator is in Normal mode
		01	Large regulator is in Standby (reduced power) mode
		10	Large regulator is in Normal mode and the LRSTDBY field is write-protected until the next reset
		11	Large regulator is in Standby mode and the LRSTDBY field is write-protected until the next reset

Power Control (SIM_PWR) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LRSTDBY	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

PIT

Control (CTRL) Register

Bits	Name	Description
15	SLAVE	**Slave**
		This field is used to place this PIT module in Slave mode. This means the CNT_EN field is ignored and instead this PIT uses the master count enable signal broadcast from the PIT0 module. This bit allows synchronization of the counts across multiple PIT modules. This bit is only useful in designs with multiple PIT modules. Setting this bit in the (master) PIT0 module has no effect as its own CNT_EN field is also the master count enable.
		0 CNT_EN from this PIT is used to control operation (default)
		1 CNT_EN from master PIT is used to control operation
6 - 3	PRESCALER	**Prescaler**
		This field is used to select the prescaling of the system clock to determine the counting rate of the PIT.
2	PRF	**PIT Roll-Over Flag**
		This bit is set when the counter rolls over to $0000 after matching the value in the MOD register. This bit is cleared by reading the CTRL register with PRF set and then writing a zero to this bit position. This bit can also be cleared by setting the CNT_EN bit to zero. Writing one to the PRF bit position has no effect.
		0 PIT counter has not rolled over the modulo value (default)
		1 PIT counter reached the modulo value
1	PRIE	**Pit Roll-Over Interrupt Enable**
		This bit enables the PIT roll-over interrupt when the PRF bit becomes set.
		0 PIT roll-over interrupt disabled (default)
		1 PIT roll-over interrupt enabled
0	CNT_EN	**Count Enable**
		This bit enables the PIT prescaler and counter. When this bit is clear the counter remains at, or returns to a $0000 value. The PRF bit is also reset when CNT_EN is clear. This field is ignored when the SLAVE bit is set and the count enable signal from the master PIT is used instead.
		0 PIT counter reset (default)
		1 PIT counter active

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SLAVE	0	0	0	0	0	0	0	0	PRESCALER				PRF	PRIE	CNT_EN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

PIT

Modulo (MOD) Register

Bits	Name	Description
15 - 0	MODULO_VALUE	**Modulo Value**
		This read/write register stores the modulo value for the PIT counter. When the PIT counter rolls over to $0000 from this value, the PRF bit becomes set and the PIT counter resumes counting from $0000. Changing prescaler value with CNT_EN set may cause unexpected operation.

Modulo (MOD) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	MODULO_VALUE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

PIT

Counter (CNTR) Register

Bits	Name	Description
15 - 0	COUNTER_ VALUE	**Counter Value**
		This *read-only* register contains the current value of the PIT counter. Clearing CNT_EN resets the counter to $0000. When the PIT counter rolls over the modulo value, the PRF bit becomes set and the PIT counter resumes counting from $0000.

Counter (CNTR) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COUNTER_VALUE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

DAC

Control (CTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15 - 13	FILT_CNT	**Glitch Filter Count**
		This bit field represents the number of IPBus clock cycles the DAC output is held unchanged after new data is presented to the analog DACs inputs. Approximately 240nsec is required for worst case settling of the DAC output; therefore, a value of seven should be used for 32MHZ operation. When using the glitch filter, be sure the filter count is less than the update count otherwise the DAC output will not be updated.
12	FILT_EN	**Glitch Filter Enable**
		This bit enables the glitch suppression filter. The glitch suppression filter introduces a latency equivalent to FILT_CNT IPBus clock cycles for DAC updates.
		0 Filter disabled
		1 Filter enabled
5	UP	**Enable Up Counting**
		This bit enables counting up in Automatic mode. Please see **Section 17.4.2** for a description of how this 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. Please see **Section 17.4.2** for a description of how this affects automatic waveform generation.
		0 Down counting disabled
		1 Down counting enabled
3	AUTO	**Automatic Mode**
		This bit enables automatic waveform generation mode. In Automatic mode an external source (typically a TMR or PIT) driving SYNC_IN determines the data update rate while the STEP, MINVAL, and MAXVAL registers and the UP and DOWN bits are used to shape the waveform. Please see **Section 17.4.2** for a full description of automatic waveform generation. If SYNC_EN is not set when using this mode, then the data for the analog DAC will be updated every clock cycle, even though the maximum guaranteed update is 2µsec.
		0 Normal mode, automatic waveform generation disabled
		1 Automatic waveform generation enabled

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FILT_CNT			FILT_EN	0	0	0	0	0	0	UP	DOWN	AUTO	SYNC_EN	FORMAT	PDN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

DAC

Control (CTRL) Register Continued

Bits	Name	Description
2	SYNC_EN	**Synchronization 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 SYNC_EN bit is clear, then asynchronous mode is indicated and data written to the DATA reg will be presented to the analog DAC input in the following IPBus clock cycle.
		0 Asynchronous mode; data written to DATA register is presented to DAC inputs on next clock cycle
		1 Synchronous mode; rising edge of SYNC_IN updates data presented to DAC input
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 four unused bits are ignored.
		0 Data words are tight 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 PDN the analog DAC will output the value currently presented to its inputs. The analog block requires 12 µsec to recover from the power down state before proper operation is guaranteed.
		0 DAC is operational
		1 DAC is powered down (default)

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FILT_CNT			FILT_EN	0	0	0	0	0	0	UP	DOWN	AUTO	SYNC_EN	FORMAT	PDN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

DAC

Buffered Data (DATA) Register

Bits	Name	Description
11 - 0	DATA	**Digital Analog Converter Data with FORMAT = 0**
		When the DAC is in operational mode (PDN = 0), the digital data contained in this register is presented to the analog DAC upon the rising edge of the SYNC_IN signal (or at the next clock cycle if SYNC_EN is clear) and converted to analog and output by the DAC. The data in this register can be updated at any rate, but the DAC is only guaranteed to operate at a maximum update rate of 0.5MHz.
15 - 4	DATA	**Digital Analog Converter Data with FORMAT = 1**
		When the DAC is in operational mode (PDN = 0), the digital data contained in this register is presented to the analog DAC upon the rising edge of the SYNC_IN signal (or at the next clock cycle if SYNC_EN is clear) and converted to analog and output by the DAC. The data in this register can be updated at any rate, but the DAC is only guaranteed to operate at a maximum update rate of 0.5MHz.

Buffered Data (DATA) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	DATA											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Buffered Data (DATA) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DATA												0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

DAC

Step Size (STEP) Register

Bits	Name	Description
11 - 0	STEP	**Step Size with FORMAT = 0**
		When the DAC is in Automatic mode (AUTO = 1), the step size contained in this register is added to, or subtracted from, the current value creating the next value presented to the DAC inputs. This register is not used during Normal mode operation.
15 - 4	STEP	**Step Size with FORMAT = 1**
		When the DAC is in Automatic mode (AUTO = 1), the step size contained in this register is added to, or subtracted from, the current value creating the next value presented to the DAC inputs. This register is not used during Normal mode operation.

Step Size (STEP) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	STEP											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Step Size (STEP) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	STEP												0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

DAC

Minimum Value (MINVAL) Register

Bits	Name	Description
11 - 0	MINVAL	**Minimum Value with FORMAT = 0**
		When the DAC is in Automatic mode (AUTO = 1), the minimum value contained in this register acts as the lower range limit during automatic waveform generation. Please see **Section 17.4.2** for a description of how this value is used in automatic waveform generation. This register is not used during Normal mode operation. Please refer to the chip data sheet for limitations on the low end voltage output of the DAC.
15 - 4	MINVAL	**Minimum Value with FORMAT = 1**
		When the DAC is in Automatic mode (AUTO = 1), the minimum value contained in this register acts as the lower range limit during automatic waveform generation. Please see **Section 17.4.2** for a description of how this value is used in automatic waveform generation. This register is not used during Normal mode operation. Please refer to the chip data sheet for limitations on the low end voltage output of the DAC.

Minimum Value (MINVAL) Register Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	MINVAL											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Minimum Value (MINVAL) Register Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	MINVAL												0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

DAC

Maximum Value (MAXVAL) Register

Bits	Name	Description
11 - 0	MAXVAL	**Maximum Value with FORMAT = 0**
		When the DAC is in Automatic mode (AUTO = 1), the maximum value contained in this register acts as the upper range limit during automatic waveform generation. Please see **Section 17.4.2** for a description of how this value is used in automatic waveform generation. This register is not used during Normal mode operation. Please refer to the chip data sheet for limitations on the high end voltage output of the DAC.
15 - 4	MAXVAL	**Maximum Value with FORMAT = 1**
		When the DAC is in Automatic mode (AUTO = 1), the maximum value contained in this register acts as the upper range limit during automatic waveform generation. Please see **Section 17.4.2** for a description of how this value is used in automatic waveform generation. This register is not used during Normal mode operation. Please refer to the chip data sheet for limitations on the high end voltage output of the DAC.

Maximum Value (MAXVAL) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	1	1	1	1	MINVAL											
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Maximum Value (MAXVAL) Register Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	MINVAL												1	1	1	1
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

CMP

Control (CTRL) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	RCIE	**Rising Compare Interrupt Enable**
		This bit enables assertion of a compare interrupt when the Rising Compare (RC) status bit is high.
		0 RC has no effect on the compare interrupt (default)
		1 Compare interrupt asserts while RC is high
14	FCIE	**Falling Compare Interrupt Enable**
		This bit enables assertion of a compare interrupt when the Falling Compare (FC) status bit is high.
		0 RC has no effect on the compare interrupt (default)
		1 Compare interrupt asserts while RC is high
11 - 10	ESEL	**Export Source Select**
		This bit selects analog input source connected to EXPORT output.
		0 CIN1 input (default)
		1 CIN2 input
		2 CIN3 input
		3 DAC input
8 - 6	NSEL	**Negative Comparator Input Source Select**
		This bit field selects the analog input source connected to the Negative (-) input of the differential comparator. Setting NSEL to a reserved value connects the Negative input of the differential comparator to V_{SS_A}. Setting PSEL and NSEL to the same value treats both as if set to reserved values.
		0 CIN1 input (default)
		1 CIN2 input
		2 CIN3 input
		3 DAC input
		4 IMPORT input
		5 - 7 Reserved (V_{SS_A})

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RCIE	FCIE	0	0	ESEL		0	NSEL			0	PSEL			INV	PDN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

CMP

Control (CTRL) Register Continued

Bits	Name	Description
4 - 2	PSEL	**Positive Comparator Input Source Select**
		This bit selects the analog input source connected to the Positive (+) input of the differential comparator. Setting PSEL to a reserved value connects the Positive input of the differential comparator to V_{DD_A}. Setting PSEL and NSEL to the same value treats both as if set to reserved values.
		0 CIN1 input (default)
		1 CIN2 input
		2 CIN3 input
		3 DAC input
		4 IMPORT input
		5 - 7 Reserved (V_{DD_A})
1	INV	**Invert Control**
		Enables optional inversion of the differential comparator output CMP. By default (not inverted) the comparator outputs one when the positive (+) input voltage exceeds the negative (-) input voltage.
		0 Comparator output is not inverted (default)
		1 Comparator output is inverted
0	PDN	**Power Down**
		This bit is used to power down the differential comparator when not in use. The power down recovery time is provided under electrical specifications. This is the time required to recover from the power down state before proper operation is guaranteed. The output of the differential comparator CMP is zero while PDN is one.
		0 Differential comparator is operational
		1 Differential comparator is powered down and output CMP is set to zero (default)

Control (CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RCIE	FCIE	0	0	ESEL		0	NSEL			0	PSEL			INV	PDN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

CMP

Status (STAT) Register

Bits	Name	Description
15	RC	**Rising Comparator Output Edge Indicator**
		This bit asserts high upon the detection of a rising edge on COUT. COUT is the output of the differential comparator after optional inversion, synchronization, and optional low-pass filter. Writing one clears the RC bit while writing zero to the RC bit is ignored.
		0 Rising edge not detected on COUT since last time RC was cleared (default)
		1 Rising edge detected on COUT
14	FC	**Falling Comparator Output Edge Indicator**
		This bit asserts high upon the detection of a falling edge on COUT. COUT is the output of the differential comparator after optional inversion, synchronization, and optional low-pass filter. Writing one clears the FC bit while writing zero to the FC bit is ignored.
		0 Falling edge not detected on COUT since last time RC was cleared (default)
		1 Falling edge detected on COUT
0	COUT	**Synchronized and Filtered Compare Output**
		This bit provides read access to the COUT signal. COUT is the output of the differential comparator after optional inversion, synchronization, and optional low-pass filter.

Status Register (STAT) Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RC	FC	0	0	0	0	0	0	0	0	0	0	0	0	0	COUT
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

CMP

Filter (FILT) Register

Bits	Name	Description
10 - 8	FILT_CNT	**Filter Sample Count**
		This bit field represents the number of consecutive samples that must agree prior to the comparator output filter accepting a new output state. A value of 0x0 represents three samples. A value of 0x7 represents ten samples. Filter programming and latency details are described in **Section 18.4.6**.
7 - 0	FILT_PER	**Filter Sample Period**
		This bit field specifies the sampling period, in peripheral clock cycles, of the comparator output filter. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details are described in **Section 18.4.6**.

Filter (FILT) Register Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	FILT_CNT			FILT_PER							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ITCN

Interrupt Priority Register 0 (IPR0)

Please see the following page for continuation of this register

Bits	Name	Description
15 - 14	PLL IPL	**PLL Loss of Reference or Change in Lock Status Interrupt Priority Level**
		This field is used to set the interrupt priority levels for the PLL IRQ. This IRQ is limited to priorities one through three. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 1
		10 IRQ is priority level 2
		11 IRQ is priority level 3
13 - 12	LVI IPL	**Low Voltage Detector Interrupt Priority Level**
		This field is used to set the interrupt priority levels for the LVI IRQ. This IRQ is limited to priorities one through three and is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 1
		10 IRQ is priority level 2
		11 IRQ is priority level 3
9 - 8	RX_REG IPL	**EOnCE Receive Register Full Interrupt Priority Level**
		This field is used to set the interrupt priority level for the RX_REG IRQ. This IRQ is limited to priorities one through three. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 1
		10 IRQ is priority level 2
		11 IRQ is priority level 3
7 - 6	TX_REG IPL	**EOnCE Transmit Register Empty Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TX_REG IRQs. This IRQ is limited to priorities one through three. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 1
		10 IRQ is priority level 2
		11 IRQ is priority level 3

Interrupt Priority 0 Register (IPR0) Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PLL_IPL		LVI IPL		0	0	RX_REG IPL		TX_REG IPL		TRBUF IPL		BKPT_U IPL		STPCNT IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ITCN

Interrupt Priority Register 0 (IPR0) Continued

Bits	Name	Description
5 - 4	**TRBUF IPL**	**EOnCE Trace Buffer Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TR_BUF IRQ. This IRQ is limited to priorities one through three. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 1
		10 IRQ is priority level 2
		11 IRQ is priority level 3
3 - 2	**BKPT_U IPL**	**EOnCE Breakpoint Unit Interrupt Priority Level**
		This field is used to set the interrupt priority level for the BKPT IRQ. This IRQ is limited to priorities one through three. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 1
		10 IRQ is priority level 2
		11 IRQ is priority level 3
1 - 0	**STPCNT IPL**	**EOnCE Step Counter Interrupt Priority Level**
		This field is used to set the interrupt priority level for the STPCNT IRQ. This IRQ is limited to priorities one through three. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 1
		10 IRQ is priority level 2
		11 IRQ is priority level 3

Interrupt Priority 0 Register (IPR0) Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PLL_IPL		LVI IPL		0	0	RX_REG IPL		TX_REG IPL		TRBUF IPL		BKPT_U IPL		STPCNT IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ITCN

Interrupt Priority Register 1 (IPR1)

Please see the following page for continuation of this register

Bits	Name	Description
15 -14	GPIOD IPL	**GPIOD Interrupt Priority Level**
		This field is used to set the interrupt priority level for the GPIOD IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
13 - 12	CAN_WKUP IPL	**MSCAN Wake-Up Interrupt Priority Level**
		This field is used to set the interrupt priority level for the CAN_WKUP IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
11 - 10	CAN_TX IPL	**MSCAN Transmit Interrupt Priority Level**
		This field is used to set the interrupt priority level for the CAN_TX IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
9 - 8	CAN_RX IPL	**MSCAN Receive Interrupt Priority Level**
		This field is used to set the interrupt priority level for CAN_RX IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 1 Register (IPR1) Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	GPIOD IPL		CAN_WKUP IPL		CAN_TX IPL		CAN_RX IPL		CAN_ERR IPL		FM_CBE IPL		FM_CC IPL		FM_ERR IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

ITCN

Interrupt Priority Register 1 (IPR1) Continued

Bits	Name	Description
7 - 6	**CAN_ERR IPL**	**MSCAN Error Interrupt Priority Level**
		This field is used to set the interrupt priority level for CAN_ERR IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
5 - 4	**FM_CBE IPL**	**FM Command, Data, Address Buffers Empty Interrupt Priority Level**
		This field is used to set the interrupt priority level for the FM_CBE IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
3 - 2	**FM_CC IPL**	**FM Command Interrupt Priority Level**
		This field is used to set the interrupt priority level for the FM_CC IRQs. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
1 - 0	**FM_ERR IPL**	**FM Error Interrupt Priority Level**
		This field is used to set the interrupt priority level for the FM_ERR IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 1 Register (IPR1) Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	GPIOD IPL		CAN_WKUP IPL		CAN_TX IPL		CAN_RX IPL		CAN_ERR IPL		FM_CBE IPL		FM_CC IPL		FM_ERR IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Interrupt Priority Register 2 (IPR2)

Please see the following page for continuation of this register

Bits	Name	Description
15 - 14	SCI0_XMIT IPL	**SCI0 Transmitter Empty Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI0_XMIT IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
13 - 12	SPI1_XMIT IPL	**SPI1 Transmitter Empty Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SPI1_XMIT IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
11 - 10	SPI1_RCV IPL	**SPI1 Receiver Full Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SPI1_RCV IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
9 - 8	SPI0_XMIT IPL	**SPI0 Transmitter Empty Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SPI0_XMIT IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 2 Register (IPR2) Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SCI0_XMIT IPL		SPI1_XMIT IPL		SPI1_RCV IPL		SPI0_XMIT IPL		SPI0_RCV IPL		GPIOA IPL		GPIOB IPL		GPIOC IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

ITCN

Interrupt Priority Register 2 (IPR2) Continued

Bits	Name	Description
7 - 6	SPI0_RCV IPL	**SPI0 Receiver Full Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SPI0_RCV IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
5 - 4	SPI_XMIT IPL	**SPI Transmitter Empty Interrupt Priority Level**
		This field is used to set the interrupt priority level for the GPIOA IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
3 - 2	SPI_RCV IPL	**SPI Receiver Full Interrupt Priority Level**
		This field is used to set the interrupt priority level for the GPIOB IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
1 - 0	GPIOA IPL	**GPIOA Interrupt Priority Level**
		This field is used to set the interrupt priority level for the GPIOC IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 2 Register (IPR2) Base + $2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	SCI0_XMIT IPL		SPI1_XMIT IPL		SPI1_RCV IPL		SPI0_XMIT IPL		SPI0_RCV IPL		GPIOA IPL		GPIOB IPL		GPIOC IPL	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Interrupt Priority Register 3 (IPR3)

Please see the following page for continuation of this register

Bits	Name	Description
15 - 14	I2C_ERR IPL	**I^2C Error Interrupt Priority Level**
		This field is used to set the interrupt priority level for the I2C_ERR IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
13 - 12	SCI1_RCV IPL	**SCI1 Receiver Full Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI1_RCV IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
11 - 10	SCI1_RERR IPL	**SCI1 Receiver Error Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI1_RERR IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
9 - 8	SCI1_TIDL IPL	**SCI1 Transmitter Idle Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI_TIDL IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 3 Register (IPR3) Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	I2C_ERR IPL		SCI1_RCV IPL		SCI1_RERR IPL		SCI1_TIDL IPL		SCI1_XMIT IPL		SCI0_RCV IPL		SCI0_RERR IPL		SCI0_TIDL IPL	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Interrupt Priority Register 3 (IPR3) Continued

Bits	Name	Description
7 - 6	SCI1_XMIT IPL	**SCI1 Transmitter Empty Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI_XMIT IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
5 - 4	SCI0_RCV IPL	**SCI0 receiver Full Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI0_RCV IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
3 - 2	SCI0_RERR IPL	**SCI0 Receiver Error Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI0_RERR IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
1 - 0	SCI0_TIDL IPL	**SCI0 Transmitter Idle Interrupt Priority Level**
		This field is used to set the interrupt priority level for the SCI0_TIDL IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 3 Register (IPR3) Base + $3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	I2C_ERR IPL		SCI1_RCV IPL		SCI1_RERR IPL		SCI1_TIDL IPL		SCI1_XMIT IPL		SCI0_RCV IPL		SCI0_RERR IPL		SCI0_TIDL IPL	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Interrupt Priority Register 4 (IPR4)

Please see the following page for continuation of this register

Bits	Name	Description
15 - 14	**TMRA_3 IPL**	**Timer A, Channel 3 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRA_3 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
	00	IRQ disabled (default)
	01	IRQ is priority level 0
	10	IRQ is priority level 1
	11	IRQ is priority level 2
13 - 12	**TMRA_2 IPL**	**Timer A, Channel 2 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRA_2 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
	00	IRQ disabled (default)
	01	IRQ is priority level 0
	10	IRQ is priority level 1
	11	IRQ is priority level 2
11 - 10	**TMRA_1 IPL**	**Timer A, Channel 1 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRA_1 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
	00	IRQ disabled (default)
	01	IRQ is priority level 0
	10	IRQ is priority level 1
	11	IRQ is priority level 2
9 - 8	**TMRA_0 IPL**	**Timer A, Channel 0 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRA_0 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
	00	IRQ disabled (default)
	01	IRQ is priority level 0
	10	IRQ is priority level 1
	11	IRQ is priority level 2

Interrupt Priority 4 Register (IPR4) Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TMRA_3 IPL		TMRA_2 IPL		TMRA_1 IPL		TMRA_0 IPL		I2C_STAT IPL		I2C_TX IPL		I2C_RX IPL		I2C_GEN IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

ITCN

Interrupt Priority Register 4 (IPR4) Continued

Bits	Name	Description
7 - 6	I2C_STAT IPL	**I^2C Status Interrupt Priority Level**
		This field is used to set the interrupt priority level for the I2C_STAT IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
5 - 4	I2C_TX IPL	**I^2C Transmit Interrupt Priority Level**
		This field is used to set the interrupt priority level for the I2C_TX IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
3 - 2	I2C_RX IPL	**I^2C Receive Interrupt Priority Level**
		This field is used to set the interrupt priority level for the I2C_RX IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
1 - 0	I2C_GEN IPL	**I^2C Generate Interrupt Priority Level**
		This field is used to set the interrupt priority level for the I2C_GEN IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 4 Register (IPR4) Base + $4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	TMRA_3 IPL		TMRA_2 IPL		TMRA_1 IPL		TMRA_0 IPL		I2C_STAT IPL		I2C_TX IPL		I2C_RX IPL		I2C_GEN IPL	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

_____ Programmer: _____

ITCN

Interrupt Priority Register 5 (IPR5)

Please see the following page for continuation of this register

Bits	Name	Description
15 - 14	PIT1 IPL	**Programmable Interval Timer1 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the PIT1 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
13 - 12	PIT0 IPL	**Programmable Interval Timer0 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the PIT0 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
11 - 10	COMPB IPL	**Comparator B Interrupt Priority Level**
		This field is used to set the interrupt priority level for the COMPB IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
9 - 8	COMPA IPL	**Comparator A Interrupt Priority Level**
		This field is used to set the interrupt priority level for the COMPA IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 5 Register (IPR5) Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PIT1 IPL		PIT0 IPL		COMPB IPL		COMPA IPL		TMRB_3 IPL		TMRB_2 IPL		TMRB_1 IPL		TMRB_0 IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Interrupt Priority Register 5 (IPR5) Continued

Bits	Name	Description
7 - 6	**TMRB_3 IPL**	**Timer B, Channel 3 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRB_3 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
5 - 4	**TMRB_2 IPL**	**Timer B, Channel 2 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRB_2 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
3 - 2	**TMRB_1 IPL**	**Timer B, Channel 1 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRB_1 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
1 - 0	**TMRB_0 IPL**	**Timer B, Channel 0 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the TMRB_0 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 5 Register (IPR5) Base + $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PIT1 IPL		PIT0 IPL		COMPB IPL		COMPA IPL		TMRB_3 IPL		TMRB_2 IPL		TMRB_1 IPL		TMRB_0 IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Interrupt Priority Register 6 (IPR6)

Please see the following page for continuation of this register

Bits	Name	Description
11 - 10	PWM_F IPL	**PWM Fault Interrupt Priority Level**
		This field is used to set the interrupt priority level for the PWM_F IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
9 - 8	PWM_RL IPL	**PWM Reload Interrupt Priority Level**
		This field is used to set the interrupt priority level for the PWM_RL IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
7 - 6	ADC_ZC IPL	**ADC Zero Crossing Interrupt Priority Level**
		This field is used to set the interrupt priority level for the ADC_ZC IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
5 - 4	ADCB_CC IPL	**ADC B Conversion Complete Interrupt Priority Level**
		This field is used to set the interrupt priority level for the ADCB_CC IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 6 Register (IPR6) Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	PWM_F IPL		PWM_RL IPL		ADC_ZC IPL		ADCB_CC IPL		ADCA_CC IPL		PIT2 IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

ITCN

Interrupt Priority 6 (IPR6) Register Continued

Bits	Name	Description
3 - 2	**ADCA_CC IPL**	**ADC A Conversion Complete Interrupt Priority Level**
		This field is used to set the interrupt priority level for the ADCA_CC IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2
1 - 0	**PIT2 IPL**	**Programmable Interval Timer 2 Interrupt Priority Level**
		This field is used to set the interrupt priority level for the PIT2 IRQ. This IRQ is limited to priorities zero through two. It is disabled by default.
		00 IRQ disabled (default)
		01 IRQ is priority level 0
		10 IRQ is priority level 1
		11 IRQ is priority level 2

Interrupt Priority 6 Register (IPR6) Base + $6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	PWM_F IPL		PWM_RL IPL		ADC_ZC IPL		ADCB_CC IPL		ADCA_CC IPL		PIT2 IPL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Vector Base Address (VBA) Register

Bits	Name	Description
13 - 0	VECTOR_BASE_ADDRESS	**Vector Base Address**
		The value in this register is used as the upper 14 bits of the interrupt vector VAB[20:0]. The lower seven bits are determined based on the highest priority interrupt and are then appended onto VBA before presenting the full VAB to the Core.

Vector Base Address (VBA) Register Base + $7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	VECTOR_BASE_ADDRESS													
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ITCN

Fast Interrupt Match 0 (FIM0) Register

Bits	Name	Description
5 - 0	FAST INTERRUPT 0	**Fast Interrupt 0 Vector Number**
		These values determine which IRQ will be Fast Interrupt 0. 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 two. Unexpected results will occur if a Fast Interrupt vector is set to any other priority. A Fast Interrupt automatically becomes the highest-priority level two interrupt regardless of its 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.

Fast Interrupt Match 0 (FIM0) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	FAST INTERRUPT 0					
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ITCN

Fast Interrupt 0 Vector Address Low (FIVAL0) Register

Bits	Name	Description
15 - 0	FAST INTERRUPT 0 VECTOR ADDRESS LOW	**Fast Interrupt 0 Vector Address Low**
		The lower 16 bits of the vector address used for Fast Interrupt 0. This register is combined with FIVAH0 to form the 21-bit vector address for Fast Interrupt 0 defined in the FIM0 register.

Fast Interrupt 0 Vector Address Low (FIVAL0) Register Base + $9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FAST INTERRUPT 0 VECTOR ADDRESS LOW															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

ITCN

Fast Interrupt 0 Vector Address High (FIVAH0) Register

Bits	Name	Description
4 - 0	FAST INTERRUPT 0 VECTOR ADDRESS HIGH	**Fast Interrupt 0 Vector Address High**
		The upper five bits of the vector address used for Fast Interrupt 0. This register is combined with FIVAL0 to form the 21-bit vector address for Fast Interrupt 0 defined in the FIM0 register.

Fast Interrupt 0 Vector Address High (FIVAH0) Register Base + $A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	FAST INTERRUPT 0 VECTOR ADDRESS HIGH				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ITCN

Fast Interrupt 1 Match (FIM1) Register

Bits	Name	Description
5 - 0	FAST INTERRUPT1	**Fast Interrupt 1 Match**
		These values determine which IRQ will be Fast Interrupt 1. 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. A Fast Interrupt automatically becomes the highest priority level 2 interrupt, regardless of its 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.

Fast Interrupt 1 Match (FIM1) Register Base + $B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	FAST INTERRUPT 1					
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ITCN

Fast Interrupt 1 Vector Address Low (FIVAL1) Register

Bits	Name	Description
15 - 0	FAST INTERRUPT 1 VECTOR ADDRESS LOW	**Fast Interrupt 1 Vector Address Low**
		The lower 16 bits of the vector address used for Fast Interrupt 1. This register is combined with FIVAH1 to form the 21-bit vector address for Fast Interrupt 1 defined in the FIM1 register.

Fast Interrupt1 Vector Address Low (FIVAL1) Register Base + $C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FAST INTERRUPT 1 VECTOR ADDRESS LOW															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ITCN

Fast Interrupt 1 Vector Address High (FIVAH1) Register

Bits	Name	Description
15 - 0	FAST INTERRUPT 1 VECTOR ADDRESS HIGH	**Fast Interrupt 1 Vector Address High**
		The upper five bits of the vector are address used for Fast Interrupt 1. This register is combined with FIVAL1 to form the 21-bit vector address for Fast Interrupt 1 defined in the FIM1 register.

Fast Interrupt1 Vector Address High (FIVAH1) Register Base + $D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	FAST INTERRUPT 1 VECTOR ADDRESS HIGH				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ITCN

IRQ Pending 0 (IRQP0) Register

Bits	Name	Description	
16 - 2	PENDING[16-2]	**Pending**	
		This register combines with IRQP1, IRQP2, and IRQP3 to represent the pending IRQs for interrupt vector numbers 2 through 63.	
		0	IRQ pending for this vector number
		1	No IRQ pending for this vector number

IRQ Pending 0 (IRQP0) Register Base + $E	Bits	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

Reserved Bits

ITCN

IRQ Pending 1 (IRQP1) Register

Bits	Name	Description
32 - 17	PENDING[32:17]	**Pending**
		This register combines with IRQP0, IRQP2, and IRQP3 to represent the pending IRQs for interrupt vector numbers 2 through 63.
		0 IRQ pending for this vector number
		1 No IRQ pending for this vector number

IRQ Pending 1 (IRQP1) Register Base + $F	Bits	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

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ITCN

IRQ Pending 2 (IRQP2) Register

Bits	Name	Description	
48 - 33	PENDING[48:33]	**Pending**	
		This register combines with IRQP0, IRQP1 and IRQP3 to represent the pending IRQs for interrupt vector numbers 2 through 63.	
		0	IRQ pending for this vector number
		1	No IRQ pending for this vector number

IRQ Pending 2 (IRQP2) Register Base + $10	Bits	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

Reserved Bits

ITCN

IRQ Pending 3 (IRQP3) Register

Bits	Name	Description
63 - 49	PENDING[63:49]	**Pending**
		This register combines with IRQP0, IRQP1 and IRQP3 to represent the pending IRQs for interrupt vector numbers 2 through 63.
		0 IRQ pending for this vector number
		1 No IRQ pending for this vector number

IRQ Pending 3 (IRQP3) Register Base + $11	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PENDING[63:49]															
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ITCN

Interrupt Control (ICTRL) Register

Bits	Name	Description
15	INT	**Interrupt**
		This *read-only* bit reflects the state of the interrupt to the 56800E core.
		0 — No interrupt is being sent to the 56800E core
		1 — An interrupt is being sent to the 56800E core
14 - 13	IPIC	**Interrupt Priority Level IRQ**
		These *read-only* bits reflect the state of the new interrupt priority level bits being presented to the 56800E core. These bits indicate the priority level needed for a new IRQ to interrupt the current interrupt being sent to the 56800E core. This field is only updated when the 56800E 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 or 3
		11 — Required nested exception priority level is 3
12 - 6	VAB	**Vector Address Bus**
		This *read-only* field shows the vector number (VAB[7:1]) 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 56800E core jumps to a new interrupt service routine.
5	INT_DIS	**Interrupt Disable**
		This bit allows all interrupts to be disabled.

Interrupt Control (ICTRL) Register Base + $16

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	INT	IPIC		VAB							INT_DIS	1	1	1	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Control (SIM_CTRL) Register

Bits	Name	Description	
5	ONCEEBL	**OnCE Enable**	
		0	OnCE clock to 56800E core enabled when core TAP is enabled
		1	OnCE clock to 56800E core is always enabled
4	SW RST	**Software Reset**	
		Writing one to this field will cause the part to reset	
3 - 2	STOP_DISABLE	**Stop Disable**	
		00	Stop mode will be entered when the 56800E core executes a STOP instruction
		01	The 56800E STOP instruction will not cause entry into Stop mode
		10	Stop mode will be entered when the 56800E core executes a STOP instruction and the STOP_DISABLE field is write-protected until the next reset
		11	The 56800E STOP instruction will 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 will be entered when the 56800E core executes a WAIT instruction
		01	The 56800E WAIT instruction will not cause entry into Wait mode
		10	Wait mode will be entered when the 56800E core executes a WAIT instruction and the WAIT_DISABLE field is write-protected until the next reset
		11	The 56800E WAIT instruction will not cause entry into Wait mode and the WAIT_DISABLE field is write-protected until the next reset

Control (SIM_CTRL) Register Base + $0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	ONCE EBL	SW RST	STOP_DISABLE		WAIT_DISABLE	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Reset Status (SIM_RSTAT) Register

Bits	Name	Description
5	SWR	**Software Reset**
		When set, this bit indicates the previous system reset occurred as a result of a software reset (written 1 to SW RST bit in the SIM_CTRL register). It will not be set if a COP, external, or POR reset also occurred.
4	COPR	**COP Reset**
		When set, this bit indicates the previous system reset was caused by the Computer Operating Properly (COP) timer. It will not be set if an external or POR reset also occurred. If COPR is set as code starts executing, the COP reset vector in the vector table will be 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. It will only be set if the external reset pin was asserted or remained asserted after the power-on reset deasserted.
2	POR	**Power-On Reset**
		This bit is set during a power-on reset.

Reset Status (SIM_RSTAT) Register Base + $1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	SWR	COPR	EXTR	POR	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	1*	0	0

* Coming out of reset, the state of these bits reflects the cause of the reset; therefore, there is no specific reset value of these bits.

Reserved Bits

SIM

Software Control (SIM_SWC0-3) Registers

Bits	Name	Description
15 - 0	SOFTWARE CONTROL DATA 0-3	**Software Control Data 0-3**
		Only SIM_SWC0 is shown. The other three registers are identical in functionality. This register is reset only by the Power-On Reset (POR). It has no part-specific functionality and is intended for use by a software developer to contain data that will be unaffected by the other reset sources (RESET pin, software reset, and COP reset).

Software Control Data0-3 (SIM_SWC0-3) Registers Base + $2 - $5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SOFTWARE CONTROL DATA 0-3															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

Programmer: _____

SIM

Most Significant Half of JTAG ID (SIM_MSHID) Register

Bits	Name	Description
15 - 0	n/a	***Read-Only* Bits**
		This *read-only* register displays the most significant half of the JTAG ID for the chip. This register reads $01F2.

Most Significant Half/JTAG ID (SIM_MSHID) Register Base + $6	Bits	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	1	1	0	0	1	0
	Write																
	Reset	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	0

Reserved Bits

SIM

Least Significant Half of JTAG ID (SIM_LSHID) Register

Bits	Name	Description
15 - 0	n/a	***Read-Only* Bits**
		This *read-only* register displays the least significant half of the JTAG ID for the chip. This register reads $401D.

Least Significant Half/JTAG ID (SIM_LSHID) Register Base + $7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	1
	Write																
	Reset	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Power Control (SIM_PWR) Register

Bits	Name	Description	
1 - 0	LRSTDBY	**Large Regulator Standby Mode**	
		00	Large regulator is in Normal mode
		01	Large regulator is in Standby (reduced power) mode
		10	Large regulator is in Normal mode and the LRSTDBY field is write-protected until the next reset
		11	Large regulator is in Standby mode and the LRSTDBY field is write-protected until the next reset

Power Control (SIM_PWR) Register Base + $8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LRSTDBY	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

Clock Output Select (SIM_CLKOUT) Register

Bits	Name	Description
5	CLKDIS	**Clockout Disable**
		The Clock Output Select register can be used to multiplex out selected clocks generated inside the clock generation and SIM modules onto the CLKO clock output signal. All functionality other than CLKOSEL is for test purposes only. Glitches may be produced when the clock is enabled or switched. The delay from the clock source to the output is unspecified. The ability to observe of the CLKO clock output signal at an output pad is subject to the frequency limitations of the associated IO cell. The lower four bits of the GPIO A register can function as GPIO, PWM, or as additional clock output signals. GPIO has priority and is enabled/disabled via the GPIOA_PEREN. If GPIOA[3:0] are programmed to operate as peripheral outputs, then the choice between PWM and additional clock outputs is done here in the CLKOSR. The default state is for the peripheral function of GPIOA[3:0] to be programmed as PWM. This can be changed by altering PWM3 through PWM0.
		0 — CLKOUT output is enabled and will output the signal indicated by CLKOSEL
		1 — CLKOUT is 0
4 - 0	CLKOSEL	**Clockout Select**
		Selects clock to be muxed out on the CLKO pin as defined in the following bullets. Internal delay to CLKO output is unspecified. Signal at the output pad is undefined when CLKO signal frequency exceeds rated frequency of IO cell. CLKO may glitch when CLKDIS and CLKOSEL settings are changed. CLKDIS and CLKOSEL define what CLKO will be. Propagation of CLKO to external pad requires proper setting of related GPS*n* and GPIOx_PEREN fields.
		00 — System frequency, continuous after POR
		01 — Peripheral frequency, continuous when not in reset
		02 — 3x system frequency, continuous only while PLL selected
		03 — Master clock source before PLL (ROSC, OSC or external clock) continuous

CLKO Select (SIM_CLKOUT) Register Base + $A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	CLK DIS	CLKOSEL				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Peripheral Clock Rate (SIM_PCR) Register

Bits	Name	Description
15	TMRB_CR	**Quad Timer B Clock Rate**
		This bit selects the clock speed for the Quad Timer B module.
	0	Quad Timer B clock rate equals the core clock rate, to a maximum 32MHz (default)
	1	Quad Timer B clock rate equals 3x core clock rate
14	TMRA_CR	**Quad Timer A Clock Rate**
		This bit selects the clock speed for the Quad Timer A module.
	0	Quad Timer A clock rate equals the core clock rate, to a maximum 32MHz (default)
	1	Quad Timer A clock rate equals 3x core clock rate
13	PWM_CR	**Pulse Width Modulator Clock Rate**
		This bit selects the clock speed for the PWM module.
	0	PWM module clock rate equals the core clock rate, to a maximum 32MHz (default)
	1	PWM module clock rate equals 3x core clock rate
12	I2C_CR	**Inter-Integrated Circuit Run Clock Rate**
		This bit selects the clock speed for the I^2C run clock. I2C_CR must be set to 0 if the PLL is not on and selected in OCCS. The I^2C should be disabled prior to altering I2C_CR. See the 56F8000 Peripheral Reference Manual for details.
	0	I^2C module run clock rate equals the core clock rate, to a maximum 32MHz (default)
	1	I^2C module run clock rate equals 3x core clock rate

Peripheral Clock Rate (SIM_PCR) Register Base + $B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TMRB_CR	TMRA_CR	PWM_CR	I2C_CR	0	0	0	0	0	0	0	0	0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

Peripheral Clock Enable 0 (SIM_PCE0) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	CMPB	**Comparator B IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
14	CMPA	**Comparator A IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
13	DAC1	**Digital-to-Analog IPBus Clock Enable 1**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
12	DAC0	**Digital-to-Analog IPBus Clock Enable 0**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
10	ADC	**Analog-to-Digital Converter IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
6	I2C	**Inter-Integrated Circuit IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked

Peripheral Clock Enable 0 SIM_PCE0) Register Base + $C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CMPB	CMPA	DAC1	DAC0	0	ADC	0	0	0	12C	SCI1	SCI0	SPI1	SPI0	0	PWM
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Peripheral Clock Enable 0 (SIM_PCE0) Register Continued

Bits	Name	Description
5	SCI1	**SCI1 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
4	SCI0	**SCI0 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
3	SPI1	**SPI1 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
2	SPI0	**SPI0 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
0	PWM	**PWM IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked

Peripheral Clock Enable 0(SIM_PCE0) Register Base + $C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CMPB	CMPA	DAC1	DAC0	0	ADC	0	0	0	12C	SCI1	SCI0	SPI1	SPI0	0	PWM
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

Peripheral Clock Enable 1 (SIM_PCE1) Register

Please see the following page for continuation of this register

Bits	Name	Description
14	PIT2	**Programmable Interval Timer 2 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
13	PIT1	**Programmable Interval Timer 1 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
12	PIT0	**Programmable Interval Timer 0 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
7	TB3	**Quad Timer B, Channel 3 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
6	TB2	**Quad Timer B, Channel 2 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked

Peripheral Clock Enable 1 (SIM_PCE1) Register Base + $D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	PIT2	PIT1	PIT0	0	0	0	0	TB3	TB2	TB1	TB0	TA3	TA2	TA1	TA0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Peripheral Clock Enable 1 (SIM_PCE1) Register Continued

Bits	Name	Description
5	TB1	**Quad Timer B, Channel 1 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
4	TB0	**Quad Timer B, Channel 0 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
3	TA3	**Quad Timer A, Channel 3 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
2	TA2	**Quad Timer A, Channel 2 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
1	TA1	**Quad Timer A, Channel 1 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked
0	TA0	**Quad Timer A, Channel 0 IPBus Clock Enable**
		Each bit controls clocks to the indicated peripheral.
		0 The corresponding peripheral is not clocked
		1 The corresponding peripheral is clocked

Peripheral Clock Enable 1 (SIM_PCE1) Register Base + $D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	PIT2	PIT1	PIT0	0	0	0	0	TB3	TB2	TB1	TB0	TA3	TA2	TA1	TA0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

Stop Disable 0 (SD0) Register

Please see the following page for continuation of this register

Bits	Name	Description
15	CMPB_SD	**Comparator B IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
14	CMPA_SD	**Comparator A IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
13	DAC1_SD	**Digital-to-Analog Converter 1 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
12	DAC0_SD	**Digital-to-Analog Converter 0 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
10	ADC_SD	**Analog-to-Digital Converter IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
6	I2C_SD	**Inter-Integrated Circuit IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode

Stop Disable 0 (SIM_SD0) Register Base + $E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CMPB_SD	CMPA_SD	DAC1_SD	DAC_SD	0	ADC_SD	0	0	0	I2C_SD	SC1_SD	SCI0_SD	SPI1_SD	SPI0_SD	0	PWM_SD
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Stop Disable 0 (SD0) Register Continued

Bits	Name	Description
5	SCI1_SD	**SCI1 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
4	SCI0_SD	**SCI0 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
3	SPI1_SD	**SPI1 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
2	SPI0_SD	**SPI0 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
0	PWM_SD	**PWM IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode

Stop Disable 0 (SIM_SD0) Register Base + $E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CMPB_SD	CMPA_SD	DAC1_SD	DAC_SD	0	ADC_SD	0	0	0	I2C_SD	SC1_SD	SCI0_SD	SPI1_SD	SPI0_SD	0	PWM_SD
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

Stop Disable 1 (SD1) Register

Please see the following page for continuation of this register

Bits	Name	Description
14	PIT2_SD	**Programmable Interval Timer 2 IPBus Clock Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
13	PIT1_SD	**Programmable Interval Timer 1 IPBus Clock Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
12	PIT0_SD	**Programmable Interval Timer 0 IPBus Clock Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
7	TB3_SD	**Quad Timer B, Channel 3 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
6	TB2_SD	**Quad Timer B, Channel 2 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
5	TB1_SD	**Quad Timer B, Channel 1 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode

Stop Disable 1 (SIM_SD1) Register Base + $F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	PIT2_SD	PIT1_SD	PIT0_SD	0	0	0	0	TB3_SD	TB2_SD	TB1_SD	TB0_SD	TA3_SD	TA2_SD	TA1_SD	TA0_SD
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Stop Disable 1 (SD1) Register Continued

Bits	Name	Description
4	TB0_SD	**Quad Timer B, Channel 0 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
3	TA3_SD	**Quad Timer A, Channel 3 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
2	TA2_SD	**Quad Timer A, Channel 2 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
1	TA1_SD	**Quad Timer A, Channel 1 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode
0	TA0_SD	**Quad Timer A, Channel 0 IPBus Clock Stop Disable**
		Each bit controls clocks to the indicated peripheral.
		0 The clock is not provided to the peripheral in STOP mode
		1 The clock is enabled in STOP mode

Stop Disable 1 (SIM_SD1) Register Base + $F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	PIT2_SD	PIT1_SD	PIT0_SD	0	0	0	0	TB3_SD	TB2_SD	TB1_SD	TB0_SD	TA3_SD	TA2_SD	TA1_SD	TA0_SD
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

I/O Short Address Location High (SIM_IOSAHI) Register

Bits	Name	Description
23 - 22	ISAL	**Input/Output Short Address Location**
		This field represents the upper two address bits of the *hard coded* I/O short address.

I/O Short Address Location High (SIM_IOSAHI) Reg. Base + $10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ISAL[23:22]	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

I/O Short Address Location Low (SIM_IOSALO) Register

Bits	Name	Description
21 - 6	ISAL	**Input/Output Short Address Location**
		This field represents the lower 16 address bits of the *hard coded* I/O short address.

I/O Short Address Location Low (SIM_IOSALO) Reg. Base + $11	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ISAL[21:6]															
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Application: _____ Date: _____

Programmer: _____

SIM

Protection (SIM_PROT) Register

Bits	Name	Description	
3 - 2	PCEP	**Peripheral Clock Enable Protection**	
		These bits enable write protection of all fields in the PCE*n*, SD*n*, and PCR 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	GIPSP	**GPIO and Internal Peripheral Select Protection**	
		This bit field enables write protection of GPS*n* and IPS*n* registers and write protection of all GPIOx_PEREN, GPIOx_PPOUTM and GPIOx_DRIVE 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

Protection (SIM_PROT) Register Base + $12	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	PCEP		GIPSP	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

GPIO Peripheral Select 0 for GPIOA (SIM_GPSA0) Register

Bits	Name	Description				
12	GPS_A6	**Configure GPIOA6**				
		This bit selects the alternate function for GPIOA6.				
		0	FAULT0	PWM	INPUT	(default)
		1	TA0	TMRA	I/O	
11 - 10	GPS_A5	**Configure GPIOA5**				
		This bit field selects the alternate function for GPIOA5.				
		00	PWM5	PWM	OUTPUT	(default)
		01	FAULT2	PWM	I/O	
		10	TA3	TMRA	I/O	
		11	Reserved			
9 - 8	GPS_A4	**Configure GPIO4**				
		This bit field selects the alternate function for GPIOA4.				
		00	PWM4	PWM	OUTPUT	(default)
		01	FAULT1	PWM	INPUT	
		10	TA2	TMRA	I/O	
		11	Reserved			

GPIO Peripheral Select 0 (SIM_GPSA0) Reg. Base + $13	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	GPS_A6	GPS_A5		GPS_A4		0	0	0	0	0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

GPIO Peripheral Select 1 (SIM_GPSA1) Register

Please see the following page for continuation of this register

Bits	Name	Description
13 - 12	GPS_A14	**Configure GPIOA14**
		This bit field selects the alternate function for GPIOA14.
		00 TB3 TMRB I/O (default)
		01 MOSI1 SPI1 I/O
		10 TA3 TMRA I/O
		11 Reserved
11 - 10	GPS_A13	**Configure GPIOA13**
		This bit field selects the alternate function for GPIOA13.
		00 TB2 TMRB I/O (default)
		01 MOSI1 SPI1 I/O
		10 TA2 TMRA I/O
		11 Reserved
9 - 8	GPS_A12	**Configure GPIOA12**
		This bit field selects the alternate function for GPIOA12.
		00 TB1 TMRB I/O (default)
		01 SCLK1 SPI1 I/O
		10 TA1 TMRA I/O
		11 Reserved
6	GPS_A11	**Configure GPIOA11**
		This bit field selects the alternate function for GPIOA11.
		0 CINB2 CMPB ANA_IN (default)
		1 TB3 TMRB I/O

GPIO Peripheral Select 1 (SIM_GPSA1) Reg. Base + $14	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	GPS_A14		GPS_A13		GPS_A12		0	GPS_A11	0	GPS_A10	GPS_A9		GPS_A8	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

GPIO Peripheral Select 1 (SIM_GPSA1) Register Continued

Bits	Name	Description				
4	GPS_A10	**Configure GPIOA10**				
		This bit field selects the alternate function for GPIOA10.				
		0	CINB2	CMPA	ANA_IN	(default)
		1	TB2	TMRA	I/O	
3 - 2	GPS_A9	**Configure GPIOA9**				
		This bit field selects the alternate function for GPIOA9.				
		00	FAULT2	PWM	I/O	(default)
		01	TB3	TMRB	I/O	
		10	CINB1	CMPB	ANA_IN	
		11	Reserved			
1 - 0	GPS_A8	**Configure GPIOA8**				
		This bit field selects the alternate function for GPIOA8.				
		00	FAULT1	PWM	INPUT	(default)
		01	TB2	TMRA	I/O	
		10	CINA1	CMPA	ANA_IN	
		11	Reserved			

GPIO Peripheral Select 1 (SIM_GPSA1) Reg. Base + $14	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	GPS_A14		GPS_A13		GPS_A12		0	GPS_A11	0	GPS_A10	GPS_A9		GPS_A8	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

GPIO Peripheral Select 0 (SIM_GPSB0) Register

Please see the following page for continuation of this register

Bits	Name	Description				
14 - 13	GPS_B6	**Configure GPIOB6**				
		This bit field selects the alternate function for GPIOB6.				
		00	RXD0	SCI0	INPUT	(default)
		01	SDA	I2C	I/O	
		10	CLKIN	OCCS	INPUT	
		11	Reserved			
12 - 11	GPS_B5	**Configure GPIOB5**				
		This bit field selects the alternate function for GPIOB5.				
		00	TA1	TMRA	I/O	(default)
		01	FAULT3	PWM	INPUT	
		10	CLKIN	OCCS	INPUT	
		11	Reserved			
10 - 8	GPS_B4	**Configure GPIOB4**				
		This bit field selects the alternate function for GPIOB4.				
		000	TA0	TMRA	I/O	(default)
		001	CLKO	SIM	OUTPUT	
		010	$\overline{SS}$1	SPI1	INPUT	
		011	TB0	TRMB	I/O	
		100	PSRC2	PWM	INPUT	
		11x	Reserved			
		1x1	Reserved			

GPIO Peripheral Select 0 (SIM_GPSB0) Reg. Base + $15	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	GPS_B6		GPS_B5		GPS_B4			GPS_B3		GPS_B2		0	GPS_B1	0	GPS_ B0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

GPIO Peripheral Select 0 (SIM_GPSB0) Register Continued

Bits	Name	Description				
7 - 6	GPS_B3	**Configure GPIOB3**				
		This bit field selects the alternate function for GPIOB3.				
		00	MOSI0	SPI0	I/O	(default)
		01	TA3	TRMA	I/O	
		10	PSRC1	PWM	INPUT	
		11	Reserved			
5 - 4	GPS_B2	**Configure GPIOB2**				
		This bit field selects the alternate function for GPIOB2.				
		00	MISO0	SPI0	I/O	(default)
		01	TA2	TRMA	I/O	
		10	PSRC0	PWM	INPUT	
		11	Reserved			
2	GPS_B1	**Configure GPIOB1**				
		This bit field selects the alternate function for GPIOB1.				
		0	$\overline{SS}$0	SPI0	I/O	(default)
		1	SDA	I2C	I/O	
0	GPS_B0	**Configure GPIOB0**				
		This bit field selects the alternate function for GPIOB0.				
		0	SCLK0	SPI0	I/O	
		1	SCL	I2C	I/O	

GPIO Peripheral Select 0 (SIM_GPSB0) Reg. Base + $15	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	GPS_B6		GPS_B5		GPS_B4			GPS_B3		GPS_B2		0	GPS_B1	0	GPS_ B0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

GPIO Peripheral Select 1 (SIM_GPSB1) Register

Bits	Name	Description
8	GPS_B11	**Configure GPIOB11**
		This bit field selects the alternate function for GPIOB11.
		0 COUTB CMPB ANA_OUT (default)
		1 TB1 TRMB I/O
6	GPS_B10	**Configure GPIOB10**
		This bit field selects the alternate function for GPIOB10.
		0 COUTA CMPA ANA_OUT (default)
		1 TB0 TRMB I/O
4	GPS_B9	**Configure GPIOB9**
		This bit field selects the alternate function for GPIOB9.
		0 SDA I2C I/O (default)
		1 CANRX CAN INPUT
2	GPS_B8	**Configure GPIOB8**
		This bit field selects the alternate function for GPIOB8.
		0 SCL I2C I/O (default)
		1 CANTX CAN INPUT
0	GPS_B7	**Configure GPIOB7**
		This bit field selects the alternate function for GPIOB7.
		0 TXD0 SCI0 I/O (default)
		1 SCL I2C I/O

GPIO Peripheral Select 1 (SIM_GPSB1) Reg. Base + $16	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	GPS_B11	0	GPS_B10	0	GPS_B9	0	GPS_B8	0	GPS_ B7
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

GPIO Peripheral Select for GPIOC&D (SIM_GPSCD) Register

Bits	Name	Description
12	GPS_D5	**Configure GPIOD5**
		This bit field selects the alternate function for GPIOD5.
		0 XTAL OSC ANA_OUT (default)
		1 CLKIN OCCS INPUT
4	GPS_C12	**Configure GPIOC12**
		This bit field selects the alternate function for GPIOC12.
		0 ANB4 ADCB ANA_IN (default)
		1 RXD1 SCI1 INPUT
2	GPS_C8	**Configure GPIOC8**
		This bit field selects the alternate function for GPIOC8.
		0 ANA4 CMPA ANA_OUT (default)
		1 TXD1 SCI1 I/O

GPIO Peripheral Select (SIM_GPSCD) Reg. Base + $17	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	GPS_D5	0	0	0	0	0	0	0	GPS_C12	0	GPS_C8	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

Internal Peripheral Select for PWM (SIM_IPSPWM) Register

Please see the following page for continuation of this register

Bits	Name	Description
13	IPS_FAULT2	**Select Input Source for FAULT2**
		This bit field selects the alternate input source to feed PWM input FAULT2.
		0 GPIO (default)
		1 COUTB CMPB
11	IPS_FAULT1	**Select Input Source for FAULT1**
		This bit field selects the alternate input source to feed PWM input FAULT1.
		0 GPIO (default)
		1 COUTA CMPA
8 - 6	IPS_PSRC2	**Select Input Source for PSRC2**
		This bit field selects the alternate input source to feed PWM input PSRC2.
		000 GPIO (default)
		001 TA3 TRMA
		010 ADC Result[2] ADC
		011 COUTA CMPA
		100 COUTB CMPB
		11x Reserved
		1x1 Reserved

Internal Peripheral Select Reg. (SIM_IPSPWM) Base + $18	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	IPS_FAULT2	0	IPS_FAULT1	0	0	IPS_PSRC2			IPS_PSRC1			IPS_PSRC0		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Internal Peripheral Select for PWM (SIM_IPSPWM) Reg. Continued

Bits	Name	Description	
5 - 3	**IPS_PSRC1**	**Select Input Source for PSRC1**	
		This bit field selects the alternate input source to feed PWM input **PSRC1**.	
		000	GPIO (default)
		001	TA3 TMRA
		010	ADC Result[1] ADC
		011	COUTA CMPA
		100	COUTB CMPB
		11x	Reserved
		1x1	Reserved
2 - 0	**IPS_PSRC0**	**Select Input Source for PSRC0**	
		This bit field selects the alternate input source to feed PWM input **PSRC0**.	
		000	GPIO (default)
		001	TA0 TMRA
		010	ADC Result[0] ADC
		011	COUTA CMPA
		100	COUTB CMPB
		11x	Reserved
		1x1	Reserved

Internal Peripheral Select Reg. (SIM_IPSPWM) Base + $18	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	IPS_FAULT2	0	IPS_FAULT1	0	0	IPS_PSRC2			IPS_PSRC1			IPS_PSRC0		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SIM

Internal Peripheral Select for DAC (SIM_IPSDAC) Register

Bits	Name	Description			
6 - 4	**IPS_DSYNC1**	**Select Input Source for NC INput to DAC1**			
		This bit field selects the alternate input source to feed DAC1 SYNC input.			
		000	PIT0	PIT	(default)
		001	PIT1	PIT	
		010	PIT2	PIT	
		011	Reload-Sync	PWM	
		100	TA0	TRMA	
		100	TA1	TRMA	
		11x	Reserved		
2 - 0	**IPS_DSYNC0**	**Select Input Source for PSRC0**			
		This bit field selects the alternate input source to feed DAC0 SYNC input.			
		000	PIT0	PIT	(default)
		001	PIT1	PIT	
		010	PIT2	PIT	
		011	Reload-Sync	PWM	
		100	TA0	TRMA	
		100	TA1	TRMA	
		11x	Reserved		

Internal Peripheral Select (SIM_IPSDAC) Reg. Base + $19	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	IPS_DSYNC1			0	IPS_DSYNC0		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

Programmer: _____

SIM

Internal Peripheral Select for TMRA (SIM_IPSTMRA) Register

Bits	Name	Description
12	IPS_TA3	**Select Source for TA3**
		This bit field selects the alternate input source signal to feed Quad Timer A, input three. If PWM_CR is set, TRMA_CR must also be set in order for TRMA to detect the Reload-Sync signal.
		0 GPIO (default)
		1 Reload-Sync PWM
8	IPS_TA2	**Select Source for TA2**
		This bit field selects the alternate input source signal to feed Quad Timer A, input two.
		0 GPIO (default)
		1 COUTB CMPB
4	IPS_TA1	**Select Source for TA1**
		This bit field selects the alternate input source signal to feed Quad Timer A, input one.
		0 GPIO (default)
		1 COUTA CMPA

Internal Peripheral Select (SIM_IPSTMRA) Reg. Base + $1A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	IPS_ TA3	0	0	0	IPS_ TA2	0	0	0	IPS_ TA1	0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

Sheet

Index

I

J

L

M

N

O

T

U

V

W

X

Y

Z

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

MC56F80XXRM
Rev. 3, 02/2007

freescale™
semiconductor