DSP56F826/827

User Manual

56F800
16-bit Digital Signal Controllers

DSP56F826-827UM
Rev. 3.0
09/2005

freescale.com

freescale™
semiconductor

This manual is one of a set of three documents. You need the following manuals to have complete product information: Family Manual, User's Manual, and Technical Data Sheet.

Order this document by **DSP56F826-827UM - Rev. 5.0**
March, 2005

Summary of Changes and Updates:

Clarified SPI Chapter Section 12.9.1.7
Clarified statement in GPIO Chapter immediately aboveTable 8-2
Converted to Freescale look and feel

TABLE OF CONTENTS

Chapter 1
56F826/827 Overview

Chapter 2
Pin Descriptions

Chapter 3
Memory and Operating Modes

Chapter 4
On-Chip Clock Synthesis (OCCS)

Chapter 5
Interrupt Controller (ITCN)

Chapter 6
Flash Memory Interface (FLASH)

Chapter 7
External Memory Interface (EMI)

Chapter 8
General Purpose Input/Output (GPIO)

Chapter 9
Analog-to-Digital Converter (ADC)

Chapter 10
Serial Communications Interface (SCI)

Chapter 11
Serial Peripheral Interface (SPI)

Chapter 12
Synchronous Serial Interface (SSI)

Chapter 12
Quad Timer Module (TMR)

Chapter 13
Time-of-Day (TOD)

Chapter 15
Reset, Low Voltage, Stop and Wait Operations

Chapter 16
OnCE Module

Chapter 17
JTAG Port

Appendix A
Glossary

Appendix B
Programmer's Sheets

LIST OF FIGURES

LIST OF TABLES

Preface

About This Manual

Features of the 56F826 and 56F82716-bit devices are described in this manual. Details of Memory, Operating modes, and Peripheral modules are documented here. This manual is intended to be used with the *DSP56800 Family Manual* (DSP56800FM), describing the Central Processing Unit (CPU), programming models, and instruction set details. The *Technical Data Sheet* for each part provides electrical specifications as well as timing, pinout, and packaging descriptions.

Audience

Information in this manual is intended to assist design and software engineers to integrate the 56F826 and/or 56F827 digital signal processors into a design and/or while developing application software.

Manual Organization

Manual information is organized into chapters by topic.

Chapter 56F826/827 Overview provides a brief overview of the 56F826/827 devices.

Chapter Pin Descriptions describes pins on the 56F826/827 chips and how those pins are grouped into various interfaces.

Chapter Memory and Operating Modes recounts the On-Chip Memory, structures, registers, and interfaces.

Chapter On-Chip Clock Synthesis (OCCS) establishes the internal oscillator Phase Lock Loop (PLL) and timers distribution chain for the 56F826/827.

Chapter Interrupt Controller (ITCN) details the 56F826/827 External Memory Interface also referenced as Port A.

Chapter Flash Memory Interface (FLASH) describes the Program Flash, Data Flash, and Boot Flash features and registers.

Chapter External Memory Interface (EMI) provides the External Memory Interface available on the 56F826/827.

Chapter General Purpose Input/Output (GPIO) describes how GPIO pins share package pins with other peripherals on the chip.

Chapter Analog-to-Digital Converter (ADC) provides data regarding the package feature in the 56F827 only.

Chapter Serial Communications Interface (SCI) delineates the peripheral's ability to communicate with other devices such as codecs, microprocessors, and peripherals to provide the primary data input path.

Chapter Serial Peripheral Interface (SPI) outlines the ability of the peripheral to communicate with external devices such as Liquid Crystal Displays (LCDs) and Micro Controller Units (MCUs).

Chapter Synchronous Serial Interface (SSI) elaborates on the capabilities of SSI as a part of Port C and how it communicates with devices such as codecs, microprocessors, other devices, and peripherals providing the primary data input path.

Chapter Quad Timer Module (TMR) expands on the available internal Quad Timer devices, including features and registers.

Chapter Time-of-Day (TOD) develops instruction about the sequence of counters to track elapsed time and its ability to track time up to 179.5 years, or 65,535 days, including keeping track of leap year adjustments.

Chapter Reset, Low Voltage, Stop and Wait Operations is devoted to the on-chip Watchdog Timer and the real-time interrupt generator and the modes of operation.

Chapter OnCE Module contains the specifics of the 56F826/827 On-Chip Emulation (OnCE™) module, accessed through the Joint Test Action Group (JTAG) port.

Chapter JTAG Port provides specifics of the 56F826/827 JTAG port.

Chapter Glossary lists an index of abbreviations and acronyms along with their definitions used in this manual.

Appendix B Programmer's Sheets offers programming references and master programming sheets used to program the 56F826/827 registers.

Additional information

- See http//:www.freescale.com/ for the most current BSDL listings.
- See device Techical Data Sheet for package and pin-out information.

Suggested Reading

A list of books is provided here as an aid:

Advanced Topics in Signal Processing, Jae S. Lim and Alan V. Oppenheim (Prentice-Hall: 1988).

Applications of Digital Signal Processing, A. V. Oppenheim (Prentice-Hall: 1978).

Digital Processing of Signals: Theory and Practice, Maurice Bellanger (John Wiley and Sons: 1984).

Digital Signal Processing, Alan V. Oppenheim and Ronald W. Schafer (Prentice-Hall: 1975).

Digital Signal Processing: A System Design Approach, David J. DeFatta, Joseph G. Lucas, and William S. Hodgkiss (John Wiley and Sons: 1988).

Discrete-Time Signal Processing, A. V. Oppenheim and R.W. Schafer (Prentice-Hall: 1989).

Foundations of Digital Signal Processing and Data Analysis, J. A. Cadzow (Macmillan: 1987).

Handbook of Digital Signal Processing, D. F. Elliott (Academic Press: 1987).

Introduction to Digital Signal Processing, John G. Proakis and Dimitris G. Manolakis (Macmillan: 1988).

Multirate Digital Signal Processing, R. E. Crochiere and L. R. Rabiner (Prentice-Hall: 1983).

Signal Processing Algorithms, S. Stearns and R. Davis (Prentice-Hall: 1988).

Signal Processing Handbook, C. H. Chen (Marcel Dekker: 1988).

Signal Processing: The Modern Approach, James V. Candy (McGraw-Hill: 1988).

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*.
- Pins, or signals asserted low, made active when pulled to ground, have an overbar 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: $FFFB is the X memory address for the Interrupt Priority Register (IPR).
- Code examples are displayed in a monospaced font, illustrated below:

```
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. The word *pin* is a generic term for any pin on the chip. They are described as:
  - 1) There is a *reset* pin always written as $\overline{RESET}$, in uppercase, using the over bar.
  - 2) The processor state occurs when the $\overline{RESET}$ pin is asserted is always written as Reset.
  - 3) The word *reset* refers to the reset *function* is written in lowercase with a leading capital letter as grammar dictates.
- The word *pin* is a generic term for any pin on the chip.
- 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}, illustrated in **Table 0-1.**
- Shaded areas in registers represents reserved bits. They are written as zero, ensuring future compatibility.
- Throughout this manual, Data Memory locations are noted as X:$0000 while Program Memory locations are noted as P:$0000 where $ represents a memory location in hex.
- The PWM value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins. Reading PWMVALx reads value in a buffer and not necessarily the value the PWM generator is currently using.

Table 0-1. Pin Conventions

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

1.Values for VIL, VOL, VIH, and VOH are defined by individual product specifications.

Chapter 1
56F826/827 Overview

1.1 Introduction

Differing in size of memories and choice of peripherals, 56F826/827 multi-functional chips offer solutions for a wide variety of applications. The core provides more processing power than any other available control chip while saving design space and money. The 56F826 and 56F827 are members of the Freescale 56800 core-based family of digital signal controllers.

The processing power of these controllers and the functionality of a microcontroller with a flexible set of peripherals are combined on a single chip. This chip design provides an extremely cost-effective and compact solution for a number of uses. The family of chips is designed to provide control for such applications as:

- AC induction motors—for industrial and appliances such as washing machines, HVAC, fans, vacuums, and motor drives
- Brush DC motors—for car window lifters, electric antennas, toys, cordless tools
- Brushless DC motors—used in automotive and appliances including PC fans, ceiling fans, blowers, washing machines, electric power steering systems
- Switched and variable reluctance motors—such as washing machines, electric power steering, dynamic body control, refrigerator compressors, HVAC, fans, vacuums and for the power control for:
  - General converter/inverter applications
  - Uninterruptable power systems (in-line, line interactive, and standby)
  - Inverter output stages (push-pull, half-bridge, and full bridge)

Either the 56F826/827 may be designed into the following applications:

- Automotive control
- Power line modem
- Uninterruptable power supplies
- Telephony system implementation
- Noise cancellation applications
- Home security
- Steppers and encoders
- Temperature regulation
- HVAC applications
- Remote monitoring and control
- Digital telephone answering machine

- Fuel management systems
- Voice enabled appliances
- Cable test equipment
- Electric energy meter with embedded power line modem
- Underwater acoustics
- Glass breakage detection and security systems
- Traffic light control
- Identification tag readers
- Servo drives

1.2 56800 Family Description

The 56800 core is based on a Harvard-style architecture consisting of three execution units operating in parallel. The Microcontroller Unit (MCU) style programming model and optimized instruction set provide straightforward generation of efficient, compact, and control code. The instruction set is highly efficient for C-compilers, facilitating rapid development of optimized control applications.

The 56800 chips support program execution from either internal or external memories, providing two external dedicated interrupt lines and up to 32-General-Purpose Input/Output (GPIO) lines. The controller includes Program Flash and Data Flash, each programmable through the Joint Test Action Group (JTAG) port, with Program RAM and Data RAM. The controller also supports program execution for external memory. The 56800 core is capable of accessing two data operands from the on-chip Data RAM per instruction cycle.

The controller also provides a full set of standard programmable peripherals: Serial Communications Interface (SCI), Serial Peripheral Interface (SPI), and an additional Quad Timer (TMR). Any of these interfaces can be used as a General-Purpose In/Out (GPIO) if those functions are not required. An internal Interrupt Controller and dedicated GPIO, are also included on some of the parts.

1.3 56800 Core Description

The 56800 core consists of functional units operating in parallel, increasing the throughput of the machine. The Harvard-style architecture consists of three execution units operating in parallel. These three execution units allow as many as six operations during each instruction cycle. The instruction set is also highly efficient for C-compilers. Major features of the 56800 core are:

- Efficient 16-bit 56800 family engine with dual Harvard architecture
- As many as 40 Million Instructions Per Second (MIPS) at 80MHz core frequency
- Single-cycle 16 × 16-bit parallel Multiplier-Accumulator (MAC)
- Two 36-bit accumulators, including extension bits
- 16-bit bidirectional barrel shifter
- Parallel instruction set with unique addressing modes
- Hardware DO and REP loops
- Three internal address buses and one External Address Bus (EAB) available
- Four internal data buses and one EAB available
- Instruction set supports both and controller functions
- Controller style addressing modes and instructions for compact code
- Software subroutine and interrupt stack with depth limited only by memory
- JTAG/OnCE™ debug programming interface

1.3.1 56800 Core Block Diagram

An overall block diagram of the 56800 core architecture is illustrated in **Figure 1-1.** The 56800 core is fed by an internal program and Data Memory, an External Memory Interface (EMI), and various peripherals suitable for embedded applications. The blocks include:

- Data Arithmetic Logic Unit (Data ALU)
- Address Generation Unit (AGU)
- Program controller and hardware looping unit
- Bit Manipulation Unit
- On-Chip Emulation (OnCE) port
- Interrupt Controller
- External Bus Bridge
- Address buses
- Data buses

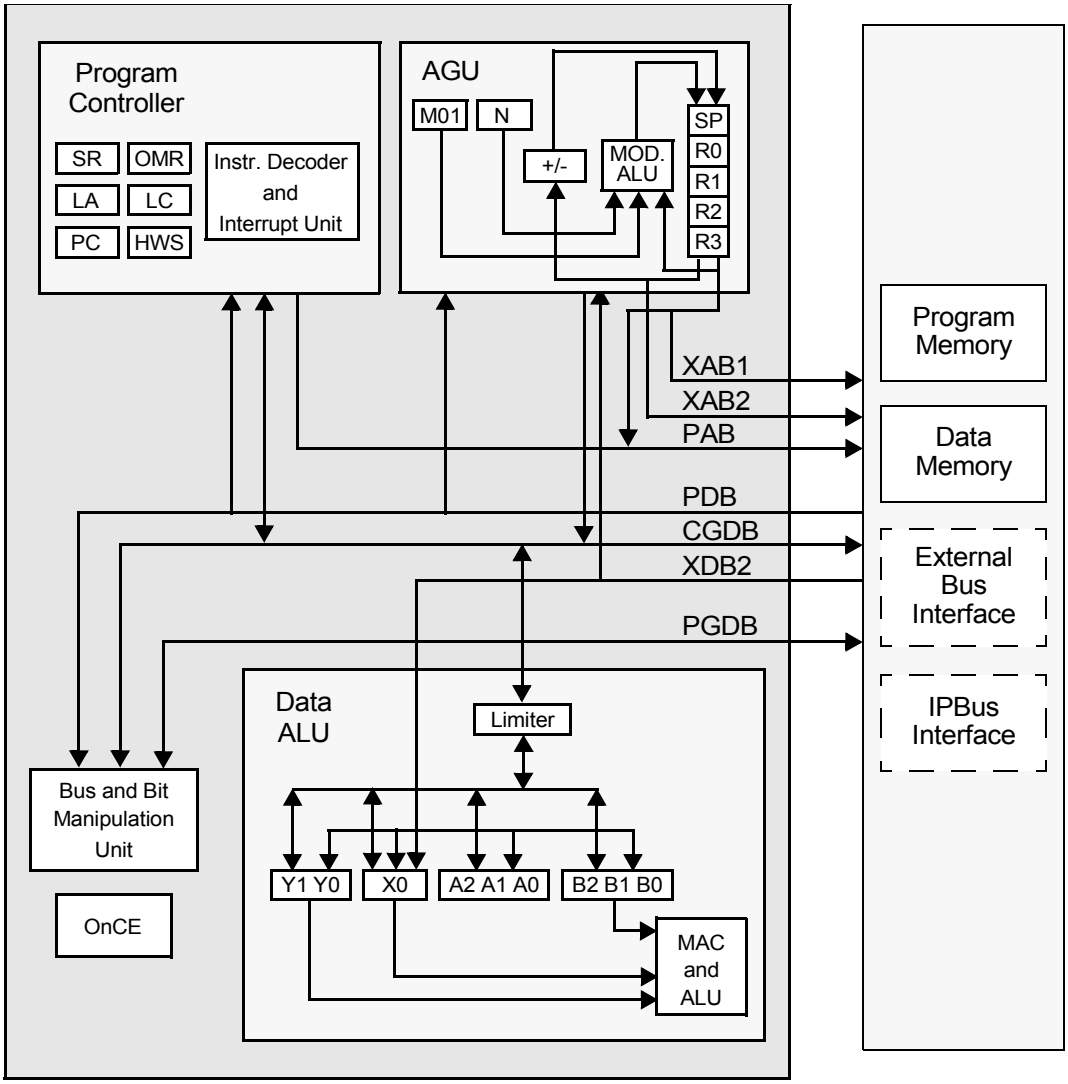


Figure 1-1. 56800 Core Block Diagram

The Program Controller, AGU and Data ALU each contain a discrete register set and control logic so each can operate independently and in parallel with the others. Likewise, each functional unit interfaces with other units, with memory, and with memory-mapped peripherals over the core's internal address and data buses, illustrated in **Figure 1-2.**

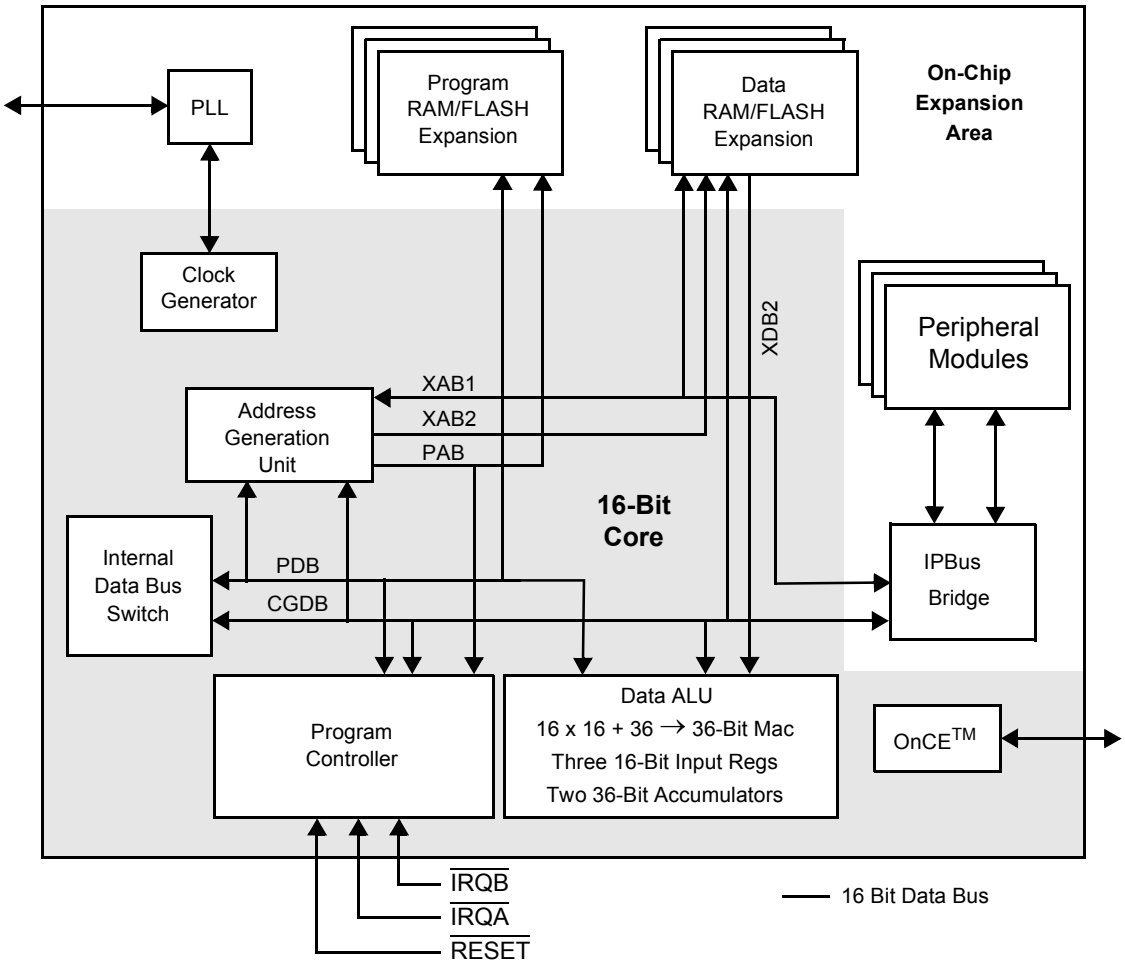


Figure 1-2. 56800 Bus Block Diagram

It is possible in a single instruction cycle for the program controller to be fetching a first instruction, the AGU to generate two addresses for a second instruction, and the Data ALU to perform a multiply in a third instruction. In a similar manner, the Bit Manipulation Unit (BMU) can perform an operation of the third instruction previously described instead of the multiplication in the Data ALU. The architecture is pipelined to take advantage of the parallel units and significantly decrease the execution time of each instruction.

1.4 Architectural Overview

The 56F826/827 consists of the 56800 core, Program and Data Memory, and peripherals useful for embedded control applications. Block diagrams for each chip describing the differences in available peripheral sets and memory are illustrated in **Figure 1-3** and **Figure 1-4.**

1.5 56F826 Description

The 56F826 is a member of the 56800 core-based family of devices. On a single chip, it combines the processing power of a controller and the functionality of a microcontroller with a flexible set of peripherals creating an extremely cost-effective solution for general-purpose applications. Because of its low cost, configuration flexibility and compact program code, the 56F826 is well-suited for many applications such as:

- Noise suppression
- Identification tag readers
- Sonic/subsonic detectors
- Security access devices
- Remote metering
- Sonic alarms
- Point of Sale (POS) terminals
- Feature phones

The 56F826 supports program execution from either internal or external memories. Two data operands can be accessed from the on-chip Data RAM per instruction cycle. Both chips also provide two external dedicated interrupt lines and up to 46-GPIO lines, depending on peripheral configuration.

The 56F826 controller includes 31.5K words of Program Flash and 2K words of Data Flash, each programmable through the JTAG port, with 512 words of Program RAM, and 4K words of Data RAM. The controller also supports program execution from external memory.

The 56F826 incorporates a total of 2K words of Boot Flash for easy inclusion of field-programmable software routines, used to program the main Program and Data Flash memory areas. Both Program and Data Flash memories can be independently bulk-erased or erased in page sizes of 256 words. The Boot Flash memory can also be either bulk- or page-erased.

1.5.1 56F826 Features

- Up to 40MIPS at 80MHz core frequency
- MCU functionality in a unified, C-efficient architecture
- Hardware DO and REP loops
- 31.5K × 16-bit words Program Flash
- 512 × 16-bit words Program RAM
- 2K × 16-bit words Data Flash Memory
- 4K × 16-bit words Data RAM
- 2K × 16-bit words Boot Flash Memory
- Up to 64K × 16-bit words each of external memory expansion for Program and Data memory
- One Serial Port Interface (SPI)
- One additional SPI, or two optional Serial Communication Interfaces (SCIs)
- One Synchronous Serial Interface (SSI)
- One General-Purpose Quad Timer
- JTAG/OnCE for debugging
- 16 dedicated and 30 shared GPIO
- Time-of-Day (TOD) timer
- 100-pin LQFP package

1.5.2 56F826 Benefits

- Flash Memory provides reliable, non-volatile memory storage, thereby eliminating required external storage devices.
- Easy to program with flexible application development tools
- Optimized for C-Compiler efficiency
- Simple updating of Flash Memory through SPI, SCI, or OnCE using on-chip boot-loader
- Supports 9-bit communication protocol
- Simple port interface with other asynchronous serial communication devices
- Simple port interface with other asynchronous serial peripheral communication devices and off-chip EE memory
- Sophisticated debugging using OnCE to view core, peripheral, and memory contents

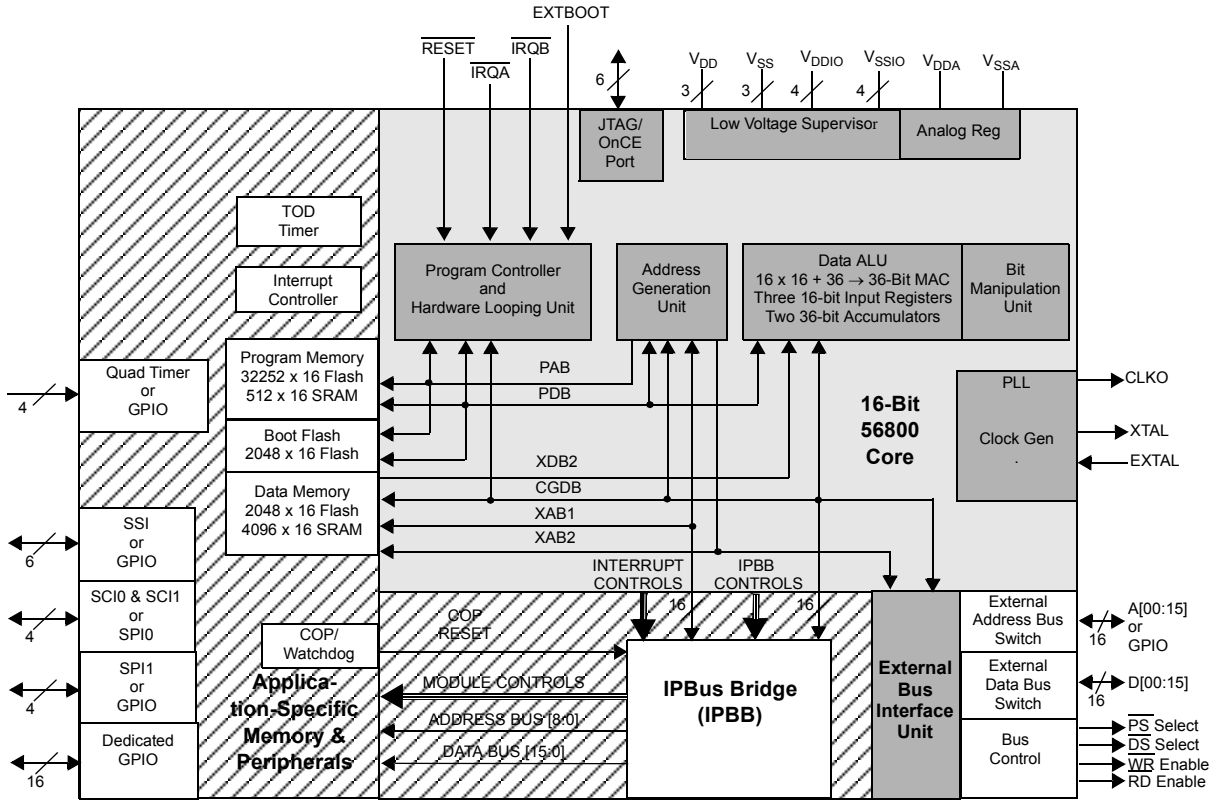


Figure 1-3. 56F826 Block Diagram

1.6 56F827 Description

The 56F827 is a member of the Harvard-style architecture 56800 core-based family of devices. It combines the processing power of a controller and the functionality of a microcontroller with a flexible set of peripherals, creating an extremely cost-effective solution for general-purpose applications. Because of its low cost, configuration flexibility, and compact program code, the 56F827 is well-suited for many applications. Two data operands per instruction cycle can be accessed from the on-chip Data RAM per instruction cycle. The 56F827 incorporates easy customer field-programmable software routines used to develop the main program. The 56F827 is well-suited for many applications such as:

- Noise suppression
- Identification tag readers
- Sonic/subsonic detectors
- Security access devices
- Remote metering

- Sonic alarms
- Point of Sale (POS) terminals
- Feature phones

The 56800 core consists of three execution units operating in parallel, allowing as many as six operations per instruction cycle. The microprocessor-style programming model and optimized instruction set allow forthright generation of efficient, compact code for MCU applications. The instruction set is also highly efficient for C-Compilers to enable rapid development of optimized control applications.

The 56F827 supports program execution from either internal or external memories. Two data operands can be accessed from the on-chip Data RAM per instruction cycle. The 56F827 also provides two external dedicated interrupt lines, and up to 64-GPIO lines, depending on peripheral configuration.

The 56F827 controller includes 63K words of Program Flash and 4K words of Data Flash, each programmable through the JTAG port, with 1K words of Program RAM, and 4K words of Data RAM. It also supports program execution from external memory.

This controller also provides a full set of standard programmable peripherals including one each SSI, SCI, SPI, the option to select a second SPI or two SCIs, and one Quad Timer (TMR). The SSI, SPI, and TMR can be used as GPIOs if those function are not required.

1.6.1 56F827 Features

- Up to 40MIPS at 80MHz core frequency
- MCU functionality in a unified, C-efficient architecture
- Hardware DO and REP loops
- 63K × 16-bit words Program Flash Memory
- 1K × 16-bit words Program RAM
- 4K × 16-bit words Data Flash Memory
- 4K × 16-bit words Data RAM
- Up to 64K × 16-bit words external memory expansion for program and Data Memory
- JTAG/OnCE for debugging
- MCU-friendly instruction set supports controller functions: MAC, Bit Manipulation Unit (BMU), 14 addressing modes
- 8-channel programmable chip select
- 10-channel, 12-bit Analog-to-Digital Converter (ADC)

- Synchronous Serial Interface (SSI)
- Serial Port Interface (SPI)
- Serial Communication Interface (SCI)
- General-Purpose Quad Timer (TMR)
- Time-of-Day (TOD) timer
- 128-pin LQFP package

1.6.2 56F827 Benefits

- Flash Memory provides reliable, non-volatile memory storage, thus eliminating required external storage devices.
- Easy to program with flexible application development tools
- Optimized for C-Compiler efficiency
- Simple updating of Flash Memory through SPI, SCI, or OnCE using on-chip boot loader
- Supports 9-bit communication protocol
- Simple interface with other asynchronous serial communication devices and off-chip EE memory
- Capable of Analog-to-Digital Converter (ADC) functionality utilizing Quad Timer to provide independence to each channel timer
- Sophisticated debugging using OnCE to view the core, peripheral, and memory contents
- Power-saving analog shut-down mode

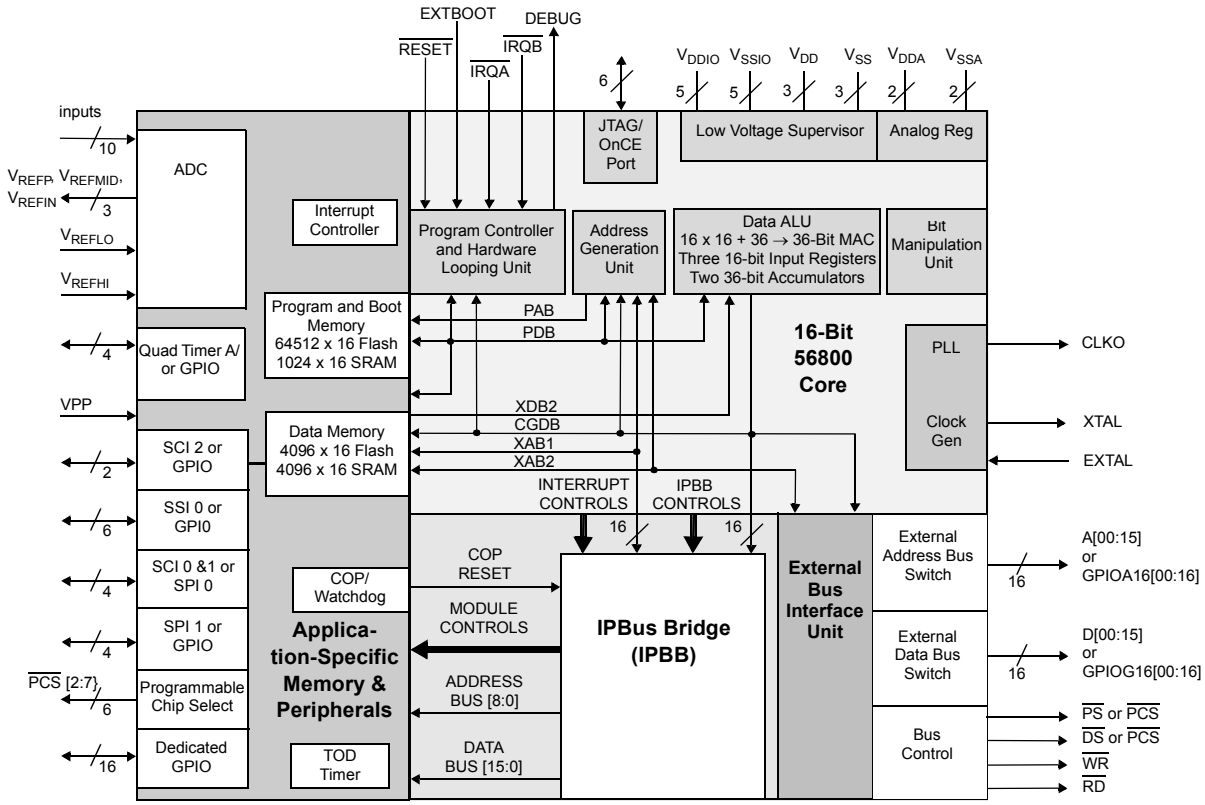


Figure 1-4. 56F827 Block Diagram

1.7 56F826/827 Features

Table 1-1. Feature Matrix

Feature	56F826	56F827
Speed (MIPS)	40	40
Program Flash	31.5K X 16	63K X 16
Data Flash	2K X 16	4K X 16
Program RAM	512 X 16	1K X 16
Data RAM	4K X 16	4K X 16
Boot Flash	2K X 16	—
ADC	—	1
Oscillator	External Crystal	External Crystal
PLL	1	1
SCI	2[1]	3[1] or 1[2]
SPI	1 or 2[1]	1 or 2[1]
TOD	1	1
SSI	1	1
Watchdog	1	1
General-Purpose Timer	1 4-pin Quad Timer	1 4-pin Quad Timer
Dedicated GPIO	16	16
Shared GPIO	32	48
Muxed GPIO	48	64
JTAG/OnCE	1	1
Interrupt Controller	1	1
Programmable Chip Selects	—	8
External Bus	32	32
Package	100 LQFP	128 LQFP

1. Two SCIs can be configured as an additional SPI.
2. Can be used as GPIO

1.7.1 Data Arithmetic Logic Unit (Data ALU)

The Data ALU performs all of the arithmetic and logical operations on data operands. It contains:

- Three 16-bit input registers
- Two 32-bit accumulator registers
- Two 4-bit accumulator extension registers
- One parallel, single cycle, non-pipelined MAC unit
- An accumulator shifter
- One data limiter
- One MAC output limiter
- One 16-bit barrel shifter

The Data ALU is capable of performing the following in one instruction cycle:

- Multiplication
- Multiply-accumulate with positive or negative accumulation
- Addition
- Subtraction
- Shifting
- Logical operations

Arithmetic operations are completed using two's-complement fractional or integer arithmetic. Support is also provided for unsigned and multi-precision arithmetic.

Data ALU source operands can be 16, 32, or 36 bits and can originate from input registers and/or accumulators. ALU results are stored in one of the accumulators. Additionally, some arithmetic instructions store their 16-bit results in any of the three Data ALU input registers, or write directly to memory. Arithmetic operations and shifts have a 16-bit or 36-bit result, and logical operations are performed on 16-bit operands yielding 16-bit results. Data ALU registers can be read or written by the Core Global Data Bus (CGDB) as 16-bit operands, and the X0 register can also be written by the X Data Bus two (XDB2) with a 16-bit operand.

1.7.2 Address Generation Unit (AGU)

The Address Generation Unit (AGU) performs all of the effective address calculations and address storage necessary to address data operands in memory. This unit operates in parallel with other chip resources to minimize address generation overhead. It contains two ALUs, allowing the generation of up to two 16-bit addresses every instruction cycle—one for either the XAB1 or Program Address Bus (PAB) and one for the XAB2 bus. The ALU can directly address 65,536 locations on the XAB1 or XAB2 bus. The ALU can also directly address 65,536 locations on the PAB, for a total capability of 131,072 16-bit data words. Hooks are provided on the 56800 core allowing expansion of address space. Its arithmetic unit can perform linear and modulo arithmetic.

1.7.3 Program Controller and Hardware Looping Unit

The Program Controller performs:

- Instruction prefetch
- Instruction decoding
- Hardware loop control
- Interrupt (exception) processing

Instruction execution is carried out in other core units, such as the Data ALU or AGU. The program controller consists of a Program Counter (PC) unit, Hardware Looping Control Logic, Interrupt Control Logic, and Status and Control registers.

Two interrupt control pins provide input to the program interrupt controller. The External Interrupt Request A ($\overline{\text{IRQA}}$) pin and the External Interrupt Request B ($\overline{\text{IRQB}}$) pin receive interrupt requests from external sources.

The $\overline{\text{RESET}}$ pin resets the 56F826/827. When asserted, it initializes the chip and places it in the Reset state. When the $\overline{\text{RESET}}$ pin is deasserted, the initial chip operating mode is latched into the Operating Mode Register (OMR) based upon the value present on the EXTBOOT pin. The 56F826/827 is internally pulled low. Please refer to **Section 3.3.2** for additional details.

1.7.4 Bit Manipulation Unit (BMU)

The Bit Manipulation Unit (BMU) performs bit field manipulations on X memory words, peripheral registers, and registers on the DSP56800 core. It is capable of testing, setting, clearing, or inverting any bits specified in a 16-bit mask. This unit tests bits on the upper or lower byte of a 16-bit word for branch on-bit field instructions, meaning the mask tests a maximum of eight bits at a time.

Transfers between buses are accomplished in the bus unit. The bus unit is similar to a switch matrix, and can connect any two of the three data buses together without adding any pipeline delays. This procedure is required for transferring a core register to a peripheral register because the core register is connected to the CGDB bus.

As a general rule, when reading any register less than 16 bits wide, unused bits are read as zero. Reserved and unused bits should always be written with a zero, ensuring future compatibility.

1.7.5 Address and Data Buses

Addresses are provided to the internal X Data Memory on two unidirectional, 16-bit buses—(XAB1, and XAB2). Program memory addresses are provided on the unidirectional 16-bit PAB.

Note: The XAB1 can provide addresses for accessing both internal and external memory, whereas the XAB2 can only provide addresses for accessing internal *read-only* memory. The External Address Bus (EAB) provides addresses for external memory.

Data movement on both the 56F80x occurs over three bidirectional, 16-bit buses and at least one unidirectional 16-bit bus:

- CGDB bidirectional
- PDB bidirectional

- PGDB bidirectional
- XDB2 unidirectional

When one memory access is performed, data transfer between the Data ALU and the X Data Memory occurs over the CGDB. When two simultaneous memory reads are performed, transfers occur over the CGDB and the XDB2. All other data transfers to core blocks occur over the CGDB, transferring all memory to and from peripherals over the PGDB. Instruction word fetches occur simultaneously over the PDB. The External Data Bus (EDB) provides bidirectional access to external Data Memory.

The bus structure supports:

- General register-to-register
- Register-to-memory
- Memory-to-register transfers

Each can transfer up to three 16-bit words in the same instruction cycle. Transfers between buses are accomplished in the Bit Manipulation Unit (BMU). **Table 1-2** lists the address and data buses for the 56800 core.

Table 1-2. 56800 Address and Data Buses

Bus	Bus Name	Bus Width	Direction	Use
XAB1	X Address Bus 1	16-bit	Unidirectional	Internal and External Memory Address
XAB2	X Address Bus 2	16-bit	Unidirectional	Internal Memory Address
PAB	Program Address Bus	16-bit	Unidirectional	Internal Memory Address
EAB	External Address Bus	16-bit	Unidirectional	External Memory Address
CGDB	Core Global Data Bus	16-bit	Unidirectional	Internal Data Movement
PDB	Program Data Bus	16-bit	Unidirectional	Instruction Word Fetches
PGDB	Peripheral Global Data Bus	16-bit	Unidirectional	Internal Data Movement
XDB2	X Data Bus 2	16-bit	Unidirectional	Internal Data Movement
EDB	External Data Bus	16-bit	Bidirectional	External Data Movement

1.7.6 On-Chip Emulation (OnCE) Module

The OnCE module allows interaction in a debug environment with the 56800 core and its peripherals. Its capabilities include:

- Examining registers
- Setting breakpoints in memory
- Stepping or tracing instructions

It provides simple, inexpensive, and speed independent access to the 56800 core for sophisticated debugging and economical system development. The JTAG port allows access to the OnCE module, through the 56F826/827, to its target system where it retains debug control without sacrificing other user accessible on-chip resources. This technique eliminates costly cabling and the access to processor pins required by traditional emulator systems. The OnCE interface is described in **Chapter 16.**

1.7.7 On-Chip Clock Synthesis (OCCS) Block

The Clock Synthesis module generates the clocking for the 56F826/827. It generates clocks used by the 56800 core and 56F826/827 peripherals. The module contains a PLL capable of multiplying up the frequency or being bypassed. A prescaler/divider distributes clocks to peripherals and to lower power consumption on the 56F826/827. Further, the prescaler divider selects which clock, if any, is routed to the CLKO pin of both chips.

1.7.8 Oscillators

The 56F826/827 are clocked either from an external crystal or external clock generator input. The crystal oscillator uses a 4MHz crystal:

- Optionally, can use ceramic resonator in place of crystal
- Optionally, can be divided down by a programmable prescaler

1.7.9 Phase Locked Loop (PLL)

- The PLL will generate an interrupt to instruct the device to gracefully shutdown the system in the event reference clock is stopped.
- The PLL is designed to run for at least 100 instruction cycles if the oscillator source is removed.
- The PLL generates output frequencies up to 80MHz.
- The PLL can be bypassed to use an oscillator or prescalar outputs directly.

A clock gear shifter guarantees smooth transition from one clock source to the next during normal operation. Clock sources available for Normal operation include:

- Prescaler output
- Postscaler output
- Programmable prescalar output, a divided down version of the oscillator clock— legal divisors are one, two, four, or eight

1.7.10 Resets

- Integrated POR release occurs when V_{DD} exceeds 1.8V.
- Integrated low voltage detector generates an interrupt when V_{DD} drops below 2.2V in the core or 2.7V in the I/O.

Note: Voltage level is designed to allow the host processor to continue running at speed. There is nominally about 50mV of hysteresis present on each of the low voltage interrupt inputs.

1.7.11 Energy Supply Voltages

Dual power supply:

- Chip I/O supply voltage = 3.3V
- Core voltage = 2.5V plus or minus 10 percent

1.7.12 IPBus Bridge

The IPBus Bridge converts Data Memory and interrupt interfaces to IPBus-compliant interfaces for peripherals.

This IPBus Bridge permits communication between the core and peripherals, utilizing the CGDB for data and XAB for addresses. All peripherals, except the COP/Watchdog Timer, and TOD Timer run off the IPBus Clock. (The COP/Watchdog Timer runs at half of the system, or processor frequency.) The IPBus is 40MHz while the oscillator frequency is 80MHz.

The IPBus Bridge translates the four-phase clock bus protocol of the 56800 core to the single clock environment of the IPBus protocol used to communicate with the peripherals. All IPBus transfers are completed in one core clock cycle.

Unaligned word,16-bit and byte, or long word 32-bit accesses are not supported on this IPBus. The 56800 supports only 16-bit word transfers on word boundaries.

The IPBus Bridge also provides upper level address decoding and peripheral module enable generation.

1.8 Memory Modules

Harvard architecture permits as many as three simultaneous accesses to program and Data Memory.

- On-chip memory, depending on specific chip selected
  - **56F826**
    - 31.5K words of Program Flash
    - 512 words of Program RAM
    - 2K words of Data Flash
    - 4K words of Data RAM
    - 2K words of Boot Flash
  - **56F827**
    - 63K words of Program Flash
    - 1K words of Program RAM
    - 4K words of Data Flash
    - 4K words of Data RAM

1.8.1 Program Flash

- Single port memory is compatible with the pipelined program bus structure
- Split-gate cell, NOR type structure
- Single-cycle reads at 40MHz
- Intelligent word programming feature
- Memory is organized into a two row information block (equals 64 bytes) and main memory block
- Pages are 256 words long
- Intelligent page erase and mass erase modes
- Can be programmed and erased under software control
- Optional interrupt on completion of intelligent program and erase functions

1.8.2 Program RAM

- Single port RAM is compatible with the pipelined program bus structure
- Single cycle reads at 40MHz

1.8.3 Data Flash

- Single port memory is compatible with the pipelined Data Bus structure
- Muxing allows this memory to be read from PAB, XAB1 or XAB2 databases
- Split-gate cell, NOR type structure
- Single-cycle reads at 40MHz across the automotive temperature range
- Intelligent word programming feature
- Intelligent page erase and mass erase modes
- Can be programmed under software control in the user's system

1.8.4 Data RAM

- Single read, dual read, or single-write memory compatible with the pipelined data bus structure
- Single cycle reads/writes at 40MHz

1.9 56F826/827 Peripheral Blocks

The 56F826/827 provides the following peripheral blocks:

- One 10-channel, 12-bit, ADC (56F827 only)
- Up to eight programmable chip select signals available (56F827 only)
- One general-purpose Quad Timer totaling four pins
- One dedicated SPI, plus a second multiplexed with two Serial Communications Interfaces (SCI0 and SCI1) totaling four GPIO pins
- Up to three SCIs with two pins, or six additional GPIO pins (56F827 only)
- Two SPIs, SPI0 and SPI1, with configurable four-pin port
- The 56F827 also has a third SPI (SPI2) available as four additional GPIO pins
- One SSI with configurable six-pin port, or six additional GPIO lines
- Sixteen dedicated and 48 (56F826), or 64 (56F827) multiplexed GPIO pins
- A COP/Watchdog Timer
- Two external interrupt pins
- Eight-channel programmable chip selects (56F827 only)
- External $\overline{\text{RESET}}$ pin for hardware reset
- JTAG/OnCE for unobtrusive, processor speed-independent debugging
- Software-programmable, PLL-based frequency synthesizer for the core clock

- Fabricated in high-density CMOS with 5V tolerant, TTL-compatible digital inputs
- One TOD timer

1.10 Peripheral Descriptions

The IPBus Bridge converts Data Memory and interrupt interfaces to the IPBus-compliant interface for peripherals. The IPBus Bridge permits communication between the core and peripherals utilizing the CGDB for data and XAB for addresses.

Peripherals run off the IPBus clock at 40MHz. The IPBus clock frequency is half of the oscillator frequency. Interrupt Priority Register (IPR) and BCR run at 80MHz while the COP/Watchdog Timer runs at 40MHz.

1.10.1 External Memory Interface (EMI)

The 56F826/827 provide an External Memory Interface (EMI). This port provides a total of 36 pins—16 pins for an External Address Bus (EAB), 16 pins for an External Data Bus (EDB), and four pins for bus control.

1.10.2 Programmable Chip Select

The primary function of the chip select is to provide the chip enable for external memory and peripheral devices. All chip select pins are software programmable.

- 56F826
  - — No chip select
- 56F827
  - — Up to eight programmable chip select signal available

The programmable chip select provide the following features:

- Reduced system complexity
- Eight programmable active-low chip select
- Control for external boot device
- Programmable base addresses with programmable block sizes
- Maximum block size = 64K (16-bit) words
- Minimum block size = 0.5K (16-bit) words
- Wait states programmable through BCR register of DSP56800 core. Changing number of wait states affects all chip select together.
- Each chip select can be assigned either program memory or Data Memory or both.

- All chip selects are assigned for both read/write.
- Each chip select can be individually enabled or disabled.

1.10.3 General-Purpose Input/Output Port (GPIO)

This port is configured to make it possible to generate interrupts when a transition is detected on any of its lower eight pins.

- 56F826
  - 32 shared GPIO pins, 48 when multiplexed with other peripherals
  - 16 dedicated GPIO pins
- 56F827
  - 48 shared GPIO pins, 64 when multiplexed with other peripherals
  - 16 dedicated GPIO pins
  - Each bit can be individually configured as an input or output
  - Each bit can be configured as an interrupt input

1.10.4 Serial Peripheral Interface (SPI)

The SPI is an independent serial communications subsystem allowing 56F826/827 microcontrollers to communicate synchronously with peripheral devices such as LCD display drivers, A/D subsystems, and MCU microprocessors. The SPI is also capable of interprocessor communication in a multiple master system. The SPI system can be configured as either a master or a slave device with high data rates. The SPI works in a demand-driven mode. In Master mode, a transfer is initiated when data is written to the SPI Data register. In Slave mode, a transfer is initiated by the reception of a clock signal.

- The 56F826 chip can be configured to have one or two SPIs; the 56F827 chip can be configured to have up to three SPIs
- Full-duplex synchronous operation via four-wire interface
- Configurable for either master or slave operation
- Multiple slaves may be enabled using GPIO pins
- Double-buffered operation with separate Transmit and Receive registers

1.10.5 COP/Watchdog Timer and Modes of Operation Module

The COP module provides two separate functions:

1. A Watchdog Timer
2. An interrupt generator

These two functions monitor processor activity and provide an automatic reset signal if a failure occurs. Both functions are contained in the same block because the input clock for both comes from a common clock divider:

- 12-bit counter to provide 4096 different timeout periods
- COP timebase is the CPU clock divided by 16384
- At 80MHz, minimum timeout period is 204.8µs, maximum is 839ms, with a resolution of 204.8µs

1.10.6 JTAG/OnCE Port

The JTAG/OnCE port introduces the 56F826/827 into a target system while retaining debug control. The JTAG port provides board-level testing capability for scan-based emulation compatible with the IEEE 1149.1a-1993 IEEE Standard Test Access Port and Boundary Scan Architecture specification defined by the JTAG. Five dedicated pins interface to a Test Access Port (TAP) containing a 16-state controller.

The OnCE module interacts in a debug environment with the 56800 core and its peripherals nonintrusively. Its capabilities include:

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

The OnCE module provides simple, inexpensive, and speed-independent access to the DSP56800 core for sophisticated debugging and economical system development. The JTAG/OnCE port provides access to the OnCE module. Through the 56F80x to its target system, it retains debug control without sacrificing other accessible on-chip resources.

1.10.7 Quad Timer Module (TMR)

- 56F826
  - — Timer A with four pins
- 56F827
  - — Timer A with four pins

Quad timer features:

- Four channels, independently programmable as input capture or output compare

- Each channel has its own timebase
- Each of four channels can use any of four timer inputs
- Rising-edge, falling-edge, or any edge input capture trigger
- Set, clear, or toggle output capture action
- Programmable clock sources and frequencies, including external clock
- External synchronization input

1.10.8 Analog-to-Digital Converter (ADC)

- 56F826
  - — No ADC
- 56F827
  - — One 10 channel, 12-bit, ADC

1.10.9 Serial Communications Interface (SCI)

- 56F826 (Two SCI maximum)
  - — Two optional SCIs, each with two pins, or one SPI
- 56F827 (Three SCI maximum)
  - — Two optional SCIs, each with two pins, or one SPI and
  - — One SCI with two pins (or two additional GPIO pins)

SCI features:

- — Asynchronous operation
- — Baud rate generation
- — IR interface support

1.10.10 Synchronous Serial Interface (SSI)

The SSI is a full-duplex, serial port designed to allow the device to communicate with a variety of serial devices, including industry-standard codecs, other hybrid controllers, microprocessors, and peripherals to implement the Serial Peripheral Interface (SPI).

SSI features:

- SSI with configurable six-pin port (or six additional GPIO lines)
- Normal mode operation using frame sync
- Gated Clock mode operation requiring no frame sync

- Program options for frame sync and clock generation
- Programmable word length from eight to 16 bits

1.10.11 Time-of-Day (TOD)

- 56F826/56F827
  - — Sequence counters to track seconds, minutes, hours, and days
  - — Works with crystal frequency of two to 4MHz
  - — Generates interrupt capability of pulling the part out of Sleep
  - — Capability to track time up to 179.5 years
  - — Configurable to generate an alarm at a designated time

1.10.12 Peripheral Interrupts

The peripherals on the 56F826/827 use the interrupt channels found on the 56800 core. Each peripheral has its own interrupt vector, often more than one interrupt vector for each peripheral, and can be selectively enabled or disabled via the IPR found on the 56800 core and the Priority Level Registers (PLRs) found in the Interrupt Controller (ITCN). **Chapter 5** provides more details on interrupt vectors.

Chapter 2
Pin Descriptions

2.1 Introduction

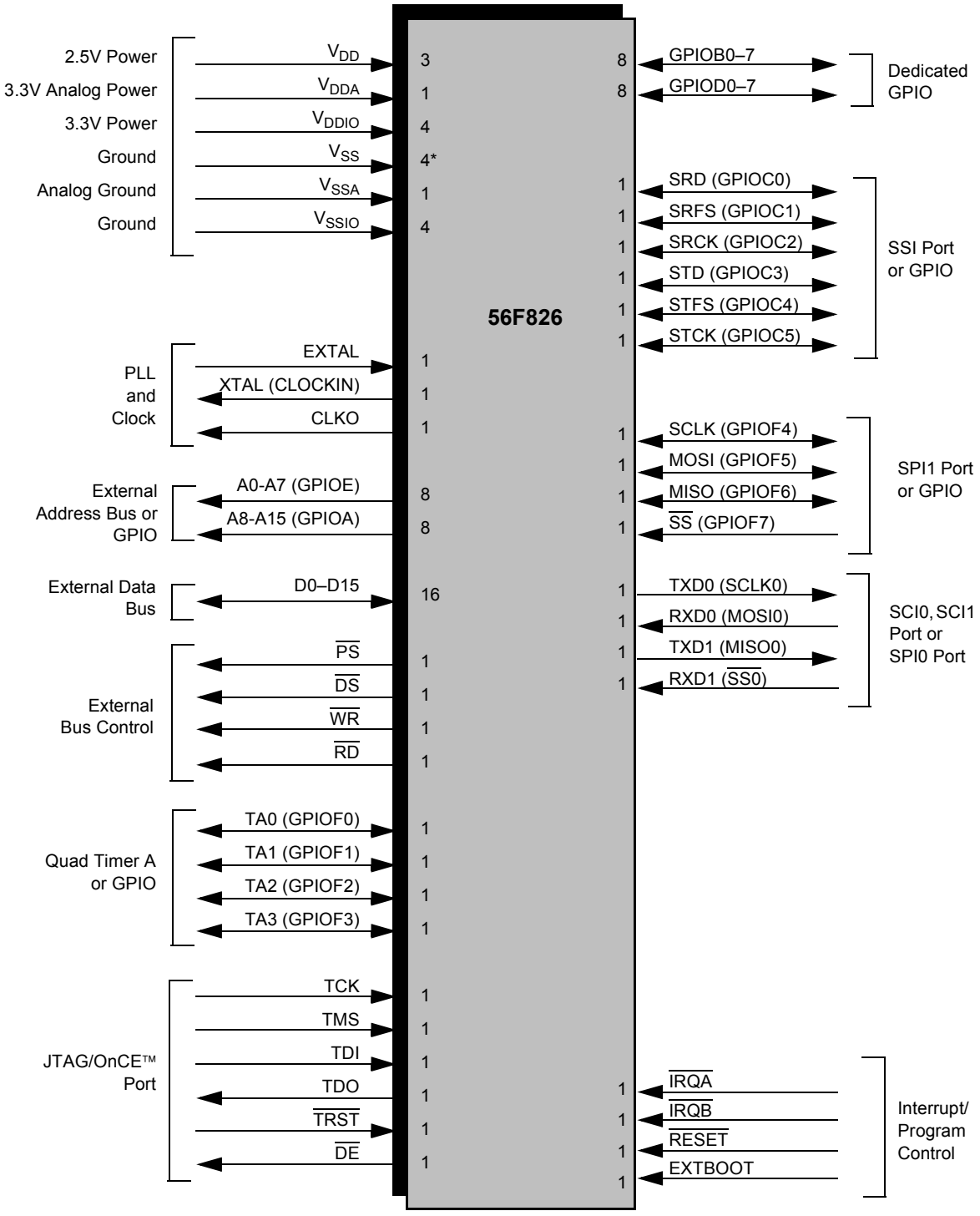
The input and output signals of the 56F826/827 are organized into functional groups illustrated in **Table 2-1.** Each row of **Table 2-2** through **Table 2-17** describes the signals present on an individual pin. The 56F826/827 have SCI0 and SCI1 pins labeled TXD0, TXD1, RXD0 and RXD1.

Note: Some pins can carry more than one signal, depending on chip configuration.

Table 2-1. 56F826/827 Functional Group Pin Allocations

Functional Group	826 Number of Pins	827 Number of Pins	Detailed Description
Power (V_{DD}, V_{DDIO}, V_{DDA} or V_{DDA_ADC})	(3,4,1,0)	(3,5,1,1)	Table 2-2
Ground (V_{SS}, V_{SSIO}, V_{SSA}, or V_{SSA_ADC})	(3,4,1,0)	(3,5,1,1)	Table 2-3
V_{PP}	-	1	Table 2-4
PLL and Clock	3	3	Table 2-5
Address Bus[1]	16	16	Table 2-6
Data Bus[1]	16	16	Table 2-7
Bus Control	4	4	Table 2-8
Quad Timer Module Ports[1]	4	4	Table 2-9
JTAG/On-Chip Emulation (OnCE)	6	6	Table 2-10
Dedicated General Purpose Input/Output	16	16	Table 2-11
Synchronous Serial Interface (SSI) Port[1]	6	6	Table 2-12
Serial Peripheral Interface (SPI) Port[1]	4	4	Table 2-13
Serial Communications Interface (SCI) Ports[1]	-	2	Table 2-14
Serial Communications Interface (SCI) Ports[2]	4	4	Table 2-15
ADC Port	-	15	Table 2-16
Programmable Chip Select ($\overline{PCS}$)[3]	-	6	Table 2-17
Interrupt and Program Control	5	5	Table 2-18

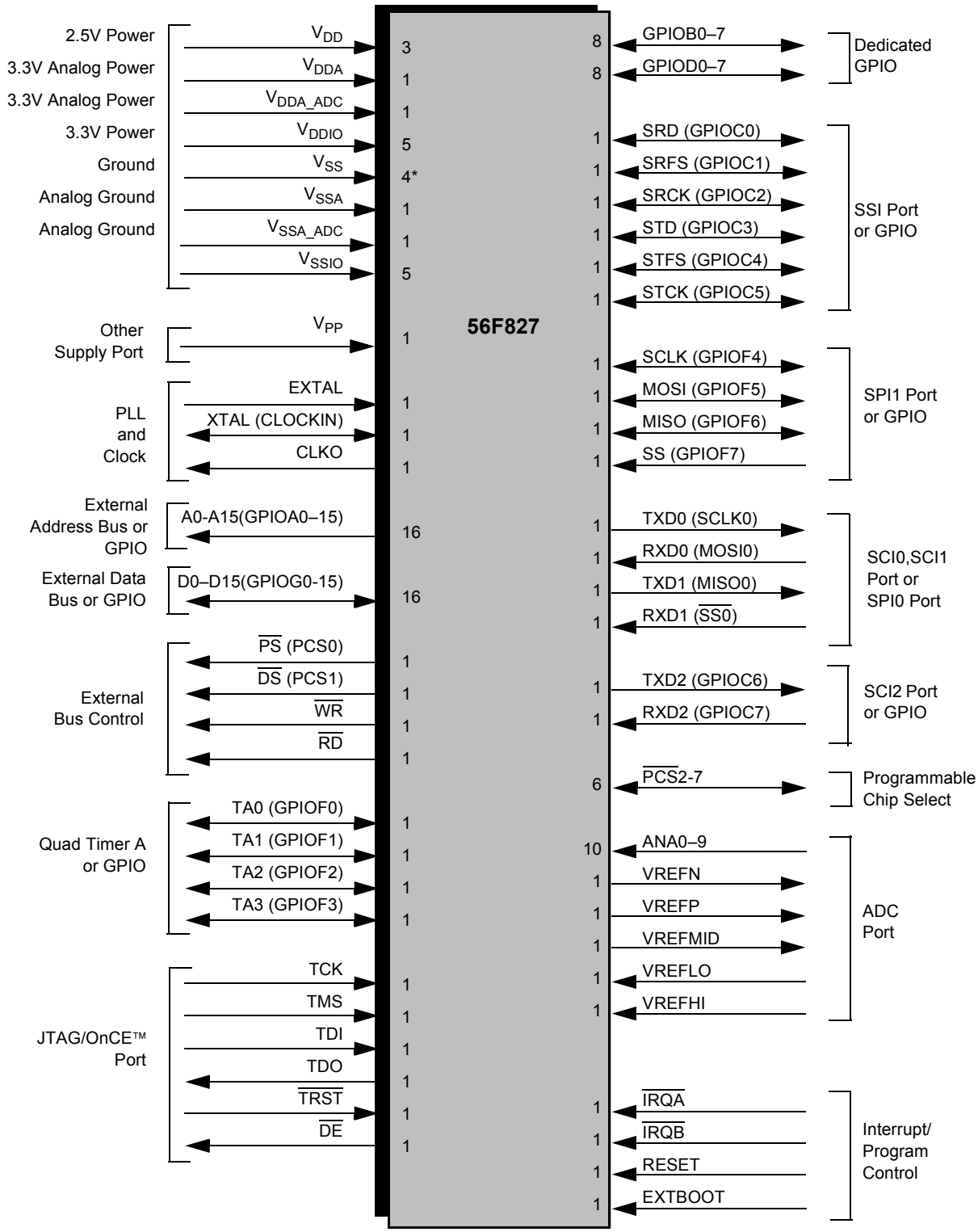
1. Alternately, GPIO pins (56F826 SCIs cannot be used as GPIO)
2. Alternately, two SCIs can be used as an SPI
3. In addition, two Bus Control pins can be programmed as PCS [0-1]



*Includes TCS pin, which is reserved for factory use and is tied to V_{SS}

Figure 2-1. 56F826 Functional Group Pin Allocations[1]

1. Alternate pin functionality is shown in parentheses



*Includes TCS pin, which is reserved for factory use and is tied to VSS

Figure 2-2. 56F827 Functional Group Pin Allocations[1]

1. Alternate pin functionality is shown in parentheses

2.2 Power and Ground Signals

Table 2-2. Power Inputs

No. of Pins		Signal Name	Signal Description
826	827		
3	3	V_{DD}	**Power**—These pins provide power to the core of the chip, are generally connected to a 2.5V supply.
1	1	V_{DDA}	**Analog Power**—This pin is a dedicated power pin for the analog portion of the chip and should be connected to a low-noise 3.3V supply.
0	1	V_{DDA_ADC}	**Analog Power**—This pin is a dedicated power pin for the analog portion of the ADC module and should be connected to a low-noise 3.3V supply.
4	5	V_{DDIO}	**Power In/Out**—These pins provide power to the I/O structures of the chip, and are generally connected to a 3.3V supply.

Note: Analog pins must be connected to a power source.

Table 2-3. Grounds

No. of Pins		Signal Name	Signal Description
826	827		
3	3	V_{SS}	**GND**—These pins provide grounding for the internal structures of the chip and should all be attached to V_{SS}.
1	1	V_{SSA}	**Analog Ground**—This pin supplies an analog ground.
—	1	V_{SSA_ADC}	**Analog Ground**—This pin is a dedicated ground pin for the analog portion of the ADC module.
4	5	V_{SSIO}	**GND In/Out**—These pins provide grounding for the I/O ring on the chip. All should be attached to V_{SS}.
1	1	**TCS**	**TCS**—This pin is reserved for factory use. It must be tied to V_{SS} for normal use. In block diagrams, this pin is considered an additional V_{SS}. Signal Type is Input/Output (Schmitt).

Table 2-4. Other Supply Port

No. of Pins		Signal Name	Signal Description
826	827		
—	1	V_{PP}	**V_{PP}—**This pin should be left unconnected as an open circuit for normal functionality.

2.3 Clock and Phase Lock Loop Signals

Table 2-5. PLL and Clock

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**EXTAL**	Input	Input	**External Crystal Oscillator Input**—This input should be connected to a 4MHz external crystal or ceramic resonator. For additional information, please refer to **section 4.5.2.**
1	1	**XTAL**	Input	Chip-driven	**Crystal Oscillator Output**—This output connects the internal crystal oscillator output to an external crystal. If an external clock source other than a crystal oscillator is used, XTAL must be used as the input and EXTAL connected to $V_{DDA}/2$. For additional information, please refer to **section 4.5.1.**
		(CLOCKIN)	Input	Input	**External Clock Input**—This input should be asserted when using an external clock or ceramic resonator.
1	1	**CLKO**	Output	Chip-driven	**Clock Output**—This pin outputs a buffered clock signal. By programming the CLKO Select Register (CLKOSR), the user can select between outputting a version of the signal applied to XTAL and a version of the device master clock at the output of the PLL. The clock frequency on this pin can be disabled by programming the CLKO Select Register (CLKOSR).

2.4 Address, Data, and Bus Control Signals

Table 2-6. Address Bus Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
8	—	**A0–A7** **(GPIOE0-7)**	Output Input/ Output	Tri-stated	**Address Bus**—A0–A7 specify the address for external program or data memory accesses. **Port E GPIO—**These eight General Purpose I/O (GPIO) pins can be individulally programmed as input or output pins. After reset, the default state is Address Bus.
8	—	**A8-15** **(GPIOA8-15)**	Output Input/ Output	Tri-stated	**Addressd Bus**—A8-A15 specify the address for external program or data memory accesses. **Port A GPIO—**These eight General Purpose I/O (GPIO) pins can be individulally programmed as input or output pins. After reset, the default state is Address Bus.
—	16	**A0-15** **(GPIOA0-15)**	Output Input/ Output	Tri-stated	**Address Bus**—A0-A15 specify the address for external program or data memory accesses. **Port A GPIO**—These 16 General Purpose I/O (GPIO) pins can be individulally programmed as input or output pins. After reset, the default state is Address Bus.

Table 2-7. Data Bus Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
16	—	**D0–D15**	Input/ Output	Tri-stated	**Data Bus**—D0–D15 specify the data for external Program or Data memory accesses. D0–D15 are tri-stated when the external bus is inactive.
—	16	**D0–D15** **(GPIOG0-15)**	Input/ Output Input/ Output	Tri-stated	**Data Bus—**D0-D15 specify the address for external Program or Data memory accesses. **Port G GPIO—**These 16 General Purpose I/O (GPIO) pins can be individually programmed as input or output pins. After reset, the default state is Address Bus.

Table 2-8. Bus Control Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	—	**$\overline{\text{PS}}$**	Output	Tri-stated	**Program Memory Select**—$\overline{\text{PS}}$ is asserted low for external program memory access.
—	1	**$\overline{\text{PS}}$**	Output	Tri-stated	**Program Memory Select**—$\overline{\text{PS}}$ is asserted low for external program memory access. This pin can also be programmed as a programmable chip select.
1	—	**$\overline{\text{DS}}$**	Output	Tri-stated	**Data Memory Select**—$\overline{\text{DS}}$ is asserted low for external data memory access.
—	1	**$\overline{\text{DS}}$**	Output	Tri-stated	**Data Memory Select**—$\overline{\text{DS}}$ is asserted low for external Data memory access. This pin can also be programmed as a programmable chip select.
1	1	**$\overline{\text{WR}}$**	Output	Tri-stated	**Write Enable**—$\overline{\text{WR}}$ is asserted during external memory write cycles. When $\overline{\text{WR}}$ is asserted low, pins D0–D15 become outputs and the device puts data on the bus. When $\overline{\text{WR}}$ is deasserted high, the external data is latched inside the external device. When $\overline{\text{WR}}$ is asserted, it qualifies the A0–A15, $\overline{\text{PS}}$, and $\overline{\text{DS}}$ pins. $\overline{\text{WR}}$ can be connected directly to the $\overline{\text{WE}}$ pin of a Static RAM.
1	1	**$\overline{\text{RD}}$**	Output	Tri-stated	**Read Enable**—$\overline{\text{RD}}$ is asserted during external memory read cycles. When $\overline{\text{RD}}$ is asserted low, pins D0–D15 become inputs and an external device is enabled onto the device data bus. When $\overline{\text{RD}}$ is deasserted high, the external data is latched inside the device. When $\overline{\text{RD}}$ is asserted, it qualifies the A0–A15, $\overline{\text{PS}}$, and $\overline{\text{DS}}$ pins. $\overline{\text{RD}}$ can be connected directly to the OE pin of a Static RAM or ROM.

2.5 Quad Timer Module Signals

Table 2-9. Quad Timer Module Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
4	4	**TA0-3** **(GPIOF0-GPIOF3)**	Input/ Output Input/ Output	Input Input	**TA0–3**—Timer A Channels 0, 1, 2, and 3 **Port F GPIO**—These four General Purpose I/O (GPIO) pins can be individually programmed as input or output. After reset, the default state is Quad Timer.

2.6 JTAG/OnCE Port Signals

Table 2-10. JTAG/OnCE Port Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**TCK**	Input (Schmitt) (Input/ Output)	Input, pulled low internally	**Test Clock Input**—This input pin provides a gated clock to synchronize the test logic and shift serial data to the JTAG/OnCE port. The pin is connected internally to a pull-down resistor.
1	1	**TMS**	Input (Schmitt) (Input/ Output)	Input, pulled high internally	**Test Mode Select Input**—This input pin is used to sequence the JTAG TAP controller's state machine. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
1	1	**TDI**	Input (Schmitt) (Input/ Output)	Input, pulled high internally	**Test Data Input**—This input pin provides a serial input data stream to the JTAG/OnCE port. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
1	1	**TDO**	Output (Schmitt) (Input/ Output)	Tri-stated	**Test Data Output**—This tri-statable output pin provides a serial output data stream from the JTAG/OnCE port. It is driven in the shift-IR and shift-DR controller states, and changes on the falling edge of TCK.

Table 2-10. JTAG/OnCE Port Signals (Continued)

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**$\overline{\text{TRST}}$**	Input (Schmitt) (Input/ Ouput)	Input, pulled high internally	**Test Reset**—As an input, a low signal on this pin provides a reset signal to the JTAG TAP controller. To ensure complete hardware reset, $\overline{\text{TRST}}$ should be asserted whenever $\overline{\text{RESET}}$ is asserted. The only exception occurs in a debugging environment, since the OnCE/JTAG module is under the control of the debugger. In this case it is not necessary to assert $\overline{\text{TRST}}$ when asserting $\overline{\text{RESET}}$. Outside of a debugging environment $\overline{\text{TRST}}$ should be permanently asserted by grounding the signal, thus disabling the OnCE/JTAG module on the device.
1	1	**$\overline{\text{DE}}$**	Output	Output	**Debug Event**—$\overline{\text{DE}}$ provides a low pulse on recognized debug events.

2.7 Synchronous Serial Interface

Table 2-11. Dedicated General Purpose Input/Output (GPIO) Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
8	8	**GPIOB0–GPIOB7**	Input/ Output	Input	**Port B GPIO**—These eight dedicated General Purpose I/O (GPIO) pins can be individually programmed as input or output pins. After reset, the default state is GPIO input.
8	8	**GPIOD0–GPIOD7**	Input/ Output	Input	**Port D GPIO**—These eight dedicated General Purpose I/O (GPIO) pins can be individually programmed as input or output pins. After reset, the default state is GPIO input.

Table 2-12. Synchronous Serial Interface (SSI) Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**SRD** **(GPIOC0)**	Input/ Output Input/ Output	Input Input	**SSI Receive Data (SRD)—**This input pin receives serial data and transfers the data to the SSI Receive Shift Receiver. **Port C GPIO**—This is a General Purpose I/O (GPIO) pin with the capability of being individually programmed as input or output. After reset, the default state is GPIO input.
1	1	**SRFS** **(GPIOC1)**	Input/ Output Input/ Output	Input Input	**SSI Serial Receive Frame Sync (SRFS)—**This bidirectional pin is used by the receive section of the SSI as frame sync I/O or flag I/O. The STFS can be used only by the receiver. It is used to synchronize data transfer and can be an input or an output. **Port C GPIO**—This is a General Purpose I/O (GPIO) pin with the capability of being individually programmed as input or output. After reset, the default state is GPIO input.
1	1	**SRCK** **(GPIOC2)**	Input/ Output Input/ Output	Input Input	**SSI Serial Receive Clock (SRCK)—**This bidirectional pin provides the serial bit rate clock for the Receive section of the SSI. The clock signal can be continuous or gated and can be used by both the transmitter and receiver in synchronous mode. **Port C GPIO—**This is a General Purpose I/O (GPIO) pin with the capability of being individually programmed as input or output. After reset, the default state is GPIO input.
1	1	**STD** **(GPIOC3)**	Output Input/ Output	Input Input	**SSI Transmit Data (STD)—**This output pin transmits serial data from the SSI Transmitter Shift Register. **Port C GPIO—**This is a General Purpose I/O (GPIO) pin with the capability of being individually programmed as input or output. After reset, the default state is GPIO input.

Table 2-12. Synchronous Serial Interface (SSI) Signals (Continued)

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**STFS**	Input	Input	**SSI Serial Transmit Frame Sync (STFS)—**This bidirectional pin is used by the Transmit section of the SSI as frame sync I/O or flag I/O. The STFS can be used by both the transmitter and receiver in synchronous mode. It is used to synchronize data transfer and can be an input or output pin.
		(GPIOC4)	Input/ Output	Input	**Port C GPIO—**This is a General Purpose I/O (GPIO) pin with the capability of being individually programmed as input or output. After reset, the default state is GPIO input.
1	1	**STCK**	Input/ Output	Input	**SSI Serial Transmit Clock (STCK)—**This bidirectional pin provides the serial bit rate clock for the transmit section of the SSI. The clock signal can be continuous or gated. It can be used by both the transmitter and receiver in synchronous mode.
		(GPIOC5)	Input/ Output	Input	**Port C GPIO—**This is a General Purpose I/O (GPIO) pin with the capability of being individually programmed as input or output. After reset, the default state is GPIO input.

2.8 Serial Peripheral Interface (SPI) Signals

Table 2-13. Serial Peripheral Interface (SPI1) Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**SCLK**	Input/ Output	Input	**SPI Serial Clock**—In master mode, this pin serves as an output, clocking slaved listeners. In slave mode, this pin serves as the data clock input.
		(GPIOF4)	Input/ Output	Input	**Port F GPIO**—This General Purpose I/O (GPIO) pin can be individually programmed as input or output. After reset, the default state is SCLK.
1	1	**MOSI**	Input/ Output	Input	**SPI Master Out/Slave In (MOSI)**—This serial data pin is an output from a master device and an input to a slave device. The master device places data on the MOSI line a half-cycle before the clock edge that the slave device uses to latch the data.
		(GPIOF5)	Input/ Output	Input	**Port F GPIO**—This General Purpose I/O (GPIO) pin can be individually programmed as input or output.
1	1	**MISO**	Input/ Output	Input	**SPI Master In/Slave Out (MISO)**—This serial data pin is an input to a master device and an output from a slave device. The MISO line of a slave device is placed in the high-impedance state if the slave device is not selected.
		(GPIOF6)	Input/ Output	Input	**Port F GPIO**—This General Purpose I/O (GPIO) pin can be individually programmed as input or output. After reset, the default state is MISO.
1	1	**$\overline{SS}$**	Inputt/ Output	Input	**SPI Slave Select**—In master mode, this pin is used to arbitrate multiple masters. In slave mode, this pin is used to select the slave.
		GPIOF7	Input/ Output	Input	**Port F GPIO**—This General Purpose I/O (GPIO) pin can be individually programmed as input or output. After reset, the default state is $\overline{SS}$.

2.9 Serial Communications Interface (SCI) or Serial Peripheral Interface (SPI0) Signals

Table 2-14. Serial Communications Interface (SCI0 & SCI1) Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**TXD0** **(SCLK0)**	Output Input/ Output	Input Input	**Transmit Data (TXD0)**—transmit data output **SPI Serial Clock—**In master mode, this pin serves as an output, clocking slaved listeners. In slave mode, this pin serves as the data clock input. After reset, the default state is SCI output.
1	1	**RXD0** **(MOSI0)**	Input Input/ Output	Input Input	**Receive Data (RXD0)**— receive data input. **SPI Master Out/Slave In**—This serial data pin is an output from a master device, and an input to a slave device. The master device places data on the MOSI line one half-cycle before the clock edge the slave device uses to latch the data. After reset, the default state is SCI input.
1	1	**TXD1** **(MISO0)**	Output Input/ Output	Input Input	**Transmit Data (TXD1)**—transmit data output. **SPI Master In/Slave Out—**This serial data pin is an input to a master device and an output from a slave device. The MISO line of a slave device is placed in the high-impedance state if the slave device is not selected. After reset, the default state is SCI output.
1	1	**RXD1** **($\overline{\text{SS0}}$)**	Input (Schmitt) Input	Input Input	**Receive Data (RXD1)**— receive data input **SPI Slave Select**—In master mode, this pin is used to arbitrate multiple masters. In slave mode, this pin is used to select the slave. After reset, the default state is SCI input.

2.10 Serial Communications Interface (SCI) or General Purpose Input/Output (GPIO) Signals

(56F827 only)

Table 2-15. Serial Communications Interface (SCI2) Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
—	1	**TXD2** **(GPIOC6)**	Output Input/ Output	Input Input	**Transmit Data (TXD2)**—transmit data output **Port C GPIO**—This General Purpose I/O (GPIO) pin can be individually programmed as input or output. After reset, the default state is GPIO output.
—	1	**RXD2** **(GPIOC7)**	Input/ Output Input/ Output	Input Input	**Receive Data (RXD2)**— receive data input **Port C GPIO**—This General Purpose I/O (GPIO) pin can be individually programmed as input or output. After reset, the default state is GPIO input.

2.11 Analog-to-Digital Converter (ADC) Signals (56F827 only)

Table 2-16. Analog-to-Digital Converter Signals

No. of Pins		Signal Name	Signal Type	State During Retset	Signal Description
826	827				
—	10	**ANA0-9**	Input	Input	**ANA0-9**—Analog inputs to ADC
—	1	**VREFP**	Input	Output	**ADC Reference**—The VREFP pin is connected to the positive side of the ADC range. This pin requires a 0.1µF ceramic capacitor to V_{SSA} and a start-up time of 25ms, prior to beginning conversions.
—	1	**VREFMID**	Input	Output	**ADC Reference**—The VREFMID pin is connected to the center of the ADC input range. This pin requires a 0.1µF ceramic capacitor to V_{SSA} and a start-up time of 25ms, prior to beginning conversions.
—	1	**VREFN**	Input	Output	**ADC Reference**—The VREFN pin is connected to the negative side of the ADC input range. This pin requires a 0.1µF ceramic capacitor to V_{SSA} and a start-up time of 25ms, prior to beginning conversions.
—	1	**VREFHI**	Input	Input	**ADC Reference**—These pins are Positive Reference for ADC and are generally connected to a 3.3V Analog (V_{DDA_ADC}) supply.

Table 2-16. Analog-to-Digital Converter Signals

No. of Pins		Signal Name	Signal Type	State During Retset	Signal Description
826	827				
—	1	**VREFLO**	Input	Input	**ADC Reference**—These pins are Negative Reference for ADC and are generally connected to a V_{SSA}.

2.12 Programmable Chip Select Signals (56F827 only)

Table 2-17. Programmable Chip Selects

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
—	6	**$\overline{\text{PCS}}$2-7**	Input/ Output	Tri-stated	**Programmable Chip Select** - PCS2-7 are asserted low for external peripheral chip select

2.13 Interrupt and Program Control Signals

Table 2-18. Interrupt and Program Control Signals

No. of Pins		Signal Name	Signal Type	State During Reset	Signal Description
826	827				
1	1	**$\overline{\text{IRQA}}$**	Input (Schmitt)	Input	**External Interrupt Request A**—The $\overline{\text{IRQA}}$ input is a synchronized external interrupt request indicating an external device is requesting service. It can be programmed to be level-sensitive or negative-edge-triggered. If level-sensitive triggering is selected, an external pull-up resistor is required for wired-OR operation. If the processor is in the Stop state and $\overline{\text{IRQA}}$ is asserted, the processor will exit the Stop state.
1	1	**$\overline{\text{IRQB}}$**	Input (Schmitt)	Input	**External Interrupt Request B**—The $\overline{\text{IRQB}}$ input is an external interrupt request indicating an external device is requesting service. It can be programmed to be level-sensitive or negative-edge-triggered. If level-sensitive triggering is selected, an external pull-up resistor is required for wired-OR operation.
1	1	**$\overline{\text{RESET}}$**	Input (Schmitt)	Input	**Reset**—This input is a direct hardware reset on the processor. When $\overline{\text{RESET}}$ is asserted low, the device is initialized and placed in the Reset state. A Schmitt trigger input is used for noise immunity. When the $\overline{\text{RESET}}$ pin is deasserted, the initial chip operating mode is latched from the external boot pin. The internal reset signal will be deasserted synchronous with the internal clocks, after a fixed number of internal clocks. To ensure complete hardware reset, $\overline{\text{RESET}}$ and $\overline{\text{TRST}}$ should be asserted together. The only exception occurs in a debugging environment when a hardware device reset is required and it is necessary not to reset the OnCE/JTAG module. In this case, assert $\overline{\text{RESET}}$, but do not assert $\overline{\text{TRST}}$.
1	1	**EXTBOOT**	Input (Schmitt)	Input	**External Boot**—This input is tied to V_{DD} to force device to boot from off-chip memory. Otherwise, it is tied to ground.

Chapter 3
Memory and Operating Modes

3.1 Introduction

Devices 56F826 and 56F827 are members of the 56F800 Family core-based hybrid controllers. Both combine the processing power and the functionality of a microcontroller with a flexible set of peripherals on a single chip. Because of their low cost, configuration flexibility and compact program code, both chips are well suited for many applications.

3.2 The 56F826/827 Memory Map Description

The 56F826/827 uses two independent memory spaces:

1. Data
2. Program

Both memory spaces use Harvard-style architecture. RAM and Flash memory are used for the on-chip data and on-chip program memories.

Table 3-1. Chip Memory Configurations

On-Chip Memory	56F826	56F827
Program Flash (PFLASH)	31.5K x 16	63K x 16
Data Flash (XFLASH)	2K x 16	4K x 16
Program RAM (PRAM)	512 x 16	1K x 16
Data RAM (XRAM)	4K x 16	
Program Boot Flash	2K x 16	—

On-chip memory sizes for each of the parts are summarized in **Table 3-1.** Both Program and Data Memories can be expanded off-chip. The Program Memory map is located in **Table 3-2.** The Operating Mode control bits (MA and MB) in the Operating Mode Register (OMR), coupled with the $\overline{\text{BOOTMAP}}$ bit in the SYS_CNTL register, control the Program Memory Map and select the vector address. For additional information about the SYS_CNTL register, please see **Figure 15-8** and **Section 15.8** in its entirety**.**

Table 3-2. Program Memory Map for 56F826/827

Begin/ End Address	Mode 0A		Mode 0B		Mode 3
	826	827	826	827	
0000 0003	BFlash 4	PFlash 31K	PFlash 31.5K	PFlash 31K	External Program Memory 64K
0004 7BFF	PFlash 31.5K-4				
7C00 7DFF		PRAM 1K		PRAM 1K	
7E00 7FFF	PRAM 0.5K		P. RAM 0.5K		
8000 87FF	BFlash 2K	PFlash 32K	External Program Memory 32K	External Program Memory 32K	
8800 FFFF	Reserved				

In all modes except Mode 3, the first four logical addresses of the 56F826 mirror the first four physical addresses of boot flash, which provides interrupt vectors for hardware reset and COP (Watchdog Timer) reset. This is largely an academic point in Mode 0B, as any reset or COP reset changes the memory map back to Mode 0A.

Note: Modes 0A and 0B are supported in this group of devices. For more information about other modes, please see **Section 3.7.2** and **Section 3.7.3.**

Table 3-3. Data Memory Map for 56F826/827

<table>
<tr><th rowspan="2">Begin/End Address</th><th colspan="2">EX = 0</th><th rowspan="2">EX = 1
826 and 827</th></tr>
<tr><th>826</th><th>827</th></tr>
<tr><td>0000

0FFF</td><td>XRAM
4K</td><td>XRAM
4K</td><td rowspan="10">External Data Memory
64K</td></tr>
<tr><td>1000

13FF</td><td>On-Chip Peripheral Registers</td><td>On-Chip Peripheral Registers</td></tr>
<tr><td>1400
17FF</td><td>Reserved</td><td rowspan="2">Reserved</td></tr>
<tr><td>1800
1FFF</td><td>XFlash
2K</td></tr>
<tr><td>2000
2FFF</td><td rowspan="3">External Data Memory
56K -128</td><td>XFlash
4K</td></tr>
<tr><td>3000
3FFF</td><td>Reserved</td></tr>
<tr><td>4000

FF7F</td><td>External Data Memory
48K -128</td></tr>
<tr><td>FF80
FFFF</td><td colspan="2">On-Chip Core Configuration Registers</td></tr>
</table>

When EX = 1, all 64K address space is external, unless I/O short addressing is used. If I/O short addressing is used, then the on-chip core configuration registers become available.

Note: Table 3-3 summarizes the Data Memory map. The External X memory control bit (EX) in the Operating Mode Register (OMR) controls the Data Memory map.

3.3 Data Memory

The 56F826 part has 4K words of on-chip Data RAM and 2K words of Data Flash. The 56F827 also has 4K words of on-chip Data RAM; and 4K words of Data Flash.

The 128-Data Memory addresses at the top of the memory map ($FF80 to $FFFF) are reserved for 56800 on-chip core configuration registers. When EX=0, the internal Data Memory map is enabled. When EX=1, the segment (hole) does not exist and may be accessed like all other external data memory.

Note: For 56800 core instructions performing two reads from the data memory in a single instruction, the *second* access using the R3 pointer always occurs to on-chip memory,

regardless how the OMR's EX bit is programmed. For example, the second read of the following code sequence accesses on-chip X data memory X:$0:

```
MOVE #$0000,R3
NOP
MOVE X:(R1)+,Y0 X:(R3)+,X0
```

Data Memory may be expanded off-chip for both the 56F826 and 56F827. When the OMR EX bit is programmed with 1, there are 65,536 addressable off-chip data memory locations.

When the EX bit is set, the on-chip core configuration registers may only be accessed using the I/O Short Addressing mode.

The External Data Memory bus access time is controlled by four bits of the Bus Control Register (BCR) located at X:$FFF9. This register is illustrated in **Figure .**

The External X bit (EX, bit three) of the OMR in the 56800 core, determines the mapping of the Data Memory, illustrated in **Table 3-3.** Setting the EX bit to 1 completely disables the on-chip data memory and enables a full 64K External Data Memory map.

Note: There are two exceptions to this rule.

1. The first exception is if a MOVE, TSTW, or BIT FIELD instruction is used with the I/O Short Addressing mode, the EX bit is ignored. This allows the on-chip core control registers to be accessed when the EX bit is set. Access time to external memory is always controlled by the BCR register or it wouldn't be possible to read/write to non-zero wait state external memory when EX = 0.
2. The second exception is for instructions performing two reads from the Data Memory in a single cycle. The EX bit is ignored during the second access using the R3 pointer.

A complete description of the Operating Mode Register (OMR) is provided in *DSP56800 Family Manual.*

3.3.1 Bus Control Register (BCR)

BCR—X:$FFF9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	DRV	0	WAIT STATE FIELD for EXTERNAL X-MEMORY				WAIT STATE Field for EXTERNAL P-MEMORY			
Write																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0

Figure 3-1. Bus Control Register (BCR)

See Programmer's Sheet on Appendix page C-19

3.3.1.1 Reserved—Bits 15–10

This bit field is reserved or not implemented. It is read/written as 0.

3.3.1.2 Drive (DRV)—Bit 9

The Drive Control bit is used to specify what occurs on the external memory port pins when no external access is performed—whether the pins remain driven or are tri-stated. The Drive (DRV) bit is cleared on hardware reset. Please see **Table 3-4** and **3-5.**

Table 3-4. Port A Operation with DRV Bit = 0

Mode	Pins		
	A0–A15	$\overline{PS}$, $\overline{DS}$, $\overline{RD}$, $\overline{WR}$	D0–D15
Normal Mode, External Access	Driven	Driven	Driven
Normal Mode, Internal Access	Tri-Stated	Tri-Stated	Tri-Stated
Stop Mode	Tri-Stated	Tri-Stated	Tri-Stated
Wait Mode	Tri-Stated	Tri-Stated	Tri-Stated
Reset Mode	Tri-Stated	Pulled High Internally	Tri-Stated

Table 3-5. Port A Operation with DRV = 1

Mode	Pins		
	A0–A15	$\overline{PS}$, $\overline{DS}$, $\overline{RD}$, $\overline{WR}$	D0–D15
Normal Mode, External Access	Driven	Driven	Driven
Normal Mode, Internal Access	Driven	Driven	Tri-Stated
Stop Mode	Driven	Driven	Tri-Stated
Wait Mode	Driven	Driven	Tri-Stated
Reset Mode	Tri-stated	Pulled high internally	Tri-stated

3.3.1.3 Reserved—Bit 8

This bit field is reserved or not implemented. It is read/written as 0.

3.3.1.4 Wait State Data Memory (WSX[3:0])—Bits 7–4

These bits provide programming of the wait states for External Data Memory. The bottom two bits, five and four, are hardcoded to zero. The WSX[3:0] bits are programmed as illustrated in **Table 3-6.**

Table 3-6. Programming WSX[3:0] Bits for Wait States

Bit String	Hex Value	Number of Wait States
0000	$0	0
0100	$4	4
1000	$8	8
1100	$C	12
All Others		Illegal

These bits allow programming of the wait states for external program memory. The bottom two bits, one and zero, are hardcoded to zero. The WSP[3:0] bits are programmed as illustrated in **Table 3-7.**

Table 3-7. Programming WSP [3:0] Bits for Wait States

Bit String	Hex Value	Number of Wait States
0000	$0	0
0100	$4	4
1000	$8	8
1100	$C	12
All Others		Illegal

3.3.2 Operating Mode Register (OMR)

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	NL	0	0	0	0	0	0	CC	0	SD	R	SA	EX	0	MB	MA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*

Figure 3-2. Operating Mode Register (OMR)

See Programmer's Sheet on Appendix page C-20

* MA and MB are latched from the EXTBOOT pin on reset.

3.3.2.1 Nested Looping (NL)—Bit 15

The Nested Looping (NL) bit displays the status of program DO looping and the Hardware Stack (HWS). When the NL bit is set, it indicates the program is currently in a nested DO loop. That is, two DO loops are active. When the NL bit is cleared, it indicates the program is currently not in a nested DO loop. There may be a single active DO loop or no DO loop active. This bit is necessary for saving and restoring the contents of the hardware stack. REP looping does not affect this bit.

It is important to never put the processor in the reserved combination as specified in **Table 3-8.** This can be avoided by ensuring the Loop Flag (LF) bit is never cleared when the NL bit is set. The NL bit is cleared on processor reset.

Table 3-8. Looping Status

NL (In OMR)	LF (in SR)	DO Loop Status
0	0	No DO loops active
0	1	Single DO loop active
1	0	Reserved
1	1	Two DO loops active

If both the NL and LF bits are set, that is: two DO loops are active and a DO instruction is executed, a Hardware Stack Overflow Interrupt occurs. The overflow occurs because space in the Hardware Stack was used, therefore not supporting a third DO loop.

The NL bit is also affected by any accesses to the HWS. Any move instruction writing to the HWS register should include copying the old contents of the LF bit into the NL bit. Before clearing the NL bit, its value should be moved into the LF bit (NL→LF, 0 → NL). For more detail on the interaction between NL, LF and HWS, please see **Section 5.1.7,** *Hardware Stack,* in the *DSP56800 Family Manual* and the DO and ENDDO instructions described in *Appendix A, Instruction Set Details,* in the *DSP56800 Family Manual*.

3.3.2.2 Reserved—Bits 14–9

This bit field is reserved or not implemented. It is a read/write field using 0.

3.3.2.3 Condition Codes (CC)—Bit 8

The Condition Code (CC) bit is set if the code is generated using a 36-bit result from the Multiplier/Accumulator (MAC) array, or a 32-bit result. When the CC bit is set, the C, N, V, and Z condition codes are generated based on bit 31of the Data Arithmetic Logic Unit (Data ALU) result. When cleared, the C, N, V, and Z condition codes are generated based on bit 35 of the data ALU result. The generation of the L, E, and U condition codes is not affected by the CC bit. The CC bit is cleared by the processor reset.

Note: The unsigned condition tests for branching or jumping (HI, HS, LO, or LS) can be used only when the condition codes are generated with the CC bit set. Otherwise, the chip does not generate the unsigned conditions correctly.

For more detail on the effects of the CC bit on condition codes generated by the data ALU operators, see **Section 3.6, Condition Code Generation**, in the *DSP56800 Family Manual*.

3.3.2.4 Reserved—Bit 7

This bit field is reserved or not implemented. It is a read/write field using 0.

3.3.2.5 Stop Delay (SD)—Bit 6

The Stop Delay (SD) bit selects the delay required by the device to exit the Stop mode. When set, the processor quickly exits from Stop mode. When the SD bit is cleared, the processor exits slowly from Stop mode. The SD bit is cleared by processor reset.

3.3.2.6 Rounding (R)—Bit 5

The Rounding (R) bit selects between two's-complement rounding and convergent rounding. When the R bit is set, two's-complement rounding is used. Rounding is always up. Cleared, the R bit uses convergent rounding. The R bit is cleared by processor reset.

3.3.2.7 Saturation (SA)—Bit 4

The Saturation (SA) bit enables automatic saturation on 32-bit arithmetic results, providing a user-enabled Saturation mode for those algorithms not able to exploit the four extra bits of A and B accumulators. When the SA bit is set the accumulators behave as output as if they were only 32 bits wide. The SA bit is cleared by processor reset.

When the SA bit is set automatic saturation occurs at the output of the MAC unit for basic arithmetic operations. Those basic arithmetic operations are of course, multiplication, addition, and so on. The saturation is performed by a special saturation circuit inside the MAC unit.

The saturation logic operates by checking three bits of the 36-bit result out of the MAC unit—exp[3], exp[0], and msp[15]. (The MSB and LSB of the extension register, EXP[3], EXP[0], and the MSB of the AI register AI[15],

When the SA bit is set, three bits determine if saturation is performed on the MAC unit's output and whether to saturate to the maximum positive or negative value, as illustrated in **Table 3-9.**

Table 3-9. MAC Unit Outputs With Saturation Mode Enabled (SA=1)

EXP[3]	EXP[0]	AI[15]	Result Stored in Accumulator
0	0	0	(Unchanged)
0	0	1	$0 7FFF FFFF
0	1	0	$0 7FFF FFFF
0	1	1	$0 7FFF FFFF
1	0	0	$F 8000 0000
1	0	1	$F 8000 0000
1	1	0	$F 8000 0000
1	1	1	(Unchanged)

Note: The Saturation mode is always *disabled* during the execution of the following instructions: ASLL, ASRR, LSLL, LSRR, ASRAC, LSRAC, IMPY16, MPYSU, MACSU, OR, EOR, NOT, LSL, LSR, ROL, AND, ROR. For these instructions, no saturation is performed at the output of the MAC unit.

3.3.2.8 External X Memory (EX)—Bit 3

Setting the External X (EX) memory bit forces all primary data memory accesses to be external. When the EX bit is set, all accesses to X memory on the X Address Bus One (XAB1) and Core Global Data Bus (CGDB) or Peripheral Global Data Bus (PGDB) are forced to be external except using the I/O Short Addressing mode. In this case, the EX bit is ignored and the access is performed to the on-chip location. When the EX bit is cleared, internal X memory can be accessed with all addressing modes.

The EX bit is cleared by processor reset. The EX bit is ignored by the second read of a dual-read instruction (using the X Address Bus Two (XAB2) and X Data Bus Two (XDB2)). For instructions with two parallel reads, the second read is always performed to internal on-chip memory.

Note: When the EX bit is set, only the upper 64-peripheral memory-mapped locations are accessible (X:$FFCO-x:$FFFF) with the I/O Short Addressing mode. The *lower* 64-memory-mapped locations (X:$FF80–$FFBF) are not accessible when the EX bit is set. Access to these addresses results in an entree to external memory. Operating Mode B (MB)—Bit 1.

This bit is latched from the EXTBOOT pin on reset. Please see **Section 3.7** in the *DSP56800 Family Manual*.

3.3.2.9 Reserved—Bit 2

This bit field is reserved or not implemented. It is a read/write field using 0.

3.3.2.10 Operating Mode B (MB)—Bit 1

This bit is latched from the EXTBOOT pin on reset. Please refer to **Section 3.7** in the *DSP56800 Family Manual.*

3.3.2.11 Operating Mode A (MA)—Bit 0

This bit is latched from the EXTBOOT pin on reset. Please refer to **Section 3.7** in the *DSP56800 Family Manual.*

3.4 Core Configuration Memory Map

Core configuration registers are part of the Data Memory map on the 56F826/827 part. These locations may be accessed with the same addressing modes used for ordinary data memory when the EX bit is cleared. However, when the EX bit is set, the addresses can only be accessed using the I/O Short Addressing mode. These registers are implemented as part of the 56800 core itself; therefore, they will be present on all family members based on the core. This is not necessarily true for the on-chip configuration registers discussed in the next section.

Table 3-10 illustrates the on-chip memory mapped core configuration registers.

Table 3-10. 56800 On-Chip Core Configuration Register Memory Map

Core Config	Accessibility	Register
X:$FFFF	Accessible Using I/O Short Addressing	OnCE PGDB Bus Transfer Register (OPGDBR)
		Reserved
		Reserved
		Reserved
X:$FFFB		Interrupt Priority Register (IPR)
		Reserved
X:$FFF9		Bus Control Register (BCR)
		Reserved
	Not Accessible At All When EX=1	Reserved

3.5 On-Chip Peripheral Memory Map

On-chip peripheral registers are part of the Data Memory map of the 56F826/827. These locations may be accessed with the same addressing modes used for ordinary data memory. However, they may not be entered by write and single read operations when the EX bit is set. (Since the EX bit is ignored on all second reads of a dual parallel move, these addresses are readable when EX = 1 as the second move.)

Table 3-11 and **Table 3-12** illustrate the on-chip memory mapped peripheral registers. The base address represents the starting address used for each peripheral's registers. Register memory locations are assigned in each peripheral chapter established on this base address, plus a given offset.

Not all peripherals are used on each device. For example, only the 56F827 uses the ADC peripheral. Also, the Data Memory map for the 56F826 starts from a different location of the mapped peripheral registers. So, 56F826 base addresses are different from those for the 56F827. It is important to read the address range and base address corresponding to the chip being used in **Table 3-13.** The following list of peripheral abbreviations appears in the same table.

Unless noted, each listed peripheral is available on both chips.

- System Integration Module (SIM)
- Program Flash Interface Unit number Two (PFIU2) (56F826 only)
- Data Flash Interface Unit (DFIU)
- Boot Flash Interface Unit (BFIU)
- Quad Timer A (TMRA)
- Time-of-Day (TOD)
- Synchronous Serial Interface (SSI)
- Clock Generation (CLKGEN)
- Interrupt Controller (ITCN)
- Computer Operating Properly (COP)
- Serial Peripheral Interface 0 (SPI0)
- Serial Peripheral Interface 1 (SPI1)
- Serial Communications Interface 0 (SCI0)
- Serial Communications Interface 1 (SCI1)
- General-Purpose I/O port A (GPIOA)
- General-Purpose I/O port B (GPIOB)

- General-Purpose I/O port C (GPIOC)
- General-Purpose I/O port D (GPIOD)
- General-Purpose I/O port E (GPIOE)
- General-Purpose I/O port F (GPIOF)
- Program Flash Interface Unit A Lower (PFIUAL) (56F827 only)
- Program Flash Interface Unit A Upper (PFIUAH) (56F827 only)
- Analog-to-Digital Converter (ADC) (56F827 only)
- Programmable Chip Select (PCS) (56F827only)

Table 3-11. 56F826 Data Memory Peripheral Address Map

Peripheral Name	Address Range	Base Address	Peripheral Register Address Tables
SIM	$1000 - $100F	SYS_BASE = $1000	Table 3-13
		Reserved	
PFIU	$1020 - $103F	PFIU_BASE = $1020	Table 3-14
		Reserved	
DFIU	$1060 - $107F	DFIU_BASE = $1060	Table 3-16
BFIU	$1080 - $109F	BFIU_BASE = $1080	Table 3-17
TMRA	$10A0 - $10BF	TMRA_BASE = $1200	Table 3-19
TOD	$10C0 - $10CF	TOD_BASE = $10C0	Table 3-21
		Reserved	
SSI	$10E0 - $10EF	SSI_BASE = $10E0	Table 3-22
CLKGEN	$10F0 - $10FF	CLKGEN_BASE = $10F0	Table 3-29
ITCN	$1100 - $111F	ITCN_BASE = $1100	Table 3-18
COP	$1120 - $112F	COP_BASE = $1120	Table 3-28
		Reserved	
SPI0	$1140 -$114F	SPI0_BASE = $1140	Table 3-26
SPI1	$1150 -$115F	SPI1 _BASE = $1150	Table 3-27
SCI	$1160 - $116F	SCI0_BASE = $1160	Table 3-23
SCI1	$1170 - $117F	SCI1_BASE = $1170	Table 3-24
		Reserved	
GPIOA	$11A0 - $11AF	GPIOA_BASE = $11A0	Table 3-30
GPIOB	$11B0 - $11BF	GPIOB_BASE = $11B0	Table 3-31
GPIOC	$11C0 - $11CF	GPIOC_BASE = $11C0	Table 3-32
GPIOD	$11D0 - $11DF	GPIOD_BASE = $11D0	Table 3-33
GPIOE	$11E0 - $11EF	GPIOE_BASE = $11E0	Table 3-34
GPIOF	$11F0 - $11FF	GPIOF_BASE = $11F0	Table 3-35
		Reserved	

Table 3-12. F56827 Data Memory Peripheral Address Map

Peripheral Name	Address Range	Base Address	Peripheral Register Address Tables
SIM	$1000 - $101F	SYS_BASE = $1000	Table 3-13
PFIU	$1020 - $103F	PFIU_BASE = $1020	Table 3-14
PFIU2	$1040 - $105F	PFIU2_BASE = $1040	Table 3-14
DFIU	$1060 - $107F	DFIU_BASE = $1060	Table 3-16
		Reserved	
TOD	$10C0 - $10CF	TOD_BASE = $10C0	Table 3-21
		Reserved	
SSI	$10E0 - $10EF	SSI_BASE = $10E0	Table 3-22
CLKGEN	$10F0 - $10FF	CLKGEN_BASE = $10F0	Table 3-29
ITCN	$1100 - $111F	ITCN_BASE = $1100	Table 3-19
COP	$1120 - $112F	COP_BASE = $1120	Table 3-28
		Reserved	
SPI0	$1140 -$114F	SPI0_BASE = $1140	Table 3-26
SPI1	$1150 -$115F	SPI1 _BASE = $1150	Table 3-27
SCI0	$1160 - $116F	SCI0_BASE = $1160	Table 3-23
SCI1	$1170 - $117F	SCI1_BASE = $1170	Table 3-24
SCI2	$1180 - $118F	SCI2_BASE = $1180	Table 3-25
PCS	$1190 - $119F	PCS_BASE = $1190	Table 3-38
GPIOA	$11A0 - $11AF	GPIOA_BASE = $11A0	Table 3-30
GPIOB	$11B0 - $11BF	GPIOB_BASE = $11B0	Table 3-31
GPIOC	$11C0 - $11CF	GPIOC_BASE = $11C0	Table 3-32
GPIOD	$11D0 - $11DF	GPIOD_BASE = $11D0	Table 3-33
		Reserved	
GPIOF	$11F0 - $11FF	GPIOF_BASE = $11F0	Table 3-35
		Reserved	
TMRA	$1200 - $123F	TMRA_BASE = $1200	Table 3-20
GPIOG	$1240 - $124F	GPIOG_BASE = $1240	Table 3-36
		Reserved	
ADC	$12C0 - $13FF	ADC_BASE = $12C0	Table 3-37
		Reserved	

Table 3-13. System Control Registers Address Map (SYS_BASE = $1000)

Register Abbreviation	Register Description	Address Offset	Base Address
SYS_CTRL	System Control Register	$0	SYS_BASE
SYS_STS	System Status Register	$1	SYS_BASE
MSH_ID	Most Significant Half of JTAG_ID	$6	SYS_BASE
LSH_ID	Least Significant Half of JTAG_ID	$7	SYS_BASE

Table 3-14. Program FLASH Interface Registers Address Map (PFIU_BASE = $1020)

Register Abbreviation	Register Description	Address Offset	Base Address
PFIU_CNTL	Program Flash Control Register	$0	PFIU_BASE
PFIU_PE	Program Flash Program Enable Register	$1	PFIU_BASE
PFIU_EE	Program Flash Erase Enable Register	$2	PFIU_BASE
PFIU_ADDR	Program Flash Address Register	$3	PFIU_BASE
PFIU_DATA	Program Flash Data Register	$4	PFIU_BASE
PFIU_IE	Program Flash Interrupt Enable Register	$5	PFIU_BASE
PFIU_IS	Program Flash Interrupt Source Register	$6	PFIU_BASE
PFIU_IP	Program Flash Interrupt Pending Register	$7	PFIU_BASE
PFIU_CKDIVISOR	Program Flash Clock Divisor Register	$8	PFIU_BASE
PFIU_TERASEL	Program Flash Terase Limit Register	$9	PFIU_BASE
PFIU_TMEL	Program Flash Time Limit Register	$A	PFIU_BASE
PFIU_TNVSL	Program Flash Tnvs Limit Register	$B	PFIU_BASE
PFIU_TPGSL	Program Flash Tpgs Limit Register	$C	PFIU_BASE
PFIU_TPROGL	Program Flash Tprog Limit Register	$D	PFIU_BASE
PFIU_TNVHL	Program Flash TNVH Limit Register	$E	PFIU_BASE
PFIU_TNVH1L	Program Flash TNVH1 Limit Register	$F	PFIU_BASE
PFIU_TRCVL	Program Flash TRCV Limit Register	$10	PFIU_BASE

Table 3-15. 56F827 Program Flash Interface Unit #2 Registers Address Map (PFIU2_BASE = $1040)

Register Abbreviation	Register Description	Address Offset	Base Address
PFIU_CNTL	Program Flash Control Register	$0	PFIU2_BASE
PFIU_PE	Program Flash Program Enable Register	$1	PFIU2_BASE
PFIU_EE	Program Flash Erase Enable Register	$2	PFIU2_BASE
PFIU_ADDR	Program Flash Address Register	$3	PFIU2_BASE
PFIU_DATA	Program Flash Data Register	$4	PFIU2_BASE
PFIU_IE	Program Flash Interrupt Enable Register	$5	PFIU2_BASE
PFIU_IS	Program Flash Interrupt Source Register	$6	PFIU2_BASE
PFIU_IP	Program Flash Interrupt Pending Register	$7	PFIU2_BASE
PFIU_CKDIVISOR	Program Flash Clock Divisor Register	$8	PFIU2_BASE
PFIU_TERASEL	Program Flash Terase Limit Register	$9	PFIU2_BASE
PFIU_TMEL	Program Flash Time Limit Register	$A	PFIU2_BASE
PFIU_TNVSL	Program Flash Tnvs Limit Register	$B	PFIU2_BASE
PFIU_TPGSL	Program Flash Tpgs Limit Register	$C	PFIU2_BASE
PFIU_TPROGL	Program Flash Tprog Limit Register	$D	PFIU2_BASE

Table 3-15. 56F827 Program Flash Interface Unit #2 Registers Address Map (PFIU2_BASE = $1040) (Continued)

Register Abbreviation	Register Description	Address Offset	Base Address
PFIU_TNVHL	Program Flash TNVH Limit Register	$E	PFIU2_BASE
PFIU_TNVH1L	Program Flash TNVH1 Limit Register	$F	PFIU2_BASE
PFIU_TRCVL	Program Flash TRCV Limit Register	$10	PFIU2_BASE

Table 3-16. Data Flash Interface Unit Registers Address Map (DFIU_BASE = $1060)

Register Abbreviation	Register Description	Address Offset	Base Address
DFIU_CNTL	Data Flash Control Register	$0	DFIU_BASE
DFIU_PE	Data Flash Program Enable Register	$1	DFIU_BASE
DFIU_EE	Data Flash Erase Enable Register	$2	DFIU_BASE
DFIU_ADDR	Data Flash Address Register	$3	DFIU_BASE
DFIU_DATA	Data Flash Data Register	$4	DFIU_BASE
DFIU_IE	Data Flash Interrupt Enable Register	$5	DFIU_BASE
DFIU_IS	Data Flash Interrupt Source Register	$6	DFIU_BASE
DFIU_IP	Data Flash Interrupt Pending Register	$7	DFIU_BASE
DFIU_CKDIVISOR	Data Flash Clock Divisor Register	$8	DFIU_BASE
DFIU_TERASEL	Data Flash Terase Limit Register	$9	DFIU_BASE
DFIU_TMEL	Data Flash TME Limit Register	$A	DFIU_BASE
DFIU_TNVSL	Data Flash TNVS Limit Register	$B	DFIU_BASE
DFIU_TPGSL	Data Flash TPGS Limit Register	$C	DFIU_BASE
DFIU_TPROGL	Data Flash TPROG Limit Register	$D	DFIU_BASE
DFIU_TNVHL	Data Flash TNVH Limit Register	$E	DFIU_BASE
DFIU_TNVHL1	Data Flash TNVH1 Limit Register	$F	DFIU_BASE
DFIU_TRCVL	Data Flash TRCV Limit Register	$10	DFIU_BASE

Table 3-17. 56F826 Boot Flash Interface Unit Registers Address Map (BFIU_BASE = $1080)

Register Abbreviation	Register Description	Address Offset	Base Address
BFIU_CNTL	Boot Flash Control Register	$0	BFIU_BASE
BFIU_PE	Boot Flash Program Enable Register	$1	BFIU_BASE
BFIU_EE	Boot Flash Erase Enable Register	$2	BFIU_BASE
BFIU_ADDR	Boot Flash Address Register	$3	BFIU_BASE
BFIU_DATA	Boot Flash Data Register	$4	BFIU_BASE
BFIU_IE	Boot Flash Interrupt Enable Register	$5	BFIU_BASE
BFIU_IS	Boot Flash Interrupt Source Register	$6	BFIU_BASE

Table 3-17. 56F826 Boot Flash Interface Unit Registers Address Map (BFIU_BASE = $1080)

Register Abbreviation	Register Description	Address Offset	Base Address
BFIU_IP	Boot Flash Interrupt Pending Register	$7	BFIU_BASE
BFIU_CKDIVISOR	Boot Flash Clock Divisor Register	$8	BFIU_BASE
BFIU_TERASEL	Boot Flash Terase Limit Register	$9	BFIU_BASE
BFIU_TMEL	Boot Flash TME Limit Register	$A	BFIU_BASE
BFIU_TNVSL	Boot Flash TNVS Limit Register	$B	BFIU_BASE
BFIU_TPGSL	Boot Flash TPGS Limit Register	$C	BFIU_BASE
BFIU_TPROGL	Boot Flash TPROG Limit Register	$D	BFIU_BASE
BFIU_TNVHL	Boot Flash TNVH Limit Register	$E	BFIU_BASE
BFIU_TNVHL1	Boot Flash TNVH1 Limit Register	$F	BFIU_BASE
BFIU_TRCVL	Boot Flash TRCV Limit Register	$10	BFIU_BASE

Table 3-18. Interrupt Controller Registers Address Map (ITCN_BASE = $1100)

Register Abbreviation	Register Description	Address Offset	Base Address
ITCN_GPR0	Group Priority Register 0	$0	ITCN_BASE
ITCN_GPR1	Group Priority Register 1	$1	ITCN_BASE
ITCN_GPR2	Group Priority Register 2	$2	ITCN_BASE
ITCN_GPR3	Group Priority Register 3	$3	ITCN_BASE
ITCN_GPR4	Group Priority Register 4	$4	ITCN_BASE
ITCN_GPR5	Group Priority Register 5	$5	ITCN_BASE
ITCN_GPR6	Group Priority Register 6	$6	ITCN_BASE
ITCN_GPR7	Group Priority Register 7	$7	ITCN_BASE
ITCN_GPR8	Group Priority Register 8	$8	ITCN_BASE
ITCN_GPR9	Group Priority Register 9	$9	ITCN_BASE
ITCN_GPR10	Group Priority Register 10	$A	ITCN_BASE
ITCN_GPR11	Group Priority Register 11	$B	ITCN_BASE
ITCN_GPR12	Group Priority Register 12	$C	ITCN_BASE
ITCN_GPR13	Group Priority Register 13	$D	ITCN_BASE
ITCN_GPR14	Group Priority Register 14	$E	ITCN_BASE
ITCN_GPR15	Group Priority Register 15	$F	ITCN_BASE

Table 3-19. 56F826 Quad Timer A Registers Address Map (TMRA_BASE = $10A0)

Register Abbreviation	Register Description	Address Offset	Base Address
TMRA0_CMP1	Compare Register 1	$0	TMRA_BASE
TMRA0_CMP2	Compare Register 2	$1	TMRA_BASE
TMRA0_CAP	Capture Register	$2	TMRA_BASE
TMRA0_LOAD	Load Register	$3	TMRA_BASE
TMRA0_HOLD	Hold Register	$4	TMRA_BASE
TMRA0_CNTR	Counter Register	$5	TMRA_BASE
TMRA0_CTRL	Control Register	$6	TMRA_BASE
TMRA0_SCR	Status and Control Register	$7	TMRA_BASE
TMRA1_CMP1	Compare Register 1	$8	TMRA_BASE
TMRA1_CMP2	Compare Register 2	$9	TMRA_BASE
TMRA1_CAP	Capture Register	$A	TMRA_BASE
TMRA1_LOAD	Load Register	$B	TMRA_BASE
TMRA1_HOLD	Hold Register	$C	TMRA_BASE
TMRA1_CNTR	Counter Register	$D	TMRA_BASE
TMRA1_CTRL	Control Register	$E	TMRA_BASE
TMRA1_SCR	Status and Control Register	$F	TMRA_BASE
TMRA2_CMP1	Compare Register 1	$10	TMRA_BASE
TMRA2_CMP2	Compare Register 2	$11	TMRA_BASE
TMRA2_CAP	Capture Register	$12	TMRA_BASE
TMRA2_LOAD	Load Register	$13	TMRA_BASE
TMRA2_HOLD	Hold Register	$14	TMRA_BASE
TMRA2_CNTR	Counter Register	$15	TMRA_BASE
TMRA2_CTRL	Control Register	$16	TMRA_BASE
TMRA2_SCR	Status and Control Register	$17	TMRA_BASE
TMRA0_CMP1	Compare Register 1	$18	TMRA_BASE
TMRA0_CMP2	Compare Register 2	$19	TMRA_BASE
TMRA0_CAP	Capture Register	$1A	TMRA_BASE
TMRA0_LOAD	Load Register	$1B	TMRA_BASE
TMRA0_HOLD	Hold Register	$1C	TMRA_BASE
TMRA0_CNTR	Counter Register	$1D	TMRA_BASE
TMRA0_CTRL	Control Register	$1E	TMRA_BASE
TMRA0_SCR	Status and Control Register	$1F	TMRA_BASE

Table 3-20. 56F827 Quad Timer A Registers Address Map
(TMRA_BASE = $1200)

Register Abbreviation	Register Description	Address Offset	Base Address
TMRA0_CMP1	Compare Register 1	$0	TMRA_BASE
TMRA0_CMP2	Compare Register 2	$1	TMRA_BASE
TMRA0_CAP	Capture Register	$2	TMRA_BASE
TMRA0_LOAD	Load Register	$3	TMRA_BASE
TMRA0_HOLD	Hold Register	$4	TMRA_BASE
TMRA0_CNTR	Counter Register	$5	TMRA_BASE
TMRA0_CTRL	Control Register	$6	TMRA_BASE
TMRA0_SCR	Status and Control Register	$7	TMRA_BASE
TMRA0_CMPLD1	Comparator Load Register 1	$8	TMRA_BASE
TMRA0_CMPLD2	Comparator Load Register 2	$9	TMRA_BASE
TMRA0_COMSCR	Comparator Status and Control Register 0	$A	TMRA_BASE
	Reserved		
TMRA1_CMP1	Compare Register 1	$10	TMRA_BASE
TMRA1_CMP2	Compare Register 2	$11	TMRA_BASE
TMRA1_CAP	Capture Register	$12	TMRA_BASE
TMRA1_LOAD	Load Register	$13	TMRA_BASE
TMRA1_HOLD	Hold Register	$14	TMRA_BASE
TMRA1_CNTR	Counter Register	$15	TMRA_BASE
TMRA1_CTRL	Control Register	$16	TMRA_BASE
TMRA1_SCR	Status and Control Register	$17	TMRA_BASE
TMRA1_CMPLD1	Comparator Load Register 1	$18	TMRA_BASE
TMRA1_CMPLD2	Comparator Load Register 2	$19	TMRA_BASE
TMRA1_COMSCR	Comparator Status and Control Register 1	$1A	TMRA_BASE
	Reserved		
TMRA2_CMP1	Compare Register 1	$20	TMRA_BASE
TMRA2_CMP2	Compare Register 2	$21	TMRA_BASE
TMRA2_CAP	Capture Register	$22	TMRA_BASE
TMRA2_LOAD	Load Register	$23	TMRA_BASE
TMRA2_HOLD	Hold Register	$24	TMRA_BASE
TRMA2_CNTR	Counter Register	$25	TMRA_BASE
TRMA2_CTRL	Control Register	$26	TMRA_BASE
TRMA2_SCR	Status and Control Register	$27	TMRA_BASE
TMRA2_CMPLD1	Comparator Load Register 1	$28	TMRA_BASE
TMRA2_CMPLD2	Comparator Load Register 2	$29	TMRA_BASE
TMRA2_COMSCR	Comparator Status and Control Register 2	$2A	TMRA_BASE
	Reserved		

Table 3-20. 56F827 Quad Timer A Registers Address Map (TMRA_BASE = $1200) (Continued)

Register Abbreviation	Register Description	Address Offset	Base Address
TRMA3_CMP1	Compare Register 1	$30	TMRA_BASE
TRMA3_CMP2	Compare Register 2	$31	TMRA_BASE
TRMA3_CAP	Capture Register	$32	TMRA_BASE
TRMA3_LOAD	Load Register	$33	TMRA_BASE
TMRA3_HOLD	Hold Register	$34	TMRA_BASE
TMRA3_CNTR	Counter Register	$35	TMRA_BASE
TMRA3_CTRL	Control Register	$36	TMRA_BASE
TRMA3_SCR	Status and Control Register	$37	TMRA_BASE
TMRA3_CMPLD1	Comparator Load Register 1	$38	TMRA_BASE
TMRA3_CMPLD2	Comparator Load Register 2	$39	TMRA_BASE
TMRA3_COMSCR	Comparator Status and Control Register 3	$3A	TMRA_BASE

Table 3-21. Time-of-Day Registers Address Map (TOD_BASE = $10C0)

Register Abbreviation	Register Description	Address Offset	Base Address
TODCS	Control/Status	$0	TOD_BASE
TODCSL	Clock Scaler Load	$1	TOD_BASE
TODSEC	Seconds	$2	TOD_BASE
TODSAL	Seconds Alarm	$3	TOD_BASE
TODMIN	Minutes	$4	TOD_BASE
TODMAL	Minutes Alarm	$5	TOD_BASE
TODHR	Hours	$6	TOD_BASE
TODHAL	Hours Alarm	$7	TOD_BASE
TODDAY	Days	$8	TOD_BASE
TODDAL	Days Alarm	$9	TOD_BASE

Table 3-22. SSI Registers Address Map (SSI_BASE = $10E0)

Register Abbreviation	Register Description	Address Offset	Base Address
STX	Transmit Register	$0	SSI_BASE
SRX	Receive Register	$1	SSI_BASE
SCSR	Control/Status Register 1	$2	SSI_BASE
SCR2	Control Register 2	$3	SSI_BASE
STXCR	Transmit Control Register	$4	SSI_BASE
SRXCR	Receive Control Register	$5	SSI_BASE
STSR	Time Slot Register	$6	SSI_BASE
SFCSR	FIFO Control/Status Register	$7	SSI_BASE
	Reserved		
SOR	Option Register	$9	SSI_BASE

Table 3-23. SCI0 Registers Address Map (SCI0_BASE = $1160)

Register Abbreviation	Register Description	Address Offset	Base Address
SCI0_SCIBR	Baud Rate Register	$0	SCI0_BASE
SCI0_SCICR	Control Register	$1	SC0_BASE
SCI0_SCISR	Status Register	$2	SCI0_BASE
SCI0_SCIDR	Data Register	$3	SCI0_BASE

Table 3-24. SCI1 Registers Address Map (SCI1_BASE = $1170)

Register Abbreviation	Register Description	Address Offset	Base Address
SCI1_SCIBR	Baud Rate Register	$0	SCI1_BASE
SCI1_SCICR	Control Register	$1	SCI1_BASE
SCI1_SCISR	Status Register	$2	SCI1_BASE
SCI1_SCIDR	Data Register	$3	SCI1_BASE

Table 3-25. 56F827 SCI2 Registers Address Map (SCI2_BASE = $1180)

Register Abbreviation	Register Description	Address Offset	Base Address
SCI2_SCIBR	Baud Rate Register	$0	SCI2_BASE
SCI2_SCICR	Control Register	$1	SCI2_BASE
SCI2_SCISR	Status Register	$2	SCI2_BASE
SCI2_SCIDR	Data Register	$3	SCI2_BASE

Table 3-26. SPI0 Registers Address Map (SPI0_BASE = $1140)

Register Abbreviation	Register Description	Address Offset	Base Address
SPI0_SPSCR	Status and Control Register	$0	SPI0_BASE
SPI0_SPDSR	Data Size Register	$1	SPI0_BASE
SPI0_SPDRR	Data Receive Register	$2	SPI0_BASE
SPI0_SPDTR	Data Transmit Register	$3	SPI0_BASE

Table 3-27. SPI1 Registers Address Map (SPI1_BASE = $1150)

Register Abbreviation	Register Description	Address Offset	Base Address
SPI1_SPSCR	Status and Control Register	$0	SPI1__BASE
SPI1_SPDSR	Data Size Register	$1	SPI1__BASE
SPI1_SPDRR	Data Receive Register	$2	SPI1__BASE
SPI1_SPDTR	Data Transmit Register	$3	SPI1__BASE

Table 3-28. COP Registers Address Map (COP_BASE = $1120)

Register Abbreviation	Register Description	Address Offset	Base Address
COPCTL	Control Register	$0	COP_BASE
COPTO	Timeout Register	$1	COP_BASE
COPSRV	Service Register	$2	COP_BASE

Table 3-29. Clock Generation Registers Address Map (CLKGEN_BASE = $10F0)

Register Abbreviation	Register Description	Address Offset	Base Address
PLLCR	Control Register	$0	CLKGEN_BASE
PLLDB	Divide-By Register	$1	CLKGEN_BASE
PLLSR	Status Register	$2	CLKGEN_BASE
	Reserved		
CLKOSR	CLKO Select Register	$4	CLKGEN_BASE

Table 3-30. GPIO Port A Registers Address Map (GPIOA_BASE = $11A0)

Register Abbreviation	Register Description	Address Offset	Base Address
GPIO_A_PUR	Pull-up Enable Register	$0	GPIOA_BASE
GPIO_A_DR	Data Register	$1	GPIOA_BASE
GPIO_A_DDR	Data Direction Register	$2	GPIOA_BASE
GPIO_A_PER	Peripheral Enable Register	$3	GPIOA_BASE
GPIO_A_IAR	Interrupt Assert Register	$4	GPIOA_BASE
GPIO_A_IENR	Interrupt Enable Register	$5	GPIOA_BASE
GPIO_A_IPOLR	Interrupt Polarity Register	$6	GPIOA_BASE
GPIO_A_IPR	Interrupt Pending Register	$7	GPIOA_BASE
GPIO_A_IESR	Interrupt Edge-Sensitive Register	$8	GPIOA_BASE

Table 3-31. GPIO Port B Registers Address Map (GPIOB_BASE = $11B0)

Register Abbreviation	Register Description	Address Offset	Base Address
GPIO_B_PUR	Pull-up Enable Register	$0	GPIOB_BASE
GPIO_B_DR	Data Register	$1	GPIOB_BASE
GPIO_B_DDR	Data Direction Register	$2	GPIOB_BASE
GPIO_B_PER	Peripheral Enable Register	$3	GPIOB_BASE
GPIO_B_IAR	Interrupt Assert Register	$4	GPIOB_BASE
GPIO_B_IENR	Interrupt Enable Register	$5	GPIOB_BASE
GPIO_B_IPOLR	Interrupt Polarity Register	$6	GPIOB_BASE
GPIO_B_IPR	Interrupt Pending Register	$7	GPIOB_BASE
GPIO_B_IESR	Interrupt Edge-Sensitive Register	$8	GPIOB_BASE

Table 3-32. GPIO Port C Registers Address Map (GPIOC_BASE = $11C0)

Register Abbreviation	Register Description	Address Offset	Base Address
GPIO_C_PUR	Pull-up Enable Register	$0	GPIOC_BASE
GPIO_C_DR	Data Register	$1	GPIOC_BASE
GPIO_C_DDR	Data Direction Register	$2	GPIOC_BASE
GPIO_C_PER	Peripheral Enable Register	$3	GPIOC_BASE
GPIO_C_IAR	Interrupt Assert Register	$4	GPIOC_BASE
GPIO_C_IENR	Interrupt Enable Register	$5	GPIOC_BASE
GPIO_C_IPOLR	Interrupt Polarity Register	$6	GPIOC_BASE
GPIO_C_IPR	Interrupt Pending Register	$7	GPIOC_BASE
GPIO_C_IESR	Interrupt Edge-Sensitive Register	$8	GPIOC_BASE

Table 3-33. GPIO Port D Registers Address Map (GPIOD_BASE = $11D0)

Register Abbreviation	Register Description	Address Offset	Base Address
GPIO_D_PUR	Pull-up Enable Register	$0	GPIOD_BASE
GPIO_D_DR	Data Register	$1	GPIOD_BASE
GPIO_D_DDR	Data Direction Register	$2	GPIOD_BASE
GPIO_D_PER	Peripheral Enable Register	$3	GPIOD_BASE
GPIO_D_IAR	Interrupt Assert Register	$4	GPIOD_BASE
GPIO_D_IENR	Interrupt Enable Register	$5	GPIOD_BASE
GPIO_D_IPOLR	Interrupt Polarity Register	$6	GPIOD_BASE
GPIO_D_IPR	Interrupt Pending Register	$7	GPIOD_BASE
GPIO_D_IESR	Interrupt Edge-Sensitive Register	$8	GPIOD_BASE

Table 3-34. 56F826 GPIO Port E Registers Address Map (GPIOE_BASE = $11E0)

Register Abbreviation	Register Description	Address Offset	Base Address
GPIO_E_PUR	Pull-up Enable Register	$0	GPIOE_BASE
GPIO_E_DR	Data Register	$1	GPIOE_BASE
GPIO_E_DDR	Data Direction Register	$2	GPIOE_BASE
GPIO_E_PER	Peripheral Enable Register	$3	GPIOE_BASE
GPIO_E_IAR	Interrupt Assert Register	$4	GPIOE_BASE
GPIO_E_IENR	Interrupt Enable Register	$5	GPIOE_BASE
GPIO_E_IPOLR	Interrupt Polarity Register	$6	GPIOE_BASE
GPIO_E_IPR	Interrupt Pending Register	$7	GPIOE_BASE
GPIO_E_IESR	Interrupt Edge-Sensitive Register	$8	GPIOE_BASE

Table 3-35. 56F826 GPIO Port F Registers Address Map (GPIOF_BASE = $11F0)

Register Abbreviation	Register Description	Address Offset	Base Address
GPIO_F_PUR	Pull-up Enable Register	$0	GPIOF_BASE
GPIO_F_DR	Data Register	$1	GPIOF_BASE
GPIO_F_DDR	Data Direction Register	$2	GPIOF_BASE
GPIO_F_PER	Peripheral Enable Register	$3	GPIOF_BASE
GPIO_F_IAR	Interrupt Assert Register	$4	GPIOF_BASE
GPIO_F_IENR	Interrupt Enable Register	$5	GPIOF_BASE
GPIO_F_IPOLR	Interrupt Polarity Register	$6	GPIOF_BASE
GPIO_F_IPR	Interrupt Pending Register	$7	GPIOF_BASE
GPIO_F_IESR	Interrupt Edge-Sensitive Register	$8	GPIOF_BASE

Table 3-36. 56F827 GPIO Port G Registers Address Map (GPIOG_BASE = $1240)

Register Abbreviation	Register Description	Address Offset	Base Address
GPIO_G_PUR	Pull-up Enable Register	$0	GPIOG_BASE
GPIO_G_DR	Data Register	$1	GPIOG_BASE
GPIO_G_DDR	Data Direction Register	$2	GPIOG_BASE
GPIO_G_PER	Peripheral Enable Register	$3	GPIOG_BASE
GPIO_G_IAR	Interrupt Assert Register	$4	GPIOG_BASE
GPIO_G_IENR	Interrupt Enable Register	$5	GPIOG_BASE
GPIO_G_IPOLR	Interrupt Polarity Register	$6	GPIOG_BASE
GPIO_G_IPR	Interrupt Pending Register	$7	GPIOG_BASE
GPIO_G_IESR	Interrupt Edge-Sensitive Register	$8	GPIOG_BASE

Table 3-37. 56F827 ADC Registers Address Map (ADC_BASE = $12C0)

Register Abbreviation	Register Description	Address Offset	Base Address
ADCR1	Control Register 1	$0	ADC_BASE
ADCR2	Control Register 2	$1	ADC_BASE
ADZCC1	Zero Crossing Control Register 1	$2	ADC_BASE
ADZCC2	Zero Crossing Control Register 2	$3	ADC_BASE
ADLST1	Channel List Registers 1	$4	ADC_BASE
ADLST2	Channel List Register 2	$5	ADC_BASE
ADLST3	Channel List Register 3	$6	ADC_BASE
ADLST4	Channel List Register 4	$7	ADC_BASE
ADLST5	Channel List Register 5	$8	ADC_BASE
ADSDIS	Sample Disable Register	$9	ADC_BASE
ADSTAT1	Status Register 1	$A	ADC_BASE
ADSTAT2	Status Register 2	$B	ADC_BASE
ADLLSTAT	Low Limit Status Register	$C	ADC_BASE
ADHLSTAT	High Limit Status Register	$D	ADC_BASE
ADZCSTAT	Zero Crossing Status Register	$E	ADC_BASE
ADRSLT0–9	ADC Result Registers 0–9	$F, $10, $11, $12, $13, $14, $15, $16, $17, $18	ADC_BASE
ADLLMT0–9	ADC Low Limit Registers 0–9	$19, $1A, $1B, $1C, $1D, $1E, $1F, $20, $21, $22	ADC_BASE
ADHLMT0–9	ADC High Limit Registers 0–9	$23, $24, $25, $26, $27, $28, $29, $2A, $2B, $2C	ADC_BASE
ADOFS0–9	ADC Offset Registers 0–9	$2D, $2E, $2F, $30, $31, $32, $33, $34, $35, $36	ADC_BASE
	Reserved		

Table 3-38. 56F827 PCS Registers Address Map (PCS_BASE = $1190)

Register Abbreviation	Register Description	Address Offset	Base Address
PCSBAR 0	Base Address Register 0	$0	PCS_BASE
PCSBAR 1	Base Address Register 1	$1	PCS_BASE
PCSBAR 2	Base Address Register 2	$2	PCS_BASE
PCSBAR 3	Base Address Register 3	$3	PCS_BASE
PCSBAR 4	Base Address Register 4	$4	PCS_BASE
PCSBAR 5	Base Address Register 5	$5	PCS_BASE
PCSBAR 6	Base Address Register 6	$6	PCS_BASE
PCSBAR 7	Base Address Register 7	$7	PCS_BASE
PCSOR 0	Option Control Register 0	$8	PCS_BASE
PCSOR 1	Option Control Register 1	$9	PCS_BASE
PCSOR 2	Option Control Register 2	$A	PCS_BASE
PCSOR 3	Option Control Register 3	$B	PCS_BASE
PCSOR 4	Option Control Register 4	$C	PCS_BASE
PCSOR 5	Option Control Register 5	$D	PCS_BASE
PCSOR 6	Option Control Register 6	$E	PCS_BASE
PCSOR 7	Option Control Register 7	$F	PCS_BASE

3.6 Program Memory

As illustrated in **Table 3-1,** the 56F826 has 31.5K words of on-chip Program Flash and 512 words of on-chip Program RAM and 2K words of Boot Flash. The 56F827 however, has 63K words of on-chip Program Flash Memory and 1K words on on-chip Program RAM.

Both 56F826 and 56F827 chip program memories may be expanded off-chip up to 64K. The external program bus access time is controlled by two of four bits on the Bus Control Register (BCR), located at X:$FFF9[1]. This register is provided in **Section .**

The on-chip Program Flash and RAM may hold a combination of interrupt vectors and program codes. Both may be modified by the application itself.

When mode three is selected, all 64K words of program memory are external.

1.All 56800 chips have two low order wait state bits in BCR hardcoded to zero.

3.7 Operating Modes

Both chips have two valid operating modes determining the memory maps for Program Memory. Operating modes can be selected either by applying the appropriate signal to the EXTBOOT pin during Reset, or by writing to the OMR and changing the MA and MB bits. The EXTBOOT pin is sampled as the chip leaves the reset state, and the initial operating mode of the chip is set accordingly.

Table 3-39. Program Memory Chip Operating Modes

State of EXTBOOT Upon Reset	MB	MA	Chip Operating Mode
0	0	0	Mode 0 Normal Operation
N/A	0	1	Not Supported
N/A	1	0	
1	1	1	Mode 3 External ROM

Chip operating modes can also be changed by writing to the operating mode bits MB and MA in the OMR. Changing operating modes does not reset the chip. Interrupts should be disabled immediately after an interrupt and before changing the OMR. This will prevent an interrupt from going to the wrong memory location. Also, one No-Operation (NOP) instruction should be included after changing the OMR to allow re-mapping to occur.

Note: When COP is reset, the MA and MB bits will revert to the values originally latched from the EXTBOOT pin in contradiction of $\overline{\text{RESET}}$, hardware reset. These original mode values determine the COP reset vector. EXTBOOT is *read-only* at the time of reset.

3.7.1 Single Chip Mode: Start-Up (Mode 0)

Mode zero is the single-chip mode internal Program RAM (PRAM) and PFLASH are enabled for reads and fetches. The 56F826 has two submodes:

1. Mode 0A Boot mode where all memory is internal
2. Mode 0B non-Boot mode where the first 32K of memory is internal and the second 32K is external

Note: Please refer to **Section 15.8** and **Figure 15-8** for additional information about SYS_CNTL.

If EXTBOOT is asserted low during reset, then Mode 0A boot is automatically entered when exiting the Reset mode.

Note: Locations zero through three in the program memory space are actually mapped to the first four locations in the Boot Flash.

Mode zero is useful to enter when coming out of reset for applications while executing primarily from internal Program Memory. The reset vector location in modes zero and three is in the program memory space at P:$0000, P:$0002 for the COP timer reset. For mode zero, this is in internal program memory. For mode three, it is an off-chip program memory.

3.7.2 Modes One and Two (Modes 1 and 2)

Modes one and two are not supported for these parts. They are used for ROM-based members of the 56800 Family.

3.7.3 External Mode (Mode 3)

Mode three is a development mode. Its entire 64K program memory space is external. No internal Program Memory may be accessed, except as a secondary read of Data RAM. The reset vector location in mode three is located in the external program memory space at location P:$0000 (P:$0002 for a COP timer reset).

3.8 Boot Flash Operation – 56F826 Only

Boot Flash is provided on the 56F826 only. It is to handle device initialization in the event the Program Flash becomes corrupted for any reason. The hardware and COP reset vectors, $0000 and $0002, are mapped into the Boot Flash at locations $8000 and $8002. The entire Boot Flash is available at locations $8000 through $87FF. The Boot Flash would normally be programmed only once just before or after the device is mounted in the end application. The code in the Boot Flash is typically responsible for checking the contents of PFLASH to determine if they are correct, reloading the PFLASH from SPI, SCI, and so on, if necessary.

Table 3-40. Loading Program Words

Word Number	Description
1	N = Number of program words to be loaded (must be at least one word LESS than the size of the area to be loaded).
2	Starting Address for Load
3 -> N + 2	Code To Be Loaded

The code in the Boot Flash could contain an algorithm for checking the current contents of the PFLASH against a previously computed signature stored in the last entry of the PFLASH. If these entries agree, the PFLASH contents are assumed good, and the memory map can be switched, redirecting the program control to the PFLASH. However, if the entries do not agree, then the PFLASH is reloaded through one of the external ports. Control is then redirected to the code just loaded into memory.

The above mechanism applies only in the case where the device is being booted in mode zero. It does not apply when the device is booted in mode three, where all program memory is external.

The 56F826 chip contains 2K of Boot Flash. Here's an example algorithm for the Boot Flash:

Example 3-1. Example Boot Flash Algorithm

```
assume modulus1 = a prime number
assume max = maximum address of program flash

/* Calculate a signature for the current contents of the program flash */
/* (assume addresses are normalized to the pflash root */
or (i = 0; i<max; i++) hash_no = (64*hash_no + pflash[i]) % modulus1;

If (hash_no == pflash[max]) begin
reset memory map by writing to the system control register
JUMP to jumpAddr
end else begin /* need to reload contents of PFLASH */

LOAD CODE VIA SPI PORT
COMPUTE NEW SIGNATURE AS DATA IS LOADED VIA SPI
ERASE FLASH IF NECESSARY
PROGRAM FLASH WITH DATA FROM SPI
SET LAST LOCATION TO BE LOADED = HASHCODE
SET jumpAddr = Starting Address From SPI

end
reset memory map by writing to the system control register
jump to jumpAddr
```

Note: The previous algorithm assumes an unlimited amount of on-chip RAM. If utilizing limited RAM, it is necessary to download data in buckets. Further, the algorithm also needs to be extended to support multiple EEPROM types.

3.9 Executing Programs from XRAM

The 56F826/827 devices do not support execution of program from Data RAM (XRAM).

3.10 56800 Reset and Interrupt Vectors

The reset and interrupt vector maps specify the processor jumped address when it recognizes an interrupt or encounters a reset condition. The instruction located at this address must be a Jump Subroutine (JSR) instruction for an interrupt vector, or a Jump (JMP) instruction for a reset vector. The interrupt vector map for a given chip is specified by all possible interrupt sources on the 56800 core as well as from the peripherals. Interrupt Priority Level (IPL) is not specified for hardware reset or for COP reset because these conditions Reset the chip. Resetting takes precedence over all other interrupts. **Table 3-41** provides the reset and interrupt priority structure for the 56F826/827 chip, including on-chip peripherals. **Table 3-42** lists the reset and interrupt

vectors for the 56800 Family. A full description of interrupts is provided in the *DSP56800 Family Manual (DSP56800FM).*

Note: In operating modes Zero and Three, the hardware Reset vector is at $0000 and the COP Reset vector is at $0002.

Table 3-41. Reset and Interrupt Priority

Priority	Exception	IPR Bits[1]
Level 1 (Non-maskable)		
Highest	Hardware $\overline{\text{RESET}}$	—
	COP Timer Reset	—
	Illegal Instruction Trap	—
	Hardware Stack Overflow	—
	OnCE Trap	—
Lower	SWI	—
Level 0 (Maskable)		
Higher	$\overline{\text{IRQA}}$ (External interrupt)	2, 1, 0
	$\overline{\text{IRQB}}$ (External interrupt)	5, 4, 3
	Channel 6 Peripheral Interrupt	9
	Channel 5 Peripheral Interrupt	10
	Channel 4 Peripheral Interrupt	11
	Channel 3 Peripheral Interrupt	12
	Channel 2 Peripheral Interrupt	13
	Channel 1 Peripheral Interrupt	14
Lowest	Channel 0 Peripheral Interrupt	15

1. Please refer to Section 5.7

Table 3-42. Reset and Interrupt Starting Addresses

Reset and Interrupt Starting Addresses[1]	Interrupt Priority Level	Interrupt Source
$0000[2]	—	Hardware $\overline{\text{RESET}}$
$0002[2]	—	COP Timer Reset
$0004	—	Reserved
$0006	1	Illegal Instruction Trap
$0008		Software Interrupt (SWI)
$000A		Hardware Stack Overflow
$000C		OnCE Trap
$000E		Reserved

Table 3-42. Reset and Interrupt Starting Addresses

Reset and Interrupt Starting Addresses[1]	Interrupt Priority Level	Interrupt Source
$0010	0	$\overline{\text{IRQA}}$
$0012		$\overline{\text{IRQB}}$
$0014		Peripheral Interrupt Vectors[3]
$0016		
$0018		
• • • • •		
$0042		
$007C		
$007E		

1. These are 56F80x addresses, not IPBus addresses.
2. Reset vectors are aliased to the first two vectors located in Boot Flash.
   56F826 : $0000 = $8000
   $0002 = $8002
3. Please see Chapter 5, Interrupt Controller (ITCN), for specific peripheral interrupt memory map locations.

3.11 Memory Architecture

The multiple bus Harvard-style architecture of the core within this device allows certain dual, parallel moves. This greatly increases throughput on algorithms requiring a high bandwidth feed of data values. The simplified diagram on the following page shows chips' on-board address. Data buses help to illustrate the allowed and disallowed parallel data.

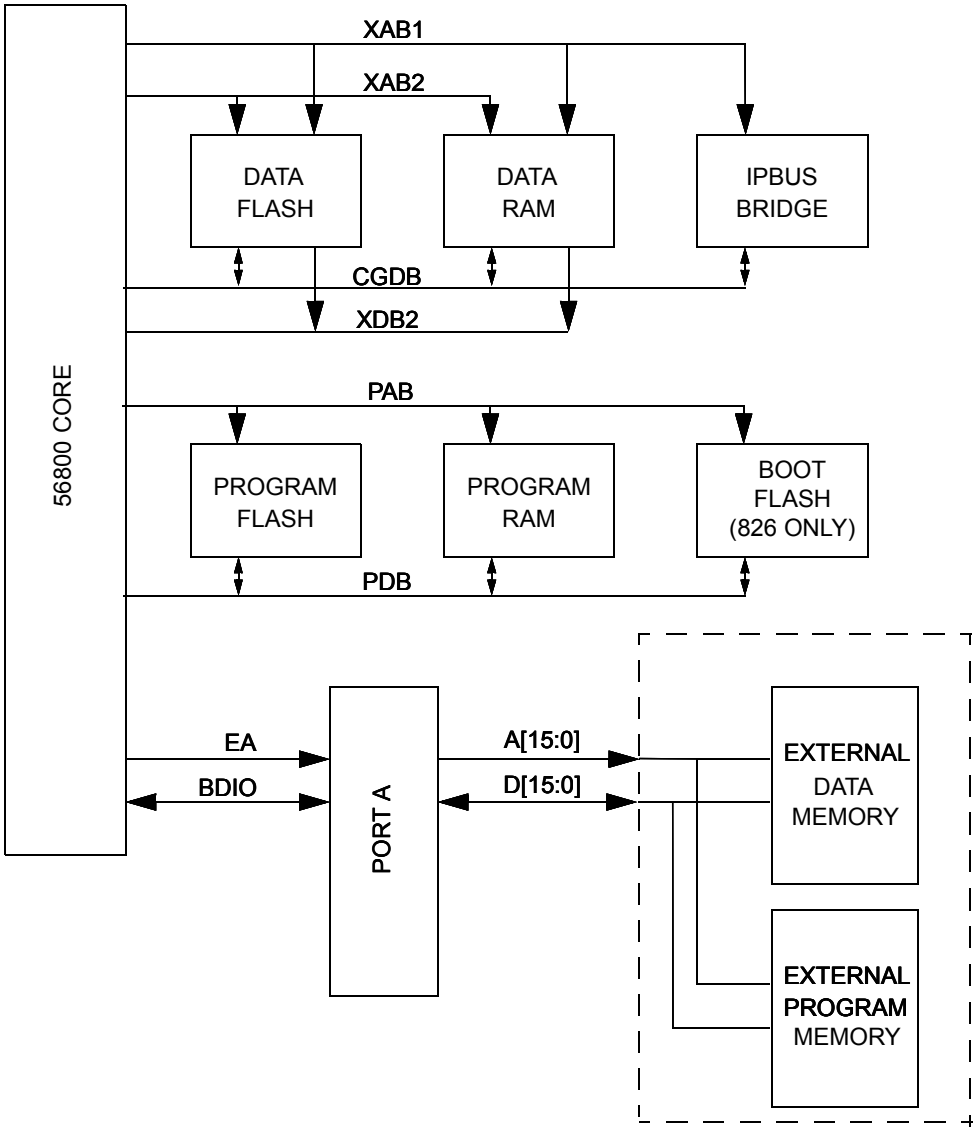


Figure 3-3. 56F80x On-Board Address and Data Buses

Data Memory is often noted as X memory. For instance, Data RAM is called XRAM. The following data space reads are possible:

- Dual read XRAM using CGDB and XRAM using XDB2
- Dual read RAM using CGDB and XFLASH using XDB2
- Dual read XRAM using XDB2 and XFLASH using CGDB
- Single read XRAM using CGDB
- Single read XFLASH using CGDB

Note: Dual read XFLASH using XDB2 and XFLASH using the Core Global Data Bus (CGDB) is not allowed.

Dual reads solely from XFLASH are not allowed. CGDB reads/writes are a result of XAB1 addressing. XDB2 reads are a result of XAB2 addressing. Writes are not allowed. XAB2 addressing is always the second field of a dual move operation, and can only be sourced by the R3 register.

The XRAM is a true dual-port RAM. The XFLASH is not available as a dual-ported memory.

Note: Due to the IPBus methodology used for busing from core to peripherals, the 56F826/827 do not connect the Peripheral Global Data Bus (PGDB) from the core to the peripherals. There are certain addressing modes of the move command anticipated to very likely cause a transfer, developing on the core's peripheral global data bus. Neither chip connects the PGDB from the core to the peripherals, so instructions utilizing this data bus such as MOVEP, are not useful. The MOVEP instruction only functions for registers internal to the core such as OnCE PGDB Bus Transfer Register (OPGDBR), IPR, and BCR.

Chapter 4
On-Chip Clock Synthesis (OCCS)

4.1 Introduction

The On-Chip Clock Synthesis (OCCS) module provides a flexible means of creating the internal clock signals to run the 56F826/827. This module generates three clock signals for use by the 56800 core and the chip peripherals. The input clock source can be an external 4MHz crystal (preferred), or this input frequency can be multiplied by the on-chip PLL to generate higher internal operating frequencies up to 80MHz. This section describes the module's architecture (programming model) and different low-power modes of operation.

In addition to a Phase Lock Loop (PLL) capable of multiplying the input frequency, as this module contains a prescaler divider. The prescaler divider is used to distribute lower frequency clocks to peripherals. This feature leads to lower power consumption on the chip. It also selects which clock is routed to the Clock Output (CLKO) pin on the 56F826/827 chip.

With the exception of the COP/Watchdog Timer and Time-of-Day (TOD), all peripherals on the 56F826/827 run off the IPBus clock frequency; the chip operating frequency is divided by two. The maximum frequency of operation is 80MHz correlating to a 40MHz IPBus clock frequency.

4.2 Features

The On-Chip Clock Synthesis (OCCS) module interfaces to the oscillator and PLL. The module has an on-chip prescaler. The OCCS module features:

- 2-bit prescaler
- 2-bit postscaler
- Ability to power-down the internal PLL
- Selectable PLL source clock
- Selectable system Clock (ZCLK) from three sources

The clock generation module provides the programming interface for both the PLL and on- and off-chip oscillators.

4.3 Block Diagram

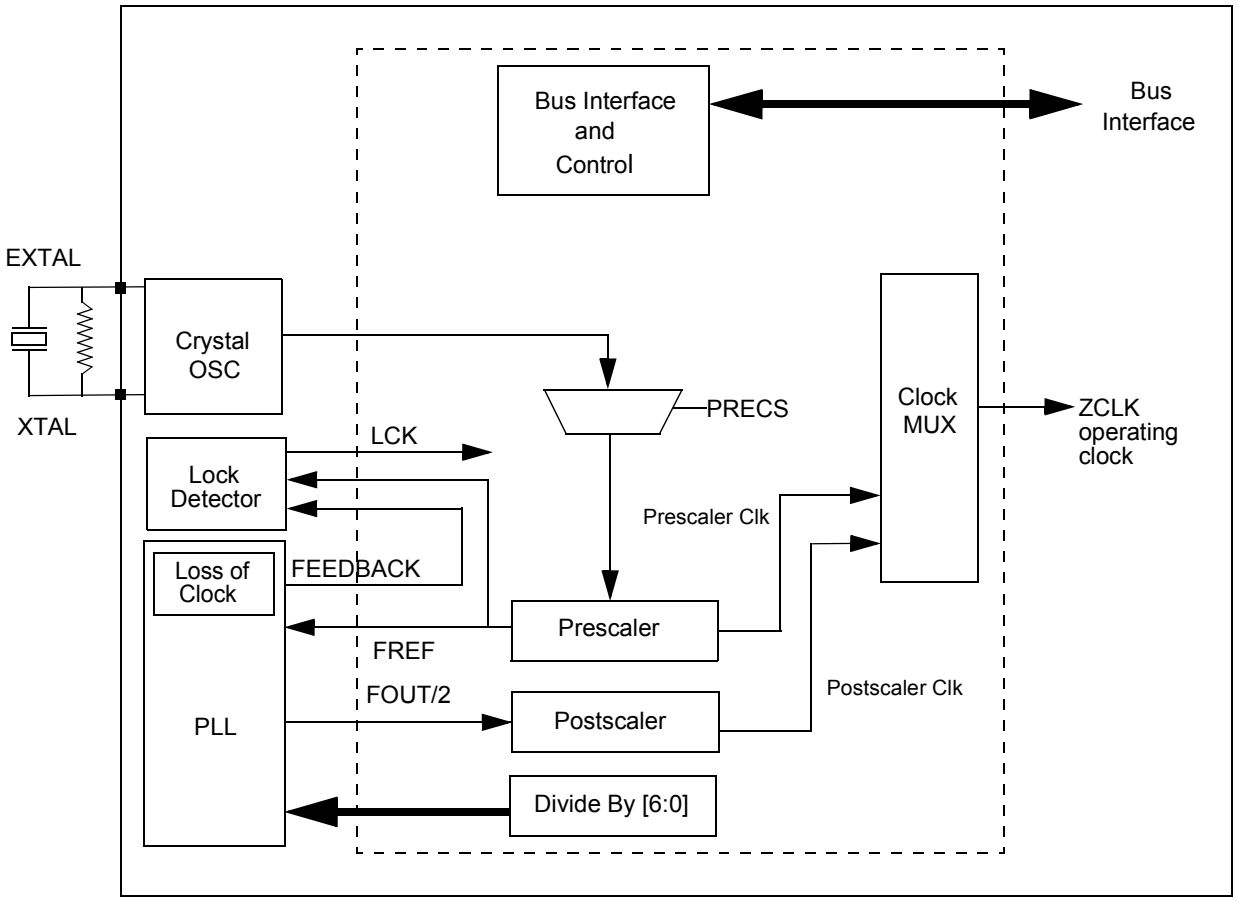


Figure 4-1. OCCS Block Diagram

4.4 Functional Description

This section describes the clocking methods available on the 56F826/827.

Figure 4-2 illustrates the types of acceptable reference clocks. These include, from left to right:

- Crystal oscillator, uses EXTAL and XTAL pins
- External clock source, uses XTAL pin

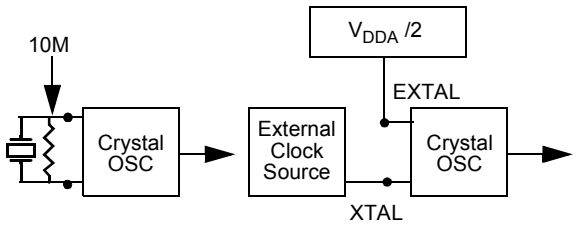


Figure 4-2. Reference Clock Sources

A block diagram of the On-Chip Clock Synthesis module is illustrated in **Figure 4-1.**

The Clock Multiplexer (MUX) selects the Oscillator (OSC) clock on power-up. A different clock source can be selected by writing to the PLL Control Register (PLLCR). Once a new clock source is selected, the new clock will be activated within four periods of the new clock after the clock selection request is re-clocked by the current IPBus clock.

Possible clock source choices:

- Crystal oscillator clock, derived from either a clock fed into the EXTAL pin, or an external crystal connected between the EXTAL and XTAL pins
- Crystal oscillator clock adjusted by the prescaler
- PLL clock

The PLL uses the crystal oscillator clock or divided version from the prescaler to derive its clock.

Frequencies going into and out of the PLL are controlled by the prescaler, postscaler, and the divide-by ratio within the PLL. For proper operation of the PLL, the Voltage Controlled Oscillator (VCO) must be kept within the PLL operational range of 80 - 240MHz. Output of the VCO is depicted as F_{OUT} in **Figure 4-1** or **Figure 4-11.** Prescaling the input frequency or adjusting the divide-by-ratio determines the frequency of the running VCO. The input frequency divided by the prescaler ratio, multiplied by the divide-by ratio is the frequency of the VCO running.

The PLL lock time is 10ms or less when coming from a powered-down state to a power-up state. It is recommended when changing the prescaler ratio or the divide-by ratio, powering down, or powering up, the PLL should be deselected as the clocking source. Only after a lock is achieved should the PLL be used as a valid clocking source.

4.4.1 Timing

A timing diagram used to change the clock source from the PLL clock to the prescaler clock is illustrated in **Figure 4-3.**

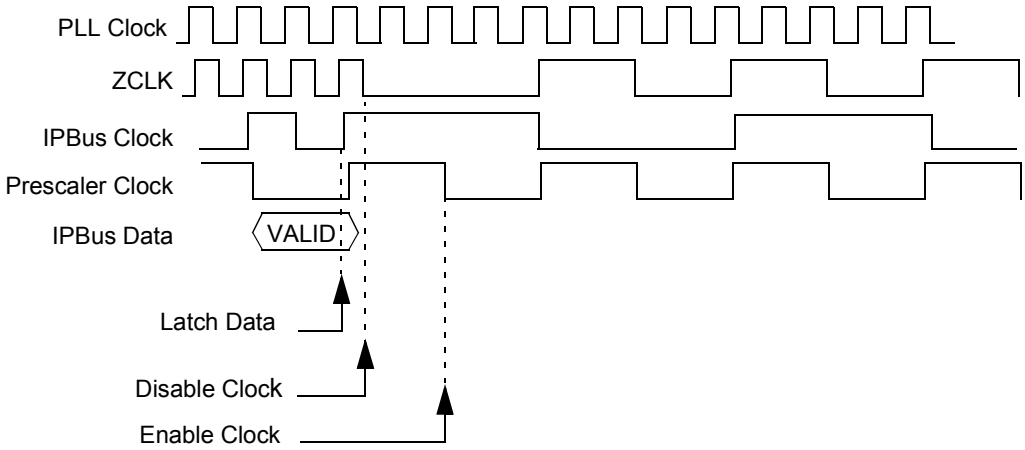


Figure 4-3. Changing Clock Sources

4.5 Pin Descriptions

4.5.1 Oscillator Inputs (XTAL, EXTAL)

The oscillator inputs can be used to connect an external crystal or to directly drive the chip with an external clock source, thus bypassing the internal oscillator circuit. Design considerations for the External Clock mode of operations are discussed in **Section 4.5.2.**

4.5.2 External Crystal Design Considerations

Either an external crystal oscillator or an external frequency source can be used to provide a reference clock to the 56F826/827.

4.5.3 Crystal Oscillator

The internal crystal oscillator circuit is designed to interface with a parallel-resonant crystal resonator whose frequency range is optimized for 4MHz. **Figure 4-4** illustrates a typical crystal 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.

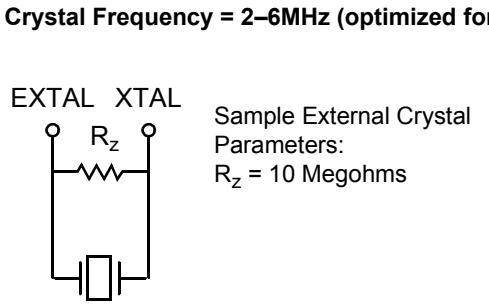


Figure 4-4. External Crystal Oscillator CircuitExternal Clock Source

The recommended method of connecting an external clock is provided in **Figure 4-5.** The external clock source is connected to XTAL while the EXTAL pin is held at $V_{DDA}/2$.

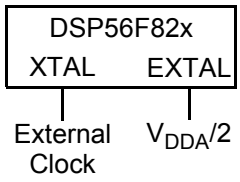


Figure 4-5. Connecting an External Clock Signal using XTAL

4.6 Register Definitions

Table 4-1. OCCS Memory Map

Device	Peripheral	Address
826/827	CLKGEN_BASE	$10F0

Address of a register is the sum of a base address and an address offset. The base address is designed at the Microcontroller Unit (MCU) level and the address offset is defined at the module level. For the memory location of these registers, please refer to **Table 3-29.**

Table 4-2. OCCS Register Summary

Address Offset	Address Acronym	Register Name	Access Type	Chapter Location
Base + $0	PLLCR	Control Register	Read/Write	**Section 4.6.1**
Base + $1	PLLDB	Divide-By Register	Read/Write	**Section 4.6.2**
Base + $2	PLLSR	Status Register	Read/Write	**Section 4.6.3**
		Reserved		
Base + $4	CLKOSR	CLKO Select Register	Read/Write	**Section 4.6.4**

Bit fields of each of the five registers are illustrated in **Figure 4-6.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	PLLCR	R	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMTRI	0	PLLPD	0	0	ZSRC	
		W																
$1	PLLDB	R	LORTP				PLLCOD		PLLCID		0	PLLDB						
		W																
$2	PLLSR	R	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLL PDN	0	0	ZSRC	
		W																
RESERVED																		
$4	CLKOSR	R	0	0	0	0	0	0	0	0	0	0	0	CLKOSEL				
		W																

R	0	Read as 0
W		Reserved

Figure 4-6. OCCS Register Map

4.6.1 PLL Control Register (PLLCR)

CLKGEN_BASE+$0	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	0	ZSRC[1:0]	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 4-7. PLL Control Register (PLLCR)

See Programmer's Sheet on Appendix page B- 109

In the event of clock loss, an optional interrupt can be generated to monitor the output of the on-chip oscillator circuit.

4.6.1.1 PLL Interrupt Enable 1 (PLLIE1)—Bits 15–14

An interrupt is generated when the PLL Lock (LCK1, fine lock) status bit changes in the PLL Status Register (PLLSR) and interrupts are enabled by the PLLIE. Options for enabling/disabling interrupts are:

- 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

4.6.1.2 PLL Interrupt Enable 0 (PLLIE0)—Bits 13–12

An interrupt is generated if the PLL Lock (LCK0) status bit in the PLLSR changes and interrupts are enabled by the PLLIE0. Options for enabling/disabling interrupts are:

- 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

4.6.1.3 Loss of Clock Interrupt Enable (LOCIE)—Bit 11

An optional interrupt can be generated if the oscillator circuit output clock is lost.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

4.6.1.4 Reserved—Bits 10–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

4.6.1.5 Lock Detector On (LCKON)—Bit 7

- 0 = Lock detector disabled
- 1 = Lock detector enabled

4.6.1.6 Charge Pump Tri-State (CHPMPTRI)—Bit 6

During normal chip operation the CHPMTRI bit should be set to a value of zero. In the event of loss of clock reference the CHPMTRI bit must be set to a value of one. A value of one isolates the charge pump from the loop filter allowing the PLL output to stabilize and to provide enough time to shut-down the chip. Activating this bit will render the PLL inoperable and should not be administered during standard operation of the chip.

Further, this bit is used to isolate the charge pump from the loop filter, allowing the filter voltage to be driven from an external source during bench test evaluations.

4.6.1.7 PLL Reserved—Bit 5

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

4.6.1.8 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[2:0] = 001b (zclock source = prescaler output) in order to prevent loss of clock to the core.

- 0 = PLL enabled
- 1 = PLL powered-down

4.6.1.9 Reserved—Bits 3–2

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

4.6.1.10 ZCLOCK Source (ZSRC)—Bits 1–0

The ZCLOCK source determines the clock source to the core. The ZCLOCK is also the source of the IPBus clock. ZSRC is automatically set to 001b during STOP_MODE, or if PLLPD is set in order to prevent loss of the clock to the core. For 56F826/827, ZSRC may have the following values.

- 01 = Prescaler output
- 10 = Postscaler output

4.6.2 PLL Divide-By Register (PLLDB)

CLKGEN_BASE+$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LORTP[3:0]				PLLCOD[1:0]		PLLCID[1:0]		0	PLLDB[6:0]						
Write																
Reset	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	1

Figure 4-8. PLL Divide-By Register (PLLDB)

See Programmer's Sheet on Appendix page B- 110

4.6.2.1 Loss of Reference Timer Period (LORTP[3:0])—Bits 15–12

These bits control the amount of time required for the loss of reference interrupt to be generated. This failure detection time is LORTP × 10 × PLL-clock-time-period.

4.6.2.2 PLL Clock-Out Divide (PLLCOD[1:0])—Bits 11–10

The PLL output clock can be divided down by a two-bit postscaler shown in **Figure 4-1** and **Figure 4-11.**

- 00 = Divide by one
- 01 = Divide by two
- 10 = Divide by four
- 11 = Divide by eight

4.6.2.3 PLL Clock-In Divide (PLLCID[1:0])—Bits 9–8

The PLL input clock, EXTAL_CLK, **Figure 4-11,** can be divided down by a two-bit prescaler. The output of the prescaler is a selectable clock source for the core as determined by the ZSRC in the PLLCR.

- 00 = Divide by one
- 01 = Divide by two
- 10 = Divide by four
- 11 = Divide by eight

4.6.2.4 Reserved—Bit 7

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

4.6.2.5 PLL Divide-By (PLLDB)—Bits 6–0

The output frequency of the PLL is primarily controlled by the PLLDB register. If *N* is the value written to the PLLDB register, the output frequency F_{OUT_PLL} is then given by

$$F_{OUT\text{-}PLL} = F_{IN\text{-}PLL} - (N + 1)$$

where

$F_{IN\text{-}PLL}$ is the PLL input frequency

Example:

F_{XTAL}= 4MHz and PLLCD is set to divide by 1

then

$$F_{IN\_PLL} = F_{XTAL}/1 = 4MHz$$

Let $N = 19$ then

$$F_{OUT\text{-}PLL} = F_{IN\text{-}PLL} \times (19 + 1)$$

$$= 4Mhz \times 20$$

$$= 80MHz$$

Before changing the PLLDB register it is recommended the core clock be first switched from the postscaler output to the prescaler output (ZSRC = 001b). Notice also, the lock detect circuit is reset when writing to the PLLDB register.

The value written to this register, plus one, is used by the PLL to directly multiply the input frequency and present it at its output. For example, if the input frequency is 8MHz and the PLLDB[6:0] register is set to 14, then the PLL output frequency is 120MHz.

Using the default bits shown in **Figure 4-7,** the value is (19 +1), or 20. For an 8MHz EXTAL_CLK, seen in **Figure 4-11,** this default value sets the output of the PLL, FOUT, to 160MHz, resulting in an FOUT/2 of 80MHz. Before changing the divide-by value, it is recommended the core clock be first switched to the prescaler clock.

Note: Upon writing to the PLLDB register, the lock detect circuit is reset.

4.6.3 PLL Status Register (PLLSR)

CLKGEN_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	PLLPDN	0	0	ZSRC[1:0]	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 4-9. PLL Status Register (PLLSR)

See Programmer's Sheet on Appendix page B- 111

The loss of lock interrupt is cleared by writing 1 to the respective LOLI bit in the PLLSR. The loss of clock interrupt is cleared by writing 1 to the LOCI bit in the LOCSR. A PLL interrupt is generated if any of the LOLI or LOCI bits are set and the respective interrupt enable is set in the PLLCR. The address of the ISR launched by any PLL interrupt is stored in the interrupt vector table in slot 62, address $007C. Please see **Table 5-2.**

4.6.3.1 PLL Loss of Lock Interrupt 1 (LOLI1)—Bit 15

- 0 = Loss of Lock One; interrupt pending
- 1 = No interrupt pending

This bit is cleared by writing 1 to LOLI1 bit.

4.6.3.2 PLL Loss of Lock Interrupt 0 (LOLI0)—Bit 14

- 0 = Loss of Lock Zero; interrupt pending
- 1 = No interrupt pending

This bit is cleared by writing 1 to LOLI0 bit.

4.6.3.3 Loss of Clock (LOCI)—Bit 13

- 0 = PLL reference clock normal; no interrupt pending
- 1 = Lost PLL reference clock; interrupt pending

This bit is cleared by writing 1 to the LOCK bit.

4.6.3.4 Reserved—Bits 12-7

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

4.6.3.5 Loss of Lock 1 (LCK1)—Bit 6

- 0 = PLL is unlocked (fine)
- 1 = PLL is locked (fine)

4.6.3.6 Loss of Lock 0 (LCK0)—Bit 5

- 0 = PLL is unlocked (coarse)
- 1 = PLL is locked (coarse)

4.6.3.7 PLL Power-Down (PLLPDN)—Bit 4

PLL power-down status is delayed by four IPBus clocks from the PLLPD bit in the PLLCR.

- 0 = PLL not powered down
- 1 = PLL powered down

4.6.3.8 Reserved—Bits 3–2

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

4.6.3.9 ZCLK Source (ZSRC[1:0])—Bits 1–0

ZSRC indicates the current ZCLK source. Since the synchronizing circuit switches the ZCLK source, ZSRC takes more than one IPBus clock to indicate the new selection.

- 00 = Synchronizing in progress
- 01 = Prescaler output
- 10 = Postscaler output
- 11 = Synchronizing in progress

4.6.4 CLKO Select Register (CLKOSR)

The CLKO Select Register can be used to multiplex out any one of the clocks generated inside the clock generation module. Please see **Figure 4-10.** The default value is ZCLK. The ZCLK is the processor clock and should be used for any external memory accesses requiring a clock. This path has been optimized to minimize any delay and clock duty cycle distortion. All other clocks possibly muxed out are for test purposes only.

CLKGEN_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	CLKOSEL				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-10. CLKO Select Register (CLKOSR)

See Programmer's Sheet on Appendix page B- 112

4.6.4.1 Reserved—Bits 15–5

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

4.6.4.2 CLKO Select (CLKOSEL)—Bits 4–0

Selects clock to be muxed out on the CLKO pin.

- 00000 = ZCLK - *default*
- 10000 = No clock

4.6.5 Clock Operation in the Power-Down Modes

The 56F826/827 operates in one of three power-down modes: Run, Wait, or Stop.

- Run—All clocks are active in the Run mode.
- Wait—All chip's clocks are active except those internal to the 56800 core itself in the Wait mode. All peripheral devices are still active and able to produce interrupt requests. When an interrupt is not masked off, assertion will cause the core to come out of its suspended state, and resume normal operation. Wait is typically used for power sensitive applications.
- Stop - This mode causes ZCLK and the IPBus_Clock to turn-off. Like the Wait mode, the Stop mode causes all internal clocks to the 56F800 core to shut down. Processing is suspended. The only possible recoveries from the Stop mode are a TOD alarm interrupt, TOD one second interrupt, an external interrupt IRQA/B or a power-on reset.

A typical application example is demonstrated in **Figure 4-11.** Usually an inexpensive 4MHz crystal is selected to provide a stable time reference for the system. Choose the PLL by selecting a prescaler value to be written to the PLLCID bit. A recommended prescaler value is one. The output frequency of the prescaler is the input frequency to the PLL. This output frequency to the PLL is multiplied by the (PLLDB + 1) value. The resulting frequency is depicted as F_{OUT} in **Figure 4-11.** Divide the frequency by two through a fixed divider at the output of the PLL prior to being presented to the postscaler divider. The postscaler divider further apportions the frequency by its user-defined value before presenting it as ZCLK through the sync mux. Setting the frequency using the programmable divider is discussed in

•

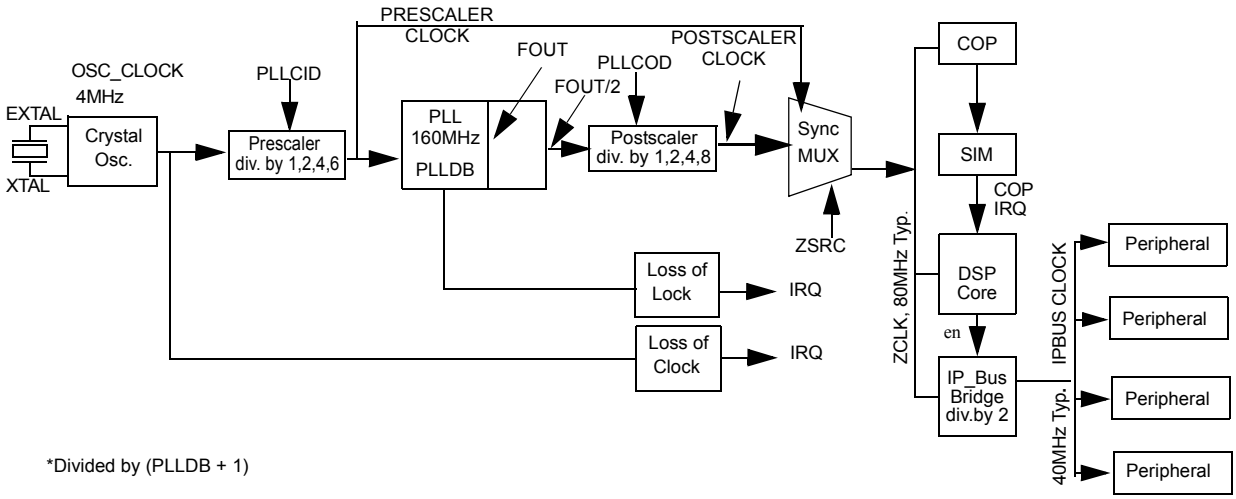


Figure 4-11. Relationship of IPBus Clock and ZCLK

Note: The maximum frequency of ZCLK operation is 80MHz correlating to a 40MHz IPBus clock frequency.

The following table summarizes the state of the on-chip clocks in typical operating frequencies during the various power-down modes.

Table 4-3. On-Chip Clock States

State	ZCLK	IPBus_CLK
Run	80MHz	40MHz
Wait	80MHz	40MHz
Stop	Off	Off

Note: All peripherals, except the COP/Watchdog Timer and TOD, run off the IPBus clock frequency. That frequency is the chip operating frequency divided by two. The maximum frequency of operation is 80MHz, correlates to a 40MHz IPBus clock frequency.

4.6.6 PLL Recommended Range of Operation

The PLL's Voltage Controlled Oscillator (VCO) has a characterized operating range extending from 80MHz to 240MHz. The output of the PLL, F_{OUT} in **Figure 4-11** is followed by a hard wired, divide-by two, circuit yielding a 40MHz to 120MHz clock at the input of the postscaler. The PLL is programmable via a divide-by n+1 register, capable of taking on values varying between 1 and 128. PLLDB bits determine this value, referenced in **Section 4.6.2.5.** For higher values of n, PLL lock time becomes an issue. It is recommended to avoid values of n resulting in the VCO frequency greater than 240MHz or less than 80MHz. **Figure 4-13** shows the

recommended range of available output frequencies from the PLL verses the input frequency. Generally, the lower the value of *n*, the quicker the PLL will be able to lock

.

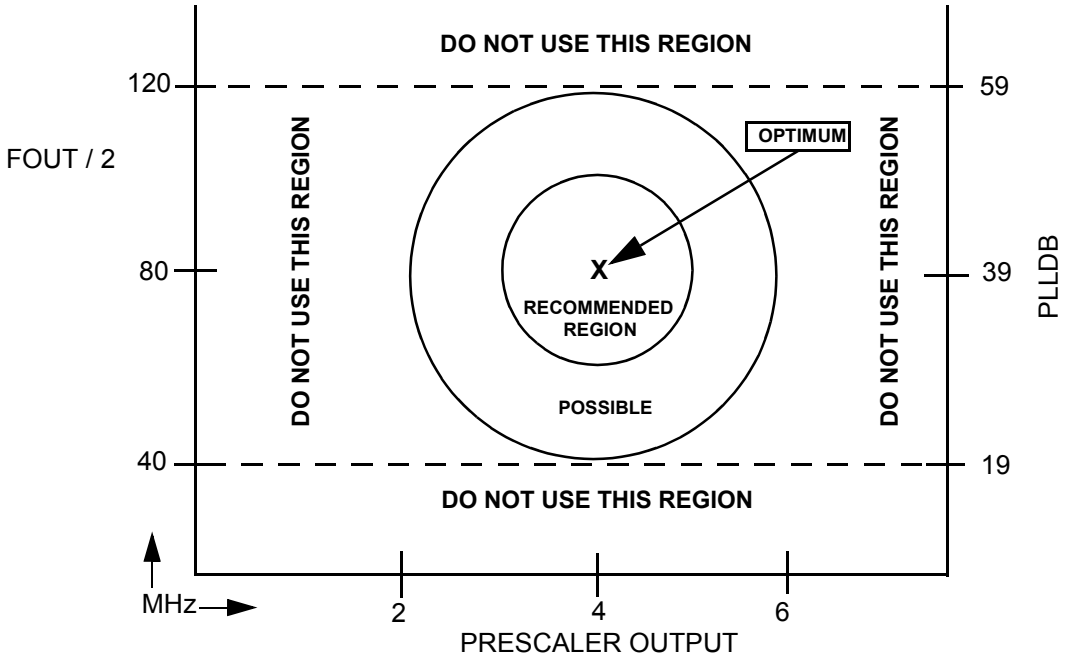


Figure 4-12. Recommended Design Regions of OCCS PLL Operation

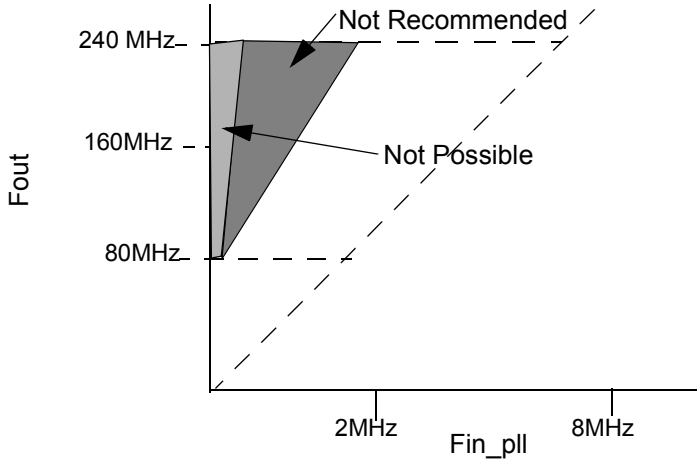


Figure 4-13. PLL Output Frequency vs. Input Frequency

Note: The value of *n* and the specific coordinates of the *Not Possible* and *Not Recommended* areas are yet to be determined.

4.7 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.

4.7.1 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 high) to a powered up condition (PLL_PDN low) the maximum lock time, with a divide-by count of 16 or less is 10ms.

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.

4.7.1.1 Parametric Influences on Reaction Time

Lock time is designed to be as short as possible while still providing the highest possible stability. However, the reaction time is not constant. Many factors directly and indirectly affect the lock time.

4.8 PLL Frequency Lock Detector Block

This digital block monitors the VCO output clock and sets the LCK[1:0] bits in the PLL Status Register (PLLSR) based on its frequency accuracy. The lock detector is enabled with the LCKON bit of the PLL Control Register (PLLCR), as well as the PLL_PDN bit of PLLCR. Once enabled, the detector starts two counters whose outputs are periodically compared. The input clocks to these counters are the VCO output clock divided by the value in the *n*-factor called feedback, and the crystal reference clock, shown as FREF in **Figure 4-1.** The period of the pulses being compared cover one whole period of each clock. This is due to the feedback clock not guaranteeing a 50 percentage duty cycle. This block was designed with the assumption the feedback clock transitions high on the rising edge of FREF.

Feedback and FREF 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 FREF there are the same number of FREF clocks as feedback clocks, the LCK1 bit is also set. The LCK bits stay set until:

- Clocks fail to match

- A new value is written to the PLLDB
- On reset caused by LCKON, PLL_PDN
- Chip level reset

When the circuit sets the LCK1, the two counters are reset and begin the count again. The lock detector is designed so if LCK1 is reset to zero, because the clocks did not match, LCK0 can stay high. This provides the processor the accuracy of the two clocks with respect to each other.

Chapter 5
Interrupt Controller (ITCN)

5.1 Introduction

The IPBus Interrupt Controller (ITCN) accepts Interrupt Request (IRQ) signals for Interrupt Priority bus-based peripherals, assigning defined levels to each IRQ. This is accomplished before selecting the highest pending IRQ for presentation to the 56F826/827 core. The core includes circuitry supporting seven levels of interrupts. The core awards priority to the interrupts associated with the highest level.

ITCN augments the 56F826/827 Interrupt Controller by expanding the maximum number of peripheral interrupts to 64 and adding the ability to assign an interrupt priority level to each of the 64-Interrupt Request (IRQ) sources. The assignment of a non-zero interrupt priority links the interrupt to one of the seven interrupt channels of the core's Interrupt Controller.

5.2 Interrupt Source

Each Interrupt Source has a fixed vector number, regardless of its assigned interrupt priority level. For any given level, the priority of the interrupts is determined by their respective vector number. The Interrupt Source with the highest vector has the highest priority within a given level.

5.3 Interrupt Control

The Interrupt Controller module does not mask, generate, or clear IRQs. The peripheral module issuing the IRQ only defines the enabling and clearing methods of a particular interrupt. The Interrupt Controller provides only the priority assignments and the interrupt vector generation.

5.4 Priority Level Register (PLR)

There are 64, three-bit Priority Level Registers (PLRs) specifying the Interrupt Priority Level assignment (0-7) for each Interrupt Request signal. Four PLRs are assembled together in each Group Priority Register (GPR). Each 16-bit GPR is read/write. Each PLR value has a default value on reset, but each value may be altered to suit the application's requirements. If a PLR value is set to zero, the effect is to disable that interrupt.

5.5 Interrupt Exceptions

The I1 and I0 bits in the 56F826/827 core Status Register (SR) determine the class of the permitted exceptions.

- To permit maskable exceptions, *I* bits should be set to 01.
- To disable the maskable exceptions, *I* bits should be set to 11.

The Interrupt Priority Register (IPR), also part of the 56800 core, is located at its X:$FFFB address. The IPR has an Interrupt Priority Level (IPL) bit assigned to each of the seven interrupt priority levels and generated by the Interrupt Controller. Each IPL bit enables interrupts associated with its Interrupt Priority Level.

5.6 Interrupt Enable

To enable a specific peripheral interrupt, an enable bit in the peripheral must be set, the PLR in the Interrupt Controller must be initialized to a non-zero Interrupt Priority Level, the corresponding IPL bit must also be set in the Processor Status register, and maskable interrupts must be enabled in the Status Register (SR). ITCN is designed for 64 interrupts; however, the 56F826/827 reserves the first 10-interrupt vectors for internally generated exceptions and the IRQ_A and IRQ_B external interrupts, bypassing the ITCN to have higher priority than all ITCN-generated interrupts.

5.7 Interrupt Priority Register (IPR)

The core's Interrupt Priority Register (IPR) is a read/write register memory-mapped to location X:$FFFB. The IPR enables/disables maskable interrupts via seven interrupt channel bits (CH0 through CH6) and six additional configuration bits for IRQA and IRQB.

Channel bits CH0 through CH6 enable/disable all ITCN-generated interrupts with IPLs of one through six. For example, writing 1 to bit CH5 enables all ITCN-generated interrupts with an IPL equal to 6. So the rule is CHn bit enables/disables interrupts with IPLs equal to n+1.

The linkage of peripheral interrupt to IPL level to channel bit allows interrupt nesting. For example, supposed all interrupt channels are enabled (CH0 through CH7 are 1, and I1:I0 in the SR is 116). An interrupt with the IPL of 6 can prevent the execution of all interrupts with lower IPLs by changing CH0 through CH4 to zero (thereby disabling all interrupts with lower priority) before resetting I1:I0 back to 11b to re-enable interrupts of higher priority.

Note: The embedded Software Development Kit (SDK) provides full support for interrupt nesting, plus content saving. For the details of how to use SDK's interrupt services, see the Interrupts Chapter of the Embedded SDK Programmer's Guide.

The $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$ interrupts bypass the ITCN, having higher priority than all ITCN-generated interrupts. (That is, all peripheral interrupts have lower priority than $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$).

While these interrupts have the highest priority among all maskable interrupts (those controlled by the I1:I0 bits in SR) these are nonmaskable interrupts, such as illegal instruction and COP Timer reset with an even higher priority. For more details, see **Table 7-7** in the *DSP56800 Family Manual*.

The Trigger mode for $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$ is defined in bits IAL1 and IAINV (for $\overline{\text{IRQA}}$) and IBL1 and IBINV (for $\overline{\text{IRQB}}$). Interrupts are enabled or disabled by bits IAL0 and IBL0. For details, please see **Table 5-1.**

Peripheral interrupts are enabled by writing to the IPR after preparing them to use the Status Register (SR). **Figure 5-1** illustrates the Interrupt Priority order and how IPR bits are programmed, as well as displaying interrupt programming. Unused bits are read as zero, ensuring future compatibility.

IPR_BASE- $X:FFFB	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CH0	CH1	CH2	CH3	CH4	CH5	CH6	0	0	0	IBL1	IBL0	IBINV	IAL1	IAL0	IAINV
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit Definition	*0 IPL	*1 IPL	*2 IPL	*3 IPL	*4 IPL	*5 IPL	*6 IPL	Reserved			$\overline{\text{IRQB}}$ Mode			$\overline{\text{IRQA}}$ Mode		

Note: * = Channel Channel 0 corresponds to an IPL of 0; Channel 6 corresponds to an IPL of 6.
Reserved bits read/write as zero

Figure 5-1. 56F826 IPR Programming Model

Table 5-1. Interrupt Programming

IBL1 IAL1	IBINV IAINV	Trigger Mode
0	0	Low-level sensitive
0	1	High-level sensitive
1	0	Falling-edge sensitive
1	1	Rising-edge sensitive

IBL0 IAL0	Enabled?	IPL
0	No	—
1	Yes	0

CH0-CH6	Enabled?	IPL
0	No	—
1	Yes	0

Note: To avoid spurious interrupts, it may be necessary to disable $\overline{\text{IRQx}}$ interrupts by clearing the IxL0 bit before modifying IxL1 or IxINV.

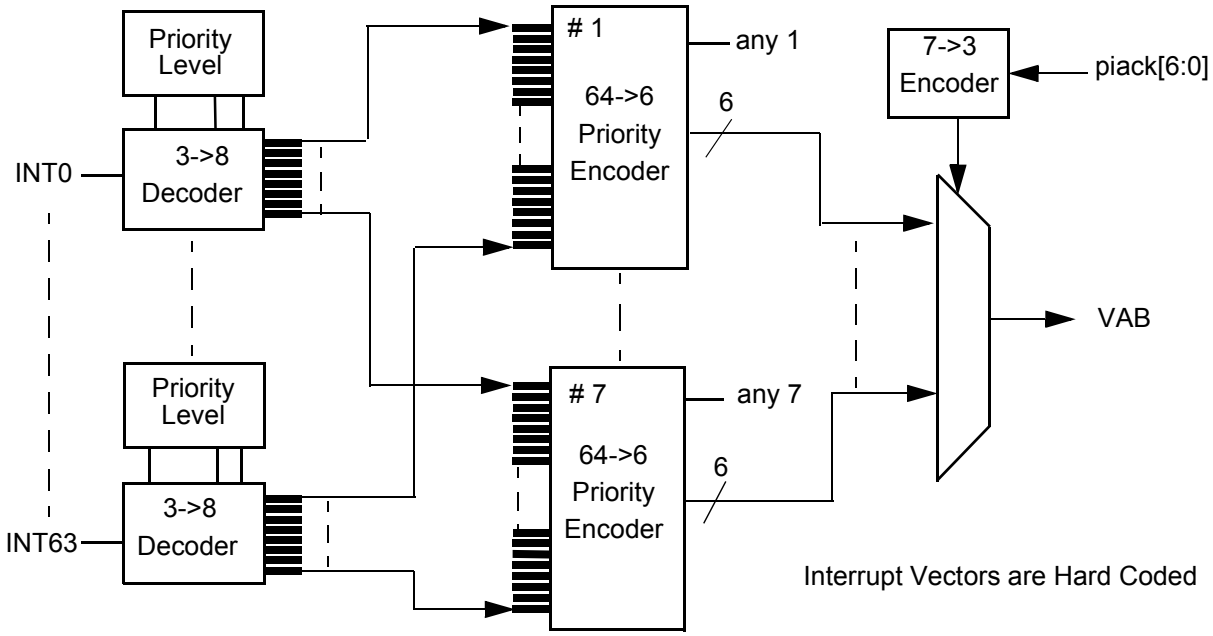


Figure 5-2. Extension to the Interrupt Controller

5.8 Interrupt Request Signals

Lists presented in this section document the Interrupt Request signals from each module.

5.8.1 Synchronous Serial Interface (SSI)

- Receives with exception
- Receives without exception
- Transmits with exception
- Transmits without exception
- Transmit and receive with exception

5.8.2 Serial Peripheral Interface (SPI0)

- Receiver full and/or error
- Transmitter empty

5.8.3 Serial Peripheral Interface (SPI1)

- Receiver full and/or error
- Transmitter empty

5.8.4 Serial Communications Interface (SCI0)

- Receiver full
- Receiver error
- Transmitter empty
- Transmitter complete

5.8.5 Serial Communications Interface (SCI1)

- Receiver full
- Receiver error
- Transmitter empty
- Transmitter complete

5.8.6 Serial Communications Interface (SCI2) (56F827 Only)

- Receiver full
- Receiver error
- Transmitter empty
- Transmitter idle

5.8.7 Analog-to-Digital Converter (ADC) (56F827 Only)

- Value out of limits or zero crossing
- Conversion complete

5.8.8 Timer Module (TMR A)

- Channel three flag
- Channel two flag
- Channel one flag
- Channel zero flag

5.8.9 Time-of-Day Module (TOD)

- Alarm interrupt
- One second interrupt

5.8.10 Combined Interrupt Requests for Port A (GPIOA)

- Eight GPIO pins in the 56F826
- Sixteen GPIO pins in the 56F827

5.8.11 Combined Interrupt Requests for Port B (GPIOB)

- Eight GPIO pins

5.8.12 Combined Interrupt Requests for Port C (GPIOC)

- Six GPIO pins in the 56F826
- Eight GPIO pins in the 56F827

5.8.13 Combined Interrupt Requests for Port D (GPIOD)

- Eight GPIO pins

5.8.14 Combined Interrupt Requests for Port E (GPIOE) (56F826 Only)

- Eight GPIO pins

5.8.15 Combined Interrupt Requests for Port F (GPIOF)

- Eight GPIO pins

5.8.16 Combined Interrupt Requests for Port G (GPIOG) (56F827 Only)

- Sixteen GPIO pins

5.8.17 Data Flash Interface Unit (DFIU)

- 2048 × 16 Flash

5.8.18 Program Flash Interface Unit (PFIU)

- 32252 × 16 Flash

5.8.19 Upper Program Flash Interface Unit (PFIU2) (56F827 Only)

5.8.20 Phase Lock Loop Module (PLL)

5.8.21 Low Voltage Detect (LVD)

5.9 Priority Level and Vector Assignments

Table 5-2 indicates the vector for each interrupt source. For a selected level, the highest vector has the highest priority.

5.10 ITCN Register Summary

The Interrupt Controller provides the following registers:

- IPR core at X:$FFFB. For additional information, please refer to **Figure 7-2** in the *DSP56800 Family Manual*
- SR core program controller unit
- Group Priority Register 2–15, GPR 0–15, contains 3-bit, PLR10–63

For interrupt controller registers of the address map, please refer to **Table 3-18.**

5.11 Priority Level and Vector Assignments

Table 5-2 indicates the vector for each interrupt source. For a selected level, the highest vector has the highest priority. The *x* in the table below indicates the interrupt source is available on either the 56F826 and/or 56F827 chips.

Table 5-2. Interrupt Vector Source and Addresses

826	827	Vector	IRQ Table Address	Interrupt Source Description
x	x	63	$007E	Low Voltage Detector
x	x	62	$007C	PLL Loss of Lock
x	x	61	$007A	SSI Transmit and Receive with Exception
x	x	60	$0078	SSI Receive with Exception
x	x	59	$0076	SSI Receive without Exception
x	x	58	$0074	SSI Transmit with Exception
x	x	57	$0072	SSI Transmit without Exception
—	x	56	$0070	Reserved (56F826) ADC Zero Crossing or Limit Error (56F827)
—	x	55	$006E	Reserved (56F826) ADC Conversion Complete (56F827)
x	x	54	006C	Reserved
x	x	53	$006A	SCI #0 Receiver Full
x	x	52	$0068	SCI #0 Receiver Error
x	x	51	$0066	SCI #0 Transmission Ready
x	x	50	$0064	SCI #0 Transmission Complete
x	x	49	$0062	SCI #1 Receiver Full
x	x	48	$0060	SCI #1 Receiver Error
x	x	47	$005E	SCI #1 Transmission Ready
x	x	46	$005C	SCI #1 Transmission Complete
—	x	45	$005A	Reserved (56F826) SCI #2 Receiver Full (56F827)
—	x	44	$0058	Reserved (56F826) SCI #2 Receiver Error (56F827)

Table 5-2. Interrupt Vector Source and Addresses (Continued)

826	827	Vector	IRQ Table Address	Interrupt Source Description
—	x	43	$0056	Reserved (56F826) SCI #2 Transmitter Ready (56F827)
—	x	42	$0054	Reserved (56F826) SCI #2 Transmit Complete (56F827)
—	x	41	$0052	Reserved
—	x	40	$0050	Reserved
—	x	39	$004E	Reserved
—	x	38	$004C	Reserved
x	x	37	$004A	Timer A Channel 3
x	x	36	$0048	Timer A Channel 2
x	x	35	$0046	Timer A Channel 1
x	x	34	$0044	Timer A Channel 0
x	x	33	$0042	TOD Alarm
x	x	32	$0040	TOD one sec
x	x	31	$003E	Reserved
x	x	30	$003C	Reserved
x	x	29	$003A	Reserved
x	x	28	$0038	Reserved
x	x	27	$0036	SPI #0 Receiver Full and/or Error
x	x	26	$0034	SPI #0 Transmitter Empty
x	x	25	$0032	SPI #1Receiver Full and/or Error
x	x	24	$0030	SPI #1 Transmitter Empty
x	x	23	$002E	GPIO A
x	x	22	$002C	GPIO B
x	x	21	$002A	GPIO C
x	x	20	$0028	GPIO D
x	—	19	$0026	GPIO E (56F826) Reserved (56F827)
x	x	18	$0024	GPIO F
—	x	17	$0022	Reserved (56F826) GPIO G (56F827)
x	x	16	$0020	Reserved
x	x	15	$001E	Reserved
—	x	14	$001C	Reserved (56F826) Data Flash Interface (56F827)
x	x	13	$001A	Data Flash Interface (56F826) Upper Program Flash Interface AU (56F827)
x	x	12	$0018	Program Flash Interface (56F826) Lower Program Flash Interface AL (56F827)
x	—	11	$0016	Boot Program Flash Interface (56F826) Reserved (56F827)
x	x	10	$0014	Reserved

Table 5-2. Interrupt Vector Source and Addresses (Continued)

826	827	Vector	IRQ Table Address	Interrupt Source Description
All vectors listed below in the table are controlled by the ITCN module				
All vectors listed below this point in the table are controlled by the 56800 Core				
x	x	9	$0012	Allocated to $\overline{\text{IRQB}}$
x	x	8	$0010	Allocated to $\overline{\text{IRQA}}$
x	x	7	$000E	Allocated to Factory
x	x	6	$000C	Allocated to OnCE Trap
x	x	5	$000A	Allocated to HW Stack Overflow
x	x	4	$0008	Allocated to SWI
x	x	3	$0006	Allocated to Illegal Instruction
x	x	2	$0004	Allocated to Factory
x	x	1	$0002	Allocated to COP / Watchdog Reset
x	x	0	$0000	Allocated to Hardware Reset

5.12 Register Definitions

Table 5-3. ITCN Memory Map

Device	Peripheral	Address
826/827	ITCN_BASE	$1100

The address of a register is the sum of a base address plus an address offset. The base address is defined as ITCN_BASE and the address offset is defined for each register in the subsections below.

Table 5-4. ITCN Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + $0	ITCN_GPR0	Group Priority Register 0	Read/Write	Section 5.12.1
Base + $1	ITCN_GPR1	Group Priority Register 1	Read/Write	
Base + $2	ITCN_GPR2	Group Priority Register 2	Read/Write	
Base + $3	ITCN_GPR3	Group Priority Register 3	Read/Write	
Base + $4	ITCN_GPR4	Group Priority Register 4	Read/Write	
Base + $5	ITCN_GPR5	Group Priority Register 5	Read/Write	
Base + $6	ITCN_GPR6	Group Priority Register 6	Read/Write	
Base + $7	ITCN_GPR7	Group Priority Register 7	Read/Write	
Base + $8	ITCN_GPR8	Group Priority Register 8	Read/Write	
Base + $9	ITCN_GPR9	Group Priority Register 9	Read/Write	
Base + $A	ITCN_GPR10	Group Priority Register 10	Read/Write	

Table 5-4. ITCN Register Summary (Continued)

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + $B	ITCN_GPR11	Group Priority Register 11	Read/Write	Section 5.12.1
Base + $C	ITCN_GPR12	Group Priority Register 12	Read/Write	
Base + $D	ITCN_GPR13	Group Priority Register 13	Read/Write	
Base + $E	ITCN_GPR14	Group Priority Register 14	Read/Write	
Base + $F	ITCN_GPR15	Group Priority Register 15	Read/Write	

Bit fields of the eight registers, each with 16 modules, are summarized in **Figure 5-3.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	GPR0	R		PLR3				PLR2				PLR2				PLR0		
		W																
$1	GPR1	R		PLR7				PLR6				PLR5				PLR4		
		W																
$2	GRP2	R		PLR11				PLR10				PLR9				PLR0		
		W																
$3	GRP3	R		PLR15				PLR14				PLR13				PLR12		
		W																
$4	GRP4	R		PLR19				PLR18				PLR17				PLR16		
		W																
$5	GRP5	R		PLR23				PLR22				PLR21				PLR20		
		W																
$6	GRP6	R		PLR27				PLR26				PLR25				PLR24		
		W																
$7	GRP7	R		PLR31				PLR30				PLR29				PLR28		
		W																
$8	GRP8	R		PLR35				PLR34				PLR33				PLR32		
		W																
$9	GRP9	R		PLR39				PLR38				PLR37				PLR36		
		W																
$A	GRP10	R		PLR43				PLR42				PLR41				PLR40		
		W																
$B	GRP11	R		PLR47				PLR46				PLR45				PLR44		
		W																
$C	GRP12	R		PLR51				PLR50				PLR49				PLR48		
		W																
$D	GRP13	R		PLR55				PLR54				PLR53				PLR52		
		W																
$E	GRP14	R		PLR59				PLR58				PLR57				PLR56		
		W																
$F	GRP15	R		PLR63				PLR62				PLR61				PLR60		
		W																

R	0	Read as 0
W		Reserved

Figure 5-3. ITCN Register Map Summary

5.12.1 Register Definitions (GPR2–GPR15)

Each interrupt source has an associated 3-bit PLR defining its priority level. The PLRs are read/write registers. They are packed together in groups of four per 16-bit register, except the Group Priority Register 1 (GPR2). These 16-bit registers are called GPR. Each vector register has the following format:

ITCN_BASE+$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR3				PLR2				PLR1				PLR0		
Write		Illegal Instruction				Factory Use				COP?Watchdog Reset				Hardware Reset		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-4. Group Priority Register 0 (GPR0)

See Programmer's Sheet on Appendix page B-21

ITCN_BASE+$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR7				PLR6				PLR5				PLR4		
Write		Factory Use				OnCE Trap				Hardware Stack Overflow				Software Interrupt		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-5. Group Priority Register 1 (GPR1)

See Programmer's Sheet on Appendix page B-21

ITCN_BASE+$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR11				PLR10				PLR9				PLR8		
Write		Boot Flash Interface-826 Reserved-827								$\overline{\text{IRQB}}$				$\overline{\text{IRQA}}$		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-6. Group Priority Register 2 (GPR2)

See Programmer's Sheet on Appendix page B-21

ITCN_BASE+$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR15				PLR14				PLR13				PLR12		
Write						Reserved-826 Data Flash Interface-827				Data Flash Interface-826 Upper Prog. Flash Interface AU-827				Prog.Flash Interface-826 Lower Prog. Flash INterface AL-827		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-7. Group Priority Register 3 (GPR3)

See Programmer's Sheet on Appendix page B-21

ITCN_BASE+$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR19 GPIO E-826 Reserved-827				PLR18 GPIO F				PLR17 Reserved-826 GPIO G-827				PLR16		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-8. Group Priority Register 4 (GPR4)

See Programmer's Sheet on Appendix page B-22

ITCN_BASE+$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR23 GPIO A				PLR22 GPIO B				PLR21 GPIO C				PLR20 GPIO D		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-9. Group Priority Register 5 (GPR5)

See Programmer's Sheet on Appendix page B-22

ITCN_BASE+$6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR27 SPI 0 Receiver Full and/or Error				PLR26 SPI 0 Transmitter Empty				PLR25 SPI 1 Receiver Full and/or Error				PLR24 SPI 1 Transmitter Empty		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-10. Group Priority Register 6 (GPR6)

See Programmer's Sheet on Appendix page B-22

ITCN_BASE+$7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR31				PLR30				PLR29 SPI 0 Reserved				PLR28 SPI 0 Reserved		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-11. Group Priority Register 7 (GPR7)

See Programmer's Sheet on Appendix page B-22

ITCN_BASE+$8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR35 Timer A Channel 1				PLR34 Timer A Channel 0				PLR33 TOD Alarm				PLR32 TOD One Sec		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-12. Group Priority Register 8 (GPR8)

See Programmer's Sheet on Appendix page B-23

ITCN_BASE+$9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR39				PLR38				**PLR37** Timer A Channel 3				PLR36 Timer A Channel 2		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-13. Group Priority Register 9 (GPR9)

See Programmer's Sheet on Appendix page B-23

ITCN_BASE+$A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		**PLR43** Reserved-826 SCI 2 Transmit Ready-827				**PLR42** Reserved-826 SCI 2 Transmit Complete-827				**PLR41**				**PLR40**		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-14. Group Priority Register 10 (GPR10)

See Programmer's Sheet on Appendix page B-23

ITCN_BASE+$B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		**PLR47** Transmission Ready				**PLR46** SCI 1 Transmission Complete				**PLR45** Reserved-826 SCI 2 Receiver Full-827				**PLR44** Reserved-826 SCI 2 Receiver Error-827		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-15. Group Priority Register 11 (GPR11)

See Programmer's Sheet on Appendix page B-23

ITCN_BASE+$C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		**PLR51** SCI 0 Transmission Ready				**PLR50** SCI 0 Transmission Complete				**PLR49** SCI 1 Receiver Full				**PLR48** SCI 1 Receiver Error		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-16. Group Priority Register 12 (GPR12)

See Programmer's Sheet on Appendix page B-24

ITCN_BASE+$C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		**PLR55** Reserved-826 ADC Conversion Complete-827				PLR54				**PLR53** SCI 0 Receiver Full				**PLR52** SCI 0 Receiver Error		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-17. Group Priority Register 13 (GPR13)

See Programmer's Sheet on Appendix page B-24

ITCN_BASE+$E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR59 SSI Transmit w/o Exception				PLR58 SSI Transmit w/Exception				PLR57 SSI Transmit w/o Exception				PLR56 Reserved-826 ADC Zero Crossing or Limit Error-827		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-18. Group Priority Register 14 (GPR14)

See Programmer's Sheet on Appendix page B-24

ITCN_BASE+$F	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR63 Low Voltage Detector				PLR62 PLL Loss of Lock				PLR61				PLR60 SSI w/Exception		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-19. Group Priority Register 15 (GRP15)

See Programmer's Sheet on Appendix page B-24

Chapter 6
Flash Memory Interface (FLASH)

6.1 Introduction

Flash Memory provides reliable, non-volatile memory storage, eliminating required external storage devices. Its software is easily programmable and can be booted directly from Flash.

6.2 Features

- Endurance: 100,000 cycles (typical)
- Memory organization for 56F826

	Main Block	Information Block
— Program Flash =	31.5K × 16	64 × 16
— Data Flash =	2K × 16	64 × 16
— Boot Flash =	2K × 16	64 × 16

- Memory organization for 56F827

	Main Block	Information Block
— Program Flash =	63K × 16	64 × 16
— Data Flash =	4K × 16	64 × 16

- Page erase capability: 256 words per page
- Block (mass) erase capability
- Access time: 20ns (max)
- Word (16-bit) program time: 20μs (min)
- Page Erase time: 40ms (min)
- Mass Erase time: 100ms (min)

6.3 Flash Description

The FLASH is a CMOS page erasable and up to 256 erasable words capability contained in a 16-bit program embedded Flash memory partitioned into two memory blocks:

1. Main memory block
2. Information block

The Main memory block is used for storage of application code and data. The information block can be used for the storage of device information.The Active block is selected by the IFREN bit in the FIU_CNTL register. Program and erase operations are performed on the Active block through the associated Flash Interface Unit (FIU).

The Flash Erase and programming voltages are regulated from the extreme 3.3V supply within the device. No special erase and programming voltages are required. The Page Erase operation erases all bytes within a page. A page is composed of eight adjacent rows for the main memory block or the two adjacent rows of the information block. PFLASH erases and programs with a 2.5V power supply. The Mass Erase operation erases all bytes within the Flash block. The Program operating may be performed on a single word or multiple words up to one row.

Table 6-1. Truth Table

Mode	XE[1]	YE[2]	SE[3]	OE[4]	PROG[5]	ERASE[6]	MAS1[7]	NVSTR[8]
Standby	L	L	L	L	L	L	L	L
Read	H	H	H	H	L	L	L	L
Word Program	H	H	L	L	H	L	L	H
Page Erase	H	L	L	L	L	H	L	H
Mass Erase	H	L	L	L	L	H	H	H

1. X address Enable, all rows are disabled when XE = 0
2. Y address Enable, YMUX is disabled when YE = 0
3. Sense amplifier enable; very useful if Flash is set in standby mode, cutting power requirement
4. Output Enable, tri-state Flash data out bus when OE = 0
5. Defines program cycle
6. Defines erase cycle
7. Defines mass erase cycle, erase whole block
8. Defines non-volatile store cycle

Table 6-2. IFREN Truth Table

Mode	IFREN = 1	IFREN = 0
Read	Reads Information Block	Read Main Memory Block
Word Program	Programs Information Block	Program Main Memory Block
Page Erase	Erases Information Block	Erase Main Memory Block
Mass Erase	Erases Both Block	Erase Main Memory Block

Table 6-3. Internal FLASH Timing Variables FLASH Timing Relationships

Parameter	Symbol
X Address Access Time	t_{XA}
Y Address Access Time	t_{YA}
OE Access Time	t_{OA}
PROG/ERASE to NVSTR Setup Time	t_{NVS}
NVSTR Hold Tme	t_{NVH}
NVSTR Hold Time (Mass Erase)	t_{NVH1}
NVSTR to Program Setup Time	t_{PGS}
Program Hold Time	t_{PGH}
Program Time	t_{PROG}
Address/Data Setup Time	t_{ADS}
Address/Data Hold Time	t_{ADH}
Recovery Time	t_{RCV}
Cumulative Program HV Period	t_{HV}
Erase Time	t_{ERASE}
Mass Erase Time	t_{ME}

6.4 Program Flash (PFLASH)

Devices described within this section use Flash blocks as non-volatile memory. The Flash block is instantiated within the Program Address Space (PFLASH). PFLASH is connected to core buses via a Program Flash Interface. PFLASH configurations for the devices described here are presented in **Table 6-4.**

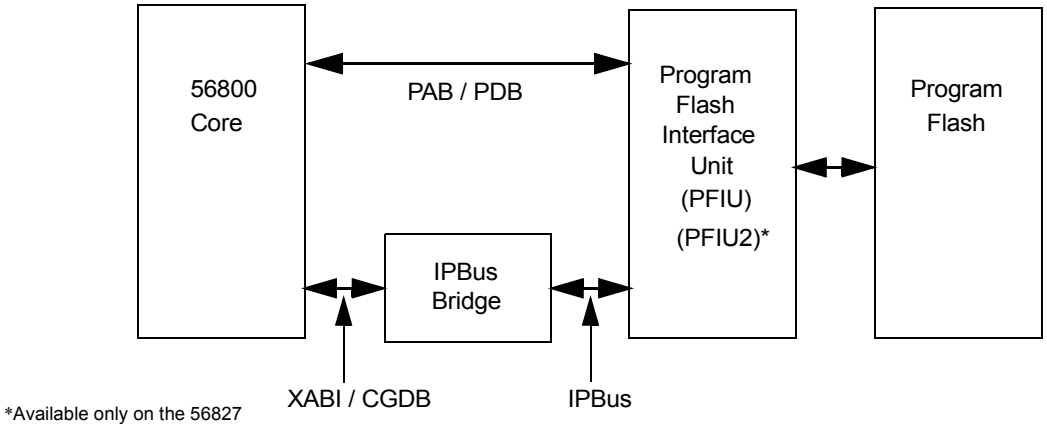


Figure 6-1. Program Flash Block Integration

Table 6-4. Program Flash Main Block Organization

Chip	Total Size	Row Size	#of Rows	Page Size	# of Pages
56F826	31.5 x 16 bits	512 bits (32 words)	1008	256 words (8 rows)	128
56F827	63 x 16 bits	512 bits (32 words)	2016	256 words (8 rows)	256

In addition to the Main Block sizes shown in the preceding table, each PFLASH contains a two-row Information Block.

Table 6-5. Program Flash Information Block Organization

Chip	Total Size	Row Size	#of Rows	Page Size	# of Pages
56F826	31.5 x 16 bits	512 bits (32 words)	2	31.5 words	
56F827	63 x 16 bits	512 bits (32 words)	2	63 words	

6.5 Data Flash (DFLASH)

The devices described in this section use Flash blocks as non-volatile memory. This section contains a description of the Flash block instantiated within the Data Address Space (DFLASH). The DFLASH is connected to the buses via a Data Flash Interface.

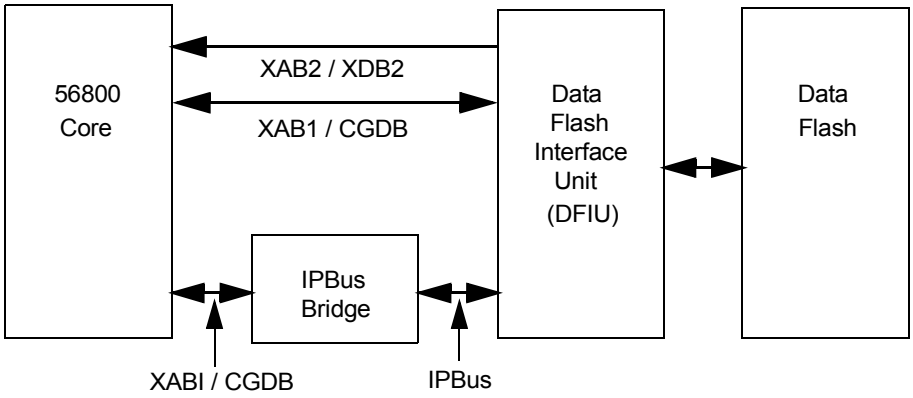


Figure 6-2. Data Flash Block Integration

DFLASH configurations for the devices are described in **Table 6-6.**

Table 6-6. Data Flash Main Block Organization

Chip	Total Size	Row Size	# of Rows	Page Size	# of Pages
56F826	2 x 16 bits	512 bits (32 words)	64	256 words (8 rows)	8
56F827	4 x 16 bits	512 bits (32 words)	128	256 words (8 rows)	16

In addition to the Main Block sizes shown in the preceding table, each DFLASH contains a two-row Information Block. The Data Flash is a single-port Flash. Other than size, its features are similar to those of the Program Flash.

Table 6-7. Data Flash Information Block Organization

Chip	Total Size	Row Size	# of Rows	Page Size	# of Pages
56F826	2 x 16 bits	512 bits (32 words)	2		
56F827	4 x 16 bits	512 bits (32 words)	2		

6.6 Boot Flash (BFLASH) 56F826 Only

Devices described in this section use Flash blocks as non-volatile memory. The Flash block is instantiated within the Boot Address Space (BFLASH). BFLASH is connected to the core buses via a Boot Flash Interface.

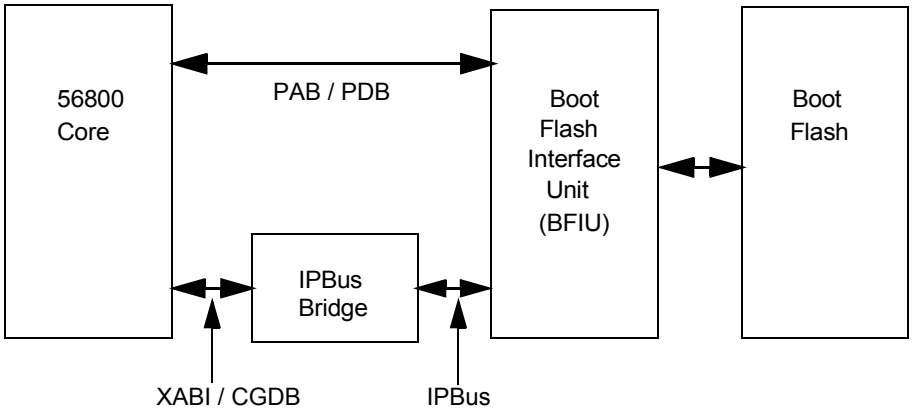


Figure 6-3. Boot Flash Block Integration

BFLASH configurations for the devices are described in **Table 6-8.**

Table 6-8. Boot Flash Main Block Organization

Chip	Total Size	Row Size	# of Rows	Page Size	# of Pages
826	2 x 16 bits	512 bits (32 words)	64	256 words (8 rows)	8

In addition to the Main Block sizes shown in the preceding table, each BFLASH contains a two-row information block.

Table 6-9. Boot Flash Main Information Organization

Chip	Total Size	Row Size	# of Rows	Page Size	# of Pages
826	2 x 16 bits	512 bits (32 words)	2		

6.7 Program/Data/Boot Flash Interface Unit Features

This section details Program, Data, and Boot Flash Interface Unit features:

- Single port memory compatible with core pipelined program bus structure
- Single cycle reads at 40MHz across the automotive temperature range

- Word (16-bit) readable and programmable
- Supports memory organization defined in **Chapter 3, Memory and Operating Modes**
- Page Erase and Mass Erase modes
- Can be programmed under software control in the developer's system

6.8 Program/Data/Boot Flash Modes

By changing any of the following registers, the Information Block may be swapped for the Main Memory Block:

- IFREN bit of the PFIU_CNTL
- IFREN bit of the DFIU_CNTL
- IFREN bit of the BFIU_CNTL

Modes of operation include:

- Standby
- Read word
- Simple single word program operation
- Up to one row of multiword programming is possible
- Page erase
- Mass erase
- The PFLASH, DFLASH, or BFLASH is automatically placed in Standby mode when not being read, programmed, or erased
- Reads are disabled during program/erase operation
- An interrupt can be individually maskable
- An interrupt will be generated for:
  - Reads out of range, trying to access a row past the first two in the Information Block
  - Read attempted during program
  - Read attempted during erase
  - Any of seven internal timer timeouts used during program/erase operations
- Intelligent erase and word programming features

6.9 Functional Description of the PFIU, PFIU2, DFIU and BFIU

- PFIU, PFIU2, DFIU, and BFIU have internal timers for:
  - t_{NVS} represents PROG/ERASE to NVSTR Setup Time
  - t_{NVH} and t_{NVH1} represent NVSTR Hold Times
  - t_{PGS} represents NVSTR to Program Setup Time
  - t_{PROG} represents Program Time
  - t_{REC} represents Recovery Time
  - t_{ERASE} represents Erase Time
  - t_{ME} represents Mass Erase Time
- These are initialized upon reset to default values are expected to work correctly for Word Programming and Page Erase operations. They may be overwritten by application code after reset if it is necessary to adjust operation based upon Flash characterization data.
- All Read/Program/Erase functions are terminated by a system reset
- The Flash will automatically be put in a Standby mode when not being accessed
- During an Erase Cycle, all bits within the affected area are changed to one. Programming a bit results in it being changed to zero. Bits can only be changed back to one by erasing that section of memory. *A bit cannot be programmed to one*.

6.10 Flash Programming and Erase Models

Please refer to the figures in **Section 6.3** for Flash timing relationships required in the following Intelligent Word, Dumb Word Programming, and Intelligent Erase operation.

The recommended Flash Programming mode is the Intelligent Word Programming, described in **Section 6.10.1.** *Although the Dumb Word Programming is described in* **Section 6.10.2,** *it is not the preferred nor recommended method of programming Flash.*

6.10.1 Intelligent Word Programming

Assuming the word in question has already been erased, a single 16-bit word may be programmed through the following method:

1. Start with the Flash in either Standby or Read modes. Read FIU_CNTL register to insure this is true. All bits should be clear with the exception of IFREN when programming the Information Block. Please refer to **Section 6.9.**
2. Enable programming by setting the Intelligent Program Enable (IPE) bit and row number in the FIU_PE register. To calculate the row number, use the following algorithm:
   - Target_Address is the logical AND operation of $7FFF divided by $20 = ROW
   - Put differently, set the MSB of the target address to zero and right shift the result five places
3. Set the IE[8] bit in the FIU_IE register enabling to t_{RCV} interrupt if desired to flag completion of programming.
4. Write the value desired to the proper word in the Flash Memory Map. Note a single location in the Flash may map to different locations in the Memory Map based upon the mode selected on startup. The FIU will adjust accordingly. This write to the Flash memory map while the IPE bit is set will start the internal state machine to run the Flash through its programming paces.
5. Do not attempt to access the Flash again until the BUSY signal clears in the FIU_CNTL register, corresponding to the t_{RCV} interrupt when enabled.
6. When programming is completed, be certain to clear the IPE bit in the FIU_PE register.

The Intelligent Programming feature can program *only* one word at a time. Multiple words may be programmed by repeating the process, or by switching to the Dumb Word Programming method, subsequently outlined.

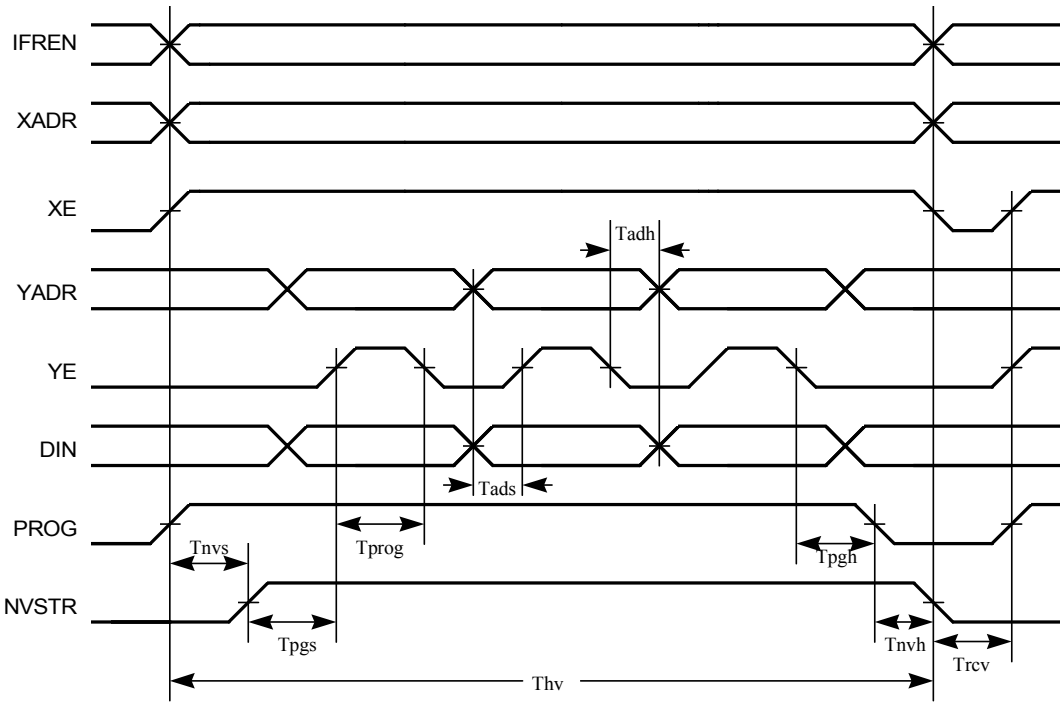


Figure 6-4. Flash Program Cycle

6.10.2 Dumb Word Programming

Flash allows programming of one or more words in a row. Assuming the words in question have already been erased, 16-bit words may be programmed using the following method:

1. Start with the Flash in either Standby or Read modes. Read the BUSY bit in the FIU_CNTL register ensuring it is clear.
2. Enable programming by setting the DPE bit and row number in the FIU_PE register.
3. Write the desired value to the applicable word in the Flash Memory space assuring IS[0] is clear before proceeding.
4. Set the XE and PROG bits in the FIU_CNTL register.
5. Delay for Tnvs
6. Set the NVSTR bit in the FIU_CNTL register.
7. Delay for t_{PGS}
8. Set the YE bit in the FIU_CNTL register.
9. Delay for T_{PROG}.
10. Clear the YE bit in the FIU_CNTL register.

11. By writing to another location in the Flash Memory space, update the values in FIU_ADDR and FIU_DATA if additional words are required, loop back to step eight above. Address changes must be limited to within a single row making certain IS[0] is clear before proceeding. Attempts to change the FIU_CNTL register with IS[0] set will have no effect.
12. Clear the PROG bit in the FIU_CNTL register, creating a separate step from clearing YE.
13. Delay for t_{NVH}
14. Clear the NVSTR and XE bits in the FIU_CNTL register.
15. Delay for t_{RCV}

The FIU timer registers *cannot* be used to generate interrupts for each of the delays above. If interrupts are desired to generate the timing breakpoints for the Dumb Programming process, they must be generated from some other timing or interrupt source.

6.10.3 Intelligent Erase Operation

1. Start with the Flash in either Standby or Read modes. Read the FIU_CNTL register ensuring this is true. All bits should be clear with the exception of IFREN when erasing the Information Block. Please see **Section 6.9.**
2. To flag the completion of Erase, set the IE[8] bit in the FIU_IE register, enabling a t_{RCV} interrupt.
3. Enable Erase by setting the IEE bit and page number in the FIU_EE register. To calculate a page number, use the following algorithm:
   — Target_Address AND $7FFF divided by $100 = PAGE.
   — Put differently, set the MSB of the target address to zero and right shift the result eight places.
4. For a Mass Erase, set the page to $0 and set MAS1 bit of the FIU_CNTL register.
5. While the IEE bit is set, write any value to an address in the page to be erased. This write to the Flash Memory Map will start the FIU internal state machine to run the Flash through its erase paces. Flash logic will erase the entire row, consequently it is not necessary to YADR[4.0] sequence within FIU.

Note: In MODE0A, the first four locations of the program memory map are mapped to BFLASH.

6. Do not attempt to access the Flash again until the BUSY signal clears in the FIU_CNTL register, corresponding to the t_{RCV} interrupt, when it is enabled.
7. Ensure FIU_CNTL and FIU_EE registers are cleared when finished.

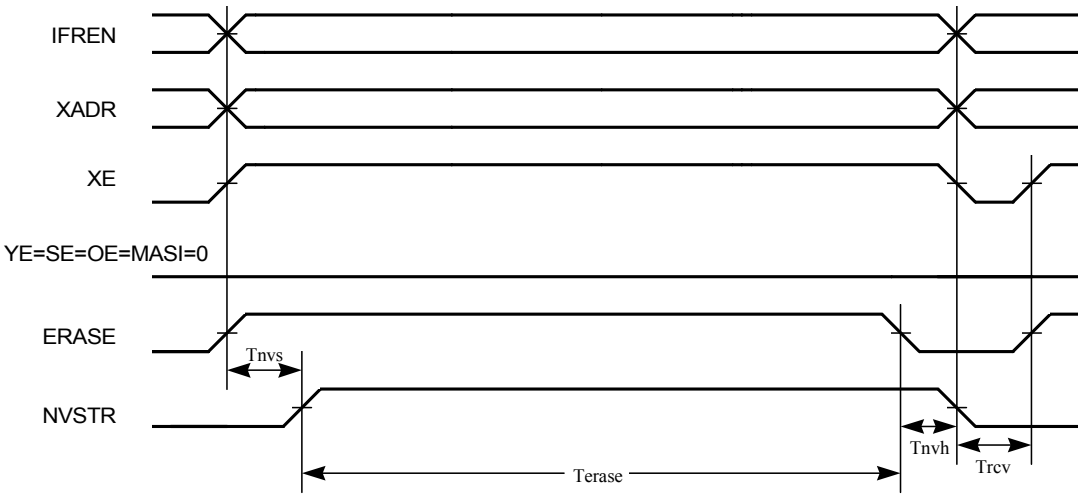


Figure 6-5. FLASH Page Erase Cycle

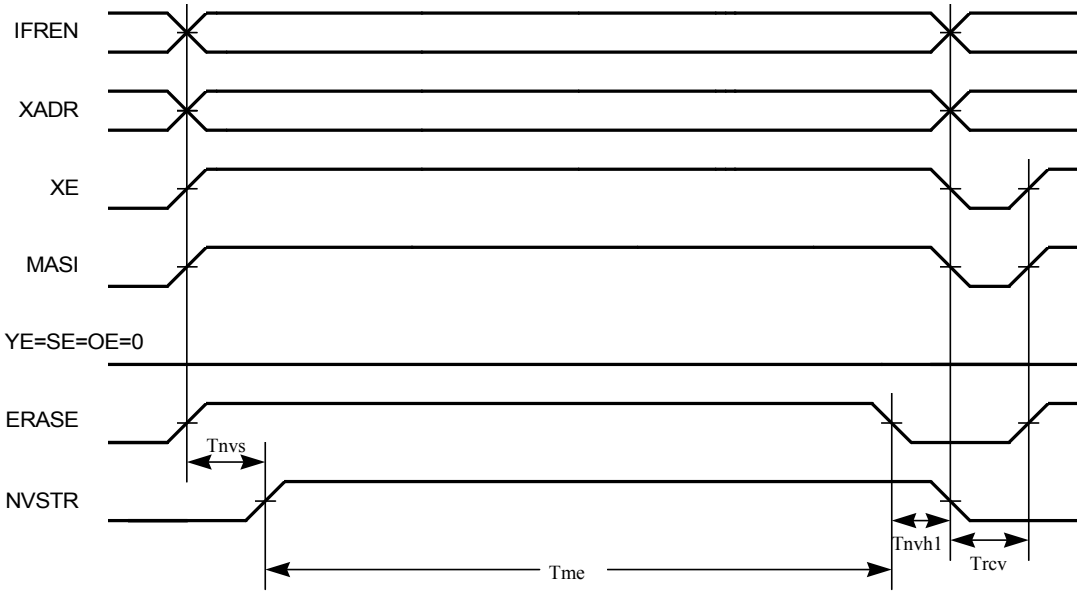


Figure 6-6. Flash Mass Erase Cycle

6.11 Register Definitions

Table 6-10. Flash Memory Map

Device	Peripheral	Address
826/827	PFIU_BASE	$1020
	DFIU_BASE	$1060
826 Only	BFIU_BASE	$1080
827 Only	PFIU2_BASE	$1040

The Memory Map and Register Definitions are identical for Program Flash Interface Units (PFIU, PFIU2), Data Flash Interface Unit (DFIU), and Boot Flash Interface Unit (BFIU).

For each Flash Interface Unit (FIU) use the corresponding letters:

- *P* = Program
- *D* = Data
- *B* = Boot

One of the above letters must be included in front of FIU in *both the register name and base address to indicate the Flash type*. For example, the FIU_CNTL definition is same for PFIU_CNTL at memory location PFUI_BASE+0, DFIU_CNTL at memory location DFUI_BASE+0, and BFIU_CNTL at memory location BFUI_BASE+0.

For more information about PFIU registers, please refer to **Table .** Also, please refer to **Table** for DFIU registers, and **Table 3-32** for BFIU registers. For PFIU2 registers used *only* for the 56F827, please refer to **Table 3-13.**

The address of a register is the sum of a base address and an address offset. The base address is defined as the corresponding FIU_BASE and the address. Offset is defined for each register for the PFIU_BASE, PFIU2_BASE, DFIU_BASE, and BFIU_BASE definition in **Table 3-11.**

The interface also block-write protects all Flash registers except the active one. The Flash Interface Units are the same for Program Flash Interface Unit (PFIU). Since the 56F826/827 has 64K of Program Flash, 60K are usable while 4K is reserved, an extra Program Flash Interface Unit (PFIU2) is required to access the second half of Flash.

Table 6-11. FLASH Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + $0	FIU_CNTL	Control Register	Read/Write	Section 6.11.1
Base + $1	FIU_PE	Program Enable Register	Read/Write	Section 6.11.2
Base + $2	FIU_EE	Erase Enable Register	Read/Write	Section 6.11.3
Base + $3	FIU_ADDR	Address Register	Read/Write	Section 6.11.4
Base + $4	FIU_DATA	Data Register	Read/Write	Section 6.11.5
Base + $5	FIU_IE	Interrupt Enable Register	Read/Write	Section 6.11.6
Base + $6	FIU_IS	Interrupt Source Register	Read/Write	Section 6.11.7
Base + $7	FIU_IP	Interrupt Pending Register	Read/Write	Section 6.11.8
Base + $8	FIU_CKDIVISOR	Clock Divisor Register	Read/Write	Section 6.11.9
Base + $9	FIU_TERASEL	TERASE Limit Register	Read/Write	Section 6.11.10
Base + $A	FIU_TMEL	TME Limit Register	Read/Write	Section 6.11.11
Base + $B	FIU_TNVSL	TNVS Limit Register	Read/Write	Section 6.11.12
Base + $C	FIU_TPGSL	TPGS Limit Register	Read/Write	Section 6.11.13
Base + $D	FIU_TPROGL	TPROG Limit Register	Read/Write	Section 6.11.14
Base + $E	FIU_TNVHL	TNVH Limit Register	Read/Write	Section 6.11.15
Base + $F	FIU_TNVH1L	TNVH1L Limit Register	Read/Write	Section 6.11.16
Base + $10	FIU_TRCVL	TRCV Limit Register	Read/Write	Section 6.11.17

Bit fields of each of the 17 registers are summarized in **Figure 6-7.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CNTL	R	BUSY	0	0	0	0	0	0	0	0	IFREN	XE	YE	PROG	ERASE	MAS1	NVSTR
		W																
$1	PE	R	DPE	IPE	0	0	0	0	ROW									
		W																
$2	EE	R	DEE	IEE	0	0	0	0	0	0	0	PAGE						
		W																
$3	ADDR	R	A															
		W																
$4	DATA	R	D															
		W																
$5	IE	R	0	0	0	0	IE											
		W																
$6	IS	R	0	0	0	0	IS											
		W																
$7	IP	R	0	0	0	0	IP											
		W																
$8	CKDIVISOR	R	0	0	0	0	0	0	0	0	0	0	0	0	N			
		W																
$9	TERASEL	R	0	0	0	0	0	0	0	0	0	TERASEL						
		W																
$A	TMEL	R	0	0	0	0	0	0	0	0	TMEL							
		W																
$B	TNVSL	R	0	0	0	0	0	TNVSL										
		W																
$C	TPGSL	R	0	0	0	0	TPGSL											
		W																
$D	TPROGL	R	0	0	TPROGL													
		W																
$E	TNVHL	R	0	0	0	0	0	TNVHL										
		W																
$F	TNVH1L	R	0	TNVH1L														
		W																
$10	TRCVL	R	0	0	0	0	0	0	0	TRCVL								
		W																

R	0	Read as 0
W		Reserved

Figure 6-7. FLASH Register Map Summary

6.11.1 Flash Control Register (FIU_CNTL)

FIU_BASE+$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	BUSY	0	0	0	0	0	0	0	0	IFREN	XE	YE	PROG	ERASE	MAS1	NVSTR
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-8. Flash Control Register (FIU_CNTL)

See Programmer's Sheet on Appendix page B-25

6.11.1.1 BUSY (BUSY)—Bit 15

When the BUSY bit is asserted in Intently Programming mode, *no register can receive writing*. In the Dumb Programming mode, *only* the Address and Data registers can receive writing. The BUSY bit in the FIU_CNTL register will be set if any of the following are true:

- If PROG is asserted
- If NVSTR is asserted
- If ERASE is asserted

If the T_{RCV} Counter is running Intelligent Program/Erase modes, Flash reads should not be attempted while BUSY is asserted. The core should be prevented from entering Sleep or Low-Power mode if BUSY is asserted OR when FIU_IS is not equal to $0000. The BUSY bit is cleared once PROG, NVSTR, ERASE, XE and YE bits have been cleared.

6.11.1.2 Reserved—Bits 14–7

This bit field is reserved or is not implemented. It is read as 0 and cannot be modified by writing.

6.11.1.3 Information Block Enable (IFREN)—Bit 6

When IFREN is asserted, the information block of the Flash is enabled. The bit permits write in all modes when the BUSY bit is clear. The IFREN bit has the effect presented in **Table 6-12.**

Table 6-12. IFREN Bit Effect

Function	IFREN=1	IFREN=0
Read	Read Information Block	Read Main Memory Block
Word Program	Program Information Block	Program Main Memory Block
Page Erase	Erase Information Block	Erase Main Memory Block
Mass Erase	Erase Both Blocks	Erase Main Memory Block

6.11.1.4 X Address Enable (XE)—Bit 5

This bit enables the X address when asserted. It can be written *only* in the Dumb Programming mode when the IS[0] bit is clear. The bit can be read *only* in Intelligent Programming mode, reflecting the state of the Flash XE pin.

6.11.1.5 Y Address Enable (YE)—Bit 4

This bit enables the Y Address when asserted. It can be written *only* in the Dumb Programming mode when the IS[0] bit is clear. The bit can be read *only* in the Intelligent Programming mode, reflecting the state of the Flash YE pin.

6.11.1.6 Program Cycle Definition (PROG)—Bit 3

The Program Cycle is enabled when the PROG bit is asserted. It can be written *only* when the DPE bit is set and the IS[0] bit is clear. The bit can be read *only* in the Intelligent Programming mode, reflecting the state of the Flash PROG pin. This bit *cannot* be set when either ERASE or MAS1 is set.

6.11.1.7 Erase Cycle Definition (ERASE)—Bit 2

The Erase Cycle is enabled when the ERASE bit is asserted. It can be written *only* when the DEE bit is set and the IS[0] bit is clear. The bit is read *only* in the Intelligent Programming mode, reflecting the state of the Flash ERASE pin. This bit *cannot* be set when PROG is set.

6.11.1.8 Mass Erase Cycle Definition (MAS1)—Bit 1

When the MAS1 bit is asserted, the Mass Erase Cycle is enabled causing all pages to be automatically enabled. It can be written *only* when IEE or DEE are set and the BUSY bit is clear. The read value in the Intelligent Programming mode is a reflection of the state of the Flash MAS1 pin and may not represent what was last written to the MAS1 bit. This bit *cannot* be set when PROG is set.

6.11.1.9 Non-Volatile Store Cycle Definition (NVSTR)—Bit 0

The Non-Volatile Store Cycle is enabled when the NVSTR bit is asserted. The bit can be written *only* in the Dumb Programming mode when the IS[0] bit is clear. The bit is read *only* in Intelligent Programming mode, reflecting the state of the Flash NVSTR pin.

6.11.2 Flash Program Enable Register (FIU_PE)

This register is reset upon any system reset. This register *cannot* be modified while the BUSY bit is asserted. IS[11] is asserted when this occurs.

FIU_BASE+$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DPE	IPE	0	0	0	0	ROW									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-9. Flash Program Enable Register (FIU_PE)

See Programmer's Sheet on Appendix page B-26

6.11.2.1 Dumb Program Enable (DPE)—Bit 15

Dumb Programming is enabled when DPE is asserted. This bit *cannot* be set if any of IPE, DEE, or IEE bits are already active. Before DPE can be cleared, PROG, NVSTR, XE, and YE bits in the FIU_CNTL must also be cleared.

6.11.2.2 Intelligent Program Enable (IPE)—Bit 14

Intelligent Programming is enabled when IPE is asserted. This bit *cannot* be set if any of DPE, DEE, or IEE bits are already active.

6.11.2.3 Reserved—Bits 13–10

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.2.4 Row Number (ROW)—Bits 9–0

ROW represents the number of the row to be programmed. The ROW number is calculated by forcing the MSB of the target address to zero, then right shifting the result five places.

To program a row, the row number must be written as ROW[9:0]. A *write must* be performed to the same row to begin the programming sequence. Writing to a different row will *not* start the state machine nor enable PROG and NVSTR. The Interrupt Source bit IS[0] is set to indicate an error. This is intended as a double check against accidental programming. These checks also apply for the Dumb Programming mode.

6.11.3 Flash Erase Enable Register (FIU_EE)

This register is reset upon any system reset. This register may *not* be modified while the BUSY bit is asserted. Should this happen, IS[11] is also asserted.

FIU_BASE+$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DEE	IEE	0	0	0	0	0	0	0	PAGE						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-10. Flash Erase Enable Register (FIU_EE)

See Programmer's Sheet on Appendix page B-27

6.11.3.1 Dumb Erase Enable (DEE)—Bit 15

Dumb Erase Enable (DEE) is enabled when the DEE bit is asserted. This bit *cannot* be set if any DPE, IPE, or IEE bits are already active. ERASE, NVSTR, XE, and YE must be cleared first before DEE can be cleared.

6.11.3.2 Intelligent Erase Enable (IEE)—Bit 14

Intelligent Erase Enable (IEE) is enabled when the IEE bit is asserted. This bit cannot be set if any of DPE, IPE, or DEE bits are already active.

6.11.3.3 Reserved—Bits 13–7

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.3.4 Page Number (PAGE)—Bits 6–0

The PAGE bit field must be set to the page number about to be erased. The page number is calculated by forcing the MSB of the target address to zero, then right shifting the result eight places.

To erase a page when MAS1 = 0 in FIU_CNTL, that page number to be erased must be written to PAGE. A *write must* be performed to the same page will begin the erase sequence. Writing to a different page will *not* start the intelligent erase state machine nor will it enable ERASE and NVSTR. Interrupt Source IS[0] bit is set to indicate an error. This is intended as a double check against accidental erasure.

Similarly, set PAGE to be $00 when MAS1 = 1. These checks also apply for the Dumb Erase mode.

6.11.4 Flash Address Register (FIU_ADDR)

This register can be read as a register on the IPBus and is cleared upon any system reset. This register is designed for program development and error analysis.

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

Figure 6-11. Flash Address Register (FIU_ADDR)

See Programmer's Sheet on Appendix page B-28

This value of A[15:0] represents the current program or erase address. It can be read at any time. This register is set by attempting a write to memory space occupied by the Flash Memory.

When this write is enabled, the event can begin the Intelligent Program/Erase state machines through their paces. This register can *only* be set once at the start of an Intelligent Program/Erase operation. Until the BUSY bit clears, subsequent writes have no effect on A[15:0], and an interrupt source (IS[1] or IS[2]) is set to indicate an error. In Dumb Program/Erase modes, this register will be latched whenever FIU_CNTL bits PROG, ERASE, or NVSTR first go true and may *only* be changed after a negative edge of YE.

6.11.5 Flash Data Register (FIU_DATA)

This register is cleared upon any system reset. This register is designed for program development and error analysis.

FIU_BASE+$4	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 6-12. Flash Data Register (FIU_DATA)

See Programmer's Sheet on Appendix page B-29

Data bits (D[15:0]) reflect the last value programmed, or the value written to initiate an erase. FIU_ADDR and FIU_DATA are set simultaneously by writing to Flash Memory space. This value can be read at any time.

The register can only be set *once* at the start of an Intelligent Program or Intelligent Erase operation. Until the BUSY bit clears, subsequent writes have no effect on D[15:0], and an interrupt source (IS[1] or IS[2]) will be set to indicate an error.

6.11.6 Flash Interrupt Enable Register (FIU_IE)

This register is reset upon any system reset. It may *only* be written when the BUSY bit is clear. IS[11] is asserted should this occur.

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

Figure 6-13. Flash Interrupt Enable Register (FIU_IE)

See Programmer's Sheet on Appendix page B-30

6.11.6.0.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.6.0.2 Flash Interrupt Enable (IE)—Bits 11–0

Interrupt enables for IS[11:0] respectively. For example, if IE[3] is enabled, it can be activated when its *interrupt source* is triggered. It is bit wise and of IE[3] and IS[3] IP[11:0] = IE[11:0] is the logical AND result of IS[11:0]. To see a specific list of interrupt sources, refer to **Section 6.11.7** for bit details.

6.11.7 Flash Interrupt Source Register (FIU_IS)

Under normal operation, *only* clear bits in FIU_IS can be cleared. The IS source bits can be cleared at any time. This register is reset upon any system reset.

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

Figure 6-14. Flash Interrupt Source Register (FIU_IS)

See Programmer's Sheet on Appendix page B-31

6.11.7.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. bits cannot be modified. Bits are read as zero.

6.11.7.2 Interrupt Source (IS[11])—Bit 11

This Interrupt Source bit is asserted when a write to a register is attempted and while the BUSY bit is asserted.

6.11.7.3 Interrupt Source (IS[10])—Bit 10

This Interrupt Source bit is asserted when a write is attempted to FIU_CNTL during Intelligent Program or Intelligent Erase.

6.11.7.4 Interrupt Source (IS[9])—Bit 9

This Interrupt Source bit is asserted when there is a t_{ERASE} or t_{MEL} (Erase Time) timeout.

6.11.7.5 Interrupt Source (IS[8])—Bit 8

This Interrupt Source bit is asserted when there is a t_{RCV} (Recovery Time) timeout.

6.11.7.6 Interrupt Source (IS[7])—Bit 7

This Interrupt Source bit is asserted when there is a t_{PROG} (Program Time) timeout.

6.11.7.7 Interrupt Source (IS[6])—Bit 6

This Interrupt Source bit is asserted when there is a t_{PGS} (NVSTR to Program Setup Time) timeout.

6.11.7.8 Interrupt Source (IS[5])—Bit 5

This Interrupt Source bit is asserted when there is a t_{NVH1} (NVSTR Hold Time) timeout.

6.11.7.9 Interrupt Source (IS[4])—Bit 4

This Interrupt Source bit is asserted when there is a t_{NVH} (NVSTR Hold Time) timeout.

6.11.7.10 Interrupt Source (IS[3])—Bit 3

This Interrupt Source bit is asserted when there is a t_{NVS} (PROG/ERASE to NVSTR Setup Time) timeout.

6.11.7.11 Interrupt Source (IS[2])—Bit 2

This Interrupt Source bit is asserted when an illegal Flash read/write access is attempted during erase.

6.11.7.12 Interrupt Source (IS[1])—Bit 1

This Interrupt Source bit is asserted when an illegal Flash read/write access is attempted during program.

6.11.7.13 Interrupt Source (IS[0])—Bit 0

This Interrupt Source bit is asserted when a Flash Program or Flash Erase access out-of-range is attempted.

6.11.8 Flash Interrupt Pending Register (FIU_IP)

This register is read *only* and reset upon any system reset. Use this register for indirectly controlling the interrupt service routine.

FIU_BASE+$7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	IP11	IP10	IP9	IP8	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-15. Flash Interrupt Pending Register (FIU_IP)

See Programmer's Sheet on Appendix page B-32

6.11.8.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.8.2 Interrupt Pending ([IP11:0])—Bits 11–0

Flash Interface Unit is asserting an interrupt to the core when any of IP[11:0] are asserted. Interrupts are cleared by terminating the appropriate bit(s) in FIU_IS. Write 1 to bits not affected. IP[x] is a bit wise logical *and* operation of IE[x] and IS[x].

For IP[3] to be asserted, IE[3] has to be enabled *and* IS[3] must be asserted. This is because of the corresponding interrupt source being triggered: IP[11:0] = IE[11:0] *and* IS[11:0]. To see a specific list of interrupt sources, please refer to **Section 6.11.7** for bit details.

6.11.9 Flash Clock Divisor Register (FIU_CKDIVISOR)

This register is reset during any system reset. It may only be changed when the BUSY bit is clear. IS[11] is asserted should BUSY bit be cleared.

FIU_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	N			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 6-16. Flash Clock Divisor Register (FIU_CKDIVISOR)

See Programmer's Sheet on Appendix page B-33

6.11.9.1 Reserved—Bits 15–4

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.9.2 Clock Divisor (N)—Bits 3–0

The bus clock is divided by $2^{(N+1)}$ prior to clocking the t_{ERASE} and t_{ME} timers where N is the Clock Divisor. The inverse of this ratio determines the prescale clock value and is given by the equation below:

$$PS = t_{IPBUS} \times 2^{(N+1)}$$

A typical bus clock period, based on a 40 MHz IP_BUS frequency, is:

$$t_{IPBUS} = 25ns$$

The typical default prescale clock value is:

$$PS_{TYP} = 25ns \times 2^{16} = 16.384 \times 10^{5}ns$$

This prescale clock value is ONLY used by the t_{ERASE} and t_{ME} timers.

6.11.10 Flash T_{ERASE} Limit Register (FIU_TERASEL)

FIU_BASE+$9	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	TERASEL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 6-17. Flash T_{ERASE} Limit Register (FIU_TERASEL)

See Programmer's Sheet on Appendix page B-34

This register is reset during any system reset. The register may *only* be changed when the BUSY bit is clear. IS[11] is asserted should this occur.

6.11.10.1 Reserved—Bits 15–7

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.10.2 Timer Erase Limit (TERASEL)—Bits 6–0

The bus clock is divided by the Clock Divisor (PS) prior to clocking the t_{ERASE} timer. TERASEL is the maximum erase time expressed as a multiple of this divided clock. The actual count use is t_{ERASE} +1. Therefore, the erase times based on a 40MHz IP_BUS frequency, are:

$$t_{ERASE} = PS \times (t_{ERASEL} + 1)$$
$$t_{ERASE}\ Max = PS_{TYP} \times 256 = 419.4ms$$
$$t_{ERASE30}\ Reset = PS_{TYP} \times 16 = 26.2ms$$

6.11.11 Flash T_{ME} Limit Register (FIU_TMEL)

This register is reset during any system reset. This register may *only* be changed when the BUSY bit is clear. IS[11] is asserted when the BUSY bit is cleared.

FIU_BASE+$A	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	TMEL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 6-18. Flash T_{ME} Limit Register (FIU_TMEL)

See Programmer's Sheet on Appendix page B-35

6.11.11.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.11.2 Timer Mass Erase Limit (TMEL)—Bit 7–0

The bus clock is divided by the Clock Divisor (PS) prior to clocking the t_{ME} timer. TMEL is the maximum mass erase time expressed as a multiple of this divided clock. Therefore, the maximum erase times is:

$$t_{ME} = PS \times (t_{MEL} + 1)$$
$$t_{ME}\ Max = PS_{TYP} \times 256 = 419.4ms$$
$$t_{ME}\ Reset = PS_{TYP} \times 32 = 52.4ms$$

6.11.12 Flash T_{NVS} Limit Register (FIU_TNVSL)

This register is reset during any system reset and may only be changed when the BUSY bit is clear. IS[11] is asserted when BUSY bit is clear.

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

Figure 6-19. Flash T_{NVS} Limit Register (FIU_TNVSL)

See Programmer's Sheet on Appendix page B-36

6.11.12.1 Reserved—Bits 15–11

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.12.2 Timer Non-Volatile Storage Limit (TNVSL)—Bits 10–0

Timeout value for the t_{NVS} counter is in terms of bus clocks. This counter controls the PROG/ERASE to NVSTR Setup Time of the Flash. At a 40MHz bus clock, maximum value of t_{NVS} is:

$$t_{NVS} = t_{IPBUS} \times (t_{NVSL} + 1)$$
$$t_{NVS}\ \text{Max} = 25\text{ns} \times 2048 = 51.2\mu\text{s}$$
$$t_{NVS}\ \text{Reset} = 25\text{ns} \times 256 = 6.4\mu\text{s}$$

6.11.13 Flash T_{PGS} Limit Register (FIU_TPGSL)

This register is reset during any system reset. This register may *only* be changed when the BUSY bit is clear. IS[11] is asserted should this occur.

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

Figure 6-20. Flash T_{PGS} Limit Register (FIU_TPGSL)

See Programmer's Sheet on Appendix page B-37

6.11.13.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.13.2 Timer Program Setup Limit (TPGSL)—Bits 11–0

Timeout value for the t_{PGS} counter is in terms of bus clocks. The counter controls the NVSTR to program setup time of the Flash. The program setup times based on a 40MHz IP_BUS frequency are:

$$t_{PGS} = t_{IPBUS} \times (t_{PGSL} + 1)$$
$$t_{PGVS}\ \text{Max} = 25\text{ns} \times 4096 = 102.4\mu\text{s}$$
$$t_{PGS}\ \text{Reset} = 25\text{ns} \times 512 = 12.8\mu\text{s}$$

6.11.14 Flash T_{PROG} Limit Register (FIU_TPROGL)

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

Figure 6-21. Flash T_{PROG} Limit Register (FIU_TPROGL)

See Programmer's Sheet on Appendix page B-38

6.11.14.1 Reserved—Bits 15–14

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.14.2 Timer Program Limit (TPROGL)—Bits 13–0

Timeout value for the t_{PROG} counter is in terms of bus clocks. It controls the program time of the Flash. At a 40MHz bus clock, the maximum value of t_{PROG} is:

$$t_{PROG} = t_{IPBUS} \times (t_{PROG} + 1)$$
$$t_{PROG}\ \text{Max} = 25\text{ns} \times 16384 = 409.64\mu\text{s}$$
$$t_{PROG}\ \text{Reset} = 25\text{ns} \times 1024 = 25.6\mu\text{s}$$

6.11.15 Flash T_{NVH} Limit Register (FIU_TNVHL)

This register is reset during any system reset. The register may *only* be changed when the BUSY bit is clear. IS[11] is asserted should this occur.

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

Figure 6-22. Flash T_{NVH} Limit Register (FIU_TNVHL)

See Programmer's Sheet on Appendix page B-39

6.11.15.1 Reserved—Bits 15–11

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.15.2 Timer Non-Volatile Hold Limit (TNVHL)—Bits 10–0

Timeout value for the T_{NVHHL} counter during page erase or program is in terms of bus clocks. This counter controls the NVSTR Hold Time of the Flash. At a 40MHz bus clock, the maximum value of t_{NVH} is:

$$t_{NVH} = t_{IPBUS} \times (t_{NVHL} + 1)$$
$$t_{NVH}\ \text{Max} = 25\text{ns} \times 2048 = 51.24\mu\text{s}$$
$$t_{NVH}\ \text{Reset} = 25\text{ns} \times 256 = 6.4\mu\text{s}$$

The counter controlled by this register is used for Program and Page Erase mode *only*.

6.11.16 Flash T_{NVH1} Limit Register (FIU_TNVH1L)

This register is reset during any system reset and may *only* be changed when the BUSY bit is clear. IS[11] is asserted should this occur.

FIU_BASE+$F	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	TNVH1L														
Write																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 6-23. Flash T_{NVH1} Limit Register (FIU_TNVH1L)

See Programmer's Sheet on Appendix page B-40

6.11.16.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.16.2 Timer Non-Volatile Hold 1 Limit (TNVH1L)—Bits 14–0

Timeout value for the t_{NVH1} counter during mass erase in terms of bus clocks. The counter controls the NVSTR Hold Time of the Flash. The non-volatile hold 1 times based on a 40MHz IPBus frequency are:

$$t_{NVH1} = t_{IPBUS} \times (t_{NVH1L} + 1)$$
$$t_{NVH1}\text{Max} = 25\text{ns} \times 32768 = 819.2\mu\text{s}$$
$$t_{NVH1}\text{Reset} = 25\text{ns} \times 4096 = 102.4\mu\text{s}$$

The counter controlled by this register is used for Mass Erase mode *only*.

6.11.17 Flash T_{RCV} Limit Register (FIU_TRCVL)

This register is reset during any system reset. This register may *only* be changed when the BUSY bit is clear. IS[11] is asserted should this occur.

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

Figure 6-24. Flash T_{RCV} Limit Register (FIU_TRCVL)

See Programmer's Sheet on Appendix page B-41

6.11.17.1 Reserved—Bits 15–9

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

6.11.17.2 Timer Recovery Limit (TRCVL)—Bits 8–0

Timeout value for the t_{RCV} counter in terms of bus clocks. This counter controls the recovery time of the Flash. The recovery times based on a 40MHz IPBus frequency are:

$$t_{RCV} = t_{IPBUS} \times (t_{RCVL} + 1)$$
$$t_{RCV}\text{ Max} = 25\text{ns} \times 512 = 12.8\mu\text{s}$$
$$t_{RCV}\text{ Reset} = 25\text{ns} \times 64 = 1.6\mu\text{s}$$

The counter controlled by this register is used for both Page and Mass Erase modes.

6.11.18 Flash Interface Unit Timeout Registers

Timeout Limit registers default to the correct values of 40MHz bus operation and will require reprogramming of other bus frequencies. If programming is required, it would occur during the power-up sequence or as an integral part of the programming algorithm. Program/erase times and their registers should never require reprogramming by an end user.

Timeout Limit registers are not allowed to be written while BUSY is asserted. IS[11] is asserted should this occur. IS[3] through IS[9] will be set when the various timers
reach the values specified in the Timeout Limit registers.

6.12 Reset

A reset asserted for any reason should switch the Flash into the Standby mode. If a program/erase is interrupted in progress by a reset, data within the enabled area of the Flash will be indeterminate.

6.13 Interrupts

The FIU OR's all bits in the Interrupt Pending (FIU_IP) register to determine the value of the single interrupt signal presented to the host processor interrupt controller.

The FIU_IP register is equal to the bit wise *and*ing operation of the Interrupt Enable (FIU_IE) register and the Interrupt Source (FIU_IS) register. Interrupt sources may be polled at any time in the FIU_IS register, even when those sources are disabled. The FIU_IP register reflects only those sources causing an interrupt to the host processor.

- Interrupts must be cleared by clearing the appropriate bit in the FIU_IS register. This is achieved by the core under software control.
- The FIU is responsible for generating read out-of-range interrupts when the Information Block is enabled. But an attempt must also be made to access past the two rows of that particular block, but within the overall size of the Main Block. The FIU is not responsible for generating interrupts for accesses outside of this range.
- If the host processor attempts a read or write to the Flash while it is being erased or programmed, the FIU will generate an interrupt. It will not affect the contents of the Limit register in question if the corresponding enable bit in FIU_IE is set.
- Writes to the FIU_CNTL register are prohibited once the Intelligent Program or Intelligent Erase sequences are begun by the FIU state machine. It is illegal to change these settings until after the program/erase task is complete and the BUSY bit is clear. Any attempt to do so will not affect the register contents, and it may cause an error interrupt to be posted if the corresponding enable bit in FIU_IE is set.

Chapter 7
External Memory Interface (EMI)

7.1 Introduction

Hybrid controllers 56F826 and 56F827 provide a port for external memory interfacing. This chapter details the pins and programming specifics for an External Memory Interface (EMI), also known as Port A. This port provides 16 pins for an external address bus, 16 pins for an external data bus, and four pins for bus control. Together, these 36 pins comprise Port A.

7.2 External Memory Port Architecture

Figure 7-1 illustrates the general Block Diagram of the 56F826/827 input/output.

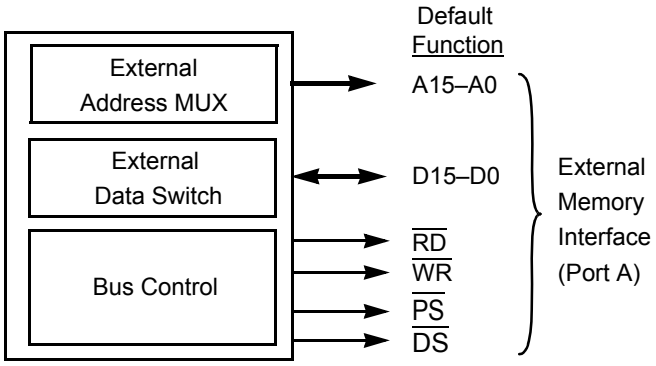


Figure 7-1. 56F826/827 Input/Output Block Diagram

7.3 Pin Descriptions

The names of the 56F826/827 External Memory port pins are as follows:

- Address Bus Output (A15–0)
- Data Bus (D15–0)
- Read Enable ($\overline{RD}$)
- Write Enable ($\overline{WR}$)
- Program Memory Select ($\overline{PS}$)
- Data Memory Select ($\overline{DS}$)

7.4 Register Description

The External Memory Interface (EMI), also referred to as Port A, is the port through which all accesses to external memories and external memory-mapped peripherals are made. This port contains a 16-bit address bus, a 16-bit data bus, and four-bus control pins for strobes. The EMI uses one programmable register for bus control, the Bus Control Register (BCR).

Software-controlled wait states can be introduced when accessing slower memories, or peripherals. Wait states are programmable using the BCR register. **Figure 7-3** and **Figure 7-4** illustrate bus cycles with and without wait states.

Table 7-1. EMI Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + $9	BCR	Bus Control Register	Read/Write	Section 7.4.1

7.4.1 Bus Control Register (BCR)

The BCR, located at X:$FFF9, is a 16-bit, read/write register used for inserting software wait states on accesses to external program or data memory. Two, 4-bit wait state fields are provided, each capable of specifying from 0, 4, 8, or 12-wait states. On processor reset, each wait state field is set to $C so 12-wait states are inserted, allowing slower memory to be used immediately after reset. Bits 0, 1, 4, and 5 are ignored in determining the wait states. Therefore, writing values other than 0, 4, 8, and 12 will result in the nearest lower count wait state.

BCR—$FFF9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	DRV	0	WAITSTATE FIELD for EXTERNAL WSX[4:7]				WAIT STATE FIELD for EXTERNAL WSP[0:3]			
Write																
Reset	0	0	0	0	0	0	0	7	1	1	0	0	1	1	0	0

Figure 7-2. Bus Control Register (BCR)

See Programmer's Sheet on Appendix page B-22

7.4.1.1 Reserved—Bits 15–10

This bit field is reserved or not implemented. It is read and written using 0.

7.4.1.2 Drive (DRV)—Bit 9

The Drive (DRV) control bit is used to specify what occurs on the external memory port pins when no external access is performed—whether the pins remain driven or are placed in tri-state. Accessing external memory when DRV = 0 may cause unintentional results. Therefore, it is recommended DRV = 1 be defined by the user. Please see **Section 7.4.2.**

7.4.1.3 Reserved—Bit 8

This bit is reserved or not implemented. It is read and written using 0.

7.4.1.4 Wait State X Data Memory (WSX)—Bits 7–4

The wait state X data memory (WSX[3:0]) control bits allow programming of the wait states of external X data memory. **Table 7-2** illustrates the wait states provided with these bits. The WSX[3:0] and the WSP[3:0] bits are programmed in the same fashion but do not need to be set to the same value.

Table 7-2. Programming WSP[3:0] and WSX[3:0] Bits for Wait States

Bit String	Hex Value	Number of Wait States
0000	$0	0
0100	$4	4
1000	$8	8
1100	$C	12

7.4.1.5 Wait State P Program Memory (WSP)—Bits 3–0

The wait state P program memory (WSP[3:0]) control bits allow for programming of the wait states for external program memory. These bits are programmed and provided in **Table 7-2.**

Note: Bits zero and one in the 4-bit string of **Table 7-2** for both WSP[3:0] and WSX[3:0] are ignored in evaluating the number of wait states to be inserted. That is, writing a value of seven (0111) will result in only four (0100) four wait states.

Figure 7-3 illustrates an example of the bus cycles without wait states. This timing relationship will not function ideally at an 80MHz frequency, but it will function according to the figure at lower frequencies. The critical factor is the data must have stabilized before the beginning of the T3 period, demonstrated by the dark gray area in **Figure 7-3.**

Figure 7-4 exhibits an example of the bus cycles with wait states. For more information on wait states, see the corresponding chip's technical data sheet.

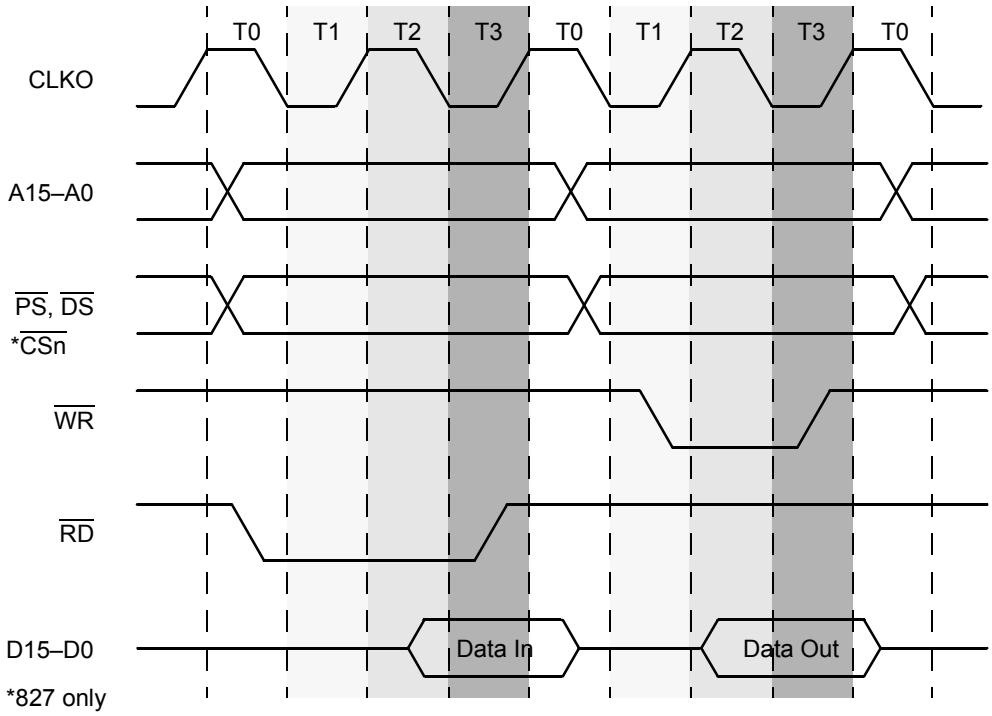


Figure 7-3. Bus Operation (Read/Write–Zero Wait States)

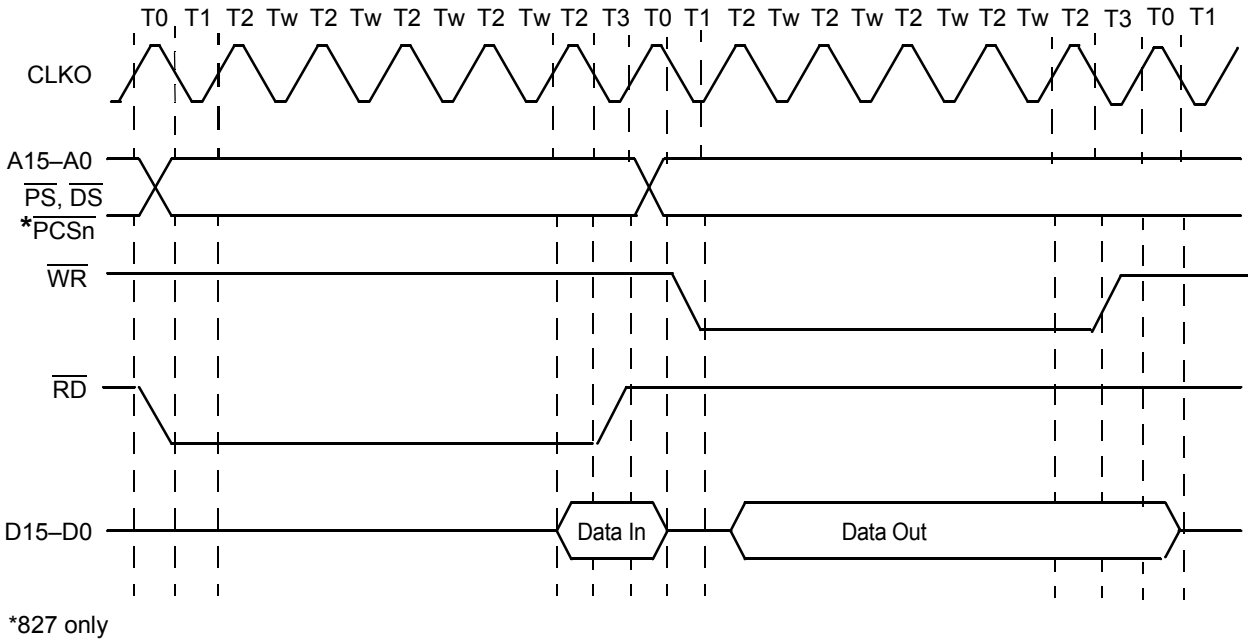


Figure 7-4. Bus Operation (Read/Write–Four Wait States)

7.4.2 State of Pins in Different Processing States

The DSP56800 core can be in one of six processing states, also called modes:

1. Normal
2. Exception
3. Reset
4. Wait
5. Stop
6. Debug

In Normal mode, each instruction cycle has two possible cases—either the processor needs to perform an external access, or all accesses are accomplished internally. Exception processing is similar. In the case of an external access, the Address and bus control pins are driven to perform a normal bus cycle. The data bus is also driven if the access is in a write cycle.

In the case where there is no external access on a particular instruction cycle, one of two things may occur:

1. The external address bus and control pins can remain driven with their previous values.
2. They can be tri-stated.

Note: The pull-ups for the address, data, and control pins are normally enabled by the state of the DRV bit. The data, control, and address [5:0] pull-ups may be disabled by setting the appropriate bits in the SYS_CNTL register in the reset wait module. The pull-ups for address bits [15:8] are controlled by the GPIOA PUR register. The pull-ups for address bits [7:6] are controlled by the GPIOE PUR register bits[3:2].

Reset mode *always* tri-states the external address bus and internally pulls control pins high. Stop and Wait modes, however, either tri-state the address bus and control pins or let them remain driven with their previous values. This selection is made based on the value of the DRV bit in the BCR. **Table 7-3** and **Table 7-4** describe the operation of the external memory port in these different modes. Debug mode is a special state used to debug and test programming code. This state is detailed in **Section 9** of the *DSP56800 Family Manual,* DSP56800FM. The state is not described in the following tables.

During a normal mode, external access, the data lines are driven only during an external write cycle.

Table 7-3. Port A and PCS Operation with DRV Bit = 0

Mode	Pins			
	A0-A15	$\overline{PS}$, $\overline{DS}$, $\overline{RD}$, $\overline{WR}$	$\overline{PCSn}$ (827 only)	D0-D15
Normal Mode, External Access	Driven	Driven	Driven	Driven
Normal Mode, Internal Access	Tri-stated	Tri-stated	Tri-stated	Tri-stated
Stop Mode	Tri-stated	Tri-stated	Tri-stated	Tri-stated
Wait Mode	Tri-stated	Tri-stated	Tri-stated	Tri-stated
Reset Mode	Tri-stated	Pulled High Internally	Pulled High Internally	Tri-stated

Table 7-4. Port A and PCS Operation with DRV Bit = 1

Mode	Pins			
	A0-A15	$\overline{PS}$, $\overline{DS}$, $\overline{RD}$, $\overline{WR}$	$\overline{PCSn}$ (827 only)	D0-D15
Normal Mode, External Access	Driven	Driven	Driven	Driven
Normal Mode, Internal Access	Driven	Driven	Tri-stated	Tri-stated
Stop Mode	Driven	Driven	Tri-stated	Tri-stated
Wait Mode	Driven	Driven	Tri-stated	Tri-stated
Reset Mode	Driven	Pulled High Internally	Pulled High Internally	Tri-stated

During a Normal mode external access, data lines are driven *only* during an external write cycle.

7.5 Programmable Chip-Select (56F827 only)

Typical systems require some external hardware to provide chip-selects for external peripherals. Many of these devices do not have all the signals required for an asynchronous bus interface. Additionally, higher levels of integration can provide cost, speed, and reliability advantages over external logic. The programmable chip-selects described here provide the means to minimize this external logic. The primary function of the chip-selects is to provide the chip enables for external memory and peripheral devices. There are up to eight programmable chip-select signals available. All chip-select pins are software programmable.

7.6 Chip Select Features (56F827 Only)

- Reduced system complexity
  - No external glue logic required for typical systems, if the chip-selects are used.
- Eight programmable active-low chip-selects ($\overline{PCS7}$ - $\overline{PCS0}$)
- Control for external boot device

  - — $\overline{\text{PCS0}}$ is assigned as both read/write program memory of size 64K upon reset (PS).
  - — $\overline{\text{PCS1}}$ is assigned as both read/write data memory of size 64K upon reset ($\overline{\text{DS}}$).
- Programmable base addresses with programmable block sizes
- Maximum block size = 64K (16-bit) words
- Minimum block size = 0.5K (16-bit) words
- Wait states programmable through BCR register of 56800 core. Changing number of wait states effects all chip-selects together.
- Each chip-select can be assigned either program memory or data memory, or both.
- All chip-selects are assigned for both read/write.
- Each chip-select can be individually enabled or disabled.

7.7 Programmability

7.7.1 Default Function of PCS0 and PCS1 (56F827 Only)

$\overline{\text{PCS0}}$ and $\overline{\text{PCS1}}$ are configured as $\overline{\text{PS}}$ and $\overline{\text{DS}}$ respectively upon reset giving access to the full 64K of Program and Data memories respectively. This default function is achieved by setting the reset values of the corresponding PCSBAR and PCSOR registers. The remaining chip-selects $\overline{\text{PCS2}}$ to $\overline{\text{PCS7}}$ are disabled at reset.

This provides compatibility to the other 56800 Family parts by using $\overline{\text{PCS0}}$ and $\overline{\text{PCS1}}$ as $\overline{\text{PS}}$ and $\overline{\text{DS}}$ respectively. All individual chip-selects including $\overline{\text{PCS0}}$ and $\overline{\text{PCS1}}$ are programmable through Base Address registers PCSBARn and Option registers PCSORn.

7.8 Register Definitions

Table 7-5. EMI Memory Map

Device	Peripheral	Address
827 Only	PCS_BAR0	$1190
	PSC_BAR1	$1191
	PSC_BAR2	$1192
	PSC_BAR3	$1193
	PSC_BAR4	$1194
	PSC_BAR5	$1195
	PSC_BAR6	$1196
	PSC_BAR7	$1197
	PCS_OR0	$1198
	PCS_OR1	$1199
	PCS_OR2	$119A
	PCS_OR3	$119B
	PCS_OR4	$119C
	PCS_OR5	$119D
	PCS_OR6	$119E
	PCS_OR7	$119F

The PCS occupies 16 words in the memory space starting at PCS_BASE. All PCS registers are implemented as 16-bit registers.

Individual detailed register descriptions associated with the PCS and their relative offset from the base address follow. Reserved bits within a register always read as zero. Reserved functions (i.e., Read/Write) are indicated by shading the bit.

For each programmable chip-select output supported by the PCS, the following attributes may be assigned through the use of register variables.

Note: Register Address = Base Address + Address Offset, where the Base Address is defined at chip level and the Address Offset is defined at the module level.

Table 7-6. PCS Registers Summary

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + $0	PCSBAR0	Base Address Register 0	Read/Write	Section 7.8.1
Base + $1	PCSBAR1	Base Address Register 1	Read/Write	
Base + $2	PCSBAR2	Base Address Register 2	Read/Write	
Base + $3	PCSBAR3	Base Address Register 3	Read/Write	
Base + $4	PCSBAR4	Base Address Register 4	Read/Write	
Base + $5	PCSBAR5	Base Address Register 5	Read/Write	
Base + $6	PCSBAR6	Base Address Register 6	Read/Write	
Base + $7	PCSBAR7	Base Address Register 7	Read/Write	
Base + $8	PCSOR0	PCS Option Register 0	Read/Write	Section 7.8.2
Base + $9	PCSOR1	PCS Option Register 1	Read/Write	
Base + $A	PCSOR2	PCS Option Register 2	Read/Write	
Base + $B	PCSOR3	PCS Option Register 3	Read/Write	
Base + $C	PCSOR4	PCS Option Register 4	Read/Write	
Base + $D	PCSOR5	PCS Option Register 5	Read/Write	
Base + $E	PCSOR6	PCS Option Register 6	Read/Write	
Base + $F	PCSOR7	PCS Option Register 7	Read/Write	

Bit fields of each of the 16 registers are summarized in **Figure 7-5.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	PCSBAR0	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$1	PCSBAR1	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$2	PCSBAR2	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$3	PCSBAR3	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$4	PCSBAR4	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$5	PCSBAR5	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$6	PCSBAR6	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$7	PCSBAR7	R	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
		W																
$8	PCSOR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																
$9	PCSOR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																
$A	PCSOR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																
$B	PCSOR3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																
$C	PCSOR4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																
$D	PCSOR5	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																
$E	PCSOR6	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																
$F	PCSOR7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDSEN	
		W																

R	0	Read as 0
W		Reserved

Figure 7-5. PCS Registers Map Summary

7.8.1 PCS Base Address Registers (PCSBAR0, ... ,PCSBAR7)

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

Figure 7-6. PCS Base Address Register 0 (PCSBAR0)

See Programmer's Sheet on Appendix page B - 74

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

Figure 7-7. PCS Base Address Register 1 (PCSBAR1)

See Programmer's Sheet on Appendix page B - 74

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

Figure 7-8. PCS Base Address Register 2 (PCSBAR2)

See Programmer's Sheet on Appendix page B - 74

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

Figure 7-9. PCS Base Address Register 3 (PCSBAR3)

See Programmer's Sheet on Appendix page B - 74

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

Figure 7-10. PCS Base Address Register 4 (PCSBAR4)

See Programmer's Sheet on Appendix page B - 74

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

Figure 7-11. PCS Base Address Register 5 (PCSBAR5)

See Programmer's Sheet on Appendix page B - 74

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

Figure 7-12. PCS Base Address Register 6 (PCSBAR6)

See Programmer's Sheet on Appendix page B - 74

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

Figure 7-13. PCS Base Address Register 7 (PCSBAR7)

See Programmer's Sheet on Appendix page B - 74

Each chip-select has an associated base address register. A base address is the lowest address in the block of addresses enabled by a chip-select. Block size is the extent of the address block above the base address. Block size is determined by the value contained in BLKSZ[2:0].

BLKSZ[2:0] determines which bits in the base address field are compared to corresponding bits on the Address Bus during an access. Provided other constraints determined by the register fields in corresponding PCSORn register are also satisfied when a match occurs, the associated chip-select signal is asserted. **Table 7-7** details BLKSZ[2:0] encoding.

Table 7-7. PCSBAR Encoding of the BLKSZ Field

BLKSZ[2:0]	Block Size	Address Lines Compared	Example Base Address	Example Address Range
000	0.5K	ADDR[15:9]	$0000	$0000 - $01FF
001	1K	ADDR[15:10]	$0400	$0400 - $07FF
010	2K	ADDR[15:11]	$0800	$0800 - $0FFF
011	4K	ADDR[15:12]	$1000	$1000 - $1FFF
100	8K	ADDR[15:13]	$2000	$2000 - $3FFF
101	16K	ADDR[15:14]	$4000	$4000 - $7FFF
110	32K	ADDR[15:15]	$8000	$8000 - $FFFF
111	64K	NONE	$0000	$0000 - $FFFF

The Chip-Select Address Compare Logic uses only the Most Significant Bits to match an address within a block. The value of the base address must be an integer multiple of the block size. For example, the base address must be at the block size boundary only. If the base address is not at

the block size boundary, chip-select will not become enabled for the entire block size. This restriction arises from because for chip-select generation, only the higher order address lines are compared as given in **Table 7-8.** For example, for a block size of 4k, the base address can be $0000, $1000, $2000, $3000, $4000,... etc. For block size of 4k, address lines ADDR[15:12] are compared for chip-select generation and ADDR[11:0] are ignored. The reset value of PCSBAR0 and PCSBAR1 registers is $0007 to enable $\overline{\text{PCS0}}$ and $\overline{\text{PCS1}}$ for the entire 64K external memory range on reset.

7.8.2 PCS Option Registers (PCSOR0, PCSOR1,..., PCSOR7)

For every chip-select, a Chip-Select Option Register is required. This register specifies the mode of operation of the chip-select.

$\overline{\text{PCS0}}$ is enabled at reset for program space to optionally select an external boot device. $\overline{\text{PCS1}}$ is enabled at reset for data space.

PCS_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	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 7-14. PCS Option Register 0 (PCSOR0)

See Programmer's Sheet on Appendix page B - 75

PCS_BASE+$9	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	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 7-15. PCS Option Register 1 (PCSOR1)

See Programmer's Sheet on Appendix page B - 75

PCS_BASE+$A	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	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-16. PCS Option Register 2 (PCSOR2)

See Programmer's Sheet on Appendix page B - 75

PCS_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	0	0	0	0	0	0	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-17. PCS Option Register 3 (PCSOR3)

See Programmer's Sheet on Appendix page B - 75

PCS_BASE+$C	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	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-18. PCS Option Register 4 (PCSOR4)

See Programmer's Sheet on Appendix page B - 75

PCS_BASE+$D	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	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-19. PCS Option Register 5 (PCSOR5)

See Programmer's Sheet on Appendix page B - 75

PCS_BASE+$E	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	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-20. PCS Option Register 6 (PCSOR6)

See Programmer's Sheet on Appendix page B - 75

PCS_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	0	0	PDSEN	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-21. PCS Option Register 7 (PCSOR7)

See Programmer's Sheet on Appendix page B - 75

7.8.2.0.1 Reserved (PCSORn)—Bits 15-2

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

7.8.2.1 Program Data Select Encode (PDSEN)—Bits 1–0

The mapping of a Chip-Select to program and/or data space is provided in **Table 7-8.** When set to 11, the Chip-Select will be enabled for both program and data spaces.

Table 7-8. PCSOR Encoding of PCS $\overline{PS}$ / $\overline{DS}$ Functionality

PDSEN[1:0]	Description
00	Disable
01	$\overline{DS}$ Only
10	$\overline{PS}$ Only
11	Both $\overline{DS}$ and $\overline{PS}$

Note: The Programmable Chip-Select $\overline{PCS}$n can be disabled by setting the PDSEN field to zero (00) of the PCSORn.

Chapter 8
General Purpose Input/Output (GPIO)

8.1 Introduction

The General Purpose Input/Output (GPIO) is designed to share package pins with other peripherals on the chip. If a peripheral is not required, the pin may be programmed as an input/output, or level-sensitive interrupt input. GPIOs are placed on the chip in groups of eight or 16 bits. **Figure 8-3** illustrates the logic associated with just one bit of GPIO. **Figure 8-1** and **Figure 8-2** defines the GPIO pins and their respective connections with some peripheral devices.

Note: Ports B and D are dedicated GPIOs, hence no peripherals are associated with these ports.

Dedicated and multiplexed GPIOs:

- 56F826—Has 16 dedicated GPIOs and 30 multiplexed GPIOs on Ports A, C, E, F
- 56F827—Has 16 dedicated GPIOs and 48 multiplexed GPIOs on Ports A, C, F, G

8.2 Features

The GPIO module design includes these characteristics:

- Individual control for each pin to be in either Normal or GPIO mode discussed in **Section 8.3**
- Individual direction control for each pin in GPIO mode
- Individual pull-up enable control for each pin in either Normal or GPIO mode

8.3 Operating Modes

The GPIO module contains two major modes of operation:

1. Normal mode—This can also be thought of as Peripheral Controlled mode. The peripheral module controls the output enable and any output data to the pin and any input data from the pin is passed to the peripheral. Pull-up enables are controlled by a GPIO register.
2. GPIO mode—In this mode the GPIO module controls the output enable to the pin and supplies any data to be output. Also, any input data can be read from a GPIO memory mapped register. Pull-up enables are controlled by a GPIO register. Each GPIO pin can also be configured as an interrupt input.

8.4 Chip Specific Configurations

Dedicated GPIOs are, of course, intended only for use as GPIOs. Shared indicates pins may be alternately used as a GPIO. The programming model is identical for dedicated versus shared GPIO.

Note: Care must be exercised when using a shared GPIO because of possible conflict of the shared pin between the GPIO and the peripheral.

Peripheral Address Maps for the 56F826 are located in **Table 3-11**, while the same information for the 56F827 is located in **Table 3-12.**

Table 8-1. GPIO Assignments

Part	Port A	Port B	Port C	Port D	Port E	Port F	Port G
56F826	8 Shared	8 Dedicated	6 Shared	8 Dedicated	8 Shared	8 Shared	–
56F827	16 Shared	8 Dedicated	8 Shared	8 Dedicated	–	8 Shared	16 Shared

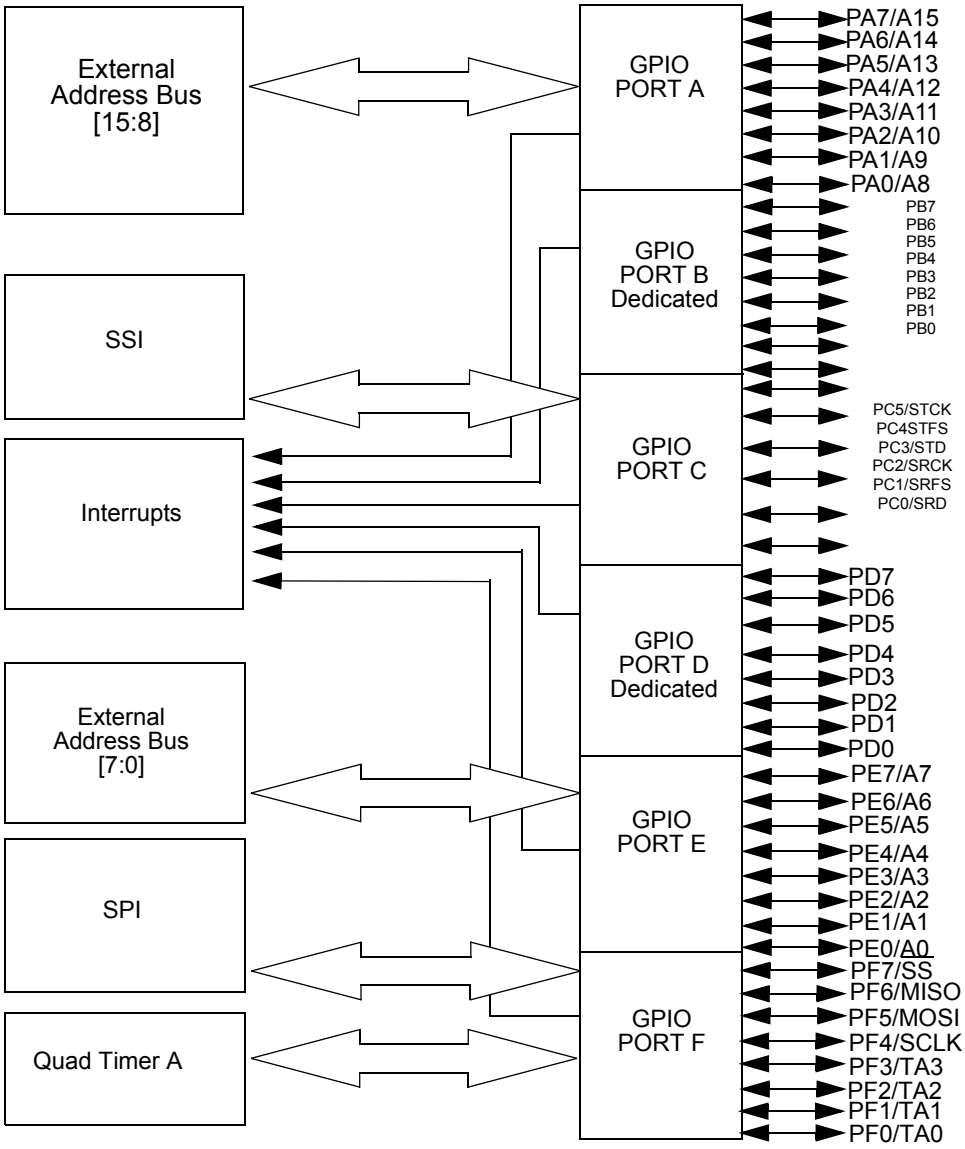


Figure 8-1. Block Diagram Showing GPIO Port Connections for 56F826

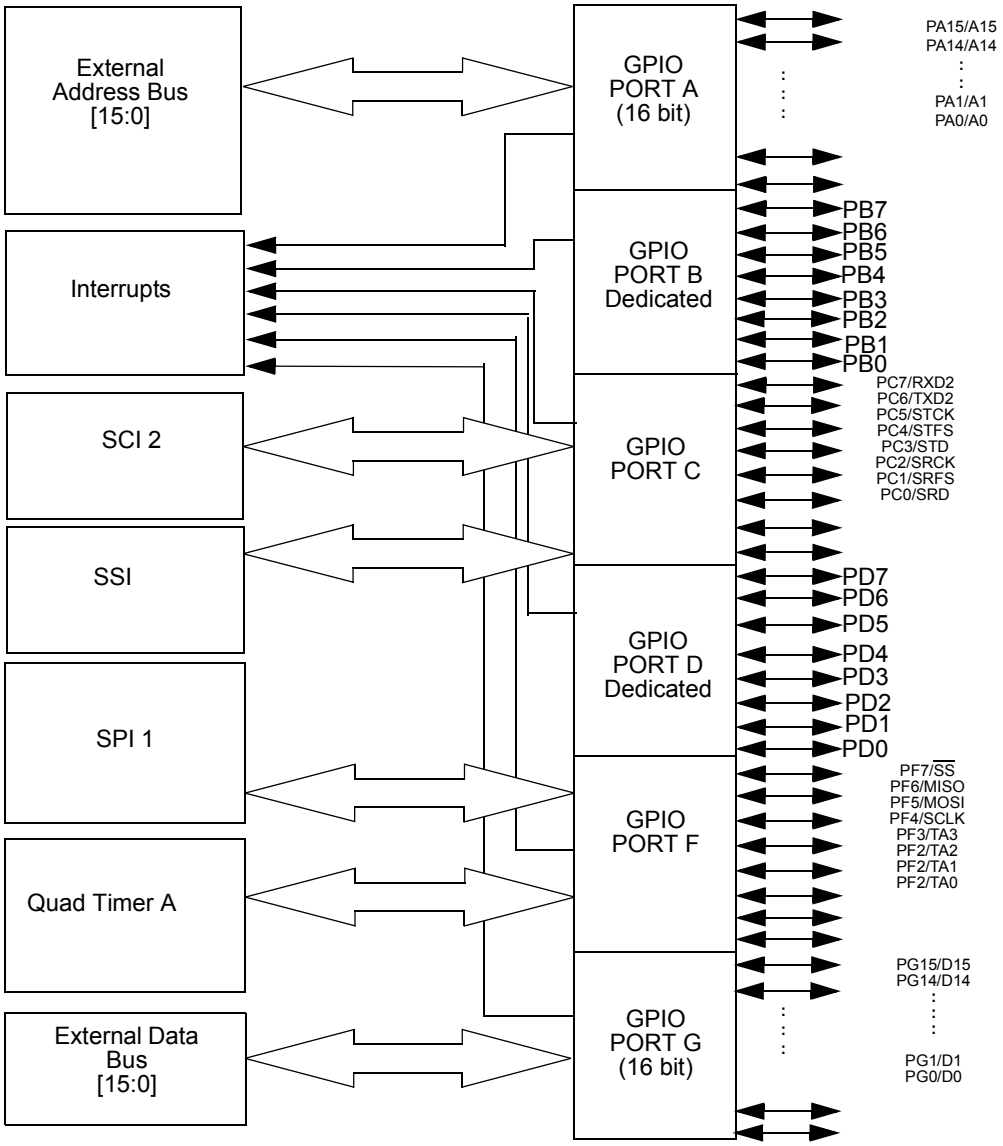


Figure 8-2. Block Diagram Showing GPIO Port Connections for 56F827

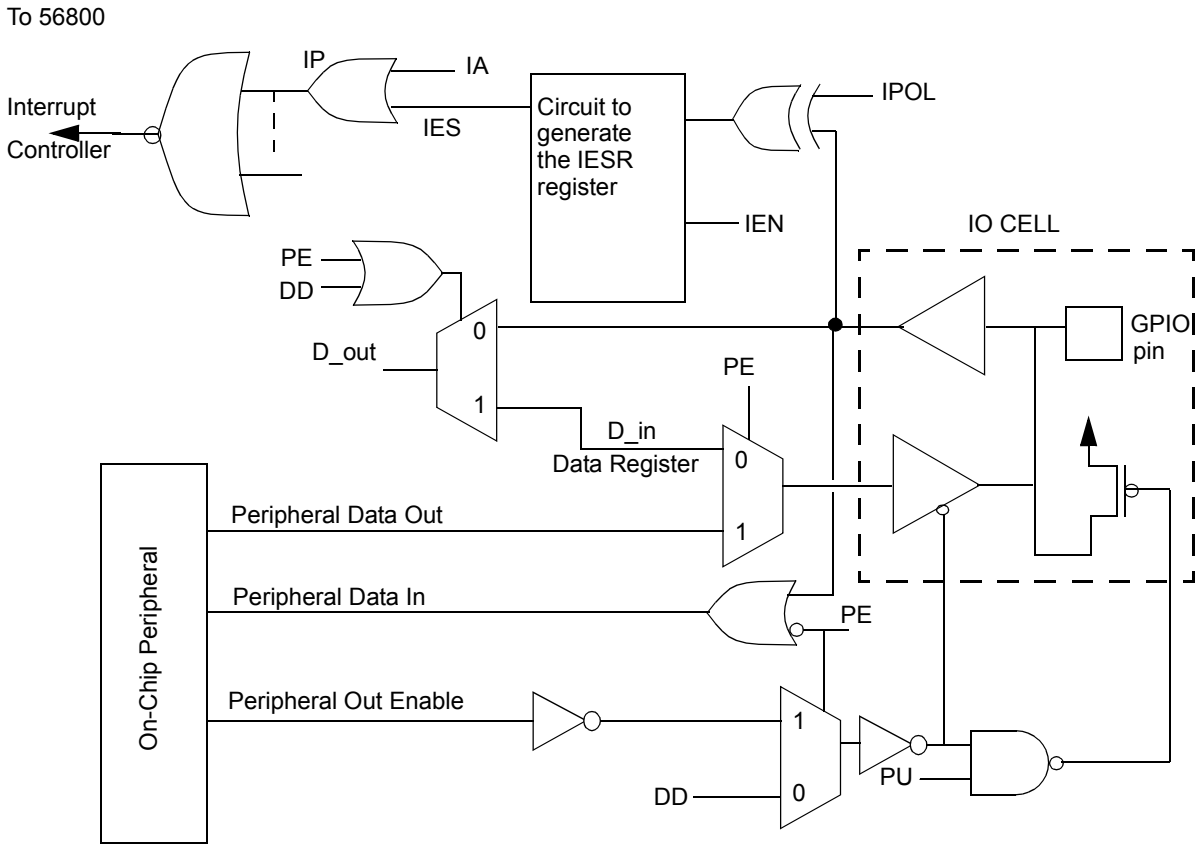


Figure 8-3. Bit-Slice View of the GPIO Logic

Each GPIO pin can be configured in three ways:

1. An input, with or without pull-up
2. An interrupt
3. An output

The 56F826/827 GPIO's pull-up is configured by writing the GPIO Pull-Up Enable Register (PUR) Control register and the Data Direction Register (DDR) when the Peripheral Enable Register (PER) is set to zero. When PER is set to one, the pull-ups are controlled by the PUR register and the direction of the peripheral used. In any case, if the I/O is set to be an output, the pull-up is disabled.

The 56F826/827 GPIO interfaces with the following on-chip devices:

- External Address Bus
- SCI (56F827 only)
- SPI

- SSI
- Timer
- External Data Bus

8.5 GPIO Interrupts

The GPIO has two types of interrupts:

1. Software interrupt for testing purposes
2. Hardware interrupt from the external pin

The software Interrupt Asset Register (IAR) can be tested by writing ones to the IAR register. The General Purpose In/Out Interrupt Pending Register (GPIO_IPR) will record the value of the IAR. The GPIO_IPR can be cleared by writing zeros (56F826) or ones (56F827) to the IAR register during the IAR testing.

Note: When testing the IAR, the Interrupt Polarity Register (IPOLR), Interrupt Edge Sensitive Register (IESR), and Interrupt Enable Register (IENR) must be set to zero to guarantee the interrupt registered in the GPIO_IPR is due to IAR only.

When a GPIO is used as an interrupt, the IAR must be set to zero. When the GPIO pin signal is asserted, polarity will be detected going through the edge detection mechanism. Please refer to **Figure 8-4** for an illustration. The value will be seen at the IESR, and recorded by the GPIO_IPR. The GPIO_IPR can be cleared by writing zeros into the IESR for 56F826. In the case of the 56F827, the GPIO_IPR can be cleared by writing ones into the IESR. If IPOLR is set to zero, the interrupt at the GPIO pin is active high. If IPOLR is set to one, the interrupt at the GPIO pin is active low.

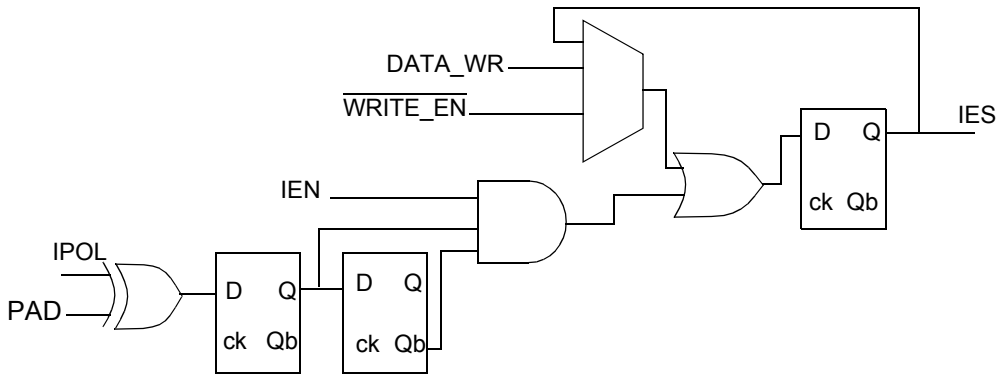


Figure 8-4. Edge Detector Circuit

The eight (or 16 in GPIO port A and G on the 56F827) interrupt signals are read together to present only a single interrupt to the core. The interrupt service routine must then check the contents of the interrupt pending register, determining which pin(s) caused the interrupt.

External interrupt sources do not need to remain asserted due to the edge sensitive nature of the detection mechanism.

Table 8-2. GPIO Interrupt Assert Functionality

IPOLR	Interrupt Asserted	Remark
0	High	If the IENR is set to one, as the pin goes to high, an interrupt will be recorded by the GPIO_IPR register.
1	Low	If the IENR is set to one, as the pin goes to low, an interrupt will be recorded by the GPIO_IPR register.

8.6 Register Summary

Each GPIO module has nine, 8-bit registers or 16-bit registers. They are provided in **Table 8-3.**

Table 8-3. GPIO Registers With Reset Values

Register	Description	Binary Reset State for Lower 8 bits	Remark
PUR	Pull-up Enable Register	$11111111	Pull-ups are enabled. (Please see **Table 8-4.**)
DR	Data Register	$00000000	DR is used for data interface between the pin and the IP Bus.
DDR	Data Direction Register	$00000000	GPIO is set to an input. If DDR is one the GPIO becomes an output.
PER	Peripheral Enable Register	$11111111	Peripheral controls the GPIO. The DDR register does not determine the direction of the I/O.
IAR	Interrupt Assert Register	$00000000	No interrupt.
IENR	Interrupt Enable Register	$00000000	Interrupt is disabled.
IPOLR	Interrupt Polarity Register	$00000000	When set to one, the interrupts are active low. When set to zero, interrupts are active high.
GPIO_IPR	Interrupt Pending Register	$00000000	No interrupt is registered. A one indicates an interrupt.
IESR	Interrupt Edge Sensitive Register	$00000000	A one indicates an edge has been detected.

Table 8-4 illustrates the state of the PAD and pull-up resistor.

Table 8-4. GPIO Pull-Up Enable Functionality

Peripheral Out Enable	PER	PUR	DDR	PAD State	PAD Pull-Up
x	0	0	0	Input	Disabled
x	0	0	1	Output	Disabled
x	0	1	0	Input	Enabled
x	0	1	1	Output	Disabled
0	1	x	x	Output	Disabled
0	1	x	x	Output	Disabled
1	1	x	x	Input	Disabled
1	1	x	x	Input	Enabled

x = Don't Care

When the Peripheral Enable Register (PER) is zero, the Pull-Up Enable Register (PUR) and the Data Direction Register (DDR) control the pin pull-up. When the PER is one, the pin pull-up is controlled by the PUR register and peripheral output enable. The PUR value is affected when the pin is configured as an input only.

The Data Register (DR) is used to pass data to the GPIO pin or the IPBus. It is also used to store data from the GPIO pin and the IPBus.

8.7 GPIO Programming Algorithms

Example 8-1. Algorithm for Port B Pins as Inputs Used for Receiving Data in the 56F826

```
• INPUT for Port B
  – START         EQU     $0080       ;Start of program
  – PBPUR         EQU     $11B0       ;Port B pull-up register
  – PBDR          EQU     $11B1       ;Port B data register
  – PBDDR         EQU     $11B2       ;Port B data direction register
  – PBPER         EQU     $11B3       ;Port B peripheral register
  – PBIAR         EQU     $11B4       ;Port B interrupt assert register
  – PBIENR        EQU     $11B5       ;Port B interrupt enable register
  – PBIPOLR       EQU     $11B6       ;Port B interrupt polarity register
  – PBIPR         EQU     $11B7       ;Port B interrupt pending register
  – PBIESR        EQU     $11B8       ;Port B interrupt edge sensitive register
  – data_i        EQU     $0001       ;data input
• Vector Setup
  – ORG           P:$0000             ;Cold Boot
  – JMP           START               ;Hardware RESET vector
  – ORG           P:START             ;Start of program
• General Setup
  – MOVE          #$0200,X:BCR        ;External memory has 0 states
• Port B Setup
```

```
    — MOVE          #$0000,X:PBIAR      ;Disable Port B interrupts
    — MOVE          #$0000,X:PBIENR     ;Disable Port B interrupts
    — MOVE          #$0000,X:PBIPOLR    ;Disable Port B interrupts
    — MOVE          #$0000,X:PBIESR     ;Disable Port B interrupts
    — MOVE          #$0000,X:PBPER      ;Configures Port B pins as dedicated GPIOs
    — MOVE          #$0000,X:PBDDR      ;Selects Port B pins as inputs (default)
•   Main Routine
    — INPUT                             ;Input Loop
    — MOVE          X:PBDR,X0           ;Read PB0-PB7 into bits 0-7 of data_i
    — MOVE          X0,X:data_i         ;Memory to memory move requires 2 moves
    — BRA           INPUT
```

Example 8-2. Algorithm for Port B Pins as Outputs Used for Sending Data

```
•   INPUT for Port B
    — START         EQU     $0080       ;Start of program
    — PBPUR         EQU     $0FC0       ;Port B pull-up register
    — PBDR          EQU     $0FC1       ;Port B data register
    — PBDDR         EQU     $0FC2       ;Port B data direction register
    — PBPER         EQU     $0FC3       ;Port B peripheral register
    — PBIAR         EQU     $0FC4       ;Port B interrupt assert register
    — PBIENR        EQU     $0FC5       ;Port B interrupt enable register
    — PBIPOLR       EQU     $0FC6       ;Port B interrupt polarity register
    — PBIPR         EQU     $0FC7       ;Port B interrupt pending register
    — PBIESR        EQU     $0FC8       ;Port B interrupt edge sensitive register
    — data_o        EQU     $0001       ;data input
•   Vector Setup
    — ORG           P:$0000             ;Cold Boot
    — JMP           START               ;Hardware RESET vector
    — ORG           P:START             ;Start of program
•   General Setup
    — MOVE          #$0200,X:BCR        ;External memory has 0 states
•   Port B Setup
    — MOVE          #$0000,X:PBIAR      ;Disable Port B interrupts
    — MOVE          #$0000,X:PBIENR     ;Disable Port B interrupts
    — MOVE          #$0000,X:PBIPOLR    ;Disable Port B interrupts
    — MOVE          #$0000,X:PBIESR     ;Disable Port B interrupts
    — MOVE          #$0000,X:PBPER      ;Configures Port B pins as dedicated GPIOs
    — MOVE          #$FFFF,X:PBDDR      ;Selects Port B pins as outputs
•   Main Routine
    — OUTPUT                            ;Output Loop
    — MOVE          X:data_o,X0         ;Put bits 0-7 of "data_o" on pins PB0-PB7
    — MOVE          X0,X:PBDR           ;Memory to memory move requires 2 moves
    — BRA           OUTPUT
```

8.8 Register Definitions

Each GPIO register contains eight bits (or 16 bits in GPIO port A and G on the 56F827), each performing an identical function for one of the eight or 16 GPIO pins controlled by its port. 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 is different for each port. The address offset is defined at the module level. Please refer to **Table 3-11** for the Data Memory Peripheral Address Map.

Table 8-5. GPIO Memory Map

Device	Name	Base Address
826/827	GPIOA_BASE	$11A0
	GPIOB_BASE	$11B0
	GPIOC_BASE	$11C0
	GPIOD_BASE	$11D0
	GPIOE_BASE	$11E0
	GPIOF_BASE	$11F0

Table 8-6. GPIO Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + $0	GPIOX_PUR	Pull-Up Enable Register	Read/Write	Section 8.8.1
Base + $1	GPIOX_DR	Data Register	Read/Write	Section 8.8.2
Base + $2	GPIOX_DDR	Data Direction Register	Read/Write	Section 8.8.3
Base + $3	GPIOX_PER	Peripheral Enable Register	Read/Write	Section 8.8.4
Base + $4	GPIOX_IAR	Interrupt Assert Register	Read/Write	Section 8.8.5
Base + $5	GPIOX_IENR	Interrupt Enable Register	Read/Write	Section 8.8.6
Base + $6	GPIOX_IPOLR	Interrupt Polarity Register	Read/Write	Section 8.8.7
Base + $7	GPIOX_IPR	Interrupt Pending Register	Read/Write	Section 8.8.8
Base + $8	GPIOX_IESR	Interrupt Edge-Sensitive Register	Read/Write	Section 8.8.9

Each GPIO module has nine, 8-bit registers provided in **Figure 8-5.**

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	PUR (826)	R	0	0	0	0	0	0	0	0	PUR							
		W																
$0	PUR (827)	R	PUR															
		W																
$1	DR-(826)	R	0	0	0	0	0	0	0	0	DR							
		W																
$1	DR-(827)	R	DR															
		W																
$2	DDR (826)	R	0	0	0	0	0	0	0	0	DDR							
		W																
$2	DDR (827)	R	DDR															
		W																
$3	PER (826)	R	0	0	0	0	0	0	0	0	PER							
		W																
$3	PER (827)	R	PER															
		W																
$4	IAR (826)	R	0	0	0	0	0	0	0	0	IAR							
		W																
$4	IAR (827)	R	IAR															
		W																
$5	IENR (826)	R	0	0	0	0	0	0	0	0	IENR							
		W																
$5	IENR (827)	R	IENR															
		W																
$6	IPOLR (826)	R	0	0	0	0	0	0	0	0	IPOLR							
		W																
$6	IPOLR (827)	R	IPOLR															
		W																
$7	IPR (826)	R	0	0	0	0	0	0	0	0	IPR							
		W																
$7	IPR (827)	R	IPR															
		W																
$8	IESR (826)	R	0	0	0	0	0	0	0	0	IESR							
		W																
$8	IESR (827)	R	IESR															
		W																

R	0	Read as 0
W		Reserved

Figure 8-5. GPIO Register Map Summary

8.8.1 GPIO Pull-Up Enable Register (PUR)

Bits within this read/write register control whether pull-ups are enabled in either Normal or GPIO mode. Pull-ups are automatically disabled for outputs in both modes.

- 1 = Pull-ups enabled for inputs by default
- 0 = Pull-ups disabled for inputs

GPIO_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	PU							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 8-6. Pull-Up Enable Register (PUR)

See Programmer's Sheets on Appendix page B-42

GPIO_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 8-7. Pull-Up Enable Register (PUR) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-42

8.8.2 Data Register (DR)

The value of the data on the I/O pin can be read by reading Data Register (DR) if this I/O pin is set as an input in GPIO mode. The DR supplies the output data if this I/O pin is set as an output in GPIO mode. This is a read/write register.

GPIO_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	D							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-8. Data Register (DR)

See Programmer's Sheets on Appendix page B-43

GPIO_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 8-9. Data Register (DR) for Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-43

8.8.3 Data Direction Register (DDR)

Bits within this read/write register control the pin's direction when in the GPIO mode. In the Normal mode these bits have no effect.

- 1 = Pin is an output; pull-ups are disabled
- 0 = Pin is an input; pull-ups are dependent on value of PUR registers

GPIO_BASE+$2	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	DD							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-10. Data Direction Register (DDR)

See Programmer's Sheets on Appendix page B-44

GPIO_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 8-11. Data Direction Register (DDR) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-44

8.8.4 Peripheral Enable Register (PER)

This read/write register determines the GPIO's configuration. When the PER value is one, the peripheral manages the GPIO pin. This mastery includes configuring the GPIO pin as a required input, with or without pull-up. It also may be an output, depending on the status of the peripheral output enable. It includes data transfers from the GPIO pin to the peripheral.

- 1 = Normal mode; pin operation is controlled by peripheral
- 0 =GPIO mode; pin operation is controlled by GPIO registers

GPIO_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	PE							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 8-12. Peripheral Enable Register (PER)

See Programmer's Sheets on Appendix page B-45

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

Figure 8-13. Peripheral Enable Register (PER) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-45

8.8.5 Interrupt Assert Register (IAR)

This read/write register is only for software testing. When the IAR is a one, an interrupt is asserted. It can be cleared by writing zeros (56F826) or ones (56F827) to the IAR register.

GPIO_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	IA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-14. Interrupt Assert Register (IAR)

See Programmer's Sheets on Appendix page B-46

GPIO_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 8-15. Interrupt Assert Register (IAR) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-46

8.8.6 Interrupt Enable Register (IENR)

This read/write register enables or disables edge detection for any incoming interrupt (external) from the GPIO pin. The interrupts are recorded in the GPIO_IPR.

- 1 = Enable the edge detection for external interrupt from the GPIO pin
- 0 = Disable the edge detection

GPIO_BASE+$5	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	IEN							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-16. Interrupt Enable Register (IENR)

See Programmer's Sheets on Appendix page B-47

GPIO_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 8-17. Interrupt Enable Register (IENR) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-47

8.8.7 Interrupt Polarity Register (IPOLR)

This read/write register is used to configure the external interrupts polarity detection external interrupts.

- 1 = The interrupt at the GPIO pin is active LOW
- 0 = The interrupt at the GPIO pin is active HIGH

GPIO_BASE+$6	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	IPOL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-18. Interrupt Polarity Register (IPOLR)

See Programmer's Sheets on Appendix page B-48

GPIO_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 8-19. Interrupt Polarity Register (IPOLR) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-47

8.8.8 GPIO Interrupt Pending Register (GPIO_IPR)

This *read-only* register is used to record any incoming interrupts. This register is read to determine which pin caused an interrupt by looking at which corresponding bit is set. Clear this register by writing zeros in the IAR if the interrupt is caused by software. The GPIO_IPR is cleared by writing zeros (56F826) or ones (56F827) into the interrupt edge-sensitive register (IESR) if interrupt is caused by an external interrupt.

GPIO_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	IPR							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-20. GPIO Interrupt Pending Register (GPIO_IPR)

See Programmer's Sheets on Appendix page B-49

GPIO_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 8-21. GPIO Interrupt Pending Register (GPIO_IPR) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-49

8.8.9 Interrupt Edge Sensitive Register (IESR)

When an edge is detected by the edge detector circuit, illustrated in **Figure 8-4,** and the IENR is set to one, the IESR records the interrupt. This register is read/write; however, the only time the register will receive writing is when the GPIO_IPR is being cleared.

Note: For the 56F826 device, write only zeros into this register to clear the GPIO_IPR. On the 56F827 device, write only ones into this register to clear the GPIO_IPR.

GPIO_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	IES							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-22. Interrupt Edge-Sensitive Register (IESR)

See Programmer's Sheets on Appendix page B-50

GPIO_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 8-23. Interrupt Edge-Sensitive Register (IESR) Port A and Port G (56F827 Only)

See Programmer's Sheets on Appendix page B-50

Table 8-7. GPIO Data Transfers Between I/O Pad and IPBus

Peripheral Out Enable	PER	DDR	Pin State	Access	Data Access Result
x	0	0	Input	Write to DR	Data is written into DR by IPbus. No effect on the pin value.
x	0	1	Output	Write to DR	Data is written into the DR by the IPbus. DR value seen at pin.
x	0	0	Input	Read from DR	Pin state is read by the IPbus. No effect on DR value.
x	0	1	Output	Read from DR	DR value is read by the IPbus. DR value seen at pin.
1	1	x	Input	Write to DR	Data is written into the DR by IPbus. No effect on pin value.
0	1	x	Output	Write to DR	Data is written into the DR by the IP bus. Data is seen at pin.
1	1	x	Input	Read from DR	DR value is read by the IPbus. No effect on the pin or DR value.
0	1	x	Output	Read from DR	DR value is read by the IPbus. Data is seen at the pin.

x = Don't Care

Chapter 9
Analog-to-Digital Converter (ADC)

56F827 Only

9.1 Introduction

Of the two devices discussed in this manual only the 56F827 possesses analog-to-digital capability. It has a single,10-input Analog-to-Digital Converter (ADC). It is labeled ADC as its reference name. Please refer to ADC in Appendix B for complete details.

9.2 Features

The ADC consists of a digital control module and an analog core. Characteristics include:

- 12-bit resolution, 10-bit accuracy
- ± 1 Least Significant Bit maximum absolute error (all sources included)
- Sampling rate up to 0.416 million samples per second[1]
- Maximum ADC Clock frequency is 2.5MHz with 400ns period
- Single conversion time of 8.5 ADC Clock cycles (8.5 × 400 ns = 3.4µs)
- 10 conversions in 62.5 ADC Clock cycles (62.5 × 400ns = 25µs)
- ADC can be synchronized to the timer via the SYNC signal using TA0
- Sequential sampling with programmable sequence
- Single-ended or differential inputs
- Internal multiplexer to select 1 of 10 inputs (single-ended) or 2 of 10 inputs (differential)
- Analog buffer to increase input impedance
- Ability to sequentially scan and store up to 10 measurements
- 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
- Low power mode
- Output Voltage References
- Operating Voltage 3.3V

1.Once in Loop Mode, the time between each conversion is six ADC Clock cycles (2.4 µs). Samples per second is calculated according to 2.4µs per sample or 416666 samples per second.

9.3 Block Diagram

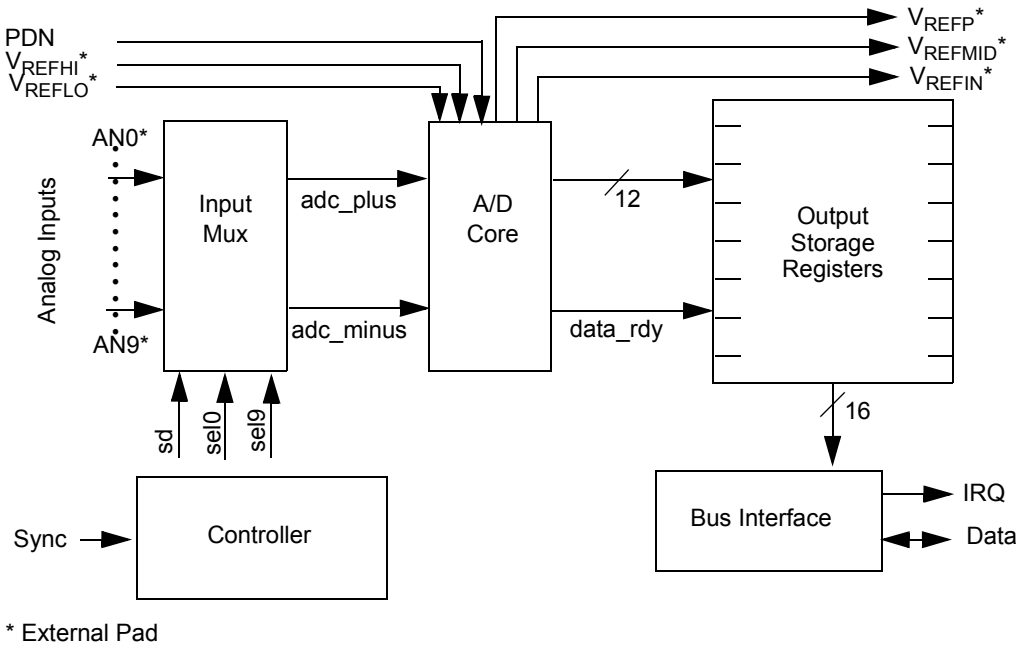


Figure 9-1. ADC Block Diagram

9.4 Functional Description

The ADC function, illustrated in **Figure 9-1,** consists of a10-channel input select function feeding a 12-bit ADCs. The A/D converter stores its results in an accessible buffer for further processing by the internal functions. The conversion process is either initiated by a SYNC signal or a write to the Control Register (ADCR1) Start bit.

The ADC consists of a cyclic, algorithmic architecture using two recursive sub ranging sections. Each suborning section resolves a single bit for each conversion clock, resulting in an overall conversion rate of two bits per clock cycle. Each suborning section was designed to run at a maximum clock speed of 2.5MHz, so a complete 12-bit conversion can be accommodated in 2.4µs, not including sample or post processing time.

$$\text{Result} = ((Vx / (Vrh-Vrl)) \times 4096 \text{ (rounded to the nearest LSB)}$$

Starting a conversion really starts a sequence of conversions, or a *scan*. A scan can sample and convert up to 10 unique single-ended channels, and up to 10 differential channel pairs, or some combination of these.

The ADC can be configured to perform a single scan and halt, perform a scan whenever triggered, or perform the programmed scan sequence repeatedly until manually stopped.

The ADC can be configured for sequential conversion. When configured for sequential configuration, up to 10 channels, all single-ended and differential or a mix of these, can be sampled and stored in any order specified by the Channel List register.

The Channel List register is programmed with the scan order of the desired channels.

Optional interrupts can be generated at the end of the scan sequence, if a channel is out of range as determined by the High and Low Limit registers, or at several different zero crossing conditions.

The method for initiating a conversion, or a scan consisting of multiple conversions, is programmable. It can be set to be either the SYNC signal or a write to the ADC Control Register (ADCR1) Start bit. If a scan is initiated while another scan is in process, the start signal is ignored until the conversion is complete, or when Conversion In Progress (CIP) is *deasserted* to zero.

9.5 Operating Modes

9.5.1 Normal Mode

The ADC has two modes of normal operation. The mode of operation for a given sample is determined by SAMPLEn[6] bit of ADLSTx register. Here are the two normal modes of operation:

1. Single-Ended Mode (SAMPLEn[6] = 0): In the single-ended mode, the ADC converts the voltage between one of the analog inputs and the reference voltage, to a digital representation. The input mux of the 56827 ADC selects one of the 10 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 reference voltage during this mode.
2. Differential Mode (SAMPLEn[6] = 1): In the differential mode, the ADC converts the voltage between two analog inputs, to a digital representation. One of the analog inputs should be an even analog input such as: AN0, AN2, AN4, AN6, or AN8 and the other should be an odd analog input using either: AN1, AN3, AN5, AN7, or AN9. In this mode, the plus terminal of the A/D core is connected to the even analog inputs while the minus terminal is connected to the odd analog input.

The number of conversions in a scan is programmable from 1 to 10. Up to 10 conversions can be sampled and stored in any order specified by the Channel List Register.

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

- AN0 and AN3 are differential
- AN4 and AN9 are both single-ended

- AN8 and AN1 are differential
- AN2 and AN6 are both signal-ended
- AN0 and AN7 are differential

If the scan parameter indicates more than one conversion, a second conversion is immediately initiated upon completion of the first conversion.

9.5.2 Low Power Mode

The analog core of the ADC can be shut down allowing reduced power consumption when the ADC module is not being used. In the low-power mode current of the ADC is <1µA.. The ADC analog core can be powered down by setting the PDN bit of the ADCR1. When set to one, the PDN bit powers down the analog core of the ADC, stops any conversion sequence in progress and sets the PDNS status bit in the ADSTAT1 register. The results registers can be modified by writing to the processor after the ADC is in the power-down mode. Any write to the result register in power-down mode is treated as if the analog core supplied the data, therefore, limit checking and zero crossing and associated interrupts can occur if enabled.

When ADC is powered down the output reference voltages are set to Low (V_{SSA}) and ADC data output is driven low. The ADC analog core is powered down on (PDN = 1) on reset.

When the PDN is *deasserted*, or set to 0, a new conversion sequence cannot be started until the PDNS status bit gets *deasserted*, also set to 0, after ADC power-up recovery time of 16 ADC_CLK cycles. Coming out of the Low power mode, ADC does not resume its vacated state when it was powered down. A new scan sequence may be started with a new sync pulse, or a write to the Start bit as configured.

Note: Only when the ADC is powered up are voltage reference levels V_{REVP}, V_{REFMID}, V_{REFN} generated. This requires that some delay occur to allow the bypass capacitors on these pins to charge when powering up the ADC. If the bypass capacitors are not fuly charged, ADC accuracy is adversely affected. This delay may be longer than the 16 ADC_CLK cycles required to allow a conversion.

9.5.3 STOP Mode

Any conversion sequence in progress can be stopped by setting the ADCSTOP bit in the ADCR1. Any further SYNC pulses or writes to the Start bit are ignored until the ADCSTOP bit is cleared. After the ADC is in ADC-STOP mode, the results registers can be modified by writing to the processor. Any write to the result 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.

Coming out of the ADC-STOP mode, ADC does not resume its vacated state when it was stopped. A new scan sequence may be started with a new sync pulse, or a write to the Start bit as configured.

9.6 Timing

Figure 9-2 illustrates a timing diagram for the ADC module.

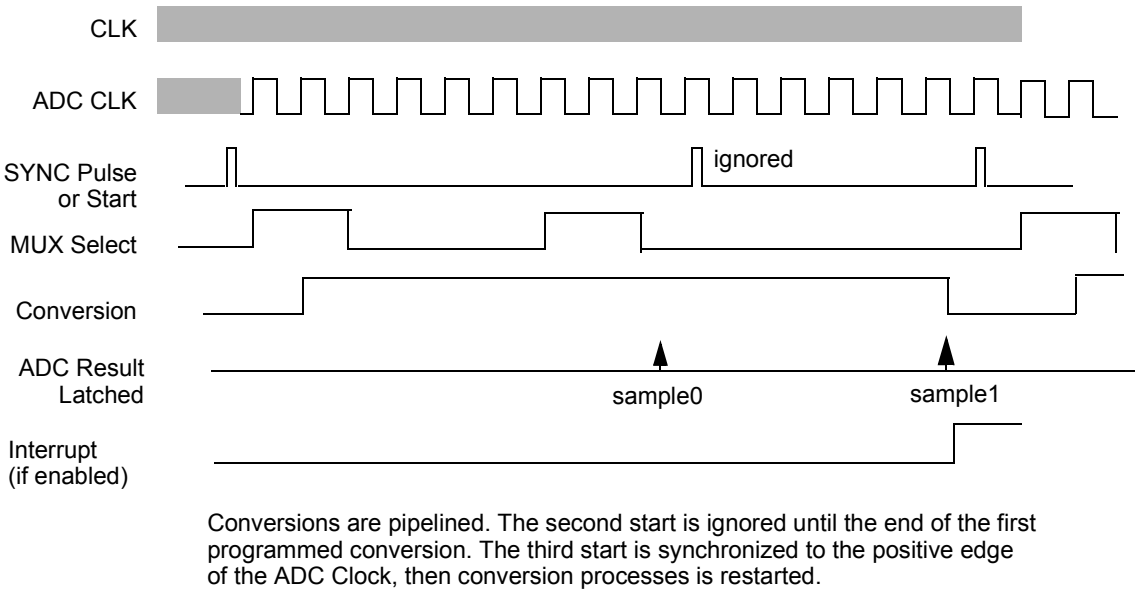


Figure 9-2. ADC Timing

ADC_CLK is derived from the IPBus Clock. The frequency relationship is programmable.

A conversion is initiated by a SYNC pulse originating from the timer module, shown in **Figure 9-2,** or by a write to the ADCR1 Start bit. In the Clock cycle following this pulse the conversion is initiated. The ADC Clock (ADC_CLK) period is determined by the DIV[4:0] value in the ADCR2 register. The positive edge of the ADC clock aligns to the negative edge of the IPBus Clock (CLK).

Once the first conversion is started, the first result takes 8.5 ADC Clocks to be valid. Each additional sample only takes 6 ADC Clocks. The start conversion command is latched and the real conversion process is synchronized to the positive edge of the ADC Clock.

Because the conversion is a pipeline process, once the last sample has been acquired in the sample and hold circuit, the ADC can not be restarted until the pipeline is emptied. However, the conversion cycle can be aborted by issuing an ADC-STOP command.

9.7 Pin Descriptions

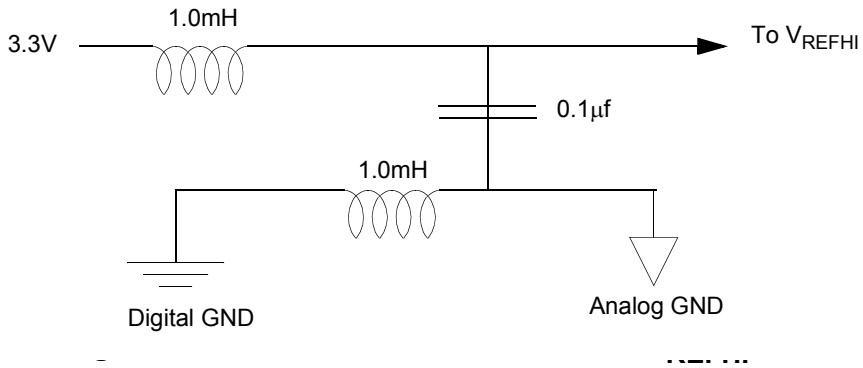
9.7.1 Analog Input Pins (AN[0-9])

The ADC module had 10 analog inputs connected to an internal multiplexer. The multiplexer selects the analog voltage to be converted. When in single-ended mode, there are 10 analog inputs to perform a conversion. When in the differential mode, only an odd numbered input can be paired with an even numbered input. Please refer to **Section 9.8.4** and **Figure 9-5** for further details.

9.7.2 Voltage Reference (V_{REF})

The difference between V_{REFHI} and V_{REFLO} provides the reference voltage all analog inputs are measured against. The V_{REFLO} signal nominally sets the lowest value achieved by any of the analog input signals typically the analog ground. The V_{REFHI} signal is nominally set to the highest value achieved by any of the analog input signals. V_{SSA} is the ground reference supplied to the ADC. All analog input signals measured by the ADC should not go below this level. The reference voltages should be provided from a low noise filtered source. Bypass capacitors should be connected between V_{REFHI} and V_{REFLO} (V_{SSA} if V_{REFLO} is tied to V_{SSA}).

The V_{REFHI} signal is expected to be as noise free as possible for the ADC to maintain good accuracy. If the analog input signals, being measured by the ADC, exceed either V_{REFHI} or V_{SSA}, the ADC will produce unpredictable results.



The ADC provides three Output reference voltages between V_{REFHI} and V_{REFLO}:

- V_{REFP}–Output reference voltage Positive
- V_{REFMID}–Output reference voltage Middle
- V_{REFN}–Output reference voltage Negative

The voltage reference levels V_{REFP}, V_{REFMID}, and V_{REFN} are first internally generated using a high impedance resistive divider. They are then buffered and sent out to package pins where they must be filtered using bypass capacitors. During ADC operation charge is pushed and pulled from the external bypass capacitors. The voltage change on the bypass capacitor, as charge is being pushed and pulled from it, must never exceed half of a Least Significant Bit (LSB), otherwise the ADC accuracy is adversely affected.

Each output reference voltage pin (V_{REFP}, V_{REFMID}, and V_{REFN}) should be connected to a 0.1 µF bypass capacitor to VSSA and a startup time of 25 ms prior to beginning conversions. These pins should be bypassed to V_{REFLO}.

Note: Only when the ADC is powered up are voltage reference levels V_{REVP}, V_{REFMID}, V_{REFN} generated. This requires that some delay occur to allow the bypass capacitors on these pins to charge when powering up the ADC. If the bypass capacitors are not fully charged, ADC accuracy is adversely affected. This delay may be longer than the 16 ADC_CLK cycles required to allow a conversion.

9.7.3 Supply Pins (V_{DDA} and V_{SSA})

To reduce noise coupling and improve accuracy, dedicated power supply pins are provided for the analog portion of the ADC module. The power provided to these pins should be connected between V_{DDA} and V_{SSA}. These pins must be connected to a nominal 3.3V power supply.

9.8 Register Definitions

Table 9-4. ADC Memory Map

Device	Peripheral	Address
827 Only	ADC_BASE	$12C0

The actual memory address of a register is the sum of a base address and the registers' address offset. The base address is defined at the core. The address offset is defined at the module level. Please reference Table 3-37**.**

Note: Register Address = Base Address + Address Offset, where the Base Address is defined at Chip level and the Address Offset is defined at the module level.

Table 9-5. ADC Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + $0	ADCR1	Control Register 1	Read/Write	Section 9.8.1
Base + $1	ADCR2	Control Register 2	Read/Write	Section 9.8.2
Base + $2	ADZCC1	Zero Crossing Control Register 1	Read/Write	Section 9.8.3
Base + $3	ADZCC2	Zero Crossing Control Register 2	Read/Write	
Base + $4	ADLST1	Channel List Register 1	Read/Write	Section 9.8.4
Base + $5	ADLST2	Channel List Register 2	Read/Write	
Base + $6	ADLST3	Channel List Register 3	Read/Write	
Base + $7	ADLST4	Channel List Register 4	Read/Write	
Base + $8	ADLST5	Channel List Register 5	Read/Write	
Base + $9	ADSDIS	Sample Disable Register	Read/Write	Section 9.8.5
Base + $A	ADSTAT1	Status Register 1	Read/Write	Section 9.8.6
Base + $B	ADSTAT2	Status Register 2	Read/Write	
Base + $C	ADLLSTAT	Low Limit Status Register	Read/Write	Section 9.8.7
Base + $D	ADHLSTAT	High Limit Status Register	Read/Write	
Base + $E	ADCZSTAT	Zero Crossing Status Register	Read/Write	Section 9.8.8
Base + $F - $18	ADRSLT0-9	Result Registers 0-9	Read/Write	Section 9.8.9
Base + $19 - $22	ADLLMT0-9	Low Limit Registers 0-9	Read/Write	Section 9.8.10
Base + $23 - $2C	ADHLMT0-9	High Limit Registers 0-9	Read/Write	
Base + $2D - $36	ADOFS0-9	Offset Registers 0-9	Read/Write	Section 9.8.11

The ADC occupies 57 words in the memory space beginning at ADC_BASE. The register decode map is fixed and begins at the first address of the module address offset. It shows how the registers are organized. All ADC registers are implemented as 16-bit registers.

The ADC registers in **Figures 9-1** through **Figures 9-19** illustrate individual registers associated with the ADC and their relative offset from the base address. The detailed register descriptions follow in the order they appear in the register map in **Table 9-4.** Reserved bits within a register always read as 0 and a write does nothing. Functions that are not implemented are indicated by a shaded bit.

Bit fields of each of the registers are summarized in **Figure 9-6.** Details of each follow.

Add.. Off-set	Reg. Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	ADCR1	R	PDN	ADC STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	0	0	0	0	0	0	SMODE	
		W			START													
$1	ADCR2	R	0	0	0	0	0	0	0	0	0	0	0	DIV[4:0]				
		W																
$2	ADZCC1	R	ZCE7[1:0]		ZCE6[1:0]		ZCE5[1:0]		ZCE4[1:0]		ZCE3[1:0]		ZCE2[1:0]		ZCE1[1:0]		ZCE0[1:0]	
		W																
$3	ADZCC2	R	0	0	0	0	0	0	0	0	0	0	0	0	ZCE9[1:0]		ZCE8[1:0]	
		W																
$4	ADLST1	R	0	SAMPLE1[6:0]							0	SAMPLE0[6:0]						
		W																
$5	ADLST2	R	0	SAMPLE3[6:0]							0	SAMPLE2[6:0]						
		W																
$6	ADLST3	R	0	SAMPLE5[6:0]							0	SAMPLE4[6:0]						
		W																
$7	ADLST4	R	0	SAMPLE7[6:0]							0	SAMPLE6[6;0]						
		W																
$8	ADLST5	R	0	SAMPLE9[6:0]							0	SAMPLE8[6:0]						
		W																
$9	ADSDIS	R	0	0	0	0	0	0	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
		W																
$A	ADSTAT1	R	CIP	PDNS	0	0	EOSI	ZCI	LLLMT	HLMTI	0	0	0	0	0	0	0	0
$B	ADSTAT2	R	0	0	0	0	0	0	RDY9	RDY8	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
		W																
$C	ADLLSTAT	R	0	0	0	0	0	0	LLMT9	LLMT8	LLMT7	LLMT6	LLMT5	LLMT4	LLMT3	LLMT2	LLMT1	LLMT0
		W																
$D	ADHLSTAT	R	0	0	0	0	0	0	HLMT9	HLMT8	HLMT7	HLMT6	HLMT5	HLMT4	HLMT3	HLMT2	HLMT1	HLMT0
		W																
$E	ADZCSTAT	R	0	0	0	0	0	0	ZCS9	ZCS8	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
		W																
$F	ADRSLT0	R	SEXT	RSLT0												0	0	0
		W																
$10	ADRSLT1	R	SEXT	RSLT1												0	0	0
		W																
$11	ADRSLT2	R	SEXT	RSLT2												0	0	0
		W																
$12	ADRSLT3	R	SEXT	RSLT3												0	0	0
		W																
$13	ADRSLT4	R	SEXT	RSLT4												0	0	0
		W																
$14	ADRSLT5	R	SEXT	RSLT5												0	0	0
		W																
$15	ADRLST6	R	SEXT	RSLT6												0	0	0
		W																
$16	ADRSLT7	R	SEXT	RSLT7												0	0	0
		W																
$17	ADRSLT8	R	SEXT	RSLT8												0	0	0
		W																
$18	ADRSLT9	R	SEXT	RSLT9												0	0	0
		W																

Figure 9-6. ADC Register Map Summary

$19	ADLLMT0	R	0	LLMT0	0	0	0
		W					
$1A	ADLLMT1	R	0	LLMT1	0	0	0
		W					
$1B	ADLLMT2	R	0	LLMT2	0	0	0
		W					
$1C	ADLLMT3	R	0	LLMT3	0	0	0
		W					
$1D	ADLLMT4	R	0	LLMT4	0	0	0
		W					
$1E	ADLLMT5	R	0	LLMT5	0	0	0
		W					
$1F	ADLLMT6	R	0	LLMT6	0	0	0
		W					
$20	ADLLMT7	R	0	LLMT7	0	0	0
		W					
$21	ADLLMT8	R	0	LLMT8	0	0	0
		W					
$22	ADLLMT9	R	0	LLMT9	0	0	0
		W					
$23	ADHLMT0	R	0	HLMT0	0	0	0
		W					
$24	ADHLMT1	R	0	HLMT1	0	0	0
		W					
$25	ADHLMT2	R	0	HLMT2	0	0	0
		W					
$26	ADHLMT3	R	0	HLMT3[	0	0	0
		W					
$27	ADHLMT4	R	0	HLMT4	0	0	0
		W					
$28	ADHLMT5	R	0	HLMT5	0	0	0
		W					
$29	ADHLMT6	R	0	HLMT6	0	0	0
		W					
$2A	ADHLMT7	R	0	HLMT7	0	0	0
		W					
$2B	ADHLMT8	R	0	HLMT8	0	0	0
		W					
$2C	ADHLMT9	R	0	HLMT9	0	0	0
		W					
$2D	ADOFS0	R	0	OFFSET0	0	0	0
		W					
$2E	ADOFS1	R	0	OFFSET1	0	0	0
		W					
$2F	ADOFS2	R	0	OFFSET2	0	0	0
		W					
$30	ADOFS3	R	0	OFFSET3[	0	0	0
		W					
$31	ADOFS4	R	0	OFFSET4	0	0	0
		W					
$32	ADOFS5	R	0	OFFSET5	0	0	0
		W					

Figure 9-6. ADC Register Map Summary (Continued)

$33	ADOFS6	R	0	OFFSET6												0	0	0	
		W																	
$34	ADOFS7	R	0	OFFSET7												0	0	0	
		W																	
$35	ADOFS8	R	0	OFFSET8												0	0	0	
		W																	
$36	ADOFS9	R	0	OFFSET9												0	0	0	
		W																	
$37	ADTEST0	R	RDY	0	0	0	DOUT												
		W																	
$38	ADTEST1	R	TST	0	CLK	CONV	DIFF	0	SEL9	SEL8	SEL7	SEL6	SEL5	SEL4	SEL3	SEL2	SEL1	SEL0	
		W																	

R	0	Read as 0
W		Reserved

Figure 9-6. ADC Register Map Summary (Continued)

9.8.1 ADC Control Register 1 (ADCR1)

BASE+$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	PDN	ADC STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	0	0	0	0	0	1	SMODE	
Write			START													
Reset	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0

Figure 9-1. ADC Control Register 1 (ADCR1)

See Programmer's Sheets on Appendix page B- 51

9.8.1.1 Power Down (PDN)—Bit 15

This bit provides program on/off control over the ADC Analog Core allowing reduced power consumption when ADC module is not being used. When set to one, the PDN bit powers down the analog core of the ADC, stops any conversion sequence in progress and sets the PDNS status bit in the ADSTAT1 register. Any further sync pulses written to the Start bit are ignored until both the PDN (ADCR1) and PDNS (ADSTAT1[14]) bits are cleared. After the ADC is in power-down mode, the results registers can be written to by the processor. Any write to the result register in power-down mode is treated as if the analog core supplied the data. Therefore, limit checking and zero crossing and associated interrupts can occur if enabled.

Because the analog electronics is turned-off when powered down, the ADC analog core requires a recovery time period of 16 ADC_CLK cycles when PDN bit is cleared. To prevent an accidental starting any conversion while the analog core is getting powered up, PDNS bit of ADSTAT1 gets cleared only after 16 ADC_CLK cycles, and after the PDN bit is cleared. A new conversion sequence cannot be started until PDNS status bit gets de asserted, or set to zero.

When ADC is powered down, the output reference Voltages VREFP, VREFMID and VREFN are set to low (V_{SSA}) and ADC data output is driven low.

Note: Only when the ADC is powered up are voltage reference levels V_{REVP}, V_{REFMID}, V_{REFN} generated. This requires some delay (about 3mS delay if each V_{REVP}, V_{REFMID}, V_{REFN} pin is connected to 0.1µF capacitor, and the other terminal of the capacitor to ground) occur to allow the bypass capacitors on these pins to charge when powering up the ADC. If the bypass capacitors are not fuly charged, ADC accuracy is adversely affected. This delay may be longer than the 16 ADC_CLK cycles required to allow a conversion.

The ADC analog core is powered down (PDN = 1) on reset.

- 0 = Power-up ADC analog core command is issued
- 1 = Power-down ADC analog core command is issued

9.8.1.2 ADCSTOP (STOP)—Bit 14

If ADCSTOP is asserted, the current conversion process is stopped. Any further sync pulses or writes to the Start bit are ignored until the ADCSTOP bit is cleared. After the ADC is in ADC-Stop mode, the results registers can accept writing by the processor. Any write-to the result register in 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. The ADC is in ADC-Stop mode (ADCSTOP=1) on reset.

- 0 = Normal operation
- 1 = ADC-Stop command issued

Setting either PDN or ADCSTOP bit has the same effect on ADC except when PDN is set, the ADC analog core is also powered down. When powering up the analog core, wait for the recovery time by checking the PDNS bit before starting normal ADC conversions.

9.8.1.3 START Conversion (START)—Bit 13

The conversion process is initiated by a write to Start bit. This is a *write-only* bit and is always read a 0. Once a start command is issued, the conversion process is synchronized and started on the positive edge of the ADC Clock. A Start bit reasserted while the ADC is executing a conversion cycle is ignored. The ADC must be idle in order for it to recognize this event.

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

9.8.1.4 SYNC Select (SYNC)—Bit 12

A conversion can be initiated by the sync input or by a write to the Start bit. A sync pulse reasserted while the ADC is executing a conversion cycle is ignored. The ADC must be idle in order for it to recognize this event.

- 0 = Conversion is initiated by a write to Start bit only
- 1 = Use the sync input or Start bit to initiate a conversion

9.8.1.5 End Of Scan Interrupt Enable (EOSIE)—Bit 11

This bit enables an interrupt to be generated upon completion of any scan and convert sequence, except for the Loop Sequential mode. This bit is ignored when the ADC is configured for the Loop Sequential mode.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

9.8.1.6 Zero Crossing Interrupt Enable (ZCIE)—Bit 10

This enables the optional interrupt if the current result value has a sign change from the previous result.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

9.8.1.7 Low Limit Interrupt Enable (LLMTIE)—Bit 9

This enables the optional interrupt if the current result value is less than the Low Limit register value. The raw result value is compared to the Limit register (LLMT[11:0]) before the Offset register value is subtracted.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

9.8.1.8 High Limit Interrupt Enable (HLMTIE)—Bit 8

This enables the optional interrupt if the current result value is greater then the High Limit register value. The raw result value is compared to the Limit register (HLMT[11:0]) before the offset register value is subtracted.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

9.8.1.9 Reserved—Bits 7–2

These bits cannot be modified. They are read as zero.

9.8.1.10 Scan Mode (SMODE)—Bits 1–0

A conversion sequence can be initiated by asserting the ADCR1-Start bit or by a sync pulse. A conversion sequence can take up to 10 samples. These 10 sample slots are enabled by the ASDIS register. A sample sequence is terminated when the first disabled sample is encountered. Analog input channels are assigned to each of the 10 sample slots by the ADLST1, ADLST2, ADLST3, ADLST4 and ADLST5 registers. A conversion sequence can run only once every time a trigger occurs, or it will continuously loop.

Samples may be taken one at a time.

- 00 = Once Sequential
  Upon start or a first 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 the SAMPLE9.
- Provided a sampling sequence has completed, a start pulse will always initiate another Once sequence. A sync pulse however must be re-armed (via another write to the ADCR1 register), in order for a second sampling sequence to occur.
- 01 = Loop Sequential
  Upon start or a first sync pulse, samples are taken one at a time, until a disabled sample is encountered. The process repeats forever until the ADCSTOP bit is set in ADCR1. Setting the ADCSTOP bit immediately stops any conversion in progress. For example, if samples 0 1 and 5 were enabled, the loop would look like SAMPLE0, SAMPLE1, SAMPLE0, SAMPLE1, and so on. Additional start commands or sync pulses are ignored while a loop mode is running.
- 10 = Triggered Sequential
  A single sequential sampling begins with every recognized start command or sync pulse. Samples are taken sequentially as described above.

  Reasserted start bits and sync pulses presented during an active sample sequence are not recognized. The ADC must be in its idle state (CIP low) to recognize these events.
- 11 = Factory Reserved Use

The SMODE is set to *Triggered Sequential* on Reset.

9.8.2 ADC Control Register 2 (ADCR2)

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	DIV				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-2. ADC Control Registers 2 (ADCR2)

9.8.2.1 Reserved—Bits 15–5

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

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

ADC_CLK can be run at a slower rate than the IPBus clock. The divider circuit provides a clock of period 2N of the IPBus clock where N = DIV[4:0]+1. A divisor must be chosen allowing the ADC clock to remain within specified limits. For a IPBus Clock frequency of 40MHz and a desired ADC_CLK frequency of 2.5MHz, a DIV[4:0] value of 7d is required.

Note: The maximum frequency ADC can operate is 2.5MHz. DIV[4:0]. ADC should not be programmed for ADC_CLK frequency more than 2.5MHz. At higher frequencies, the accuracy of the ADC will be less than its specified value.

9.8.3 Zero Crossing Control Register (ADZCC1 and ADZCC2)

The Zero Crossing Control (ADZCC) register provides the ability to monitor the selected channels and select the direction of zero crossing that triggers the optional interrupt. Zero crossing logic monitors only the sign change between current and previous sample. ZCE0 monitors the sample stored in ADRSLT0 and ZCE9 monitors ADRSLT9. When the zero crossing is disabled for a selected result register, sign changes are not monitored and updated in the ADZSTAT but are properly stored in the respective result register.

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 9-3. ADC Zero Crossing Control Register (ADZCC1)

See Programmer's Sheets on Appendix page B- 53

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	ZCE9		ZCE8	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-4. ADC Zero Crossing Control Register (ADZCC2)

See Programmer's Sheets on Appendix page B- 53

ZCEn - Zero Crossing Enable *n*

n represents the channel number used to enable or disable zero crossing.

- 00 = Zero crossing disabled
- 01 = Zero crossing enabled for positive to negative sign change
- 10 = Zero crossing enabled for negative to positive sign change
- 11 = Zero crossing enabled for any sign change

9.8.4 ADC Channel List Registers (ADLST1–ADLST5)

The channel list register contains an ordered list of the channels to be converted when the next scan is initiated. If all samples are enabled (ADSDIS register), a sequential scan of inputs proceeds in the order of SAMPLE0 up through SAMPLE9.

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

Figure 9-5. ADC Channel List Register 1 (ADLST1)

See Programmer's Sheets on Appendix page B- 55

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

Figure 9-6. ADC Channel List Register 2 (ADLST2)

See Programmer's Sheets on Appendix page B- 55

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

Figure 9-7. ADC Channel List Register 3 (ADLST3)

See Programmer's Sheets on Appendix page B- 55

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

Figure 9-8. ADC Channel List Register 4 (ADLST4)

See Programmer's Sheets on Appendix page B- 55

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

Figure 9-9. ADC Channel List Register 5 (ADLST5)

See Programmer's Sheets on Appendix page B- 55

9.8.4.1 Channel Select (SAMPLEn)

This is a binary representation of the analog input to be converted. SAMPLEn[6] acts as a select for single-ended or differential input select for the corresponding sample.

9.8.4.2 Configuration Select (SAMPLEn)

- 0 - Single ended sample input
- 1 - Differential sample input

Table 9-7. ADC Input Conversion for Sample Bits

SAMPLEn[6:0]							Single Ended	Differential
6	5	4	3	2	1	0		
0	0	0	0	0	0	0	AN0	–
0	0	0	0	0	0	1	AN1	–
0	0	0	0	0	1	0	AN2	–
0	0	0	0	0	1	1	AN3	–
0	0	0	0	1	0	0	AN4	–
0	0	0	0	1	0	1	AN5	–
0	0	0	0	1	1	0	AN6	–
0	0	0	0	1	1	1	AN7	–
0	0	0	1	0	0	0	AN8	–
0	0	0	1	0	0	1	AN9	–
0	Others						Reserved	–
	AN_{even}+			AN_{odd}-				
1	0	0	0	x	x	x	–	AN0+, AN0-
1	0	0	1	x	x	x	–	AN2+, AN0-
1	0	1	0	x	x	x	–	AN4+, AN0-

Table 9-7. ADC Input Conversion for Sample Bits (Continued)

SAMPLEn[6:0]							Single Ended	Differential
6	5	4	3	2	1	0		
1	0	1	1	x	x	x	–	AN6+, AN0-
1	1	0	0	x	x	x	–	AN8+, AN0-
1	x	x	x	0	0	0	–	ANe+, AN1-
1	x	x	x	0	0	1	–	ANe+, AN3-
1	x	x	x	0	1	0	–	ANe+, AN5-
1	x	x	x	0	1	1	–	ANe+, AN7-
1	x	x	x	1	0	0	–	ANe+, AN9-
1	Others			Others			–	Reserved
For example, to configure a sample as differential pair of AN4+, AN3-, SAMPLE[6:0] would be								
1	0	1	0	0	0	1	-	AN4+, AN3-
LEGEND: AN_{even}+, ANe+ : any of the ADC PLUS CHANNEL (even only) -AN0,AN2,AN4,AN6,AN8 AN_{odd}-, ANo- : any of the ADC MINUS CHANNEL (odd only) -AN1,AN3,AN5,AN7,AN9								

When channels are configured as *single-ended* inputs (SAMPLEn[6] = 0), there are no restrictions on assigning an analog channel (AN[0:9]) to a sample. Any sample slot, configured for single-ended input, may contain any channel assignment.

However, when channels are configured as *differential* inputs (SAMPLEn[6] = 1), there is one restriction. The plus terminal of the Analog Core can be one of the even channels (AN0, AN2, AN4, AN6, AN8) only and the minus terminal of the Analog Core can be one of the odd channels (AN1, AN3, AN5, AN7, AN9) only. This restriction is taken care with the channel select (SAMPLEn[6:0]) code. When SAMPLEn[6] = 1, SAMPLEn[5:3] decode the plus channel and SAMPLEn[2:0] decode the minus channel as shown in **Table 9-7.** Any sample slot, configured for differential inputs, may contain any channel pair, but with the above restriction.

Codes not mentioned in **Table 9-7** are reserved and should *not* be programmed. No damage will occur to the ADC if these restrictions are violated, however, measurement results are unpredictable.

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

- SAMPLE0: AN0+ and AN3- differential
- SAMPLE1: AN4 single-ended
- SAMPLE2: AN9 single-ended
- SAMPLE3: AN8+ and AN1- differential
- SAMPLE4: AN2 single-ended
- SAMPLE5: AN4+ and AN5- differential
- SAMPLE6: AN1 single-ended

- SAMPLE7: AN4 single-ended
- SAMPLE8: AN8+ and AN3- differential
- SAMPLE9: AN2+ and AN7- differential.

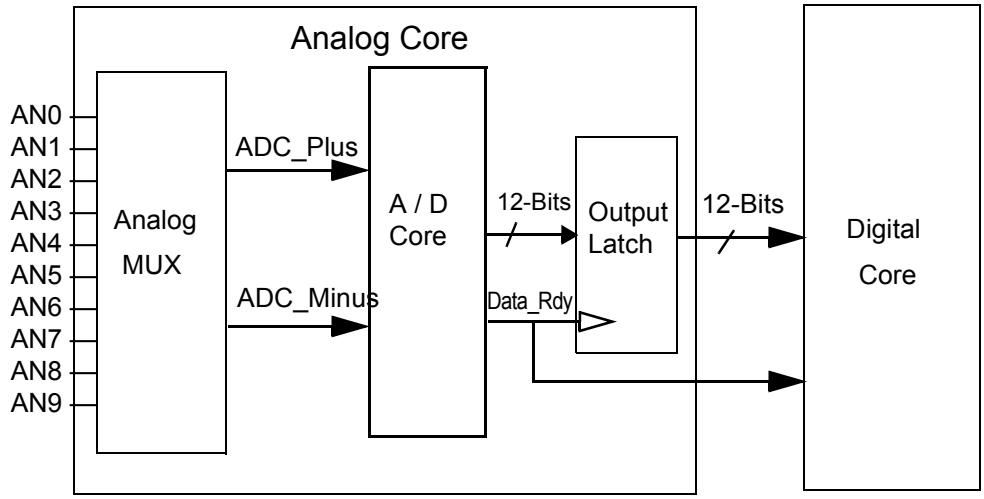


Figure 9-8. ADC Core

9.8.5 ADC Sample Disable Register (ADSDIS)

This register is an extension to the ADLST1, 2, 3,4, and 5 providing the ability to enable only the desired samples programmed in the SAMPLE0–SAMPLE9. A sample sequence is terminated when the first disabled sample is encountered. The power on default is all samples enabled. For example: if in sequential mode, DS5 is set to 1 and rest all set to 0, only SAMPLE0 through SAMPLE4 are sampled.

BASE+$9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-10. ADC Sample Disable Register (ADSDIS)

See Programmer's Sheets on Appendix page B- 56

9.8.5.1 Reserved—Bits 15–10

This bit field is reserved or not implemented. It cannot be read or modified.

9.8.5.2 Disable Sample (DS)—Bits 9–0

DSn – Disable Sample respective SAMPLEn can be enabled or disabled.

- 0 = Enable SAMPLEn
- 1 = Disable SAMPLEn

9.8.6 ADC Status Registers (ADSTAT1 and ADSTAT2)

This register provides the current status of the ADC module. RDYn bits are cleared by reading their corresponding result registers (ADRSLTn). HLMTI and LLMTI bits are cleared by clearing all asserted bits in the limit status registers (ADLLSTAT, ADHLSTAT). Likewise, the ZCI bit is cleared by clearing all asserted bits in the zero crossing status register (ADZCSTAT). The EOSI bit is cleared by writing a one to it. ADSTAT1 and ADSTAT2 register bits are *sticky*. Once set to a one state, they require some specific action to be taken to clear them. They are not cleared automatically on the next scan sequence.

BASE+$A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CIP	PDNS	0	0	EOSI	ZCI	LLMTI	HLMT	0	0	0	0	0	0	0	0
Write																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-11. ADC Status Register 1 (ADSTAT1)

See Programmer's Sheets on Appendix page B- 57

9.8.6.1 Conversion in Progress (CIP)—Bit 15

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

9.8.6.2 Power-Down Status (PDNS)—Bits 14

This bit indicates whether the analog core of the ADC is in power down mode or normal operational mode. Setting PDN bit of ADCR1 register sets the PDNS bit in the next clock cycle. Because the analog electronics is turned off when powered down, the ADC analog core requires a recovery time period of 16 ADC_CLK cycles when the ADCR1-PDN bit is cleared. Therefore, to prevent accidental starting any conversion, the PDNS bit is cleared 16 ADC_CLK cycles after the PDN bit has been cleared. The analog portion of the ADC cannot be used until the PDNS bit gets cleared. For example, a new conversion sequence cannot be started until PDNS status bit is deasserted.

Note: Only when the ADC is powered up are voltage reference levels V_{REVP}, V_{REFMID}, V_{REFN} generated. This requires that some delay occur to allow the bypass capacitors on these pins to charge when powering up the ADC. If the bypass capacitors are not fuly charged, ADC accuracy is adversely affected. This delay may be longer than the 16 ADC_CLK cycles required to allow a conversion.

Any *sync* pulses or writes to the Start bit are ignored unless both the PDN (ADCR1[15]) and PDNS (ADSTAT1[14]) bits are cleared. After the ADC is in power- down mode, the Results registers can receive writing by the processor. Any write to the Result register in the power-down mode is treated as if the analog core supplied the data; therefore, limit checking and zero crossing and associated interrupts can occur if enabled.

- 0 = Normal operation
- 1 = ADC analog core is powered down

9.8.6.3 Reserved—Bits 13–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

9.8.6.4 End of Scan Interrupt (EOSI)—Bit 11

This bit indicates if a scan of analog inputs have been completed since the last read of the Status Register, or since a reset. The EOSI bit is cleared by writing a one to it. For instance, a write of $0800 to ADSTAT1 will clear the EOSI bit. This bit is ignored when the ADC is configured for the Loop Sequential mode. A write of zero to the EOSI bit has no effect.

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

9.8.6.5 Zero Crossing Interrupt (ZCI)—Bit 10

If the respective Offset register is enabled by having a value greater than 0000h, zero crossing checking is enabled. If the Offset register is programmed with 7FF8h, the result will always be less than, or equal to zero. On the other hand, if 0000h is programmed into the Offset register, the result will always be greater than, or equal to zero. Zero crossing cannot occur because the sign of the result will not change. This bit is cleared by clearing all asserted bits in the zero crossing status register (ADZCSTAT).

- 0 = No ZCI IRQ
- 1 = Zero Crossing encountered, IRQ pending if ZCIE is set

9.8.6.6 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 bit is cleared by clearing all asserted bits in the limit status registers (ADLLSTAT).

- 0= No Low Limit IRQ
- 1 = Low Limit exceeded, IRQ pending if LLMTIE is set

9.8.6.7 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 bit is cleared by clearing all asserted bits in the limit status registers (ADHLSTAT).

- 0 = No High Limit IRQ
- 1 = High Limit exceeded, IRQ pending if HLMTIE is set

9.8.6.8 Ready Channel (RDY)—Bits 9–0

These bits indicate channels nine through zero are ready to be read. These bits are cleared after a read from the respective Results 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	0	RDY9	RDY8	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY2	RDY0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-12. ADC Status Register 2 (ADSTAT2)

See Programmer's Sheets on Appendix page B- 58

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

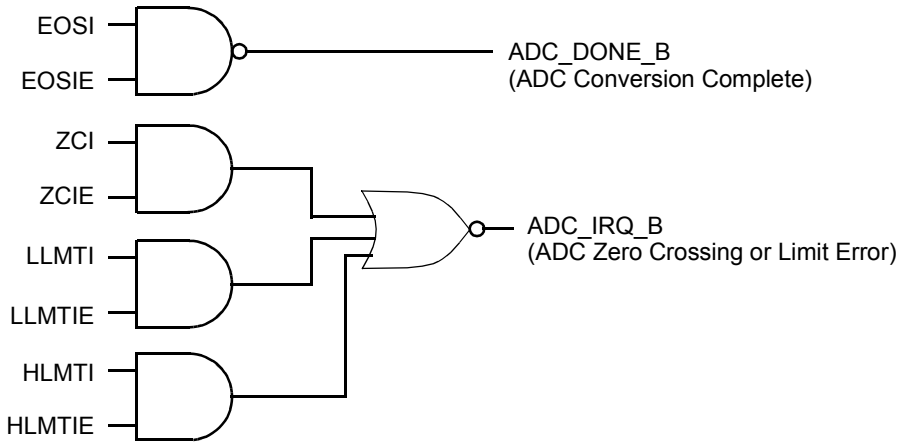


Figure 9-9. ADC Interrupts

9.8.7 ADC High and Low Limit Status Registers (ADHLSTAT and ADLLSTAT)

The Limit Status Registers latch-in the result of the comparison between the result of the sample and the respective Limit register. Note the raw result value is compared to the Limit register (HLMT[11:0]) before the Offset register value is subtracted. 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 HLMT0 bit is set to one. An interrupt is generated if the HLMTI bit is set in ADCR1. A bit may only be cleared by writing a value of one to that specific bit. These bits are *sticky*. Once set, they require a specific write to clear them. They are not automatically cleared by subsequent conversions.

BASE+$C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	LLS9	LLS8	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 9-13. ADC Limit Status Register (ADLLSTAT)

See Programmer's Sheets on Appendix page B- 59

BASE+$D	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	HLS9	HLS8	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-14. ADC Limit Status Register (ADHLSTAT)

See Programmer's Sheets on Appendix page B- 59

9.8.7.1 Reserved—Bits 15–10

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

9.8.7.2 Limit Status—Bits 9–0

- 0 = Channel n exceeded limitn. Results > high limit for ADHLSTAT at some point. Result < low imit for ADLLSTAT at some point.
- 1 = Channel n is within the limitn. Result <= high limit for ADHLSTAT. Results >= low limit for ADLLSTAT.

9.8.8 ADC Zero Crossing Status Register (ADZCSTAT)

The Zero Crossing Status 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 is set to 11b (any edge change), then the ZCS0 bit is set to one. An interrupt is generated if the ZCI bit is set in ADCR1. A bit is can only be cleared by writing a value of one to that specific bit. These bits are *sticky*. Once set, they require a write to clear them. They are not automatically cleared by subsequent conversions.

BASE+$E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	ZCS9	ZCS8	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 9-15. ADC Zero Crossing Status Register (ADZCSTAT)

See Programmer's Sheets on Appendix page B- 60

9.8.8.1 Reserved—Bits 15–10

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

9.8.8.2 Zero Crossing Status—Bits 9–0

- 0 = Channel *n* has not crossed offset *n* value
- 1 = Channel *n* crossed offset *n* value

9.8.9 ADC Result Registers (ADRSLT0,...,ADRSLT9)

The 10 Result registers contain the converted results from a scan. The SAMPLE0 result is loaded into ADRSLT0, SAMPLE1 result in ADRSLT1, and so on.

ADC Result Register 0—Address: ADC_BASE + $0F
ADC Result Register 1—Address: ADC_BASE + $10
ADC Result Register 2—Address: ADC_BASE + $11
ADC Result Register 3—Address: ADC_BASE + $12
ADC Result Register 4—Address: ADC_BASE + $13
ADC Result Register 5—Address: ADC_BASE + $14
ADC Result Register 6—Address: ADC_BASE + $15
ADC Result Register 7—Address: ADC_BASE + $16
ADC Result Register 8—Address: ADC_BASE + $17
ADC Result Register 9—Address: ADC_BASE + $18

BASE+$F	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SEXT	RSLTn												0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-16. ADC Result Register (ADRSLT0–9)

See Programmer's Sheets on Appendix page B- 61

9.8.9.1 Sign Extend (SEXT)—Bit 15

SEXT is the sign extend bit of the result. SEXT set to one implies a negative result, and zero, a positive one. If positive results are required, then the respective Offset register must be set to a value of zero.

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

ADRSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the ADRSLT can be used directly. As a signed integer, the user has the option to right shift with sign extend (ASR) three places and interpret the number that way, 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 complement format. If it is a requirement of the application the result registers always be positive, it is necessary to set the Offset registers to zero.

The interpretation of the numbers programmed into the Limit and Offset registers (ADLLMT, ADHLMT, ADOFS) should match the application interpretation of the result register.

Each Result register may only receive writing when the ADC is in ADC-STOP mode, or when the ADCSTOP bit is set to one. It may also receive writing when the ADC is in a power-down mode, or when the PDN bit is set to one. This write operation is treated as if it came from the ADC analog core. Therefore, the limit checking, zero crossing, and the offset registers function

as if in a Normal mode. For example, if the ADCSTOP bit is set to one and the processor writes to ADRSLT5, the data written to the ADRSLT5 is muxed to the digital core, processed, and stored into ADRSLT5 as if the analog core had provided the data. The data written will be handled as shown by **Figure 9-10.**

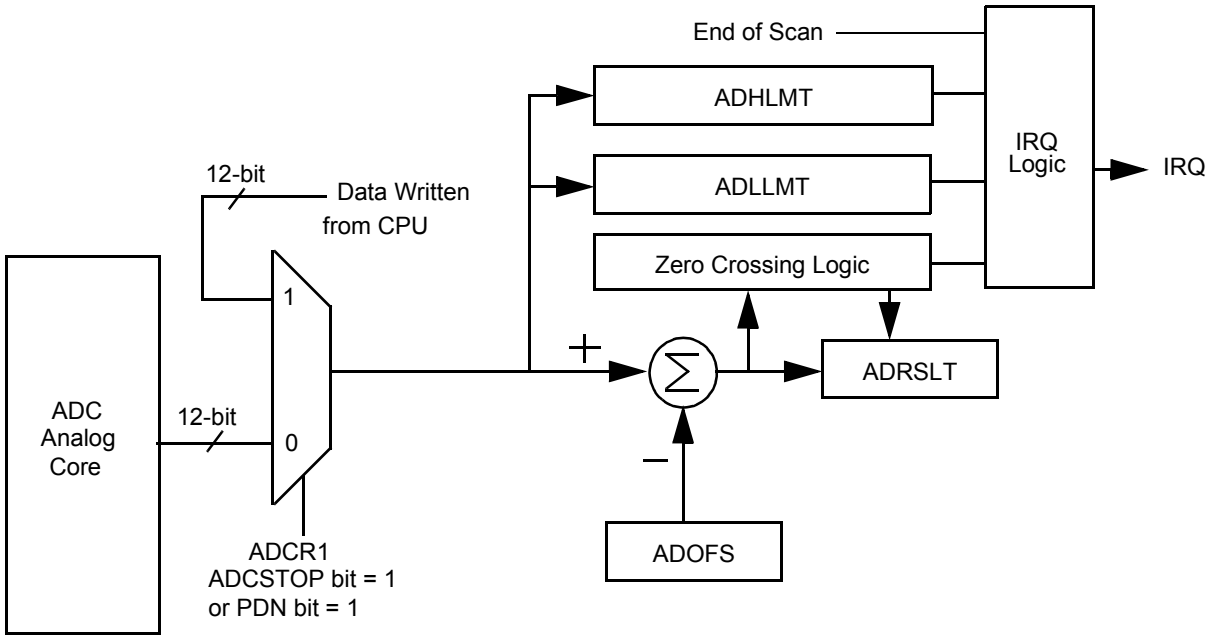


Figure 9-10. Result Register Data Manipulation

The result value sign is determined from the ADC unsigned result minus the respective Offset register value. If the Offset register is programmed with a value of zero, the result register value is unsigned and equals the cyclic converter unsigned result.

9.8.9.3 Reserved—Bits 2–0

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

9.8.10 Low and High Limit Registers (ADLLMT0–9,..,ADHLMT0–9)

The Limit Registers are programmed with the value the result is compared against. 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 $FFFF for the High Limit and $0000 for the Low Limit register. The power-up default is limit checking disabled.

ADC High Limit Register 0—Address: ADC_BASE + $19
ADC Low Limit Register 0—Address: ADC_BASE + $19
ADC Low Limit Register 1—Address: ADC_BASE + $1A
ADC Low Limit Register 2—Address: ADC_BASE + $1B
ADC Low Limit Register 3—Address: ADC_BASE + $1C
ADC Low Limit Register 4—Address: ADC_BASE + $1D
ADC Low Limit Register 5—Address: ADC_BASE + $1E
ADC Low Limit Register 6—Address: ADC_BASE + $1F
ADC Low Limit Register 7—Address: ADC_BASE + $20
ADC Low Limit Register 8—Address: ADC_BASE + $21
ADC Low Limit Regsiter 9—Address: ADC_BASE + $22

BASE+$19	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 9-17. ADC Low Limit Status Register (ADLLMTn)

See Programmer's Sheets on Appendix page B- 62

ADC High Limit Register 0—Address: ADC_BASE + $23
ADC High Limit Register 1—Address: ADC_BASE + $24
ADC High Limit Register 2—Address: ADC_BASE + $25
ADC High Limit Register 3—Address: ADC_BASE + $26
ADC High Limit Register 4—Address: ADC_BASE + $27
ADC High Limit Register 5—Address: ADC_BASE + $28
ADC High Limit Register 6—Address: ADC_BASE + $29
ADC High Limit Register 7—Address: ADC_BASE + $2A
ADC High Limit Register 8—Address: ADC_BASE + $2B
ADC High Limit Register 9—Address: ADC_BASE + $2C

BASE+$23	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

Figure 9-18. ADC High Limit Status Register (ADHLMTn)

See Programmer's Sheets on Appendix page B- 62

9.8.11 ADC Offset Registers (ADOFS0–9)

Value of the Offset Register is used to correct the ADC result before it is stored in the ADRSLT registers. The OFFSETn value is subtracted from the ADC result. In order to obtain unassigned results, the respective offset register should be programmed with a value of $0000, thus providing a result range of $0000 to $7FF8 (seen as 16-bit numbers in ADRSTn registers).

ADC Offset Register 0—Address: ADC_BASE + $2D
ADC Offset Register 1—Address: ADC_BASE + $2E
ADC Offset Register 2—Address: ADC_BASE + $2F
ADC Offset Register 3—Address: ADC_BASE + $30
ADC Offset Register 4—Address: ADC_BASE + $31
ADC Offset Register 5—Address: ADC_BASE + $32
ADC Offset Register 6—Address: ADC_BASE + $33
ADC Offset Register 7—Address: ADC_BASE + $34
ADC Offset Register 8—Address: ADC_BASE + $35
ADC Offset Register 9—Address: ADC_BASE + $36

BASE+$2D	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 9-19. ADC Offset Register (ADOFS0-9)

See Programmer's Sheets on Appendix page B- 63

Chapter 10
Serial Communications Interface (SCI)

10.1 Introduction

This chapter describes the Serial Communications Interface (SCI) module. The module allows asynchronous serial communications with peripheral devices and other controllers.

10.2 Features

- Full-duplex or Single-Wire Operation
- Standard mark/space Non-Return-to-Zero (NRZ) format
- 13-bit baud rate selection
- Programmable 8- or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter 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
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

10.3 Block Diagram

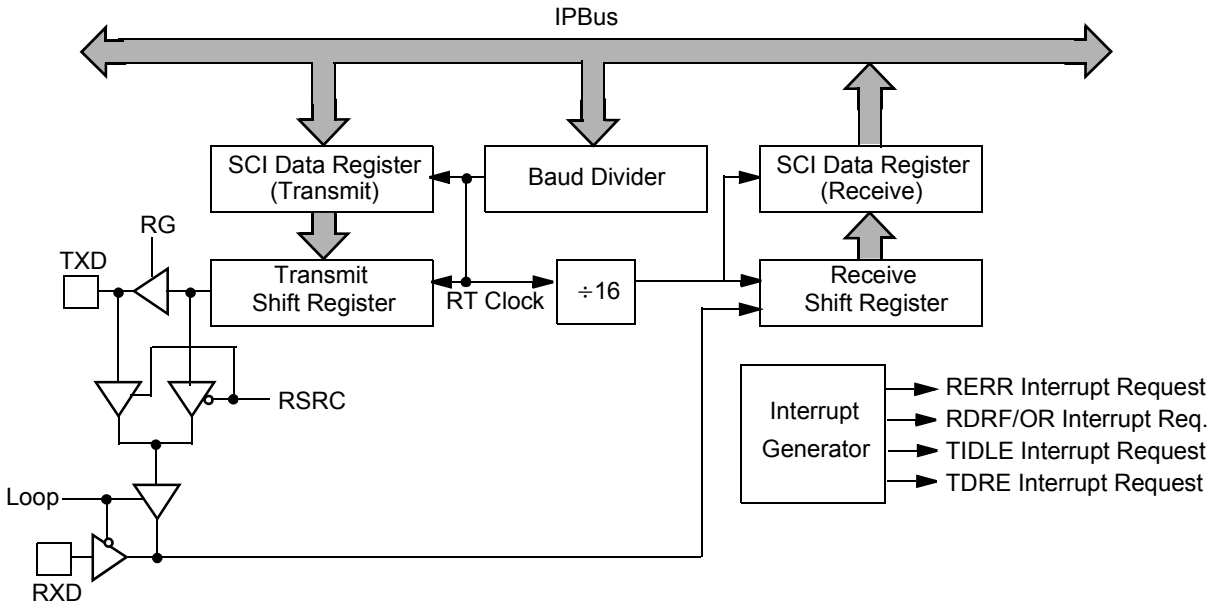


Figure 10-1. SCI Block Diagram

10.4 Functional Description

Figure 10-1 explains the SCI module structure. The SCI allows full duplex, asynchronous, Non-Return-to-Zero (NRZ) serial communication between the controller and remote devices, including other controllers. The SCI transmitter and receiver operate independently although they use the same baud rate generator. The controller monitors the status of the SCI, writes the data to be transmitted, and processes received data.

The SCI has one input signal and one output signal. Data transmits on the TXD pin and receives on the RXD pin. SCI Data Register (SCIDR) holds received bytes and bytes to be transmitted, actually consists of two physically different registers. To send software writes a byte to SCIDR; to receive, software reads a byte from SCIDR. However, if software has not read the first byte by the time the second byte is received, the second byte will be lost and Overrun (OR) flag will be set.

When initializing the SCI, be certain to set the proper peripheral enable bits in the General Purpose Input/Output (GPIO) registers as well as any pull-up enables if the SCI pins are multiplexed with GPIO pins.

10.4.1 Data Frame Format

The SCI uses the standard NRZ mark/space data frame format illustrated in **Figure 10-2.**

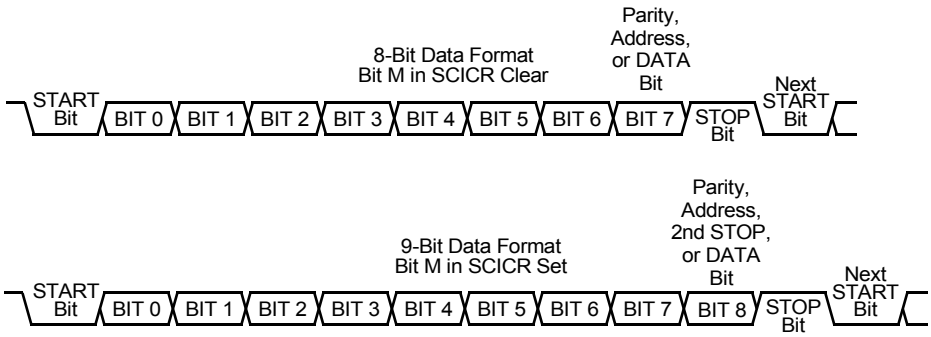


Figure 10-2. SCI Data Frame Formats

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 SCI Control Register (SCICR) configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Formats are provided in **Table 10-1.**

Table 10-1. 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 10.4.4.6, Receiver Wake-Up.

Setting the M bit configures the SCI for 9-bit data characters. A frame with nine data bits has a total of 11 bits. Formats are provided in **Table 10-2.**

Table 10-2. 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[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 10.4.4.6, Receiver Wake-Up.
2. The user must implement the second stop bit by setting the MSB of the data bits when transmitting and by masking the MSB when receiving.

10.4.2 Baud Rate Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. A value of 1 to 8191 written to the SBR bit in the SCI Baud Rate (SCIBR) register determines the module clock divisor. A value of zero disables the baud rate generator. The clock generated by SCIBR is called RT clock; a frequency of 16 times the baud rate. The RT clock is synchronized with the IPBus clock, driving the receiver. The RT 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 10-3 lists examples of achieving target baud rates with a module clock frequency of 40MHz.

Table 10-3. Example Baud Rates (Module Clock = 40MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
65	615,384.6	38,461.5	38,400	0.16
130	307,692.3	19,230.8	19,200	0.16
260	153,846.1	9,615.4	9,600	0.16
521	76,775.4	4,798.5	4,800	0.03
1042	38,387.7	2,399.2	2,400	0.03
2083	19,203.1	1,200.2	1,200	0.02
4167	9,599.2	600.0	600	0.01

Note: Maximum baud rate is IPBus clock rate divided by 16. System overhead may preclude processing the data at this speed

10.4.3 Transmitter

Figure 10-3 illustrates the transmitter functions block diagram with detailed discussion of the transmitter in the following sections.

10.4.3.1 Character Length

The SCI transmitter accommodates either 8- or 9-bit data characters, determined by the state of the M bit in the SCICR.

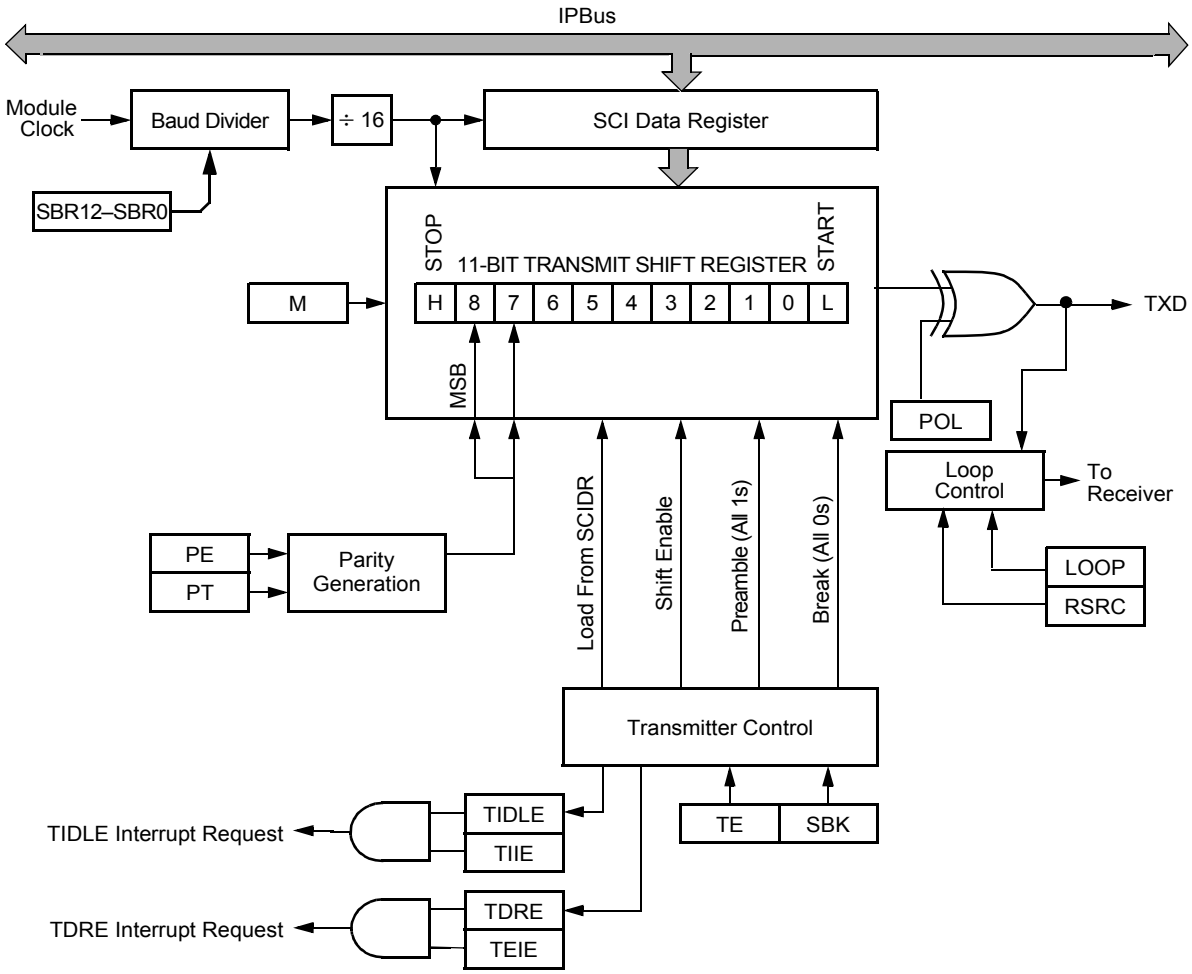


Figure 10-3. SCI Transmitter Block Diagram

10.4.3.2 Character Transmission

During an SCI transmission, the Transmit Shift register moves a frame out to the TXD pin. The SCI Data Register (SCIDR) is the buffer between the internal data bus and the Transmit Shift registers.

Modifying the Transmitter Enable (TE) bit from zero to one automatically loads the Transmit Shift register with a preamble containing all Logic 1s with no START, STOP, or PARITY bit. After the preamble shifts out, control logic automatically transfers the data from the SCIDR 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 SCISR becomes set when the SCIDR transfers a character to the Transmit Shift register. The TDRE flag indicates the SCIDR can accept new transmit data. If the TEIE bit in the SCICR is also set, the TDRE flag generates a transmitter empty interrupt request.

When the SCI Transmit Shift register is not transmitting a frame and TE = 1, the TXD pin goes to the idle condition, Logic 1. If software clears TE while a transmission is in progress, the frame in the SCI Transmit Shift register continues to shift-out the transmitter, then relinquishes control of the port I/O pin upon completion of the current transmission. This action causes the TXD pin to go into a HighZ state even if there is data pending in the SCIDR. 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 initiate a SCI transmission:

1. Enable the transmitter by writing a Logic 1 to the TE bit in the SCICR.
2. Wait for the TDRE flag to be set
3. Clear the TDRE flag by first reading the SCISR, then write to the SCIDR.
4. Repeat Steps 2 and 3 for each subsequent transmission.

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 SCIDR.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the Transmit Shift register.
3. Queue a preamble by clearing, then set the TE bit.
4. Write the first character of the second message to SCIDR.

10.4.3.3 Break Characters

Writing a Logic 1 to the Send Break (SBK) bit in the SCICR 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 SCICR. 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 finishes 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 SCI 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 SCI registers:

- Sets the Framing Error (FE) flag
- Sets the Receive Data Register Full (RDRF) flag
- Clears the SCI Data Register (SCIDR)
- May set the Overrun (OR) flag, Noise Flag (NF), Parity Error (PE) flag, or the Receiver Active Flag (RAF). Please see SCISR in **Section 10.6.3.**

10.4.3.4 Preambles

A preamble contains all Logic 1s with no START, STOP, or PARITY bit. A preamble length depends on the M bit in the SCICR. The preamble is a synchronizing mechanism initiating the first transmission begun after modifying the TE bit from zero to one.

To queue a preamble between two transmissions:

1. After writing the last character of the first transmission to the Transmit register, wait for TDRE flag to be set.
2. When the TDRE flag becomes set, clear and set the TE bit. This will queue a preamble after the last character of the first transmission.
3. After setting the TE bit, immediately write the first character of the second transmission to the SCIDR.

10.4.4 Receiver

Figure 10-4 illustrates the block diagram of the SCI receiver function.

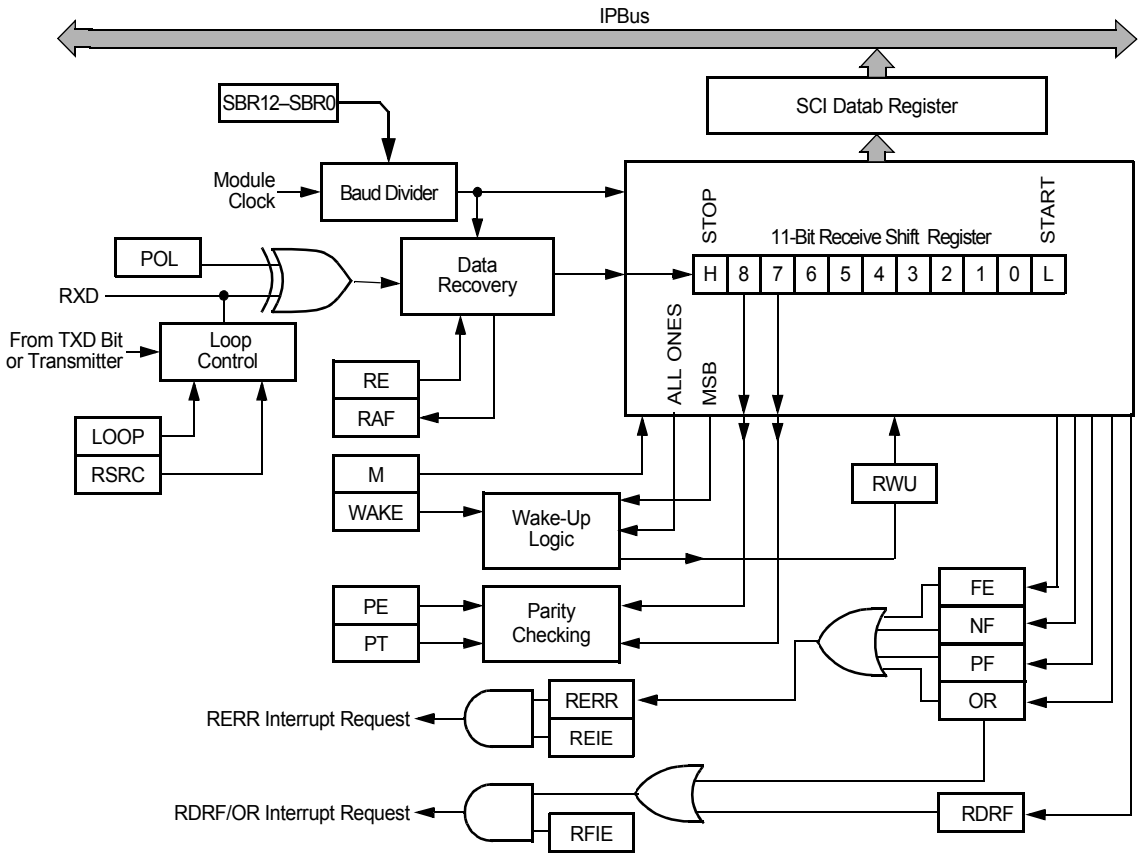


Figure 10-4. SCI Receiver Block Diagram

10.4.4.1 Character Length

The SCI receiver can accommodate either 8- or 9-bit data characters determined by the state of the M bit in the SCICR.

10.4.4.2 Character Reception

During an SCI reception, the Receive Shift register alters a frame in from the RXD pin. The data is read from the SCIDR.

After a complete frame shifts into the Receive Shift register, the data portion of the frame transfers to the SCIDR. The Receive Data Register Full (RDRF) flag in the SCISR is set, indicating the received character can be read. If the Receive Full Interrupt Enable (RFIE) bit in the SCICR is also set, the RDRF flag generates an RDRF interrupt request.

10.4.4.3 Data Sampling

The receiver samples the RXD pin at the Rate Tolerance (RT) clock rate. To adjust the baud rate mismatch, the RT clock illustrated in **Figure 10-5,** is resynchronized:

- After every START bit
- After the receiver detects a data bit change from Logic 1 to 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.

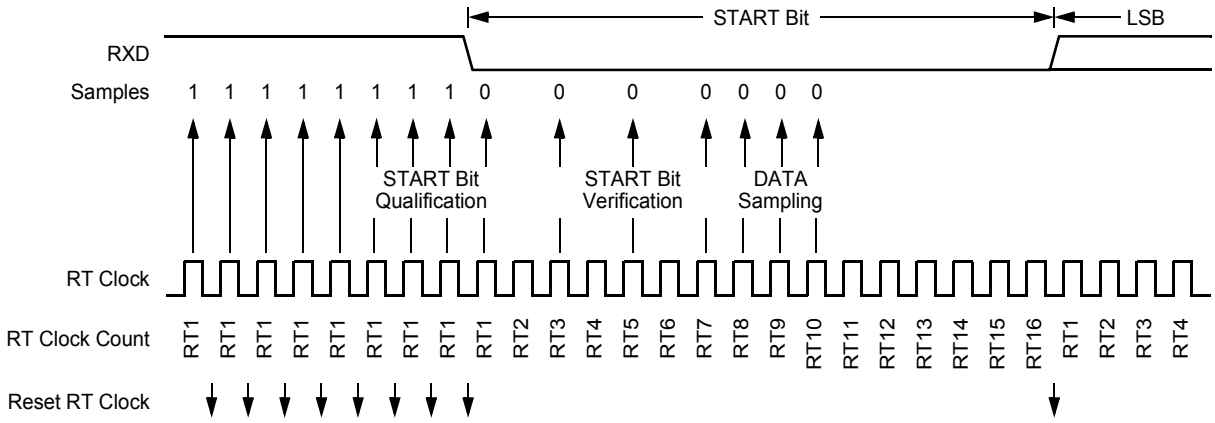


Figure 10-5. Receiver Data Sampling

To verify the START bit and detect noise, data recovery logic takes samples at RT3, RT5, and RT7. **Table 10-4** summarizes the results of the START bit verification samples and the Noise Flag (NF). A majority vote of the three samples is used as the value of the bit.

Table 10-4. 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 two of three samples are 0s (not all 0s), Noise Flag (NF) is set. 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. A majority vote of the three samples is used as the value of the bit. **Table 10-5** summarizes the results of the DATA bit samples. If all three samples are not the same, Noise Flag (NF) is set.

Table 10-5. 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 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. A majority vote of the three samples is used as the value of the bit. **Table 10-6** summarizes the results of the STOP bit samples. If all three samples are not the same, Noise Flag (NF) is set. If STOP bit detecting fails, Framing Error Flag (FE) is set.

Table 10-6. 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

10.4.4.4 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 SCISR. Reception of 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. The FE flag inhibits further data reception until it is cleared. The FE flag is cleared by reading the SCISR, thereafter write any value to the SCISR.

10.4.4.5 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 either Noise Error, Framing Error, or both.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames may correct misalignments between transmitter bit times and receiver bit times. The worst case is the data is all 0s or 1s without resynchronization occuring during the frame transmit.

10.4.4.5.1 Slow Data Tolerance

Figure 10-6 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.

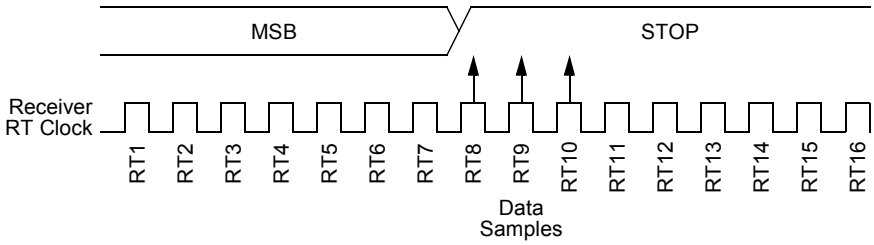


Figure 10-6. Slow Data

For an 8-bit (all 0s) data character, data sampling of the STOP bit takes the receiver:

$$9\text{-bit} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 154 \text{ RT cycles}$$

With the misaligned character, illustrated in Figure 10-6, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

$$9\text{-bit} \times 16 \text{ RT cycles} + 3 \text{ RT cycles} = 147 \text{ RT cycles}$$

The maximum percentage difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

For a 9-bit (all 0s) data character, data sampling of the STOP bit takes the receiver:

$$\left|\frac{154 - 147}{154}\right| \times 100 = 4.54\%$$

10-bit × 16 RT cycles + 10 RT cycles = 170 RT cycles

With the misaligned character illustrated in **Figure 10-6,** 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

The maximum percentage difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$\left|\frac{170 - 163}{170}\right| \times 100 = 4.12\%$$

10.4.4.5.2 Fast Data Tolerance

Figure 10-7 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.

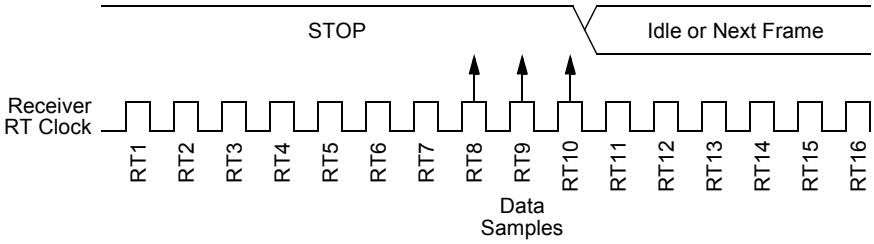


Figure 10-7. Fast Data

For an 8-bit (all 0s or 1s) 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 10-7,** 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:

$$\text{10-bit} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 170 \text{ RT cycles}$$

With the misaligned character, illustrated in **Figure 10-7,** the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

$$\text{11-bit} \times 16 \text{ RT cycles} = 176 \text{ RT cycles}$$

The maximum percentage difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$\left| \frac{170 - 176}{170} \right| \times 100 = 3.53\%$$

10.4.4.6 Receiver Wake-Up

In order for the SCI to ignore transmissions intended only for other receivers in multiple receiver systems, the receiver can be put into a standby state. Setting the Receiver Wake-Up (RWU) bit in the SCICR 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 SCICR determines how the SCI is brought out of the standby state to process an incoming message. The WAKE bit enables either Idle Input Line Wake-Up or Address Mark Wake-Up:

- Idle Input Line Wake-Up (WAKE = 0)—In this wake-up method, an idle condition on the RXD pin clears the RWU bit, waking up the SCI. Idle Input Line Wake-Up requires messages be separated by at least one preamble, and no message contains preambles. The initial frame or frames of every message contains 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.

  The receiver-waking preamble does not set the Receiver Idle (RIDLE) bit or the Receive Data Full Register (RDRF) flag.

  With the WAKE bit clear, setting the RWU bit after the RXD pin has been idle can cause the receiver to wake-up immediately.

- Address Mark Wake-Up (WAKE = 1)—In this wake-up method, a Logic 1 in the MSB position of a frame clears the RWU bit, awakening the SCI. The Logic 1 in the MSB position marks a frame as an address frame containing addressing information. The address frame

also sets the RDRF bit in the SCISR. 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. Address Mark Wake-Up allows messages to contain preambles but it requires the MSB to be reserved for use in address frames.

10.5 Special Operating Modes

Table 10-7 summarizes how to configure for Normal Operation, Loop Back, or Single-Wire Operation as described in Section 10.5.1 and Section 10.5.2.

Table 10-7. 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

10.5.1 Single-Wire Operation

Normally, the SCI uses two pins for transmitting and receiving. In the Single-Wire Operation, the RXD pin is disconnected from the SCI and is available for other peripherals, illustrated in Figure 10-8. The SCI uses the TXD pin for both receiving and transmitting.

Setting the TE bit in the SCICR enables the transmitter, configuring TXD as the output for transmitted data. Clearing the TE bit disables the transmitter, configuring TXD as the input for received data.

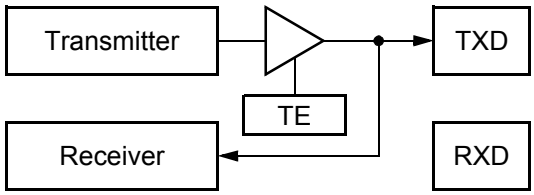


Figure 10-8. Single-Wire Operation (LOOP = 1, RSRC = 1)

Enable Single-Wire Operation by setting the LOOP bit and the Receiver Source (RSRC) bit in the SCICR. 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. To enable Receiver, Receiver Enable (RE) bit in the SCICR must be set.

10.5.2 Loop Operation

In Loop Operation, the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI and is available as a GPIO pin. Please see **Figure 10-9.**

Setting the TE bit in the SCICR enables the transmitter, connecting the transmitter output to the TXD pin. Clearing the TE bit disables the transmitter, disconnecting the transmitter output from the TXD pin.

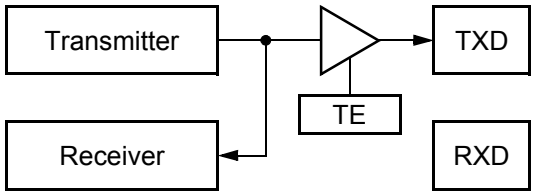


Figure 10-9. Loop Operation (LOOP = 1, RSRC = 0)

Enable Loop Operation by setting the LOOP bit and clearing the RSRC bit in the SCICR. 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 SCICR must be set.

10.5.3 Low-Power Options

10.5.3.1 Run Mode

Clearing the Transmitter Enable (TE) or Receiver Enable (RE) bits in the SCICR reduces power consumption in Run mode. SCI registers are still accessible when TE or RE bits are cleared, but clocks to the SCI module are disabled.

10.5.3.2 Wait Mode

SCI operation in Wait mode depends on the state of the SWAI bit in the SCICR.

- If SWAI is clear, SCI operates normally when CPU is in Wait mode.
- If SWAI is set, SCI clock generation ceases and the SCI module enters a power conservation state when the CPU is in Wait mode. Setting SWAI does not affect the state of the RE bit or the TE bit.

When 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 processor out of Wait mode.

When SWAI is set the SCI module cannot generate interrupt requests during Wait mode.

Any enabled SCI interrupt request can bring the processor out of Wait mode as long as SWAI is clear.

10.5.3.3 Stop Mode

The SCI is inactive in STOP mode for reduced power consumption. The Stop instruction does not affect the register states. SCI operation resumes after an external interrupt brings the processor out of Stop mode.

10.6 Register Descriptions

Table 10-8. SCI Memory Map

Device	Name	Address
826	SCI0_BASE	$1160
	SCI1_BASE	$1170
827	SCI2_BASE	$1180

There are four accessible registers on SCI described in **Table 10-9** and summarized in **Table 10-10.** A register address 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 10-9. SCI Register Summary

Address Offset	Register Acronym	Register Description	Access Type	Chapter Location
Base + $0	SCIBR	Baud Rate Register	Read/Write	Section 10.4.2
Base + $1	SCICR	Control Register	Read/Write	Section 10.6.2
Base + $2	SCISR	Status Register	*Read-Only*	Section 10.6.3
Base + $3	SCIDR	Data Register	Read/Write	Section 10.6.4

Bit fields of each of the four registers are illustrated in **Figure 10-10.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	SCIBR	R	0	0	0	SBR												
		W																
$1	SCICR	R	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RIE	REIE	TE	RE	RWU	SBK
		W																
$2	SCISR	R	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0	0	0	0	0	0	0	RAF
		W																
$3	SCIDR	R	0	0	0	0	0	0	0	RECEIVE DATA								
		W								TRANSMIT DATA								

R 0 = Read as 0
W = Reserved

Figure 10-10. SCI Register Map

10.6.1 SCI Baud Rate Register (SCIBR)

This register can be read at any time. Bits 12 through 0 can be written at any time, but bits 15 through 13 are reserved.

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

Figure 10-11. SCI Baud Rate Register (SCIBR)

See Progammer's Sheets on Appendix page B - 101

The count in this register determines the baud rate of the SCI. The formula for calculating baud rate is:

$$\text{SCI Baud Rate} = \frac{\text{IPBus Clock}}{16 \times \text{SBR}}$$

See **Section 10.4.2** for more details and examples.

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.

10.6.1.1 Reserved—Bits 15–13

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

10.6.1.2 SCI Baud Rate (SBR)—Bits 12–0

Contents of the Baud Rate register has a value of 1 to 8191.

10.6.2 SCI Control Register (SCICR)

The SCI Control Register (SCICR) 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 10-12. SCI Control Register (SCICR)

See Progammer's Sheets on Appendix page B - 102

10.6.2.1 Loop Select Bit (LOOP)—Bit 15

This bit enables loop operation. Please see **Section 10.5.2.** The loop operation disconnects the RXD pin from the SCI and the transmitter output goes into the receiver input.

- 0 = Normal operation enabled
- 1 = Loop operation enabled

10.6.2.2 Stop in Wait Mode (SWAI)—Bit 14

The SWAI bit disables the SCI in Wait mode. See **Section 10.5.3.2.**

- 0 = SCI enabled in Wait mode
- 1 = SCI disabled in Wait mode

10.6.2.3 Receiver Source (RSRC)— Bit 13

When LOOP = 1, the RSRC bit determines the internal feedback path for the receiver. See **Section 10.5.1** and **Section 10.5.2** for more details.

- 0 = Receiver input internally connected to transmitter output
- 1 = Receiver input connected to TXD pin

10.6.2.4 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

10.6.2.5 Wake-Up Condition (WAKE)—Bit 11

This bit determines which condition wakes up the SCI.

- 0 = Idle Line Wake-Up
- 1 = Address Mark Wake-Up (a Logic 1 in the MSB position of a receive data character)

Note: Address Mark Wake-Up is not a valid option when the parity function is enabled (PE = 1) since the ADDRESS bit and the PARITY bit both occupy the MSB.

10.6.2.6 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.

10.6.2.7 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

Note: Address Mark Wake-Up (WAKE = 1) is not a valid option when the parity function is enabled because the ADDRESS bit and the PARITY bit both occupy the MSB.

10.6.2.8 Parity Type (PT)—Bit 8

This bit determines whether the SCI generates and checks for even parity or odd parity of the data bits. With *even parity*, an *even* number of *ones, clears* the PARITY bit, while an *odd* number of *ones, sets* the PARITY bit. However, with *odd parity*, an *odd* number of *ones, clears* the PARITY bit, while an *even* number of *ones, sets* the PARITY bit.

- 0 = Even parity
- 1 = Odd parity

10.6.2.9 Transmitter Empty Interrupt Enable (TEIE)—Bit 7

This bit enables the TDRE flag to generate interrupt requests.

- 0 = TDRE interrupt requests disabled
- 1 = TDRE interrupt requests enabled

10.6.2.10 Transmitter Idle Interrupt Enable (TIIE)—Bit 6

This bit enables the TIDLE flag to generate interrupt requests.

- 0 = TIDLE interrupt requests disabled
- 1 = TIDLE interrupt requests enabled

10.6.2.11 Receiver Full Interrupt Enable (RFIE)—Bit 5

This bit enables the RDRF flag, or the OR flag to generate interrupt requests.

- 0 = RDRF and OR interrupt requests disabled
- 1 = RDRF and OR interrupt requests enabled

10.6.2.12 Receive Error Interrupt Enable (REIE)—Bit 4

This bit enables the receive error flags (NF, PF, FE, 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

10.6.2.13 Transmitter Enable (TE)—Bit 3

This bit enables the SCI transmitter and configures the TXD pin as the SCI transmitter output. The TE bit can be used to queue an idle preamble.

- 0 = Transmitter disabled
- 1 = Transmitter enabled

10.6.2.14 Receiver Enable (RE)—Bit 2

This bit enables the SCI Receiver.

- 0 = Receiver disabled
- 1 = Receiver enabled

10.6.2.15 Receiver Wake-Up (RWU)—Bit 1

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 10.4.4.6** for a description of Receive Wake-Up.

- 0 = Normal operation
- 1 = Standby state

10.6.2.16 Send Break (SBK)—Bit 0

Setting SBK sends one break character (all Logic 0s, include START, DATA, STOP). As long as SBK is set, transmitter sends uninterrupted break characters.

- 0 = No break characters
- 1 = Transmit break characters

10.6.3 SCI Status Register (SCISR)

This register can be read at anytime; however, it cannot be modified by writing. Writes clear flags.

Base + $2	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	0	0	0	RAF
Write																
RESET	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-13. SCI Status Register (SCISR)

See Progammer's Sheets on Appendix page B - 104

10.6.3.1 Transmit Data Register Empty Flag (TDRE)—Bit 15

This bit is set when the Transmit Shift register receives a character from the SCIDR. Clear TDRE by reading SCISR, then write to the SCIDR.

- 0 = No character transferred to Transmit Shift register
- 1 = Character transferred to Transmit Shift register; TDRE

10.6.3.2 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 SCISR, then write to the SCIDR.

- 0 = Transmission in progress
- 1 = No transmission in progress

10.6.3.3 Receive Data Register Full Flag (RDRF)—Bit 13

This bit is set when the data in the Receive Shift register transfers to the SCIDR. Clear RDRF by reading the SCISR, then read the SCIDR.

- 0 = Data not available in SCIDR
- 1 = Received data available in SCIDR

10.6.3.4 Receiver Idle Line Flag (RIDLE)—Bit 12

This bit is set when ten consecutive Logic 1s (if M = 0) or eleven consecutive Logic 1s (if M = 1) appear on the receiver input. Once the RIDLE flag is cleared by the receiver detecting a Logic 0, a valid frame must again set the RDRF flag before an idle condition can set the RIDLE flag.

- 0 = Receiver input is either active now or has never become active since the RIDLE flag was last cleared by reset
- 1 = Receiver input has become idle (after receiving a valid frame)

Note: When the Receiver Wake-Up (RWU) bit is set, an idle line condition does not set the RIDLE flag.

10.6.3.5 Overrun Flag (OR)—Bit 11

This bit is set when software fails to read the SCIDR before the Receive Shift register receives the next frame. The data in the Shift register is lost, but the data already in the SCIDR is not affected. Clear OR by reading the SCISR, then write the SCISR with any value.

- 0 = No overrun
- 1 = Overrun

10.6.3.6 Noise Flag (NF)—Bit 10

This bit is set when the SCI 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 SCISR, then write the SCISR with any value.

- 0 = No noise
- 1 = Noise

10.6.3.7 Framing Error Flag (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. FE inhibits further data reception until it is cleared. Clear FE by reading the SCISR, then write the SCISR with any value.

- 0 = No framing error
- 1 = Framing error

10.6.3.8 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 SCISR, then write the SCISR with any value.

- 0 = No parity error
- 1 = Parity error

10.6.3.9 Reserved—Bits 7–1

These bits are reserved or not implemented. They are read as zero and cannot be modified by writing.

10.6.3.10 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

10.6.4 SCI Data Register (SCIDR)

The SCIDR can be read and modified at any time. Reading accesses the SCI Receive Data register. Writing to the register accesses the SCI Transmit Data register.

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	RECEIVE DATA								
Write								TRANSMIT DATA								
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-14. SCI Data Register (SCIDR)

See Progammer's Sheets on Appendix page B - 105

10.6.4.1 Reserved—Bits 15–9

These bits are reserved or not implemented. They are read as zero and cannot be modified by writing.

10.6.4.2 Receive/Transmit Data—Bits 8–0

Writing to these bits loads the transmit data. Reading these bits accesses the receive data.

Note: When configured for 8-bit data, bits 7 to 0 contain the data received.

10.7 Clocks

All timing is derived from the IPBus clock, operating at the system clock rate. Please see **Section 10.4.2** for a description of how the data rate is determined.

10.8 Resets

Any system reset completely resets the SCI.

10.9 Interrupts

Table 10-10. SCI Interrupt Sources

Interrupt Source	Flag	Local Enable	Description
Transmitter	TDRE	TEIE	Transmit Empty
	TIDLE	TIIE	Transmit Idle
Receiver	RDRF	RFIE	Receive Full
	OR		
	FE	REIE	Receive Error
	PE		
	NF		
	OR		

10.9.1 Transmitter Empty Interrupt

This interrupt is enabled by setting the TEIE bit of the SCICR. When this interrupt is enabled, an interrupt is generated when data is transferred from the SCIDR to the Transmit Shift register.

10.9.2 Transmitter Idle Interrupt

This interrupt is enabled by setting the TIIE bit of the SCICR. This interrupt indicates the TIDLE flag is set and the transmitter is no longer sending data, preamble, or break characters. The interrupt service routine should initiate a preamble, a break, write a data character to the SCIDR or disable the transmitter.

10.9.3 Receiver Full Interrupt

This interrupt is enabled by setting the RFIE bit of the SCICR. This interrupt indicates either receive data is available in the SCIDR or a data overrun occurred. The interrupt service reoutine should read SCISR to determine which of RDRF flag, OR flag, or both were set.

10.9.4 Receive Error Interrupt

This interrupt is enabled by setting the REIE bit of the SCICR. This interrupt indicates any of the listed errors was detected by the receiver:

1. Noise Flag (NF) set
2. Parity Error Flag (PF) set
3. Framing Error (FE) flag set
4. Overrun (OR) flag set

The interrupt service routine should read the SCISR to determine which of the error flags was set. The error flag is cleared by writing anything to the SCISR. Then the appropriate action should be taken by the software to handle the error condition.

Chapter 11
Serial Peripheral Interface (SPI)

11.1 Introduction

This chapter describes the Serial Peripheral Interface (SPI) module. The module allows full-duplex, synchronous, serial communication between the 16-bit controller and peripheral devices, including other 16-bit controllers.

Characteristics of the SPI module include:

- Full-duplex operation
- Master and Slave modes
- Double-buffered operation with separate transmit and receive registers
- Programmable length transmissions (two to 16 bits)
- Programmable transmit and receive shift order (MSB first or last bit transmitted)
- Eight Master mode frequencies (maximum = module clock ÷ 2)
- Maximum Slave mode frequency = module clock
- Clock ground for reduced Radio Frequency (RF) interference
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts
  - SPI Receiver Full (SPRF)
  - SPI Transmitter Empty (SPTE)
- Mode Fault Error flag interrupt capability

Note: Throughout this chapter, there are references to the SPI module clock. The SPI module clock is the IPBus clock.

11.2 Block Diagram

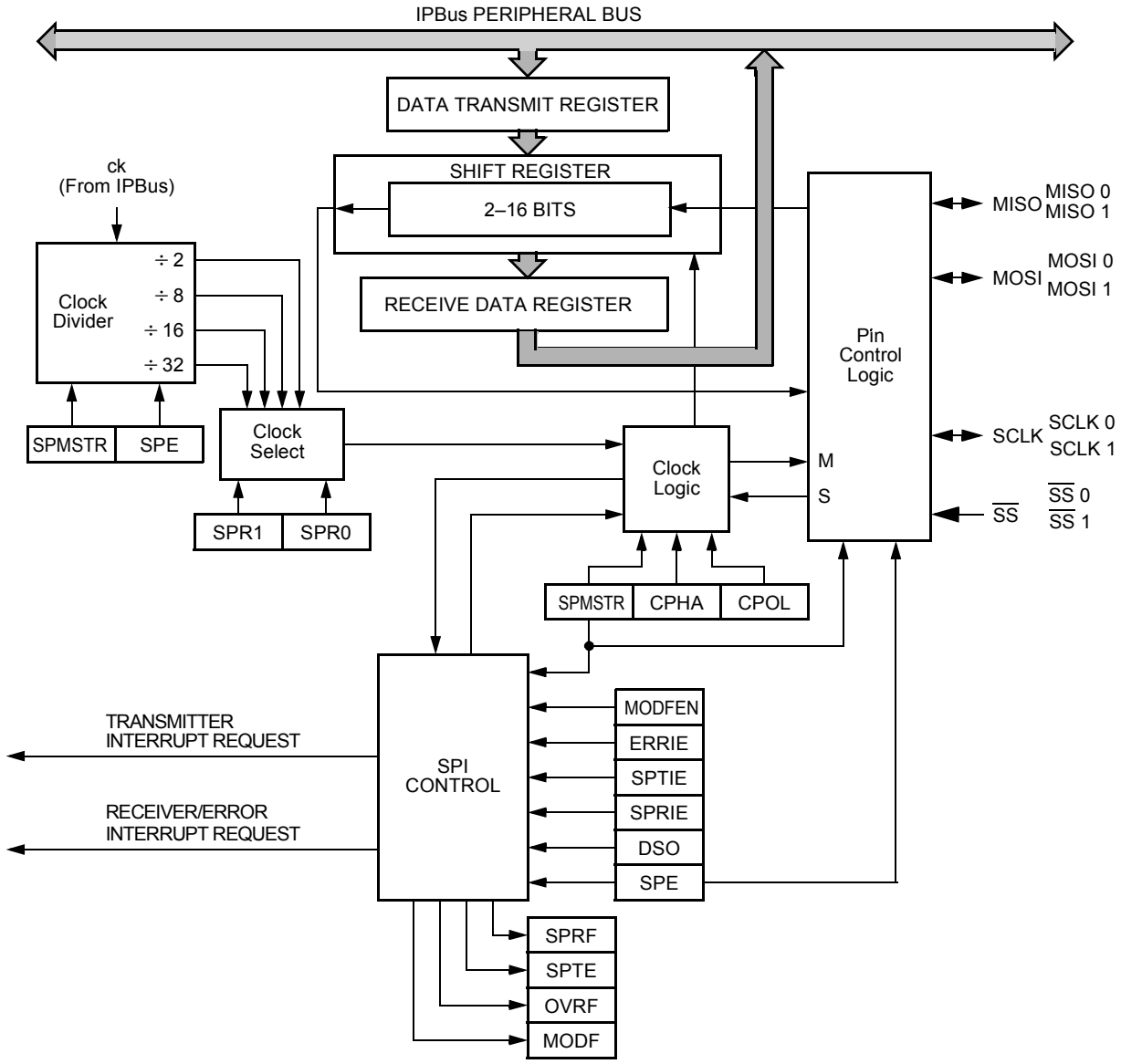


Figure 11-1. SPI Block Diagram

11.3 Operating Modes

The SPI has two operating modes:

- Master
- Slave

An operating mode is selected by the SPMSTR bit in the SPI Status and Control Register (SPSCR) as follows:

- SPMSTR = 0 Slave mode
- SPMSTR = 1 Master mode

Note: The SPMSTR bit should be configured before enabling the SPI (setting the SPE bit in the SPSCR). The master SPI should be enabled before enabling any slave SPI. All slave SPIs should be disabled before disabling the master SPI.

11.3.1 Master Mode

The SPI operates in Master mode when the SPI master bit, SPMSTR, is set. Only a master SPI module can initiate transmissions. With the SPI enabled, software begins the transmission from the master SPI module by writing to the SPI Data Transmit Register (SPDTR). If the Shift register is empty, the data immediately transfers to the Shift register, setting the SPI Transmitter Empty (SPTE) bit. The data begins shifting out on the Master Out/Slave In (MOSI) pin under the control of the SPI Serial Clock (SCLK).

The SPR[2:0] bits in the SPI Status and Control Register (SPSCR) control the Baud Rate Generator, determining the speed of the Shift register. The Baud Rate Generator of the master also controls the Shift register of the slave peripheral via the SCLK pin.

As the 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 SPI Receiver Full (SPRF) bit in the SPSCR becomes set. At the same time the SPRF becomes set, the data from the slave transfers to the SPI Data Receive Register (SPDRR). In a normal operation, SPRF signals the end of a transmission. Software clears the SPRF by reading the SPSCR with SPRF set and then reading the SPDRR. Writing to the SPDTR clears the SPTE bit.

Figure 11-2 is an example configuration for a Full-Duplex Master-Slave Configuration. Having the Slave Select ($\overline{SS}$) bit of the master 16-bit controller held high is only necessary if MODFEN = 1. Tying the Slave 16-bit controller $\overline{SS}$ bit to ground should only be executed if CPHA = 1.

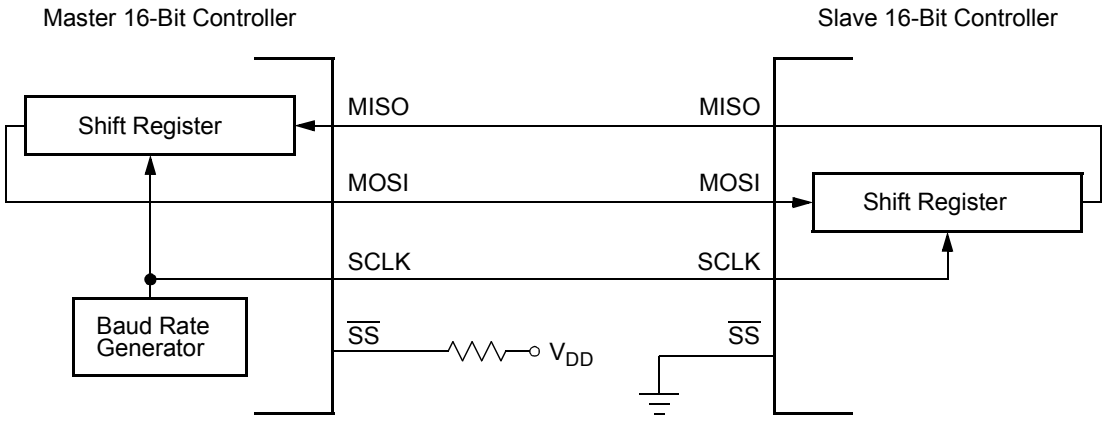


Figure 11-2. Full Duplex Master/Slave Connections

11.3.2 Slave Mode

The SPI operates in Slave mode when the SPMSTR bit is cleared. While in Slave mode, the SCLK pin acts as the input for the serial clock from the master 16-bit controller. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be at Logic 0. $\overline{SS}$ must remain low until the transmission is complete or a Mode Fault Error occurs.

Note: The SPI must be enabled (SPE = 1) for slave transmissions to be received.

Note: Data in the transmitter Shift register will be unaffected by SCLK transitions in the event the SPI is operating as a slave but is deselected.

In a slave SPI module, data enters the Shift register under the control of the Serial Clock, (SCLK), from the master SPI module. After a full length data transmission enters the Shift register of a slave SPI, it transfers to the SPI Data Receive Register (SPDDR) and the SPI Receiver Full (SPRF) bit in the SPSCR is set. If the SPI Receive Interrupt Enable (SPRIE) bit in the SPSCR is set, a Receive Interrupt is also generated. To prevent an overflow condition, slave software must read the SPDRR before another full length data transmission enters the Shift register.

The maximum frequency of the SCLK for the SPI configured as a slave is the module clock ÷ 2, or half the module clock. Frequency of the SCLK for the SPI configured as a slave does not have to correspond to any particular SPI baud rate. The baud rate only controls the speed of the SCLK generated by the SPI configured as a master. Therefore, the frequency of the SCLK for a SPI configured as a slave can be any frequency less than or equal to half of the module clock.

When the master SPI starts a transmission, the data in the slave Shift register begins shifting out on the MISO pin. The slave can load its Shift register with new data for the next transmission by writing to its SPDTR. The slave must write to its SPDTR at least one bus cycle before the master

starts the next transmission. Otherwise, the data already in the slave Shift register shifts out on the MISO pin. Data written to the Slave Shift register during a transmission remains in a buffer until the end of the transmission.

When the 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: SCLK must be in the proper idle state (depends on setting of CPOL bit) before the slave is enabled to prevent SCLK from appearing as a clock edge.

11.4 Pin Descriptions

There are four external SPI pins. Each is summarized in **Table 11-1.**

Table 11-1. External I/O Signals

Signal Name	Description	Direction
MISO	Master-In Slave-Out Pad Pin	Bi-Directional
MOSI	Master-Out Slave-In Pad Pin	Bi-Directional
SCLK	Serial Clock Pad Pin	Bi-Directional
$\overline{SS}$	Slave Select Pad Pin (Active Low)	Input

11.4.1 Master In/Slave Out (MISO)

MISO is one of the two SPI module pins dedicated to transmit serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin. The slave SPI 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 SPI is configured as a slave. The SPI is configured as a slave when the SPMSTR bit, discussed in **Section 11.8.1,** is Logic 0 and its $\overline{SS}$ pin is at Logic 0. To support a multiple slave system, a Logic 1 on the $\overline{SS}$ pin puts the MISO pin in a High Impedance state.

11.4.2 Master Out/Slave In (MOSI)

MOSI is the other SPI module pin dedicated to transmit serial data. In full duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin. The slave SPI simultaneously receives data from its MOSI pin and transmits data on its MISO pin.

11.4.3 Serial Clock (SCLK)

The serial clock synchronizes data transmission between master and slave devices. In a master 16-bit controller, the SCLK pin is the clock output. In a slave 16-bit controller, the SCLK pin is the clock input. In full duplex operation, the master and slave 16-bit controller exchange data in the same number of clock cycles as the number of bits of transmitted data.

11.4.4 Slave Select ($\overline{SS}$)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For a SPI configured as a slave, the $\overline{SS}$ is used to select a slave. When the Clock Phase (CPHA) bit in the SPSCR is cleared, the $\overline{SS}$ is used to define the start of a transmission, so it must be toggled high and low between each full length data transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format as illustrated in **Figure 11-4.**

When a SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. The MODFEN bit can prevent the state of the $\overline{SS}$ from creating a MODF error.

Note: A Logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. The slave SPI ignores all incoming SCLK clocks, even if it was already in the middle of a transmission. A Mode Fault occurs if the $\overline{SS}$ pin changes state during a transmission.

When a SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SCLK. For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SCLK register must be set.

Table 11-2. SPI I/O Configuration

SPE	SPMSTR	MODFEN	SPI Configuration	State of $\overline{SS}$ Logic
0	x	x	Not Enabled	$\overline{SS}$ ignored by SPI
1	0	x	Slave	Input-only to SPI
1	1	0	Master without MODF	$\overline{SS}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

x = Don’t care

11.5 Transmission Formats

During a SPI 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 SPI device; slave devices not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

11.5.1 Data Transmission Length

The SPI can support data lengths from two to 16 bits. This can be configured in the Data Size Register (SPDSR). When the data length is less than 16 bits, the Receive Data register will pad the upper bits with zeros.

Note: Data can be lost if the data length is not the same for both master and slave devices.

11.5.2 Data Shift Ordering

The SPI can be configured to transmit or receive the MSB of the desired data first or last. This is controlled by the Data Shift Order (DSO) bit in the SPSCR. Regardless which bit is transmitted or received first, the data shall always be written to the SPDTR and read from the Receive Data Register (SPDRR) with the LSB in bit zero and the MSB in the correct position, depending on the data transmission size.

11.5.3 Clock Phase and Polarity Controls

Software can select any of four combinations of Serial Clock (SCLK) phase and polarity using two bits in the SPSCR. The Clock Polarity is specified by the (CPOL) control bit. In turn, it selects an active high or low clock and has no 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 SPI 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 the CPOL bit or the CPHA bit, disable the SPI by clearing the SPI Enable (SPE) bit.

11.5.4 Transmission Format When CPHA = 0

Figure 11-3 exhibits a SPI 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:

1. CPOL = 0
2. CPOL = 1

The diagram may be interpreted as a master or slave timing diagram since the Serial Clock (SCLK), Master In/Slave Out (MISO), and Master Out/Slave In (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master.

The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at Logic 0, because only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown, but it is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or a Mode Fault Error will occur. When CPHA = 0, the first SCLK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SCLK edge and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each full length data transmitted as depicted in Figure 11-4.

Note: Figure 11-3 assumes 16-bit data lengths and the MSB shifted out first.

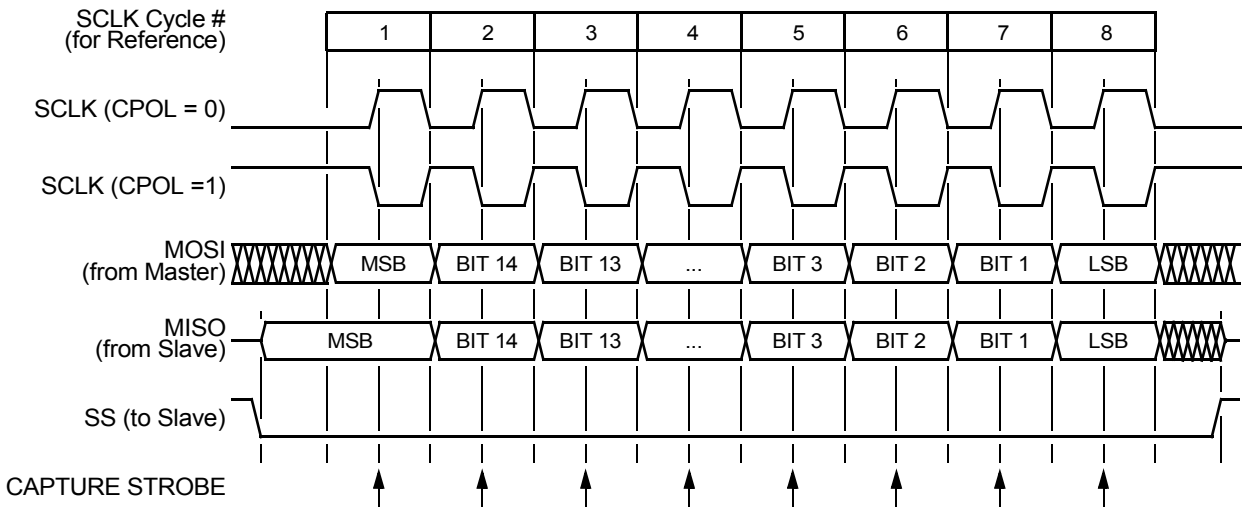


Figure 11-3. Transmission Format (CPHA = 0)

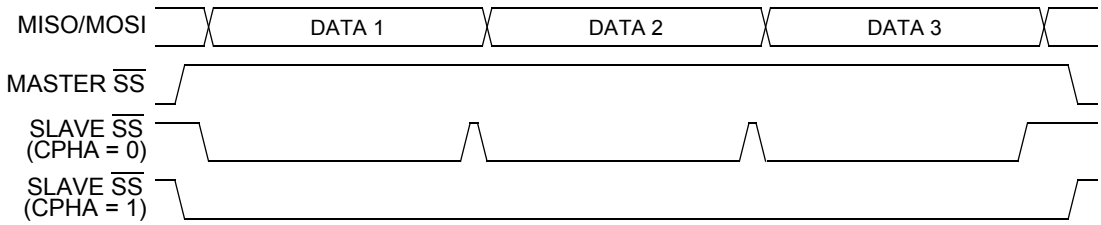


Figure 11-4. CPHA/SS Timing

When CPHA = 0 for a slave, the falling edge of SS indicates the beginning of the transmission. This causes the SPI 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 SPDTR. Therefore, the SPI Data register of the slave must be loaded with transmit data before the falling edge of SS. Any data written after the falling edge is stored in the SPDTR and transferred to the Shift register after the current transmission.

11.5.5 Transmission Format When CPHA = 1

A SPI transmission is shown in **Figure 11-5** 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: 1 for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), Master In/ Slave Out (MISO), and Master Out/Slave In (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The SS line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input (SS) is at Logic 0, so only the selected slave drives to the master. The SS pin of the master is not shown but is assumed to be inactive. The SS pin of the master must be high or a Mode Fault Error occurs. When CPHA = 1, the master begins driving its MOSI pin on the first SCLK edge. Therefore, the slave uses the first SCLK edge as a start transmission signal. The SS pin can remain low between transmissions. This format may be preferable in systems having only one master and slave driving the MISO data line.

Note: **Figure 11-5** assumes 16-bit data lengths and the MSB shifted out first.

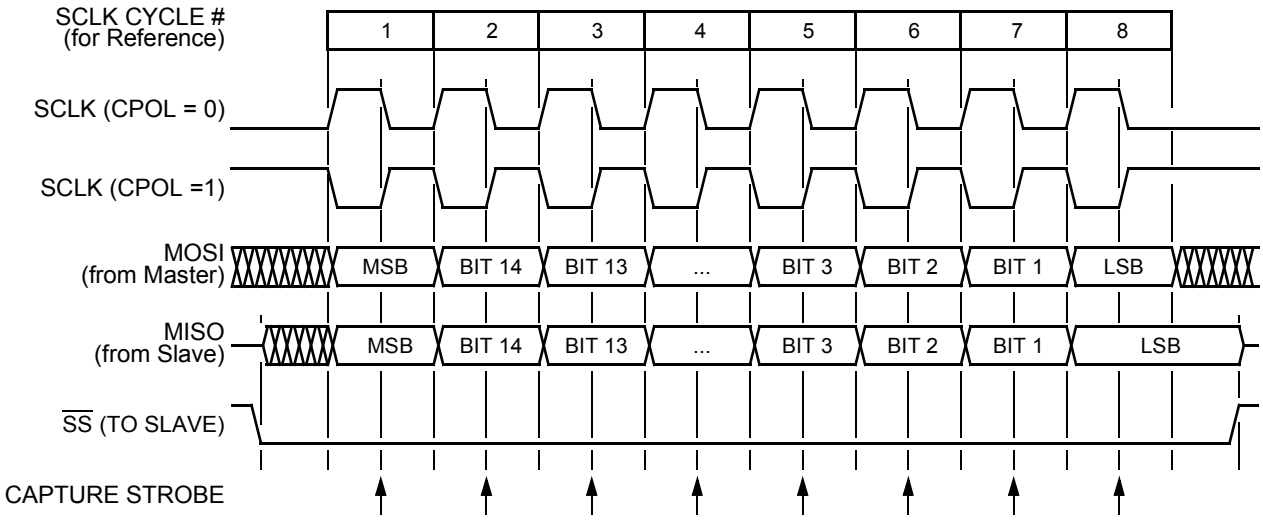


Figure 11-5. Transmission Format (CPHA = 1)

When CPHA = 1 for a slave, the first edge of the SCLK indicates the beginning of the transmission. This causes the SPI 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 SPDTR. Therefore, the SPI Data register of the slave must be loaded with transmit data before the first edge of SCLK. Any data written after the first edge is stored in the SPDTR and transferred to the Shift register after the current transmission.

11.5.6 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDTR starts 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 SPI clock rate, selected by SPR[2:0], affects the delay from the write to SPDTR and the start of the SPI transmission. The internal SPI clock in the master is a free-running derivative of the internal clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. Since the SPI clock is free-running, it is uncertain where the write to the SPDTR occurs relative to the slower SCLK. This uncertainty causes the variation in the initiation delay, demonstrated in **Figure 11-6.** This delay is no longer than a single SPI bit time. That is, the maximum delay is two bus cycles for DIV2, four bus cycles for DIV4, eight bus cycles for DIV8, and so on up to a maximum of 256 cycles for DIV256.

Note: Figure 11-6 assumes 16-bit data lengths and the MSB shifted out first.

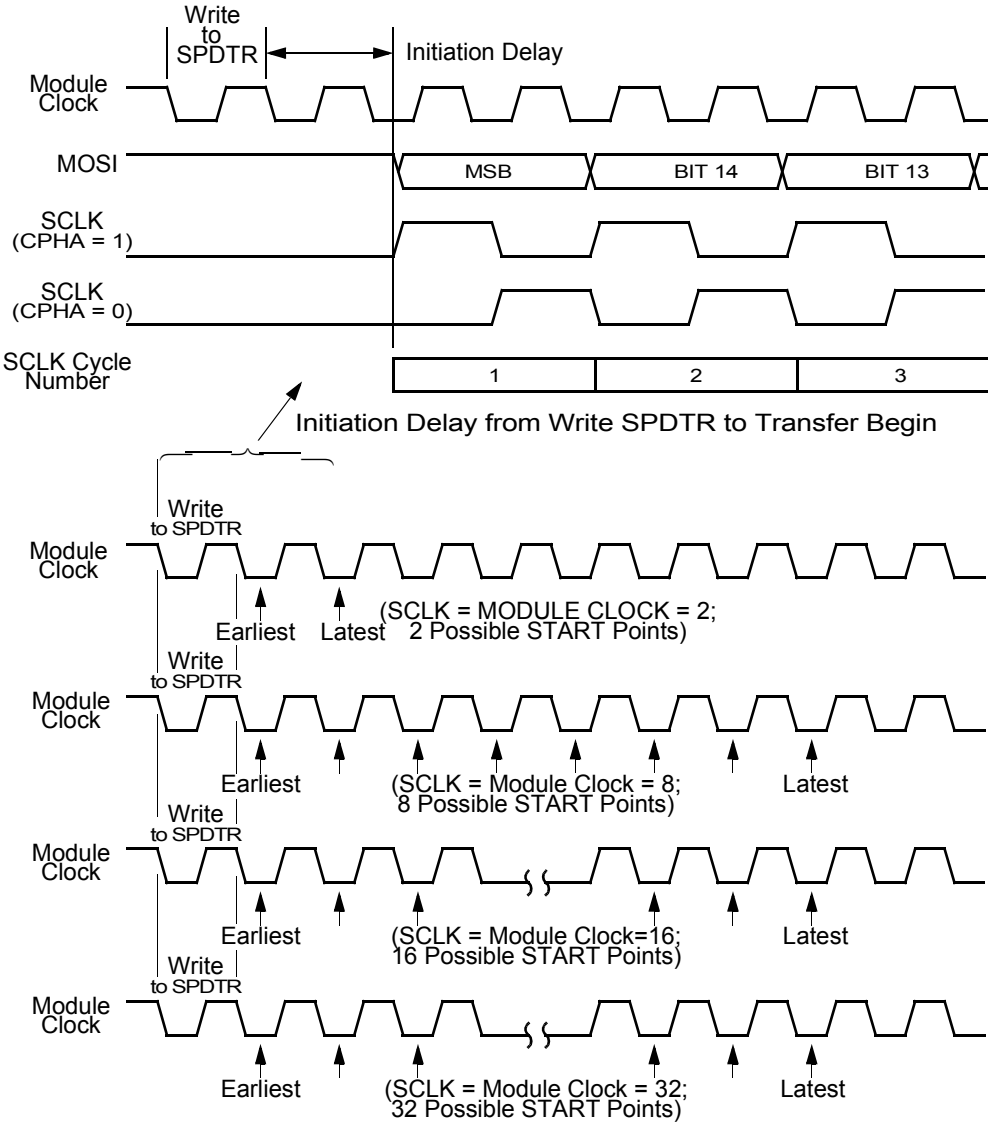


Figure 11-6. Transmission Start Delay (Master)

11.6 Transmission Data

The double-buffered SPDTR allows data to be queued and transmitted. For a SPI configured as a master, the queued data is transmitted immediately after the previous transmission has completed. The SPTE flag indicates when the transmit data buffer is ready to accept new data. Write to SPDTR only when the SPTE bit is high. **Figure 11-7** illustrates the timing associated with doing back-to-back transmissions with the SPI (SCLK has CPHA: CPOL = 1:0).

Note: **Figure 11-7** assumes 16-bit data lengths and the MSB shifted out first.

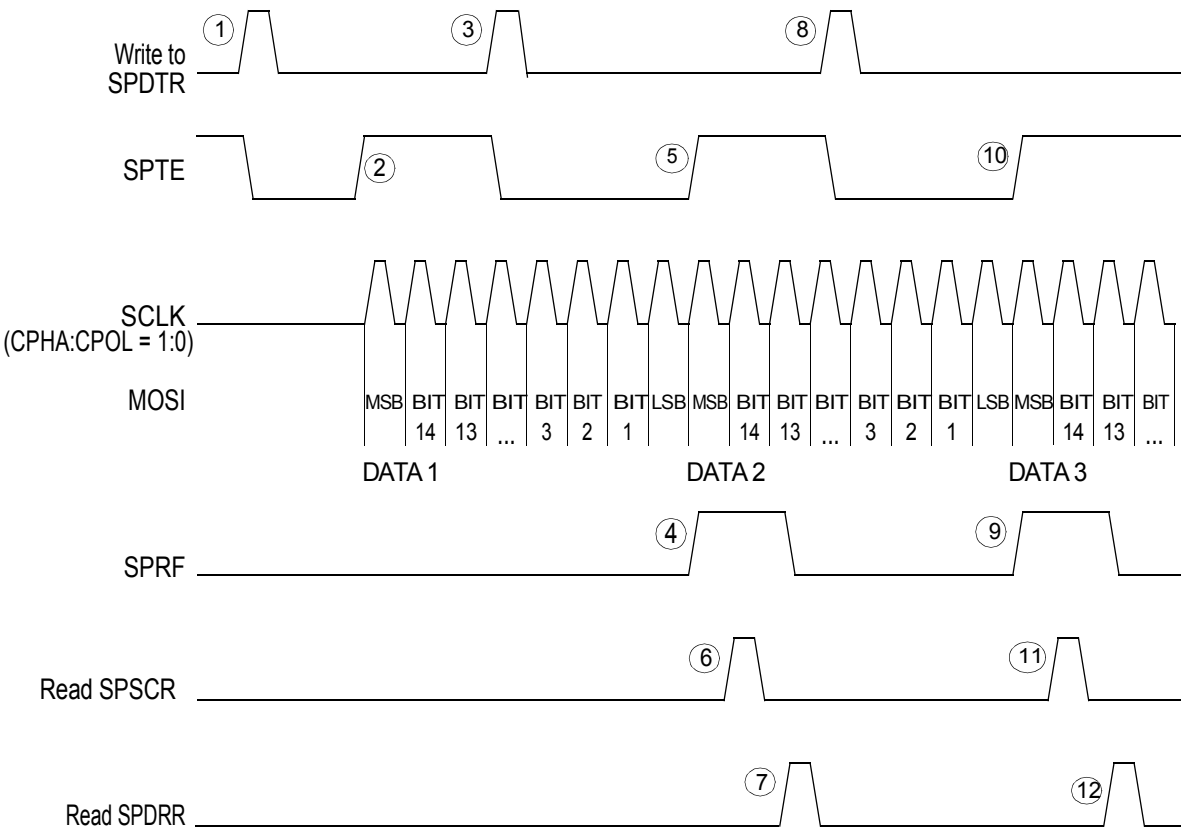


Figure 11-7. SPRF/SPTE Interrupt Timing

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, if no new data is written to the data buffer, the last value contained in the Shift register is the next data to be transmitted.

An idle master or idle slave without loaded data in its transmit buffer, sets the SPTE again no more than two bus cycles after the transmit buffer empties into the Shift register. This allows a queue to send up to a 32-bit value. For an already active slave, the load of the Shift register cannot occur until the transmission is completed. This implies a back-to-back write to the SPDTR is not possible. The SPTE indicates when the next write can occur.

11.7 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the SPI Data register before the next full length data enters the Shift register sets the OVRF bit. The new data will not transfer to the Receive Data register, and the unread data can still be read. OVRF is in the SPSCR.
- Mode Fault Error (MODF) — The MODF bit indicates the voltage on the Slave Select pin ($\overline{SS}$) is inconsistent with SPI mode. MODF is in the SPSCR.

11.7.1 Overflow Error

The Overflow Flag (OVRF) becomes set if the last bit of a current transmission is received and the Receive Data 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 Receive Data Register (SPRDR). It does not set the SPI Receiver Full (SPRF) bit. The unread data transferred to the Receive Data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPSCR, then read the SPRDR.

OVRF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. **Figure 11-8** explains how it is possible to miss an overflow. The first element of the same figure illustrates how it is possible to read the SPSCR and SPDRR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set between the time SPSCR and SPDRR are read.

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious data is being lost as more transmissions are completed. To prevent this loss, either enable the OVRF interrupt or take another read of the SPSCR following the read of the SPDRR. This ensures the OVRF was not set before the SPRF was cleared and future transmissions can set the SPRF bit.

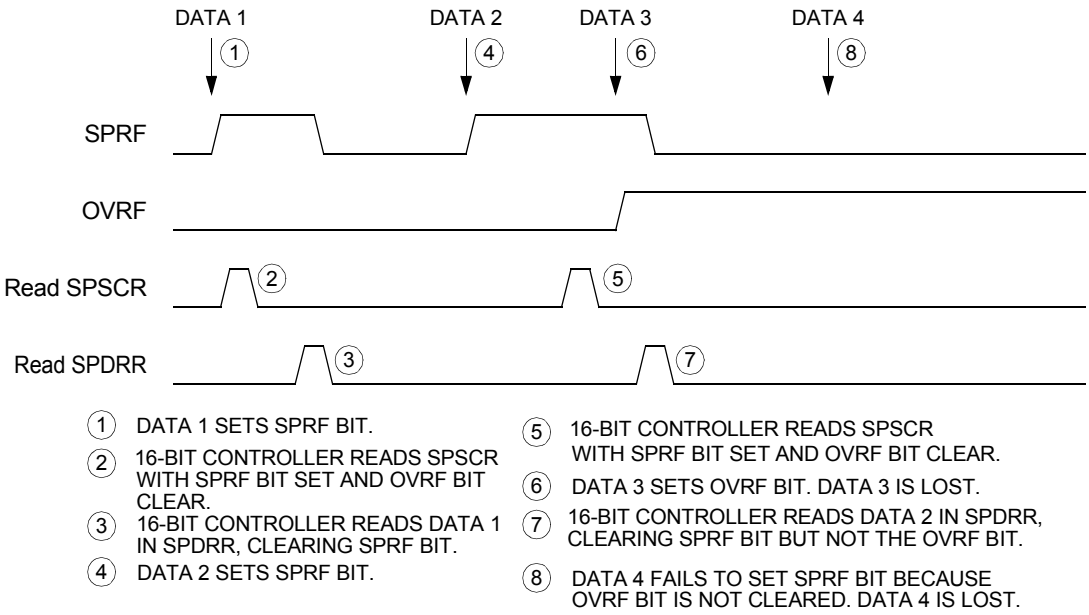


Figure 11-8. Missed Read of Overflow Condition

Figure 11-9 illustrates the described process. Generally, to avoid a second SPSCR read, enable the OVRF by setting the ERRIE bit.

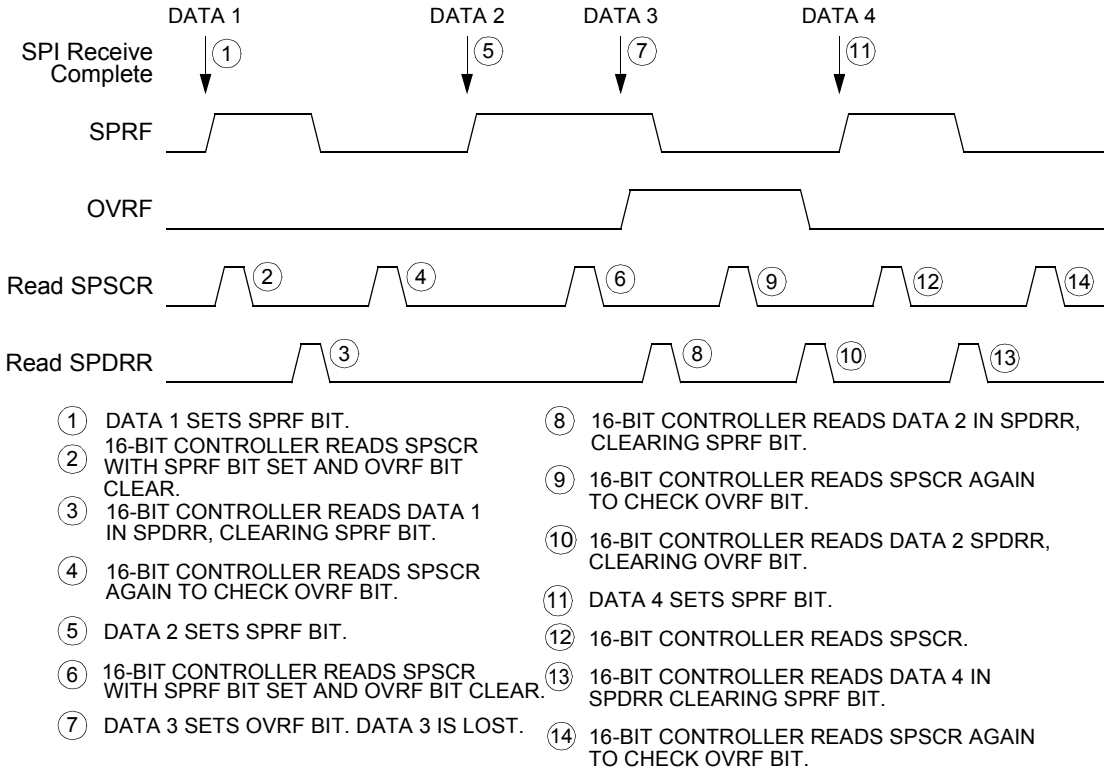


Figure 11-9. Clearing SPRF When OVRF Interrupt Is Not Enabled

11.7.2 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. To prevent SPI pin contention and damage to the 16-bit controller, a Mode Fault Error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transmission.
- The $\overline{SS}$ pin of a master SPI goes low at any time.

To set the MODF flag, Mode Fault Error Enable (MODFEN) bit must be set. Clearing the MODFEN bit does not clear the MODF flag but it does prevent the MODF from being set again after the MODF is cleared.

MODF generates a receiver/error interrupt request if the Error Interrupt Enable (ERRIE) bit is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

11.7.2.1 Master SPI Mode Fault

In a master SPI with Mode Fault Enable (MODFEN) bit set, Mode Fault (MODF) flag is set if $\overline{SS}$ goes to Logic 0. A Mode Fault in a Master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates a SPI receiver/error interrupt request.
- The SPE bit is cleared
- The SPTE bit is set
- The SPI state counter is cleared

In a master SPI, the MODF flag will not be cleared until the $\overline{SS}$ pin is at a Logic 1 or the SPI is configured as a slave.

Note: When CPHA = 0, 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) for CPHA = 0. 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.

11.7.2.2 Slave SPI Mode Fault

In a slave SPI (SPMSTR = 0), the MODF bit generates a SPI Receiver/Error Interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by clearing the SPE bit of the slave.

Note: A Logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SCLK clocks, even if it was already in the middle of a transmission.

When configured as a slave (SPMSTR = 0), the MODF flag is set if the $\overline{SS}$ goes high during a transmission. 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. 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.

To clear the MODF flag, read the SPSCR with the MODF bit set and then write to the SPSCR. This entire clearing mechanism must occur with no MODF condition existing or else the flag is not cleared.

In a slave SPI, if the MODF flag is not cleared by writing a one to the MODF bit, the condition causing the Mode Fault still exists. In this case, the interrupt caused by the MODF flag can be cleared by disabling the EERIE or MODFEN bits (if set) or by disabling the SPI. Disabling the SPI using the SPE bit will cause a partial reset of the SPI and may cause the loss of a message currently being received or transmitted.

11.8 Register Definitions

Table 11-1. SPI Memory Map

Device	Peripheral	Address
82x	SPI0_BASE	$1140
	SPI1_BASE	$1150

Table 11-2 lists the SPI registers in ascending address, including their acronyms and address of each register. The read/write registers should be accessed only with word accesses. Accesses other than word lengths result in undefined results. Figure 11-10 portrays a map summary of the registers.

Table 11-2. SPI Register Summary

Address Offset	Register Acronym	Register Description	Access Type	Chapter Location
Base + $0	SPSCR	Status/Control Register	Read/Write	Section 11.8.1
Base + $ 1	SPDSR	Data Size Register	Read/Write	Section 11.8.2
Base + $2	SPDRR	Data Receive Register	Read/Write	Section 11.8.3
Base + $3	SPDTR	Data Transmit Register	Read/Write	Section 11.8.4

Bit fields of the four registers are summarized in **Figure 11-10.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	SPSCR	R	0	DSO	SPRF	EERIE	OVRF	MODF	SPTE	MOD FEN	SPR1	SPR0	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE
		W																
$1	SPDSR	R	0	0	0	0	0	0	0	0	0	0	0	0	DS3	DS2	DS1	DS0
		W																
$2	SPDRR	R	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
		W																
$3	SPDTR	R																
		W	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

R	0	Read as 0
W		Reserved

Figure 11-10. SPI Register Map Summary

11.8.1 SPI Status and Control Register (SPSCR)

The SPSCR:

- Selects master SPI baud rate
- Determines data shift order
- Enables SPI module interrupt requests
- Configures SPI module as master or slave
- Selects serial clock polarity and phase
- Indicates the SPI status

SPI_BASE+$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	DSO	SPRF	ERRIE	OVRF	MODF	SPTE	MOD FEN	SPR1	SPR0	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-11. SPI Status and Control Register (SPSCR)

See Programmer's Sheet on Appendix page C - 69

Note: Using BFCLR or BFSET instructions on the SPSCR can cause unintended side effects on the status bits. This is due to these instructions being a read/write modify instruction.

11.8.1.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

11.8.1.2 Data Shift Order (DSO)—Bit 14

This read/write bit determines whether the MSB or LSB bit is transmitted or received first. Both Master and Slave SPI modules must transmit and receive the same length packets. Regardless how this bit is set, when reading from the SPDRR or writing to the SPDTR, the LSB will always be at bit location zero. If the data length is less than 16 bits, the data will be zero padded on the upper bits.

- 0 = MSB transmitted first
- 1 = LSB transmitted first

11.8.1.3 SPI Receiver Full (SPRF)—Bit 13

This *read-only* flag is set each time data transfers from the Shift register to the SPDRR. SPRF generates an interrupt request if the SPRIE bit in the SPI Control register is set. This bit is cleared by reading the SPDRR.

- 0 = Data Receive Register (SPDRR) not full
- 1 = Data Receive Register (SPDRR) full

11.8.1.4 Error Interrupt Enable (ERRIE)—Bit 12

This read/write bit enables the MODF, if MODFEN is also set, and OVRF bits to generate interrupt requests.

- 0 = MODF and OVRF cannot generate interrupt requests
- 1 = MODF and OVRF can generate interrupt requests

11.8.1.5 Overflow (OVRF)—Bit 11

This *read-only* flag is set if software does not read the data in the SPDRR before the next full data enters the Shift register. In an overflow condition, the data already in the SPDRR is unaffected, and the data shifted in last is lost. Clear the OVRF bit by reading the SPSCR and then reading the SPDRR. OVRF generates a Receiver/Error interrupt if the EERIE bit is set. See **Section 11.7.1** for more details.

- 0 = No overflow
- 1 = Overflow

11.8.1.6 Mode Fault (MODF)—Bit 10

This *read-only* flag is set in a slave SPI if the $\overline{SS}$ pin goes high during a transmission with the MODFEN bit set. In a master SPI, the MODF flag is set if the $\overline{SS}$ pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing a one to the MODF bit when it is set. MODF generates a Receive/Error interrupt if the EERIE bit is set. Please see **Section 11.7.2** for more details

- 0 = $\overline{SS}$ pin at appropriate Logic level
- 1 = $\overline{SS}$ pin at inappropriate Logic level

11.8.1.7 SPI Transmitter Empty (SPTE)—Bit 9

This *read-only* flag is set each time the SPDTR transfers data into the Shift register. SPTE generates an interrupt request if the SPTIE bit in the SPI Control register is set. This bit is cleared by writing to the SPDTR.

- 0 = Data Transmit Register (SPDTR) not empty
- 1 = Data Transmit Register (SPDTR) empty

Note: Do not write to the SPI Data register unless the SPTE bit is high

11.8.1.8 Mode Fault Enable (MODFEN)—Bit 8

This read/write bit, when set to one, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag.

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation.

11.8.1.9 SPI Baud Rate Select (SPR1 and SPR0)—Bits 7- 6

While in the Master mode, these read/write bits select one of eight baud rates depicted in **Table 11-3.** SPR[2:0] have no effect in Slave mode. Use the formula below to calculate the SPI baud rate.

$$\text{Baud Rate} = \frac{\text{Module Clock}}{\text{Baud Rate Divisor}}$$

Table 11-3. SPI Master Baud Rate Selection

SPR[2:0]	Baud Rate Divisor (BD)
00	2
01	8
10	16
11	32

11.8.1.10 SPI Receiver Interrupt Enable (SPRIE)—Bit 5

This read/write bit enables interrupt requests generated by the SPRF bit. The SPRF bit is set when data transfers from the Shift register to the Receive Data register.

- 0 = SPRF interrupt requests disabled
- 1 = SPRF interrupt requests enabled

11.8.1.11 SPI Master (SPMSTR)—Bit 4

This read/write bit selects Master or Slave modes operation.

- 0 = Slave mode
- 1 = Master mode

11.8.1.12 Clock Polarity (CPOL)—Bit 3

This read/write bit determines the logic state of the SCLK pin between transmissions. To transmit data between SPI modules, the SPI modules must have identical CPOL values. Please see **Figure 11-3** and **Figure 11-5.**

11.8.1.13 Clock Phase (CPHA)—Bit 2

This read/write bit controls the timing relationship between the serial clock and SPI data. Please see **Figure 11-3** and **Figure 11-5.** To transmit data between SPI modules, the SPI modules must

have identical CPHA values. When CPHA = 0, the $\overline{SS}$ pin of the Slave SPI module must be set to Logic 1 between data transmissions, illustrated in **Figure 11-4.**

11.8.1.14 SPI Enable (SPE)—Bit 1

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. When setting/clearing this bit, *no* other bits in the SPSCR should be changed. Failure to following this statement may result in spurious clocks.

- 0 = SPI module disabled
- 1 = SPI module enabled

11.8.1.15 SPI Transmit Interrupt Enable (SPTIE)—Bit 0

This read/write bit enables interrupt requests generated by the SPTE bit. SPTE is set when data transfers from the SPDTR to the Shift register.

- 0 = SPTE interrupt requests disabled
- 1 = SPTE interrupt requests enabled

11.8.2 SPI Data Size Register (SPDSR)

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 will only take effect at the time the SPI is enabled (SPE bit in SPSCR set from zero to one). In order to have a new value take effect, disable then re-enable the SPI with the new value in the register.

SPI_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	0	DS3	DS2	DS1	DS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-12. SPI Data Size Register (SPDSR)

See Programmer's Sheet on Appendix page B- 71

11.8.2.1 Reserved—Bits 15-4

These bits are reserved or not implemented. They are read as 0 and cannot be modified by writing.

11.8.2.2 Data Size (DS)—Bits 3-0

Please see **Table 11-4** for detailed transmission data.

Table 11-4. Data Size

DS[3:0]	Size of Transmission
0000	Not Allowed
0001	2 bits
0010	3 bits
0011	4 bits
0100	5 bits
0101	6 bits
0110	7 bits
0111	8 bits
1000	9 bits
1001	10 bits
1010	11 bits
1011	12 bits
1100	13 bits
1101	14 bits
1110	15 bits
1111	16 bits

11.8.3 SPI Data Receive Register (SPDRR)

This *read-only* register will show the last data word received after a complete transmission. The SPRF bit is set when new data is transferred to this register.

SPI_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 11-13. SPI Data Receive Register (SPDRR)

See Programmer's Sheet on Appendix page B- 72

11.8.4 SPI Data Transmit Register (SPDTR)

This *write-only* register holds data to be transmitted. When the 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 will not be initiated until this register is written. When in Slave mode, the old data will be re-transmitted. All data should be written with the LSB at bit zero.

SPI_BASE+$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-14. SPI Data Transmit Register (SPDTR)

See Programmer's Sheet on Appendix page B- 73

11.9 Resets

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following will occur:

- SPTE flag is set
- Any Slave mode transmission currently in progress is aborted
- Any Master mode transmission currently in progress is continued to completion
- SPI state counter is cleared, making it ready for a new complete transmission
- All the SPI port Logic is disabled

The following items are reset only by a system reset:

- The SPDTR and SPDRR registers
- All control bits in the SPSCR (MODFEN, ERRIE, SPR[2:0])
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, it is possible to clear SPE between transmissions without having to set all control bits again when SPE is set back high for the next transmission.

By not resetting the SPRF, OVRF, and MODF flags, it is possible to service the interrupts after the SPI has been disabled. Disable SPI by writing zero to the SPE bit. SPI can also be disabled by a Mode Fault occurring in a SPI configured as a master.

11.10 Interrupts

Four SPI status flags can be enabled to generate interrupt requests.

Table 11-5. SPI Interrupts

Flag	Interrupt Enabled By	Description
SPTE (Transmitter Empty)	SPI Transmitter Interrupt Request (SPTIE = 1,SPE = 1)	The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter interrupt requests provided that the SPI is enabled (SPE = 1). The SPTE bit becomes set every time data transfers from the SPDTR to the Shift register. The clearing mechanism for the SPTE flag is a write to the SPDTR.
SPRF (Receiver Full)	SPI Receiver Interrupt Request (SPRIE = 1)	The SPI Receiver Interrupt Enable bit (SPRIE) enables the SPRF bit to generate receiver interrupt requests regardless of the state of the SPE bit. The SPRF is set every time data transfers from the Shift register to the SPDRR. The clearing mechanism for the SPRF flag is to read the SPDRR.
OVRF (Overflow)	SPI Receiver/Error Interrupt Request (ERRIE = 1)	The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error interrupt request.
MODF (Mode Fault)	SPI Receiver/Error Interrupt Request (ERRIE = 1, MODFEN = 1)	The Mode Fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error 16-bit controller interrupt requests.

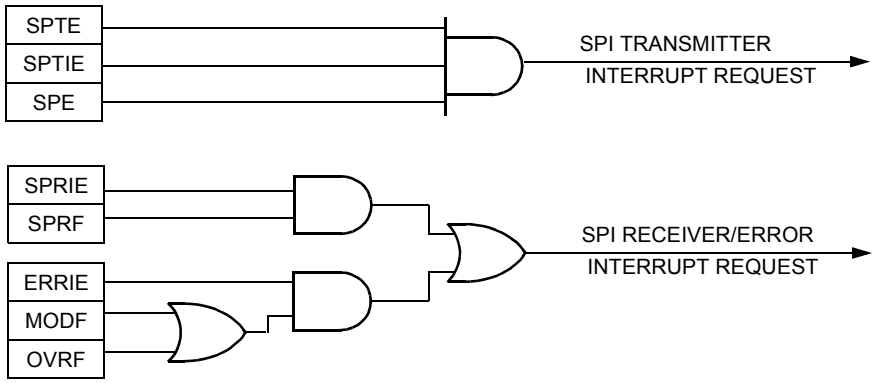


Figure 11-15. SPI Interrupt Request Generation

Chapter 12
Synchronous Serial Interface (SSI)

12.1 Introduction

The Synchronous Serial Interface (SSI) of the 56F826/827, its architecture, programming model, operating modes, and its initialization are discussed in this section.

The SSI is a full-duplex serial port allowing the device to communicate with a variety of serial devices, including industry-standard codecs, other devices, microprocessors, and peripherals implementing the Serial Peripheral Interface (SPI). It is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

Capabilities of the SSI include:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame synchronizations
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as 32-time slots
- Gated clock mode operation requiring no frame sync
- Programmable internal clock divider
- Programmable word length from 8,10, 12, or 16 bits
- Program options for frame sync and clock generation
- SSI power-down feature
- Completely separate clock and frame sync selections for the receive and transmit sections

12.2 SSI Architecture

The SSI input/output is illustrated in **Figure 12-1.**

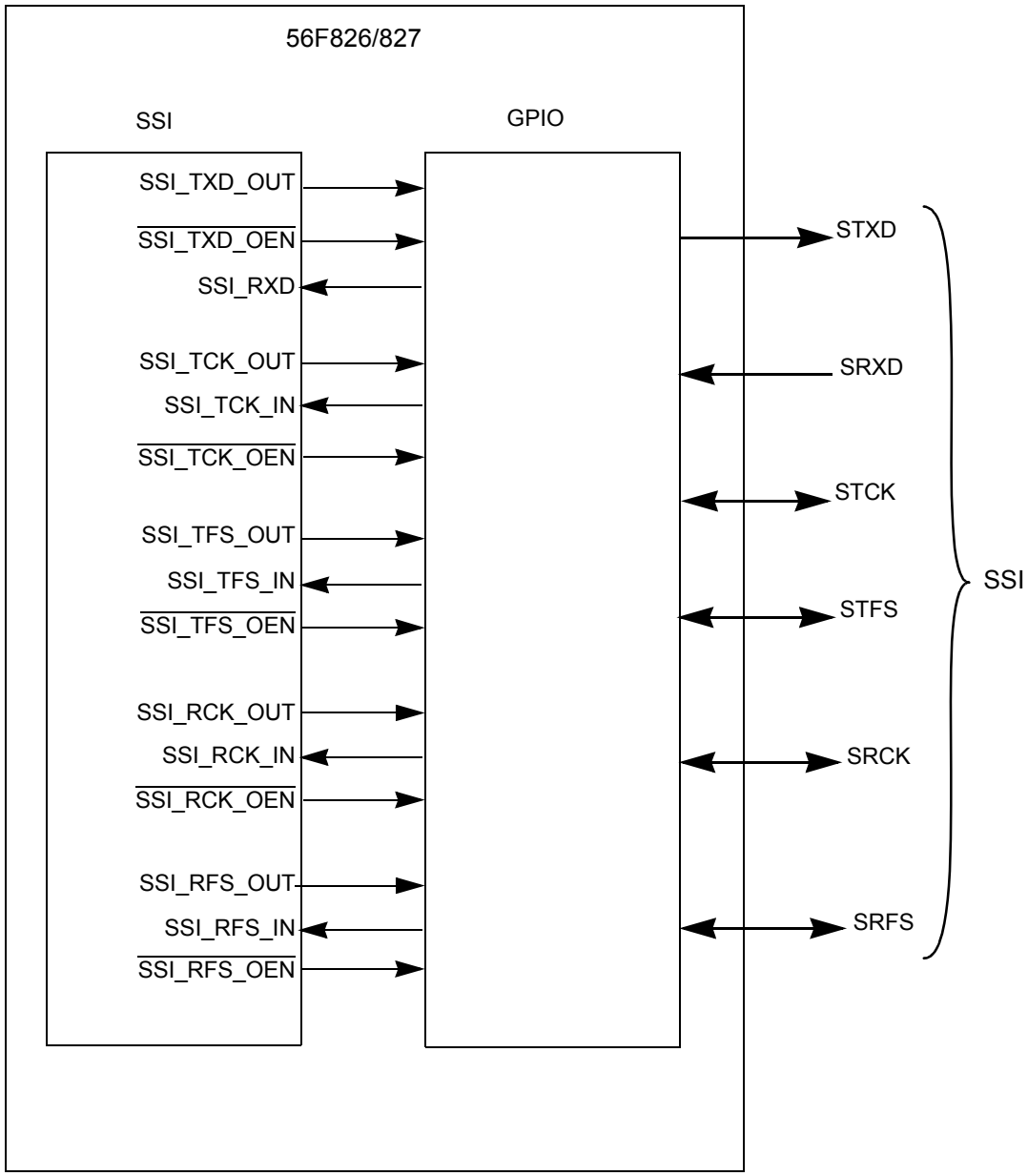


Figure 12-1. SSI Input/Output Block Diagram

The SSI block diagram consists of three control registers. The control registers set up the port, one status/control register, separate transmit and receive circuits FIFO registers, and separate serial clock and frame sync generation for the transmit and receive sections.

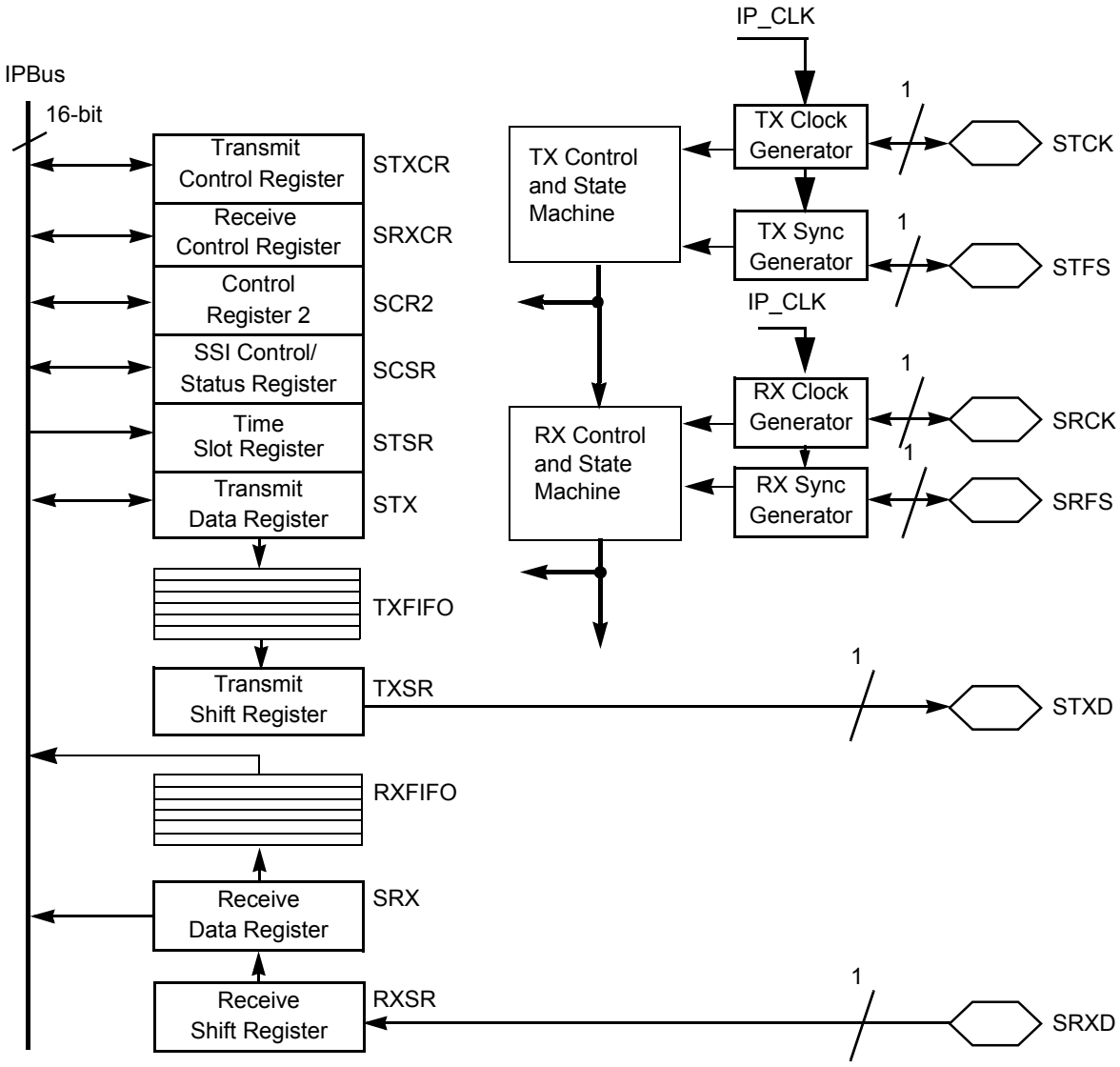


Figure 12-2. SSI Block Diagram

12.2.1 SSI Clocking

SSI uses the following three clocks:

- Bit clock— used to serially clock the data bits in and out of the SSI port
- Word clock—used to count the number of data bits per word 8, 10, 12, or 16 bits
- Frame clock—used to count the number of words in a frame

Used to serially clock the data, the bit clock is visible on the serial transmit clock:

- SSI_TCK_OUT pin or SSI_TCK_IN pin

The bit clock, used to serially clock the data, is visible on the serial receive clock:

- SSI_RCK_OUT pin or SSI_RCK_IN pin

The word *clock* is defined as an *internal clock* used to determine when transmission of an 8-, 10-, 12-, or 16-bit word has completed. The word *clock* in turn times the frame clock, responsible for counting the number of words in the frame.

The frame clock can be viewed on the frame sync pins as transmit and receive:

- SSI_TFS_OUT or SSI_TFS_IN
- SSI_RFS_OUT or SSI_RFS_IN

because a frame sync is generated after the correct number of words in the frame have passed**.** The bit clock can be received from an SSI clock pin or can be generated from the system clock through a divider, as illustrated in **Figure 12-3.**

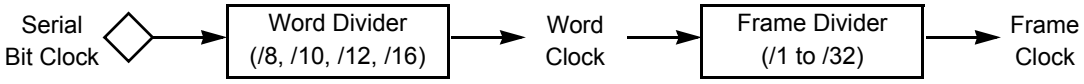


Figure 12-3. SSI Clocking

12.2.2 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally by the SSI or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the system clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit-rate clock generation. In the gated clock mode, the data clock is valid only when data is being transmitted. Otherwise the clock enable signals, $\overline{\text{SSI\_TX\_CLK\_OEN}}$ *or* $\overline{\text{SSI\_RX\_CLK\_OEN}}$ are disabled. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

Figure 12-4 is a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the Transmit Direction (TXDIR) bit in the SSI Control Register 2 (SCR2) register. The receive section contains an equivalent clock generator circuit.

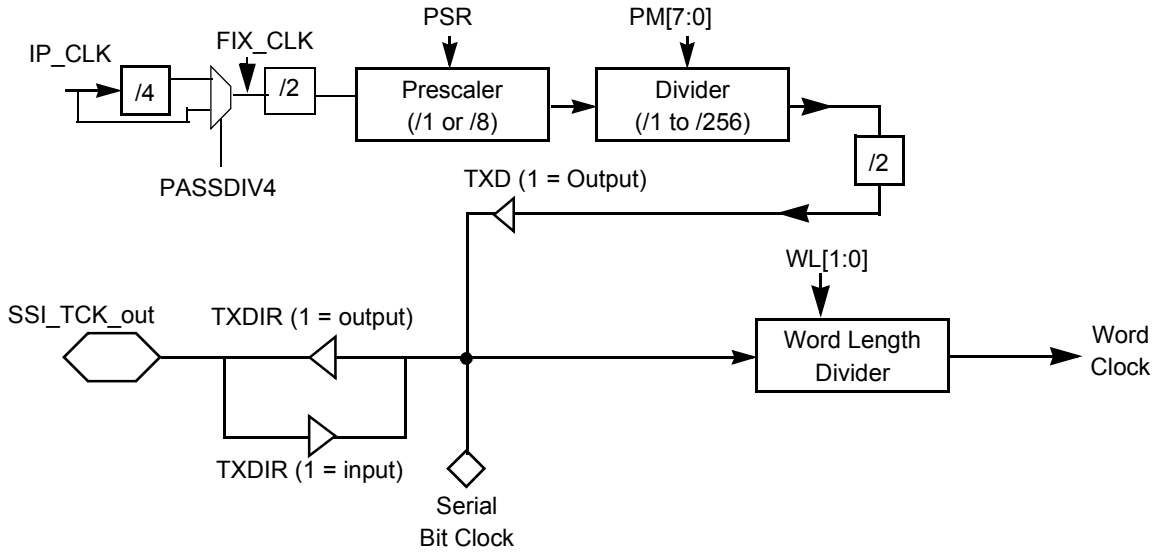


Figure 12-4. SSI Transmit Clock Generator Block Diagram

Figure 12-5 illustrates the frame sync generator block for the transmit section. When internally generated, both receive and transmit frame sync are generated from the word clock and are defined by the frame rate divider (DC[4:0]) bits and the word length (WL[1:0]) bits of the SSI Transmit Control Register (SCRTX). The receive section contains an equivalent circuit for the frame sync generator.

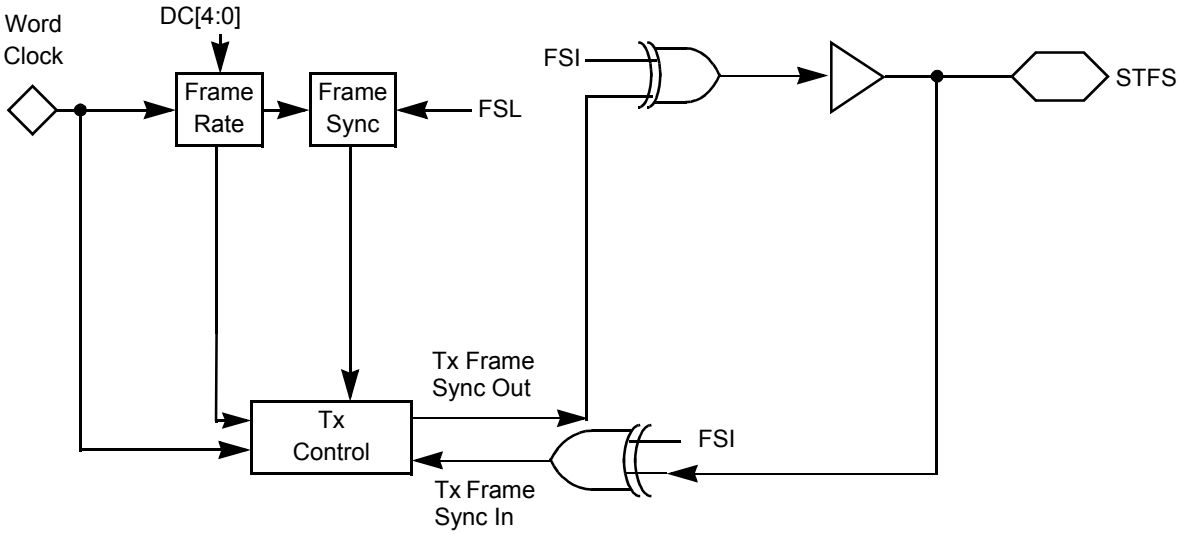


Figure 12-5. SSI Transmit Frame Sync Generator Block Diagram

12.3 Programming Model

The registers associated with the SSI include:

- SSI Transmit Shift Register (TXSR) is not user accessible
- SSI Transmit FIFO Register is not user accessible
- SSI Transmit Data (STX) register is *write-only*
- SSI Receive Shift Register (RXSR) is not user accessible
- SSI Receive FIFO Register is not user accessible
- SSI Receive Data (SRX) register is *read-only*
- SSI Transmit Control Register (STXCR)
- SSI Receive Control Register (SRXCR)
- SSI Control Register 2 (SCR2)
- SSI Control/Status Register (SCSR) is lower byte *read-only*
- SSI Time Slot Register (STSR) is *write-only*

The control registers associated with the SSI are exhibited in the following figures. The TXSR, RXSR, RxFIFO, TxFIFO are not accessible.

12.4 Register Definitions

Table 12-1. SSI Memory Map

Device	Peripheral	Address
826/827	SSI_BASE	$10E0

The following figures and associated definitions provide detailed descriptions of various SSI registers. These definitions serve as a key for the registers:

- Register bit defines the register bit's behavior. Possible values are:
  - — *Read-only*. Writing to this bit has no effect.
  - — *Write-only*.
  - — Standard read/write bit. Other than a hardware reset, only software can change the bit's value.
- Provides the reset value of the bit. Possible values:
  - — 0 = Will reset to a Logic 0
  - — 1 = Will reset to a Logic 1

Table 12-2. SSI Register Summary

Address Offset	Register Acronym	Register Description	Access Type	Chapter Location
Base + $0	STX	Transmit Register	Read/Write	Section 12.4.1
Base + $ 1	SRX	Receive Register	Read/Write	Section 12.4.4
Base + $2	SCSR1	Control/Status Register 1	Read/Write	Section 12.4.7
Base + $3	SCSR2	Control/Status Register 2	Read/Write	Section 12.4.8
Base + $4	STXCR	Transmit Control Register	Read/Write	Section 12.4.9
Base + $5	SRXCR	Receive Control Register	*Read-Only*	Section 12.4.9
Base + $6	STSR	Time Slot Register	*Write-Only*	Section 12.4.10
Base + $7	SFCSR	FIFO Control/Status Register	*Read-Only*	Section 12.4.11

The SSI module has eight, 16-bit registers found in **Figure 12-3.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	STX	R W	HIGH BYTE								LOW BYTE							
$1	SRX	R	HIGH BYTE								LOW BYTE							
		W																
$2	SCSR1	R	DIV4 DIS	RSHFD	RSCKP	RDMAE	TDMAE	RFSI	RFSL	REFS	RDR	TDE	ROE	TUE	TFS	RFS	RFF	TFE
		W																
$3	SCSR2	R W	RIE	TIE	RE	TE	RFEN	TFEN	RXDIR	TXDIR	SYN	TSHFD	TSCKP	SSIEN	NET	TFSI	TFSL	TEFS
$4	STXCR	R W	PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
$5	SRXCR	R W	PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
$5	SRXCR	R W	RSV[32:16]															
$6	STSR	R	DUMMY REGISTER, WRITTEN DURING INACTIVE TIME SLOTS (NETWORK MODE)															
		W																
$7	SFCSR	R	RFCNT				TFCNT				RFWM				TFWM			
		W																

R	0	Read as 0
W		Reserved

Figure 12-3. SSI Register Map Summary

12.4.1 SSI Transmit Register (STX)

SSI_BASE+$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIGH BYTE								LOW BYTE							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-4. SSI Transmit Register (STX)

See Programmer's Sheet on Appendix page B-76

The STX is a 16-bit, read/write register. Transmitted data is written into this register. If the transfer FIFO is used data is transferred from this register to the transfer FIFO register when it becomes empty. Otherwise, data written to this register is transferred to the TXSR when shifting of previous data is completed. The data written occupies the most significant portion of the STX register. Unused bits, the least significant portion, of the STX register are ignored. The interrupt is asserted whenever the STX register becomes empty. Then when both STX and SSI transfer FIFO registers are empty, and if transfer FIFO is enabled, both the Transmit Data Empty (TDE) register bit in the SCSR and the TIE bit in the SCR2 are set.

Note: Enable SSI (SSIEN = 1) before writing to SSI transmit register.

12.4.2 SSI Transmit FIFO Register

The SSI transfer FIFO register is a 7 × 16-bit register used to buffer samples written to the STX register. It is written by the contents of the STX whenever the transmit FIFO feature is enabled. When enabled, the TXSR receives its values from this FIFO register. Transmitted data is on a first-in, first-out (FIFO) basis. If the transmit FIFO feature is not enabled, this register is bypassed and the contents of the STX register is transferred into the TXSR. When the TIE bit in the SCR2 and TDE register bit in the SCSR are set, the interrupt is asserted whenever the STX is empty and the data level in SSI transmit FIFO falls below the selected threshold. When both transmit FIFO and STX are full, any further write will overwrite the content of STX, transmitting FIFO.

Note: Enable SSI, before writing to transmit data register and transmit FIFO.

12.4.3 SSI Transmit Shift Register (TXSR)

The SSI TXSR is a 16-bit shift register containing data being transmitted. When a continuous clock is used, data is shifted out to the serial transmit data, SSI_TXD_OUT, pin by the selected internal or external bit clock when the associated internal/external frame sync is asserted. When a gated clock is used, data is shifted out to the SSI_TXD_OUT pin by the selected internal/external gated clock. The two Word Length (WL) control bits in the STXCR determine[1] the number of

bits to be shifted out of the TXSR before it is considered empty and can again receive writing. This word length can be 8, 10, 12, or 16 bits. Data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is always shifted out of this register with the Most Significant Bit (MSB) first when the Shift Direction (SHFD) bit of the SCR2 is cleared. If this bit is set, the Least Significant Bit (LSB) is shifted out first.

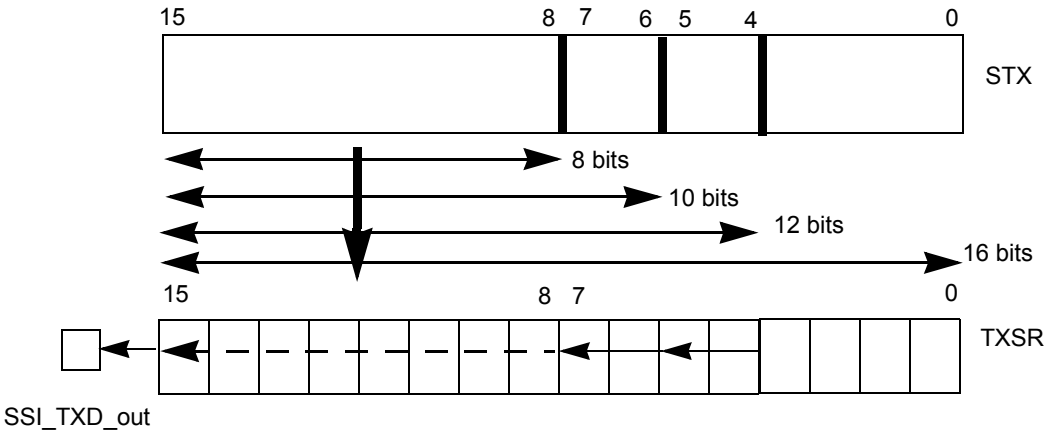


Figure 12-5. Transmit Data Path (TSHFD=0)

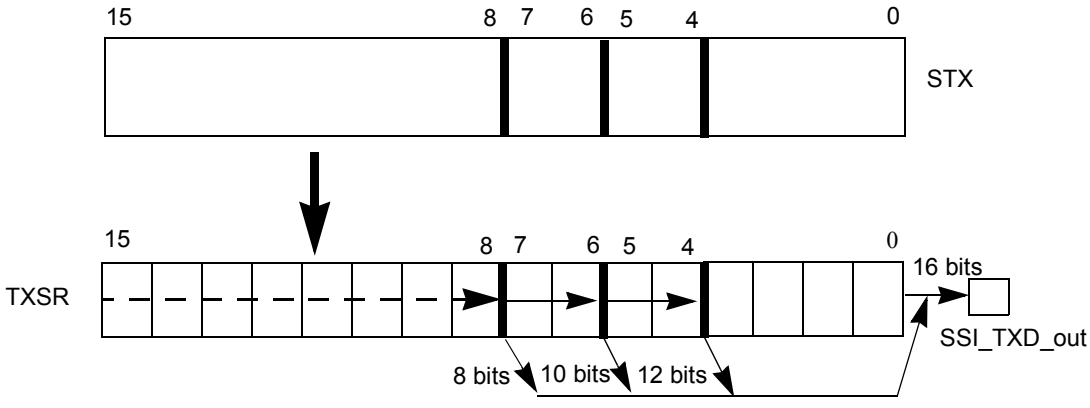


Figure 12-6. Transmit Data Path (TSHFD=1)

1.This is described as SSI transmit and receive control registers.

12.4.4 SSI Receive Data Register (SRX)

The SSI SRX 16-bit register is *read-only*. It always accepts Receive (data) Shift Register (RXSR) as it becomes full. The data read occupies the most significant portion of the SRX register. Unused bits, or the least significant portions, are read as zeros. The interrupt is asserted whenever:

- SRX register becomes full
- The interrupt is asserted if the FIFO is enabled
- When both the SRX register and receive FIFO register are full

The previous asserted interrupt is conditional if the receive data full interrupt is enabled.

SSI_BASE+$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Read	HIGH BYTE								LOW BYTE						
Write															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-7. SSI Receive Data Register (SRX)

See Programmer's Sheet on Appendix page B-77

12.4.5 SSI Receive FIFO Register

The SSI Receive FIFO register is a 7 × 16-bit, used to buffer samples received in the SRX register. It is written by the receive data register whenever the receive FIFO feature is enabled. The FIFO feature is enabled by setting the Receive FIFO Enable (RFEN) bit in the SCR2. When set, the SRX receives its values from the FIFO register after SSI receive data register is full. The interrupt is asserted whenever the SRX register is full and the data level in the SSI receive FIFO reaches the selected threshold, but only if the associated interrupt is enabled. When the FIFO feature is not enabled, the register is bypassed and the Receive Data Shift (RXSH) register is automatically transferred into the SRX.

12.4.6 SSI Receive Shift Register (RXSR)

The SSI Receive (data) Shift Register (RXSR) is a 16-bit, shift register receiving incoming data from the serial receive data SSI_RDX pin. When a continuous clock is used, data is shifted in by the selected internal/external bit clock when the associated internal/external frame synchronization is asserted. If a gated clock is used, data is shifted in by the selected internal/external gated clock. Data is assumed to be received through the most significant bit first if the Shift Direction (SHFD) bit of the SCR2 is cleared. If this bit is set, the data is received through the least signicant bit first. Data is transferred to the SSI Receive (data) SRX register. If the receive buffer is enabled after 8, 10, 12, or 16 bits have been shifted in, data may be received by

the FIFO data buffer register depending on the WL[1:0] control bits. For receiving 8, 10, or 12 bit data, the least significant bits are appended with zero.

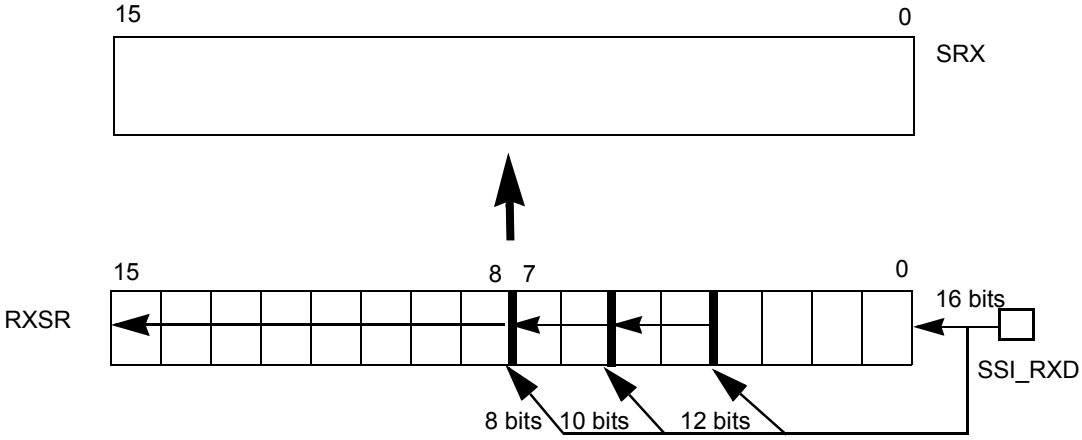


Figure 12-8. Receive Data Path (RSHFD = 0)

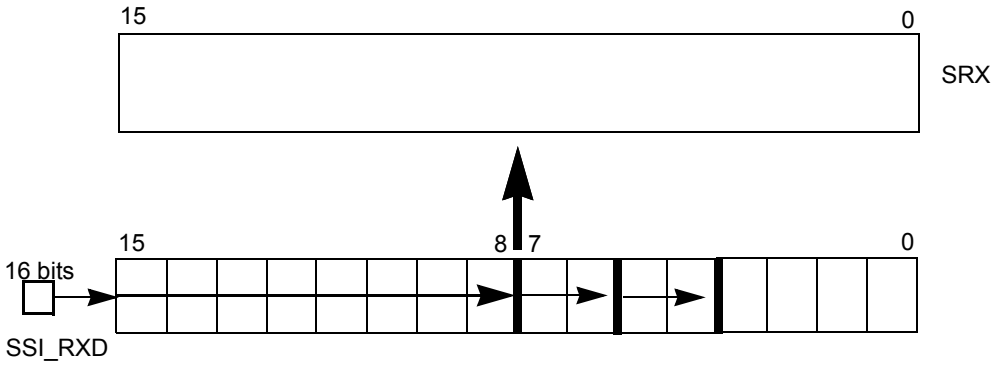


Figure 12-9. Receive Data Path (RSHFD = 1)

12.4.7 SSI Control/Status Register 1 (SCSR)

The SSI Control/Status Register (SCSR), a 16-bit register, is used to set-up and monitor the SSI. The top half of the register (bits 15-8) is the read/write portion of the register. The read/write portion is used for SSI setup. The bottom half of the register (bits 7-0) is read-only. This register section is used by the SSI to interrogate the status and serial input flags of the SSI. The control and status bits are described in the following paragraphs.

Note: SSI status flag is updated when SSI is enabled.

Note: All flags in the status portion of the SCSR are update after the first bit of the next SSI word has completed transmission or reception. Some status bits (ROE and TUE) are cleared by reading the SCSR followed by a read/write to either the SRX or STX register.

SSI_BASE+$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DIV4 DIS	RSHFD	RSCKP	RDMAE	TDMAE	RFSI	RFSL	REFS	RDR	TDE	ROE	TUE	TFS	RFS	RFF	TFE
Write																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Figure 12-10. SSI Control/Status Register 1 (SCSR1)

See Programmer's Sheet on Appendix page B-78

12.4.7.1 Divider Four Disable (DIV4DIS)—Bits 15

When divider four disable control bit is set, FIX_CLK is equal to MCU_CLK. By default, DIV4DIS = 0, FIX_CLK = MCU_CLK/4 for both transmitter and receiver clock generator circuits.

12.4.7.2 Receive Shift Direction (RSHFD)—Bit 14

The Receive Shift Direction (RSHFD) control bit governs whether the most or least significant bit is received first as the receive section. If the RSHFD bit is cleared, data is received by the MSB first. If the RSHFD bit is set, the LSB is received first.

Note: The codec device labels the MSB as bit zero, whereas the SSI labels the LSB as bit zero. Therefore, when using a standard codec, the SSI MSB (or codec bit zero) is shifted out first, and the RSHFD bit should be cleared.

12.4.7.3 Receive Clock Priority (RSCKP)—Bit 13

The Receive Clock Polarity (RSCKP) control bit determines which bit clock edge is used to latch in data for the receive section. If the RSCKP bit is cleared, the data is latched in on the falling edge of the clock. If the RSCKP bit is set, the rising edge of the clock is used to latch the data in. This bit is unused in the synchronous mode only Transmit Clock Polarity (TSCKP) affects polarity, in both the transmitter and receiver when Sync (SYN) = 1.

12.4.7.4 Receive Direct Memory Access Enable (RDMAE)—Bit 12

The Receive Direct Memory Access (DMA) Enable (RDMAE) control bit is instrumental in the SSI requesting a DMA transfer. When the RDMAE bit is set, a DMA request is generated when the SSI RFF bit in the SCSR is set, only if receive FIFO is enabled (RFEN = 1). Otherwise, if RFEN = 0, a DMA request is generated when the RDR bit is set.

RIE has higher priority than RDMAE. That is, if RIE is set, an interrupt will be generated to CPU instead of DMA.

There is no DMA on this part. This bit should always be cleared.

12.4.7.5 Transmit Direct Memory Access Enable (TDMAE)—Bit 11

The Transmit DMA Enable (TDMAE) control bit designates the SSI to request for a DMA transfer. When TDMAE bit is set, a DMA request is generated when the TFE bit in the SCSR is set if transmit FIFO is enabled (TFEN = 1). Otherwise, if TFEN = 0, a DMA request is generated when the Transmit Data Empty (TDE) register bit is set.

TIE has higher priority than TDMAE. That is, if TIE is set, an interrupt will be generated to CPU instead of DMA.

There is no DMA on this part. This bit should always be cleared.

12.4.7.6 Receive Frame Sync Invert (RFSI)—Bit 10

The Receive Frame Sync Invert (RFSI) control bit selects the logic of frame sync I/O for the receive section. If the RFSI bit is set, the frame synchronization is active low. If the RFSI bit is cleared, the frame sync is active high.

12.4.7.7 Receive Frame Sync Length (RFSL)—Bit 9

The Receive Frame Sync Length (RFSL) control bit selects the length of the frame synchronization signal to be generated, or recognized for the receive section. If the RFSL bit is set, a one-clock-bit-long frame sync is selected. The length of this word-long frame synchronization is the same as the length of the data word selected by the two WL bits.

12.4.7.8 Receive Early Frame Sync (REFS)—Bit 8

The Receive Early Frame Sync (REFS) control bit determines when the frame synchronization is initiated for the receive section. When the REFS bit is cleared, the synchronized frame is initiated as the first bit of data received. When the REFS bit is set, the frame synchronization is initiated one bit before the data is received. The frame coordination is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync.

12.4.7.9 Receive Data Ready (RDR)—Bit 7

The SSI Receive Data Ready (RDR) flag bit is set when receive data register SRX or receive FIFO is loaded with a new value.

RDR is cleared when the CPU reads the SRX register. If receive FIFO is enabled, RDR is cleared when receive FIFO is empty.

If the RIE bit is set, a receive data interrupt request is issued when the RDR bit is set. The interrupt request vector depends on the state of the ROE bit, in the SCSR. The RDR bit is cleared by POR and SSI reset.

12.4.7.10 Transmit Data Empty (TDE)—Bit 6

The SSI Transmit Data Empty (TDE) register flag bit is set when there is no data waiting to be transferred to the SSI Transmit Data (STX) register. If the transmit FIFO is enabled, there is at least one empty slot in the transmit data register, or transmit FIFO. However, if the transmit FIFO is not enabled, there is an empty slot in the transmit data register. For instance, when the contents of the STX register are transferred into the transmit shift register and set, the TDE bit indicates data should be written to the STX register or to the STSR before the transmit shift register becomes empty. If the transmit shift register becomes empty it will cause an underrun error.

The TDE bit is cleared when the data is written to the STX register, or to the STSR to disable transmission of the next time slot. If the TIE bit is set, an SSI transmit data interrupt request is issued when the TDE bit is set. The vector of the interrupt depends on the state of the TUE bit, in the SCSR. The TDE bit is set by power-on and SSI reset.

12.4.7.11 Receive Overrun Error (ROE)—Bit 5

The ROE flag is set when the RXSR is filled and ready to transfer to the SSI Receive Data (SRX) register. Another possibility would be the enabled receive FIFO register and the registers are already full.

If FIFO is enabled, indicated by the Receive FIFO Full (RFF), otherwise it is indicated by the Receive Data Ready (RDR) bit being set. The Receive Data Shift Register (RXSR) is not transferred in this case. A receive overrun error does not cause interrupts. However, when the Receive Overrun Error (ROE) bit is set, it causes a change in the interrupt vector used. The change in the interrupt vector reveals the use of the different interrupt handler for a receive overrun condition.

If a receive interrupt occurs with the Receive Overrun Error (ROE) bit set, the received data with exception status interrupt in generated. The ROE bit is cleared by power-on or SSI reset and is cleared by reading the SCI Control Status Register (SCSR) with the ROE bit set. It is followed by reading the SRX register. Clearing the Receive Enable (RE) bit does not affect the ROE bit.

12.4.7.12 Transmitter Underrun Error (TUE)—Bit 4

When the Transmit Data Empty (TDE) register bit is set, indicating there is no data to transmit, the next time slot occurs. At this point the TUE bit will be set, indicating a transmit underrun condition has occurred. When a TUE occurs, the previously sent data is retransmitted. A transmit time slot in the normal mode occurs when the frame synchronization is asserted. In network mode, each time slot requires data transmission and is, therefore, a transmit time slot (TE = 1).

The TUE bit does not cause any interrupts. However, the TUE bit does cause a change in the interrupt vector used for transmit interrupts so a different interrupt handler can be used for a transmit underrun condition. The transmit data with exception status interrupt is generated if a transmit interrupt occurs with a set TUE bit. If a transmit interrupt occurs with the TUE bit cleared, the transmit data interrupt is generated.

The TUE bit is cleared by power-on or SSI reset. The TUE bit is also cleared by reading the SCSR with the TUE bit set, followed by writing to the STX register or to the STSR.

12.4.7.13 Transmit Frame Sync (TFS)—Bit 3

When set, the Transmit Frame Sync (TFS) flag bit indicates a frame synchronization occurred during transmission of the last word written to the STX register. Data written to the STX register during the time slot when the TFS bit is set is sent during the second time slot in network mode. Transmission can also be during the next first time slot in the normal mode. In the Network mode, the TFS bit is set during transmission of the first slot of the frame. The bit is then cleared when starting transmission of the next slot. The TFS bit is cleared by power-on or SSI reset.

12.4.7.14 Receive Frame Sync (RFS)—Bit 2

When set, the Receive Frame Sync (TFS) flag bit indicates a frame synchronization occurred during receiving of the next word into the SRX register. In the Network mode, the RFS bit is set while the first slot of the frame is being received. It is cleared when the next slot of the frame begins to be received. The RFS bit is cleared by power-on or SSI reset.

12.4.7.15 Receive FIFO Full (RFF)—Bit 1

The Receive FIFO Full (RFF) flag bit is set when the receive section is programmed with the receive FIFO enabled and the data level in the receive FIFO reaches the selected Received FIFO Watermark (RFMW) threshold. When set, the Receive FIFO Full register indicates data can be read via the Receive Data register.

An interrupt is only generated if both the Receive FIFO Full (RFF) and Receive Interrupt Enable (RIE) bits are set when receive FIFO is enabled. The RFF bit is cleared by power-on positioned on reset and when SSI is disabled. When receive FIFO is full, all further received data is ignored until the data is read.

12.4.7.16 Transmit FIFO Empty (TFE)—Bit 0

The Transmit FIFO Empty (TFE) flag bit is set when the transmit section is programmed with the transmit FIFO enabled and when the data level in the transmit FIFO falls below the selected TFWM threshold. When set, the TFE bit notifies data can be written to the transmit FIFO register.

An interrupt is generated only if both the TFE and the TIE bits are set if transmit FIFO is enabled. The TFE bit is set by power-on reset and when SSI is disabled.

12.4.8 SSI Receive Control Register 2 (SCR2)

The SSI Receive Control Register 2 (SCR2) is one of three 16-bit read/write control registers used to direct the operation of the SSI. The SSI Reset is controlled by a bit in SCR2. The SCR2 bit controls the direction of the bit clock and frame synchronized pins. Interrupt enabled bits for the receive and transmit sections are provided in this control register. SSI operating modes are also selected in this register. Power-on reset clears all SCR2 bits. However, SSI reset does not affect the SCR2 bits. The SCR2 bits are described in the following paragraphs. The Interrupt Priority Register (IPR) must be set to enable the interrupt. Finally, the interrupt can be enabled from within the SSI.

SSI_BASE+$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RIE	TIE	RE	TE	RFEN	TFEN	RXDIR	TXDIR	SYN	TSHFD	TSCKP	SSIEN	NET	TFSI	TFSL	TEFS
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-11. SSI Receive Control Register 2 (SCR2)

See Programmer's Sheet on Appendix page B-79

12.4.8.1 SSI Receive Interrupt Enable (RIE)—Bit 15

The SSI Receive Interrupt Enable (RIE) control bit prescribes the program controller interruption. When the RIE and RE bits are set, the program controller is interrupted when the SSI Receive Data Full (RDF) register bit in the SCSR is set, but only if the receive FIFO is enabled. If receive FIFO is not enabled the interrupt is generated when SSI Receive Data Ready (RDR) bit in the SCSR is set.

With an enabled FIFO, the maximum of eight values are available to be read. If FIFO is not enabled, just one value can be read from the SRX register. When the RIE bit is cleared, the interrupt is disabled. However, the RFF and RDR bits still indicates the Receive Data register full condition. Reading the SRX register clears the RDR bit, thus clearing the pending interrupt.

Two Receive Data Interrupts with separate interrupt vectors are available:

1. Receive data with exception status
2. Receive data without exception status

Table 12-12 illustrates the conditions of the generated interrupts.

Table 12-12. SSI Receive Data Interrupts

Interrupt	RIE	ROE	RFF/RDF
Receive Data with Exception Status	1	1	1
Receive Data (without exception)	1	0	1

12.4.8.2 Transmit Interrupt Enable (TIE)—Bit 14

The SSI Transmit Interrupt Enable (TIE) control bit orchestrates the program controller interruption. When the TIE and TE bits are set, the program controller is interrupted when the SSI Transmit FIFO Empty (TFE) flag in the SCSR is set, but only when the transmit FIFO is enabled. If FIFO is not enabled, an interrupt is generated when SSI Transmit Data Empty (TDE) register flag in the SCSR is set and the TE bit is set.

When the transmit FIFO is enabled, eight values can be written to the SSI. However, if the FIFOs are not enabled, only one value can be written to the STX register. When the TIE bit is cleared, this interrupt is disabled. The TDE bit always indicates the STX register empty condition, even when the transmitter is disabled by the TE bit, in the SCR2. Writing data to the STX or STSR clears the TDE bit, thus clearing the interrupt.

Two transmit data interrupts with separate interrupt vectors are available:

1. Transmit data with exception status
2. Transmit data without exceptions

Figure 12-4 illustrates the conditions of these interrupts.

Table 12-13. SSI Transmit Data Interrupts

Interrupt	TIE	TUE	TFE/TDE
Transmit Data with Exception Status	1	1	1
Transmit Data (without exception)	1	0	1

12.4.8.3 Receive Enable (RE)—Bit 13

The SSI Receive Enable (RE) control bit enables the receive portion of the SSI. When the RE bit is set, the receive portion of the SSI is enabled. When the RE bit is cleared, the receiver is disabled by inhibiting data transfer into the Receive FIFO (RX). If data is being received when this bit is cleared, any interrupted data is not shifted in nor is it transferred to the SRX register. However, if the RE bit is re-enabled during a time slot before the second-to-last bit, then the data will be received. Clearing this bit while clearing SSI Enable (SSIEN) is recommended.

12.4.8.4 Transmit Enable (TE)—Bit 12

The SSI Transmit Enable (TE) control bit enables the transfer of the contents of the STX register to the Transmit Shift register as well as enabling the internal gated clock. When the TE bit is set and a word boundary is detected, the transmit portion of the SSI is enabled. A cleared TE bit, results in the transmitter continuing to send the data currently in the SSI Transmit Shift register prior to disabling the transmitter. When the serial output enable signal is disabled, any current data in the STX register is not transmitted. That is, data can be written to the STX register with a cleared TE bit. The TDE bit is cleared but data is not transferred to the Transmit Shift register. A cleared TE bit set again during the same transmitted word permits continuation of transmitted data. If the TE bit is set again during a different time slot, data is not transmitted until the next word boundary.

The normal transmit-enable sequence is to write data to the STX register or to the STSR before setting the TE bit. The normal transmit-disable sequence is to clear the TE bit and the TIE bit after the TDE bit is set. When an internal gated clock is being used, the gated clock runs during valid time slots if the TE bit is set. When the TE bit is cleared, the transmitter continues to send the data currently in the SSI Transmit Shift register until it is empty. The clock then stops. When the TE bit is set again, the gated clock starts immediately, running during any valid time slots. This bit should be cleared when clearing SSIEN.

12.4.8.5 Receive FIFO Enable (RFEN)—Bit 11

The Receive FIFO Enable (RFEN) control bit enables the FIFO register for the receive section. When the RFEN bit is set, eight samples are permitted to be received by the SSI. A ninth sample can be shifting in before the RDR bit is set. An interrupt request is generated when enabled by the RIE bit. Results of the FIFO register not being used are cleared RFEN bits generating an interrupt request. This occurs when a single sample is received by the SSI. Interrupts must be enabled.

12.4.8.6 Transmit FIFO Enable (TFEN)—Bit 10

The Transmit FIFO Enable (TFEN) control bit enables the FIFO register for the transmit section. The TFEN bit set, a maximum of eight samples can be written to the SSI. A ninth sample can be shifting out before the TDE bit is set, and an interrupt request can be generated when enabled by the TIE bit. When the TFEN bit is cleared, however, the FIFO register is not used. It results in an interrupt request being generated when a single sample needs to be written to the SSI.

12.4.8.7 Receive Clock Direction (RXDIR)—Bit 9

The Receive Direction (RXDIR) bit selects the direction and source of the clock used to clock the RXSR. Conditionally, if the RXDIR bit is set, the clock is generated internally and it is output to the SSI_RCK_OUT pin, but only if it is not configured as GPIO. When the RXDIR bit is cleared, the clock source is external. The internal clock generator is disconnected from the

SSI_RCK_OUT pin, and an external clock source can drive this pin to clock the RXSR. **Figure 12-4** defines the clock pin configuration.

Note: RXDIR and SYN must both be set for the SSI to be in gated clock mode.

Table 12-14. Frame Sync and Clock Pin Configuration

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRFS	STFS	SRCK	STCK
Asynchronous Mode								
0	0	0	0	0	RFS IN	TFS IN	RCK IN	TCKIN
0	0	1	0	1	RFS IN	TFS OUT	RCK IN	TCK OUT
0	1	0	1	0	RFS OUT	TFS IN	RCK OUT	TCK IN
0	1	1	1	1	RFS OUT	TFS OUT	RCK OUT	TCK OUT
Synchronous Mode								
1	0	0	—	0	GPIO	FS IN	GPIO	CK IN
1	0	1	—	1	GPIO	FS OUT	GPIO	CK OUT
1	1	0	—	—	GPIO	GPIO	GPIO	Gated IN
1	1	1	—	—	GPIO	GPIO	GPIO	Gated OUT

12.4.8.8 Transmit Clock Direction (TXDIR) - Bit 8

The Transmit Direction (TXDIR) control bit selects the direction and source of the clock used to time the TXSR. If it is not configured as GPIO and when the TXDIR bit is set, the clock is generated internally and is an output to the SSI_TCK_OUT pin. When cleared, the TXDIR bit clock source is external. The internal clock generator is disconnected from the SSI_TCK_OUT pin. An external clock source can drive this pin to clock the TXSR.

12.4.8.9 Synchronous Mode (SYN)—Bit 7

The Synchronous (SYN) mode control bit enables the Synchronous mode of operation. In this mode, the transmit and receive sections share a common clock pin and frame sync pin. SYN, along with RXDIR, controls the Gated Clock mode. The SSI is in a Gated mode when both SYN and RXDIR are set.

12.4.8.10 Transmit Shift Direction (TSHFD)—Bit 6

The Transmit Shift Direction (TSHFD) control bit determines whether the most or least significant bit is transmitted first for the transmit section. If the TSHFD bit is set, the LSB is transmitted first. If the TSHFD bit is cleared, the data is transmitted to the MSB first.

Note: The codec device labels the MSB as bit zero, whereas the SSI labels the LSB as bit zero. Therefore, when using a standard codec, the SSI MSB, or codec bit zero, is shifted out first and the TSHFD bit should be cleared.

12.4.8.11 Codec Device Labels MSB Transmit Clock Polarity (TSCKP)–Bit 5

The Transmit Clock Polarity (TSCKP) control bit determines which bit clock edge is used to clock out data and latch-in data for the transmit section. If the TSCKP bit is set, the falling edge of the bit clock is used to clock data out. If the TSCKP bit is cleared, the data is clocked out on the rising edge of the bit clock.

12.4.8.12 SSI Enable (SSIEN)—Bit 4

The SSI Enable (SSIEN) control bit enables and disables the SSI. If the SSIEN bit is set, the SSI is enabled, causing an output frame synchronous to be generated when set up for internal frame synchronization. Or it causes the SSI to wait for the input frame synchronous when set up for external frame synchronization. If the SSIEN bit is cleared, the SSI is disabled. Disabled, $\overline{\text{SSI\_TCK\_OEN}}$, $\overline{\text{SSI\_TFS\_OEN}}$, and $\overline{\text{SSI\_TFS\_OEN}}$ pins are inactive.

The Status register bits are preset to the same state produced by the power-on reset. The control register bits are unaffected. When the contents of transmit FIFO and receive FIFO are cleared this bit is reset. Reset SSI by writing zero to the bit. When SSI is disabled, all internal clocks are disabled, except clocks for register access. When clearing SSIEN, it is recommended to also clear RE and TE.

12.4.8.13 Network Mode (NET)—Bit 3

The Network (NET) mode control bit selects the operational mode of the SSI. When the NET bit is cleared, the Normal mode is selected. When the NET bit is set, the Network mode is selected.

12.4.8.14 Frame Sync Invert (TFSI)—Bit 2

The Frame Sync Invert (TFSI) control bit selects the logic of frame sync I/O. If the TFSI bit is set, the Frame Sync is active low. If the FSL bit is cleared, the frame sync is active high.

12.4.8.15 Frame Sync Length (TFSL)—Bit 1

The Frame Sync Length (TFSL) control bit selects the length of the frame sync signal to be generated or recognized. If the TFSL bit is set, a one-clock-bit-long frame synchronization is selected. The FSL bit cleared, a one-word-long frame sync is selected. The length of this single word-long frame sync is selected as the length of the data word selected by WL bits.

12.4.8.16 Early Frame Sync (TEFS)—Bit 0

The Early Frame Sync (TEFS) control bit determines when the frame sync is initiated for the transmit and receive sections. When the TEFS bit is cleared, the frame sync is initiated as the first bit of data transmitted or received. The TEFS bit set, the frame sync is initiated one bit before the data is transmitted or received. The frame sync is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync.

12.4.9 SSI Transmit and Receive Control Registers

The SSI Transmit and Receive Control Registers (SRXCR and STXCR) are 16-bit, read/write control registers used to direct the operation of the SSI. These two registers control the SSI clock generator bit and frame synchronization rates, word length, and number of words per frame for the serial data. The STXCR register is dedicated to the transmit section. The SRXCR register is devoted to the receive section, except in Synchronous mode. In that case, the Synchronization mode, the STXCR register controls both the receive and transmit sections. Power-on reset clears all STXCR and SRXCR bits. Control bits are described in the following paragraphs.

Although the bit patterns of the SRXCR and SRXCR registers are the same, the contents of these two registers can be programmed differently.

SSI_BASE+$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-15. SSI Transmit Register (STXCR)

See Programmer's Sheet on Appendix page B-80

SSI_BASE+$5	32-16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RSV	PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
Write																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-16. SSI Receive Control Register (SRXCR)

See Programmer's Sheet on Appendix page B-81

12.4.9.1 Prescaler Range (PSR)—Bit 15

The Prescaler Range (PSR) control bit manages a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is desired. When the PSR bit is cleared, the fixed prescaler is bypassed. When the PSR bit is set, the fixed divide-by-eight prescaler is operational. Its use allows a 128kHz master clock to be generated for MC1440x series codecs. The maximum internally generated bit clock frequency is $F_{OSC}/4$, and the minimum internally generated bit clock frequency is $F_{OSC}/(4 \times 8 \times 256)$.

12.4.9.2 Word Length Control (WL)—Bits 14–13

The Word Length (WL) control bits are used to select the length of the data words being transferred by the SSI. Word lengths of 8, 10, 12, or 16 bits can be selected, as shown in **Table 12-1.**

Table 12-1. SSI Data Word Lengths

WL1	WL0	Number of Bits/Word
0	0	8
0	1	10
1	0	12
1	1	16

These bits control the word length divider shown in the SSI clock generator. The WL control bits also control the frame synchronization pulse length when the FSL bit is cleared.

12.4.9.3 Frame Rate Divider Control (DC)—Bits 12–8

The Frame Rate Divider Control (DC) bits control the divide ratio for the programmable frame rate dividers. The divide ratio operates on the word clock. In Normal mode, this ratio determines the word transfer rate. While in the Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 (DC = 00000 to 11111) in the Normal mode, and from 2 to 32 (DC = 00001 to 11111) in the Network mode.

A divide ratio of one (DC = 00000) in the Network mode is a special case. It is a demand mode. In the Normal mode, a divide ratio of one (DC = 00000) provides continuous periodic data word transfer. A bit-length frame synch must be used in this case.

12.4.9.4 Prescale Modulus Select (PM)—Bits 7–0

The Prescale Modulus (PM) select control bits specify the divide ratio of the prescale divider in the SSI clock generator. This prescaler is used only in internal clock mode to divide the internal clock of the core. A divide ratio from 1 to 256 (PM = $00 to $FF) can be selected. The bit clock output is available at the clock SCK. The bit clock on the SSI can be calculated from the system clock value using the equation in **Figure 12-17.**

$$f_{INT\_BIT\_CLK} = f_{FIX\text{-}CLK} / [4 \times (7 \times PSR + 1) \times (PM + 1)]$$

where PM = PM[7:0]

$$f_{FRAME\_SYN\_CLK} = (f_{INT\_BIT\_CLK}) / [(DC + 1) \times WL]$$

where DC = DC[4:0] and WL = (8, 10, 12, or 16)

$$f_{FIX\_CLK} = f_{mcu\_clk} /4 *$$

$$f_{FIX\_CLK} = f_{mcu\_clk} **$$

* if DIV4DIS = 0

**if DIV4DIS = 1

Figure 12-17. SSI Bit Clock Equation

Illustratively, an 8-bit word in the Normal mode with DC set to 1 (00001), PM set to 71 (0100 0111), the PSR bit is cleared. And in a 36.864MHz system clock, a bit clock rate of 36.864MHz ÷ [1 × 4 × 72] = 128kHz is generated. Since the 8-bit word rate is equal to two, the sampling rate (FS rate) would then be 128kHz ÷ [2 × 8] = 16kHz.

The bit clock output is also available internally for use as the bit clock to shift the Transmit and Receive Shift registers. Careful choice of the crystal oscillator frequency and the prescaler modulus allows the telecommunication industry-standard codec master clock frequencies of 2.048MHz, 1.544MHz, and 1.536MHz to be generated. For example, a 24.576MHz clock frequency can be used to generate the standard 2.048MHz and 1.536 MHz rates, and a 24.704MHz clock frequency can be used to generate the standard 1.544 MHz rate. **Table 12-2** illustrates PM values available to generate different bit clocks.

Table 12-2. Chip Clock Rates as a Function of SSI Bit Clock Frequency and Prescale Modulus

Frame Rate	Words/Frame (DC+1)	Bits/ Word	Bit Clock Rate (Hz)	PM	PSR	FIX_CLK Rate	DIV4DIS	IP_CLK Rate[a]	SYS_CLK Rate[b]
8000	32	8	2,048,000	6	0	57,344,000	1	57,344,000	114,688,000
8000	32	8	2,048,000	7	0	65,536,000	1	65,536,000	131,072,000
8000	24	8	1,536,000	8	0	55,296,000	1	55,296,000	110,592,000
8000	24	8	1,536,000	9	0	61,440,000	1	61,440,000	125,880,000
8000	2	10	160,000	10	1	56,320,000	1	56,320,000	112,640,000
8000	2	10	160,000	11	1	61,440,000	1	61,440,000	122,880,000
8000	2	10	160,000	22	0	14,720,000	0	58,880,000	117,760,000
8000	2	10	160,000	92	0	59,520,000	1	59,520,000	119,040,000
8000	2	10	160,000	93	0	60,160,000	1	60,160,000	120,320,000
8000	2	12	192,000	8	1	55,296,000	1	55,296,000	110,592,000
8000	2	12	192,000	9	1	61,440,000	1	61,440,000	122,880,000
8000	2	12	192,000	18	0	14,593,000	0	58,368,000	116,736,000
8000	2	12	192,000	77	0	59,904,000	1	59,904,000	119,808,000
8000	2	12	192,000	78	0	60,672,000	1	60,672,000	120,344,000
8000	2	16	256,000	6	1	57,344,000	1	57,344,000	114,688,000
8000	2	16	256,000	7	1	65,536,000	1	65,536,000	131,072,000
8000	2	16	256,000	13	0	14,336,000	0	57,344,000	116,736,000
8000	2	16	256,000	57	0	59,904,000	1	59,904,000	118,784,000
8000	2	16	256,000	58	0	60,416,000	1	60,416,000	120,832,000

a. **Shaded cells denote** configuration where the required clock rate is higher than the chip specification can support.

b. IP_CLK is 1/2 the SYS_CLK rate

12.4.10 SSI Time Slot Register (STSR)

The SSI Time Slot Register (STSR) is used when data is not to be transmitted in an available transmit time slot. For the purposes of timing, the Time Slot register is a *write-only* register behaving like an alternate Transmit Data register. Its exception is instead of transmitting data, the SSI_TXD_oen_b signal is disabled. Using this register is important to avoid overflow/underflow during inactive time slots.

SSI_BASE+$6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DUMMY REGISTER, WRITTEN DURING INACTIVE TIME SLOTS (NETWORK MODE)															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-18. SSI Time Slot Register (STSR)

See Programmer's Sheet on Appendix page B-82

12.4.11 SSI FIFO Control/Status Register (SFCSR)

SSI_BASE+$7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RFCNT				TFCNT				RFWM				TFWM			
Write																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Figure 12-19. SSI FIFO Control/Status Register (SFCSR)

See Programmer's Sheet on Appendix page B-83

12.4.11.1 Receive FIFO Counter (RFCNT)—Bit 15-12

These status bits illustrate the number of data words in the Receive FIFO, shown in **Table 12-3.**

Table 12-3. Number of Data Words Available

Bits	Description
0000	0 Data Word in Receive FIFO
0001	1 Data Word in Receive FIFO
0010	2 Data Words in Receive FIFO
0011	3 Data Words in Receive FIFO
0100	4 Data Words in Receive FIFO
0101	5 Data Words in Receive FIFO
0110	6 Data Words in Receive FIFO
0111	7 Data Words in Receive FIFO
1000	8 Data Words in Receive FIFO

12.4.11.2 Transmit FIFO Counter (TFCNT)—Bits 11-8

These status bits illustrate the number of data words in the Transmit FIFO. Please refer to **Table 12-4.**

Table 12-4. Data FIFO Transmit

Bits	Description
0000	0 Data Word in TxFIFO
0001	1 Data Word in TxFIFO
0010	2 Data Word in TxFIFO
0011	3 Data Word in TxFIFO
0100	4 Data Word in TxFIFO
0101	5 Data Word in TxFIFO
0110	6 Data Word in TxFIFO
0111	7 Data Word in TxFIFO
1000	8 Data Word in TxFIFO

12.4.11.3 Receive FIFO Full WaterMark (RFWM)—Bits 7-4

The Receive FIFO Full Watermark (RFWM) bits control the Receive FIFO Full flag threshold, determining its setting. Please see **Table 12-5.** Receive FIFO Full flag is set whenever the data level in the Receive FIFO reaches the selected threshold. For example, if RFWM = 1, Receive FIFO Full flag will be set after the SSI has received two data, one in Receive Data Register, and the other in Receive FIFO.

Table 12-5. Receive FIFO WaterMark

Bits	Description	Set When RxFIFO =
0000	Reserved	—
0001	RFF set when at least one data word has been written to the Receive FIFO	1,2,3,4,5,6,7,8, Data Words
0010	RFF set when more than or equal to two data words have been written to the Receive FIFO	2,3,4,5,6,7,8 Data Words
0011	RFF set when more than or equal to three data words have been written to the Receive FIFO	3,4,5,6,7,8 Data Words
0100	RFF set when more than or equal to four data words have been written to the Receive FIFO	4,5,6,7,8 Data Words
0101	RFF set when more than or equal to five data words have been written to the Receive FIFO	5,6,7,8 Data Words
0110	RFF set when more than or equal to six data words have been written to the Receive FIFO	6,7,8 Data Words
0111	RFF set when more than or equal to seven data words have been written to the Receive FIFO	7,8 Data Words
1000	RFF set when more than or equal to eight data words have been written to the Receive FIFO	8 Data Words

12.4.11.4 Transmit FIFO Empty WaterMark (TFWM)—Bits 3-0

These bits control the threshold setting of the Transmit FIFO Empty flag. Please see Table 12-6 for status of Transmit FIFO Empty flag will be set whenever the data level in the transmitter FIFO falls below the selected threshold. Please see Table 12-7 for TFWM settings.

Table 12-6. Transmit FIFO Empty Flag

Bits	Description	Transmit FIFO Empty is Set When TxFIFO is
0000	Reserved	—
0001	TFE set when there are more than, or equal to one empty slots in Transmit FIFO (default)	<=7 Data
0010	TFE set when there are more than, or equal to two empty slots in Transmit FIFO	<=6 Data
0011	TFE set when there are more than, or equal to three empty slots in Transmit FIFO	<=5 Data
0100	TFE set when there are more than, or equal to four empty slots in Transmit FIFO	<=4 Data
0101	TFE set when there are more than, or equal to five empty slots in Transmit FIFO	<=3 Data
0110	TFE set when there are more than, or equal to six empty slots in Transmit FIFO	<=2 Data
0111	TFE set when there are more than, or equal to seven empty slots in Transmit FIFO	<=1 Data
1000	TFE set when there are eight empty slots in Transmit FIFO	= 0 Data

Table 12-7. TFWM Settings

Transmit FIFO WaterMark (TFWM)	Number of Data in TXFIFO								
	0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	0	0	0
4	1	1	1	1	1	0	0	0	0
5	1	1	1	1	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0
7	1	1	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0

12.4.12 SSI Option Register (SOR)

SSI_BASE+$9	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	RFDIR	TFDIR	0	0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-20. SSI Option Register (SOR)

See Programmer's Sheet on Appendix page B-84

12.4.12.1 Receive Frame Direction (RFDIR)—Bit 5

The Receive Frame Direction (RFDIR) control bit selects the direction and source of the receive frame synchronous signal. When the RFDIR bit is set, the frame sync is generated internally and are output through the SSI_RFS_OUT pin, if it is not configured as GPIO. When the RFDIR bit is cleared, the receive frame sync is external, meaning the receive frame sync is supplied from an external source.

12.4.12.2 Transmit Frame Direction (TFDIR)—Bit 4

The Transmit Frame Direction (TFDIR) control bit selects the direction and source of the transmit frame synchronous signal. When the TFDIR bit is set, the frame sync is generated internally and it is output through the SSI_TFS_OUT pin, if it is not configured as GPIO. When the TFDIR bit is cleared, the transmit fame sync is external, meaning the transmit frame sync is supplied from an external source.

12.4.12.3 Initialize (INIT)—Bit 3

This bit is used to initialize the state machine to reset state. This bit (INIT) is for test purposes.

12.4.12.4 Wait (WAIT)—Bits 2-1

This bit controls the number of wait states to be added to the transaction between the CPU and SSI. On the 56F826, these are not used and should be left at 0 × 00 default value.

12.4.12.5 Frame Syn Reset (SYNRST)—Bit 0

The Frame Sync Reset (SYNRST) bit automatically resets the accumulation of data in receive FIFO (RXFIFO) on frame synchronization.

12.5 SSI Data and Control Pins

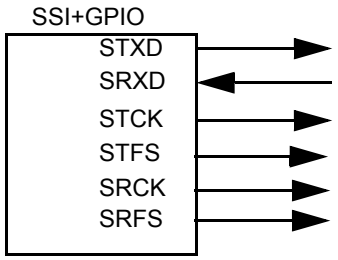
The SSI has the following dedicated pins:

- Serial Transmit Data (SSI_TXD_OUT)
- Serial Transmit Data Enable ($\overline{\text{SSI\_TXD\_OEN}}$)
- Serial Receive Data (SSI_RXD)
- Serial Transmit Clock Out (SSI_TCK_OUT)
- Serial Transmit Clock In (SSI_TCK_IN)
- Serial Transmit Clock Enable ($\overline{\text{SSI\_TCK\_OEN}}$)
- Serial Transmit Frame Sync Out (SSI_TFS_OUT)
- Serial Transmit Frame Sync In (SSI_TFS_IN)
- Serial Transmit Frame Sync Enable ($\overline{\text{SSI\_TFS\_OEN}}$)
- Serial Receive Clock Out (SSI_RCK_OUT)
- Serial Receive Clock In (SSI_RCK_IN)
- Serial Receive Clock Enable ($\overline{\text{SSI\_RCK\_OEN}}$)
- Serial Receive Frame Sync Out (SSI_RFS_OUT)
- Serial Receive Frame Sync In (SSI_RFS_IN)
- Serial Receive Frame Sync Enable ($\overline{\text{SSI\_RFS\_OEN}}$)

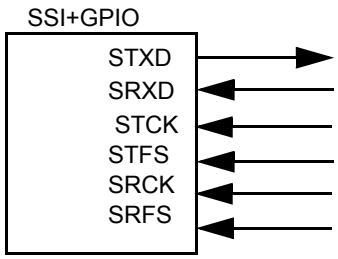
Figure 12-21 and **Figure 12-22** exhibit the main SSI configurations. These pins support all transmit and receive functions with a continuous or gated clock, as shown.

Note: Gated clock implementations do not require the use of the frame sync pins. In this case these pins can be used as GPIO pins, if desired.

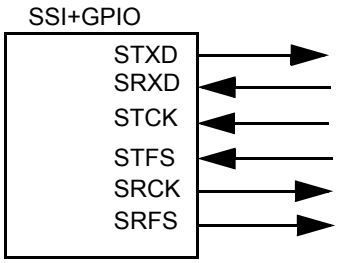
Note: This GPIO is a separate module itself, multiplexing and connecting the SSI data and control signals to the six dedicated I/O pins for SSI.



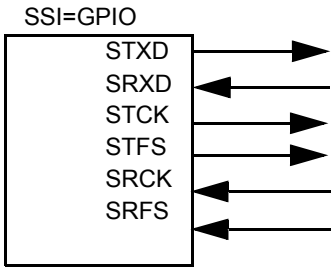
SSI Internal Continuous Clock (RXDIR=1, TXDIR=1, RFDIR=1, TFDIR=1, SYN=0)



SSI External Continuous Clock (RXDIR=0,TXDIR=0, RFDIR=0, TFDIR=0, SYN=0)



SSI Continuous Clock (RXDIR=1, TXDIR=0, RFDIR=1, TFDIR=0, SYN=0)



SSI Continuos Clock (RXDIR=0, TXDIR=1, RFDIR=0, TFDIR=1, SYN=0)

Figure 12-21. Asynchronous (SYN=0) SSI Configurations-Continuous Clock

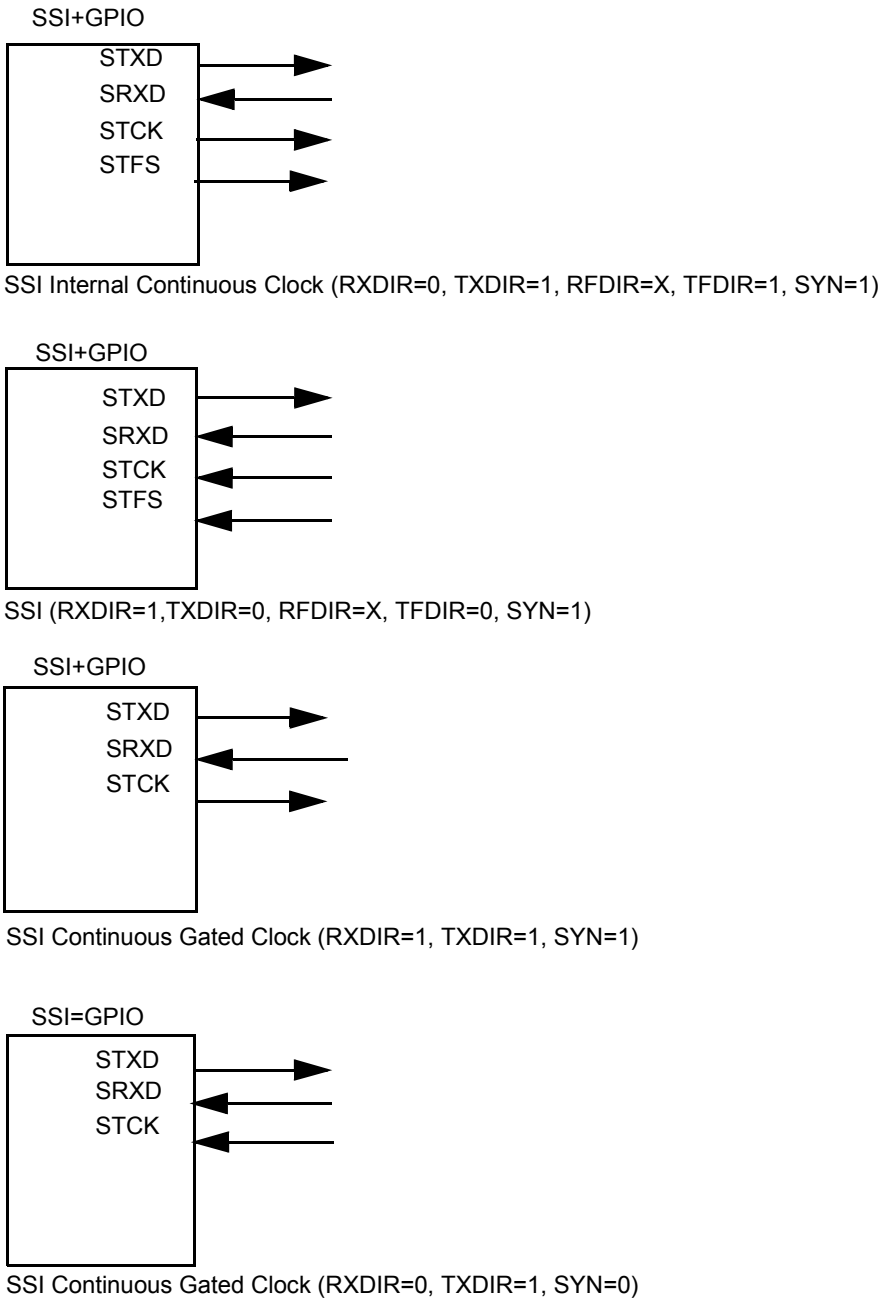


Figure 12-22. Synchronous SSI Configuration-Continuous and Gated Clock

12.6 Configuration of the SSI Pins

The following bulleted items describe the configuration of the SSI pins.

- Serial Transmit Data Out—The SSI_TXD_OUT pin transmits data from the Serial Transmit Shift register. The SSI_TXD_OUT pin is an output pin when data is being transmitted and is inactive, or low, between data word transmissions and on the trailing edge of the bit clock after the last bit of a word is transmitted.
- Serial Transmit Data Enable—This allows the SSI_TXD_OUT to pass through the GPIO to the port. If this is not active, the corresponding port pin can be used as GPIO.
- Serial Receive Data—The SSI_RXD pin is used to bring serial data into the Receive Data Shift register.
- Serial Transmit Clock—The SSI_TCK_OUT pin is an output and the SSI_TCK_IN pin an input. This clock signal is used by the transmitter and can be either continuous or gated. During Gated Clock mode, data on the SSI_TCK_OUT pin is valid only during the transmission of data, otherwise it is inactive or low. In Synchronous mode, this pin is used by both the transmit and receive sections.
- Serial Transmit Clock Enable—This allows the SSI_TCK_OUT to pass through the GPIO to the port. If this is not active, the corresponding port pin can be used as GPIO.
- Serial Transmit Frame Sync—The SSI_TFS_OUT pin is an output and SSI_TFS_IN pin is an input. The frame sync is used by the transmitter to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. In Synchronous mode, this pin is used by both the transmit and receive sections. In Gated Clock mode, frame sync signals are not used.
- Serial Transmit Frame Sync Enable—This allows the SSI_TFS_OUT to pass through the GPIO to the port. If this is not active, the corresponding port pin can be used as GPIO.
- Serial Receive Clock—The SSI_RCK_OUT pin is an output and SSI_RCK_IN is an input. This clock signal is used by the receiver and is always continuous. During Gated Clock mode, the pin is used instead for clocking in data. This pin is not used in Synchronous mode.
- Serial Receive Clock Enable—This allows the SSI_RCK_OUT to pass through the GPIO to the port. If this is not active, the corresponding port pin can be used as GPIO.
- Serial Receive Frame Sync—The SSI_RFS_OUT pin is an output and SSI_RFS_IN pin is an input. The frame sync is used by the receiver to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data.

- Serial Receive Frame Sync Enable—This allows the SSI_RFS_OUT to pass through the GPIO to the port. If this is not active, the corresponding port pin can be used as GPIO.

An example of pin signals for an 8-bit data transfer is illustrated in **Figure 12-23.** Continuous and gated clock signals are shown, as well as the bit-length frame sync signal and the word-length frame sync signal.

Note: The shift direction can be defined as MSB first, or LSB first. There are other options on the clock and frame sync.

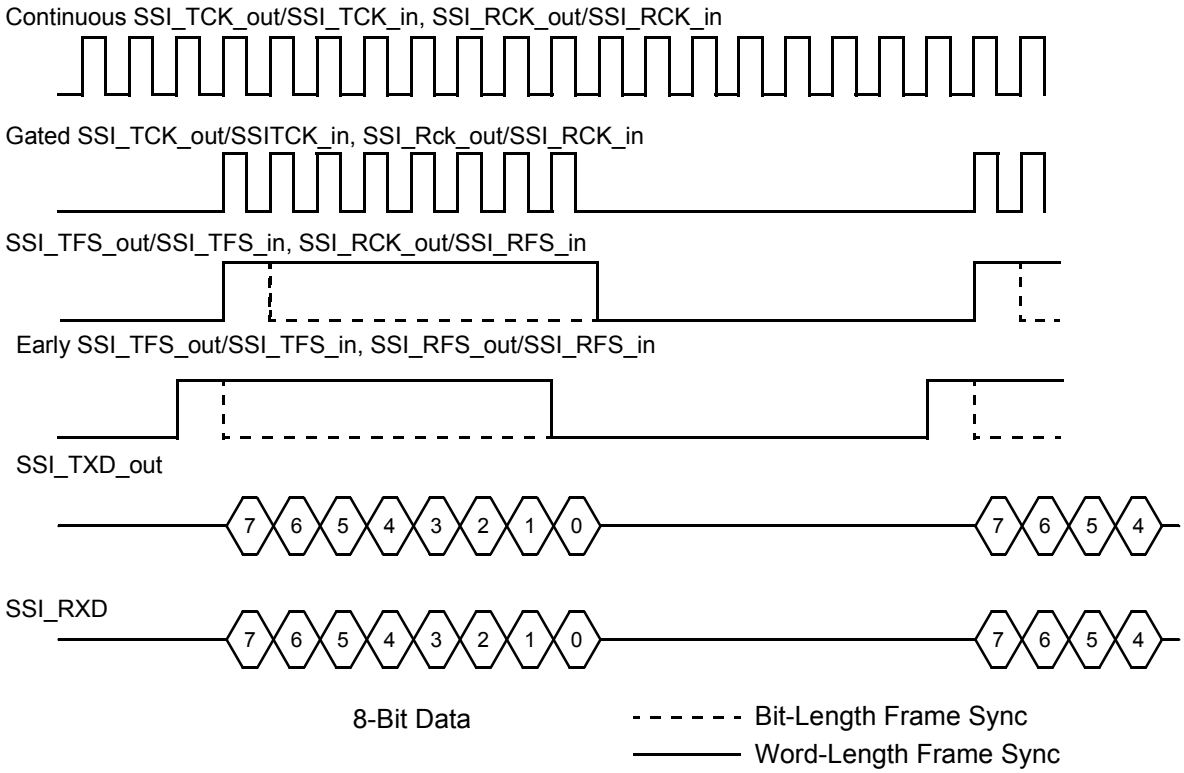


Figure 12-23. Serial Clock and Frame Sync Timing

12.7 Operating Modes

The SSI has three basic operating modes, with the option of asynchronous or synchronous protocol, as follows:

1. Normal mode
   - — Asynchronous protocol
   - — Synchronous protocol
2. Network mode
   - — Asynchronous protocol

— Synchronous protocol

3. Gated clock mode

— Synchronous protocol only

These modes can be programmed by several bits in the SSI control registers. **Table 12-8** lists these operating modes and some of their typical applications:

Table 12-8. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Applications
Asynchronous	Continuous	Normal	Multiple Synchronous Codecs
Asynchronous	Continuous	Network	TDM Codec or Networks
Synchronous	Continuous	Normal	Multiple Synchronous Codecs
Synchronous	Continuous	Network	TDM Codec or Networks
Synchronous	Gated	Normal	SPI-Type Devices; to MCU

The transmit and receive sections of the SSI can be synchronous or asynchronous. In the Sync mode, the transmitter and the receiver use a common clock and frame sync signal. In the Asynchronous mode, the transmitter and receiver each has its own clock and frame sync signals. Continuous or gated Clock mode can be selected. In a Continuous mode, the clock runs continuously. In gated Clock mode, the clock is only functioning during transmission.

Normal or Network modes can also be selected. In Normal mode, the SSI functions with one data word of I/O per frame. In Network mode, any number from 2 to 32-data words of I/O per frame can be used. The Network mode is typically used in star or ring time division multiplex networks with other processors or codecs, thereby allowing an interface to time division multiplexed networks without additional logic. Gated clock can not be used in the Network mode. These distinctions result in the basic operating modes allowing the SSI to communicate with a wide variety of devices.

The SSI supports both the Normal and Network modes. They can be selected independently of the transmitter and receiver as either synchronous or asynchronous. Typically, these protocols are used in a periodic manner where data is transferred at regular intervals, such as at the sampling rate of an external codec.

Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with a continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC bits in either the SRXCR or STXCR register, depending on whether data is being transferred or received. The number of words transferred per frame depends on the mode of the SSI.

In the Normal mode, one data word is transferred per frame. In the Network mode, the frame is divided into anywhere between two and 32-time slots where, in each time slot, one-data word can optionally be transferred.

12.7.1 Normal Mode

The Normal mode is the simplest mode of the SSI. It is used to transfer one word per frame. In the continuous Clock mode, a frame sync occurs at the beginning of each frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock, PSR, PM bits for internal clock or the frequency of the external clock on the SSI_TCK_IN pin
- The number of bits per sample, WL bits
- The number of time slots per frame, DC bits

If a Normal mode is configured to provide more than one time slot per frame, data is transmitted only in the first time slot. No data is transmitted in subsequent time slots.

12.7.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in a Normal mode are:

- SSI enabled, SSIEN = 1
- Enable FIFO and configure Transmit and Receive Watermark if FIFO is used
- Write data to the SSI Transmit Data (STX) register
- Transmitter Enabled (TE) = 1
- Frame sync active for continuous clock case
- Bit clock begins for gated clock case

When the preceding conditions occur in Normal mode, the next data word is transferred into theTXSR from the STX, or from the Transmit FIFO register if it is enabled. The new data word is transmitted immediately. If transmit FIFO is not enabled the TDE bit is set and the transmit interrupt occurs when the TIE bit is set. When FIFO is enabled, TDE bit is set then a transmit interrupt occurs when the TIE bit is set. Eight values must also have been transferred to the TXSR. If the Transmit FIFO is enabled, an eighth data word can be transferred and shifted prior to writing new data to the STX register.

The $\overline{\text{SSI\_TXD\_OEN}}$ signal is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not inactive, even if both receiver and transmitter are disabled.

12.7.1.2 Normal Mode Receive

The conditions for data reception from the SSI are when:

- SSI is enabled, SSIEN = 1
- Receiver Enabled (RE) = 1
- Frame sync is active for continuous clock case
- Bit clock begins for gated clock case

With the preceding conditions, in Normal mode with a continuous clock, each time the frame sync signal is generated, or detected, a data word is clocked in. With the previous conditions and a gated clock, each time the clock begins, a data word is clocked in. If Receive FIFO is not enabled, after receiving the data word, it is transferred from the RXSR to the SRX. The RDF flag is set, and the Receive Interrupt occurs when enabled and the RIE bit is set. If the Receive FIFO is enabled, after receiving the data word, it is transferred to the Receive FIFO. The RFF flag is set if the SRX is full and the Receive FIFO register reaches the select threshold. The Receive FIFO register reaches the select threshold and the Receive Interrupt occurs when the RIE bit is set.

The SSI program has to read the data from the SRX before a new data word is transferred from the RXSR; otherwise the ROE bit is set. If Receive FIFO is enabled, the ROE bit is set when both the SRX and the receive FIFO register contain data and a new data word is ready to be transferred to the Receive FIFO. **Figure 12-24** illustrates the transmitter and receiver timing for an 8-bit word having two words per time slot in normal mode and the continuous clock with a late length frame synchronization.

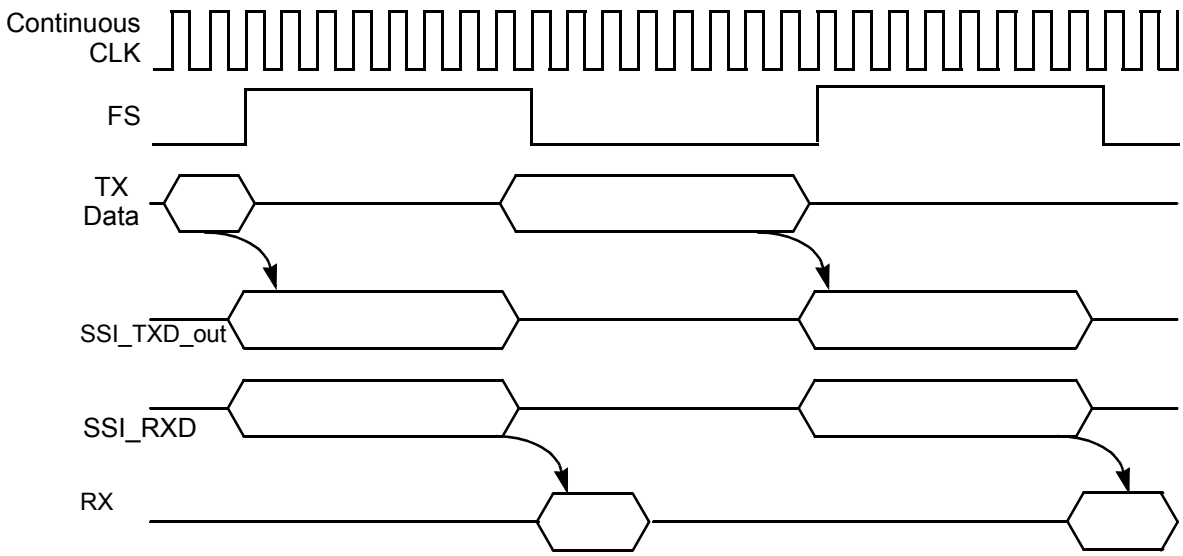


Figure 12-24. Normal Mode Timing—Continuous Clock

Figure 12-25 illustrates a similar case for a gated clock.

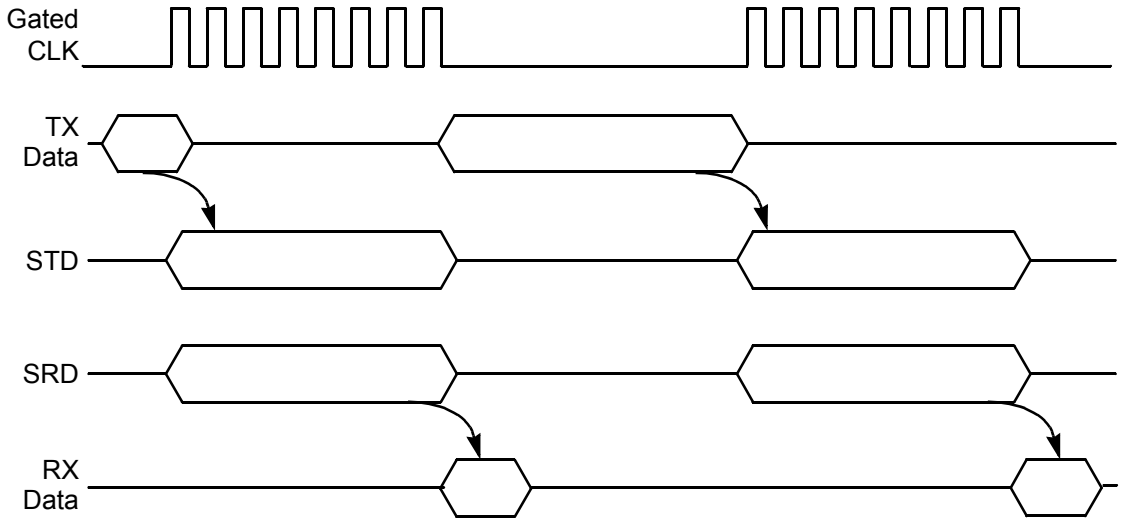


Figure 12-25. Normal Mode Timing—Gated Clock

12.7.2 Network Mode

The Network mode is used for creating a Time Division Multiplexed (TDM) network, such as a TDM codec network or a network of devices. In the continuous Clock mode, a frame sync occurs at the beginning of each frame. In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate codec or on the network. The can be a master device controlling its own private network, or a slave device connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots, and transmission and/or reception of one data word can occur in each time slot, rather than in just the frame sync time slot as in Normal mode. The frame rate dividers, controlled by the five Frame Rate Divider Control (DC) bits, select 2 to 32-time slots per frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock Prescaler Range (PSR) and the eight PM bits for internal clock, or the frequency of the external clock on the SSI_TCK_IN pin
- The number of bits per sample, two WL bits
- The number of time slots per frame, five DC bits

In the Network mode, data can be transmitted in any time slot. The distinction of the Network mode is: each time slot is identified with respect to the frame synchronization, data word time. This time slot identification allows the option of transmitting data during the time slot by writing to the STX register or ignoring the time slot by writing to STSR. The receiver is treated in the same manner. The exception to this is when data is always being shifted into the RXSR and transferred to the SRX register. The core reads the SRX register and either uses the data or discards it.

12.7.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SCR2 are both set. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new time slot. The condition is only if the TE bit is set during a slot other than the first. Software has to find the start of the next frame.

The normal startup sequence for transmission is to do the following:

1. Write the data to be transmitted to the STX register. This clears the TDE flag.
2. Set the TE bit enabling the transmitter on the next word boundary for a continuous clock case.
3. Enable transmit interrupts.

If not transmitting in a time slot by writing to the STSR, this clears the TDE flag just as if data were going to be transmitted. The $\overline{\text{SSI\_TXD\_OEN}}$ signal remains disabled during the time slot.

When the frame sync is detected, or generated continuous clock, the first enabled data word is transferred from the STX register to the TXSR and is shifted out, or transmitted. If the STX register is empty, and the TDE bit is set, a transmitter interrupt occurs and is sent when the TIE bit is set. Software can poll the TDE bit or use interrupts to reload the STX register, or writing to the STSR, before the TXSR is finished shifting, or emptying, causing a transmitter underrun. The TUE error bit to be set, and the $\overline{\text{SSI\_TXD\_OEN}}$ is disabled for the next time slot.

Clearing the TE bit disables the transmitter after the completion of transmission of the current data word. Setting the TE bit enables the transmission of the next word. During that time, the $\overline{\text{SSI\_TXD\_OEN}}$ signal is disabled. The TE bit should be cleared after the TDE bit is set, ensuring all pending data is transmitted.

Summarizing, the Network mode transmitter generates interrupts every enabled time slot, requiring the program to respond to each enabled time slot. These responses may be one of the following:

- Write the Data register with data, enabling transmission in the next time slot
- Write the Time Slot register, disabling transmission in the next time slot
- Do nothing—transmitter underrun occurs at the beginning of the next time slot and the previous data is retransmitted

12.7.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SCR2 are set. However, the Receive Enable only takes place during that time slot if RE is enabled before the second-to-last bit of the word. If the RE bit is cleared, the receiver is disabled immediately.

SSI is capable of finding the start of the next frame automatically when the word is completely received, it is transferred to the SRX register, setting the Receive Data Ready bit. Setting the RDR bit causes a receive interrupt to occur if the receiver interrupt is enabled, or the Receiver Interrupt Enable (RIE) bit is set. The second data word, second time slot in the frame, begins shifting in immediately after the transfer of the first data word to the SRX register. The program has to read the data from the receive data register, clearing Receive Data Full (RDF) before the second data word is completely received and is ready to transfer to RX data register, or a receiver overrun error occurs, causing the ROE bit to set.

An interrupt can occur after the reception of each enabled data word, or the programmer can poll the RDF flag. The SSI program response can be one of the following:

- Read RX and use the data
- Read RX and ignore the data
- Do nothing—the receiver overrun exception occurs at the end of the current time slot

Note: For a continuous clock, the optional frame synchronization output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame synchronization generation. The only way to disable the bit clock and the frame synchronization generation is to disable the SSIEN bit in the SCR2.

The transmitter and receiver timing for an 8-bit word with a continuous clock, FIFO disabled, and three words per frame synchronization in the network mode is shown in **Figure 12-26.**

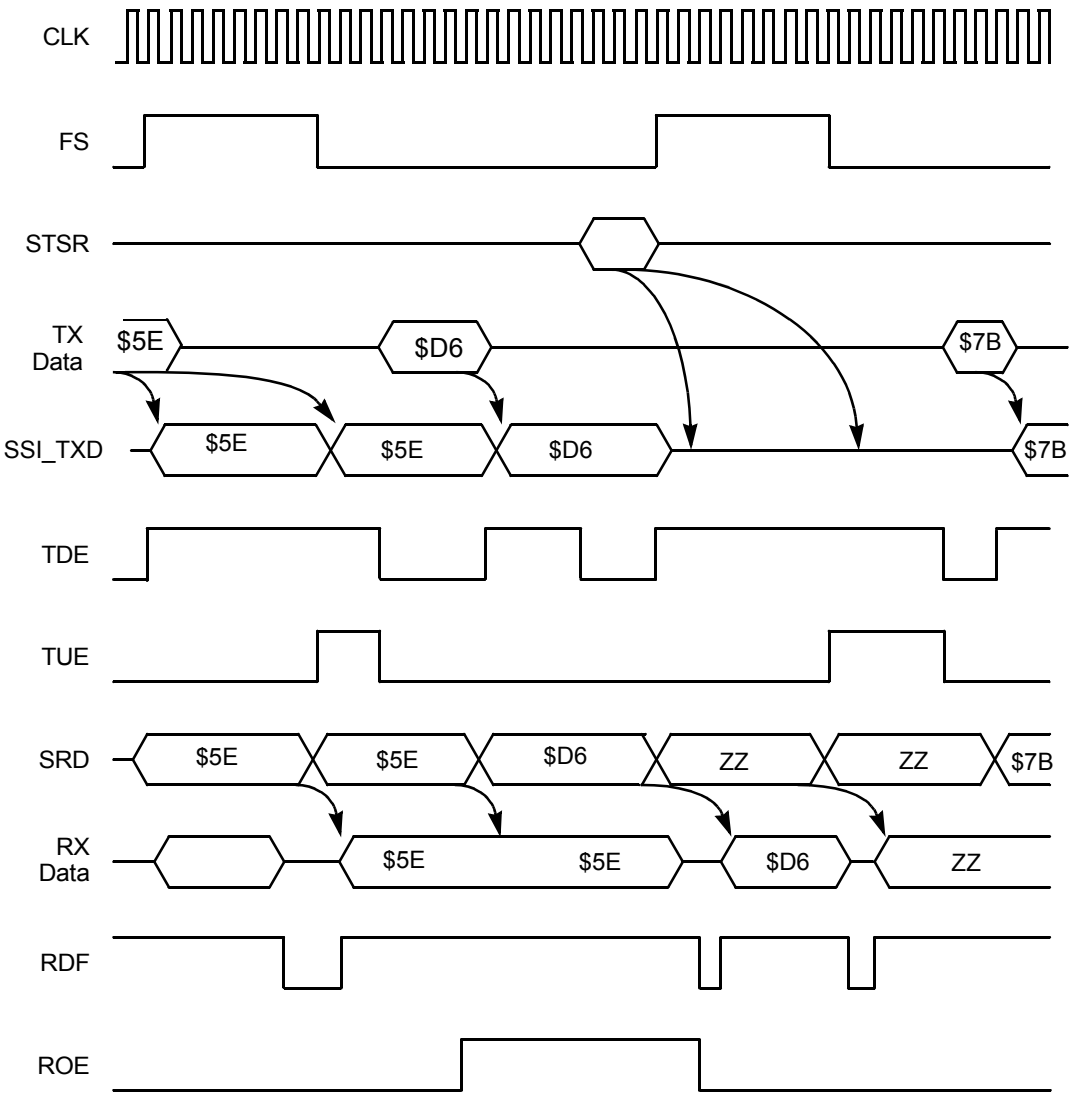


Figure 12-26. Network Mode Timing—Continuous Clock

12.7.3 Gated Clock Operation

Gated Clock mode is often used to hook up to SPI-type interfaces on microcontroller units or external peripheral chips. In the Gated Clock mode, the presence of the clock indicates valid data is on the SSI_TXD_OUT or SSI_RXD pins. For this reason, no frame synchronization is required in this mode. Once the transmission of data has completed, the clock enable pin is disabled. Gated clocks are allowed for both the transmit and receive sections with either an internal or external clock and in normal mode. Gated clocks are not allowed in Network mode.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. For the case of an internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs, such as the first time slot in Normal mode, the internal bit clock is enabled onto the appropriate clock pin. This allows data to be transferred out in periodic intervals in Gated Clock mode. With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in.

Note: The bit clock pins must be kept free of timing glitches. If a single glitch occurs, all ensuing transfers will be out of synchronization.

Note: It is advisable to set the RE bit only when setting the TE bit

12.8 Reset and Initialization Procedure

The SSI is affected by two types of resets:

1. Power-on Reset—The power-on reset is generated by asserting either the RESET pin or the computer operating properly timer reset. The power-on Reset clears the SSIEN bit in SCR2, disabling the SSI.
2. SSI Reset—The SSI reset is generated when the SSIEN bit in the SCR2 is cleared. The SSI status bits are preset to the same state produced by the power-on reset. The SSI control bits are unaffected. The control bits in the top half of the SCSR are also unaffected. The SSI reset is useful for selective resetting of the SSI without changing the present SSI control bits and without affecting the other peripherals.

The three correct sequences initializing the SSI:

1. Issue a power-on or SSI reset.
2. Program SSI control registers.
3. Set the SSIEN bit in SCR2.

To ensure proper operation of the SSI, use the power-on or SSI reset before changing any of the following control bits listed in **Table 12-9.**

Note: These control bits should not be changed during SSI operation.

Table 12-9. SSI Control Bits Requiring Reset Before Change

Control Register	Bit
SRXCR STXCR	WL0 WL1
SCR2	TEFS TFSI TFSL NET RBF RXD TSCKP SYN TBF TXN
SCSR	REFS RFSI RFSL RSCKP RSHFD

Chapter 12
Quad Timer Module (TMR)

12.1 Introduction

The 56F826/827 has one Quad Timer module. Dedicated pins of Timer module A are discussed in this chapter. Timer module A contains four identical counter/timer groups, or channels. Each 16-bit counter/timer group contains the following:

- Prescaler
- Counter
- Load register
- Hold register
- Capture register
- Two Compare registers
- Status and Control register
- Control register

All of the registers, except the prescaler, are read/write registers.

Note: This document uses the terms *timer* and *counter* interchangeably because the counter/timers may perform either, or both tasks.

The prescaler provides different time bases useful for clocking the counter/timer. The counter provides the ability to count internal or external events. 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 counters are being read. This feature supports the reading of cascaded counters.

The Capture register implements an external signal to take a snapshot of the counter's current value.

Providing values enabling counters to be compared is the Compare registers' job. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled. Within a time module there is a set of four counter/timers allowing the input pins to be apportioned.

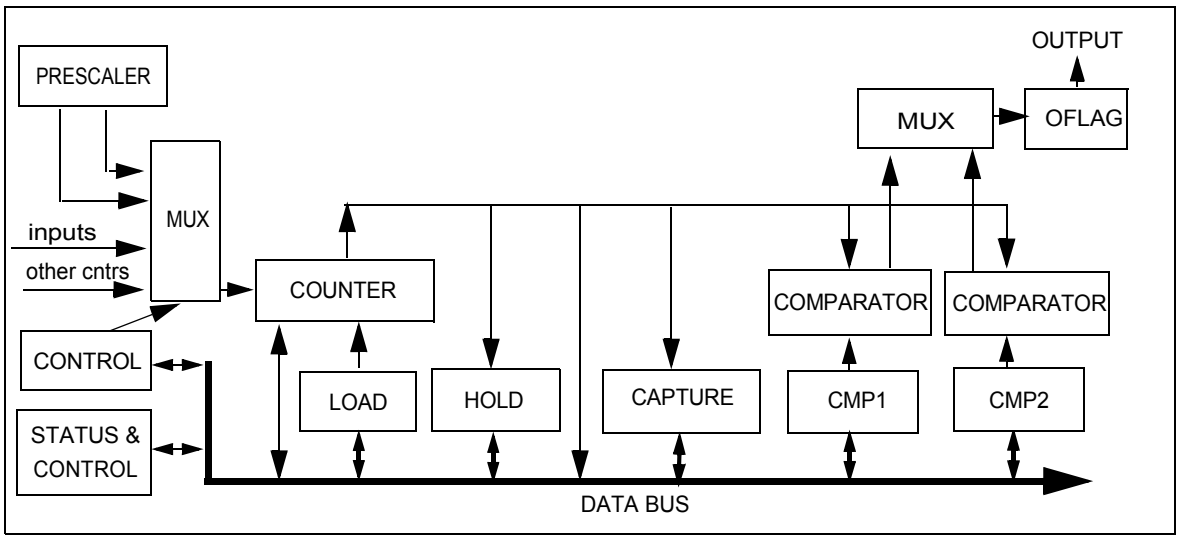


Figure 12-1. 56F826 Counter/Timer Block Diagram

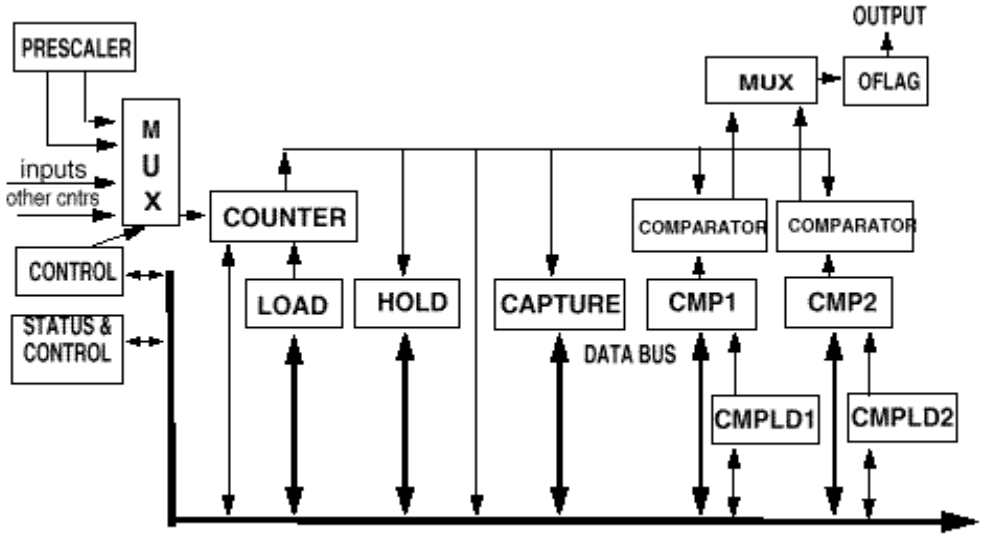


Figure 12-2. 56F827 Counter/Timer Block Diagram

12.2 Features

- Each timer module consists of four, 16-bit counters/timers
- Count up/down
- Counters may be cascaded
- Programmable count modulo
- Maximum count rate equals peripheral clock rate/2 when counting external events
- Maximum count rate equals peripheral clock rate when using internal clocks
- Count once or repeatedly
- Counters are preloadable
- Compare registers are preloadable only on the 56F827
- Counters can share available input pins
- Each counter has a separate prescaler
- Each counter has capture and compare capability

12.3 Pin Descriptions

Pins available for the 56F826 and 56F827 are different. Please pay special attention to the available pins discussed in **Section 12.8.**

12.4 Register Summary

A suffix is added to each register reflecting Quad Timer module A and channel, or group, being accessed: TMRA0, TMRA1, TMRA2, TMRA3.

For example, the CNTR register for Quad Timer A, channel zero TMRA0 module is called TMRA0_CNTR.

Each timer/counter in the 56F80x has the following registers:

- Quad Timer Counter (CNTR) register
- Quad Timer Load (LOAD) register
- Quad Timer Hold (HOLD) register
- Quad Timer Capture (CAP) register
- Quad Timer Compare (CMP1 and CMP2) registers
- Quad Timer Status and Control Register (SCR)
- Quad Timer Control (CTRL) register

Each timer/counter in the 56F827 has following additional registers:

- Quad Timer Comparator Load (CMPLD1 and CMPLD2) registers
- Quad Timer Comparator Status and Load Registers (COMSCR)

For information about TMRA registers please refer to **Table 3-11.**

12.5 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, thus timing the width of the external input signal.

12.5.1 Counting Options

The counter can count the rising, falling, or both edges of a selected input pin. The counter can decode and count quadrature encoded input signals. The counter can count up and down using dual inputs in a count with direction format. The counter's terminal count value, modulus, may be programmed. The value loaded into the counter after reaching its terminal count may also be programmed. The counter can count repeatedly, or it can stop after completing one count cycle. The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count moves to zero.

12.5.2 External Inputs

The external inputs to each counter/timer can be shared among each of the four counter/timers within a module. The external inputs can be used as count commands and timer commands. They can trigger the current counter value to be captured and then be used to generate interrupt requests. The polarity of the external inputs are selectable.

12.5.3 OFLAG Output Signal

The primary output of each counter/timer 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.

12.5.4 Master Signal

Any counter/timer can be assigned as a master signal. A master's compare signal can be broadcast to the other counters/timers within a 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.

12.6 Counting Mode Definitions

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

If a counter is programmed to count to a specific value and then stop, the count mode in the CTRL register is cleared when the count terminates.

12.6.1 Stop Mode

If the Count mode field is set to 000, the counter is inert. No counting will occur.

12.6.2 Count Mode

If the Count mode field is set to 001, the counter will count the rising edges of the selected clock source. This mode is useful for counting/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 IPS bit, then the negative edge of the selected signal is counted.

12.6.3 Edge Count Mode

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

12.6.4 Gated Count Mode

If the Count mode field is set to 011, the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the IPS bit, then the counter will count while the selected secondary input is low.

12.6.5 Quad Count Mode

If the Count mode field is set to 100, the counter will decode the primary and secondary external inputs as quad encoded signals. Quad signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quad signals are square waves 90 degrees out of phase. The decoding of quad signal provides both count and direction information.

A timing diagram illustrating the basic operation of a quad incremental position encoder is shown below:

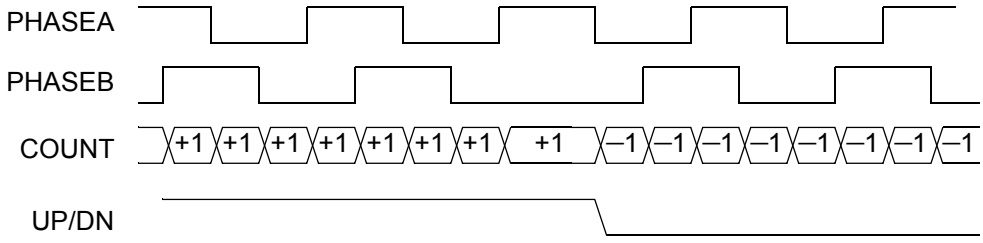


Figure 12-3. Timing Diagram

12.6.6 Signed Count Mode

If the Count mode field is set to 101, the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down). If the secondary source (direction select) input is high, the counter will count in a negative direction. If the selection direction signal is low, the counter will count in a positive direction.

12.6.7 Triggered Count Mode

If the Count mode field is set to 110, the counter will begin counting the rising edges of the primary clock source after a positive transition of the secondary input occurs; negative transition if IPS = 1. The counting will continue until a compare event occurs or another input transition is detected. Odd edges restart the counting. Even edges stop the counting until a compare event occurs.

12.6.8 One-Shot Mode

This is a sub mode of triggered event count mode; the count mode field is set to 110 while:

- Count length (LENGTH) is set
- The OFLAG output mode is set to 101
- ONCE bit of the Control Register (CTRL) is set to 1

Then 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 is passed since the Pulse Width Modulated (PWM) module asserted the synchronized signal.

12.6.9 Cascade Count Mode

If the Count mode field is set to 111, the counter is connected to the output of another counter and selected by the Primary Count Source. The counter will count up and down as compare events occur in the selected source counter. This cascade or daisy-chained mode enables multiple counters to be cascaded to yield longer counter lengths. The Cascade Count mode uses a special high speed signal path without regarding the state of the OFLAG signal. If the selected source counter experiences a CMP1 compare event while counting in a positive direction, the counter will increment. If the selected source counter experiences a CMP2 compare event while counting in a negative direction, the counter will decrement.

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

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

12.6.10 Pulse Output Mode

The Pulse Output mode is a sub mode of Count mode. If the Count mode field is set to 001while:

- The OFLAG Output mode is set to 111, gated clock output and
- The count ONCE bit is set

the counter will output a pulse stream of pulses with the same frequency of the selected clock source. The number of output pulses is equal to the compare value minus the initialization value. This mode is useful for driving step motor systems.

12.6.11 Fixed-Frequency PWM Mode

The Fixed-Frequency is a sub mode of Count mode. If the Count mode field is set to 001 while:

- Count through roll-over (LENGTH = 0)
- Continuous count (ONCE = 0)
- OFLAG Output mode is 110 (set on compare, cleared on initialization)

the counter output then yields a PWM signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers or low cost DAC

12.6.12 Variable Frequency PWM Mode

This mode is a sub mode of Count mode. If the Count mode field is set to 001 while:

- Count till compare (LENGTH = 1)
- Continuous count (ONCE = 0)
- OFLAG output mode is 100 (toggle OFLAG and alternate compare registers)

The counter output then yields a PWM signal. Its frequency and pulse width are determined by the values programmed into the CMP1 and CMP2 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. In the 56F827, The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is still underway.

12.6.13 Compare Registers Use

Dual Compare (CMP1 and CMP2) registers provide a bidirectional modulo count capability. The CMP1 register is used when the counter is *counting up*, and the CMP2 register is used when the counter is *counting down*. Alternating the compare mode is the only exception.

The CMP1 register should be set to the desired maximum count value or $FFFF to indicate the maximum unsigned value prior to roll-over, and the CMP2 register should be set to the minimum count value or $0000 to indicate the minimum unsigned value prior to roll-under.

If the Output mode is set to 100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the CMP2 value defines the desired pulse width of the on-time, and the CMP1 register defines the off-time. The variable frequency PWM mode is defined for positive counting only.

Use caution when changing CMP1 and CMP2 while the counter is active. If the counter has already passed the new value, it will count to $FFFF or $0000, roll over, then begin counting toward the new value. The check is: Count = CMPx, *not*
Count > CMP1or Count < CMP2.

With the 56F827, use of the CMPLD1 and CMPLD2 registers to compare values will help to minimize this problem.

12.6.14 Capture Register Use

The Capture register stores a copy of the counter's value when an input edge, positive or negative, is detected if IPS = 1. Once a capture event occurs, no further updating of the Capture register will occur until the Input Edge Flag (IEF) is cleared by writing 0 to the IEF.

12.7 Register Definitions

Table 12-1. TMR Memory Map

Device	Peripheral	Address
826	TMRA_BASE	$10A0
827	TMRA_BASE	$1200

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 will be TMRA_BASE.

Make certain to check which Quad Timer is available on the chip being used. Both chips have dedicated timers. Both packages have a dedicated Quad Timer A.

Table 12-2. TMR Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + $0	CMP1	Compare Register 1	Read/Write	Section 12.7.3
Base + $1	CMP2	Compare Register 2	Read/Write	Section 12.7.4
Base + $2	CAP	Capture Register	Read/Write	Section 12.7.5
Base + $3	LOAD	Load Register	Read/Write	Section 12.7.6
Base + $4	HOLD	Hold Register	Read/Write	Section 12.7.7
Base + $5	CNTR	Counter Register	Read/Write	Section 12.7.8
Base + $6	CTRL	Control Register	Read/Write	Section 12.7.1

Table 12-2. TMR Register Summary (Continued)

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + $7	SCR	Status/Control Register	Read/Write	Section 12.7.2
Base + $8	CMPLD1	Comparator Load Register 1 (827)	Read/Write	Section 12.7.9
Base + $9	CMPLD2	Comparator Load Register 2 (827)	Read/Write	Section 12.7.10
Base + $A	COMSCR	Comparator Status/Control Reg (827)	Read/Write	Section 12.7.11

Each of the TMR 11 registers is summarized in **Figure 12-1.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CMP1	R	COMPARISON VALUE															
		W																
$1	CMP2	R	COMPARISON VALUE															
		W																
$2	CAP	R	CAPTURE VALUE															
		W																
$3	LOAD	R	LOAD VALUE															
		W																
$4	HOLD	R	HOLD VALUE															
		W																
$5	CNTR	R	COUNTER															
		W																
$6	CTRL	R	COUNT MODE			PRIMARY COUNT SOURCE				SECONDARY SOURCE		ONCE	LENGTH	DIR	CoINIT	OUTPUT MODE		
		W																
$7	SCR	R	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE MODE		MSTR	EEOF	VAL	FORCE	OPS	OEN
		W																
$8	CMPLD1	R	COMPARATOR LOAD 1															
		W																
$9	CMPLD2	R	COMPARATOR LOAD 2															
		W																
$A	COMSCR	R	0	0	0	0	0	0	0	0	TCF2EN	TC1EN	TCF2	TCF1	CL2		CL1	
		W																

R	0	Read a 0
W		Reserved

Figure 12-1. TMR Register Map Summary

12.7.1 TMR Control Registers (CTRL)

TMRA0_CTRL (Timer A, Channel 0 Control)—Address: TMRA_BASE + $6
TMRA1_CTRL (Timer A, Channel 1 Control)—Address: TMRA_BASE + $E – 56F826 $16 – 56F827
TMRA2_CTRL (Timer A, Channel 2 Control)—Address: TMRA_BASE + $16 – 56F826 $26 – 56F827
TMRA3_CTRL Timer A, Channel 3 Control)—Address: TMRA_BASE + $1E – 56F826 $36 – 56F827

BASE+$6,E,16, 1E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNT MODE			PRIMARY COUNT SOURCE				SECONDARY SOURCE		ONCE	LENGTH	DIR	Co INIT	OOUTPUT MODE		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-2. TMR Control Registers (CTRL)

See Programmer's Sheet on Appendix page B - 85

12.7.1.1 Count Mode—Bits 15–13

These bits control the basic counting and behavior of the counter.

- 000 = No operation
- 001 = Count rising edges of primary source[1]
- 010 = Count rising and falling edges of primary source
- 011 = Count rising edges of primary source while secondary input high active
- 100 = Quad Count mode, uses primary and secondary sources
- 101 = Count primary source rising edges, secondary source specifies direction (1 = minus)[2]
- 110 = Edge of secondary source triggers primary count till compare
- 111 = Cascaded Counter mode, up/down[3]

12.7.1.2 Primary Count Source—Bits 12–9

These bits select the Primary Count Source.

- 0000 = Counter 0 pin
- 0001 = Counter 1 pin
- 0010 = Counter 2 pin
- 0011 = Counter 3 pin
- 0100 = Counter 0 OFLAG

1.Rising Edges counted only when IPS = 0. Falling edges counted when IPS = 1.
2.Rising Edges counted only when IPS = 0. Falling edges counted when IPS = 1.
3.Primary Count Source must be set to one of the counter outputs.

- 0101 = Counter 1 OFLAG
- 0110 = Counter 2 OFLAG
- 0111 = Counter 3 OFLAG
- 1000 = Prescaler (IPBus clock divide by 1)
- 1001 = Prescaler (IPBus clock divide by 2)
- 1010 = Prescaler (IPBus clock divide by 4)
- 1011 = Prescaler (IPBus clock divide by 8)
- 1100 = Prescaler (IPBus clock divide by 16)
- 1101 = Prescaler (IPBus clock divide by 32)
- 1110 = Prescaler (IPBus clock divide by 64)
- 1111 = Prescaler (IPBus clock divide by 128)

Note: A timer selecting its own output for input is not a legal choice. The result is no counting.

12.7.1.3 Secondary Count Source—Bits 8–7

These bits identify the external input pin to be used as a count command, or timer command. The selected input can trigger the timer to capture the current value of 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 SCR register.

- 00 = Counter 0 pin
- 01 = Counter 1 pin
- 10 = Counter 2 pin
- 11 = Counter 3 pin

12.7.1.4 Count Once (ONCE)—Bit 6

This bit selects continuous or One-Shot Counting mode.

- 0 = Count repeatedly
- 1 = Count until compare and then stop. If *counting up*, successful compare occurs when the counter reaches a CMP1 value. If *counting down*, successful compare occurs when the counter reaches a CMP2 value. When the compare occurs the timer is stopped by changing the timer's Count mode to *Stop Mode* (CM=0).

12.7.1.5 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, the binary roll over.

- 0 = Roll over
- 1 = Count until compare, then reinitialize. If counting up, successful compare occurs when counter reaches CMP1 value. If counting down, successful compare occurs when counter reaches CMP2 value.[1]

12.7.1.6 Count Direction (DIR)—Bit 4

This bit selects either the normal count direction *up*, or the reverse direction, *down*.

- 0 = Count-up
- 1 = Count-down

12.7.1.7 Co-Channel Initialization (Co Init)—Bit 3

This bit enables another counter/timer within the module to force the reinitialization of this counter/timer when it has an active compare event.

- 0 = Co-Channel counter/timers can not force a reinitialization of this counter/timer
- 1 = Co-Channel counter/timers may force a reinitialization of this counter/timer

12.7.1.8 Output Mode—Bits 2-0

These bits determine the mode of operation for the OFLAG output signal.

- 000 = Asserted while counter is active
- 001 = Clear OFLAG output on successful compare
- 010 = Set OFLAG output on successful compare
- 011 = Toggle OFLAG output on successful compare
- 100 = Toggle OFLAG output using alternating compare registers
- 101 = Set on compare, cleared on secondary source input edge
- 110 = Set on compare, cleared on counter rollover
- 111 = Enable gated clock output while counter is active

1. When Output mode $4 is used, alternating values of CMP1 and CMP2 are used to generate successful comparisons. For example, when Output mode is $4, the counter counts until CMP1 value is reached, reinitializes, then counts until CMP2 value is reached, reinitializes, then counts until CMP1 value is reached, etc.

12.7.2 TMR Status and Control Registers (SCR)

TMRA0_SCR (Timer A, Channel 0 Status and Control)—Address: TMRA_BASE + $7
TMRA1_SCR (Timer A, Channel 1 Status and Control)—Address: TMRA_BASE + $F – 56F826 $17– 56F827
TMRA2_SCR (Timer A, Channel 2 Status and Control)—Address: TMRA_BASE + $17 – 56F826 $27– 56F827
TMRA3_SCR (Timer A, Channel 3 Status and Control)—Address: TMRA_BASE + $1F – 56F826 $37– 56F827

BASE+$7,F,17,1F	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 12-3. TMR Status and Control Registers (SCR)

See Programmer's Sheet on Appendix page B - 88

12.7.2.1 Timer Compare Flag (TCF)—Bit 15

This bit is set when a successful compare occurs. In the 56F826, it is cleared by writing 0 to the bit location. In the 56F827, it is cleared by writing 1 to the bit location. The timer group will not assert another TCF interrupt until this bit has been cleared.

12.7.2.2 Timer Compare Flag Interrupt Enable (TCFIE)—Bit 14

When set, this bit activates interrupts if the Timer Compare Flag (TCF) bit is also set.

12.7.2.3 Timer Overflow Flag (TOF)—Bit 13

This bit is set when the counter rolls over its maximum value $FFFF or $0000, depending on count direction. In the 56F826, it is cleared by writing 0 to the bit location. In the 56F827, it is cleared by writing 1 to the bit location. The timer group will not assert another TOF interrupt until this bit has been cleared.

12.7.2.4 Timer Overflow Flag Interrupt Enable (TOFIE)—Bit 12

When set, this bit activates interrupts if the Timer Overflow Flag (TOF) bit is also set.

12.7.2.5 Input Edge Flag (IEF)—Bit 11

This bit is set when a positive input transition occurs on the package pin selected by the Secondary Count Source (SCS) field of the Control register. In the 56F826, it is cleared by writing 0 to the bit location. In the 56F827, it is cleared by writing 1 to the bit location.

Note: The two Capture Mode (CM) bits must be set to a non-zero value before this flag can be set. The CM field of the SCR must be set to 110 (binary) in order to have the timer module set the IEF flag.

Note: Setting the IPS bit enables the detection of negative input edge transition detection. Also, the control register's Secondary Count Source determines which external input pin is monitored by the detection circuitry.

12.7.2.6 Input Edge Flag Interrupt Enable (IEFIE)—Bit 10

When set, this bit enables interrupts if the Input Edge Flag (IEF) bit is also set. The timer group will not assert another IEF interrupt until this bit is cleared.

12.7.2.7 Input Polarity Select (IPS)—Bit 9

This bit is an exclusive OR with the state of package pin inputs selected by the PCS and SCS fields of the CTL register. When set, this bit inverts the polarity of both the primary and secondary inputs. It does not invert the polarity of an input signal when the signal is being driven by another timer group.

12.7.2.8 External Input Signal (INPUT)—Bit 8

This bit reflects the current state of the external input pin after application of the IPS bit. This is a *read-only* bit.

Note: The package pin state reflected by this bit is synchronized to the IPBus clock. For example, the INPUT bit shows the sampled state of the input pin two IPBus clocks previously.

12.7.2.9 Input Capture Mode (CAPTURE MODE)—Bits 7–6

These bits specify the operation of the Capture register as well as the operation of the input edge flag. The input Capture mode field must be 01, 10 or 11 in order for input edge flag interrupts to be asserted.

Table 12-3. Capture Register Operation

Capture	Mode	IPS Action
00	x	Capture Disabled
01	0	Load on Rising Edge
01	1	Load on Falling Edge
10	0	Load on Falling Edge
10	1	Load on Rising Edge
11	x	Load on Both Edges

12.7.2.10 Master Mode (MSTR)—Bit 5

When set, this bit enables the compare function's output to be broadcast to the other counters/timers in the module. This signal then can be used to reinitialize the other counters and/or force their OFLAG signal outputs.

12.7.2.11 Enable External OFLAG Force (EEOF)—Bit 4

When set, this bit enables the compare from another counter/timer enabled as a master to force the state of this counters OFLAG output signal.

12.7.2.12 Forced OFLAG Value (VAL)—Bit 3

This bit determines the value of the OFLAG output signal when an software triggered FORCE command, or another counter/timer (set as a master) issues a FORCE command.

12.7.2.13 Force the 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 always reads as 0. 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.

12.7.2.14 Output Polarity Select (OPS)—Bit 1

When set, this bit inverts the polarity of a signal driven by a timer group on to an external package pin. Other timer groups reading this pin will read the inverted signal.

- 0 = True polarity
- 1 = Inverted polarity

12.7.2.15 Output Enable (OEN)—Bit 0

When set, this bit enables the OFLAG output signal to be put on the external pin. Other timer groups using this external pin as their input will see the driven value. The polarity of the signal will be determined by the OPS bit.

12.7.3 TMR Compare Register 1 (CMP1)

TMRA0_CMP1 (Timer A, Channel 0 Compare #1)—Address: TMRA_BASE + $0
TMRA1_CMP1 (Timer A, Channel 1 Compare #1)—Address: TMRA_BASE + $8 – 56F826 $10 – 56F827
TMRA2_CMP1 (Timer A, Channel 2 Compare #1)—Address: TMRA_BASE + $10 – 56F826 $20 – 56F827
TMRA3_CMP1 (Timer A, Channel 3 Compare #1)—Address: TMRA_BASE + $18 – 56F826 $30 – 56F827

BASE+$0,8,10,18	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-4. TMR Compare Register 1 (CMP1)

See Programmer's Sheet on Appendix page B - 90

This read/write register stores the value used for comparison with the counter value.

12.7.4 TMR Compare Register 2 (CMP2)

TMRA0_CMP2 (Timer A, Channel 0 Compare #2)—Address: TMRA_BASE + $1
TMRA1_CMP2 (Timer A, Channel 1 Compare #2)—Address: TMRA_BASE + $9 – 56F826 $11 – 56F827
TMRA2_CMP2 (Timer A, Channel 2 Compare #2)—Address: TMRA_BASE + $11 – 56F826 $21 – 56F827
TMRA3_CMP2 (Timer A, Channel 3 Compare #2)—Address: TMRA_BASE + $19 – 56F826 $31 – 56F827

BASE+$1,9,11,19	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-5. TMR Compare Register 2 (CMP2)

See Programmer's Sheet on Appendix page B - 97

This read/write register stores the value used for comparison with the counter value.

12.7.5 TMR Capture Register (CAP)

TMRA0_CAP (Timer A, Channel 0 Capture)—Address: TMRA_BASE + $2
TMRA1_CAP (Timer A, Channel 1 Capture)—Address: TMRA_BASE + $A – 56F826 $12 – 56F827
TMRA2_CAP (Timer A, Channel 2 Capture)—Address: TMRA_BASE + $12 – 56F826 $22 – 56F827
TMRA3_CAP (Timer A, Channel 3 Capture)—Address: TMRA_BASE + $1A – 56F826 $32 – 56F827

BASE+$2,A,12,1A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTURE VALUE [15:0]															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-6. TMR Capture Register (CAP)

See Programmer's Sheet on Appendix page B - 92

This read/write register stores the value captured from the counter.

12.7.6 TMR Load Register (LOAD)

TMRA0_LOAD (Timer A, Channel 0 Load)—Address: TMRA_BASE + $3
TMRA1_LOAD (Timer A, Channel 1 Load)—Address: TMRA_BASE + $B – 56F826 $13 – 56F827
TMRA2_LOAD (Timer A, Channel 2 Load)—Address: TMRA_BASE + $13 – 56F826 $23 – 56F827
TMRA3_LOAD (Timer A, Channel 3 Load)—Address: TMRA_BASE + $1B – 56F826 $33 – 56F827

BASE+$3,B,13,1B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOAD VALUE [15:0]															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-7. TMR Load Register (LOAD)

See Programmer's Sheet on Appendix page B - 93

This read/write register stores the value used to initialize the counter.

12.7.7 TMR Hold Register (HOLD)

TMRA0_HOLD (Timer A, Channel 0 Hold)—Address: TMRA_BASE + $4
TMRA1_HOLD (Timer A, Channel 1 Hold)—Address: TMRA_BASE + $C – 56F826 $14 – 56F827
TMRA2_HOLD (Timer A, Channel 2 Hold)—Address: TMRA_BASE + $14 – 56F826 $24 – 56F827
TMRA3_HOLD (Timer A, Channel 3 Hold)—Address: TMRA_BASE + $1C – 56F826 $34 – 56F827

BASE+$4,C,14,1C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HOLD VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-8. TMR Hold Register (HOLD)

See Programmer's Sheet on Appendix page B - 94

This read/write register stores the counter's values of specific channels whenever any of the four counters within a module is read.

12.7.8 TMR Counter Register (CNTR)

TMRA0_CNTR (Timer A, Channel 0 Cntr)—Address: TMRA_BASE + $5
TMRA1_CNTR (Timer A, Channel 1 Cntr)—Address: TMRA_BASE + $D – 56F826 $15 – 56F827
TMRA2_CNTR (Timer A, Channel 2 Cntr)—Address: TMRA_BASE + $15– 56F826 $25 – 56F827
TMRA3_CNTR (Timer A, Channel 3 Cntr)—Address: TMRA_BASE + $1D – 56F826 $35 – 56F827

BASE+$5,D,15,1D	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 12-9. TMR Counter (CNTR)

See Programmer's Sheet on Appendix page B - 95

This read/write register is the counter for the corresponding channel in a timer module.

12.7.9 TMR Comparator Load Register 1 (CMPLD1)–56F827 Only

TMRA0_CMPLD1 (Timer A, Channel 0 CMPLD1)—Address: TMRA_BASE + $8 – 56F827 Only
TMRA1_CMPLD1 (Timer A, Channel 1 CMPLD1)—Address: TMRA_BASE + $18 – 56F827 Only
TMRA2_CMPLD1 (Timer A, Channel 2 CMPLD1)—Address: TMRA_BASE + $28 – 56F827 Only
TMRA3_CMPLD1 (Timer A, Channel 3 CMPLD1)—Address: TMRA_BASE + $38 – 56F827 Only

BASE+$8,18,28,38	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

Figure 12-10. TMR Comparator Load 1 (CMPLD1)

See Programmer's Sheet on Appendix page B - 96

This read/write register is the comparator one preload value for the corresponding channel in a timer module for 56F827 *only*.

12.7.10 TMR Comparator Load Register 2 (CMPLD2)–56F827 Only

TMRA0_CMPLD2 (Timer A, Channel 0 CMPLD2)—Address: TMRA_BASE + $9 – 56F827 Only
TMRA1_CMPLD2 (Timer A, Channel 1 CMPLD2)—Address: TMRA_BASE + $19 – 56F827 Only
TMRA2_CMPLD2 (Timer A, Channel 2 CMPLD2)—Address: TMRA_BASE + $29 – 56F827 Only
TMRA3_CMPLD2 (Timer A, Channel 3 CMPLD2)—Address: TMRA_BASE + $39 – 56F827 Only

BASE+$9,19,29,39	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 12-11. TMR Comparator Load 2 (CMPLD2)

See Programmer's Sheet on Appendix page B - 97

This read/write register is the comparator two preload value for the corresponding channel in a timer module for 56F827 *only*.

12.7.11 TMR Comparator Status and Control Register (COMSCR)– 56F827 Only

TMRA0_COMSCR (Timer A, Channel 0 COMSCR)—Address: TMRA_BASE + $A – 56F827 Only
TMRA1_COMSCR (Timer A, Channel 1 COMSCR)—Address: TMRA_BASE + $1A – 56F827 Only
TMRA2_COMSCR (Timer A, Channel 2 COMSCR)—Address: TMRA_BASE + $2A – 56F827 Only
TMRA3_COMSCR (Timer A, Channel 3 COMSCR)—Address: TMRA_BASE + $3A – 56F827 Only

BASE+$A,1A,2A,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	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 12-12. TMR Comparator Status and Control Register (COMSCR)

See Programmer's Sheet on Appendix page B - 98

12.7.11.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

12.7.11.2 Timer Compare 2 Interrupt Enable (TCF2EN)—Bit 7

An interrupt is issued when both this bit and the TCF2 bit are set.

12.7.11.3 Timer Compare One Interrupt Enable (TCF1EN)—Bit 6

An interrupt is issued when both this bit and the TCF1 bit are set.

12.7.11.4 Timer Compare 2 Interrupt Source (TCF2)—Bit 5

When set, this bit indicates a successful comparison of the Timer and CMP2 register has occurred. This bit is sticky. It will remain set until explicitly cleared by writing 1 to this bit location. Writing 0 to this location will have no effect.

12.7.11.5 Timer Compare One Interrupt Source (TCF1)—Bit 4

When set, this bit indicates a successful comparison of the timer and the CMP1 register has occurred. This bit is sticky. It will remain set until explicitly cleared by writing 1 to this bit location. Writing 0 to this location will have no effect.

12.7.11.6 Compare Load Control 2 (CL2)—Bit 3–2

These bits control when CMP2 is preloaded with the value from CMPLD2.

- 00 - Never preload
- 01 - Load upon successful compare with the value in CMP1
- 10 - Load upon successful compare with the value in CMP2
- 11 - Reserved

12.7.11.7 Compare Load Control 1 (CL1)—Bit 1–0

These bits control when CMP1 is preloaded with the value from CMPLD1.

- 00 - Never preload
- 01 - Load upon successful compare with the value in CMP1
- 10 - Load upon successful compare with the value in CMP2
- 11 - Reserved

12.8 Timer Group A Functionality

Some of the input and output pins of the timers are shared with other peripheral modules on the 56F826/827.

Note: Individual timers often use their own I/O pin. However, timers may use any other available pin within their group as an input. The timers are limited to their own I/O pin for use as an output pin.

12.8.1 Timer Group A

Timer Group A shares pins with Quad Decoder module 0. The decoder has primary ownership of the pins for inputs. The Decoder Control register for Quad Decoder 0 controls (DEC0_DECCR) a switch matrix connecting Timer A inputs to the I/O pins, provides filtered input data, or a remapped data format. Timer group A outputs are directly connected to the I/O pins. If a timer's output is enabled, its output will drive the I/O pin. The decoder module does not use the I/O pins for outputs.

Note: The Quad Decoder 0 module contains a quad test signal generator capable of being monitored by the Timer module on the Timer 0 (PHASEA0) input and the Timer 1 (PHASEB0) input.

- Timer A, Counter 0 I/O pin is shared with Quad Decoder 0 PHASEA0 pin
- Timer A, Counter 1 I/O pin is shared with Quad Decoder 1 PHASEB0 pin
- Timer A, Counter 2 I/O pin is shared with Quad Decoder 0 INDEX0 pin
- Timer A, Counter 3 I/O pin is shared with Quad Decoder 0 HOME0 pin
- Timer A, Counter 0 output is connected to the ADC sync input

12.8.1.1 General Input Behavior

Time inputs are sampled at the peripheral clock rate. There is a delay of one peripheral clock period before the input can affect the behavior of a counter. This is typical of synchronous counters systems. A gated counter is the only exception.

Chapter 13
Time-of-Day (TOD)

13.1 Introduction

Both 56F826 and 56F827 have a Time-of-Day (TOD) feature implemented as a sequence of counters to track elapsed time. The hardware is capable of tracking time up to 179.5 years, or 65,535 days. The starting day is determined by the application software.

TOD is comprised of a series of counters tracking elapsed seconds, minutes, hours, and days. The module expects an input clock ranging from 0-65536Hz. The clock is further scaled down to generate a 1Hz clock driving all of the time counters.

All of the time counters can be loaded with the time-of-day prior to enabling TOD. Time counters subsequently track the elapsed time. If required, the TOD feature can issue an alarm whenever the current time matches the time programmed into the TOD alarm registers. Design is based on the assumption the TOD clock is always slower than the IPBus clock. The TOD can also provide an interrupt every second.

TOD assumes the crystal is operating at a frequency of 2-4MHz, divided by 128 before arriving as the clock input to the TOD module. This procedure is discussed in the TOD module section.

13.2 Features

- Separate counters for seconds, minutes, hours, and days with a 16-bit day capacity
- Separate read/write registers for seconds, minutes, hours, and days
- Alarm clock registers for seconds, minutes, hours, and days
- Alarm interrupt with independent enable for comparison of seconds, minutes, hours and days
- One-second interrupt with independent enable
- TOD reset only at power-on, unaffected by reset pin, software reset, or COP reset
- Flexible clock prescaling for use with either external clocking or crystal oscillator
- Works with crystal frequency of 2 – 4MHz
- Capable to generate interrupt, pulling the part out of Sleep mode
- Capable to track time up to 179.5 years
- Can be configured to generate an alarm at a designated time

13.3 Counter Operation Block Diagram

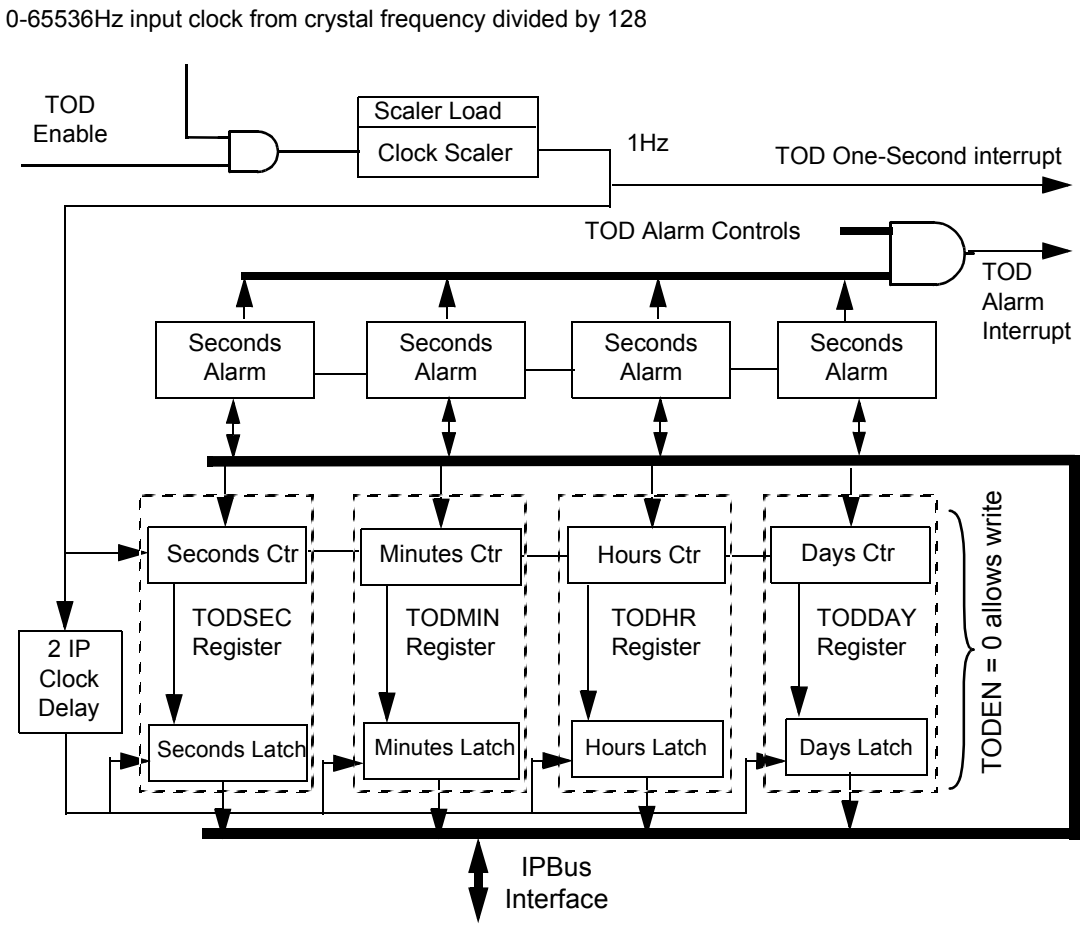


Figure 13-1. Time-of-Day Counter Operation

13.4 Functional Description

A sub block of TOD implements the scaler, seconds, minutes, hours and days counters. The counters can be initialized with the current time when the TOD module is disabled, or when TODEN = 0.

13.4.1 Scaler

Depending upon the frequency of the clock fed to the TOD module, the appropriate value should be written to the clock scaler load register to derive a 1Hz clock. The scaler counter is a 16-bit up counter, counting from zero to the value specified in the clock scaler load register. In normal operation, the scaler counter increments with TOD input clock (frequency range from 0~65536 Hz) if TODEN = 1. This clock pulse will be used to clock the next counter, SECONDS counter, in the chain.

13.4.2 Time Registers

The time registers (seconds, minutes, hours, days) are each implemented as a counter/latch pairs. All writes to the register target the counters. All read obtain the latch values.

13.4.2.1 Time Counters

The time counters are a set of modulo counters clocked by the 1Hz clock to maintain the current time in seconds, minutes, hours and days. They count only when TODEN = 1. These counters maintain the time to the accuracy of the 1Hz clock. They initialize to the current value in the time registers at the first 1Hz clock (one second) after TODEN is set to 1 and advances each second thereafter.

13.4.2.2 Time Latches

The time latches are introduced into the design to minimize the possibility of reading changing time values as the time counters update. The latches capture the counter values two IP_CLK after the 1Hz clock which gives the time counters a chance to settle, avoiding an *in flux* read of the time value.

13.5 Operating Modes

The basic process of initializing and counting time is presented in this section.

To set the time of day:

1. Set TODEN to zero
2. Write the current time into the seconds, minutes, hours, and days counters
3. Set TODEN to one

This will initialize the corresponding seconds, minutes, hours, and days counters to the desired values s and begin the counting process. The counting process is intuitive. The seconds counter advances with each time-base clock. The minutes advance each 60 seconds. The hours advance each 60 minutes. The days advance each 24 hours, wrapping to zero on overflow.

While counting is in process, the seconds, minutes, hours, and days latches are updated after two IPBus Clocks have the value of the corresponding counter. They may be read any time. When the counting is disabled the latches will retain the values the counters held when counting was disabled and the advancement of time is frozen.

13.5.1 Stop Mode

During Stop mode the TOD module will continue to keep track of the current time. The TOD alarms will still activate and can wake the processor up for TOD related processing.

13.5.2 TOD Alarms

The TOD alarms sub block is used to generate TOD alarm IRQ signals. The TOD alarm registers TODSAL, TODMAL, and TODDAL, along with TOD control/status register(TODCS, are implemented in this sub block.

TOD generates two interrupt signals:

1. TOD Alarm Interrupt
2. TOD Second Interrupt

Both interrupt signals can wake up the device from its stop mode. Hence the interrupt assertion path is an asynchronous path and is not clocked by the IPBus clock.

13.5.3 Alarm Interrupt Flag and Outputs

The alarm interrupt feature provides a flexible means of detecting events based on time of day. The TOD module contains alarm registers for seconds, minutes, hours, and days. The control and status register contains individual enables for each of these registers (TODSA, TODMA, TODHA, TODDA) as well as an overall Alarm Interrupt Enable (TODAEN). An alarm interrupt triggers during the time-of-day counting process. This interrupt occurs when at least one of the alarm enables is set and the TODAEN is set.

All alarm registers can be enabled to generate an alarm at a designated time. All enabled alarm registers must match the time latches before the TOD alarm interrupt IRQ is generated. For example, if the minutes and hour alarms are enabled, then both the minutes and hours latches must match the respective alarm registers before the IRQ will occur.

The alarm interrupt triggers one TOD input clock period after the TOD_CLK edge, advancing the time-of-day counters to their activation value. If latency minimization is desired, it is preferable to generate a faster TOD input clock and use a correspondingly higher value in the TOD module’s clock scaler register. The alarm interrupt will not trigger when TODEN is zero until TODEN is set to one and the time-of-day counters are initialized to their new values. This interrupt will function in the Stop mode; however, the TOD module must be configured and enabled prior to entering the Stop mode.

Every time the alarm interrupt triggers the Alarm Interrupt Occurred Flag (TODAL) in the control status register is set.

For the 56F826, The TODAL bit is cleared by writing 0 to it while it contains a one.

For the 56F827, the TODAL bit is cleared by writing 1 to it.

The interrupt will not be taken unless it is also enabled in the interrupt controller module.

13.5.4 One-Second Interrupt Flag and Outputs

The one-second interrupt feature provides a means of detecting the passage of one-second time intervals. The one-second interrupt is triggered and the Seconds Interrupt occurred Flag (TODSID) in the status control register is set on the rising edge of the 1Hz clock. The 1Hz clock is derived out of the prescaler. This clock only operates while the TOD Enable (TODEN) is set. The scaler remains in its prior state while the TOD module is disabled, essentially freezing the passage of time.

For the 56F826, The TODSID bit is cleared by writing a zero to it while it contains a one.

For the 56F827, the TODSID bit is cleared by writing a one to it.

This interrupt will function in the Stop mode; however, the TOD module must be configured and enabled prior to entering the Stop mode.

The one-second interrupt output is used to signal a one-second interrupt to the interrupt controller. The interrupt will not be taken unless it is also enabled in the interrupt controller module

13.6 Register Map

Table 13-1. TOD Registers

Register Address	Register Name	Register Function	Decimal Range	When Write Enabled	POR Value
Base + 0	TODCS	Control/Status Register	N/C	Always	0
Base + 1	TODCSL	Clock Scaler Load Register	0-65535	TODEN=0	0
Base + 2	TODSEC	Seconds Counter Register	0-59	TODEN=0	0
Base + 3	TODSAL	Seconds Alarm Register	0-59	Always	0
Base + 4	TODMIN	Minutes Counter Register	0-59	TODEN=0	0
Base + 5	TODMAL	Minutes Alarm Register	0-59	Always	0
Base + 6	TODHR	Hours Counter Register	0-23	TODEN=0	0
Base + 7	TODHAL	Hours Alarm Register	0-23	Always	0
Base + 8	TODDAY	Days Counter	0-65535	TODEN=0	0
Base + 9	TODDAL	Days Alarm	0-65535	Always	0

13.7 Register Definitions

Table 13-2. TOD Memory Map

Device	Peripheral	Address
826/827	TOD_BASE	$10C0

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 will be TOD_BASE.

Table 13-3. TOD Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + $0	TODCS	Control Status Register	Read/Write	Section 13.7.1
Base + $1	TODCSL	Clock Scaler Register	Read/Write	Section 13.7.2
Base + $2	TODSEC	Seconds Counter Register	Read/Write	Section 13.7.3
Base + $3	TODSAL	Seconds Alarm Register	Read/Write	Section 13.7.4
Base + $4	TODMIN	Minutes Counter Register	Read/Write	Section 13.7.5
Base + $5	TODMAL	Minutes Alarm Register	Read/Write	Section 13.7.6
Base + $6	TODHR	Hours Counter Register	Read/Write	Section 13.7.7
Base + $7	TODHAL	Hours Alarm Register	Read/Write	Section 13.7.8
Base + $8	TODDAY	Days Counter Register	Read/Write	Section 13.7.9
Base + $9	TODDAL	Days Alarm Register	Read/Write	Section 13.7.10

Bits of the 10 registers are summarized in **Figure 13-2.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	TODCS	R	TODISIO	TODAL	0	0	0	0	TEST		TODDA	TODHA	TODMA	TODSA	TODSEN	TODAEN	0	TODEN
		W																
$1	TODCSL	R	TIME-OF-DAY CLOCK SCALER															
		W																
$2	TODSEC	R	0	0	0	0	0	0	0	0	0	0	TODSEC					
		W																
$3	TODSAL	R	0	0	0	0	0	0	0	0	0	0	TODSAL					
		W																
$4	TODMIN	R	0	0	0	0	0	0	0	0	0	0	TODMIN					
		W																
$5	TODMAL	R	0	0	0	0	0	0	0	0	0	0	TODMAL					
		W																
$6	TODHR	R	0	0	0	0	0	0	0	0	0	0	TODHR					
		W																
$7	TODHAL	R	0	0	0	0	0	0	0	0	0	0	0	TODHAL				
		W																
$8	TODDAY	R	TODDAY															
		W																
$9	TODDAL	R	TODDAL															
		W																

R 0 Read as 0
W Reserved

Figure 13-2. TOD Register Map Summary

13.7.1 TOD Control Status (TODCS)

The TOD Control Status register controls TOD operation.

BASE +:$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TODSIO	TODAL	0	0	0	0	TEST		TODDA	TODHA	TODMA	TODSA	TODSEN	TODAEN	0	TODEN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-3. TOD Control Status (TODCS)

See Programmer's Sheet on Appendix page B-99

13.7.1.1 TOD One-Second Interrupt Occurred Flag (TODSIO)—Bit 15

This bit is set when one-second interrupt occurs.

For the 56F826, this bit is cleared by writing 0 to the bit position.

For the 56F827, the TODSIO bit is cleared by writing 0 to it.

This bit must be cleared before exiting the Interrupt Service Routine (ISR).

13.7.1.2 TOD Alarm Interrupt Occurred Flag (TODAL)—Bit 14

This bit is set when TOD Alarm Interrupt occurs.

For the 56F826, this bit is cleared by writing 0 to the bit position.

For the 56F827, the TODSIO bit is cleared by writing 0 to it.

This bit must be cleared before exiting the Interrupt Service Routine (ISR).

13.7.1.3 Reserved—Bits 13–10

This bit field is reserved or not implemented. It is read/written as 0, ensuring future compatibility.

13.7.1.4 TEST—Bits 8-9

These are factory test bits.

13.7.1.5 TOD Days Alarm Enable (TODDA)—Bit 7

- 0 = Days alarm interrupt is disabled
- 1 = Days alarm interrupt, requiring match of days alarm register to days counter, is enabled

13.7.1.6 TOD Hours Alarm Enable (TODHA)—Bit 6

- 0 = Hours alarm interrupt is disabled
- 1 = Hours alarm interrupt, requiring match of days alarm register to days counter, is enabled

13.7.1.7 TOD Minutes Alarm Enable (TODMA)—Bit 5

- 0 = Minutes alarm interrupt is disabled
- 1 = Minutes alarm interrupt, requiring match of days alarm register to days counter, is enabled

13.7.1.8 TOD Seconds Alarm Enable (TODSA)—Bit 4

- 0 = Seconds alarm interrupt is disabled
- 1 = Seconds alarm interrupt, requiring match of days alarm register to days counter, is enabled

13.7.1.9 TOD Seconds Interrupt Enable (TODSEN)—Bit 3

- 0 = Disables TOD seconds interrupt
- 1 = Enables TOD seconds interrupt

13.7.1.10 TOD Alarm Interrupt Enable (TODAEN)—Bit 2

- 0 = Disables TOD alarm interrupt
- 1 = Enables TOD alarm interrupt

13.7.1.11 Reserved—Bit 1

This bit is reserved or not implemented. It is read/written as 0, ensuring future compatibility.

13.7.1.12 Time-of-Day Enable (TODEN)—Bit 0

Writing zero to TODEN disables the Time-of-Day counting; all counters become read/write accessible. When this field is written with one, the seconds, minutes, hours, and days counters start counting from new values are loaded and proceed to count in a normal Time-of-Day basis with seconds overflow incrementing minutes, minutes overflow incrementing hours, and so on through days overflow wrapping back to zero.

While time-of-day counting is enabled, the seconds, minutes, hours, and day latches maintain the value of the corresponding counter so reading latches returns the current time-of-day. But writing seconds register, minutes register, hour register and days counters is prohibited.

13.7.2 TOD Clock Scaler (TODCSL)

Setting this field to X divides the TOD Input Clock frequency by X plus one to produce the time base clock, therefore incrementing the counters. The clock generation system and this scaler must be configured so the time-base clock is precisely 1Hz, allowing correct operation of the TOD module as the seconds counter increments with each time-base clock.

BASE +:$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TIME-OF-DAY CLOCK SCALER															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-4. TOD Clock Scaler (TODCSR)

See Programmer's Sheet on Appendix page B-100

13.7.3 TOD Seconds Counter (TODSEC)

When TODEN is set, this counter is continuously updated to contain the current seconds value. The value of the counter can be read through corresponding seconds latch, but it cannot be modified. When TODEN is cleared, TOD counting is disabled, and this counter can be modified. When TODEN is set to one again, the counting resumes from the new set time.

BASE +:$2	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	TODSEC					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-5. TOD Seconds Register (TODSEC)

See Programmer's Sheet on Appendix page B-101

13.7.4 TOD Seconds Alarm Register (TODSAL)

When the value contained in this register matches the value of the seconds counter, the seconds alarm is asserted if the Seconds Alarm Enable (TODSA) bit and Alarm Interrupt Enable (TODAEN) bits are set and all other enabled alarm registers also match.

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	TODSAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-6. TOD Seconds Alarm Register (TODSAL)

See Programmer's Sheet on Appendix page B-102

13.7.5 TOD Minutes Counter (TODMIN)

When TODEN is set, this counter is continuously updated to contain the current minutes value. The value of the counter can be read through the corresponding minutes latch, but it cannot be modified. When TODEN is cleared, TOD counting is disabled and this counter can be modified. When TODEN is set to one again, the counting resumes from the new set time.

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	TODMIN					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-7. TOD Minutes Register (TODMIN)

See Programmer's Sheet on Appendix page B-103

13.7.6 TOD Minutes Alarm Register (TODMAL)

When the value contained in this register matches the value of the minutes counter, the minutes alarm interrupt is asserted if the Minutes Alarm Enable (TODMA) bit and Alarm Interrupt Enable (TODAEN) bit are both set, and all other enabled alarm registers also match the current time.

BASE + $5	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	TODMAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-8. TOD Minutes Alarm Register (TODMAL)

See Programmer's Sheet on Appendix page B-104

13.7.7 TOD Hours Counter (TODHR)

When TODEN is set, this counter is continuously updated to contain the current hours value. The value of the counter can be read through the corresponding hours latch, but it can not be modified. When TODEN is cleared, TOD counting is disabled, and this counter can be modified. When TODEN is set to one again, the counting resumes with the new set time.

BASE + $6	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	TODHR					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-9. TOD Hours Register (TODHR)

See Programmer's Sheet on Appendix page B-105

13.7.8 TOD Hours Alarm Register (TODHAL)—Bits 4–0

When the value contained in this register matches the value of the hours counter, the hours alarm interrupt is asserted if the Hours Alarm Enable (TODHA) bit and the Alarm Interrupt Enable (TODAEN) bit are both set and all other enabled alarm registers also match the current time.

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	0	TODHAL				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-10. TOD Hours Alarm Register (TODHAL)

See Programmer's Sheet on Appendix page B-106

13.7.9 TOD Days Counter (TODDAY)

When TODEN is set, this counter is continuously updated to contain the current days value. The value of the counter can be read through the corresponding days latch, but it cannot be modified. When TODEN is cleared, TOD counting is disabled and this counter can be modified. When TODEN is set to one again, counting will resume counting with the new set time.

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

Figure 13-11. TOD Days Register (TODDAY)

See Programmer's Sheet on Appendix page B-107

13.7.10 TOD Days Alarm Register (TODDAL)

When the value contained in this register matches the value of the days counter, the days alarm interrupt is asserted if the Days Alarm Enable (TODDA) bit and Alarm Interrupt Enable (TODAEN) bit are both set and all other enabled alarm registers also match the current time.

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

Figure 13-12. TOD Days Alarm Register (TODDAL)

See Programmer's Sheet on Appendix page B-108

Chapter 15
Reset, Low Voltage, Stop and Wait Operations

15.1 Introduction

This chapter is devoted to the description and sources of three different types of reset methods and their effects on the system used with this product. Those three reset methods are:

1. External
2. COP timeout
3. Low voltage

Additionally, the chapter describes control of the Stop and Wait modes.

15.2 Sources of Reset

Three sources of reset present in this system are:

1. External reset
2. Computer Operating Properly (COP) reset
3. Power-On Reset (POR)

Section 15-1 illustrates how these Reset sources are used within the chip.

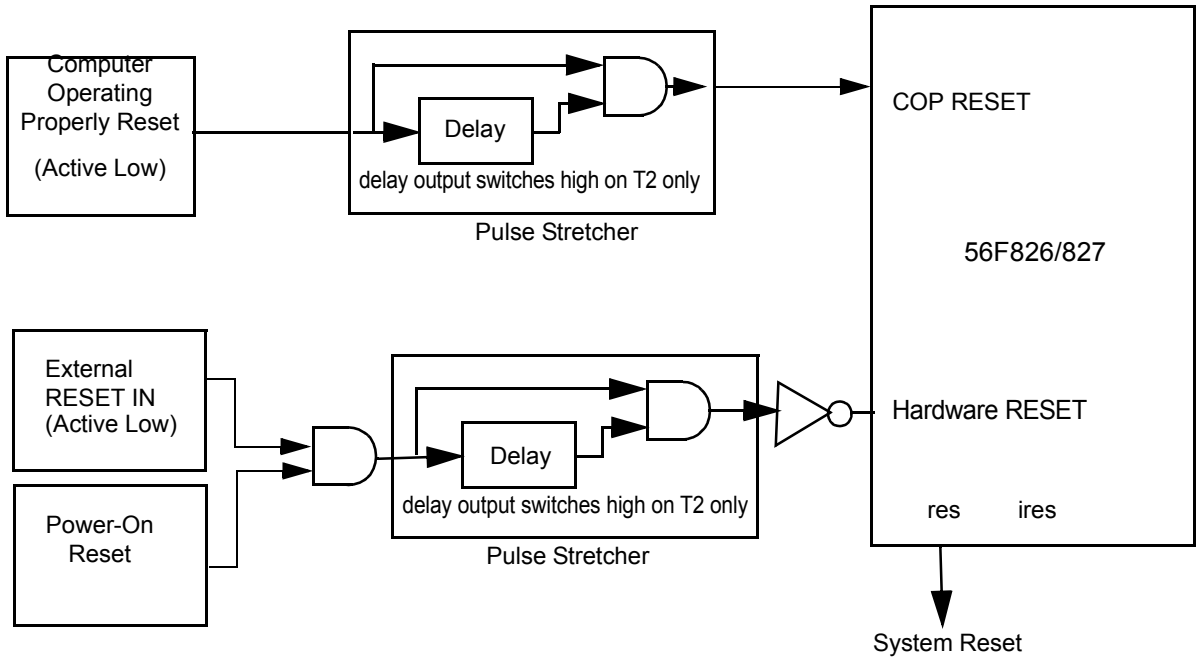


Figure 15-1. Sources of RESET

By default, the pulse stretcher function forces internal reset signals to be asserted a *minimum* of 63 oscillator clock cycles after the release of the external reset input.

If the external reset input is held high for three external clock cycles after the Power-On Reset (POR) circuit has detected V_{DD} greater than 1.8V, the pulse stretcher will force the internal reset signals for a minimum of 2,097, 151 external clock cycles. This mode of operation is called Internal Reset Generation (IRG). If the external clock crystal is 8MHz, the internal reset period is approximately 250ms. IRG mode ensures the clock oscillator has plenty of time to stabilize prior to system operation.

If an external reset circuit is connected to the external reset input, the external reset input can be held low during the POR process resulting in the IRG mode not being enabled.

The RSTO output signal mirrors the internal reset operation.

In addition to the reset sources above, two low voltage detect signals initiate a controlled shut down of the chip when voltage supply drops below acceptable levels. These low voltage detect circuits are assigned as high priority interrupts. They can be masked if desired. Please see **Section 15.8.1.**

15.3 Power-On Reset and Low Voltage Interrupt

The basic POR and low voltage detect module is shown in **Section 15-2.** The POR circuit is designed to assert the internal reset from V_{DD} = 0V to V_{DD} = 1.8V. POR switching high indicates there is enough voltage for on-chip logic to operate at the oscillator frequency. High speed operation is not guaranteed until both the 2.7V and 2.2V interrupt sources are inactive. Please see the SYS_STS register, **Section 15-9.**

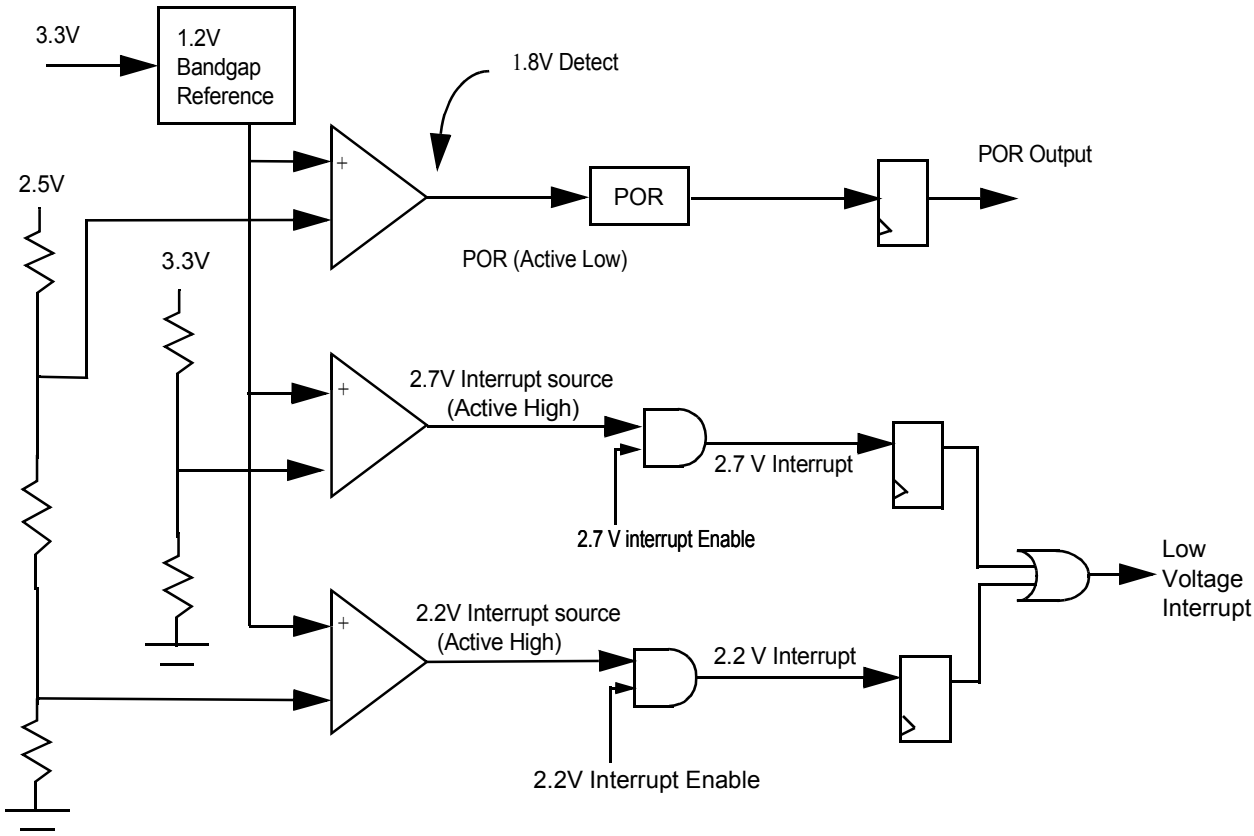


Figure 15-2. POR and Low Voltage Detection

The low voltage interrupts should be explicitly clear and disabled by the interrupt service routine responsible for shutting down the part when a low voltage is detected.

Section 15-3 illustrates one of the 2.2V and 2.7V interrupts. The 2.2V low voltage interrupts is generated based on the internal logic supply voltage and the 2.7V low voltage interrupt is generated based on the externally supplied V_{DD}.

The POR enable signal is used for test purposes only. Only the POR enable is low by default. Low voltage interrupts are masked upon POR. They must be explicitly enabled and disabled thereafter. Low voltage interrupts include roughly 50mV of hysteresis.

15.4 External Reset

The External Reset is active low, causing the part to be immediately placed in a reset condition. Some internal portions of the chip may be synchronously reset. The internal waveform shaper ensures the internal reset remains low long enough for all portions of the chip to achieve their reset value.

When the $\overline{\text{RESET}}$ pin is deasserted, the initial chip operating mode is latched into the OMR, based on the value present on the EXTBOOT pin. Additional details are found in **Section 3.3.2.**

15.5 Computer Operating Properly (COP) Module

The Computer Operating Properly (COP) module is used to help software recover from runaway code. The COP register is a free-running counter. Once enabled, it is designed to generate a reset on overflow. Software must periodically service the COP module in order to clear the counter and prevent a reset.

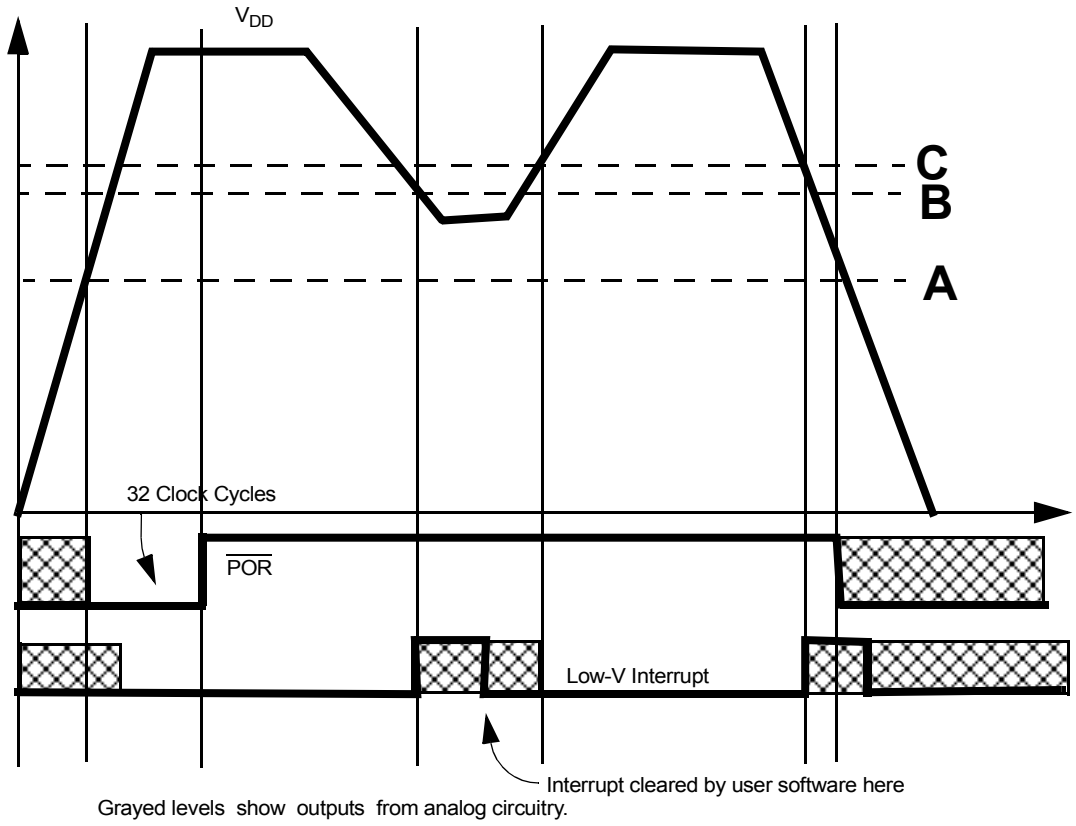


Figure 15-3. $\overline{\text{POR}}$ Vs. Low-Voltage Interrupts

15.6 COP Functional Description

15.6.1 Timeout Specifications

The COP uses a 12-bit counter with a prescaler to provide 4096 different timeout periods. The prescaler is a divide-by 16384 version of the CPU clock. At 80 MHz the CPU clock will give the COP a minimum timeout period of 250µs and a maximum of 839ms and a resolution of 205µs. The timeout period can be selected by writing to the COP Timeout (COPTO) bits (CT[11:0]) in the COPTO Register.

15.6.2 COP After Reset

When the COPCTL is cleared and out of reset, COP is disabled. Further, the COPTO is set to $FFF and the COPSRV is set to $000 during reset. The COP register can be enabled by setting the CEN bit in the COPCTL register.

The COP can be disabled by clearing the CEN bit in the COPCTL register. This action will reset the COP counter to the value in the timeout register. Resetting the COP module will also load the COP counter from the timeout register whose value is $FFF. Both COPCTL and COPTO may be modified as long as the CWP bit in the COPCTL register is clear. Once the CWP bit has been set, COPCTL and COPTO will be write-protected until a Reset occurs.

15.6.3 COP in Wait Mode

The COP module can run or be terminated in the Wait mode. If the COP Wait Enable (CWEN) bit in the COPCTL register is set, the COP counter will run in the Wait mode. However, when the CWEN bit is cleared, the COP counter will be disabled in the Wait mode.

15.6.4 COP in Stop Mode

The COP module can run or be disabled in Stop mode. When the COP Stop Enable (CSEN) bit in COPCTL is set, the COP counter will run in Stop mode. However, when the CSEN bit is cleared, the COP counter will be disabled in Stop mode.

15.7 Register Definitions

Table 15-4. COP/SIM Memory Map

Device	Peripheral	Address
826/827	COP_BASE	$1120
826/827	SIM_BASE	$1000

The address of a register is the sum of a base address and an address offset. The base address is defined at the unit level and the address offset is defined at the module level. Further data concerning the System Control register may be found in **Section 15.8.1.**

Table 15-5. COP/SIM Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + $0	COPCTL	Control Register	Read/Write	Section 15.7.1
Base + $1	COPTO	Timeout Register	Read/Write	Section 15.7.2
Base + $2	COPSRV	Service Register	Read/Write	Section 15.7.3
Base + $0	SYS_CNTL	System Control Register	Read/Write	Section 15.8.1
Base + $1	SYS_STS	Ssytem Status Register	Read/Write	Section 15.8.2
Base + $6	MSH_ID	Most Significant Half ID Register	*Read-Only*	Section 15.8.3
Base + $7	LSH_ID	Least Significant Half ID Register	*Read-Only*	Section 15.8.4

Bit fields of the above seven registers are summarized in **Figure 15-4.** Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	COPCTL	R	0	0	0	0	0	0	0	0	0	0	0	0	CSEN	CWEN	CEN	CWP
		W																
$1	COPTO	R	0	0	0	0	CT											
		W																
$2	COPSRV	R	0	0	0	0	COUNT											
		W	COP SERVICE REGISTER															
$0	SYS_CNTL	R	0	0	0	0	TMRPD	CTRLPD	ADRPD	DATAPD	0	0	0	BOOTMAP_B	LVIE27	LVIE22	PD	RPD
		W																
$1	SYS_STS	R	0	0	0	0	0	0	0	0	0	0	0	COPR	EXTR	POR	LVIS27	LVIS22
		W																
$6	MSH_ID	R	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1
		W																
$7	LSH_ID	R	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1
		W																

Both MSH and LSH register values are hard coded and cannot be reset.

R	0	Read as 0
W		Reserved

Figure 15-4. COP/SIM Registers Map Summary

15.7.1 COP Control Register (COPCTL)

COP_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	CSEN	CWEN	CEN	CWP
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-5. COP Control Register (COPCTL)

See Programmer's Sheet on Appendix page B-113

15.7.1.1 Reserved—Bits 15–4

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

15.7.1.2 Stop Enable (CSEN)—Bit 3

This bit controls the state of the COP counter in the Stop mode. This bit can only be modified when CWP is set to zero. For the COP to run in Stop mode, CEN must also be set.

- 0 = COP counter will stop in Stop mode
- 1 = COP counter will run in Stop mode

15.7.1.3 COP Wait Enable (CWEN)—Bit 2

This bit controls the state of the COP counter in the Wait mode. This bit can only be modified when CWP is set to zero. For the COP to run in the Wait mode, CEN must also be set.

- 0 = COP counter will stop in Wait mode
- 1 = COP counter will run in Wait mode

15.7.1.4 COP Enable (CEN)—Bit 1

This bit determines if the COP counter is enabled or disabled. This bit can only be modified when CWP is set to zero.

- 0 = COP is disabled
- 1 = COP is enabled

15.7.1.5 COP Write Protect (CWP)—Bit 0

This bit is used to control the write protection feature of the COP Control (COPCTL) register and COP Timeout (COPTO) register. Once set, this bit can only be cleared by system reset.

- 0 = COPCTL, COPTO may be modified
- 1 = COPCTL, COPTO are *read-only*

15.7.2 COP Timeout Register (COPTO)

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

Figure 15-6. COP Timeout Register (COPTO)

See Programmer's Sheet on Appendix page B-114

15.7.2.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

15.7.2.2 COP Timeout (CT)—Bits 11–0

COP timeout bits are used to control the length of the timeout period. COP timeout period equals [16384 × (CT[11:0]) + 1]) clock cycles.

The value in this register determines the timeout period, as shown in the above formula. The formula, CT[11:0], should be written before the COP is enabled. Once the COP has been enabled, the recommended procedure for changing CT[11:0] is to disable the COP, write to COPTO, and reactivate the COP. This ensures the new timeout value is loaded into the counter. Alternatively, the CPU can write to COPTO and then write the proper patterns to COPSRV, causing the counter to reload with the new CT[11:0] value. The COP counter is not reset by a write to COPTO.

Note: Changing CT[11:0] while the COP is enabled will result in a timeout period differing from the expected value given by the above formula. These bits can only be changed when CWP bit is set to zero.

15.7.3 COP Service Register (COPSRV)

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

Figure 15-7. COP Service Register (COPSRV)

See Programmer's Sheet on Appendix page B-115

When enabled, the COP requires a service sequence to be performed periodically, clearing the COP counter to prevent a reset. This routine consists of writing a $5555 to the COPSRV followed by writing a $AAAA to the COPSRV before the expiration of the selected timeout period. The

writes to COPSRV must be performed in the correct order before the counter times out, but any number of instructions may be executed between the two writes.

15.8 Stop and Wait Mode Disable Function

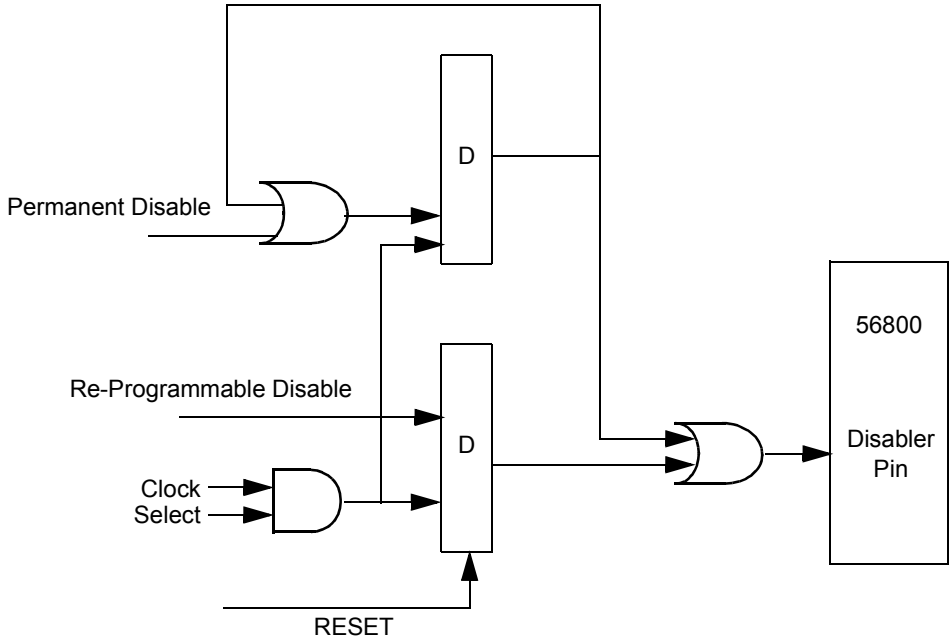


Figure 15-6. Stop/Wait Disable Circuit

The 56F826/827 contains both Stop and Wait instructions. Both instructions put the CPU to sleep. The PLL and peripheral bus continue to run in Wait mode, but not in Stop mode. The ADC will be placed in low power mode in both. In Stop mode, the clock system (ZCLK) is set equal to the oscillator output.

Some applications require the Stop/Wait instructions to be disabled. This can be accomplished on either a permanent or temporary basis by writing to the System Control (SYS_CNTL) register, as described in the next section. Permanently assigned applications are last only until their next reset.

15.8.1 System Control Register (SYS_CNTL)

SYS_BASE +$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	SPI/SCI PD	CTRLPD	SCI_SEL/ SPI_SEL	DATA PD	0	0	0	BOOT MAP_B	LVIE 27	LVIE 22	PD	RPD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-8. System Control Register (SYS_CNTL)

See Programmer's Sheet on Appendix page B-116

Note: Bit 9 is SCI_SEL for the 56F826 and SPI_SEL for the 56F827
Bit 8 is DATA_PD for the 56F826 and Reserved for the 56F827

15.8.1.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

15.8.1.2 Serial Peripheral Interface and Serial Communication Interface Pull-up Disable (SPI/SCI PD)—Bit 11

If this bit is set, the pull-ups for the SPI/SCI I/O pins are disabled. Normally the pull-ups are enabled.

15.8.1.3 Control Signal Pull-Up Disable (CTRL PD)—Bit 10

If this bit is set, the pull-ups for the $\overline{DS}$, $\overline{PS}$, $\overline{RD}$, and the $\overline{WR}$ I/O pins are disabled. Normally, the pull-ups are enabled.

15.8.1.4 Serial Communications Interface Select (SCI_SEL) 56F826–Bit 9

For the 56F826, if this bit is set, the four pins are routed to the SCI modules, SCI0 and SCI1. If this bit is clear, the four pins are routed to the SPI module, SPI0.

and Serial Communications Interface Select (SPI_SEL) 56F827—Bit 9

For the 56F827, if this bit is clear, the four pins are routed to the SPI module, SPI0. If this bit is set, the four pins are routed to the SCI modules, SCI0 and SCI1.

15.8.1.5 Data Bus I/O Pull-Up Disable (DATA PD)—Bit 8

In the 56F826 this bit is set, the pull-ups for the data bus I/O pins are disabled. Normally, the pull-ups are enabled. This bit is Reserved in the 56F827.

15.8.1.6 Reserved—Bits 7–5

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

15.8.1.7 Bootmap ($\overline{\text{BOOTMAP}}$)—Bit 4

This bit determines the memory map when the Operating Mode Register (OMR) MA and MB bits are set to enable internal program memory, illustrated in **Table 15-7.**

Table 15-7. Memory Map Controls

Mode Number	MA	MB	BOOTMAP1	Description
0A	0	0	0	Boot Mode
0B	0	0	1	Half Internal & Half External PMEM
0	0	1	X	Reserved
0	1	0	X	Reserved
3	1	1	X	Development

Note: 56F826/827 memory map details are found in **Table 3-42.**

15.8.1.8 2.7V Low Voltage Interrupt Enable (LVIE27)—Bit 3

When one, this bit enables the 2.7V low voltage interrupt. When zero, this interrupt is disabled.

15.8.1.9 2.2V Low Voltage Interrupt Enable (LVIE22)—Bit 2

When one, this bit enables the 2.2V low voltage interrupt. When zero, this interrupt is disabled.

15.8.1.10 Permanent Stop/Wait Disable (PD)—Bit 1

This bit can only be cleared by system reset. Once set, Stop and Wait instructions essentially act as loops.

15.8.1.11 Re-Programmable Stop/Wait Disable (RPD)—Bit 0

This bit can be set *and* cleared under software control. While set, Stop and Wait instructions act as loops.

15.8.2 System Status Register (SYS_STS)

This register is reset upon any system reset. It is initialized only by a Power-On Reset (POR).

SYS_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	COPR	EXTR	POR	LVIS 27	LVIS 22
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	—	—	—	0	0

Figure 15-9. System Status Register (SYS_STS)

See Programmer's Sheet on Appendix page B-118

15.8.2.1 Reserved—Bits 15–5

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

15.8.2.2 COP Reset (COPR)—Bit 4

When one, the COPR bit, indicates the computer COP timer generated reset has occurred. This bit will be cleared by a power-on reset, or by software. Writing 0 to this bit position will set the bit, while writing 1 to the bit will clear it.

15.8.2.3 External Reset (EXTR)—Bit 3

If one, the EXTR bit indicates an external system reset has occurred. This bit will be cleared by a power-on reset or by software. Writing 0 to this bit position will set the bit while writing 1 to the bit position will clear it. Setting this bit *will not* reset system.

15.8.2.4 Power-On Reset (POR)—Bit 2

When one, the POR bit indicates a power-on reset occurred some time previously. This bit can only be cleared by software. Writing 0 to this bit will set the bit, while writing 1 to the bit position will clear the bit. Setting this bit *will not* reset system.

15.8.2.5 2.7V Low Voltage Interrupt Source (LVIS27)—Bit 1

If one, the LVIS27 bit indicates a 2.7V low voltage interrupt is active. Writing 0 to this bit position will set this bit. Writing 1 to this bit position will clear this bit. Setting this bit will cause an interrupt if the corresponding interrupt enable bit is set.

15.8.2.6 2.2 Low Voltage Interrupt Source (LVIS 22)—Bit 0

If one, the LVIS22 bit indicates a 2.2V low voltage interrupt is active. Writing 0 to this bit position will set this bit. Writing 1 to this bit position will clear this bit. Setting this bit will cause an interrupt when the corresponding interrupt enable bit is set.

15.8.3 Most Significant Half of JTAG ID (MSH_ID)

This *read-only* register displays the Most Significant Half of the JTAG ID for the chip. For both 56F826/827 chips, read as $01F3.

SYS_BASE +$6	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	1
Write																
Reset	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1

Figure 15-10. Most Significant Half of JTAG_ID (MSH_ID)

See Programmer's Sheet on Appendix page B-119

15.8.4 Least Significant Half of JTAG ID (LSH_ID)

This *read-only* register displays the Least Significant Half of the JTAG ID for the chip. For both 56F826/827 chips, read as $A01D.

SYS_BASE +$7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	1	0	1	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

Figure 15-11. Least Significant Half of JTAG_ID (LSH_ID)

See Programmer's Sheet on Appendix page B-119

Chapter 16
OnCE Module

16.1 Introduction

This chapter describes the 56F800 core-based family chips providing board and chip-level testing capability through two on-chip modules. The 56F826/827 modules are accessed through the JTAG/OnCE™port. Those ports are:

- On-Chip Emulation (OnCE) module
- Test Access Port (TAP) plus a 16-state controller, known also as the Joint Test Action Group (JTAG) Port

The JTAG/OnCE port permits insertion of a chip into a target system while retaining debug control. This capability is especially important for devices without an external bus. This capability eliminates a required expensive cable bringing out the chip footprint necessary for a traditional emulator system. Additionally, the JTAG/OnCE port of the 56F826/827 can be used to program the internal flash memory and access OnCE module.

16.2 Features

The OnCE module is designed module integrated in its chips to debug application software used with the chip. The module is a separate on-chip block for non-intrusive interaction with the device, providing accessibility through the pins of the JTAG interface. The OnCE module permits examination of registers, memory, or on-chip peripherals' contents in a special debug environment, thereby avoiding sacrificing any on-chip resources while performing debugging. Capabilities of the OnCE module include:

- Interrupt or break into Debug mode on a program memory address to fetch, read, write, or access
- Interrupt or break into Debug mode on a data memory address to read, write, or access
- Interrupt or break into Debug mode on an on-chip peripheral register access to read, write, or access
- Enter Debug mode using a microprocessor instruction
- Display or modify the contents of any core register
- Display or modify the contents of peripheral memory-mapped registers
- Display or modify any desired sections of program or data memory
- Trace one, single stepping, or as many as 256 instructions
- Save or restore the current state of the chip's pipeline
- Display the contents of the real-time instruction trace buffer, regardless whether in Debug mode
- Return to user mode from Debug mode
- Set up breakpoints without being in Debug mode

- Set hardware breakpoints, software breakpoints, and trace occurrences (OnCE events) possibly forcing the chip into Debug mode, force a vectored interrupt, force the real-time instruction buffer to halt, or toggle a pin, based on the user's needs

Please refer to **Section 16.5** for a detailed description of this port.

Joint Test Action Group (JTAG) Port

The Joint Test Action Group (JTAG) port is a dedicated user-accessible Test Access Port (TAP) compatible with the IEEE 1149.1a-1993 Standard Test Access Port and Boundary Scan Architecture. Problems associated with testing high-density circuit boards have led to the development of this proposed standard under the sponsorship of the Test Technology Committee of IEEE and the JTAG. The 56F80x supports circuit board test strategies based on this standard.

Five dedicated pins interface to a TAP containing a 16-state controller. The TAP uses a boundary scan technique to test the interconnections between integrated circuits after they are assembled onto a printed circuit board. Boundary scan allow testers to observe and control signal levels at each component pin through a shift register placed next to each pin. This is important for testing continuity and determining if pins are stuck at one or zero level.

The TAP port has the following capabilities:

- Perform boundary scan operations to test circuit board electrical continuity
- Bypass the for a given circuit board test by replacing the Boundary Scan Register (BSR) with a single-bit register
- Provide a means of accessing the OnCE module controller and circuits to control a target system
- Disable the output drive to pins during circuit board testing
- Sample the system pins during operation and shift out the result in the BSR; preload values outputing pins prior to invoking the EXTEST instruction
- Query identification information, manufacturer, part number, and version from a chip
- Force test data onto the outputs of a IC, while replacing its BSR in the serial data path with a single-bit register
- Enable a weak pull-up current device on all input signals of a IC, ensuring deterministic test results in the presence of a continuity fault during interconnect testing

16.3 Combined JTAG/OnCE Interface Overview

The JTAG and OnCE blocks are tightly coupled. **Figure 16-1** illustrates the block diagram of the JTAG/OnCE port with its two distinctive parts. *The JTAG port is the master.* It must enable the OnCE module before the OnCE module can be accessed.

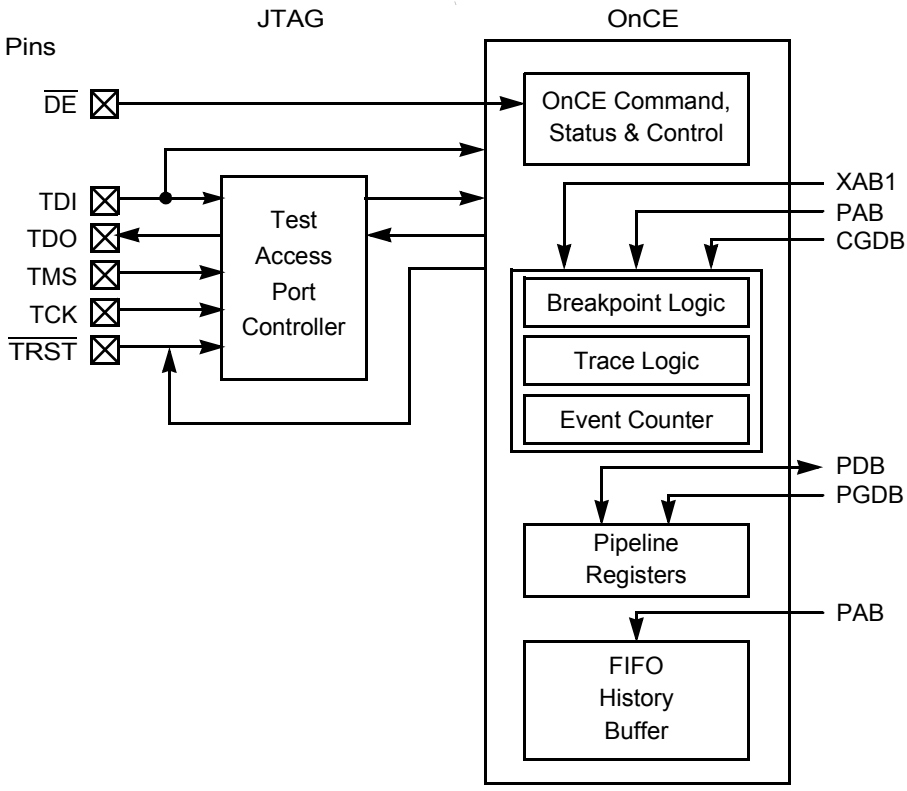


Figure 16-1. JTAG/OnCE Port Block Diagram

There are three different programming models to consider when using the JTAG/OnCE interface:

1. OnCE programming model—accessible through the JTAG port
2. OnCE programming model—accessible from the core
3. JTAG programming model—accessible through the JTAG port

The programming models are discussed in more detail in **Section 16.5** and **Section 17.4**.

16.4 JTAG/OnCE Port Pin Descriptions

As described in the IEEE 1149.1a-1993 specification, the JTAG port requires a minimum of four pins to support TDI, TDO, TCK, and TMS signals. The 56F826/827 also uses the optional Test Reset ($\overline{\text{TRST}}$) input signal and a pin used for Debug Event ($\overline{\text{DE}}$) monitoring. Pin functions are detailed in **Table 16-1**.

Table 16-1. JTAG/OnCE Pin Descriptions

Pin Name	Pin Description
TDI	**Test Data Input**—This input provides a serial data stream to the JTAG and the OnCE module. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
TDO	**Test Data Output**—This tri-stateable output provides a serial data stream from the JTAG and the OnCE module. It is driven in the Shift-IR and Shift-DR controller states of the JTAG state machine and changes on the falling edge of TCK.
TCK	**Test Clock Input**—This input proves a gated clock to synchronize the test logic and shift serial data through the JTAG/OnCE port. The maximum frequency for TCK is 1/8 the maximum frequency of the 56F80x (i.e., 5 MHz for TCK if the maximum CLK input is 40 MHz). The TCK pin has an on-chip pull-down resistor.
TMS	**Test Mode Select Input**—This input sequences the TAP controller's state machine. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
$\overline{\text{TRST}}$	**Test Reset**—This input provides a reset signal to the TAP controller. This pin has an on-chip pull-up resistor.
$\overline{\text{DE}}$	**Debug Event**—This output signals OnCE events detected on a trigger condition.

The $\overline{\text{DE}}$ pin provides a useful event acknowledge feature. This selection is performed through appropriate control bits in the OnCE Control Register (OCR).

- When $\overline{\text{DE}}$ is enabled, its output provides a signal indicating an event has occurred in the OnCE Debug Logic. This event can be any of these occurrences:
  - Hardware Breakpoint
  - Software Breakpoint
  - Trace or enter Debug mode by a DEBUG_REQUEST instruction being decoded in the JTAG port Instruction Register (IR)

The $\overline{\text{DE}}$ output indicates an OnCE event occurred. For example, a trace or breakpoint occurrence causes the pin to go low for a minimum of eight Phi clocks. This pin is detailed in **Section 16.7.5.5.**

When the JTAG Instruction Register (JTAGIR) does not contain an ENABLE_ONCE instruction, the OCR control bits are reset only by assertion of $\overline{\text{RESET}}$ or COP timer reset. When the ENABLE_ONCE instruction is in the JTAGIR at the time of reset, the OCR bits *are not* modified.

16.5 OnCE Module Architecture

While the JTAG port, described in **Section 17.4,** provides board test capability, the OnCE module provides emulation and debug capability. The OnCE module permits full-speed, non-intrusive emulation on a user's target system or on a Freescale Evaluation Module (EVM) board.

A typical debug environment consists of a target system where the resides in the user-defined hardware. The device's JTAG port interfaces to a command converter board over a seven-wire link. The converter board consists of the five JTAG serial lines, a ground, and reset wire. The reset wire is optional and is only used to reset the and its associated circuitry.

The OnCE module is composed of four different sub-blocks, each performing a different task:

- Command, status, and control
- Breakpoint and trace
- Pipeline registers
- FIFO history buffer

A block diagram of the OnCE module is shown in **Figure 16-2.** The registers serially shift information from TDI to TDO through the OnCE Shift Register (OSHR). In **Figures 16-2** and continued in **Figure 16-3** displays the OnCE module registers accessible through the JTAG port. The same module portrays the OnCE module registers accessible through the core and its corresponding OnCE interrupt vector.

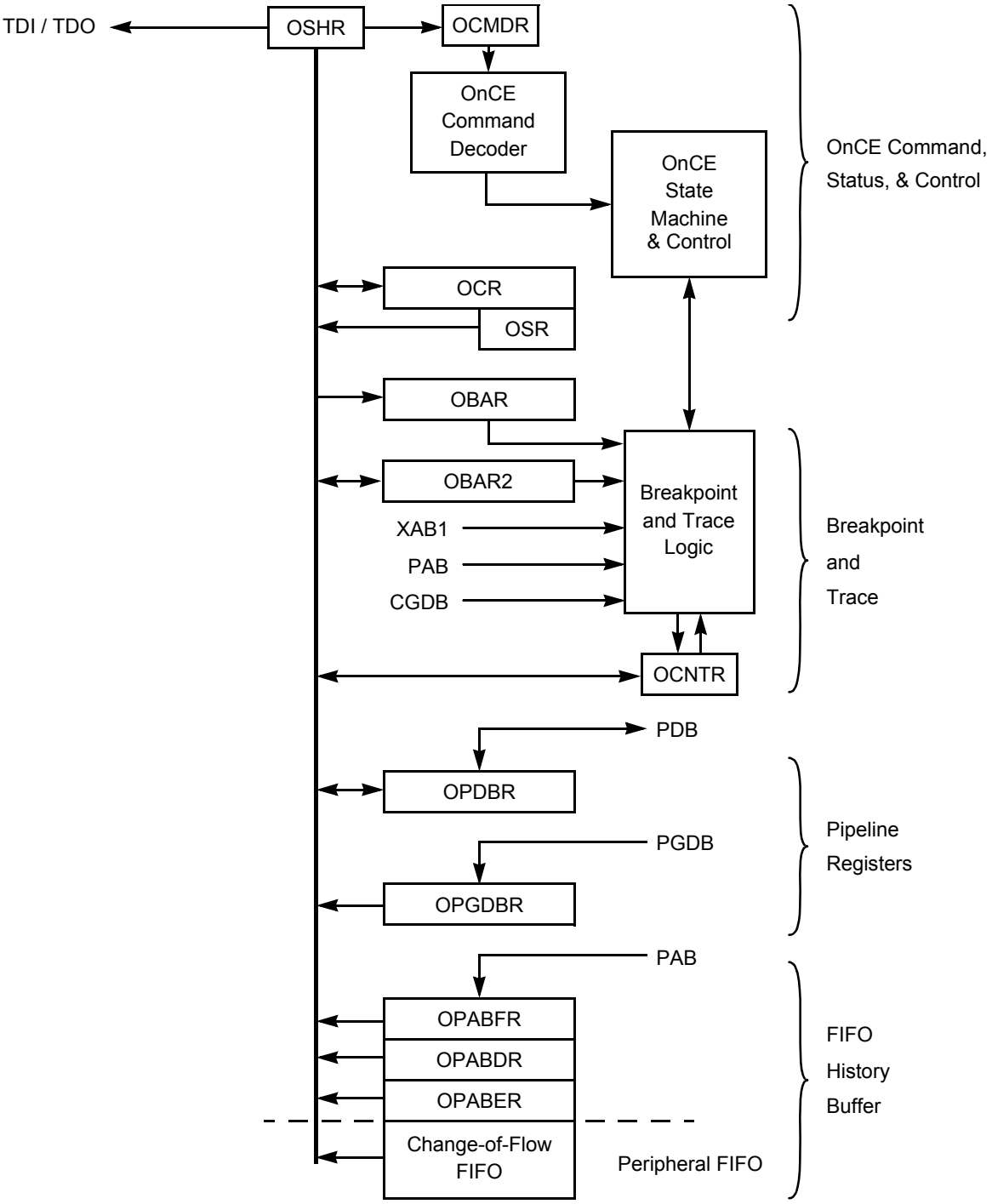


Figure 16-2. 56F80x OnCE Block Diagram

16.6 Register Summary

The OnCE module registers of the 56F826/827:

- OnCE Breakpoint Address Register (OBAR)
- OnCE Breakpoint Address 2 (OBAR2) Register
- OnCE Breakpoint Control 2 (OBCTL2) Register
- OnCE Command Register (OCMDR)
- OnCE Breakpoint Mask 2 (OBMSK2) Register
- OnCE Count Register (OCNTR)
- OnCE Control Register (OCR)
- OnCE Decoder (ODEC) (*not memory mapped)*
- OnCE Memory Address Comparator (OMAC) (*not memory mapped)*
- OnCE Memory Address Comparator 2 (OMAC2) (*not memory mapped)*
- OnCE Memory Address Latch (OMAL) (*not memory mapped)*
- OnCE Memory Address Latch 2 (OMAL2) (*not memory mapped)*
- OnCE PAB (Program Address Bus) Decode Register (OPABDR)
- OnCE PAB (Program Address Bus) Execute Register (OPABER)
- OnCE PAB (Program Address Bus) Fetch Register (OPABFR)
- OnCE PDB (Program Data Bus) Register (OPDBR)
- OnCE PAB Change of Flow (OPFIFO) (*not memory mapped)*
- OnCE PDGB (Program Global Data Bus) Register (OPGDBR)
- OnCE Shift Register (OSHR) (*not memory mapped)*
- OnCE Status Register (OSR)

OCMDR—$00	7	6	5	4	3	2	1	0
Write Only	R/W	GO	EX	RS4	RS3	RS2	RS1	RS0
Reset								

OnCE Command Register

OSR—$01	7	6	5	4	3	2	1	0
Read Only	0	0	0	EM1	EM0	TO	HBO	SBO
Reset								

OnCE Status Register OnCE Reset* = $00

Figure 16-2. OnCE Module Registers Accessed Through JTAG

OCR—$02	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COP DIS	RES	BK4	BK3	BK2	BK1	BK0	RES	FH	EM1	EM0	PWD	BS1	BS0	BE1	BE0
Write																
Read	0				0				0				0			

OnCE Control Register

OCNTR-$03	7	6	5	4	3	2	1	0
Read	Eight-Bit Event Counter							
Write								
Reset	0				0			

OnCE Count Register

OBAR—$04	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write-Only	16-bit Breakpoint Address Register (Program or Data Memory Breakpoints)															
Reset																

OnCE Breakpoint Address Register

= Reserved bits, written as zero for future compatibility.

Note: OnCE reset occurs when hardware or COP rest occurs, and an ENABLE_OnCE instruction is not latched into the JTAG Instruction Register (IR).

OPGDBR—$08	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Register Used to Read Registers and Memory Values From the Core															
ROM																

OnCE PGDB Register

OPDBR—$09	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	16-Bit Register Used to Execute Instructions in Debug Mode and Restore Pipeline Upon Exit															
Write																
Reset																

OnCE PDB Register

Figure 16-2. OnCE Module Registers Accessed Through JTAG (Continued)

OPABFR—$0A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Program Fetch Address															
Reset																

OnCE PDB Fetch Register

OPABER—$10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Program Execute Address															
Reset																

OnCE PDB Execute Register

OPABDR—$13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Program Decode Address															
Reset																

OnCE PDB Decode Register

OBAR2—$05	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	16-bit Breakpoint Address Register (Address or Data Breakpoints)															
Write																
Reset																

OnCE Breakpoint Address Register 2

Figure 16-2. OnCE Module Registers Accessed Through JTAG (Continued)

OBMSK2—$06	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	16-Bit Breakpoint Mask Register (Address or Data Breakpoints)															
Write																
Reset																

OnCE Breakpoint Mask Register 2

OBCTL2—$07	2	1	0
Read	EN	INV	DAT
Write			
Reset	0	0	0

OnCE Breakpoint Control Register 2

Once reset occurs when hardware or Computer Operating Properly (COP) reset occurs, and an ENABL is not latched into the JTAG port Instruction Register (IR).

OPGDBR—X:$FFFF	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Register Used to Read Registers and Memory Values From the Core															
Reset																

OnCE PGDB Register

OPDBR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Register Used to Execute Instructions in Debug Mode and Restore Pipeline Upon Exit															
Reset																

OnCE PDB Register

Figure 16-3. OnCE Module Registers Accessed From the Core

OnCE Port Interrupt Vector:

OnCE Trap P:$000C

OnCE TRAPs are level 1 interrupts and are not programmable in the IPR.

Note: OnCE module registers. They share functionality with the core. If used incorrectly, they can give unexpected results.

The OnCE module has an associated interrupt vector $000C. The Event Modifier (EM) bits of the OCR can configure the event so an OnCE event, other than trace, generates a level- one non-maskable interrupt. This interrupt ability is described in **Section 16.7.5.5.**

16.7 Command, Status, and Control Registers

The OnCE command, status, and control registers are described in subsections:

- OnCE Breakpoint Control 2 (OBCTL2) register
- OnCE Command Register (OCMDR)
- OnCE Control Register (OCR)
- OnCE Decoder Register (ODEC) (*not memory mapped)*
- OnCE Status Register (OSR)
- OnCE Shift Register (OSHR) (*not memory mapped)*

16.7.1 OnCE Shift Register (OSHR)

The OnCE Shift Register (OSHR) is a JTAG Shift register sampling the TDI pin on the rising edge of TCK in shift-DR while providing output to the TDO pin on the falling edge of TCK. The last bit of TDI will be sampled upon exiting shift-DR on the rising edge of TCK. During OCMDR/OSR transfers, this register is eight bits wide. However, during all other transfers, this register is 16 bits wide. The input from TDI must be presented to the OSHR Least Significant Bit (LSB) first. The TDO pin receives data from the LSB of the OSHR. This register is *not* memory mapped. It is not possible to modify this register.

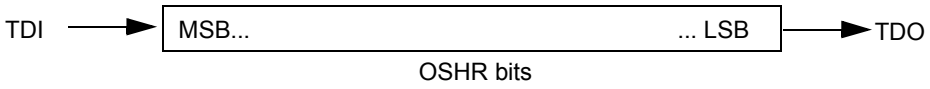


Figure 16-4. OnCE Shift Register (OSHR)

Note: OnCE instructions are shifted on the data side (Select-DR-Scan) of the TAP controller not the instruction side (Select-IR-Scan).

16.7.2 OnCE Command Register (OCMDR)

The OnCE module has its own instruction register and instruction decoder, the OnCE Module Command Register (OCMDR). After a command is latched into the OCMDR, the command decoder implements the instruction through the OnCE state machine and control block. There are two types of commands:

1. Read commands causes the chip to deliver required data.
2. Write commands transfers data into the chip, then write it in one of the on-chip resources.

The commands are eight bits long with the format displayed in **Figure 16-3.** The lowest five bits (RS[4:0]) identify the source for the operation, defined in **Table 16-5.** Bits five, six, and seven delineate the Exit (EX) command bit, shown in **Table 16-6.** The execute START (GO) bit is found in **Table 16-7,** while the Read/Write (R/$\overline{W}$) bit is illustrated in **Table 16-8,** respectively.

OCMDR	7	6	5	4	3	2	1	0
Write-Only	R/$\overline{W}$	GO	EX	RS4	RS3	RS2	RS1	RS0
Reset								

Figure 16-3. OnCE Command Format

Table 16-5. Register Select Encoding

RS[4:0]	Register/Action Selected	Mode	Read/$\overline{\text{Write}}$
00000	No register selected	All	N/A
00001	OnCE Breakpoint and Trace Counter (OCNTR)	All	Read/Write
00010	OnCE Debug Control Register (OCR)	All	Read/Write
	Reserved		
00100	OnCE Breakpoint Address Register (OBAR)	All	Write-only
	Reserved		
	Reserved		
	Reserved		
01000	OnCE PGDB Bus Transfer Register (OPGDBR)	Debug	Read-only
01001	OnCE Program Data Bus Register (OPDBR)	Debug	Read/Write
01010	OnCE Program Address Register—Fetch cycle (OPABFR)	FIFO halted	Read-only
	Reserved		
01100	Clear OCNTR	All	N/A
	Reserved		
	Reserved		
	Reserved		
10000	OnCE Program Address Register—Execute cycle (OPABER)	FIFO halted	Read-only
10001	OnCE Program address FIFO (OPFIFO)	FIFO halted	Read-only

Table 16-5. Register Select Encoding (Continued)

RS[4:0]	Register/Action Selected	Mode	Read/$\overline{\text{Write}}$
	Reserved		
10011	OnCE Program Address Register—Decode cycle (OPABDR)	FIFO halted	Read-only
	Reserved		
	Reserved		

Table 16-6. EX Bit Definition

EX	Action
0	Remain in the Debug Processing State
1	Leave the Debug Processing State

Table 16-7. GO Bit Definition

GO	Action
0	Inactive—No Action Taken
1	Execute Instruction

Note: In the OnCE command word, bit five is the exit command. To leave Debug mode and reenter Normal mode, both the EX and GO bits must be asserted in the OnCE input command register.

Table 16-8. R/$\overline{\text{W}}$ Bit Definition

R/$\overline{\text{W}}$	Action
0	Write to the register specified by the RS[4:0] bits
1	Read from the register specified by the RS[4:0] bits

16.7.3 OnCE Decoder (ODEC)

The OnCE Decoder (ODEC) decodes all OnCE instructions received in the OCMDR. The ODEC generates all the strobes required for reading and writing the selected OnCE registers. When the chip is not in Debug mode, this block prohibits access to the OnCE Program Data Bus Register (OPDBR) and the OnCE PGDB Bus Transfer Register (OPGDBR). Indeterminate result when the FIFO is halted while accessing the Program Address Bus (PAB) pipeline registers from User mode. The ODEC works closely with the OnCE state machine on register reads and writes. *This register is not memory mapped and cannot be modified.*

16.7.4 OnCE Control Register (OCR)

The 16-bit OnCE Control Register (OCR) contains bit fields determining how breakpoints are triggered, what action occurs when a OnCE event occurs, as well as controlling other miscellaneous OnCE features. **Figure 16-9** illustrates the OCR and its fields.

OCR—$02	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COP DIS	0	BK4	BK3	BK2	BK1	BK0	0	FH	EM1	EM0	PWD	BS1	BS0	BE1	BE0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-9. OCR Programming Model

16.7.5 COP Timer Disable (COPDIS)—Bit 15

The COP Disable (COPDIS) timer bit is used to prevent the COP timer from resetting the chip when it times out. When COPDIS is cleared, the COP timer is enabled. When COPDIS is set, the COP timer is disabled.

Note: When the COP Enable (CPE) bit in the COP/RTI Control (COPCTL) register is cleared, the COP timer is not enabled. In this case, the COPDIS bit has no effect on the deactivated COP timer. That is, COP reset can not be generated. However, the COPDIS bit overrides the CPE bit when both are set.

16.7.5.1 Reserved—Bit 14

This bit is reserved or not implemented. It must always remain 0.

16.7.5.2 Breakpoint Configuration (BK)—Bits 13–9

The Breakpoint (BK) configuration bits are used to configure the operation of the OnCE module when it enters the debug processing state. In addition, these bits can also be used to setup a breakpoint on one address and an interrupt on another address. **Table 16-10** lists the different breakpoint configurations.

Table 16-10. Breakpoint Configuration Bits Encoding—Two Breakpoints

BK[4:0]	Final Trigger Combination and Actions
00000	Breakpoint 1 occurs the number of times specified in the OCNTR. Then the trigger is generated.
00001	Breakpoint 1 or Breakpoint 2 occur the number of times specified in the OCNTR. Then the trigger is generated.
00010	Breakpoint 1 and Breakpoint 2 must occur simultaneously the number of times specified in the OCNTR. Then the trigger is generated.
00100	Breakpoint 1 generates trigger; Breakpoint 2 generates a OnCE Interrupt.
01011	Breakpoint 2 occurs once, followed by Breakpoint 1 occurring the number of times specified in the OCNTR. Then the trigger is generated.
01111	Breakpoint 2 occurs the number of times specified in the OCNTR, followed by Breakpoint 1 occurring once. Then the trigger is generated.
10000	Breakpoint 1 occurs once, followed Trace mode using the instruction count specified in the OCNTR. Then the trigger is generated.
10001	Breakpoint 1 or Breakpoint 2 occurs once, followed Trace mode using the instruction count specified in the OCNTR. Then the trigger is generated.
10010	Breakpoint 1 and Breakpoint 2 occur simultaneously, followed Trace mode using the instruction count specified in the OCNTR. Then the trigger is generated.
10100	Breakpoint 1 occurs once, followed Trace mode using the instruction count specified in the OCNTR. Then the trigger is generated; Breakpoint 2 generates a OnCE Interrupt.
10111	Trace mode using the instruction count specified in the OCNTR.
11011	Breakpoint 2 occurs once, followed by Breakpoint 1 occurring once, followed Trace mode using the instruction count specified in the OCNTR. Then the trigger is generated.
All other encodings of BK[4:0] are illegal and may cause unpredictable results.	

Breakpoint 2 is a simple address comparison. It is unaffected by BS/BE bits, except BE = 00 disables it. BE = 00 disables all of the above BK settings except pure Trace mode (BK[4:0] = 10111). Please see **Section 16.7.5.9** for more information.

16.7.5.3 Reserved—Bit 8

This bit is reserved or not implemented. It must always remain 0.

16.7.5.4 FIFO Halt (FH)—Bit 7

The FIFO Halt (FH) bit permits a halt address capture in the OnCE change-of-flow FIFO (OPFIFO) register, the OnCE PAB Fetch Register (OPABFR), the OnCE PAB Decode Register (OPABDR), and the OnCE PAB Execute Register (OPABER) when the bit is set. The FH bit is set only by writes to the OCR. It is never automatically set by on-chip circuitry. It is a control bit, never a status bit. This provides a simple method to halt the PAB history capture without setting up event conditions using the EM bits and breakpoint circuitry.

Note: The FIFO is halted immediately after the FH bit is set. This means the FIFO can be halted in the middle of instruction execution, leading to incoherent OPFIFO register contents.

16.7.5.5 Event Modifier (EM)—Bits 6–5

The Event Modifier (EM[1:0]) bits allow different actions to occur when an OnCE event develops. OnCE events are defined as:

Occurrences of hardware breakpoints, software breakpoints, and traces.

Each event occurrence sets the respective occurrence bit, HBO, SBO, or TO in the OnCE Status Register (OSR), regardless of EM encoding. In addition, when the $\overline{DE}$ pin is enabled, each event occurrence drives $\overline{DE}$ low until the event is rearmed, again regardless of EM encoding.

The first trigger condition of a breakpoint sequence does not set a bit in the OSR. Only completion of the final trigger condition sets the respective bit in the OSR Hardware Breakpoint Occurrence (HBO) for all but one BK encoding.

When EM[1:0] = 00, an OnCE event halts the core and the chip enters Debug mode. The core is halted on instruction boundaries only. Access to core registers, data, program memory locations, and peripheral memory mapped registers is permitted when the core is halted. It is possible to execute core instructions forced into the instruction pipeline through OnCE module transfers.

If EM[1:0] = 01, the core does not halt when an event occurs. For example: Debug mode is not entered but the OPFIFO, the OPABFR, the OPABDR, and the OPABER registers stop capturing. This allows access to the PAB history information while the application continues to execute.

If EM[1:0] =10, the core does not halt when an event occurs, but a level one interrupt occurs with a vector at location P:$000C. This interrupt permits diagnostic subroutines execution upon event occurrences, or even to patch program memory by setting a breakpoint at the beginning of the code to be patched.

Note: Trace occurrences do not trigger vectored interrupts. Only hardware and software breakpoints are allowed OnCE events for this EM encoding.

If EM[1:0] = 11, the core does not halt and no other action is taken other than the pulsing low of $\overline{DE}$ when enabled. This encoding serves to produce an external trigger without changing OnCE module or core operation.

EM encodings 11 and 10 enable automatic event rearming. That is, eight Phi clock cycles after the event occurrence flag HBO, TO, or SBO is set, it is reset, thus rearming the event. If $\overline{DE}$ is enabled, it is asserted, or driven low for eight Phi clock cycles, then released. If another event occurs within those eight Phi clock cycles or immediately after the cycles, the occurrence flag is promptly set and $\overline{DE}$ remains low.

To rearm an event in EM encoding 00, Debug mode must be exited, typically by executing a core instruction when setting EX and GO in the OCMDR, thereby clearing the status bits and releasing $\overline{DE}$.

To rearm an event in EM encoding 10, the OCR must be modified (written). If the OCR value is to remain constant, writing OCR with its current value successfully rearms the event. When $\overline{DE}$ is activated and released rearming occurs.

Enabling trace for EM = 11 or EM = 10 is not particularly useful. Since a Trace event occurs on every instruction execution once OCNTR reaches zero, the event is continuously set, meaning $\overline{DE}$ stays low after the first event. For EM = 10, vectoring is disabled on Trace occurrences, though $\overline{DE}$ goes low and stays low after the first trace occurrence. The appropriate event occurrence bit is not reset in this case, tracing and OCNTR = $0000, until Trace mode is disabled and an event-clearing action takes place, such as exiting Debug mode or writing the OCR while in User mode.

Note: Any OCR write in User mode resets the event flags, while OCR writes in Debug mode, do not reset the event flags.

Table 16-11 summarizes the different EM encodings.

Table 16-11. Event Modifier Selection

EM[1:0]	Function	Action on Occurrence of an Event
00	Enter Debug Mode	The core halts and Debug mode is entered. FIFO capture is automatically halted. The event is rearmed by exiting Debug mode.
01	FIFO Halt	Capture by the OPABFR, the OPABDR, the OPABER, and FIFO is halted. The user program is unaffected. The event is rearmed by writing to the OCR.
10	Vector Enable	The user program is interrupted by the OnCE event. Program execution goes to P:$000C, FIFO capturing continues, the event is automatically rearmed, and the user program continues to run. Trace occurrences do not cause vectoring, though the TO bit is set and $\overline{DE}$ is asserted.
11	Rearm	The event is automatically rearmed. FIFO capture continues, and the user program continues to run.

Note: When events are rearmed, OCNTR is not reloaded with its original value. It remains at zero and the next triggering condition generates an event.

Use care when changing EM bits. It is recommended a particular event being changed, such as trace, hardware, or software breakpoint is disabled first. On the next OCR write, the EM bits can be modified and the event re-enabled. This is only required when the chip is not in Debug mode. Improper operation can occur if this is not followed. For example, if the FIFO has halted due to an event occurrence with EM[1:0] = 01 and the next OCR write changes EM[1:0] to 00, the chip enters Debug mode immediately. Automatic rearming is desirable if the 10 encoding is being used for a ROM patch or the 11 encoding is used for profiling code. The EM[1:0] bits add some powerful debug techniques to the OnCE module. Profile code easily with the 01 encoding, or perform special tasks when events occur with the 10 encoding.

The most attractive feature of the 10 encoding is the ability to patch the ROM. If a section of code in ROM is incorrect, set a breakpoint at the starting address of the bad code and vector off to a program RAM location where the patch resides. There are also BK encodings available for this purpose.

The 11 encoding is useful for toggling the $\overline{DE}$ pin output. Count events on the $\overline{DE}$ output to determine how much time is being spent in a certain sub-routine or other useful things. $\overline{DE}$ is held low for two instruction cycles to avoid transmission line problems at the board level at high internal clock speeds. This restricts event recognition to no more than one event every three instruction cycles, limiting its usefulness during tracing.

16.7.5.6 Power-Down Mode (PWD)—Bit 4

Power-Down mode (PWD) bit is a power-saving option designed to reduce running current for applications not using the OnCE module. The PWD bit can be set, or reset by writing to the OCR. On hardware reset, deassertion of the $\overline{RESET}$ signal, this bit is set at Low Power mode if the JTAG TAP controller is not decoding an ENABLE_ONCE command. If the ENABLE_ONCE command is being decoded, the bit can be set or cleared only through a OnCE module write command to the OCR. Breakpoints should be completely disabled before setting PWD, assuring proper operation.

When the OnCE module is powered down (PWD = 1), much of the OnCE module is shut down, although the following two events can still occur:

1. JTAG DEBUG_REQUEST instruction still halts the core.
2. The OnCE module state machine is still accessible permitting a write to the OCR.

Note: Debug instructions executed by the core are ignored if PWD is set and no event occurs.

16.7.5.7 Breakpoint Selection (BS)—Bits 3–2

Breakpoint Selection (BS[1:0]) control bits select whether the breakpoints are recognized on Program Memory Fetch, Program Memory Access, or First X Memory Access. These bits are cleared on hardware reset, and are described in **Table 16-12.** These bits are used only in determining triggering conditions for Breakpoint 1, not for additional future breakpoint comparators.

The BS and Breakpoint Enable (BE) bits apply only to the Breakpoint 1 mechanism. Breakpoint 2 and future breakpoint mechanisms are unaffected by BS or BE bit encodings except when all breakpoint mechanisms are disabled at BE[1:0] = 00.

Table 16-12. BS[1:0] Bit Definition

BS[1:0]	Action on Occurrence of an Event
00	Breakpoint on Program Memory Fetch (fetch of the first word of instructions are actually executed, not of those killed, not of those the second word of two-word instructions, and not of jumps not taken)
01	Breakpoint on any Program Memory Access (any MOVEM instructions, fetches of instructions executed and of instructions killed, fetches of second word of two-word instructions, and fetches of jumps not taken)
10	Breakpoint on the First X Memory Access—XAB1/CGDB access
	Reserved

Note: It is not possible to set a Breakpoint on the XAB2 bus when it is used in the second access of a dual read instruction.

The BS[1:0] bits work in conjunction with the BE[1:0] bits determining how the address breakpoint hardware is setup. The decoding scheme for BS[1:0] and BE[1:0] is shown in **Table 16-13.**

16.7.5.8 Breakpoint Enable (BE[1:0])—Bits 1–0

The Breakpoint Enable (BE[1:0]) control bits enable or disable the breakpoint logic selecting the type of memory operations: read, write, or access, upon operation of the breakpoint logic. *Access* means either a read/write can take place. These bits are cleared on hardware reset. **Table 16-14** describes the bit functions.

Table 16-13. Breakpoint Programming with the BS[1:0] and BE[1:0] Bits

Function	BS[1:0]	BE[1:0]
Disable All Breakpoints *	A	00
Reserved		
Program Instruction Fetch	00	10
Reserved		
Any Program Write or Fetch	01	01
Any Program Read or Fetch	01	10
Any Program Access or Fetch	01	11
XAB1 Write	10	01
XAB1 Read	10	10
XAB1 Access	10	11
Reserved		
Reserved		
Reserved		

* When all breakpoints are disabled with the BE[1:0] bits set to 00, the full-speed instruction tracing capability is not affected. See **Section 16.12.2**.

Table 16-14. BE[1:0] Bit Definition

BE[1:0]	Selection
00	Breakpoint disabled
01	Breakpoint enabled on memory write
10	Breakpoint enabled on memory read
11	Breakpoint enabled on memory access

The BE[1:0] bits work in conjunction with the BS[1:0] bits determining how the address breakpoint hardware is setup. The decoding scheme for BS[1:0] and BE[1:0] is shown in **Table 16-13.** Breakpoints should remain disabled until after the OBAR is loaded. See **Section 16.8.5** and **Section 16.12.2** for a description of tracing and breakpoints. Breakpoints can be disabled or enabled for one memory space.

16.7.5.9 OnCE Breakpoint 2 Control Register (OBCTL2)

OnCE Breakpoint 2 Control (OBCTL2) register is a 3-bit register used to program Breakpoint 2. Please refer to **Figure 16-15.** The register is read/write by the OnCE unit. It is used to setup the second breakpoint for Breakpoint Operation. This register is accessed
as the lowest three bits of a 16-bit word. The upper bits are reserved and should be written with 0, ensuring future compatibility.

OBCTL2—$07	2	1	0
Read	EN	INV	DAT
Write			
Reset	0	0	0

Figure 16-15. OnCE Breakpoint Control Register 2 (OBCTL2)

16.7.5.10 Reserved—Bits 15–3

This bit field is reserved or not implemented. It is read as 0 during read operations. Modify these bits by writing 0, assuring future compatibility.

16.7.5.11 Enable (EN)—Bit 2

The Enable (EN) bit is used to activate the second breakpoint unit. When EN is set, the Second Breakpoint Unit is enabled. When EN is cleared, the Second Breakpoint Unit is disabled.

16.7.5.12 Invert (INV)—Bit 1

The Invert (INV) bit is used to specify whether to invert the result of the comparison before sending it to the Breakpoint and Trace units. When INV is set, the Second Breakpoint Unit inverts the result of the comparison. Inversion is not performed when INV is cleared.

16.7.5.13 Data/Address Select (DAT)—Bit 0

The Data/Address select (DAT) bit determines which bus is selected by the Second Breakpoint Unit. When DAT is set, the Second Breakpoint Unit examines the Core Global Data Bus (CGDB). When DAT is cleared, the Program Address Bus (PAB) is examined.

16.7.6 OnCE Status Register (OSR)

The OnCE Status Register (OSR) is illustrated in **Figure 16-16.** By observing the values of the five status bits in the OSR, determine if the core has halted, what caused it to halt, or why the core has not halted in response to a debug request. The OSR value is visible when shifting in a new OnCE command, writing to the OCMDR, and allowing for efficient status polling. The OSR and all other OnCE registers are inaccessible in Stop mode.

Bits	7	6	5	4	3	2	1	0
Read	0	0	0	OS1	OS0	TO	HBO9	SBO
Write								
Reset	0	0	0	0	0	0	0	0

Figure 16-16. OnCE Status Register (OSR)

16.7.6.1 Reserved—Bits 7–5

This bit field is reserved or not implemented. It is read as 0 during Read Operations. Never modify any bits in this field by writing 1 to them.

16.7.6.2 OnCE Core Status (OS[1:0])—Bits 4–3

The OnCE Core Status (OS[1:0]) bits describe the operating status of the core. It is recommended JTAGIR for OS[1:0] information be read because OSR is unreadable in Stop mode. **Table 16-17** summarizes the OS[1:0] descriptions. On transitions from 00 to 11 and from 11 to 00, there is a small chance intermediate states (01 or 10) may be captured.

Table 16-17. Core Status Bit Description

OS[1:0]	Instruction	Description
00	Normal	Core Executing Instructions or in Reset
01	Stop/Wait	Core in Stop or Wait Mode
10	Busy	Device is Performing External or Peripheral Access (Wait States)
11	Debug	Core Halted and in Debug Mode

Note: The OS bits are also captured by the JTAG Instruction Register (IR). See **Section 16.13.3** for details.

16.7.6.3 Trace Occurrence (TO)—Bit 2

The *read-only* Trace Occurrence (TO) status bit is set when a Trace event occurs. This bit is cleared by hardware reset if ENABLE_ONCE is not decoded in the JTAGIR and also by event rearm conditions described in **Section 16.7.5.5.**

16.7.6.4 Hardware Breakpoint Occurrence (HBO)—Bit 1

The *read-only* Hardware Breakpoint Occurrence (HBO) status bit is set when a OnCE hardware breakpoint event occurs. This bit is cleared by hardware reset if ENABLE_ONCE is not decoded in the JTAGIR, and also by the event rearm conditions described in **Section 16.7.5.5.** Also see **Section 16.7.5.2** to determine which encodings are defined to generate hardware breakpoint events.

16.7.6.5 Software Breakpoint Occurrence (SBO)—Bit 0

The *read-only* Software Breakpoint Occurrence (SBO) status bit is set when a debug instruction is executed, a software breakpoint event, for example, except when PWD =1 in the OCR. The SBO bit is cleared by hardware reset, provided ENABLE_ONCE is not decoded in the JTAGIR. It is also cleared by the event rearm conditions described in **Section 16.7.5.5.** The EM[1:0] bits determine if the core or the FIFO is halted.

16.8 Breakpoint and Trace Registers

The following subsections describe these OnCE Breakpoint and Trace registers:

- OnCE Breakpoint/Trace Counter (OCNTR) register
- OnCE Memory Address Latch (OMAL) *(not memory mapped)*
- OnCE Breakpoint Address Register (OBAR)
- OnCE Memory Address Comparator (OMAC) *(not memory mapped)*

16.8.1 OnCE Breakpoint/Trace Counter Register (OCNTR)

The OnCE Breakpoint/Trace Counter Register (OCNTR) is an 8-bit counter permitting as many as 256 valid address comparisons or instruction executions. The process depends on whether it is configured as a Breakpoint or Trace counter and occurs before a OnCE event. In its most common use, OCNTR is $00 and the first valid address compare or instruction execution halts the core. If the user prefers to generate a OnCE event on *n* valid address compares or *n* instructions, OCNTR is loaded with *n* – 1. Again, if Trace mode is selected and EM[1:0] is not cleared, only *n* – 1 instructions must execute before an event occurs.

OCNTR—$03	7	6	5	4	3	2	1	0
Read Write	Eight-Bit Event Counter							
Reset	0	0	0	0	0	0	0	0

Figure 16-4. OnCE Breakpoint/Trace Counter (OCNTR)

When used as a breakpoint counter, the OCNTR becomes a powerful tool to debug real-time interrupt sequences such as servicing an A/D or D/A converter or stopping after a specific number of transfers from a peripheral have occurred. OCNTR is cleared by hardware reset provided ENABLE_ONCE is not decoded in the JTAGIR.

Used as a Trace mode counter, the OCNTR is a single-step through code. Meaning after each instruction is executed, reenter Debug mode, allowing the processor state to display after each instruction. Place larger values in OCNTR, then multiple instructions are executed at full core speed before reentering Debug mode. Trace mode is most useful when the EM bits are set for Debug mode entry. However, halting the FIFO or auto rearm the events with a $\overline{\text{DE}}$ toggle is also possible. The trace feature helps debug sections of code without a normal flow, or debug codes getting hung up in infinite loops. Using the Trace counter also permits time critical areas of code to be debugged.

It is important to note the breakpoint/trace logic is only enabled for instructions executed *outside* of Debug mode. Instructions forced into the pipeline via the OnCE module do not cause trace or breakpoint events.

16.8.2 OnCE Memory Address Latch Register (OMAL)

The OnCE Memory Address Latch (OMAL) register is a 16-bit register latching the PAB or XAB1 on every cycle. Latching is disabled if the OnCE module is powered-down with the OCR's PWD bit. This is *not* a memory mapped nor a read/write register.

Note: The OMAL register does not latch the XAB2 bus. As a result, it is not possible to set an address breakpoint on any access completed on the XAB2/XDB2 bus pair used for the second read in any dual-read instruction.

16.8.3 OnCE Breakpoint Address Register (OBAR)

Memory breakpoint addresses are stored in the OnCE Breakpoint Address Register (OBAR), a 16-bit OnCE register. OBAR is available for write operations only through the JTAG/OnCE serial interface. Before enabling breakpoints, write to OCR, OBAR with its proper value. OBAR is for Breakpoint 1 only. OBAR has no effect on Breakpoint 2.

OBAR-$04	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write Only	16-bit Breakpoint Address Register (Program or Data Memory Breakpoints)															
Reset																

Figure 16-18. OnCE Breakpoint Address Register (OBAR)

OBAR-$05	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	16-bit Breakpoint Address Register (Address or Data Breakpoints)															
Reset																

Figure 16-19. OnCE Breakpoint Address Register 2 (OBAR2)

16.8.4 OnCE Memory Address Comparator (OMAC)

The OnCE Memory Address Comparator (OMAC) is a 16-bit comparator designed to compare the current memory address using a memory OBAR. It is stored by OMAL. If OMAC is equal to OMAL, then the comparator delivers a signal indicating the breakpoint address has been reached. This is *not* a memory mapped, nor a read/write register.

16.8.5 OnCE Breakpoint and Trace Section

Two capabilities useful for real-time debugging of embedded control applications are address breakpoints and full-speed instruction tracing. Traditionally, processors had set a breakpoint in program memory. Those processors were set by replacing the instruction at the breakpoint address with an illegal instruction, thus causing a breakpoint exception. This technique is limiting

because breakpoints can only be set in RAM at the beginning of an opcode and not on an operand. Additionally, this technique does not permit breakpoints to be set on data memory locations.

Instead, the 56F826 provides on-chip address comparison hardware for setting Breakpoints On-Program or Data Memory accesses. This grants breakpoints to be set on Program ROM as well as Program RAM locations. Breakpoints can be programmed for reads, writes, program fetches, or memory accesses using the OCR's BS and BE bits. See **Section 16.7.5.7.**

The breakpoint logic can be enabled for the following:

- Program instruction fetches
- Program memory accesses via the MOVE (M) instruction (read, write, or access)
- X data memory accesses (read, write, or access)
- On-chip peripheral register accesses (read, write, or access)
- On either of two program memory breakpoints (i.e., on either of two instructions)
- On a single bit or field of bits in a data value at a particular address in data memory
- On a program memory or data memory location (on XAB1)
- On a sequence of two breakpoints

Breakpoints are also possible during on-chip peripheral register accesses because these are implemented as memory-mapped registers in the X data space.

Further, the 56F826 OnCE module provides a full-speed tracing capability, the capability to execute as many as 256 instructions at full speed before reentering the debug processing state, or simply generate a OnCE event. This permits a single-step program in the simplest case, when the counter is set for one occurrence, or it can execute many instructions at full speed, all before returning to Debug mode.

Breakpoint logic and trace logic have been designed to work together. Together, they make possible creation of more sophisticated trigger conditions and combine as many as two breakpoints and trace logic. Individual events and conditions leading to triggering can also be modified.

While debugging, sometimes not enough breakpoints are available. In this case, the core-based debug instruction can be substituted for the desired breakpoint location. The JTAG instruction DEBUG_REQUEST (0111) can be used to force Debug mode.

16.9 Pipeline Registers

The OnCE module provides a halt capability of the core on any instruction boundary. Upon halting the core, instructions can be executed from Debug mode, providing access to on-chip

memory and registers. These register values can be brought out through the OnCE module by executing a sequence of instructions, moving values to Peripheral Global Data Bus (PGDB), followed by OnCE commands to read the OPGDBR. This register is detailed in **Section 16.9.6.**

Executing instructions from Debug mode destroys the pipeline information. The OnCE module provides a means to preserve the pipeline when both entering and exiting Debug mode.

A restricted set of one- and two-word instructions can be executed from Debug mode. Three-word instructions cannot be forced into the pipeline. But the pipeline can be restored regardless whether the next instruction to be executed is one, two, or three words long.

The OnCE module provides the following pipeline registers:

- OnCE PAB Fetch Register (OPABFR)
- OnCE PAB Decode Register (OPABDR)
- OnCE PAB Execute Register (OPABER)
- OnCE PGDB Register (OPDBR)
- OnCE PGDB Register (OPGDBR)
- OnCE PAB Change-of-Flow FIFO Register (OPFIFO) *(not memory mapped)*

16.9.1 OnCE PAB Fetch Register (OPABFR)

OnCE PAB Fetch Register (OPABFR) is *read-only*. The 16-bit latch stores the address of the last fetched instruction before entering Debug mode. It holds both opcode and operand addresses. OPABFR is available for read operations only through the serial interface. This register is not affected by the operations performed during Debug mode.

OPABFRC	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-bit Program Fetch Address															
Reset																

Figure 16-20. OnCE PAB Fetch Register (OPABFR)

16.9.2 OnCE PAB Decode Register (OPABDR)

The 16-bit OnCE PAB Decode Register (OPABDR) stores the opcode address of the instruction currently in the instruction latch, the Program Counter (PC) value. This instruction is decoded if the chip is not entered in Debug mode. OPABDR is available for *read-only* operations through the JTAG/OnCE port. This register is *not* affected by the operations performed during Debug mode.

OPABDR-$13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Program Decode Address															
Reset																

Figure 16-21. OnCE PAB Decode Register (OPABDR)

16.9.3 OnCE PAB Execute Register (OPABER)

The 16-bit OnCE PAB Execute Register (OPABER) stores the opcode address of the last instruction executed before entering Debug mode. OPABER is available for read operations only through the JTAG/OnCE port. This register is not affected by operations performed during Debug mode.

OPABER-$10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-Only	16-Bit Program Execute Address															
Reset																

Figure 16-5. OnCE PAB Execute Register (OPABER)

16.9.4 OnCE PAB Change-of-Flow FIFO (OPFIFO)

The OnCE PAB Change-of-Flow FIFO (OPFIFO) register consists of multiple 16-bit register locations, though all locations are accessed through the same address as RS[4:0] in the OCMDR. The registers are serially available as read to the command controller through their common OPFIFO address. The OPFIFO is not affected by the operations performed during Debug mode, except for the shifting performed after reading an OPFIFO value. This register is *not* memory mapped nor can bits be modified or read.

16.9.5 OnCE PDB Register (OPDBR)

The OnCE PDB Register (OPDBR) is a read/write register, 16-bit latch capable of storing the value of the Program Data Bus (PDB). The PDB is generated by the last program memory access of the before Debug mode is entered. OPDBR is available only for read/write operations through the JTAG/OnCE serial interface and only when the chip is in Debug mode. Any attempted read of OPDBR when the chip is not in Debug mode results in the JTAG shifter capturing and shifting unspecified data. Similarly, any attempted write has no effect.

OPDBR-$09	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	16-Bit Register Used to Execute Instructions in Debug Mode and Restore Pipeline Upon Exit															
Write																
Reset																

Figure 16-22. OnCE PDB Register (OPDBR)

Immediately upon entering Debug mode, OPDBR should be read out via a OnCE command by selecting OPDBR for read. This value must then be saved externally if the pipeline is to be restored, as is typically the case. Any OPGDBR access corrupts PDB, so if OPDBR is to be saved, it must be saved before OPGDBR read/writes.

To restore the pipeline, regardless where Debug mode was entered in the instruction flow, the same sequence must take place:

- First, the value read out of OPBDR upon entry to Debug mode is written back to OPDBR with GO = EX = 0
- Next, that same value is written again to OPDBR, this time with GO = EX = 1

The first write is necessary to restore PAB. The old PAB value is saved in the FIFO and restored on the first write. It is not restored directly by the user. The second write actually restores OPDBR and the GO = EX = 1 restarts the core.

To force execution of a one-word instruction from Debug mode, write the OPDBR with the opcode of the instruction to be executed and set GO =1 and EX =1. The instruction then implements while executing, OS[1:0] = 00. Upon completion, OS[1:0] =11, Debug mode. Poll JTAGIR to determine if the instruction has completed. The period of time OS[1:0] = 00 is typically unnoticeably small. By the time JTAGIR polls status and is read, OS[1:0] = 11. The only time this is not true is on chips with mechanisms extending wait states infinitely, for example: transfer acknowledge pins. In that case, polling is necessary. Only a restricted set of one-word instructions can be executed from Debug mode.

To force execution of a two word instruction from Debug mode:

1. Write the OPDBR with the opcode of the instruction to be executed and set GO = EX = 0
2. Write OPDBR with the operand with GO = 1 and EX = 0

The instruction then executes. As in the one-word case, JTAGIR should be polled for status. Only a restricted set of two-word instructions can be executed from Debug mode.

The set of supported instructions for execution from Debug mode, GO but not EX are:

- JMP #xxxx
- MOVE #xxxx,register

- MOVE register,x:$ffff
- MOVE register,register
- MOVE register,x:(r0)+
- MOVE x:(r0)+,register
- MOVE register,p:(r0)+
- MOVE p:(r0)+,register

Note: *r0* can be any of the *r* registers. Execution of other instructions is possible, but only the above are specified and supported. Three word instructions cannot be executed from Debug mode.

16.9.6 OnCE PGDB Register (OPGDBR)

The OnCE PGDB Register (OPGDBR) is a *read-only*, 16-bit latch stowing the value of the Global Data Bus (GDB) upon entry into Debug mode. The OPGDBR is available for read operations only through the serial interface. The OPGDBR is required as a means of passing information between the chip and the command controller. It is typically used by executing a move reg, X:$FFFF from Debug mode. The value in the register is read out onto PGDB. The value can then be accessed via a OnCE command selecting the OPGDBR for a read.

OPGDBR-$08	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Only	16-Bit Register Used Read Registers and Memory Values From the Core															
Reset																

Figure 16-23. OnCE PDGB Register (OPGDBR)

The OPGDBR is a temporary storage register where the last value written to the PGDB is found. Executing the Move register, X:$FFFF loads the OPGDBR with the correct value. When the next instruction is executed, modifying the PGDB after it has executed the Move register, X:$FFFF instruction, the OPGDBR holds the newly modified PGDB value. An example of a modifying instruction of the PGDB bus is instruction writing to any of the peripheral memory-mapped registers on a chip.

The OPGDBR is available for read operations only through the JTAG/OnCE serial interface, but only when the chip is in Debug mode. Any attempted read of the OPGDBR when the chip is not in Debug mode results in the JTAG shifter capturing and shifting unspecified data.

Note: The OPGDBR accesses corrupt PDB. Therefore, if there is a need to save the value on PDB, an OPDBR read should be executed before the first OPGDBR access in any debug session.

16.9.7 OnCE FIFO History Buffer

To aid the debugging activity and keep track of the program flow, a *read-only* FIFO buffer is provided. The FIFO stores PAB values from the instruction flow. The FIFO consists of Fetch, Decode, and Execute registers as well as an optional, or peripheral, change-of-flow FIFO. **Figure 16-24** illustrates a block diagram of the OnCE FIFO history buffer.

The following instructions are considered to be change-of-flow:

- BRA
- JMP
- Bcc (with condition true)
- Jcc (with condition true)
- BRSET
- BRCLR
- RTS
- RTI
- JSR

Note: Addresses of JSR instructions at interrupt vector locations are not stored in the change-of-flow FIFO.

When one of the listed instructions is executed, its opcode address is immediately placed in the top location of the change-of-flow FIFO, as well as being placed in OPABER. Previous addresses placed in the OPFIFO are shifted down one location and the oldest address is overwritten. The OPABDR holds the PC value. If the core is halted, the next opcode to be executed resides at the memory location pointed to by OPABDR.

Reads of the OPFIFO register return the oldest value in the OPFIFO first. The next read returns the next oldest. The *n*th read in an *n* deep OPFIFO returns the latest change-of-flow address. It is recommended all OPFIFO locations are read. For example, *n* reads for an *n* deep OPFIFO so the oldest-to-newest ordering is maintained when address capture resumes.

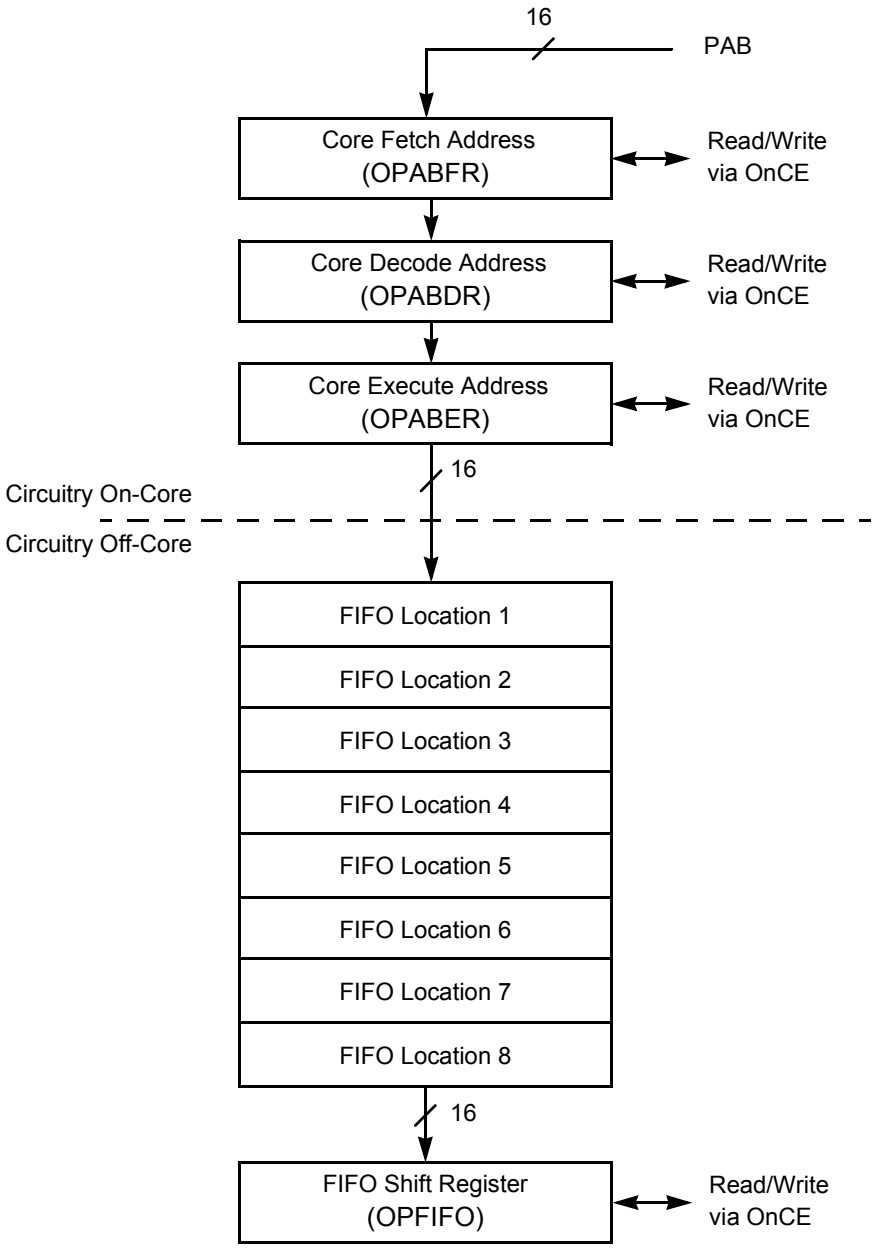


Figure 16-24. OnCE FIFO History Buffer

The change of flow nature of the FIFO begins *after* the OPABER and not the Fetch, Decode, or Execute registers. Thus, changes of flow affect only the contents of the OPFIFO.

When the OPFIFO is halted in response to setting the FH bit, PAB capture halts immediately and transfers in progress can be interrupted. For instance, while determinate values are in the registers, the values may not provide entirely coherent information regarding the recent history of program flow.

Further, the state of the OPFIFO can be different when it is halted because of an event occurring when EM = 01 than when it is halted with the core due to an event occurring when EM = 00.

16.10 Breakpoint 2 Architecture

All 56F800 chips contain a Breakpoint 1 Unit. The 56F80x provides a Breakpoint 2 Unit providing greater flexibility in setting breakpoints. Adding a Second Breakpoint greatly increases the debug capability of the device. It allows the following additional breakpoints to detect even more complex events:

- On either of two program memory breakpoints, such as on either of two instructions
- On a data value at a particular address in data memory
- On a bit or field of bits in a data value at a particular address in data memory
- On a program memory or data memory location (on XAB1)
- On a sequence of two breakpoints

Upon detecting a valid event, the OnCE module then performs one of the following actions:

- Halt the core and enter the debug processing state
- Interrupt the core
- Halt the OnCE FIFO, but let the core continue operation
- Rearm the trigger mechanism and toggle the $\overline{\text{DE}}$ pin

Breakpoint 1 Unit is used in conjunction with the Breakpoint 2 Unit for the detection of more complex trigger conditions. Breakpoint 1 Unit is the same as found on other 56F80x chips. Breakpoint 2 unit allows specifying more complex breakpoint conditions when used in conjunction with the first breakpoint. Further, it is possible to set up Breakpoint 2 for interrupts while leaving Breakpoint 1 available to the JTAG/OnCE port.

Note: When a breakpoint is determined the core global data bus with the Breakpoint 2 Unit, the breakpoint condition should be qualified by an X memory access with the Breakpoint 1 Unit.

Figure 16-25 illustrates how the two breakpoint units are combined in the Breakpoint and Tracer counter unit specifying more complex triggers to perform one of several actions upon detection of a breakpoint. In addition to simply detecting the breakpoint conditions, this unit allows the first of the two breakpoints to be qualified by the BS and BE bits found in the OnCE Control register. This process allows a breakpoint to be qualified by a read/write or access condition. The second breakpoint is unaffected by these bits and merely detects the value on the appropriate bus.

A counter is also available for detecting a specified occurrence of a breakpoint condition or for tracing a specified number of instructions.

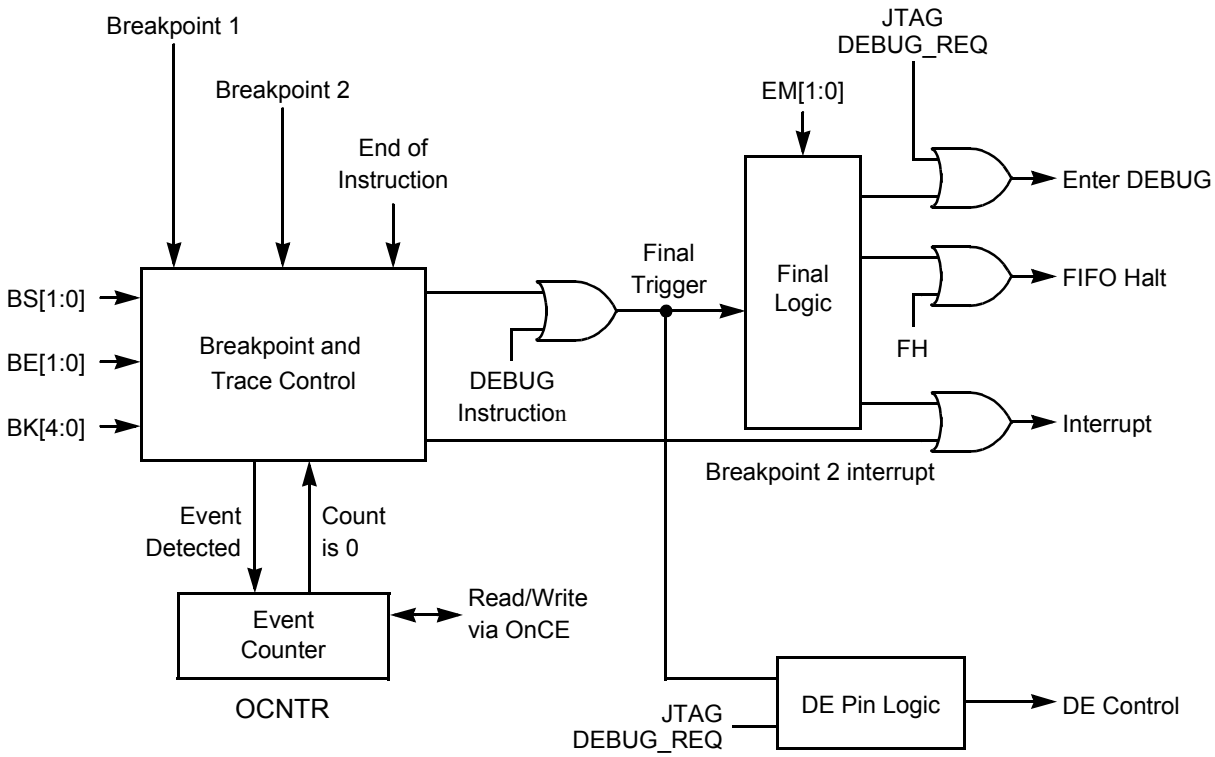


Figure 16-25. Breakpoint and Trace Counter Unit

16.11 Breakpoint Configuration

Breakpoint 1 Unit is programmed in the OnCE Control register using the BS and BE bits. Breakpoint 2 Unit is programmed by the OnCE Breakpoint Control 2 (OBCTL2) register, located within the Breakpoint 2 Unit. BK bits in the OnCE Control register specify how the two breakpoints are configured to generate trigger and interrupt conditions. However, the action performed when a final trigger is detected is specified by the EM bits in the OnCE Control register. **Figure 16-26** illustrates the breakpoint programming model for the dual breakpoint system.

Note: Registers OMAC, OMAL, OMAC2, and OMAL2 are not memory mapped and their bits cannot be modified or read.

OCR-$02	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	COP DIS	RES	BK4	BK3	BK2	BK1	BK0	RES	FH	EM1	EM0	PWD	BS1	BS0	BE1	BE0
Reset																

OnCE Control Register

OCR-$02	2	1	0
Read / Write	EN	INV	DAT
Reset	0		

OnCE Breakpoint Control Register 2

Figure 16-26. OnCE Breakpoint Programming Model

Circuitry of Breakpoint 1 Unit contains:

- OMAL
- OBAR
- OMAC
- OCNTR.

An address comparator, OMAC and OBAR, its associated breakpoint address register are useful in halting a program at a specific point for the purpose to examine or change registers or memory. OMAC permits setting breakpoints in RAM or ROM while in any operating mode. The OBAR is dedicated to Breakpoint 1 Logic. **Figure 16-27** illustrates a block diagram of the Breakpoint 1 Unit.

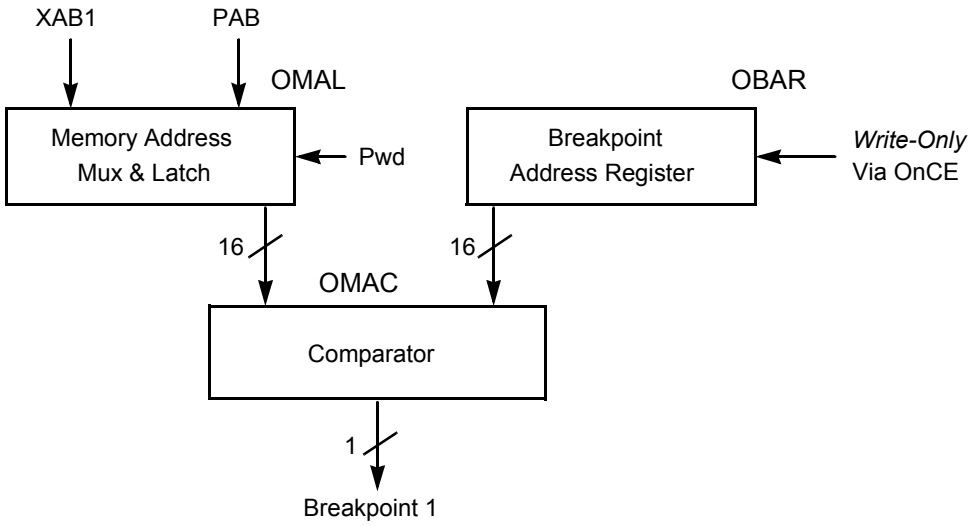


Figure 16-27. Breakpoint 1 Unit

A valid address comparison for Breakpoint 1 is defined as:

> The value in OBAR matches the value on PAB or XAB1 while meeting the breakpoint conditions specified by the BE/BS bit combination.

A valid address comparison for Breakpoint 1 can perform other tasks based on BK encodings and the state of OCNTR.

- BK could dictate the first valid address comparison enabling Trace decreasing OCNTR
- BK could dictate valid address comparison directly decreasing OCNTR
- If OCNTR = 0 because of previous decreases or a direct OCNTR write, one more valid address compare can be set, generating hardware breakpoint event

In this case, HBO is set and the $\overline{DE}$ pin is asserted. When activated, and the EM bits determine whether to halt the core or just the FIFO.

Figure 16-28 provides a block diagram of the Breakpoint 2 Unit. This unit also has its own address register OBAR2, similar to OBAR of Breakpoint 1. Address comparator OMAC2 is similar to OMAC of Breakpoint 1.

If BE = 00, Breakpoint 2 Unit is disabled. Other BE encodings and all BS encodings refer only to Breakpoint 1 functionality. The BK encoding selects which breakpoint unit or combination of units, and/or decrease OCNTR. The Breakpoint 2 Unit operates in a similar manner. A valid Breakpoint 2 address comparison occurs when the value on the PAB or CGDB matches the value in the OBAR2 for the bits selected with the OnCE Breakpoint Mask Register 2 (OBMSK2).

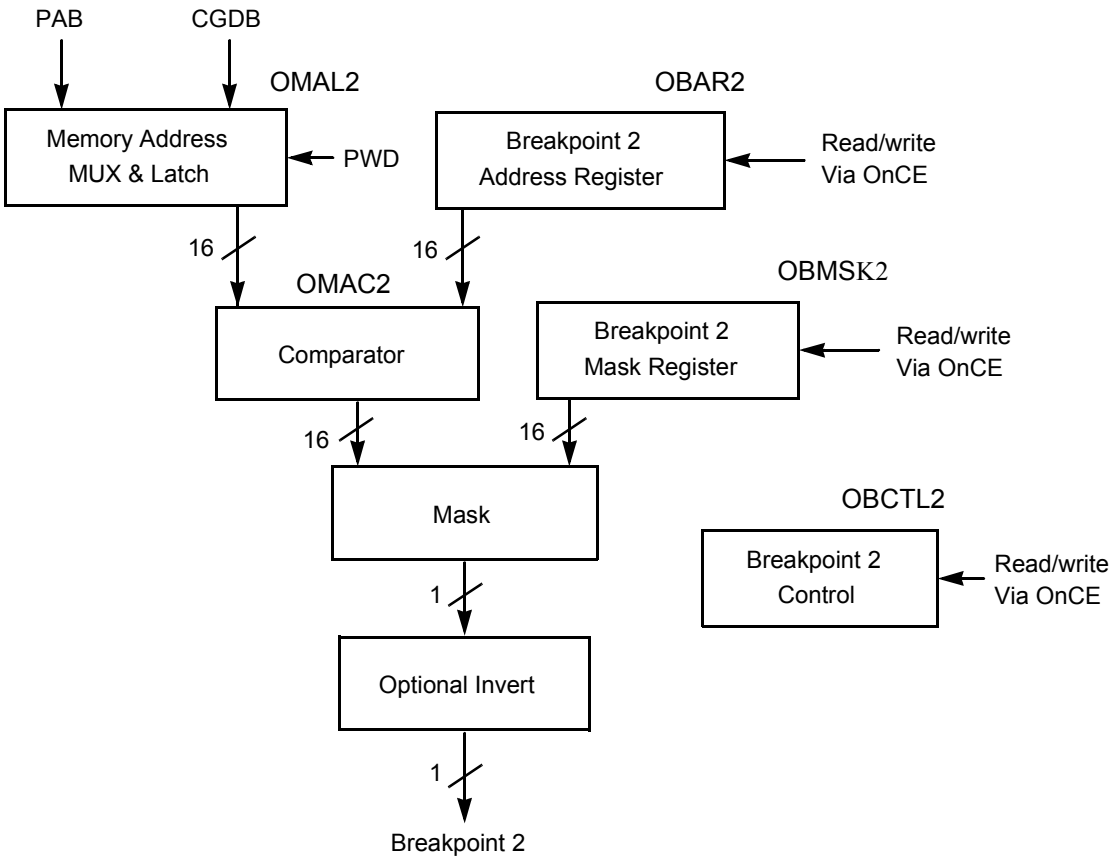


Figure 16-28. Breakpoint 2 Unit

A valid Breakpoint 2 address comparison, based on BK in the OCNTR state can preform the following:

- If OCNTR > 0, the OCNTR is decremented directly
- If OCNTR = 0, one more valid address compare can generate an HBO event
- Trigger Breakpoint 1 valid address compares to begin decrementing the OCNTR
- When OCNTR = 0, trigger Breakpoint 1 valid address compares to generate HBO event
- Generate a OnCE module interrupt. This is not considered an event, since no event flag is set. It is useful for Programmable ROM (PROM) code patching.
- Trigger trace to decrement the OCNTR
- Trigger the first valid address compare on Breakpoint 1 to allow trace to decrement the OCNTR

Note: When a breakpoint is established on the Core Global Data Bus with this unit, the breakpoint condition is qualified by an X memory access with the first breakpoint unit.

The BE/BS bits conditions may be define determining a valid breakpoint. Using these bits, breakpoints could be restricted to occur on first data memory reads or only on fetched executed

instructions. Again, these bits pertain only to Breakpoint 1. Please refer to **Section 16.7.5.7** and **Section 16.7.5.8** to understand various encodings.

Perhaps the most important breakpoint capability is to eliminate two program memory locations. Those memory locations would be a sequence where first one breakpoint is found and it is followed sequentially by a second breakpoint. Both breakpoints would be on a specified data memory location when a programmed value is read/written as data to that location. Further, the capabilities of the breakpoint would be on a specified data memory location where a programmed value is detected only at masked bits in a data value. Upon detecting a valid event, the OnCE module then performs one of the following four actions as is currently available in the OnCE module:

1. Halt the core and enter the debug processing state
2. Interrupt the core
3. Halt the OnCE FIFO, but let the core continue operation
4. Rearm the trigger mechanism (and toggle the $\overline{DE}$ pin)

If a breakpoint is set on the last instruction in a DO loop, even if BS = 00, BE = 10, a breakpoint match occurs during the execution of the DO instruction, as well as during the execution of the instruction at the end of the DO loop.

16.11.1 Programming the Breakpoints

Breakpoints and Trace can be configured while the core is executing instructions or while the core is in reset. Complete access to the Breakpoint Logic, OCNTR, OBAR, and OCR, is provided during these operating conditions. The chip may be held in reset, set OCNTR, OBAR, and OCR so Debug mode is entered on a specific condition, release reset, and debug the application. Similarly, the application can be running while the user configures breakpoints to toggle $\overline{DE}$ on each data memory access to a certain location, thereby allowing statistical information to be gathered.

Generally, to set up a breakpoint, follow this sequence while in Debug mode:

1. JTAG must be decoding ENABLE_ONCE, allowing OnCE module register to read/write.
2. Clear the PWD bit in the OCR to power-up the OnCE module.
3. The BE[1:0] bits in the OnCE Control register are set to 00.
4. Write the breakpoint address into the OBAR.
5. Write $n - 1$ into the OnCE Count register, where n is the number of valid address comparisons and before generating an OnCE event.

6. Write the OnCE Control register to set the BE, BS, and BK bits for the desired breakpoint conditions.
7. EM will determine the outcome when an event occurs.

Exit Debug mode to restart the core. However, when the above steps are completed in User mode, the breakpoint is immediately set.

Note: OnCE events can occur even if ENABLE_ONCE is not latched in the JTAGIR. This is useful in multiprocessor applications.

The first breakpoint unit is programmed in the OnCE Count register, using the BS and BE bits. The second breakpoint unit is programmed by the OnCE Breakpoint Control 2 (OBCTL2) register. This register is located within the second breakpoint unit. These two breakpoints are set up in a manner to generate triggers and interrupt conditions is specified by the BK bits in the OnCE Count register. The EM bits in the OnCE Count register specifies the action performed when a final trigger is detected.

16.11.2 OnCE Trace Logic Operation

The Trace and Breakpoint Logics are tightly coupled, sharing resources where necessary. When BK[4:0] = 10111, OCNTR is decreased each time an instruction is executed from Normal mode. Instructions executed from Debug mode do not decrement the OCNTR. The event occurrence mechanism is slightly different for Trace than for Breakpoints.

For breakpoints, the event occurs when the OnCE Count register = 0 and another valid address are compared. For Trace, the event occurs and the Trace occurrence is set when the OnCE Count register first reaches zero. If the EM bits are set for entry of Debug mode one more instruction is executed after Trace occurrence is set. Therefore, if the events are to be halted, the after executing n instructions, $n - 1$ should be placed in the OnCE Count register, much like the breakpoint case. To halt only the FIFO after n instructions, n should be placed in OnCE Count register. This is different from the Breakpoint Case and occurs because the Trace Occurrence Flag is set when the OnCE Count register first reaches zero. Trace events cannot cause OnCE interrupts, although the Trace Occurrence is set and $\overline{DE}$ is asserted, or pulled low, for this EM such as EM = 10 acts just like EM = 11 for Trace.

Since Trace events occur when the OnCE Count register reaches zero and Trace mode is enabled by one of the BK settings, rearming Trace events responds differently than rearming Breakpoint events. For example, EM = 10 and EM = 11 encodings attempt to rearm the Trace event, but since the conditions are still valid for Trace, the Trace Occurrence remains set and $\overline{DE}$ remains low. Similarly, for EM = 01, FIFO halt, an OnCE Control register write attempts to clear the Trace Occurrence, but again the flag remains set since the conditions are still valid for Trace. To clear the Trace Occurrence and capture additional FIFO values, complete the following:

1. Writing OnCE Control register, disabling Trace, FIFO begins capturing
2. Write OnCE Count register with the desired value
3. Write OnCE Control register, enabling Trace.
4. Poll for Trace Occurrence = 1

If step one is omitted, the Trace Occurrence is never reset and the FIFO does not begin capturing, because the conditions for valid Trace are still present.

Note: There are sequential Breakpoints enabling Trace mode. Trace mode operation is identical to the BK[4:0] = 10111 operation, except the HBO bit is set.

A common use of the Trace Logic is to execute a single instruction (OnCE count register =0), and then immediately returns to Debug mode. Upon returning to Debug mode, registers or memory locations can be displayed. When this process is repeated, it is possible to step through individual instructions to see their effect on the state of the processor.

16.12 The Debug Processing State

A 56F80x chip in a user application can enter any of six different processing modes:

- Reset mode
- Normal mode
- Exception mode
- Wait mode
- Stop mode
- Debug mode

The first five of these are referenced in **Chapter 7**, *Interrupts and the Processing States*, in the *DSP56800 Family Manual (DSP56800FM)*. The last processing mode, Debug mode, is described in this subsection.

Debug mode supports the on-chip emulation features of the chip. In this mode, the core is halted and set to accept OnCE commands through the JTAG port. Once the OnCE module is setup correctly, the device leaves Debug mode, returning control to the user program. The device reenters Debug mode when the previously set trigger condition occurs, provided EM = 00, OnCE events cause entry to Debug mode.

Capabilities available in Debug mode include:

- Read and write the OnCE registers
- Read the instruction FIFO
- Reset the OnCE event counter

- Execute a single instruction and return to this mode
- Execute a single instruction and exit this mode

16.12.1 OnCE Normal, Debug, and Stop Modes

The OnCE module has three operational modes:

1. Normal
2. Debug
3. Stop

Whenever a *Stop* instruction is executed by the device, the OnCE module is no longer accessible. The OnCE module remains in Normal mode except when the device enters Debug mode, or if it is in Stop mode. The OnCE module is in Debug mode whenever the device enters Debug mode. The major difference between the states is register access. The following OnCE module registers can be accessed in the Normal or Debug modes:

- OnCE Control Register (OCR)
- OnCE Status Register (OSR)
- OnCE Breakpoint and Trace Counter (OCNTR)
- OnCE Breakpoint Address Register (OBAR)
- OnCE Program Address Bus Fetch Register (OPABFR) (if FIFO halted)
- OnCE PAB Decode Register (OPABDR) (if FIFO halted)
- OnCE PAB Execute Register (OPABER) (if FIFO halted)
- OnCE PAB change-of-flow FIFO (OPFIFO) (if FIFO halted)

The following OnCE registers can only be accessed when the module is in Debug mode:

- OnCE Peripheral Global Data Bus Register (OPGDBR)
- OnCE Program Data Bus Register (OPDBR)

If a *Stop* is executed while accessing OnCE in User mode, problems may occur since few, or no internal clocks are running any longer. *This execution should be avoided*. Recognize this occurrence by capturing the OnCE Status (OS) bits in the JTAGIR Capture-Instruction Register (IR) then choose to send a DEBUG_REQUEST to bring the core out of Stop.

16.12.2 Entering Debug Mode

There are six ways to enter Debug mode:

1. JTAG DEBUG_REQUEST during hardware reset
2. JTAG DEBUG_REQUEST during Stop or Wait
3. JTAG DEBUG_REQUEST during wait states
4. Software breakpoint (DEBUG) during normal use with PWD = 0 and EM = 00
5. Trigger events, Breakpoint/Trace modes, when EM = 00
6. Execute a device instruction from Debug mode with EX = 0

16.12.2.1 JTAG DEBUG_REQUEST

The core reacts to a debug request by either of these sources in the same way. To send a JTAG DEBUG_REQUEST, the 0111 opcode must be shifted into the JTAGIR, and then the update-instruction register must be passed through. This instructs the core to halt and enter Debug mode. When the device enters Debug mode in response to these requests, the $\overline{\text{DE}}$ pin is asserted, or pulled low when it is enabled.

The JTAG/OnCE interface is accessible when $\overline{\text{RESET}}$ is asserted, provided $\overline{\text{TRST}}$ is not asserted, such as the JTAG port not being reset. DEBUG_REQUEST can be loaded into the JTAGIR while $\overline{\text{RESET}}$ is held low. If $\overline{\text{RESET}}$ is then *not disallowed*, the chip exit hardware resets directly into Debug mode. After sending the DEBUG_REQUEST instruction, poll the JTAGIR to determine whether the chip has entered Debug mode.

If the chip is in either Wait or Stop modes, either type of Debug request brings the chip out of these modes, much like an external interrupt. Leaving the Wait or Stop modes, the chip enters Debug mode. As always, it is necessary to poll the JTAGIR for status after sending the Debug request. It is important to remember the OnCE Status register cannot be polled during Stop mode because the OnCE module access is *not* allowed. However, JTAG access *is* allowed.

If the chip is in wait states, because of a non-zero value in the Bus Control Register (BCR) or transfer acknowledge *deassertion* on chips having this function, the debug request is latched and the core halts upon execution of the instruction in wait states. The period of time between a debug request and the OnCE status bits being set to 11, Debug mode is typically much shorter than the time it takes to poll status in JTAGIR, meaning OnCE Status = 11 on the first poll. The only time this is not the case is when a transfer acknowledge is generating a large or infinite number of wait states. For this reason, *polling for OS = 11 is always recommended*.

Sending a debug request when the chip is in a Normal mode results in the chip entering Debug mode as soon as the instruction currently executing finishes. Again, the JTAGIR should be polled

for status. Please refer to **Section 16.6** for summary information about using the JTAG TAP controller and its instructions.

16.12.2.2 Software Request During Normal Activity

Upon executing the DEBUG instruction, the chip enters the debug processing state provided PWD = 0 and EM = 00.

16.12.2.3 Trigger Events (Breakpoint/Trace Modes)

The 56F80x allows configuration of specific trigger events. These events can include breakpoints and Trace modes, or combinations of the two mode operations. The following conditions must occur to halt the core due to Breakpoint/Trace:

- EM = 00
- If OCNTR = 0, Breakpoints are Enabled (BE not 00), the next valid address compare causes the core to halt, provided it is not the initial enabling Breakpoint in a sequential breakpoint
- If OCNTR = 0, Trace mode is selected, one of the BK encodings with BK4 = 1, the next instruction executed causes the core to halt

16.12.2.4 Re-Entering Debug Mode with EX = 0

If an instruction is executed from Debug mode with EX = 0, Debug mode is automatically entered a second time after the instruction has completed execution. When executing the instruction, the core is not in Debug mode. The OS[1:0] bits reflect this state. This change in status is typically not observable because the core leaves and reenters Debug mode in a very short time. Still, polling for status in JTAGIR is recommended to guarantee the chip is in Debug mode.

16.12.2.5 Exiting Debug Mode

There are three ways to exit Debug mode:

- Restore the pipeline by writing original OnCE Program Data Bus Register (OPDBR) value back to OPDBR twice; first with GO = EX = 0 and last with GO = EX = 1. PAB is restored from OnCE Program Address Bus Fetch Register (OPABFR) continuing fetching from the correct address.
- Change program flow by writing JMP opcode to OPDBR with GO = EX = 0 and then writing target address to OPDBR with GO = 1, EX = 0. Next, write a NOP to OPDBR with GO = EX = 1
- Hardware reset (assertion of $\overline{\text{RESET}}$) brings the chip out of Debug mode provided DEBUG_REQUEST is not decoded in the JTAGI.

16.13 Accessing the OnCE Module

This subsection describes useful example sequences involving the JTAG/OnCE interface. The sequences are described in a hierarchical manner. Low-level sequences describe basic operations such as JTAG Instruction and Data register accesses. Building on this, the second group of sequences describe more complicated sequences, for example OnCE command entry and status polling. The final set builds further on the lower-level sequences to describe how to display core registers, set breakpoints, and change memory.

16.13.1 Primitive JTAG Sequences

The JTAG/OnCE serial protocol is identical to the protocol described in the IEEE 1149.1a-1993 Standard Test Access Port and Boundary Scan Architecture. It involves the control of four input pins:

- $\overline{\text{TRST}}$ (bidirectional)
- TDI
- TMS
- TCK

as well as the observance of one output pin:

- TDO

TDI and TDO are the serial input and output respectively. TCK is the serial clock and TMS is an input used to selectively step through the JTAG state machine. $\overline{\text{TRST}}$ is an asynchronous reset of the JTAG port.

The following descriptions refer to states in the JTAG state machine diagram described in **Figure 16-15.** Please refer to this diagram or to the IEEE 1149.1a-1993 document.

16.13.2 Entering the JTAG Test-Logic-Reset State

The Test-Logic-Reset state is the convenient starting point for primitive JTAG/OnCE module sequences. While in this state JTAG is reset, disabling TDO. No shifting is taking place and the JTAGIR is decoding the IDCODE instruction. *This state is entered only on power-up or during the initial phase of a series of OnCE module sequences*. Additionally, this state can be entered to alter JTAG to a known state. To enter the Test-Logic-Reset state on power-up, both $\overline{\text{TRST}}$ and $\overline{\text{RESET}}$ should be asserted, as shown in **Figure 16-29.** See the appropriate technical data sheet for minimum assertion pulse widths. $\overline{\text{TRST}}$ can change at any time with respect to TCK.

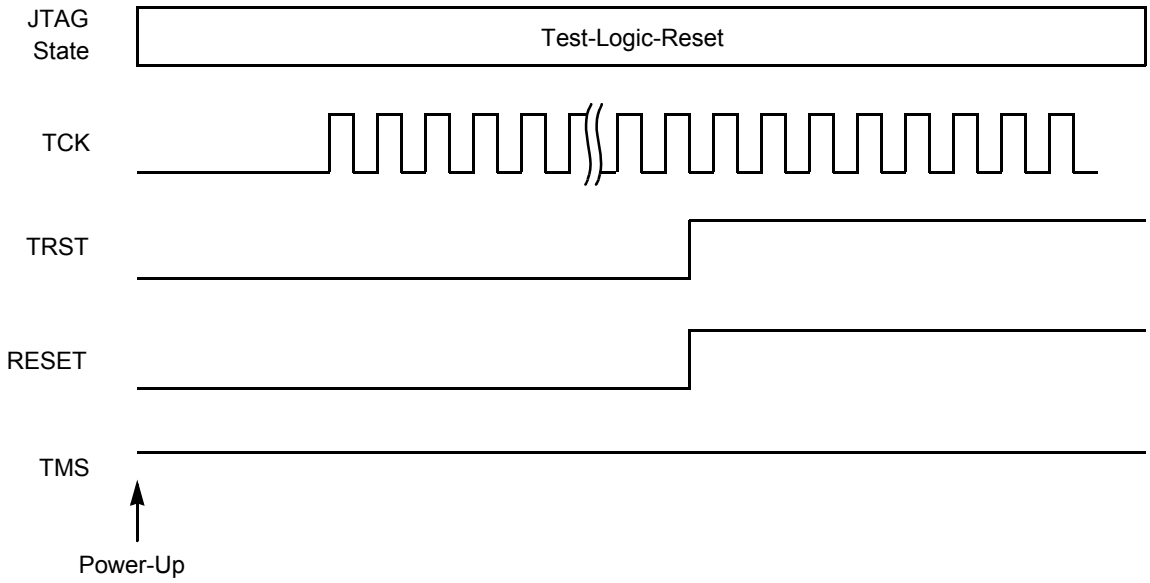


Figure 16-29. Entering the JTAG Test-Logic-Reset State

At any other time, the Test-Logic-Reset state can be entered by holding TMS high for five or more TCK pulses, as shown in **Figure 16-30.** TMS is sampled by the chip on the rising edge of TCK. To explicitly show this timing, TMS is shown to change on the falling edge of TCK. The JTAG state machine changes state on rising edges of TCK, or on $\overline{\text{TRST}}$ assertion and power-up. This sequence provides a simple way of resetting JTAG into a known state.

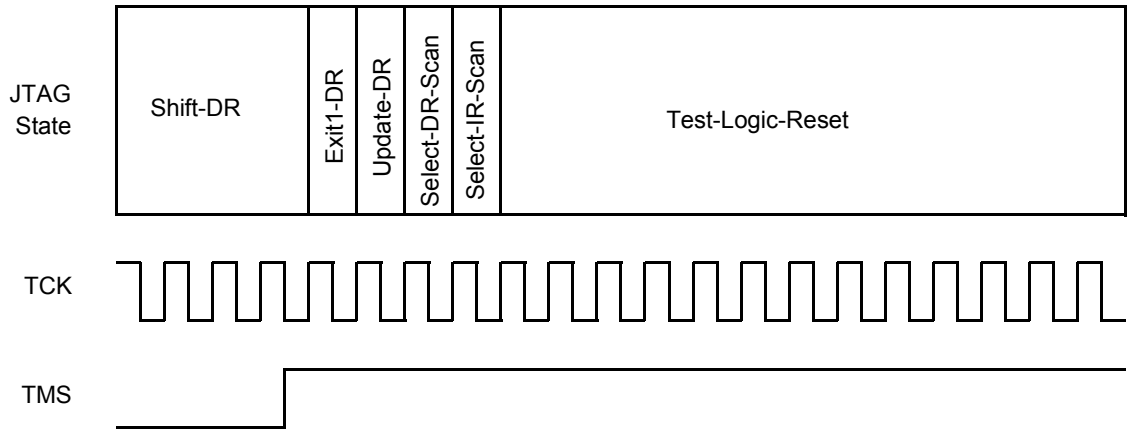


Figure 16-30. Holding TMS High to Enter Test-Logic-Reset State

16.13.3 Loading the JTAG Instruction Register

JTAG instructions are loaded through the JTAGIR path in the state machine. Shifting takes place in the Shift-IR path while the actual instruction register update occurs on Update-IR.

Note: Bit order for JTAG/OnCE shifting is always as shown in **Figure 16-31.**

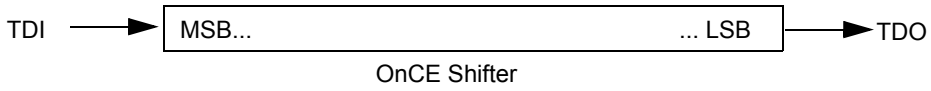


Figure 16-31. Bit Order for JTAG/OnCE Shifting

Note: OnCE instructions are loaded into the Instruction Register (IR) through the Data Register (DR) path of the state machine. JTAG instructions are loaded in the IR path.

The following sequence shown in **Figure 16-32** demonstrates how to load the instruction DEBUG_REQUEST into the JTAGIR.

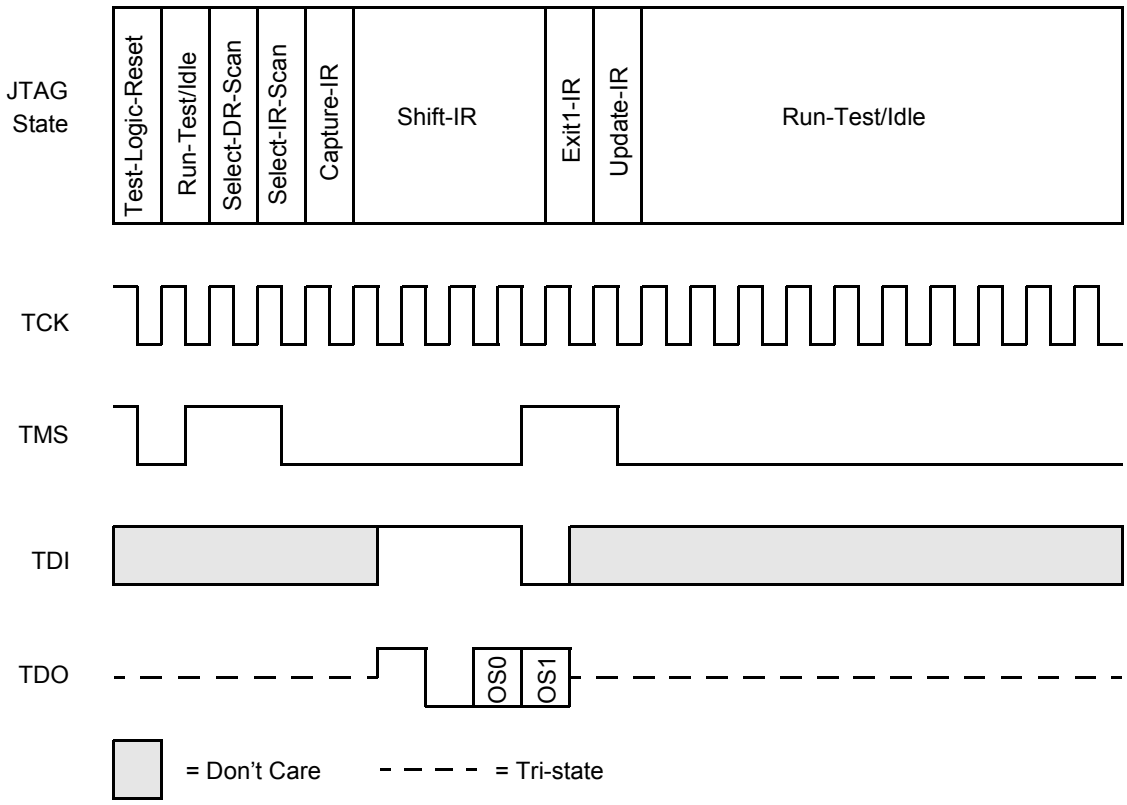


Figure 16-32. Loading DEBUG_REQUEST

During Shift-IR, a four-bit shifter is connected between TDI and TDO. The opcode shifted in is loaded into the JTAGIR on Update-IR. The data shifted out is captured on Capture-IR. TDI, like TMS, is sampled on the rising edge of TCK and changes on the falling edge of TCK. TDI is first sampled on the TCK rising edge following entry into the Shift-IR state. It is last sampled on the TCK rising edge when entering Exit1-IR. TDO changes on falling edges of TCK in Shift-IR. It switches back to tri-state on the falling edge of TCK in Exit1-IR. The first two bits shifted out of TDO are constant: first one, and then zero. The following two bits are the OnCE Status (OS[1:0]) bits.

Note: The value in OS[1:0] is shifted out whenever a new JTAG instruction is shifted in. This provides a convenient means to obtain status information.

16.13.4 Accessing a JTAG Data Register

JTAG data registers are loaded via the DR path in the state machine. Shifting takes place in the Shift-DR state and the shifter connected between TDI and TDO is selected by the instruction decoded in the JTAGIR. When applicable, data is captured in the selected register on Capture-DR. It is shifted out on Shift-DR while new data is shifted in. Finally the new data is loaded into the selected register on Update-DR.

Assume BYPASS has been loaded into the JTAGIR and the state machine is in the Run-Test/Idle state. In BYPASS, a one-bit register is selected as the Data register. The following sequence shows how data can be shifted through the BYPASS register, illustrated in **Figure 16-33.**

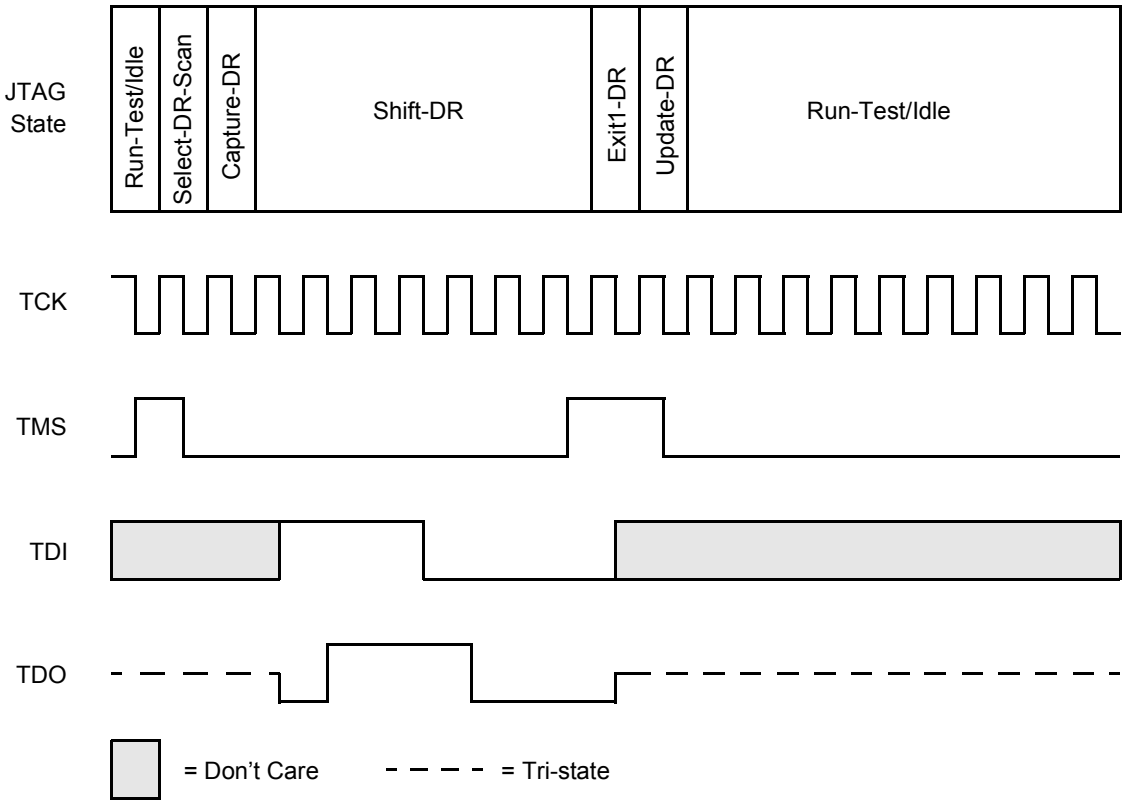


Figure 16-33. Shifting Data Through the BYPASS Register

The first bit shifted out of TDO is a constant zero because the BYPASS register captures zero on Capture-DR per the IEEE Standard. The ensuing bits are just the bits shifted into TDI delayed by one period.

16.13.4.1 JTAG/OnCE Interaction: Basic Sequences

JTAG controls the OnCE module by way of two basic JTAG instructions:

1. DEBUG_REQUEST
2. ENABLE_ONCE

DEBUG_REQUEST provides a simple way to halt the core. The halt request is latched in the OnCE module so a new JTAG instruction can be shifted in without waiting for the request to be granted. After DEBUG_REQUEST has been shifted in, JTAGIR status polling takes place, verifying if the request has been granted. This polling sequence is described in **Section 16.13.4.5.** Like any other JTAG instruction, DEBUG_REQUEST selects a data register to be connected between TDI and TDO in the DR path. The one-bit BYPASS register is selected. Values shifted into the BYPASS register have no effect on the OnCE logic.

ENABLE_ONCE is decoded in the JTAGIR for most of the time during a OnCE sequence. When ENABLE_ONCE is decoded, access to the OnCE registers is available through the DR path. Depending on which register is being accessed, the shifter connected between TDI and TDO during Shift-DR can be either eight or 16 bits long. The shifter is eight bits long for the OnCE Command Register (OCMDR) and OnCE Status Register (OSR) accesses, and 16 bits long for all other register accesses. This means if the OnCE module is expecting a command to be entered and loaded into the OCMDR, an 8-bit shifter is selected. If the OnCE command is loaded into the OCMDR has a 16-bit, read/write associated with it, a 16-bit shifter is connected between TDI and TDO during Shift-DR. The OnCE shifter selection can be understood in terms of the state diagram shown in **Figure 16-34.**

As long as ENABLE_ONCE is decoded in JTAGIR, one of the two shifters is available for shifting. If a different JTAG instruction is shifted in, the BYPASS register is selected.

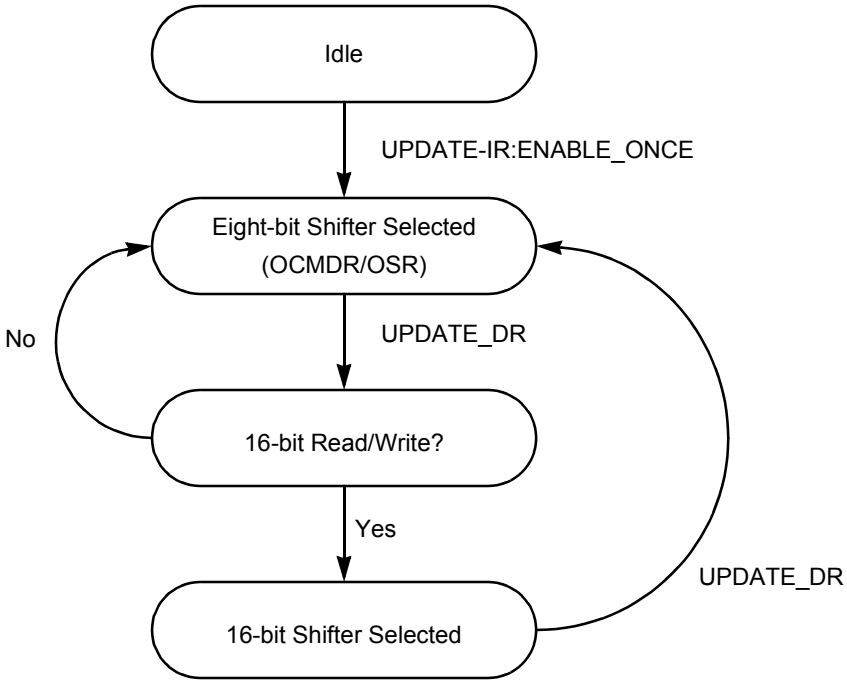


Figure 16-34. OnCE Shifter Selection State Diagram

16.13.4.2 Executing a OnCE Command by Reading the OCR

The following sequence shows how to read the OnCE Control register, assuming ENABLE_ONCE is being decoded in JTAGIR, the JTAG state machine is at Run-Test/Idle.

The DR path can not yet have been entered, meaning the OnCE module has selected the 8-bit shifter. Please refer to **Figure 16-35.**

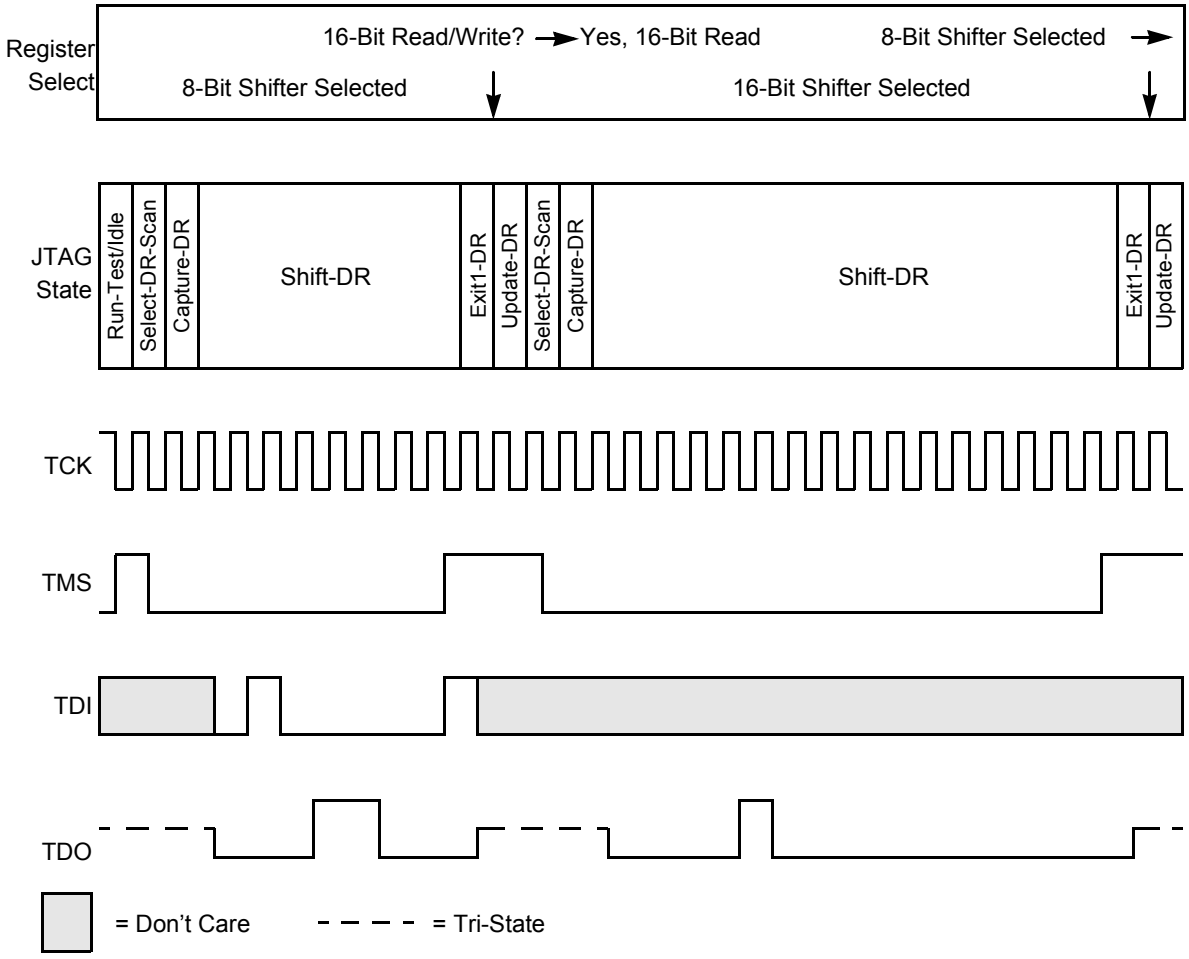


Figure 16-35. Executing a OnCE Command by Reading the OCR

In the first shift sequence, the 8-bit shifter is selected. Whenever the 8-bit shifter is selected, the OnCE Command Register is written on Update-DR, with the value shifted into TDI. In this case, $82 is shifted in. This is the OnCE opcode for a read of the OnCE Control register. Similarly, whenever the 8-bit shifter is selected, it captures the value of the OnCE Status register when passing through Capture-DR. This value is then shifted out of TDO in the ensuing shift. In this case, $18 was shifted out, indicating the device is in Debug mode.

When Update-DR is passed through in an OnCE Command Register write, the OnCE module begins decoding the opcode in that register. During command decoding the OnCE module determines whether a 16-bit shift is to occur (in this case, yes) and if so, whether it is a read or write. If it is a read, the register selected by the Register Select (RS) field in the OCMDR is

captured in the 16-bit shifter on Capture-DR. If it is a legal write, the selected register is written on Update-DR following the 16-bit shift.

16.13.4.3 Executing a OnCE Command by Writing the OCNTR

The 8-bit OCNTR is written in this sequence. First, the write OCR opcode is entered, followed by a 16-bit shift sequence even though the OCNTR is only eight bits long. If a selected register is less than 16 bits, it always reads to, or writes from, the LSB of the 16-bit shifter. The initial set-up is identical to the previous example. Please see **Figure 16-36.**

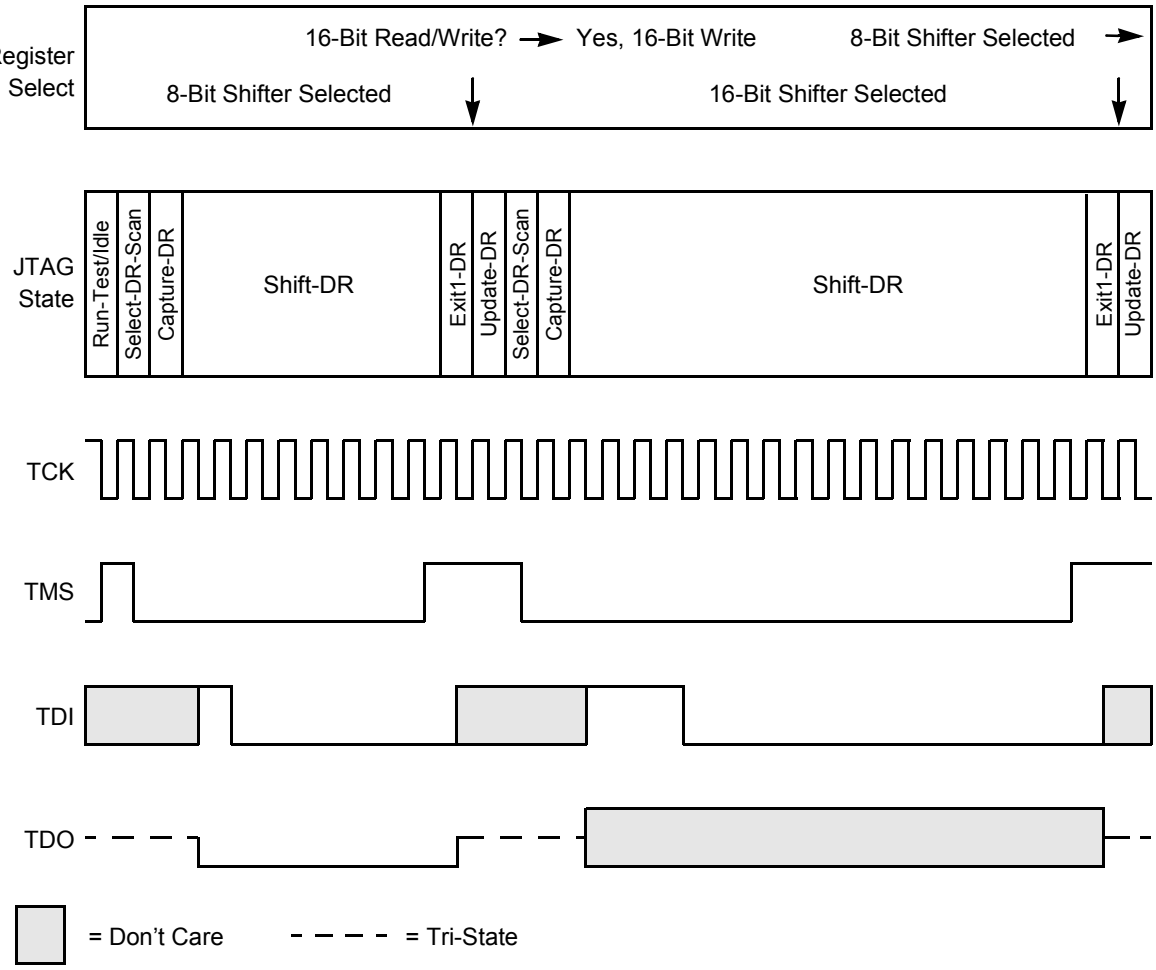


Figure 16-36. Executing a OnCE Command by Writing the OCNTR

In this sequence, $00 was read out from the OnCE Status Register, indicating the chip is in Normal mode, or the core is running. The OnCE opcode shifted in is $01, corresponding to write OnCE Count Register. In the ensuing 16-bit shift, $0003 is shifted in. Since the OnCE Count

Register is only eight bits wide, it loads the eight Least Significant Bits, or $03. The bits coming out of TDO are unspecified during 16-bit writes.

16.13.4.4 OSR Status Polling

As described in the previous examples, status information from the OnCE Status Register is made available each time a new OnCE command is shifted in. This provides a convenient means for status polling. The following sequence shows the OSR status polling. Assume a breakpoint has been set up to halt the core. Please see **Figure 16-37.**

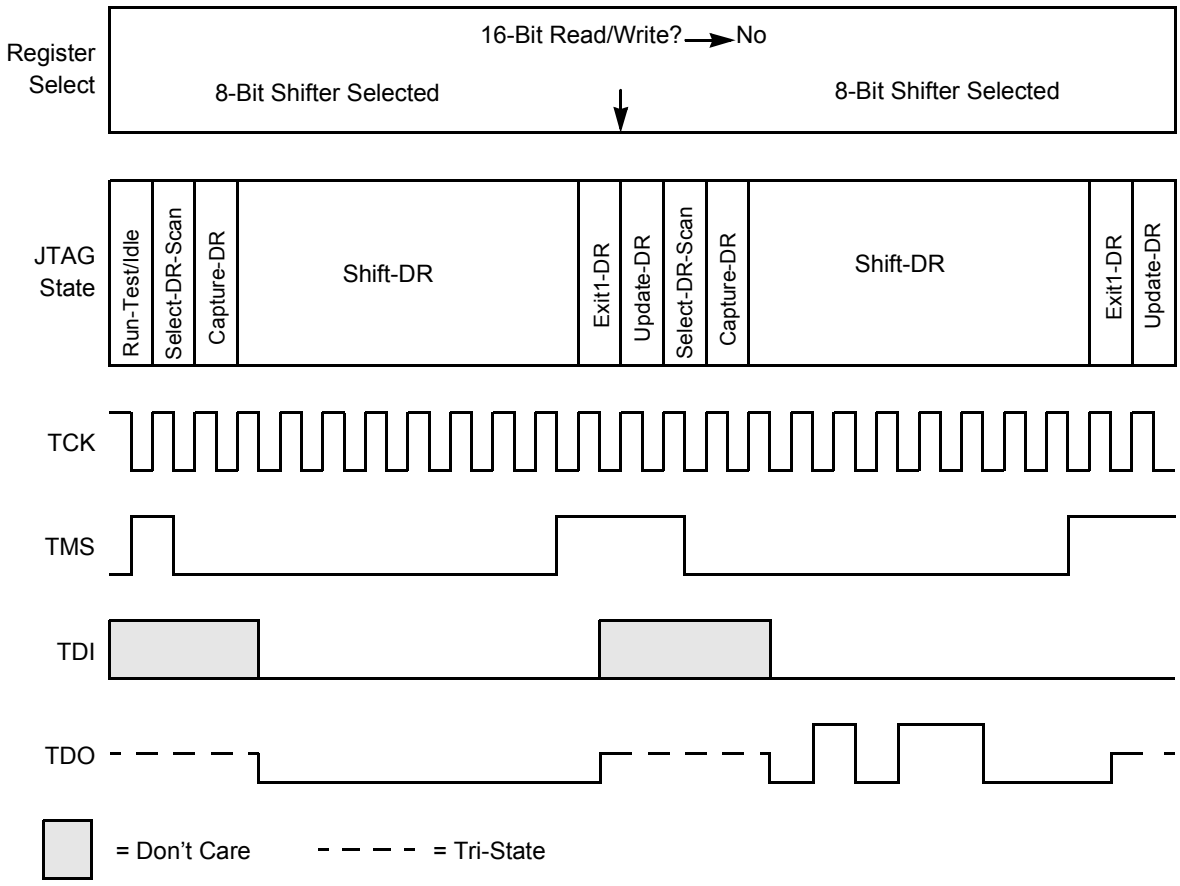


Figure 16-37. OSR Status Polling

On the first 8-bit sequence, $00 is shifted into the OnCE Command Register, corresponding to no-register selected. Next, $00 is read out from the OSR. This read-out indicates the core is still in Normal mode. The OnCE module decodes the $00 opcode and again selects the 8-bit shifter since a 16-bit access is not associated with this opcode. On the second 8-bit sequence, $1A is read from the OnCE Status register. This read indicates the chip is in Debug mode and a hardware breakpoint has occurred.

16.13.4.5 JTAGIR Status Polling

OnCE Status register status polling has a couple disadvantages.

1. The *OSR is not accessible* when the device is executing a Stop instruction. The OnCE Status register access, and all other OnCE register accesses, can continue only after the Stop state has been exited. Exit is either by an interrupt or a by DEBUG_REQUEST.
2. 8-bit shifts are required. Polling the JTAGIR provides a more efficient and more reliable means of gathering status information.

The following sequence shows how the JTAGIR can be polled. Again, assume a breakpoint has been created to halt the core. Please refer to **Figure 16-38.**

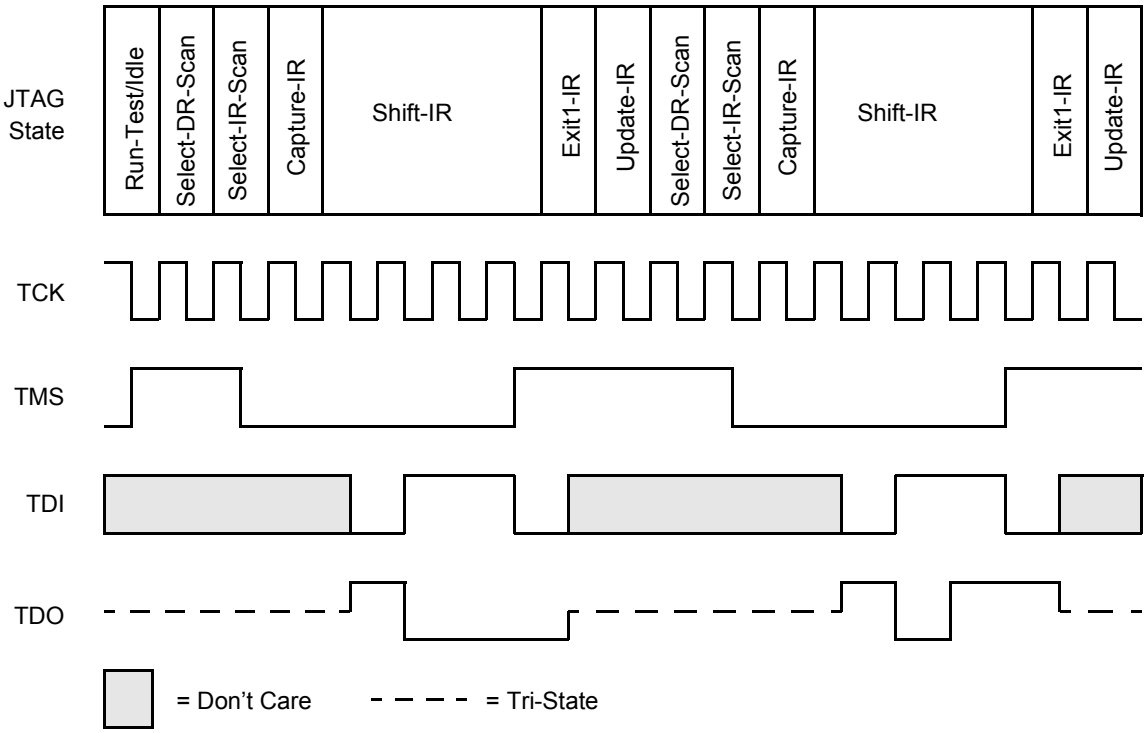


Figure 16-38. JTAGIR Status Polling

On the first 4-bit shift sequence, $6, ENABLE_ONCE, is shifted into JTAGIR. The first two bits coming out of TDO are the standard constants. The last two are Output Shifter (OS) bits. OS is 00, indicating the device is in Normal mode. On the second 4-bit shift sequence, ENABLE_ONCE is again shifted in. The OS bits are now eleven, indicating the chip is in Debug mode. ENABLE_ONCE does not have to be shifted in for JTAGIR polling. After reading proper status, the Data Register path can be entered directly for OnCE register accesses.

16.13.4.6 $\overline{\text{DE}}$ Pin Polling

The $\overline{\text{DE}}$ pin is also used to provide information about the processor state. $\overline{\text{DE}}$ goes low upon entry into Debug mode. The pin is not released until Debug mode is exited.

16.13.5 OnCE Module Low Power Operation

If OnCE module's debug capability is not required by an application, it is possible to shut off the breakpoint units and the OnCE module for low power consumption. This is achieved by setting the PWD bit in the OnCE Control register. This prevents the Breakpoint Units from latching any values for comparison.

16.13.6 Resetting the Chip Without Resetting the OnCE Unit

The device can be reset without resetting the OnCE module. This reset allows creating breakpoints on an application's final target hardware with a potential Power-On Reset circuit. The OnCE module reset is disabled using the following technique: If an ENABLE_ONCE instruction is in the JTAGIR when a hardware or COP timer reset occurs, then the OnCE module unit is *not* reset. All OnCE module registers retain their current values while all valid breakpoints remain enabled. If any other instruction is in the JTAGIR when a chip reset occurs, the OnCE module is reset. The preceding capabilities permits the following sequence, useful for setting breakpoints on final target hardware:

1. Reset the chip using the $\overline{\text{RESET}}$ pin.
2. Come out of reset and begin to execute the application code, perhaps out of program ROM if this is the location of the application code.
3. Halt the chip and enter Debug mode.
4. Program the desired breakpoint(s) and leave the ENABLE_ONCE instruction in the JTAGIR.
5. Reset the chip using the $\overline{\text{RESET}}$ pin again, but this time the OnCE module is *not* reset and the breakpoints remain valid and enabled.
6. Come out of Reset while beginning to execute the application code, perhaps out of Program ROM, if this is the location of the application code.
7. The chip then correctly triggers in the application code when the desired breakpoint condition is detected.
8. The JTAG port can be polled to detect the occurrence of this breakpoint.

Another technique for loading Breakpoints is achieved by:

1. Reset the chip by asserting both the $\overline{\text{RESET}}$ pin and the $\overline{\text{TRST}}$ pin.
2. Release the $\overline{\text{TRST}}$ pin.
3. Shift in a ENABLE_ONCE instruction into the JTAGIR.
4. Set-up the desired breakpoints.
5. Release the $\overline{\text{RESET}}$ pin.
6. Come out of Reset while beginning to execute the application code.
7. The chip then correctly triggers in the application code when the desired breakpoint condition is detected.
8. The JTAG port can be polled to detect the occurrence of this breakpoint.

Another useful sequence is to enter Debug mode directly from Reset. This sequence is approached by:

1. Reset the chip by asserting both the $\overline{\text{RESET}}$ pin and the $\overline{\text{TRST}}$ pin.
2. Release the $\overline{\text{TRST}}$ pin.
3. Shift in a DEBUG_REQUEST instruction into the JTAGIR.
4. Release the $\overline{\text{RESET}}$ pin.
5. Come out of Reset and directly enter Debug mode.

The OnCE module reset can still be forced on chip reset even if there is an ENABLE_ONCE instruction in the JTAGIR. This is accomplished by asserting the $\overline{\text{TRST}}$ pin in addition to the $\overline{\text{RESET}}$ pin, guaranteeing Reset of the OnCE module.

Chapter 17
JTAG Port

17.1 Introduction

This chapter provides further discussion about debugging and high-density circuit board testing specific to Joint Test Action Group (JTAG). Like the previous chapter, this one is dedicated to testing and debugging retention; however this chapter includes greater JTAG/OnCE detail.

56F826/827 provides board and chip-level testing capability through two on-chip modules, both accessed through the JTAG port/OnCE module interface:

- On-Chip Emulation (OnCE) module
- Test Access Port (TAP) and 16-state controller, also known as the JTAG port

The JTAG/OnCE port permits insertion of a chip into a target system while retaining debug control. This capability is especially important for devices without an external bus because it eliminates a required expensive cable to bring out the chip footprint necessary for a traditional emulator system. Additionally, the JTAG/OnCE port of the 56F826/827 can be used to program the internal Flash Memory OnCE module.

The OnCE module is a module used in hybrid controller chips to debug application software employed with the chip. The port is a separate, on-chip block, allowing non-intrusive interaction with accessibility through the pins of the JTAG interface. The OnCE module makes it possible to examine registers, memory, or on-chip peripherals' contents in a special debug environment. This avoids sacrificing any user-accessible on-chip resources to perform debugging. See **Chapter 16, OnCE Module** for details on the OnCE module implementation of the 56F826 and 827series.

The JTAG port is a dedicated user-accessible TAP compatible with the IEEE 1149.1a-1993 Standard Test Access Port and Boundary Scan Architecture. Problems associated with testing high-density circuit boards have led to the development of this proposed standard under the sponsorship of the Test Technology Committee of IEEE and the JTAG. 56F826/827 supports circuit board test strategies based on this standard.

Five dedicated pins interface with the TAP. They contain a 16-state controller. The TAP uses a boundary scan technique to test the interconnections between integrated circuits after they are assembled onto a Printed Circuit Board (PCB). A tester using boundary scans can observe and control signal levels at each component pin through a shift register placed next to each pin. This is important for testing continuity and to determine if pins are stuck at a one or zero level.

17.2 Features

- Perform boundary scan operations to test circuit board electrical continuity
- Bypass the for a given circuit board test by replacing the Boundary Scan Register (BSR) with a single-bit register

- Sample the system pins during operation and transparently shift-out the result in the BSR; pre-load values to output pins prior to invoking the EXTEST instruction
- Disable output drive to pins during circuit board testing
- Provide means of accessing the OnCE module controller and circuits to control a target system
- Query identification information, manufacturer, part number, and version from chip
- Force test data onto the outputs of an Integrated Circuit (IC) while replacing its BSR in the serial data path with single bit register
- Enable weak pull-up current device on all input signals of an IC, helping to assure deterministic test results in the presence of continuity fault during interconnect testing

Aspects of the JTAG implementation presented here are specific to the 56F826/827. For internal details and applications of the standard, refer to IEEE 1149.1a.

17.3 Pin Descriptions

As described in IEEE 1149.1a, the JTAG port requires a minimum of four pins to support TDI, TDO, TCK, and TMS signals. The 56F826/827 also uses the optional $\overline{\text{TRST}}$ input signal and $\overline{\text{DE}}$ output signal used by the OnCE module interface. The pin functions are described in **Table 17-1.**

Table 17-1. JTAG Pin Descriptions

Pin Name	Pin Description
TDI	**Test Data Input**—This input pin provides a serial input data stream to the JTAG and the OnCE module. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
TDO	**Test Data Output**—This tri-state output pin provides a serial output data stream from the JTAG and the OnCE module. It is driven in the Shift-IR and Shift-DR controller states of the JTAG state machine and changes on the falling edge of TCK.
TCK	**Test Clock Input**—This input pin provides a gated clock to synchronize the test logic and shift serial data to and from the JTAG/OnCE port. If the OnCE module is not being accessed, the maximum TCK frequency is 1/4 the maximum freqency for the 56800 core. When accessing the OnCE module through the JTAG TAP, the maximum frequency for TCK is 1/8 the maximum frequency specified for the 56800 core . The TCK pin has an on-chip pull-down resistor.
TMS	**Test Mode Select Input**—This input pin is used to sequence the JTAG TAP controller's state machine. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
$\overline{\text{TRST}}$	**Test Reset**—This input provides a reset signal to the JTAG TAP controller. The operational mode of the pin is configured by Bit 14 of the OnCE Control Register (OCR). The $\overline{\text{TRST}}$ pin has an on-chip pull-up resistor.
$\overline{\text{DE}}$	**Debug Event**—This output signal debugs events detected on a trigger condition.

17.4 JTAG Port Architecture

The TAP controller is a simple state machine used to sequence the JTAG port through its valid operations:

- Serially shift in or out of 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 OnCE module) register

Note: The JTAG port oversees the shifting of data into and out of the OnCE module through the TDI and TDO pins respectively. In this case, the shifting is guided by the same TAP controller used when shifting JTAG information.

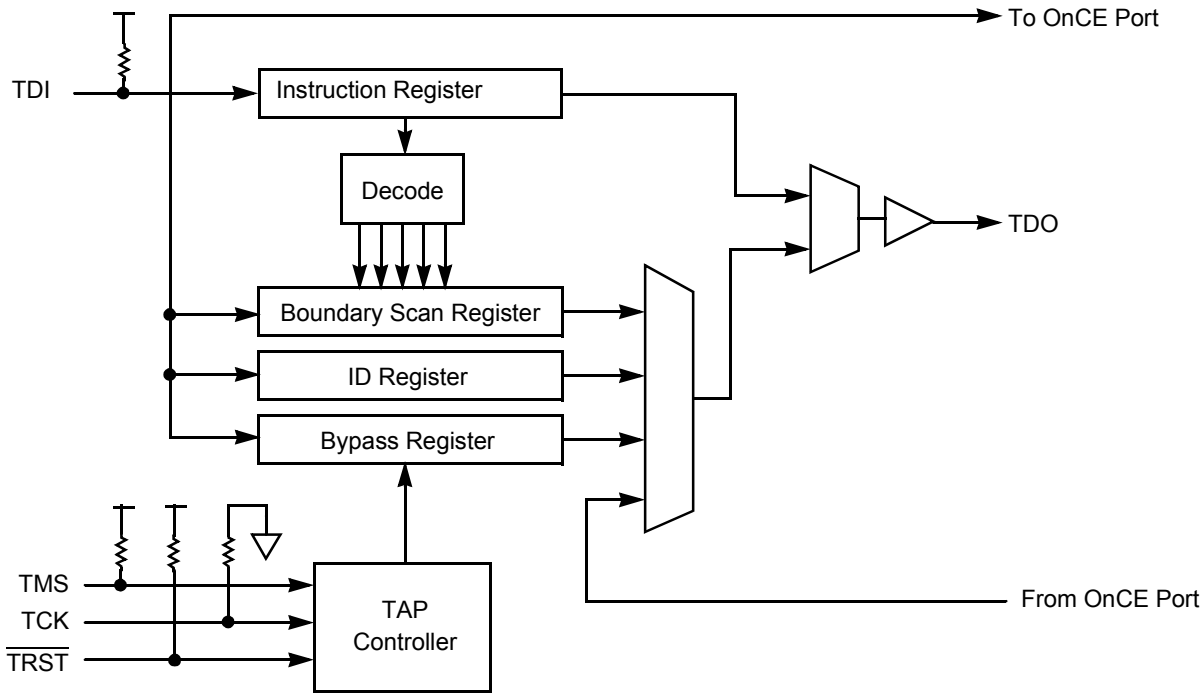


Figure 17-2. JTAG Block Diagram

A block diagram of the JTAG port is provided in **Table 17-2.** The JTAG port has four read/write registers:

1. IR
2. BSR
3. Device identification register
4. Bypass register

Access the OnCE registers described in **Chapter 16, OnCE Module.**

The TAP controller provides access to the JTAG IR through the JTAG port. The other JTAG registers must be individually selected by the JTAG IR.

17.5 Register Summary

The 56F826/827 each have four registers:

1. JTAG Instruction Register (JTAGIR)
2. Chip Identification Register (CID)
3. Boundary Scan Register (BSR)
4. JTAG Bypass Register (JTAGBR)

17.5.1 JTAG Instruction Register (JTAGIR) and Decoder

The TAP controller contains a 4-bit JTAG Instruction Register (JTAGIR). The instruction is presented to an instruction decoder during the Update-IR state. See **Section 17.6** for a description of the TAP controller operating states. The instruction decoder interprets and executes the instructions according to the conditions defined by the TAP controller state machine.

The 56F826/827 includes the three *mandatory* public instructions:

1. BYPASS, discussed in **Section 17.5.1.1**
2. EXTEST, discussed in **Section 17.5.1.2**
3. SAMPLE/PRELOAD, discussed in **Section 17.5.1.3**

The 56F826/827 includes six public instructions:

1. IDCODE, discussed in **Section 17.5.1.4**
2. EXTEST_PULLUP, discussed in **Section 17.5.1.5**
3. HIGHZ, discussed in **Section 17.5.1.6**
4. CLAMP, discussed in **Section 17.5.1.7**
5. ENABLE_ONCE, discussed in **Section 17.5.1.8**
6. DEBUG_REQUEST, discussed in **Section 17.5.1.9**

The four bits B[3:0] of the IR, decode the nine instructions, and are illustrated in **Table 17-4.** All other encodings are reserved.

JTAG Instruction Register Read/Write Reset = $2	3	2	1	0
	B3	B2	B1	B0

Figure 17-3. JTAGIR Register

CAUTION

Reserved JTAG instruction encodings *must not be used*. Hazardous operation of the chip could occur if these instructions are used.

Table 17-4. JTAGIR Encodings

B[3:0]	Instruction
0000	EXTEST
0001	SAMPLE/PRELOAD
0010	IDCODE
0011	EXTEST_PULLUP
0100	HIGHZ
0101	CLAMP
0110	ENABLE_ONCE
0111	DEBUG_REQUEST
1111	BYPASS

The JTAGIR is reset to 0010 in the test-logic-reset controller state. Therefore, the IDCODE instruction is selected on JTAG reset. In the Capture-IR state, the two Least Significant Bits (LSBs) of the instruction shift register are preset to 01, where the one is in the LSB location as required by the standard. The two Most Significant Bits (MSBs) may either capture status or be set to zero. New instructions are moved into the instruction shift register stage in Shift-IR state.

17.5.1.1 BYPASS (B[3:0] = 1111)

The BYPASS instruction enables the single-bit bypass register between TDI and TDO, illustrated in **Figure 17-5.** This instruction creates a shift register path from TDI to the bypass register and finally to TDO, circumventing the Boundary Scan Register. The BYPASS instruction is used to enhance test efficiency by shortening the overall path between TDI and TDO when no test operation of a component is required. In this instruction, the system logic is independent of the TAP. When this instruction is selected, the test logic has no effect on the operation of the on-chip system logic, as required in IEEE 1149.1-1993a.

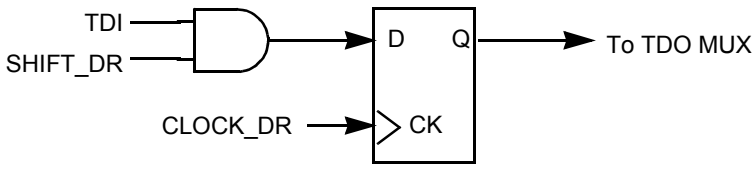


Figure 17-5. JTAGIR Bypass

17.5.1.2 EXTEST (B[3:0] = 0000)

The External Test (EXTEST) instruction enables the Boundary Scan Register between TDI and TDO, including cells for all digital device signals and associated control signals. The EXTAL pin, and any codec pins associated with analog signals, are not included in the BSR path.

In EXTEST, the BSR is capable of scanning user-defined values onto output pins, capturing values presented to input signals, and controlling the direction and value of bidirectional pins. EXTEST instruction asserts internal system reset for the system logic during its run in order to force a predictable internal state while performing external boundary scan operations.

17.5.1.3 SAMPLE/PRELOAD (B[3:0] = 0001)

The SAMPLE/PRELOAD instruction enables the BSR between TDI and TDO. When this instruction is selected, the test logic operation has no effect on the operation of the on-chip system logic. Nor does it have an effect on the flow of a signal between the system pin and the on-chip system logic, as specified by IEEE 1149.1-1993a. This instruction provides two separate functions:

1. It first provides a means to obtain a snapshot of system data and control signals (SAMPLE). The snapshot occurs on the rising edge of TCK in the Capture-DR controller state. The data can be observed by shifting it transparently through the BSR.

In a normal system configuration, many signals require external pull-ups assuring proper system operation. Consequently, the same is true for the SAMPLE/PRELOAD functionality. Data latched into the BSR during the

Capture-DR controller state may not match the drive state of the package signal if the system-requiring pull-ups are not present within the test environment.

2. The second function of the SAMPLE/PRELOAD instruction is to initialize the BSR output cells (PRELOAD) prior to selection of the CLAMP, EXTEST, or EXTEST_PULLUP instruction. This initialization ensures known data appearing on outputs when executing EXTEST. Data held in the shift register stage is transferred to the output latch on the falling edge of TCK in the Update-DR controller state. Data is not presented to the pins until the CLAMP, EXTEST, or EXTEST_PULLUP instruction is executed.

Note: Since there is no internal synchronization between the JTAG Clock (TCK) and the system Clock (CLK), some form of external synchronization to achieve meaningful results when sampling system values using the SAMPLE/PRELOAD instruction must be provided.

17.5.1.4 IDCODE (B[3:0] = 0010)

The IDCODE instruction enables the IDREGISTER between TDI and TDO. It is provided as a public instruction to allow the manufacturer, part number, and version of a component to be determined through the TAP.

When the IDCODE instruction is decoded, it selects the IDREGISTER, a 32-bit test data register. IDREGISTER loads a constant logic one into its LSB. Since the bypass register loads a logic zero at the start of a scan cycle, examination of the first bit of data shifted out of a component during a test data scan sequence immediately following exit from the test-logic-reset controller state shows whether an IDREGISTER is included in the design.

When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic, as required in IEEE 1149.1a-1993.

17.5.1.5 EXTEST_PULLUP (B[3:0] = 0011)

The EXTEST_PULLUP instruction is provided as a public instruction to aid in fault diagnoses during boundary scan testing of a circuit board. This instruction functions similarly to EXTEST, with the only difference being the presence of a weak pull-up device on all input signals. Given an appropriate charging delay, the pull-up current supplies a deterministic logic one result on an open input. When this instruction is used in board-level testing with heavily loaded nodes, it may require a charging delay greater than the two TCK periods required to transition from the Update-DR state to the Capture-DR state. Two methods of providing an increase delay are available:

1. Traverse into the run-test/idle state for extra TCK periods of charging delay, or
2. Limit the maximum TCK frequency, slowing down the TCK, so two TCK periods are adequate.

The EXTEST_PULLUP instruction asserts internal system reset for the system logic for the duration of EXTEST_PULLUP in order to force a predictable internal state while performing external boundary scan operations.

17.5.1.6 HIGHZ (B[3:0] = 0100)

The HIGHZ instruction enables the single-bit bypass register between TDI and TDO. It is provided as a public instruction in order to prevent having to drive the output signals back during circuit board testing. When the HIGHZ instruction is invoked, all output drivers are placed in an inactive-drive state. HIGHZ asserts internal system reset for the system logic for the duration of HIGHZ in order to force a predictable internal state while performing external boundary scan operations.

17.5.1.7 CLAMP (B[3:0] = 0101)

The CLAMP instruction enables the single-bit bypass register between TDI and TDO, provided as a public instruction. When the CLAMP instruction is invoked, the package output signals respond to the preconditioned values within the update latches of the Boundary Scan Register, even though the bypass register is enabled as the test data register. During circuit testing, it can be facilitated by setting up guarding signal conditions controlling the operation of logic not involved in the test, but with use of the SAMPLE/PRELOAD or EXTEST instructions. When the CLAMP instruction is executed, the state and drive of all signals remain static until a new instruction is invoked.

Features of the CLAMP instruction:

- While the signals continue to supply the guarding inputs to the in-circuit test site, the bypass mode is enabled, minimizing overall test time.
- Data in the boundary scan cell remains unchanged until a new instruction is shifted in, or the JTAG state machine is set to its reset state.
- CLAMP asserts internal system reset for the system logic for the duration of CLAMP in order to force a predictable internal state while performing external boundary scan operations.

17.5.1.8 ENABLE_ONCE (B[3:0] = 0110)

The ENABLE_ONCE instruction enables the JTAG port to communicate with the OnCE state machine and registers. It is provided as a public instruction permitting the system to perform debug functions. When the ENABLE_ONCE instruction is invoked, the TDI and TDO pins are connected directly to the OnCE registers. The particular OnCE register connected between TDI and TDO is selected by the OnCE state machine and the OnCE instruction being executed. All communication with the OnCE instruction controller is completed through the Select-DR-Scan path of the JTAG state machine. Refer to the *DSP56F80x Family Manual,* DSP56800FM, for more information.

17.5.1.9 DEBUG_REQUEST (B[3:0] = 0111)

The DEBUG_REQUEST instruction asserts a request to halt the core for entry to the Debug mode. The instruction is typically used in conjunction with ENABLE_ONCE to perform system debug functions. It is provided as a public instruction. When the DEBUG_REQUEST instruction is invoked, the TDI and TDO pins are connected to the bypass register. Refer to the *DSP56800 Family Manual* for more information.

17.5.2 JTAG Chip Identification (CID) Register

The Chip Identification (CID) Register is a 32-bit register providing a unique JTAG ID for the 56F826. It is offered as a public instruction allowing manufacturers to determine part number and version of a component through the TAP. **Figure 17-7** illustrates the CID register configuration.

ID—(IR = $2)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip Identification Regsiter	VER 3	VER 2	VER 1	VER 0	PNUM 15	PNUM 14	PNUM 13	PNUM 12	PNUM 11	PNUM 10	PNUM 9	PNUM 8	PNUM 7	PNUM 6	PNUM 5	PNUM 4
Read Only																

ID—(IR = $2)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chip Identification Regsiter	PNUM 3	PNUM 2	PNUM 1	PNUM 0	MFGID 11	MFGID 10	MFGID 9	MFGID 8	MFGID 7	MFGID 6	MFGID 5	MFGID 4	MFGID 3	MFGID 2	MFGID 1	MFGID 0
Read Only																

Figure 17-6. JTAG Chip Identification Register (CID)

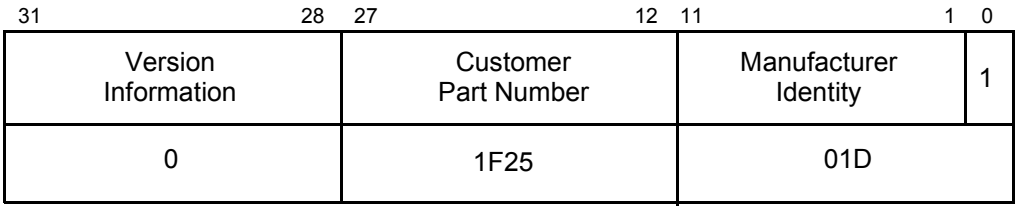


Figure 17-7. Chip Identification Register Configuration

For the initial release of the 56F826, the device ID number is $01F2501D. The version code is $0. Future revisions increment this version code. The value of the customer part number code is $1F23. **Table 17-8** provides the JTAG ID for the various versions of the chips in this series.

Table 17-8. JTAG D Codes

JTAG ID code is expressed in hex form, and is calculated as (Version, Design_Center, Part_Number, Manufacturer_ID,%1)

Part Number	Version	Design Center	JTAG ID Code Part Number	Manufacturer ID	Constant	JTAG ID Code
56F826	0	7	826	14	1	01F3A01D
56F827	0	7	827	14	1	01F3B01D

Freescale's Manufacturer Identity is 00000001110. The customer part number consists of two parts:

1. Freescale design center number, bits 27–2
2. Design center assigned sequence number, bits 21–12

The Standard Products Design Center number is 000111. Version information and design center assigned sequence number values vary depending on the current revision and implementation of a specific chip. The bit assignment for the ID code is given in **Table 17-9.**

Table 17-9. Device ID Register Bit Assignment

Bit No.	Code Use	Value for 56F80x
31–28	Version Number	0000 (For initial version only—these bits may vary)
27–22	Freescale Design Center ID	00 0111
21–12	Family and part ID	11 0010 0101
11	Freescale Manufacturer ID	0000 0000 1110
0	IEEE Requirement	Always 1

17.5.3 JTAG Boundary Scan Register (BSR)

The Boundary Scan Register (BSR) is used to examine or control the scannable pins on the 56F826. The BSR for the 56F826 contains 285 bits. The BSR for the 56F827 contains 338 bits. Each scannable pin has at least one BSR bit associated with it. **Table 17-11** illustrates the contents of the BSR for the 56F826 and 56F827.

SCAN—(IR = $0,$1,$3)	284	283	282	281	280	Pins 270 through 5	4	3	2	1	0
Boundary Scan Register											
Read-Only											

Figure 17-10. Boundary Scan Register for 56F826 (BSR)

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
1	A15—Input	Input/Output	BC_1	7	100
2	A15—Output	Input/Output	BC_1	7	100
3	A15—Output Enable	Control	BC_1	–	–
4	A15—Pull-up Enable	Control	BC_1	–	–
5	A14—Input	Input/Output	BC_1	8	99
6	A14—Output	Input/Output	BC_1	8	99
7	A14—Output Enable	Control	BC_1	–	–
8	A14—Pull-up Enable	Control	BC_1	–	–
9	A13—Input	Input/Output	BC_1	9	98
10	A13—Output	Input/Output	BC_1	9	98
11	A13—Output Enable	Control	BC_1	–	–
12	A13—Pull-up Enable	Control	BC_1	–	–
13	A12—Input	Input/Output	BC_1	10	97
14	A12—Output	Input/Output	BC_1	10	97
15	A12—Output Enable	Control	BC_1	–	–
16	A12—Pull-up Enable	Control	BC_1	–	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
17	A11—Input	Input/Output	BC_1	11	96
18	A11—Output	Input/Output	BC_1	11	96
19	A11—Output Enable	Control	BC_1	–	–
20	A11—Pull-up Enable	Control	BC_1	–	–
21	A10—Input	Input/Output	BC_1	12	95
22	A10—Output	Input/Output	BC_1	12	95
23	A10—Output Enable	Control	BC_1	–	–
24	A10—Pull-up Enable	Control	BC_1	–	–
25	A9—Input	Input/Output	BC_1	13	–
26	A9—Output	Input/Output	BC_1	13	–
27	A9—Output Enable	Control	BC_1	–	–
28	A9—Pull-up Enable	Control	BC_1	–	–
29	A8—Input	Input/Output	BC_1	14	94
30	A8—Output	Input/Output	BC_1	14	94
31	A8—Output Enable	Control	BC_1	–	–
32	A8—Pull-up Enable	Control	BC_1	–	–
33	A7—Input	Input/Output	BC_1	15	–
34	A7—Output	Input/Output	BC_1	15	–
35	A7—Output Enable	Control	BC_1	–	–
36	A7—Pull-up Enable	Control	BC_1	–	–
37	A6—Input	Input/Output	BC_1	16	–
38	A6—Output	Input/Output	BC_1	16	–
39	A6—Output Enable	Control	BC_1	–	–
40	A6—Pull-up Enable	Control	BC_1	–	–
41	A5—Input	Input/Output	BC_1	17	–
42	A5—Output	Input/Output	BC_1	17	–
43	A5—Output Enable	Control	BC_1	–	–
44	A5—Pull-up Enable	Control	BC_1	–	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
45	A4—Input	Input/Output	BC_1	18	91
46	A4—Output	Input/Output	BC_1	18	91
47	A4—Output Enable	Control	BC_1	–	–
48	A4—Pull-up Enable	Control	BC_1	–	–
49	A3—Input	Input/Output	BC_1	21	90
50	A3—Output	Input/Output	BC_1	21	90
51	A3—Output Enable	Control	BC_1	–	–
52	A3—Pull-up Enable	Control	BC_1	–	–
53	A2—Input	Input/Output	BC_1	22	89
54	A2—Output	Input/Output	BC_1	22	89
55	A2—Output Enable	Control	BC_1	–	–
56	A2—Pull-up Enable	Control	BC_1	–	–
57	A1—Input	Input/Output	BC_1	23	–
58	A1—Output	Input/Output	BC_1	23	–
59	A1—Output Enable	Control	BC_1	–	–
60	A1—Pull-up Enable	Control	BC_1	–	–
61	A0—Input	Input/Output	BC_1	24	87
62	A0—Output	Input/Output	BC_1	24	87
63	A0—Output Enable	Control	BC_1	–	–
64	A0—Pull-up Enable	Control	BC_1	–	–
65	XBOOT—Input	Input	BC_1	25	86
66	RD_B—Input	Input/Output	BC_1	26	86
67	RD_B—Output	Input/ Output	BC_1	26	–
68	RD_B—Output Enable	Control	BC_1	–	–
69	RD_B—Pull-up Enable	Control	BC_1	–	–
70	WR_B—Input	Input/Output	BC_1	27	–
71	WR_B—Output	Input/Output	BC_1	27	–
72	WR_B—Output Enable	Control	BC_1	–	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
73	WR_B—Pull-up Enable	Control	BC_1	-	85
74	DS_B—Input	Input/Output	BC_1	28	85
75	DS_B—Output	Input/ Output	BC_1	28	–
76	DS_B—Output Enable	Control	BC_1	–	–
77	DS_B—Pull-up Enable	Control	BC_1	–	–
78	PS_B—Input	Input/Output	BC_1	29	–
79	PS_B—Output	Input/Output	BC_1	29	–
80	PS_B—Output Enable	Control	BC_1	–	–
81	PS_B—Pull-up Enable	Control	BC_1	-	84
82	IREQA_B—Input	Input	BC_1	32	84
83	IREQB_B—Input	Input	BC_1	33	–
84	D0—Input	Input/Output	BC_1	34	–
85	D0—Output	Input/Output	BC_1	34	–
86	D0—Output Enable	Control	BC_1	–	–
87	D0—Pull-up Enable	Control	BC_1	–	–
88	D1—Input	Input/Output	BC_1	35	–
89	D1—Output	Input/Output	BC_1	35	–
90	D1—Output Enable	Control	BC_1	–	–
91	D1—Pull-up Enable	Control	BC_1	-	83
92	D2—Input	Input/Output	BC_1	36	83
93	D2—Output	Input/ Output	BC_1	36	–
94	D2—Output Enable	Control	BC_1	_	–
95	D2—Pull-up Enable	Control	BC_1	_	82
96	D3—Input	Input/Output	BC_1	37	82
97	D3—Output	Input/Output	BC_1	37	–
98	D3—Output Enable	Control	BC_1	–	–
99	D3—Pull-up Enable	Control	BC_1	_	–
100	D4—Input	Input/Output	BC_1	38	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
101	D4—Output	Input/Output	BC_1	38	–
102	D4—Output Enable	Control	BC_1	–	–
103	D4—Pull-up Enable	Control	BC_1	–	81
104	D5—Input	Input/Output	BC_1	39	–
105	D5—Output	Input/Output	BC_1	39	80
106	D5—Output Enable	Control	BC_1	–	–
107	D5—Pull-up Enable	Control	BC_1	–	79
108	D6—Input	Input/Output	BC_1	40	78
109	D6—Output	Input/Output	BC_1	40	77
110	D6—Output Enable	Control	BC_1	_	77
111	D6—Pull-up Enable	Control	BC_1	–	–
112	D7—Input	Input/Output	BC_1	41	–
113	D7—Output	Input/Output	BC_1	41	76
114	D7—Output Enable	Control	BC_1	_	76
115	D7—Pull-up Enable	Control	BC_1	–	–
116	D8—Input	Input/Output	BC_1	42	–
117	D8—Output	Input/Output	BC_1	42	75
118	D8—Output Enable	Control	BC_1	–	–
119	D8—Pull-up Enable	Control	BC_1	_	74
120	D9—Input	Input/Output	BC_1	43	–
121	D9—Output	Input/Output	BC_1	43	–
122	D9—Output Enable	Control	BC_1	–	–
123	D9—Pull-up Enable	Control	BC_1	–	–
124	D10—Input	Input/Output	BC_1	44	–
125	D10—Output	Input/Output	BC_1	44	73
126	D10—Output Enable	Control	BC_1	–	–
127	D10—Pull-up Enable	Control	BC_1	–	–
128	RESET_B—Input	Input	BC_1	45	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
129	D11—Input	Input/Output	BC_1	46	–
130	D11—Output	Input/Output	BC_1	46	–
131	D11—Output Enable	Control	BC_1	_	72
132	D11—Pull-up Enable	Control	BC_1	–	–
133	D12—Input	Input/Output	BC_1	47	–
134	D12—Output	Input/Output	BC_1	47	–
135	D12—Output Enable	Control	BC_1	–	–
136	D12—Pull-up Enable	Control	BC_1	–	–
137	D13—Input	Input/Output	BC_1	48	71
138	D13—Output	Input/Output	BC_1	48	–
139	D13—Output Enable	Control	BC_1	–	–
140	D13—Pull-up Enable	Control	BC_1	–	–
141	D14—Input	Input/Output	BC_1	49	–
142	D14—Output	Input/Output	BC_1	49	–
143	D14—Output Enable	Control	BC_1	_	70
144	D14—Pull-up Enable	Control	BC_1	–	–
145	D15—Input	Input/Output	BC_1	50	–
146	D15—Output	Input/Output	BC_1	50	–
147	D15—Output Enable	Control	BC_1	–	–
148	D15—Pull-up Enable	Control	BC_1	–	–
149	SRD—Input	Input/Output	BC_1	51	69
150	SRD—Output	Input/Output	BC_1	51	69
151	SRD—Output Enable	Control	BC_1	–	–
152	SRD—Pull-up Enable	Control	BC_1	–	–
153	SRFS—Input	Input/Output	BC_1	52	–
154	SRFS—Output	Input/Output	BC_1	52	–
155	SRFS—Output Enable	Control	BC_1	–	–
156	SRFS—Pull-up Enable	Control	BC_1	–	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
157	SRCK—Input	Input/Output	BC_1	53	68
158	SRCK—Output	Input/Output	BC_1	53	68
159	SRCK—Output Enable	Control	BC_1	–	–
160	SRCK—Pull-up Enable	Control	BC_1	–	–
161	STD—Input	Input/Output	BC_1	54	–
162	STD—Output	Input/Output	BC_1	54	–
163	STD—Output Enable	Control	BC_1	–	–
164	STD—Pull-up Enable	Control	BC_1	–	–
165	STFS—Input	Input/Output	BC_1	55	–
166	STFS—Output	Input/Output	BC_1	55	–
167	STFS—Output Enable	Control	BC_1	–	–
168	STFS—Pull-up Enable	Control	BC_1	–	–
169	STCK—Input	Input/Output	BC_1	56	–
170	STCK—Output	Input/Output	BC_1	56	–
171	STCK—Output Enable	Control	BC_1	–	–
172	STCK—Pull-up Enable	Control	BC_1	–	–
173	CLKO—Output	Output	BC_1	65	65
174	CLKO—Output Enable	Control	BC_1	–	65
175	MPIOB0—Input	Input/ Output	BC_1	66	–
176	MPIOB0—Output	Input/Output	BC_1	66	–
177	MPIOB0—Output Enable	Control	BC_1	–	–
178	MPIOB0—Pull-up Enable	Control	BC_1	–	–
179	MPIOB1—Input	Input/Output	BC_1	67	–
180	MPIOB1—Output	Input/Output	BC_1	67	–
181	MPIOB1—Output Enable	Control	BC_1	–	64
182	MPIOB1—Pull-up Enable	Control	BC_1	–	64
183	MPIOB2—Input	Input/Output	BC_1	68	–
184	MPIOB2—Output	Input/Output	BC_1	68	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
185	MPIOB2—Output Enable	Control	BC_1	–	–
186	MPIOB2—Pull-up Enable	Control	BC_1	–	–
187	MPIOB3—Input	Input/Output	BC_1	69	–
188	MPIOB3—Output	Input/Output	BC_1	69	–
189	MPIOB3—Output Enable	Control	BC_1	–	–
190	MPIOB3 —Pull-up Enable	Control	BC_1	–	47
191	MPIOB4—Input	Input/Output	BC_1	70	46
192	MPIOB4—Output	Input/Output	BC_1	70	45
193	MPIOB4—Output Enable	Control	BC_1	–	44
194	MPIOB4—Pull-up Enable	Control	BC_1	_	43
195	MPIOB5—Input	Input/Output	BC_1	71	–
196	MPIOB5—Output	Input/Output	BC_1	71	–
197	MPIOB5—Output Enable	Control	BC_1	–	–
198	MPIOB5—Pull-up Enable	Control	BC_1	–	–
199	MPIOB6—Input	Input/Output	BC_1	72	42
200	MPIOB6—Output	Input/Output	BC_1	72	41
201	MPIOB6—Output Enable	Control	BC_1	–	40
202	MPIOB6—Pull-up Enable	Control	BC_1	–	–
203	MPIOB7—Input	Input/Output	BC_1	73	–
204	MPIOB7—Output	Input/Output	BC_1	73	–
205	MPIOB7—Output Enable	Control	BC_1	–	–
206	MPIOB7—Pull-up Enable	Control	BC_1	–	–
207	MPIOD0—Input	Input/Output	BC_1	74	–
208	MPIOD0—Output	Input/Output	BC_1	74	–
209	MPIOD0—Output Enable	Control	BC_1	–	–
210	MPIOD0—Pull-up Enable	Control	BC_1	–	–
211	MPIOD1—Input	Input/Output	BC_1	75	–
212	MPIOD1—Output	Input/Output	BC_1	75	–

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
213	MPIOD1—Output Enable	Control	BC_1	–	–
214	MPIOD1—Pull-up Enable	Control	BC_1	–	–
215	MPIOD2—Input	Input/ Output	BC_1	76	–
216	MPIOD2—Output	Input/ Output	BC_1	76	–
217	MPIOD2—Output Enable	Control	BC_1	–	–
218	MPIOD2—Pull-up Enable	Control	BC_1	–	–
219	MPIOD3—Input	Input/Output	BC_1	77	–
220	MPIOD3—Output	Input/Output	BC_1	77	–
221	MPIOD3—Output Enable	Control	BC_1	–	–
222	MPIOD3—Pull-up Enable	Control	BC_1	–	–
223	MPIOD4—Input	Input/Output	BC_1	78	–
224	MPIOD4—Output	Input/Output	BC_1	78	–
225	MPIOD4—Output Enable	Control	BC_1	–	–
226	MPIOD4—Pull-up Enable	Control	BC_1	–	–
227	MPIOD5—Input	Input/Output	BC_1	79	–
228	MPIOD5—Output	Input/Output	BC_1	79	–
229	MPIOD5—Output Enable	Control	BC_1	–	–
230	MPIOD5—Pull-up Enable	Control	BC_1	–	–
231	MPIOD6—Input	Input/Output	BC_1	82	–
232	MPIOD6—Output	Input/Output	BC_1	82	–
233	MPIOD6—Output Enable	Control	BC_1	–	–
234	MPIOD6—Pull-up Enable	Control	BC_1	–	–
235	MPIOD7—Input	Input/Output	BC_1	83	–
236	MPIOD7—Output	Input/Output	BC_1	83	–
237	MPIOD7—Output Enable	Control	BC_1	–	–
238	MPIOD7—Pull-up Enable	Control	BC_1	–	–
239	SCLK—Input	Input/Output	BC_1	84	30
240	SCLK—Output	Input/Output	BC_1	84	30

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
241	SCLK—Output Enable	Control	BC_1	–	–
242	SCLK—Pull-up Enable	Control	BC_1	–	–
243	MOSI—Input	Input/Output	BC_1	85	29
244	MOSI—Output	Input/Output	BC_1	85	29
245	MOSI—Output Enable	Control	BC_1	–	–
246	MOSI—Pull-up Enable	Control	BC_1	–	–
247	MISO—Input	Input/Output	BC_1	86	27
248	MISO—Output	Input/Output	BC_1	86	27
249	MISO—Output Enable	Control	BC_1	–	–
250	MISO—Pull-up Enable	Control	BC_1	–	–
251	SS_B—Input	Input/Output	BC_1	87	26
252	SS_B—Output	Input/Output	BC_1	87	26
253	SS_B—Output Enable	Control	BC_1	–	–
254	SS_B—Pull-up Enable	Control	BC_1	–	–
255	TA3—Input	Input/Output	BC_1	88	25
256	TA3—Output	Input/Output	BC_1	88	25
257	TA3—Output Enable	Control	BC_1	–	–
258	TA3—Pull-up Enable	Control	BC_1	–	–
259	TA2—Input	Input/Output	BC_1	89	24
260	TA2—Output	Input/Output	BC_1	89	24
261	TA2—Output Enable	Control	BC_1	–	–
262	TA2—Pull-up Enable	Control	BC_1	–	–
263	TA1—Input	Input/Output	BC_1	90	22
264	TA1—Output	Input/Output	BC_1	90	22
265	TA1—Output Enable	Control	BC_1	–	–
266	TA1—Pull-up Enable	Control	BC_1	–	–
267	TA0—Input	Input/Output	BC_1	91	21
268	TA0—Output	Input/Output	BC_1	91	21

Table 17-11. BSR Contents for 56F80x

Bit #	Pin/Bit Name	Pin Type	BSR Cell Type	56F826 Pin Number (100 LQFP Package)	56F827 Pin Number (128 LQFP Package)
269	TA0—Output Enable	Control	BC_1	–	–
270	TA0—Pull-up Enable	Control	BC_1	–	–
271	RXD1	Input	BC_1	92	–
272	TXD1—Input	Input/Output	BC_1	93	20
273	TXD1—Output	Input/Output	BC_1	93	20
274	TXD1—Output Enable	Control	BC_1	–	–
275	TXD1—Pull-up Enable	Control	BC_1	–	–
276	RXD0—Input	Input/Output	BC_1	96	19
277	RXD0—Output	Input/Output	BC_1	96	19
278	RXD0—Output Enable	Control	BC_1	–	–
279	RXD0—Pull-up Enable	Control	BC_1	–	–
280	TXD0—Input	Input/Output	BC_1	97	18
281	TXD0—Output	Input/Output	BC_1	97	18
282	TXD0—Output Enable	Control	BC_1	–	–
283	TXD0—Pull-up Enable	Control	BC_1	–	–
284	DE_B—Output	Output	BC_1	98	17
285	DE_B—Output Enable	Control	BC_1	_	–

17.5.4 JTAG Bypass Register (JTAGBR)

The JTAG Bypass Register is a one-bit register providing a simple, direct path from the TDI pin to the TDO pin. This is useful in boundary scan applications where many chips are serially connected in a daisy-chain. Individual devices, or other devices, can be programmed with the bypass instruction so individually they become pass-through devices during testing. This allows testing of a specific chip, while still having all of the chips connected through the JTAG ports.

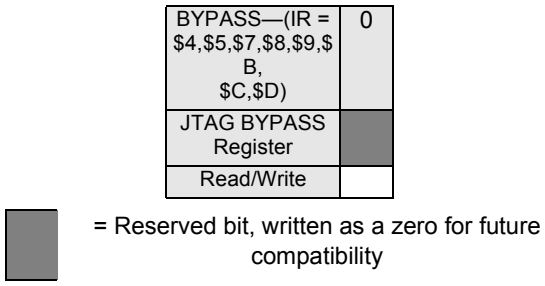


Figure 17-12. JTAG Bypass Register (JTAGBR)

17.6 TAP Controller

The TAP controller is a synchronous finite state machine containing sixteen states, as illustrated in **Figure 17-13.** The TAP controller responds to changes at the TMS and TCK signals. Transitions from one state to another occur on the rising edge of TCK. The value shown adjacent to each state transition in this figure represents the signal present at TMS at the time of a rising edge at TCK. The TDO pin remains in the high impedance state except during the Shift-DR or Shift-IR controller states. In these controller states, TDO is updated on the falling edge of TCK. TDI is sampled on the rising edge of TCK.

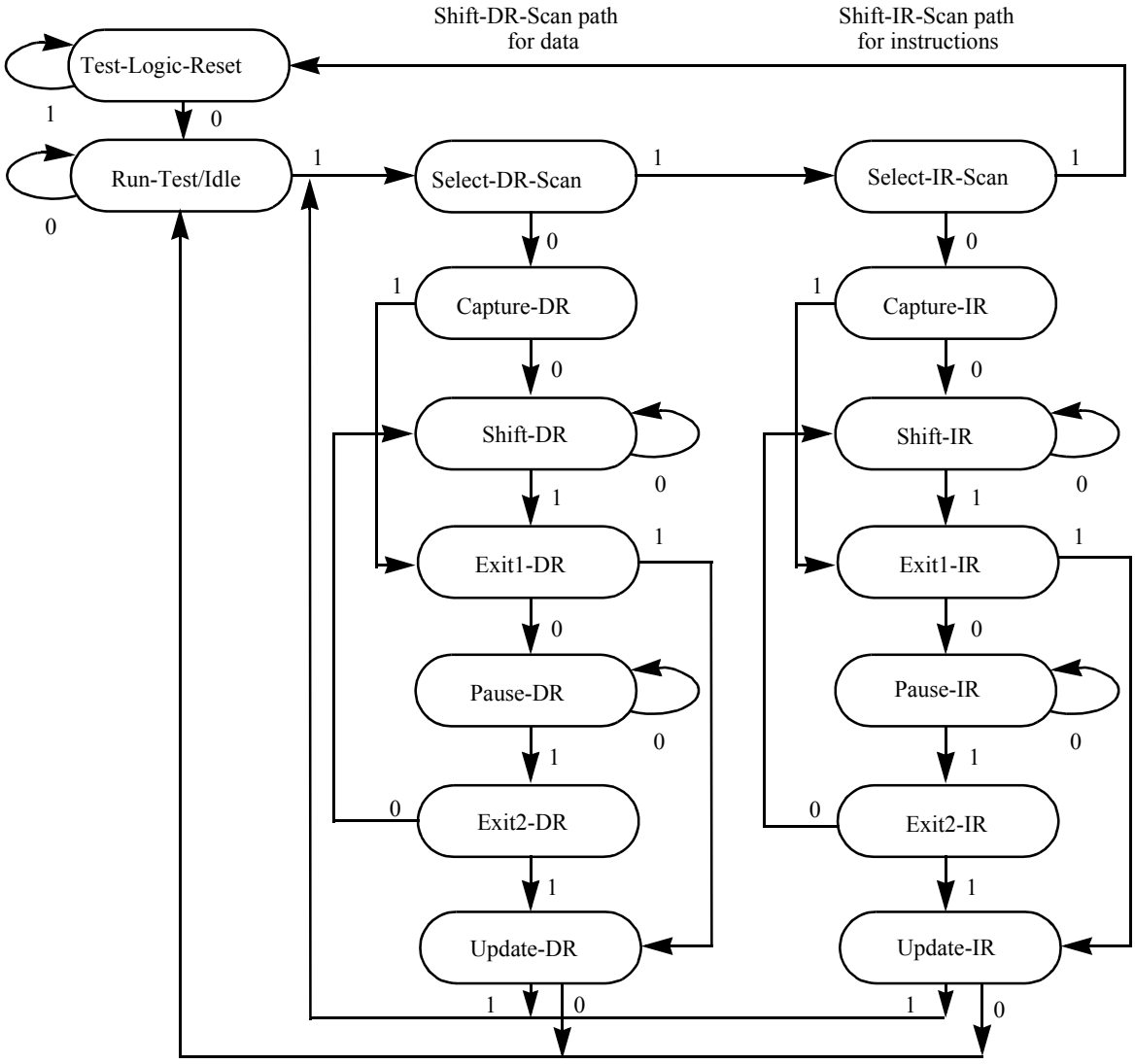


Figure 17-13. TAP Controller State Diagram

There are two paths through the 16-state machine:

1. The Shift-IR-Scan path captures and loads JTAG instructions into the JTAGIR.
2. The Shift-DR-Scan path captures and loads data into the other JTAG registers.

The TAP controller executes the last instruction decoded until a new instruction is entered at the Update-IR state, or until the test-logic-reset state is entered. When using the JTAG port to access OnCE module registers, accesses are first enabled by shifting the ENABLE_ONCE instruction into the JTAGIR. After this is selected, the OnCE module registers and commands are read and written through the JTAG pins using the Shift-DR-Scan path. Asserting the JTAG's $\overline{\text{TRST}}$ pin asynchronously forces the JTAG state machine into the test-logic-reset state.

17.7 56F826/827 Restrictions

Control afforded by the output enable signals using the BSR and the EXTEST instruction requires a compatible circuit board test environment to avoid any device-destructive configurations. *Avoid situations when the DSP56F826/827 output drivers are enabled into actively driven networks.*

During power-up, the $\overline{\text{TRST}}$ pin must be externally asserted to force the TAP controller into this state. After power-up is concluded, TMS must be sampled as a logic one for five consecutive TCK rising edges. If TMS either remains unconnected or is connected to V_{DD}, then the TAP controller cannot leave the test-logic-reset state, regardless of the state of TCK.

56F826/827 features a low-power stop mode invoked using the Stop instruction. JTAG interaction with low-power Stop mode is as follows:

1. The TAP controller must be in the test-logic-reset state to either enter or remain in stop mode. Leaving the TAP controller test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
2. The TCK input is not blocked in low-power stop mode. To consume minimal power, the TCK input should be externally connected to V_{DD} or ground.
3. The TMS and TDI pins include on-chip pull-up resistors. In low-power stop mode, these two pins should remain either unconnected or connected to V_{DD} to achieve minimal power consumption.

Because all 56F826/827 clocks are disabled during Stop mode, the JTAG interface provides the means of polling the device status, sampled in the Capture-IR state.

Appendix A
Glossary

A.1 Glossary

This glossary is intended as a resource and to reduce confusion possibly created by the use of many acronyms and abbreviations throughout this manual.

ACIM	A/C Induction Motors
A/D	Analog-to-Digital
ADC	Analog-to-Digital Converter
ADCR	ADC Control Registe
ADDR	Address
ADHLMT	ADC High Limit Registers
ADLLMT	ADC Low Limit Registers
ADLST	ADC Channel List Registers
ADLSTAT	ADC Limit Status Register
ADM	Application Development Module
ADOFS	ADC Offset Registers
ADR PD	Address Bus Pull-up Disable
ADRSLT	ADC Result Registers
ADSDIS	ADC Sample Disable Register
ADSTAT	ADC Status Register
ADZCC	ADC Zero Crossing Control Register
ADZCSTAT	ADC Zero Crossing Status Register
AGU	Address Generation Unit
ALU	Arithmetic Logic Unit
API	Application Program Interface
Barrel Shifter	Part of the ALU that allows single cycle shifting and rotating of data word
BCR	Bus Control Register
BDC	Brush DC Motor
BE	Breakpoint Enable
BFIU	Boot Flash Interface Unit
BFLASH	Boot Flash
BK	Breakpoint Configuration Bit
BLDC	Brushless DC Motor
BOTNEG	Bottom-side PWM Polarity Bit

BS	Breakpoint Selection
BSDL	Boundary Scan Description Language
BSR	Boundary Scan Register
CAN	Controller Area Network
CC	Condition Codes
CAP	Capture
CEN	COP Enable Bit
CFG	Config
CGDB	Core Global Data Bus
CHCNF	Channel Configure
CID	Chip Identification Register
CIP	Conversion in Progress
CKDIVISOR	Clock Divisor
CLKO	Clock Output pin
CLKOSEL	CLKO Select
CLKOSR	Clock Select Register
CMOS	Complementary metal oxide semiconductor. (A form of digital logic that is characterized by low power consumption, wide power suppply range, and high noise immunity.)
CMP	Compare
CNT	Count
CNTR	Counter
Codec	Coder/Decoder
COP	Computer Operating Properly
COP/RTI	Computer Operating Properly/Real Time Interface
COPCTL	COP Control
COPDIS	COP Timer Disable
COPR	COP Reset
COPSRV	COP Service
COPTO	COP Time Out
CPHA	Clock Phase
CPOL	Clock Polarity
CPU	Central Processing Unit
CRC	Cyclic Redundancy Code

CSEN	Cop Stop Enable
CTRL	Control
CTRL PD	Control signal Pull-up Disable
CWEN	COP Wait Enable Bit
CWP	COP Write Protect
DAC	Digital to Analog Converter
DAT	Data/Address Select
DATA ALU	Data Address Limit
DATA PD	Data Bus I/O Pull-up Disable
DDA	Analog Power
DDR	Data Direction Register
DEC	Quadrature Decoder Module
DEE	Dumb Erase Enable
DFIU	Data Flash Interface Unit
DFLASH	Data Flash
DIE	Watchdog Time-Out Interrupt Enable
DIRQ	Watchdog Time-Out Interrupt Request
DMA	Direct Memory Access
DPE	Dumb Programming Enable
DR	Data Register
DRV	Drive Control Bit
DSO	Data Shift Order
DSP	Digital Signal Processor
EDG	Edge-Aligned or Center-Aligned PWMs
EE	Erase Enable
EEOF	Enable External OFLAG Force
EM	Event Modifier
EN	Enable3
ENCR	Encoder Control Register
EOSI	End of Scan Interrupt
EOSIE	End of Scan Interrupt Enable
ERASE	Erase Cycle

ERRIE Error Interrupt Enable

EX External X Memory

EXTBOOT External Boot

EXTAL External pin for clocks, oscillators, and so on

EXTR External Reset

FAULT Fault Input to PWM

FE Framing Error Flag

FFLAGx FAULTx Pin Flag

FH FIFO Halt

FIEx Faultx Pin Interrupt Enable

FIR Filter Interval Register

FIU Flash Interface Unit

Flash memory Nonvolatile memory, electronically erasable and programmable

FLOCI Force Loss of Clock

FLOLI Force Loss of Lock

FMODEx FAULTx Pin Clearing Mode

FPINx FAULTx Pin

FTACKx FAULTx Pin Acknowledge

GO Execute

GPIO General Purpose Input/Output (A, B, C, D, E or F)

GPIO_IPR General Purpose Input/Output Interrupt Pending Register

GPR Group Priority Register

Harvard architecture - A microprocessor architecture using separate buses for program and data. This is typically used on DSPs to optimise the data throughput

HBO Hardware Breakpoint Occurrence

HLMTI High Limit Interrupt

HLMTIE High Limit Interrupt Enable

HOLD Hold Register

HOME Home Switch Input

HWS Hardware Stack

IA Interrupt Assert

IC Integrated Circuit

IE Interrupt Enable

IEE	Intelligent Erase Enable
IEF	Input Edge Flag
IEFIE	Input Edge Flag Interrupt Enable
IENR	Interrupt Enable Register
IES	Interrupt Edge Sensitive
IFREN	Information Block Enable
IMR	Input Monitor Register
INDEP	Independent or Complementary Pair Operation
INDEX	Index Input
INPUT	External Input Signal
INV	Invert
I/O	Input/Output
IP	Interrupt Pending
IPBus	Intellectual Properties Bus
IPE	Intelligent Program Enable
IPOL	Current Polarity
IPOLR	Interrupt Polarity Register
IPR	Interrupt Priority Register (in the Core)
IPS	Input Polarity Select
IRQ	Interrupt Request
IR	Instruction Register
IS	Interrupt Source
ISR	Interrupt Service Routine
ITCN	Interrupt Controller
JTAG	Joint Test Action Group
JTAGBR	JTAG Bypass Register
JTAGIR	JTAG Instruction Register
LCD	Liquid Crystal Display
LCK	Loss of Lock
LDOK	Load Okay
LIR	Lower Initialization Register
LLMTI	Low Limit Interrupt

LLMTIE	Low Limit Interrupt Enable
LOAD	Load Register
LOCI	Loss of Clock
LOCIE	Los of Clock Interrupt Enable
LOLI	PLL Lock of Lock Interrupt
LOOP	Loop Select Bit
LPOS	Lower Position Counter Register
LPOSH	Lower Position Hold Register
LSB	Least Significant Bit
LSH_ID	Most Significant Half of JTAG_ID
LVD	Low Voltage Detect
LVIE	Low Voltage Interrupt Enable
LVIS	Low Voltage Interrupt Source
MA	Mode A
MAC	Multiply and Accumulate
MAS	Mass Cycle Erase
MB	Mode B
MCU	Microcontroller Unit
MHz	Megahertz
MIPS	Million Instructions Per Second
MISO	Master In/Slave Out
MODF	Mode Fault Error
MODFEN	Mode Fault Enable
MOSI	Master Out/Slave In
MSB	Most Significant Bit
MSH_ID	Most Significant Half of JTAG ID
MSTR	Master Mode
MUX	Multiplexer
NF	Noise Flag
NL	Nested Looping
NOP	No Operation
NOR	An inversion of the logical OR function

NVSTR	Non-volatile Store Cycle Definition
OBAR	OnCE Breakpoint Address Register
OBCTL	OnCE Breakpoint Control Register
OCMDR	OnCE Command Register
OBMSK	OnCE Breakpont Mask Register
OCMDR	OnCE Command Register
OCCS	On-Chip Clock Synthesis
OCNTR	OnCE Count Register
OCR	OnCE Control Register
ODEC	OnCE Decoder
OEN	Output Enable
OMAC	OnCE Memory Address Comparator
OMAL	OnCE Memory Address Latch
OMR	Operating Mode Register
OnCE	On-Chip Emulation (unit)
OPABDR	OnCE Program Address Bus Decode Register
OPABER	OnCE Program Address Bus Execute Register
OPABFR	OnCE Program Address Bus Fetch Register
OPDBR	OnCE Program Data Bus Register
OPFIFO	OnCE PAB Change of Flow
OPGDBR	OnCE Program Global Data Bus Register
OPS	Output Polarity Select
OR	Overrun
OS	OnCE Status bits OS1 and OS0
OSHR	OnCE Shift Register
OS	OnCE status bits OS1 and OS0
OSC	Oscillator
OSR	OnCE Status Register
OVRF	Overflow
PAB	Program Address Bus
PD	Permanent STOP/WAIT Disable
PDB	Program Data Bus

PE	Program Enable
PE	Parity Enable Bit
PER	Peripheral Enable Register
PF	Parity Error Flag
PFIU	Program Flash Interface Unit
PFLASH	Program Flash
PGDB	Peripheral Global Data Bus
PLL	Phase Locked Loop
PLLCID	PLL Clock In Divide
PLLCOD	PLL Clock Out Divide
PLLDB	PLL Divide-by
PLLCR	PLL Control Register
PLLPDN	PLL Power Down
PLLSR	PLL Status Register
PLR	Priority Level Register
PMCCR	PWM Channel Control Register
PMCFG	PWM Configuration Register
PMCNT	PWM Counter Register
PMCTL	PWM Control Register
PMDEADTM	PWM Deadtime Register
PMDISMAP	PWM Disable Mapping Registers
PMFCTL	PWM Fault Control Register
PMFSA	PWM Fault Status Acknowledge
PMOUT	PWM Output Control Register
PMPORT	PWM Port Register
POL	Polarity
POR	Power on Reset
PRAM	Program RAM
PROG	Program Cycle
PSR	Processor Status Register
PT	Parity Type
PTM	Peripheral Test Mode

PUR	Pull-up Enable Register
PWD	Power Down Mode
PWM	Pulse Width Modulator
PWMEN	PWM Enable
PWMF	PWM Reload Flag
PWMRIE	PWM Reload Interrupt Enable
PWMVAL	PWM Value Registers
QE	Quadrature Encoder
QDN	Quadrature Decoder Negative Signal
RAF	Receiver Active Flag
RAM	Random Access Memory
RDF	Receiver Data Full
RDMAE	Receive DMA Enable
RDR	Receive Data Ready
RDRF	Receive Data Register Full
RE	Receiver Enable
REIE	Receive Error Interrupt Enable
REV	Revolution Counter Register
REVH	Revolution Hold Register
RFEN	Receive FIFO Enable
RFF	Receive FIFO Full
RIDLE	Receiver Idle Line
RIE	Receiver Full Interrupt Enable
ROM	Read Only Memory
RPD	Re-programmable Stop/Wait Disable
RS	Register Select
RSCKP	Receive Clock Priority
RSRC	Receiver Source Bit
RT	Rate Tolerance
RWU	Receiver Wake-up
RXSR	Receive (data) Shift Register
SA	Saturation

SBK	Send Break
SBO	Software Breakpoint Occurrence
SBR	SCI Baud Rate
SCI	Serial Communications Interface3
SCIBR	SCI Baud Rate Register
SCICR	SCI Control Register
SCIDR	SCI Data Register
SCISR	SCI Status Register
SCLK	Serial Clock
SCR	Status and Control
SCSR	SCI Control Status Register
SD	Stop Delay
SDK	Software Development Kit
SEXT	Sign Extend
SHFD	Shift Direction
SIM	System Integration Module
SMODE	Scan Mode
SPDRR	SPI Data Receive Register
SPDSR	SPI Data Size Register
SPDTR	SPI Data Transmit Register
SP	SPI Enable
SPI	Serial Peripheral Interface
SPMSTR	SPI Master
SPRF	SPI Receiver Full
SPRIE	SPI Receiver Interrupt Enable
SPSCR	SPI Status Control Register
SPTE	SPI Transmitter Empty
SPTIE	SPI Transmit Interrupt Enable
SR	Status Register
SRM	Switched Reluctance Motor
SRX	SSI Receive Data
$\overline{\textbf{SS}}$	Slave Select

SSI	Synchronous Serial Interface
STX	SSI Transmit Data
SWAI	Stop in Wait Mode
SWI	Software Interrupt
SYN	Synchronous/Sync
SYS_CNTL	System Control Register
SYS_STS	System Status Register
TAP	Test Access Port
TCSR	Text Control and Status Register
TCE	Test Counter Enable
TCF	Timer Compare Flag
TCFIE	Timer Compare Flag Interrupt Enable
TDE	Transmit Data Empty
TDRE	Transmit Date Register Empty
TE	Transmitter Enable
TEIE	Transmitter Empty Interrupt Enable
TEN	Test Mode Enable
TERASEL	Terase Limit
TESTR	Test Register
TFDBK	Test Feedback Clock
TFREF	Test Reference Frequency Clock
TIDLE	Transmitter Idle
TIIE	Transmitter Idle Interrupt Enable
TIRQ	Test Interrupt Request Register
TISR	Test Interrupt Source Register
TM	Test Mode bit
TMEL	Time Limit
TMODE	Test Mode bit
TMR	Quadrature Timer
TMR PD	Timer I/O Pull-up Disable
TNVHL	TNVH Limit
TNVSL	TNVS Limit

TO	Trace Occurrence
TOD	Time of Day
TOF	Timer Overflow Flag
TOFIE	Timer Oerflow Flag Interrupt Enable
TOPNEG	Top-side PWM Polarity Bit
TPROGL	Tprog Limit
TPGSL	TPGS Limit
TRCVL	TRCV Limit
TSTREG	Test Register
UIR	Upper Initialization Register
UPOS	Upper Position Hold Register
UPOSH	Upper Position Hold Register
VCO	Voltage Controlled Oscillator
$\mathbf{V_{DD}}$	Voltage Digital Drain (Power)
$\mathbf{V_{DDA}}$	Analog Power
VEL	Velocity Counter Register
VELH	Velocity Hold Register
VLMODE	Value Register Load Mode
VREF	Voltage Reference
VRM	Variable Reluctance Motor
$\mathbf{V_{SS}}$	Ground
$\mathbf{V_{SSA}}$	Analog Ground
WAKE	Wake-up Condition
WDE	Watchdog Enable
WP	Write Protect
WSPM	Wait State P Memory
WSX	Wait State Data Memory
WTR	Watchdog Timeout Register
WWW	World Wide Web
X	Memory words (data memory)
XDB2	X Data Bus
XE	X Address Enable

XIE	Index Pulse Interrupt Enable
XIRQ	Index Pulse Interrupt Request
XNE	Use Negative Edge of Index Pulse
XRAM	Data RAM
XTAL	External pins for clocks, oscillators, and so on
YE	Y Address Enable
ZCI	Zero Crossing Interrupt
ZCIE	Zero Crossing Interrupt Enable
ZCS	Zero Crossing Status
ZSRC	Zclock Source

Appendix B
Programmer's Sheets

B.1 Introduction

The following pages provide a set of reference tables and programming sheets intended to simplify programming the 56F826/827. The programming sheets provide room to add the value of each bit and the hexadecimal value for each register. These pages may be photocopied. Copy at 125 percent to enlarge these pages to an 8.5 x 11inch sheet.

B.2 Instruction Set Summary

The following tables provide brief summaries of the instruction set for the 56F826 and 56F827. For complete instruction set details, refer to Appendix A of the *DSP56800 Family Manual*.

Each described instruction contains notations used to abbreviate certain operands and operations described in **Table B-7.** Keys to the symbols and their respective descriptions of the *Set Instructions* located in **Table B-7** are listed in **Table B-1** through **Table B-6.**

Table B-2 shows the register set available for the most important move instruction. Sometimes the register field is broken into two different fields–one where the register is used as a source and the other where it is used as a destination. This is important because a different notation is used when an accumulator is being stored without saturation.

Table B-1. Register Fields for General-Purpose Writes and Reads

Register Field	Registers in This Field	Comments
HHH	A, B, A1, B1 X0, Y0, Y1	Seven data ALU registers---two accumulators, two 16-bit MSP portions of the accumulators and three 16-bit data registers
HHHH	A, B, A1, B1 X0, Y0, Y1 R0-R3, N	Seven data ALU and five AGU registers
DDDDD	A, A2, A1, A0 B, B2, B1, B0 Y1, Y0, X0 R0, R1, R2, R3 N, SP M01 OMR, SR LA, LC HWS	All CPU registers

Table B-2 illustrates the register set available for use as pointers in address-register- indirect addressing modes. The most common fields used in this table are Rn and RRR. This table also provides notations used for AGU registers in AGU arithmetic operations.

Table B-2. Data ALU Registers

Register Field	Registers in This Field	Comments
Rn	R0-R3 N	Five AGU registers available as pointers for addressing and as sources and destinations for move instructions
Rj	R0, R1, R2, R3	Four pointer registers available as pointers for addressing
N	N	One index register available only for indexed addressing modes
M01	M01	One modifier register

Table B-3 illustrates the register set available for use in data ALU arithmetic operations. The common field used in this table is FFF.

Table B-3. Data ALU Registers

Register Field	Registers in This Field	Comments
FDD	A, B X0, YO, Y1	Five data ALU registers--two 36-bit accumulators and three 16-bit data registers accessible during data ALU operations Contains the contents of the F and DD register fields
F1DD	A1, B1 X0, Y0, Y1	Five data ALU registers--two 16-bit MSP portions of the accumulators and three 16-bit data registers accessible during data ALU operations
F	A, B	Two 36-bit data registers
DD	X0, Y0, Y1	Three 16-bit data registers
F	A, B	Two 36-bit accumulators accessible during ALU operations
F1	A1, B1	The 16-bit MSP portion of two accumulators accessible as source operands in parallel move instructions

Address operands used in the instruction field sections of the instruction descriptions are provided in **Table B-4.**

Table B-4. Address Operands

Symbol	Description
ea	Effective address
eax	Effective address for X bus
xxxx	Absolute address (16 bits)
pp	I/O short address (6 bits, one-extended)
aa	Absolute address (6 bits, zero-extended)
<...>	Specifies the contents of the specified address
X:	X memory reference
P:	Program memory reference

Addressing mode operations accepted by the assembler for stipulating a specific addressing mode are provided in **Table B-5.**

Table B-5. Addressing Mode Operators

Symbol	Description
<<	I/O short or absolute addressing mode fore operator
>	Long addressing mode force operator
#	Immediate addressing more operator
#>	Immediate long addressing mode for operator
#<	Immediate short addressing mode force operator

Miscellaneous operand notations, including generic source, destination operands, and immediate data specifiers are summarized in **Table B-6.**

Table B-6. Miscellaneous Operands

Symbol	Description
S, Sn	Source operand register
D, Dn	Destination operand register
#xx	Immediate short data (7 bits for MOVE (I), 6 bits for DO/REP)
#xxxx	Immediate data (16 bits)
#ii00	8-bit immediate data mask in the upper byte
#00ii	8-bit immediate data mask in the lower byte

Table B-7. Instruction Set Summary

					CCR							
Mnemonic	**Syntax**	**Parallel Moves**	**Prog. Word**	**Clock Cycles**	**SZ**	**L**	**E**	**U**	**N**	**Z**	**V**	**C**
ABS	D	(parallel move)	1	2 + mv	*	*	*	*	*	*	*	—
ADC	S,D	(no parallel move)	1	2	—	*	*	*	*	*	*	*
ADD	S,D	(parallel move)	1	2 + mv	*	*	*	*	*	*	*	*
AND	S,D	(no parallel move)	1	2	—	*	—	—	?	?	0	—
ANDC	#iiii,X:<ea> #iiii,D	. . .	2 + ea	4 + mvb	—	—	—	—	—	—	—	?

Table B-7. Instruction Set Summary (Continued)

ASL	D	(parallel move)	1	2 + mv	*	*	*	*	*	*	?	?
ASLL	S1,S2,D	(no parallel move)	1	2	—	—	—	—	*	*	—	—
ASR	D	(parallel move)	1	2 + mv	*	*	*	*	*	*	0	?
ASRAC	S1, S2, D	(no parallel move)	1	2	—	—	—	—	*	*	—	—
ASRR	S1, S2, D	(no parallel move)	1	2	—	—	—	—	*	*	—	—
BCC	xxxx ee Rn	. . .	2 1 1	4 + j x	—	—	—	—	—	—	—	—
BFCHG	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . .	2 + ea	4 + mvb 4 + mvb 6 + mvb 4 + mvb	—	—	—	—	—	—	—	?
BFCLR	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . .	2 + ea	4 + mvb 4 + mvb 6 + mvb 4 + mvb	—	—	—	—	—	—	—	?
					CCR							
Mnemonic	**Syntax**	**Parallel Moves**	**Prog. Word**	**Clock Cycles**	**SZ**	**L**	**E**	**U**	**N**	**Z**	**V**	**C**
BFSET	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . .	2 + ea	4 + mvb 4 + mvb 6 + mvb 4 + mvb	—	—	—	—	—	—	—	?
BFTSTH	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . .	2 + ea	4 + mvb 4 + mvb 6 + mvb 4 + mvb	—	—	—	—	—	—	—	?
BFTSTL	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . .	2 + ea	4 + mvb 4 + mvb 6 + mvb 4 + mvb	—	—	—	—	—	—	—	?
BRA	xxxx aa Rn	. . .	2 1 1	6 + j x	—	—	—	—	—	—	—	—

Table B-7. Instruction Set Summary (Continued)

BRCLR	#iiii,X:<ea>,aa #iiii,D,aa	. . .	2 + ea	8 + mvb	—	—	—	—	—	—	—	?
BRSET			2 + ea	8 + mvb	—	—	—	—	—	—	—	?
CLR	D	(parallel move)	1	2 + mv	*	*	*	*	*	*	0	—
CMP	S, D	(parallel move)	1 + ea	2 + mv	*	*	*	*	*	*	*	*
DEBUG		. . .	1	4	—	—	—	—	—	—	—	—
DEC(W)	D	(parallel move)	1 + ea	2 + mv	*	*	*	*	*	?	*	*
DIV	S, D	(parallel move)	1	2	—	*	—	—	—	—	?	?
DO	X:(Rn),expr #xx,expr S, expr	(no parallel move)	2	6	—	*	—	—	—	—	—	—
ENDDO		. . .	1	2	—	—	—	—	—	—	—	—
EOR	S, D	. . .	1	2	—	*	—	—	?	?	0	—
EORC	#iiii,X:<ea> #iiii,D	. . .	2 + ea	4 + mvb	—	—	—	—	—	—	—	?
					CCR							
Mnemonic	**Syntax**	**Parallel Moves**	**Prog. Word**	**Clock Cycles**	**SZ**	**L**	**E**	**U**	**N**	**Z**	**V**	**C**
ILLEGAL		(no parallel move)	1	4	—	—	—	—	—	—	—	—
IMPY(16)	S1, S2, D	(no parallel move)	1	2	—	*	?	?	*	*	*	—
INC(W)	D	(parallel move)	1 + ea	2 + mv	*	*	*	*	*	?	*	*
JCC	xxxx (Rn)	. . .	2	4 + jx	—	—	—	—	—	—	—	—
JMP	xxxx (Rn)	. . .	2 1	6 + jx	—	—	—	—	—	—	—	—
JSR	xxxx AA	. . .	2 1	8 + jx	—	—	—	—	—	—	—	—
LEA	ea, D	(no parallel move)	1 + ea	2 + ea	—	—	—	—	—	?	?	—

Table B-7. Instruction Set Summary (Continued)

LSL	D	(no parallel move)	1	2	—	*	—	—	?	?	0	?
LSLL	S1, S2, D	(no parallel move)	1	2	—	—	—	—	*	*	—	—
LSR	D	(no parallel move)	1	2	—	*	—	—	?	?	0	?
LSRAC	S1, S2, D	(no parallel move)	1	2	—	—	—	—	*	*	—	—
LSRR	S1, S2, D	(no parallel move)	1	2	—	—	—	—	*	*	—	—
MAC	(±)S2, S1, D S2, S1, D S1, S2, D	(no parallel move) (one parallel move) (two parallel reads)	1	2 + mv	*	*	*	*	*	*	*	—
MACR	(±)S2, S1, D S2, S1, D S1, S2, D	(no parallel move) (one parallel move) (two parallel reads)	1	2 + mv	*	*	*	*	*	*	*	—
					CCR							
Mnemonic	**Syntax**	**Parallel Moves**	**Prog. Word**	**Clock Cycles**	**SZ**	**L**	**E**	**U**	**N**	**Z**	**V**	**C**
MACSU	S1, S2, D	(no parallel move)	1	2	—	*	*	*	*	*	*	—
MOVE[1]	X:<ea>, D S, X:<ea>	. . .	1 + ea	2 + ea	*	*	—	—	—	—	—	—
MOVE(C)	X:<ea>, D S, X:<ea> #xxxx, D S, D X:(R2 + xx),D S,X:(R2 + xx)	. . .	1 + ea	2 + mvc	*	?	?	?	?	?	?	?
MOVE(I)	#xx, D	. . .	1	2	—	—	—	—	—	—	—	—

Table B-7. Instruction Set Summary (Continued)

MOVE(M)	P:<ea>,D S, P:<ea> P:(R2 + xx),D S,P:(R2 + xx) P:<ea>,X:<ea> > X:<ea>,P:<ea> >	. . .	1	8 + mvm	—	*	—	—	—	—	—	—
MOVE(P)	X:<pp>,D X:<ea>,X:<pp> > S, X:<pp> X:<pp>,X:<ea> >	. . .	1	1 + mvp	—	—	—	—	—	—	—	—
MOVE(S)	X:<a>,D S,X:<aa>	. . .	1	2 + mvs	*	*	—	—	—	—	—	—
MPY	(±)S1,S2,D S1, S2, D S1, S2, D	(one parallel move) (two parallel reads) $\overline{D}$,X:(Rn) + (Nn)	1	2 + mv	*	*	*	*	*	*	*	—
MPYR	(±)S1, S2, D S1, S2, D	(one parallel move) (two parallel reads)	1	2 + mv	*	*	*	*	*	*	*	—
					CCR							
Mnemonic	**Syntax**	**Parallel Moves**	**Prog. Word**	**Clock Cycles**	**SZ**	**L**	**E**	**U**	**N**	**Z**	**V**	**C**
MPYSU	S1, S2, D	(no parallel move)	1	2	—	*	*	*	*	*	*	—
NEG	D	(parallel move)	1	2 + mv	*	*	*	*	*	*	*	*
NOP		. . .	1	2	—	—	—	—	—	—	—	—
NORM	Rn, D		1	2	—	*	*	*	*	*	?	—
NOT	D	(no parallel move)	1	2	—	*	—	—	?	?	0	—
NOTC	X:<ea> D	. . .	2 + ea	4 + mvb	—	—	—	—	—	—	—	?

Table B-7. Instruction Set Summary (Continued)

OR	S, D	(no parallel move)	1	2	—	*	—	—	?	?	0	—
ORC	#iiii,X:<ea> #iiii, D	. . .	2 + ea	4 + mvb	—	—	—	—	—	—	—	?
POP	D	. . .	2	2 + mv	—	?	?	?	?	?	?	?
REP	X:(Rn) #xx S	. . .	1	6	—	—	—	—	—	—	—	—
RND	D	(parallel move)	1	2 + mv	*	*	*	*	*	*	*	—
ROL	D	(parallel move)	1	2 + mv	—	*	—	—	?	?	0	?
ROR	D	(parallel move)	1	2 + mv	—	*	—	—	?	?	0	?
RTI		. . .	1	10 + rx	—	—	?	?	?	?	?	?
RTS		. . .	1	10 + rx	—	—	—	—	—	—	—	
SBC	S, D	(no parallel move)	1	2	—	*	*	*	*	*	*	*
STOP[2]		. . .	1	n/a	—	—	—	—	—	—	—	—
SUB	S, D S, D	(parallel move) (two parallel reads)	1 + ea	2 + mv	*	*	*	*	*	*	*	*
					CCR							
Mnemonic	**Syntax**	**Parallel Moves**	**Prog. Word**	**Clock Cycles**	**SZ**	**L**	**E**	**U**	**N**	**Z**	**V**	**C**
SWI		. . .	1	8	—	—	—	—	—	—	—	—
TCC	S, D S, D R0, R1	. . .	1	2	—	—	—	—	—	—	—	—
TFR	S, D	(parallel move)	1	2 + mv	?	—	—	—	—	—	—	—
TST	S	(parallel move)	1	2 + mv	*	*	*	*	*	*	0	—
TST(W)	S	(no parallel move)	1	2 + tst	—	*	*	*	*	*	0	0
WAIT[3]		. . .	1	n/a	—	—	—	—	—	—	—	—

1.This instruction applies only in the case when two reads are performed in parallel from the X memory.

2.The STOP instruction disables the internal clock oscillator. After the clock is turned on, an internal counter waits for 65,536 cycles before enabling the clock to the internal circuits.

3.The WAIT instruction takes a minimum of 16 cycles to execute when an internal interrupt is pending during the execution of the WAIT instruction.

B.3 Interrupt, Vector, and Address Tables

Refer to **Chapter 3, Section 3.5** and **Section 3.10** for interrupt, vector, and address tables. Table titles are listed below.

B.4 Programmer's Sheets

The following pages provide programmer's sheets summarizing functions of the bits in various registers in the 56F826/827. The programmer's sheets provide room to record the value of each bit and the hexadecimal value of each register. Programmers may photocopy these sheets. Copy at 125 percent to enlarge these pages to an 8-1/2 x 11 inch sheet.

The programmer's sheets are arranged in the same order as the sections in this manual. **Table B-1** below lists the programmer's sheets by module, the registers in each module, and the pages in this appendix where the programmer's sheets are located.

Note: Reserved bits ( ) should be set to zero unless otherwise stated.

Table B-1. List of Programmer's Sheets

Register Type	Register	Page
CPU Registers		
Bus Control Register	(BCR)	B-19
Operating Mode Register	(OMR)	B-20
ITCN		
arch.h: ArchIO.IntController, registers.h: ArchiIO_IntController	ITCN_BASE: 56F826/827 = $1100	
Group Priority Register	(GPR0)	B-21
Group Priority Register	(GPR1)	B-21
Group Priority Register	(GPR2)	B-21
Group Priority Register	(GPR3)	B-21
Group Priority Register	(GPR4)	B-22
Group Priority Register	(GPR5)	B-22
Group Priority Register	(GPR6)	B-22
Group Priority Register	(GPR7)	B-22
Group Priority Register	(GPR8)	B-23
Group Priority Register	(GPR9)	B-23
Group Priority Register	(GPR10)	B-23
Group Priority Register	(GPR11)	B-23
Group Priority Register	(GPR12)	B-24
Group Priority Register	(GPR13)	B-24
Group Priority Register	(GPR14)	B-24
Group Priority Register	(GPR15)	B-24

Table B-1. List of Programmer's Sheets (Continued)

Register Type	Register	Page
FLASH		
arch.h: ArchIO.ProgramFlash(826) ArchIO.ProgramFlashLow (827)	PFIU_BASE: 56F826/827 = $1020	
registers: ArchIO_ProgramFlash(826) ArchIO_ProgramFlashLow (827)		
arch.h: ArchIO.ProgramFlashHigh (827)	PFIUAH_BASE: 56F827 = $1040	
registers: ArchIO_ProgramFlashHigh (827)		
arch.h: ArchIO.DataFlash	DFIU_BASE: 56F826/827 = $1060	
registers:ArchIO_DataFlash		
arch.h: ArchIO.BootFlash	BFIU_BASE: 56F826 = $1080	
registers: ArchIO_BootFlash		
Flash Control Register	(FIU_CNTL)	B-25
Flash Program Enable Register	(FIU_PE)	B-26
Flash Erase Enable Register	(FIU_EE)	B-27
Flash Address Enable Register	(FIU_ADDR)	B-28
Flash Data Register	(FIU_DATA)	B-29
Flash Interrupt Enable Register	(FIU_IE)	B-30
Flash Interrupt Source Register	(FIU_IS)	B-31
Flash Interrupt Pending Register	(FIU_IP)	B-32
Flash Clock Divisor Register	(FIU_CKDIVISOR)	B-33
Flash Terase Limit Register	(FIU_TERASEL)	B-34
Flash Tme Limit Register	(FIU_TMEL)	B-35
Flash Tnvs Limit Register	(FIU_TNVSL)	B-36
Flash Tpgs Limit Register	(FIU_TPGSL)	B-37
Flash Tprog Limit Register	(FIU_TPROGL)	B-38
Flash Tnvh Limit Register	(FIU_TNVHL)	B-39
Flash Tnvh1 Limit Register	(FIU_TNVH1L)	B-40
Flash Trcv Limit Register	(FIU_TRCVL)	B-41

GPIO		
arch.h: ArchIO.PortA	GPIOA_BASE: 56F826/827 = $11A0	
registers.h: ArchIO_PortA		
arch.h: ArchIO.PortB	GPIOB_BASE: 56F826/827 = $11B0	
registers.h: ArchIO_PortB		
arch.h: ArchIO.PortC	GPIOC_BASE: 56F826/827 = $11C0	
registers.h: ArchIO_PortC		
arch.h: ArchIO.PortD	GPIOD_BASE: 56F826/827 = $11D0	
registers.h: ArchIO_PortD		
arch.h: ArchIO.PortE (826 Only)	GPIOE_BASE: 56F826 = $11E0	
registers.h: ArchIO_PortE (826 Only)		
arch.h: ArchIO.PortF	GPIOF_BASE: 56F826/827 = $11F0	
registers.h: ArchIO_PortF		
arch.h: ArchIO.PortG (827 Only)	GPIOG_BASE: 56F827 = $1240	
registers.h: ArchIO_PortG (827 Only)		

Table B-1. List of Programmer's Sheets (Continued)

Register Type	Register	Page
Pull-Up Enable Register	(PUR)	B-42
Data Register	(DR)	B-43
Data Direction Register	(DDR)	B-44
Peripheral Enable Register	(PER)	B-45
Interrupt Assert Register	(IAR)	B-46
Interrupt Enable Register	(IENR)	B-47
Interrupt Polarity Register	(IPOLR)	B-48
Interrupt Pending Register	(IPR)	B-49

ADC		
arch.h: ArchIO.Adc	ADCA_BASE: 56F827 = $12C0	
registers.h: ArchIO_Adc		
ADC Control Register 1	(ADCR1)	B-51
ADC Control Register 2	(ADCR2)	B-52
ADC Zero Crossing Control Register	(ADZCC)	B-53
ADC Channel List Register 2	(ADLST2)	B-55
ADC Channel List Register 1	(ADLST1)	B-55
ADC Sample Disable Register	(ADSDIS)	B-56
ADC Status Register	(ADSTAT)	B-57
ADC Limit Status Register	(ADLSTAT)	B-59
ADC Zero Crossing Status Register	(ADZCSTAT)	B-60
ADC Result Registers	(ADRSLT0–7)	B-61
SCI		
arch.h: ArchIO.Sci0	SCI0_BASE: 56F826/827 = $1160	
registers.h: ArchIO_Sci0		
arch.h: ArchIO.Sci1	SCI1_BASE: 56F826/827 = $1170	
registers.h: ArchIO_Sci1		
arch.h: ArchIO.Sci2 (827 Only)	SCI2_BASE: 56F827 = $1180	
registers.h: ArchIO_Sci2 (827 Only)		
SCI Baud Rate Register	(SCIBR)	B-64
SCI Control Register	(SCICR)	B-65
SCI Control Register	(SCICR)	B-66
SCI Data Register	(SCIDR)	B-68

Table B-1. List of Programmer's Sheets (Continued)

Register Type	Register	Page
SPI		
arch.h: ArchIO.Spi0	SPI0 _BASE: 56F826/827 = $1140	
registers.h: ArchIO_Spi0		
arch.h: ArchIO.Spi1	SPI1_BASE: 56F826/827 = $1150	
registers.h: ArchIO_Spi1		
SPI Status and Control Register	(SPSCR)	B-69
SPI Status and Control Register	(SPSCR)	B-69
SPI Data Size Register	(SPDSR)	B-71
SPI Data Receive Register	(SPDRR)	B-72
SPI Data Transmit Register	(SPDTR)	B-73
PCS		
arch.h: ArchIO.Pcs	PCS_BASE: 56F827 = $1190	
registers.h: ArchIO_Pcs		
PCS Base Address Registers	(PCSBAR0-7)	B-74
PCS Option Registers	(PCSOR0-7)	B-75
SSI		
arch.h: ArchIO.Ssi	SSI_BASE: 56F826/827 = $10E0	
registers.h: ArchIO_Ssi		
SSI Transmit Register	(STX)	B-76
SSI Receiver Register	(SRX)	B-77
SSI Control/Status Register	(SCSR)	B-78
SSI Control Register 2	(SCR2)	B-78
SSI Transmit Control Register	(STXCR)	B-80
SSI Receive Control Register	(SRXCR)	B-81
SSI Time Slot Register	(STSR)	B-82
SSI FIFO Control/Status Register	(SFCSR)	B-83
SSI Option Register	(SOR)	B-84
TMR		
arch.h: ArchIO.TimerA (826)	TMRA_BASE: 56F826 =$10A0	
registers.h: ArchIO_TimerA (826)		
arch.h: ArchIO.TimerA (827)	TMRA_BASE: 56F827 = $1200	
registers.h: ArchIO_TimerA (827)		
TMR Control Register	(CTRL)	B-85
TMR Status and Control Register	(SCR)	B-88
TMR Compare Register 1	(CMP1)	B-90

Table B-1. List of Programmer's Sheets (Continued)

Register Type	Register	Page
TMR Compare Register 2	(CMP2)	B-91
TMR Capture Register	(CAP)	B-92
TMR Load Register	(LOAD)	B-93
TMR Hold Register	(HOLD)	B-94
TMR Counter Register	(CNTR)	B-95
TMR Comparator Load One	(CMPLD1)	B-96
TMR Comparator Load Two	(CMPLD2)	B-97
TMR Comparator Status and Control Register	(COMSCR)	B-98
TOD		
arch.h: ArchIO.Tod	TOD_BASE: 56F826/827 = $10C0	
registers.h: ArchIO_Tod		
TOD Control/Status Register	TODCS	B-99
TOD Clock Scaler Load Register	TODCSL	B-100
TOD Seconds Register	TODSEC	B-101
TOD Seconds Alarm Register	TODSAL	B-102
TOD Minutes Register	TODMIN	B-103
TOD Minutes Alarm Register	TODMAL	B-104
TOD Hours Register	TODHR	B-105
TOD Hours Alarm Register	TODHAL	B-106
TOD Days Register	TODDAY	B-107
TOD Days Alarm Register	TODDAL	B-108

OCCS		
arch.h: ArchIO.Pll	CLKGEN_BASE: 56F826/827 = $10F0	
registers.h: ArchIO_Pll		
PLL Control Register	(PLLCR)	B-109
PLL Divide-By Register	(PLLDB)	B-110
PLL Status Register	(PLLSR)	B-111
PLL Select Register	(CLKOSR)	B-112

RESET		
arch.h: ArchIO.Cop	COP_BASE: 56F826/827 = $1120	
registers.h: ArchIO_Cop		
COP Control Register	(COPCTL)	B-113
COP Time-out Register	(COPTO)	B-114
COP Service Register	(COPSRV)	B-115

Table B-1. List of Programmer's Sheets (Continued)

Register Type	Register	Page
SIM		
arch.h: ArchIO.Sim	SYS_BASE: 56F826/827 = $1000	
registers.h: ArchIO_Sim		
COP System Control Register	(SYS_CNTL)	B-116
COP System Status Register	(SYS_STS)	B-118
COP Most Significant Half of JTAG_ID	(MSH_ID)	B-119
COP Least Significant Half of JTAG_ID	(LSH_ID)	B-119
Safe Storage Register	(SSREG0-4)	B-118

Application: _____ Date: _____

_____ Programmer: _____

MEMORY

Bus Control Register (BCR)

arch.h: ArchCore.BusControlReg
regsiters.h: ArchCore_BusControlReg

Wait State P Memory (WSPM[3:0])		
Bit String	Hex Value	Number of Wait States
0000	0	0
0100	4	4
1000	8	8
1100	C	12
All Others		Illegal

Wait State Data Memory (WSX[3:0])		
Bit String	Hex Value	Number of Wait States
0000	0	0
0100	4	4
1000	8	8
1100	C	12
All Others		Illegal

Drive Control	DRV
External memory port pins are only actively driven during external access	0
External memory port pins are actively driven all the time	1

Wait State P Memory (WSPM[3:0])		
Bit String	Hex Value	Number of Wait States
0000	0	0
0100	4	4
1000	8	8
1100	C	12
All Others		Illegal

Bus Control Register (BCR) $FFF9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	DRV	0	WAIT STATE FIELD for EXTERNAL X-MEMORY				WAITSTATE FIELD for EXTERNAL P-MEMORY			
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0

Reserved Bits

MEMORY

Operating Mode Register (OMR)

arch.h: OMR
regsiters.h: OMR

Saturation	SA
Saturation Mode Disabled	0
Saturation Mode Enabled	1

Rounding	R
Rounding Not Allowed	0
Rounding Allowed	1

Stop Delay	SD
Stop Delay Enabled	0
Stop Delay Disabled	1

Condition Code	CC
Codes not set in CCR Register	0
Codes set in CCR Register	1

Nested Looping	NL
Nested Looping Not Allowed	0
Nested Looping Allowed	1

EX	External X Memory
0	External X Memory Disabled
1	External X Memory Enabled

MB	MA	Operating Mode
0	0	Mode 0 - Single Chip
0	1	Mode 1 - Not Supported
1	0	Mode 2 - Not Supported
1	1	Mode 3 - External Memory

Looping Status		
NL In OMR	LF in SR	DO Loop Status
0	0	No DO Loops active
0	1	Single DO loop active
1	0	Caution—Illegal combination. A hardware stack overflow interrupt will occur.
1	1	Two DO loops active

Operating Mode Register (OMR) (CPU Register)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	NL	0	0	0	0	0	0	CC	0	SD	R	SA	EX	0	MB	MA
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*

*MA and MB are latched from the EXTBOOT pin on reset.

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

ITCN

Group Priority Registers (GPR0-3)

arch.h: ArchIO.IntController.GroupPriorityReg[0-3]
registers.h: ArchIO_IntController_GroupPriorityReg+0 *thru* +3
registers.h: ArchIO_IntController_GroupPriorityReg_00 *thru* _03

PRL0–PRL63	Priority Level
0	Disabled
1–7	1 = Lowest Priority Level 7 = Highest Priority Level

Group Priority Register (GPR0) ITCN_BASE+$0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR3				PLR2				PLR1				PLR0		
Write		Illegal Instruction				Factory Use				COP/ Watchdog Reset				Hardware Reset		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR1) ITCN_BASE+$1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR7				PLR6				PLR5				PLR4		
Write		Factory Use				OnCE Trap				Hardward Stack Overflow				Software Interrupt		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR2) ITCN_BASE+$2

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR11				PLR10				PLR9				PLR8		
Write		Boot Flash Interface-826 Reserved-827								$\overline{IRQB}$				$\overline{IRQA}$		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR3) ITCN_BASE+$3

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR15				PLR14				PLR13				PLR12		
Write						Reserved-826 Data Flash Interface-827				Data Flash Interface-826 Upper Prog. Flash Interface AU-827				Prog. Flash Interface -826 Lower Prog. Flash Interface AL-827		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

ITCN

Group Priority Registers (GPR4-7)

arch.h: ArchIO.IntController.GroupPriorityReg[4-7]
registers.h: ArchIO_IntController_GroupPriorityReg+4 *thru* +7
registers.h: ArchIO_IntController_GroupPriorityReg_04 *thru* _07

PRL0–PRL63	Priority Level
0	Disabled
1–7	1 = Lowest Priority Level 7 = Highest Priority Level

Group Priority Register (GPR4) ITCN_BASE+$4

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR19				PLR18				PLR17				PLR16		
Write		GPIO E-826 Reserved-827				GPIO F				Reserved-826 GPIO G-827						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR5) ITCN_BASE+$5

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR23				PLR22				PLR21				PLR20		
Write		GPIOA				GPIOB				GPIOC				GPIOD		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR6) ITCN_BASE+$6

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR27				PLR26				PLR25				PLR24		
Write		SPI 0 Receiver Full and/or Error				SPI 0 Transmitter Empty				SPI 1 Receiver Full and/or Error				SPI 1 Transmitter Empty		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR7) ITCN_BASE+$7

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR31				PLR30				PLR29				PLR28		
Write										SPI 0 Reserved				SPI 0 Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ITCN

Group Priority Registers (GPR8-11)

arch.h: ArchIO.IntController.GroupPriorityReg[8-11]
registers.h: ArchIO_IntController_GroupPriorityReg+8 *thru* +11
registers.h: ArchIO_IntController_GroupPriorityReg_08 *thru* _11

PRL0–PRL63	Priority Level
0	Disabled
1–7	1 = Lowest Priority Level 7 = Highest Priority Level

Group Priority Register (GPR8) ITCN_BASE+$8

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR35 Timer A Channel 1				PLR34 Timer A Channel 0				PLR33 TOD Alarm				PLR32 TOD One Sec		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR9) ITCN_BASE+$9

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR39				PLR38				PLR37 Timer A Channel 3				PLR36 Timer A Channel 2		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR10) :ITCN_BASE+$A

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR43 Reserved-826 SCI 2 Transmit Ready-827				PLR42 Reserved-826 SCI 2 Transmit Complete-827				PLR41				PLR40		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR11) ITCN_BASE+$B

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		PLR47 Transmission Ready				PLR46 SCI 1 Transmission Complete				PLR45 Reserved-826 SCI 2 Receiver Full-827				PLR44 Reserved-826 SCI 2 Receiver Error-827		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

ITCN

Group Priority Registers (GPR12-15)

arch.h: ArchIO.IntController.GroupPriorityReg[12-15]
registers.h: ArchIO_IntController_GroupPriorityReg+12 *thru* +15
registers.h: ArchIO_IntController_GroupPriorityReg_12 *thru* _15

PRL0–PRL63	Priority Level
0	Disabled
1–7	1 = Lowest Priority Level 7 = Highest Priority Level

Group Priority Register (GPR12) ITCN_BASE+$C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read		PLR51				PLR50				PLR49				PLR48		
	Write		SCI 0 Transmission Ready				SCI 0 Transmission Complete				SCI 1 Receiver Full				SCI 1 Receiver Error		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR13) ITCN_BASE+$D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read		PLR55				PLR54				PLR53				PLR52		
	Write		Reserved-826 ADC Conversion Complete-827								SCI 0 Receiver Full				SCI 0 Receiver Error		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR14) ITCN_BASE+$E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read		PLR59				PLR58				PLR57				PLR56		
	Write		SSI Transmit w/o Exception				SSI Transmit w/Exception				SSI Transmit w/o Exception				Reserved-826 ADC Zero Crossing or Limit Error-827		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Group Priority Register (GPR15) ITCN_BASE+$F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read		PLR63				PLR62				PLR61				PLR60		
	Write		Low Voltage Detector				PLL Loss of Lock								SSI w/Exception		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

FLASH

Flash Control Register (FIU_CNTL)

arch.h: ArchIO.ProgramFlash.ControlReg (826)
registers.h: ArchIO_ProgramFlash_ControlReg (826)
arch.h: ArchIO.ProgramFlashLow.ControlReg (827)
registers.h: ArchIO_ProgramFlashLow_ControlReg (827)
arch.h: ArchIO.ProgramFlashHigh.ControlReg (827)
registers.h: ArchIO_ProgramFlashHigh_ControlReg (827)
arch.h: ArchIO.DataFlash.ControlReg (826/827)
registers.h: ArchIO_DataFlash_ControlReg (826/827)
arch.h: ArchIO.BootFlash.ControlReg (826)
registers.h: ArchIO_BootFlash_ControlReg (826)

Non-Volatile Store Cycle Enable	NVSTR
Disabled	0
Enabled	1

Mass Erase Cycle Enable	MAS1
Disabled	0
Enabled	1

Erase Cycle Enable	ERASE
Disabled	0
Enabled	1

Busy	BUSY
Disabled	0
Write Inhibited to FIU Registers– Flash program/erase operation in progress.	1

PROG	Program Cycle Enable
0	Disabled
1	Enabled

YE	Y Address Enable
0	Disabled
1	Enabled

XE	X Address Enable
0	Disabled
1	Enabled

IFREN	Information Block Enable
0	Disabled
1	Enabled

Flash Control Register (FIU_CNTL) FIU_BASE+$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	BUSY	0	0	0	0	0	0	0	0	IFREN	XE	YE	PROG	ERASE	MAS1	NVSTR
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

FLASH

Flash Program Enable Register (FIU_PE)

arch.h: ArchIO.ProgramFlash.ProgramReg (826)
registers.h: ArchIO_ProgramFlash_ProgramReg (826)
arch.h: ArchIO.ProgramFlashLow.ProgramReg (827)
registers.h: ArchIO_ProgramFlashLow_ProgramReg (827)
arch.h: ArchIO.ProgramFlashHigh.ProgramReg (827)
registers.h: ArchIO_ProgramFlashHigh_ProgramReg (827)
arch.h: ArchIO.DataFlash.ProgramReg (826/827)
registers.h: ArchIO_DataFlash_ProgramReg (826/827)
arch.h: ArchIO.BootFlash.ProgramReg (826)
registers.h: ArchIO_BootFlash_ProgramReg (826)

IPE	Intelligent Programming
0	Disabled
1	Enabled

Dumb Programming	DPE
Disabled	0
Enabled	1

ROW	Row Number
0–1024	Row number currently allowed to be programmed.

Flash Program Enable Register (FIU_PE) FIU_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DPE	IPE	0	0	0	0	ROW									
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

FLASH

Flash Erase Enable Register (FIU_EE)

arch.h: ArchIO.ProgramFlash.EraseReg (826)
registers.h: ArchIO_ProgramFlash_EraseReg (826)
arch.h: ArchIO.ProgramFlashLow.EraseReg (827)
registers.h: ArchIO_ProgramFlashLow_EraseReg (827)
arch.h: ArchIO.ProgramFlashHigh.EraseReg (827)
registers.h: ArchIO_ProgramFlashHigh_EraseReg (827)
arch.h: ArchIO.DataFlash.EraseReg (826/827)
registers.h: ArchIO_DataFlash_EraseReg (826/827)
arch.h: ArchIO.BootFlash.EraseReg (826)
registers.h: ArchIO_BootFlash_EraseReg (826)

Dumb Erase	DEE
Disabled	0
Enabled	1

Intelligent Erase	IEE
Disabled	0
Enabled	1

PAGE	Page Number
0–128	Page number currently allowed to be erased.

Flash Erase Enable Register (FIU_EE) FIU_BASE+$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DEE	IEE	0	0	0	0	0	0	0	PAGE						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

FLASH

Flash Address Register (FIU_ADDR)

arch.h: ArchIO.ProgramFlash.AddressReg (826)
registers.h: ArchIO_ProgramFlash_AddressReg (826)
arch.h: ArchIO.ProgramFlashLow.AddressReg (827)
registers.h: ArchIO_ProgramFlashLow_AddressReg (827)
arch.h: ArchIO.ProgramFlashHigh.AddressReg (827)
registers.h: ArchIO_ProgramFlashHigh_AddressReg (827)
arch.h: ArchIO.DataFlash.AddressReg (826/827)
registers.h: ArchIO_DataFlash_AddressReg (826/827)
arch.h: ArchIO.BootFlash.AddressReg (826)
registers.h: ArchIO_BootFlash_AddressReg (826)

Note: This register is designed for program development and error analysis.

A	Current Program Address or Erase Address
$0000	This register is cleared upon any system reset.
Program Address	This register is set to the program/erase address by attempting to write to memory space occupied by flash memory.

Flash Address Register (FIU_ADDR) FIU_BASE+$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	A															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

FLASH

Flash Data Register (FIU_DATA)

arch.h: ArchIO.ProgramFlash.DataReg (826)
registers.h: ArchIO_ProgramFlash_DataReg (826)
arch.h: ArchIO.ProgramFlashLow.DataReg (827)
registers.h: ArchIO_ProgramFlashLow_DataReg (827)
arch.h: ArchIO.ProgramFlashHigh.DataReg (827)
registers.h: ArchIO_ProgramFlashHigh_DataReg (827)
arch.h: ArchIO.DataFlash.DataReg (826/827)
registers.h: ArchIO_DataFlash_DataReg (826/827)
arch.h: ArchIO.BootFlash.DataReg (826)
registers.h: ArchIO_BootFlash_DataReg (826)

Note: This register is designed for program development and error analysis.

D	Last value programmed or value written to initiate an erase
$0000	This register is cleared upon any system reset.
Program Data	Writing to Flash memory space sets this register to the program data value.

Flash Data Register (FIU_DATA) FIU_BASE+$4	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

Reserved Bits

FLASH

Flash Interrupt Enable Register (FIU_IE)

arch.h: ArchIO.ProgramFlash.IntReg (826)
registers.h: ArchIO_ProgramFlash_IntReg (826)
arch.h: ArchIO.ProgramFlashLow.IntReg (827)
registers.h: ArchIO_ProgramFlashLow_IntReg (827)
arch.h: ArchIO.ProgramFlashHigh.IntReg (827)
registers.h: ArchIO_ProgramFlashHigh_IntReg (827)
arch.h: ArchIO.DataFlash.IntReg (826/827)
registers.h: ArchIO_DataFlash_IntReg (826/827)
arch.h: ArchIO.BootFlash.IntReg (826)
registers.h: ArchIO_BootFlash_IntReg (826)

IE	Interrupt Enable Bits	
0	Interrupt Bit Disabled	
1	Bit	Interrupt Bit Enabled
	11	Write to Register Attempt–Busy
	10	Write to FIU_CNTL During Program Erase
	9	Terase or Tmel Timeout
	8	Trcv Timeout
	7	Tprog Timeout
	6	Tpgs Timeout
	5	Tnvh1 Timeout
	4	Tnvh Timeout
	3	Tnvs Timeout
	2	Illegal Flash Read/Write Access Attempt During Erase
	1	Illegal Flash Read/Write Access Attempt During Program
	0	Flash Access Out-Of-Range

Flash Interrupt Enable Register (FIU_IE) FIU_BASE+$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	IE											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

FLASH

Flash Interrupt Source Register (FIU_IS)

arch.h: ArchIO.ProgramFlash.IntSourceReg (826)
registers.h: ArchIO_ProgramFlash_IntSourceReg (826)
arch.h: ArchIO.ProgramFlashLow.IntSourceReg (827)
registers.h: ArchIO_ProgramFlashLow_IntSourceReg (827)
arch.h: ArchIO.ProgramFlashHigh.IntSourceReg (827)
registers.h: ArchIO_ProgramFlashHigh_IntSourceReg (827)
arch.h: ArchIO.DataFlash.IntSourceReg (826/827)
registers.h: ArchIO_DataFlash_IntSourceReg (826/827)
arch.h: ArchIO.BootFlash.IntSourceReg (826)
registers.h: ArchIO_BootFlash_IntSourceReg (826)

IS	Interrupt Source Bits	
0	Interrupt Source Bit Inactive	
1	**Bit**	**Interrupt Source Bit Active: Error Detected**
	11	Write to Register Attempt while Busy
	10	Write to FIU_CNTL During Program Erase
	Bit	**Interrupt Source Bit Active: Timeout Condition Met**
	9	Terase or Tmel Timeout
	8	Trcv Timeout
	7	Tprog Timeout
	6	Tpgs Timeout
	5	Tnvh1 Timeout
	4	Tnvh Timeout
	3	Tnvs Timeout
	Bit	**Interrupt Source Bit Active: Error Detected**
	2	Illegal Flash Read/Write Access Attempt During Erase
	1	Illegal Flash Read/Write Access Attempt During Program
	0	Flash Access Out-Of-Range

Flash Interrupt Source Register (FIU_IS) FIU_BASE+$6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	IS											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

FLASH

Flash Interrupt Pending Register (FIU_IP)

arch.h: ArchIO.ProgramFlash.IntPendingReg (826)
registers.h: ArchIO_ProgramFlash_IntPendingReg (826)
arch.h: ArchIO.ProgramFlashLow.IntPendingReg (827)
registers.h: ArchIO_ProgramFlashLow_IntPendingReg (827)
arch.h: ArchIO.ProgramFlashHigh.IntPendingReg (827)
registers.h: ArchIO_ProgramFlashHigh_IntPendingReg (827)
arch.h: ArchIO.DataFlash.IntPendingReg (826/827)
registers.h: ArchIO_DataFlash_IntPendingReg (826/827)
arch.h: ArchIO.BootFlash.IntPendingReg (826)
registers.h: ArchIO_BootFlash_IntPendingReg (826)

Note: Use this register for indirectly controlling the interrupt service routine.

IP	Interrupt Pending Bits	
0	Interrupt Pending Bit Inactive	
1	**Bit**	**Interrupt Pending Bit Active**
	11	Write to Register Attempt–Busy
	10	Write to FIU_CNTL During Program Erase
	9	Terase or Tmel Timeout
	8	Trcv Timeout
	7	Tprog Timeout
	6	Tpgs Timeout
	5	Tnvh1 Timeout
	4	Tnvh Timeout
	3	Tnvs Timeout
	2	Illegal Flash Read/Write Access Attempt During Erase
	1	Illegal Flash Read/Write Access Attempt During Program
	0	Flash Access Out-Of-Range

Flash Interrupt Pending Register (FIU_IP) FIU_BASE+$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	IP											
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

FLASH

Flash Clock Divisor Register (FIU_CLKDIVISOR)

arch.h: ArchIO.ProgramFlash.DivisorReg (826)
registers.h: ArchIO_ProgramFlash_DivisorReg (826)
arch.h: ArchIO.ProgramFlashLow.DivisorReg(827)
registers.h: ArchIO_ProgramFlashLow_DivisorReg (827)
arch.h: ArchIO.ProgramFlashHigh.DivisorReg (827)
registers.h: ArchIO_ProgramFlashHigh_DivisorReg (827)
arch.h: ArchIO.DataFlash.DivisorReg (826/827)
registers.h: ArchIO_DataFlash_DivisorReg (826/827)
arch.h: ArchIO.BootFlash.DivisorReg (826)
registers.h: ArchIO_BootFlash_DivisorReg (826)

N	Clock Divisor
0–15	Value of Clock Divisor = $2^{(N+1)}$ (Values = 2, 4, 8, 16, . . . 65536)

Flash Clock Divisor Register (FIU_CKDIVISOR) FIU_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	N			
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

FLASH

Flash Terase Limit Register (FIU_TERASEL)

arch.h: ArchIO.ProgramFlash.TimerEraseReg (826)
registers.h: ArchIO_ProgramFlash_TimerEraseReg (826)
arch.h: ArchIO.ProgramFlashLow.TimerEraseReg (827)
registers.h: ArchIO_ProgramFlashLow_TimerEraseReg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerEraseReg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerEraseReg (827)
arch.h: ArchIO.DataFlash.TimerEraseReg (826/827)
registers.h: ArchIO_DataFlash_TimerEraseReg (826/827)
arch.h: ArchIO.BootFlash.TimerEraseReg (826)
registers.h: ArchIO_BootFlash_TimerEraseReg (826)

TERASEL	Timer Erase Limit
0–127	Terase = $2^{(N+1)}$ x (TERASEL + 1) where N = the Clock Divisior Value Default Erase Time = 2^{16} x 2^4 x 25ns = 26.2ms

Flash Terase Limit Register (FIU_TERASEL) FIU_BASE+$9	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	TERASEL						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

FLASH

Flash Time Limit Register (FIU_TMEL)

arch.h: ArchIO.ProgramFlash.TimerMassEraseReg (826)
registers.h: ArchIO_ProgramFlash_TimerMassEraseReg (826)
arch.h: ArchIO.ProgramFlashLow.TimerMassEraseReg (827)
registers.h: ArchIO_ProgramFlashLow_TimerMassEraseReg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerMassEraseReg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerMassEraseReg (827)
arch.h: ArchIO.DataFlash.TimerMassEraseReg (826/827)
registers.h: ArchIO_DataFlash_TimerMassEraseReg (826/827)
arch.h: ArchIO.BootFlash.TimerMassEraseReg (826)
registers.h: ArchIO_BootFlash_TimerMassEraseReg (826)

TMEL	Timer Mass Erase Limit
0–255	Tme = $2^{(N+1)}$ x (TMEL + 1) where N = the Clock Divisior Value Default Erase Time = 2^{16} x 2^4 x 25ns = 26.2ms

Flash Tme Limit Register (FIU_TMEL) FIU_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	TMEL							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

FLASH

Flash Tnvs Limit Register (FIU_TNVSL)

arch.h: ArchIO.ProgramFlash.TimerNVStorageReg (826)
registers.h: ArchIO_ProgramFlash_TimerNVStorageReg (826)
arch.h: ArchIO.ProgramFlashLow.TimerNVStorageReg (827)
registers.h: ArchIO_ProgramFlashLow_TimerNVStorageReg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerNVStorageReg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerNVStorageReg (827)
arch.h: ArchIO.DataFlash.TimerNVStorageReg (826/827)
registers.h: ArchIO_DataFlash_TimerNVStorageReg (826/827)
arch.h: ArchIO.BootFlash.TimerNVStorageReg (826)
registers.h: ArchIO_BootFlash_TimerNVStorageReg (826)

TNVSL	Timer Non-Volatile Storage Limit
0–2047	Tnvs = (TNVSL + 1) Tnvs default value = 25ns x 2^8 = 6.4µs

Flash Tnvs Limit Register (FIU_TNVSL) FIU_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	TNVSL										
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Reserved Bits

FLASH

Flash Tpgs Limit Register (FIU_TPGSL)

arch.h: ArchIO.ProgramFlash.TimerProgramSetupReg (826)
registers.h: ArchIO_ProgramFlash_TimerProgramSetupReg (826)
arch.h: ArchIO.ProgramFlashLow.TimerProgramSetupReg (827)
registers.h: ArchIO_ProgramFlashLow_TimerProgramSetupReg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerProgramSetupReg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerProgramSetupReg (827)
arch.h: ArchIO.DataFlash.TimerProgramSetupReg (826/827)
registers.h: ArchIO_DataFlash_TimerProgramSetupReg (826/827)
arch.h: ArchIO.BootFlash.TimerProgramSetupReg (826)
registers.h: ArchIO_BootFlash_TimerProgramSetupReg (826)

TPGSL	Timer Erase Limit
0–4095	Tpgs = (TPGSL + 1) Tpgs default value = 25ns x 2^9 = 12.8μs

Flash Tpgs Limit Register (FIU_TPGSL) FIU_BASE+$C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	TPGSL											
	Write																
	Reset	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Reserved Bits

FLASH

Flash Tprog Limit Register (FIU_TPROGL)

arch.h: ArchIO.ProgramFlash.TimerProgramReg (826)
registers.h: ArchIO_ProgramFlash_TimerProgramReg (826)
arch.h: ArchIO.ProgramFlashLow.TimerProgramReg (827)
registers.h: ArchIO_ProgramFlashLow_TimerProgramReg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerProgramReg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerProgramReg (827)
arch.h: ArchIO.DataFlash.TimerProgramReg (826/827)
registers.h: ArchIO_DataFlash_TimerProgramReg (826/827)
arch.h: ArchIO.BootFlash.TimerProgramReg (826)
registers.h: ArchIO_BootFlash_TimerProgramReg (826)

TPROGL	Timer Program Limit
0–16383	Tprog = (TPROGL + 1) Default Program Time Limit = 2^{10} x 25ns = 25.6μs.

Flash Tprog Limit Register (FIU_TPROGL) FIU_BASE+$D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	TPROGL													
	Write																
	Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Reserved Bits

FLASH

Flash Tnvh Limit Register (FIU_TNVHL)

arch.h: ArchIO.ProgramFlash.TimerNVHoldReg (826)
registers.h: ArchIO_ProgramFlash_TimerNVHoldReg (826)
arch.h: ArchIO.ProgramFlashLow.TimerNVHoldReg (827)
registers.h: ArchIO_ProgramFlashLow_TimerNVHoldReg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerNVHoldReg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerNVHoldReg (827)
arch.h: ArchIO.DataFlash.TimerNVHoldReg (826/827)
registers.h: ArchIO_DataFlash_TimerNVHoldReg (826/827)
arch.h: ArchIO.BootFlash.TimerNVHoldReg (826)
registers.h: ArchIO_BootFlash_TimerNVHoldReg (826)

TNVHL	Timer Non-Volatile Hold Limit
0–2047	Tnvh = (TNVHL + 1) Default Tnvh Value = 2^8 x 25ns = 6.4μs

Flash Tnvh Limit Register (FIU_TNVHL) FIU_BASE+$E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	TNVHL										
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Reserved Bits

FLASH

Flash Tnvh1 Limit Register (FIU_TNVH1L)

arch.h: ArchIO.ProgramFlash.TimerNVHold1Reg (826)
registers.h: ArchIO_ProgramFlash_TimerNVHold1Reg (826)
arch.h: ArchIO.ProgramFlashLow.TimerNVHold1Reg (827)
registers.h: ArchIO_ProgramFlashLow_TimerNVHold1Reg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerNVHold1Reg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerNVHold1Reg (827)
arch.h: ArchIO.DataFlash.TimerNVHold1Reg (826/827)
registers.h: ArchIO_DataFlash_TimerNVHold1Reg (826/827)
arch.h: ArchIO.BootFlash.TimerNVHold1Reg (826)
registers.h: ArchIO_BootFlash_TimerNVHold1Reg (826)

TNVH1L	Timer Non-Volatile Hold 1 Limit
0–32767	Tnvh1 = (TNVH1L + 1) Tnvh1 Default Value = 2^{12} x 25ns = 102.4µs

Flash Tnvh1 Limit Register (FIU_TNVH1L) FIU_BASE+$F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	TNVH1L														
	Write																
	Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Reserved Bits

FLASH

Flash Trcv Limit Register (FIU_TRCVL)

arch.h: ArchIO.ProgramFlash.TimerRecoveryReg (826)
registers.h: ArchIO_ProgramFlash_TimerRecoveryReg (826)
arch.h: ArchIO.ProgramFlashLow.TimerRecoveryReg (827)
registers.h: ArchIO_ProgramFlashLow_TimerRecoveryReg (827)
arch.h: ArchIO.ProgramFlashHigh.TimerRecoveryReg (827)
registers.h: ArchIO_ProgramFlashHigh_TimerRecoveryReg (827)
arch.h: ArchIO.DataFlash.TimerRecoveryReg (826/827)
registers.h: ArchIO_DataFlash_TimerRecoveryReg (826/827)
arch.h: ArchIO.BootFlash.TimerRecoveryReg (826)
registers.h: ArchIO_BootFlash_TimerRecoveryReg (826)

TRCVL	Timer Recovery Limit
0–511	Trcv = (TRCVL + 1) Trcv Default Value = 2^6 x 25ns = 1.6μs

Flash Trcv Limit Register (FIU_TRCVL) FIU_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	TRCVL								
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Reserved Bits

GPIO

GPIO Pull-Up Enable Register (PUR)

arch.h: ArchIO.Port *?*. PullUpReg
registers.h: ArchIO_Port *?*_PullUpReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

Pull-Up Enable Functions					
Peripheral Out Enable	PER	PUR	DDR	GPIO Pin State	GPIO Pin Pull-Up
x	0	0	0	Input	Disabled
x	0	0	1	Output	Disabled
x	0	1	0	Input	Enabled
x	0	1	1	Output	Disabled
0	1	x	x	Output	Disabled
0	1	x	x	Output	Disabled
1	1	x	x	Input	Disabled
1	1	x	x	Input	Enabled

PU	Pull-Up
0	Pull-Up Disabled
1	DDR and PER = 0: Pull-Up is enabled. Each bit performs the pull-up function on the corresponding GPIO pin if the GPIO is configured as an input. PER = 1: Pull-Up is controlled by the peripheral output enable and the PUR.

Note 1: If the GPIO pin is configured as an output, the PUR is not recognized.

Note 2: PUR is set to 1 on processor reset.

Note 3: Refer to the above *Pull-Up Enable Functions* table for all possible combinations.

Pull-Up Enable Register (PUR) GPIO_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	PU							
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Pull-Up Enable* Register (PUR) GPIO_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

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Data Register (DR)

arch.h: ArchIO.Port *?*.DataReg
registers.h: ArchIO_Port *?*_DataReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

Data
This GPIO Pin / IPBus interface holds data coming from the GPIO pin or the IPBus.

Data Register (DR) GPIO:_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	D							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Data Register (DR)* GPIO:_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

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Data Direction Register (DDR)

arch.h: ArchIO.Port *?*. DataDirectionReg
registers.h: ArchIO_Port *?*_DataDirectionReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

Pull-Up Enable Functions					
Peripheral Out Enable	PER	PUR	DDR	GPIO Pin State	GPIO Pin Pull-Up
x	0	0	0	Input	Disabled
x	0	0	1	Output	Disabled
x	0	1	0	Input	Enabled
x	0	1	1	Output	Disabled
0	1	x	x	Output	Disabled
0	1	x	x	Output	Disabled
1	1	x	x	Input	Disabled
1	1	x	x	Input	Enabled

DD	Data Direction
0	PUR = 1: GPIO pin is an input with pull-up device. PUR = 0: GPIO pin is an input without pull-up device.
1	The GPIO pin is an output.
Note 1:	Refer to the above *Pull-Up Enable Functions* table for all possible combinations on using the DDR register.
Note 2:	Each bit performs the pull-up function on the corresponding GPIO pin.

Data Direction Register (DDR) GPIO_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	0	0	0	DD							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Data Direction* Register (DDR) GPIO_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

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Peripheral Enable Register (PER)

arch.h: ArchIO.Port *?*.PeripheralReg
registers.h: ArchIO_Port *?*_PeripheralReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

PE	Peripheral Enable
0	DDR determines the data flow direction: DDR = 0: GPIO pin is input only. DDR = 1: GPIO pin is output only.
1	PER masters the GPIO pin. This mastership includes configuring the GPIO pin as: – A required input (with or without pull-up), or – A required output (depending on peripheral output enable status, and includes data transfers from the GPIO pin to the peripheral.)
Note 1:	Refer to the above *Pull-Up Enable Functions* table for all possible combinations on using the PER register.
Note 2:	Each bit determines the peripheral function on the corresponding GPIO pin.

Pull-Up Enable Functions					
Peripheral Out Enable	PER	PUR	DDR	GPIO Pin State	GPIO Pin Pull-Up
x	0	0	0	Input	Disabled
x	0	0	1	Output	Disabled
x	0	1	0	Input	Enabled
x	0	1	1	Output	Disabled
0	1	x	x	Output	Disabled
0	1	x	x	Output	Disabled
1	1	x	x	Input	Disabled
1	1	x	x	Input	Enabled

Peripheral Enable Register (PER) GPIO_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	PE							
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Peripheral Enable* Register (PER) GPIO_BASE+$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PE															
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Interrupt Assert Register (IAR)

arch.h: ArchIO.Port *?*. IntAssertReg
registers.h: ArchIO_Port *?*_IntAssertReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

IA	Interrupt Assert
0	Interrupt is cleared
1	An interrupt is asserted.
Note: The IAR register is only for software testing.	

Interrupt Assert Register (IAR) GPIO:_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	IA							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interrupt Assert* Register (IAR) GPIO:_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

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Interrupt Enable Register (IENR)

arch.h: ArchIO.Port *?*. IntEnableReg
registers.h: ArchIO_Port *?*_IntEnableReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

IEN	Interrupt Enable
0	Interrupt detection disabled.
1	Interrupt detection enabled.

Interrupt Enable Register (IENR) GPIO_BASE+$5	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	IEN							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interrupt Enable* Register (IENR) GPIO_BASE+$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	Read	IEN														
	Write															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Interrupt Polarity Register (IPOLR)

arch.h: ArchIO.Port *?*. IntPolarityReg
registers.h: ArchIO_Port *?*_IntPolarityReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

	IPOL	Interrupt Polarity
When register IENR = 1	0	The interrupt at the GPIO pin is active high.
	1	The interrupt at the GPIO pin is active low.
When register IENR = 0	–	Register IPOLR is disabled

Interrupt Polarity Register (IPOLR) GPIO_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	0	IPOL							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interrupt Polarity* Register (IPOLR) GPIO_BASE+$6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	Read	IPOL														
	Write															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Interrupt Pending Register (IPR)

arch.h: ArchIO.Port *?*. IntPendingReg
registers.h: ArchIO_Port *?*_IntPendingReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

Interrupt Pending Register	IPR
Incoming interrupts do not exist.	0
A set bit indicates which pin caused an incoming interrupt.	1

To Clear the IPR Register	
Software Interrupts	Write zeros into the IAR register.
External Interrupts	Write zeros into the IESR register.

Interrupt Pending Register (IPR) GPIO:_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	IPR							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interrupt Pending* Register (IPR) GPIO:_BASE+$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	Read	IPR														
	Write															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*Ports A and G 827 Only

Reserved Bits

GPIO

GPIO Interrupt Edge Sensitive Register (IESR)

arch.h: ArchIO.Port *?*. IntEdgeSensReg
registers.h: ArchIO_Port *?*_IntEdgeSensReg

Where *?* = A, B, C, D, E, or F (826)
Where *?* = A, B, C, D, F, or G (827)

IES	Interrupt Edge Sensitive
0	Interrupts are not recorded
1	IESR recorded an interrupt. IESR records interrupts when the edge detector circuit detects an edge and the IENR is set to 1.

Interrupt Edge-Sensitive Register (IESR) GPIO_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	IES							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interrupt Edge-Sensitive* Register (IESR) GPIO_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

*Ports A and G 827 Only

Reserved Bits

Application: _____ Date: _____

Programmer: _____

ADC

ADC Control Register 1 (ADCR1)

arch.h: ArchIO.Adc.Control1Reg (826 only)
registers.h: ArchIO_Control1Reg (827 only)

Scan Mode	SMODE		
Once Sequential	0	0	0
Once Simultaneous	0	0	1
Loop Sequential	0	1	0
Loop Simultaneous	0	1	1
Triggered Sequential	1	0	0
Triggered Simultaneous	1	0	1
Reserved	1	1	0
Reserved	1	1	1

SYNC Select	SYNC
Conversion initiated by a write to START bit only	0
Use the SYNC input or START bit to initiate conversion	1

ADC Stop	STOP
Normal operation	0
ADC Stop command is issued	1

ADC Start	START
No action	0
ADC Start command is issued	1

Power-Down	PDN
Power-up ADC analog core command is issued	0
Power-down ADC the analog core command	1

Channel Configure CHNCFG			
Single Ended Input*	Single End-ed Channels	Differential Input*	Differential Channel Pairs and Polarity
Bit 4 = 0	AN0 AN1	Bit 4 = 1	AN0 is +, AN1 is -
Bit 5 = 0	AN2 AN3	Bit 5 = 1	AN2 is +, AN3 is -
Bit 6 = 0	AN4 AN5	Bit 6 = 1	AN4 is +, AN5 is -
Bit 7 = 0	AN6 AN7	Bit 7= 1	AN6 is +, AN7 is -
*Each bit acts independently.			

LLMTIE	Low Limit Interrupt Enable
0	Interrupt disabled
1	Interrupt enabled

HLMTIE	High Limit Interrupt Enable
0	Interrupt disabled
1	Interrupt enabled

ZCIE	Zero Crossing Interrupt Enable
0	Interrupt disabled
1	Interrupt enabled

EOSIE	End of Scan Interrupt Enable
0	Interrupt disabled
1	Interrupt enabled

ADC Control Register 1 (ADCR1) ADC_BASE+$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PDN	ADC STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	0	0	0	0	0	0	SMODE	
	Write			START													
	Reset	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Reserved Bits

ADC

ADC Control Register 2 (ADCR2)

arch.h: ArchIO.Adc.Control2Reg (826 only)
registers.h: ArchIO_Control2Reg (827 only)

DIV	Clock Divisor Select
0–15	Clock Divisor Select Value = N = DIV[4:] + 1 $F_{ADC} = (F_{IPR}) / 2N$ F_{ADC} = Analog-to-Digital Converter Frequency F_{IPR} = Interface Peripheral Bus Clock Frequency

ADC Control Register (ADCR2) ADC_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	DIV				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0		0	1	1	1

Reserved Bits

ADC

ADC Zero Crossing Control Register (AZCC1-2)

arch.h: ArchIO.Adc.ZeroCrossControlReg1 (827 only)
registers.h: ArchIO_ZeroCrossControlReg1 (827 only)
arch.h: ArchIO.Adc.ZeroCrossControlReg2 (827 only)
registers.h: ArchIO_ZeroCrossControlReg2 (827 only)

ZCEx	Zero Crossing Enable
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
Note:	ZCE0 uses sample in ADRSLT0 ZCE7 uses sample in ADRSLT9

ADC Zero Crossing Control Register (ADZCC1) ADC_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

ADC Zero Crossing Control Register (ADZCC2) X:ADC_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	ZCE9		ZCE8	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

ADC Channel List Registers (ADLST1-5)

This register is continued on the following page.

56F827 Only

Input Channel Number to be Converted									
SAMPLEx[6:0] "x" designates the sample number	SAMPLEn[6:0]							Single Ended	Differential
	6	5	4	3	2	1	0		
000	0	0	0	0	0	0	0	AN0	–
001	0	0	0	0	0	0	1	AN1	–
010	0	0	0	0	0	1	0	AN2	–
011	0	0	0	0	0	1	1	AN3	–
100	0	0	0	0	1	0	0	AN4	–
000	0	0	0	0	1	0	1	AN5	–
001	0	0	0	0	1	1	0	AN6	–
010	0	0	0	0	1	1	1	AN7	–
011	0	0	0	1	0	0	0	AN8	–
100	0	0	0	1	0	0	1	AN9	–
	0	Others						Reserved	–
		AN_{even}+			AN_{odd}-				
	1	0	0	0	x	x	x	–	AN0+, AN0-
	1	0	0	1	x	x	x	–	AN2+, AN0-
	1	0	1	0	x	x	x	–	AN4+, AN0-
	1	0	1	1	x	x	x	–	AN6+, AN0-
	1	1	0	0	x	x	x	–	AN8+, AN0-
	1	x	x	x	0	0	0	–	ANe+, AN1-
	1	x	x	x	0	0	1	–	ANe+, AN3-
	1	x	x	x	0	1	0	–	ANe+, AN5-
	1	x	x	x	0	1	1	–	ANe+, AN7-
	1	x	x	x	1	0	0	–	ANe+, AN9-
	1	Others			Others			–	Reserved
	For example, to configure a sample as differential pair of AN4+, AN3-, SAMPLE[6:0] would be								
	1	0	1	0	0	0	1	-	AN4+, AN3-
LEGEND:AN_{even}+, ANe+ : any of the ADC PLUS CHANNEL (even only) -AN0, AN2, AN4, AN6, AN8 AN_{odd}-, ANo- : any of the ADC MINUS CHANNEL (odd only) -AN1, AN3, AN5, AN7, AN9									

SAMPLEx
Sequential Sampling
Sequential scan of inputs proceeds in order from SAMPLE0–SAMPLE9
Configuring Channel Monitoring • Select sequential,loop or triggered Scan Mode with SMODE[1:0] bits in register ADCR1. • Select Sample range in register ADSDIS with DSx bits. Sampling stops on encountering a set DSx bit. Sequential sampling starts at SAMPLE0. • Observe differential input restrictions

Reserved Bits

ADC

ADC Channel List Registers (ADLST1-5) Continued

arch.h: ArchIO.Adc.ChannelList ?Reg
registers.h: ArchIO_ChannelList ?Reg

Where ? = 1, 2, 3, 4, or 5

ADC Channel List Register (ADLST1) ADC_BASE+$4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	SAMPLE1							0	SAMPLE0						
	Write																
	Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

ADC Channel List Register (ADLST3) ADC_BASE+$6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	SAMPLE5							0	SAMPLE4						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC Channel List Register (ADLST2) ADC_BASE+$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	SAMPLE3							0	SAMPLE2						
	Write																
	Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

ADC Channel List Register (ADLST4) ADC_BASE+$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	SAMPLE7							0	SAMPLE6						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC Channel List Register (ADLST5) ADC_BASE+$8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	SAMPLE9							0	SAMPLE8						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

ADC Sample Disable Registers (ADSDIS)

arch.h: ArchIO.Adc.DisableReg (827 only)
registers.h: ArchIO_DisableReg (827 only)

ZCS	Zero Crossing Status Register
0	1. A sign change did not occur in a comparison between the current channelx result and the previous channelx result, or 2. Zero Crossing Control is disabled for channelx in the Zero Crossing Control Register, ADSDIS.
1	In a comparison between the current channelx result and the previous channelx result, a sign change condition occurred as defined in the Zero Crossing Control Register (ADSDIS). **Note:** To clear a specific ZCS[7:0] bit, write a value of 1 to that bit.

ADC Sample Disable Register (ADSDIS) ADC_BASE+$9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

ADC Status Registers 1 (ADSTAT1)

arch.h: ArchIO.Adc.StatusReg1 (827 only)
registers.h: ArchIO_StatusReg1 (827 only)

End of Scan Interrupt	EOSI
A scan cycyle has not been completed, no end of scan IRQ pending	0
A scan cycle has been completed, end of scan IRQ pending	1

Power-Down Status	PDNS
Normal operation	0
ADC analog core is powered down	1

Conversion in Progress	CIP
Idle state	0
Scan cycle is in progress. The ADC will itgnore all sync pulses or Start commands	1

HLMTI	High Limit Interrupt
0	No High Limit IRQ
1	High Limit exceeded, IRQ pending if HLMTI is set

LLMTI	Low Limit Interrupt
0	No Low Limit IRQ
1	Low Limit exceeded, IRQ pending if LLMTI is set

ZCI	Zero Crossing Interrupt
0	No ZCI IRQ
1	Zero Crossing encountered, IRQ pending if ZCI is set

ADC Status Register1 (ADSTAT1) ADC_BASE+$A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CIP	PDNS	0	0	EOSI	ZCI	LLMTI	HLMTI	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

ADC

ADC Status Registers 2 (ADSTAT2)

arch.h: ArchIO.Adc.StatusReg2 (827 only)
registers.h: ArchIO_StatusReg2 (827 only)

RDY	Ready Channel 9-0
These bits indicate channels 9-0 are ready to be read. These bits are cleared after a read from the respective Results regsiter.	
0	Channel not ready or has been read
1	Channel ready to be read

ADC Status Register2 (ADSTAT2) ADC_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	RDY9	RDY8	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY2	RDY0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

ADC Limit Status Registers (ADLLSTAT–ADHLSTAT)

arch.h: ArchIO.Adc.LimitStatusReg1 (827 only)
registers.h: ArchIO_LimitStatusReg1 (827 only)

arch.h: ArchIO.Adc.LimitStatusReg2 (827 only)
registers.h: ArchIO_LimitStatusReg2 (827 only)

Low Limit – HLMT

The low limit value the result (ADRSLT0–9) is compared against.

ADC Limit Status Register (ADLLSTAT) ADC_BASE+$C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	LLS9	LLS8	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

High Limit – HLMT

The high limit value the result (ADRSLT0–9) is compared against.

ADC Limit Status Register (ADHLSTAT) ADC_BASE+$D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	Read	0	0	0	0	0	0	HLS9	HLS8	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1
	Write															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

ADC

ADC Zero Crossing Limit Status Register (ADZCSTAT)

arch.h: ArchIO.Adc.ZeroCrossStatusReg (827 only)
registers.h: ArchIO_ZeroCrossStatusReg (827 only)

ZCSn	Zero Crossing Status
1	Channel n result equals offset n value
0	Channel n result not equal to offset value

ADC Zero Crossing Status Register (ADZCSTAT) ADC_BASE+$E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	ZCS9	ZCS8	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

ADC

ADC Result Register (ADRSLTn)

arch.h: ArchIO.Adc.ResultReg*?* (827 only)
registers.h: ArchIO_ResultReg*?* (827 only)

Where *?* = 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9

RSLTn	Digital Result of Conversion
ADRSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the ADRSLT can be used directly. As a signed integer, there is the option to right shift with the sign extended three places and interpret the number that way, or accept the number as presented, knowing there are missing codes. The lower three bits will always be zero.	

Sign Extend	SEXT
Provides a positive result. When positive results are required the respective offset register must be set to a value of zero.	0
Provides a negative result.	1

ADC Result Register (ADRSLTn) ADC_BASE+$?	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SEXT	RSLTn												0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC Address Schedule
ADC Result Register 0–Address: ADC_BASE+$0F
ADC Result Register 1–Address: ADC_BASE+$10
ADC Result Register 2–Address: ADC_BASE+$11
ADC Result Register 3– Address: ADC_BASE+$12
ADC Result Register 4–Address: ADC_BASE+$13
ADC Result Register 5–Address: ADC_BASE+$14
ADC Result Register 6–Address: ADC_BASE+$15
ADC Result Register 7–Address: ADC_BASE+$16
ADC Result Register 8–Address: ADC_BASE+$17
ADC Result Register 9–Address: ADC_BASE+$18

Reserved Bits

ADC

ADC Low/High Limit Status Registers (ADLLMTn –ADHLMTn)

arch.h: ArchIO.Adc.LowLimitReg*?* (827 only)
registers.h: ArchIO_LowLimitReg*?* (827 only)
arch.h: ArchIO.Adc.HighLimitReg*?* (827 only)
registers.h: ArchIO_HighLimitReg*?* (827 only)

Where *?* = 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9

ADC Result Register (ADLLMTn) ADC_BASE+$?	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	LLMTn												0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC Low Limit Result Address Schedule
ADC Low Limit Register 0–Address: ADC_BASE+$19
ADC Low Limit Register 1–Address: ADC_BASE+$1A
ADC Low Limit Register 2–Address: ADC_BASE+$1B
ADC Low Limit Register 3– Address:ADC_BASE+$1C
ADC Low Limit Register 4–Address: ADC_BASE+$1D
ADC Low Limit Register 5–Address: ADC_BASE+$1E
ADC Low Limit Register 6–Address: ADC_BASE+$1F
ADC Low Limit Register 7–Address: ADC_BASE+$20
ADC Low Limit Register 8–Address: ADC_BASE+$21
ADC Low Limit Register 9–Address: ADC_BASE+$22

ADC Result Register (ADHLMTn) ADC_BASE+$?	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	HLMTn												0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC High Limit Result Address Schedule
ADC High Limit Register 0–Address: ADC_BASE+$23
ADC High Limit Register 1–Address: ADC_BASE+$24
ADC High Limit Register 2–Address: ADC_BASE+$25
ADC High Limit Register 3– Address:ADC_BASE+$26
ADC High Limit Register 4–Address: ADC_BASE+$27
ADC High Limit Register 5–Address: ADC_BASE+$28
ADC High Limit Register 6–Address: ADC_BASE+$29
ADC High Limit Register 7–Address: ADC_BASE+$2A
ADC High Limit Register 8–Address: ADC_BASE+$2B
ADC High Limit Register 9–Address: ADC_BASE+$2C

Reserved Bits

ADC

ADC Offset Register

arch.h: ArchIO.Adc.OffsetReg*?* (827 only)
registers.h: ArchIO_OffsetReg*?* (827 only)

Where *?* = 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9

Offset Value – OFFSET
The OFFSET value is subtracted from the ADC result. To obtain unsigned results, program the respective offset register with a value of $0000 to give a result range of $0000 to $7FF8.

ADC Offset Register (ADOFSn) ADC_BASE+$?	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	OFFSETn												0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC Offset Registers Address Schedule
ADC Offset Register 0–Address: ADC_BASE+$2D
ADC Offset Register 1–Address: ADC_BASE+$2E
ADC Offset Register 2–Address: ADC_BASE+$2F
ADC Offset Register 3– Address:ADC_BASE+$30
ADC Offset Register 4–Address: ADC_BASE+$31
ADC Offset Register 5–Address: ADC_BASE+$32
ADC Offset Register 6–Address: ADC_BASE+$33
ADC Offset Register 7–Address: ADC_BASE+$34
ADC Offset Register 8–Address: ADC_BASE+$35
ADC Offset Register 9–Address: ADC_BASE+$36

Reserved Bits

SCI

SCI Baud Rate Register (SCIBR)

arch.h: ArchIO.Sci *?*.BaudRateReg
registers.h: ArchIO_Sci *?*_BaudRateReg

Where *?* = 0 or 1 (826)
Where *?* = 0, 1 or 2 (827)

SBR	SCI Baud Rate
0	Baud rate generator disabled
1 – 8191	$\text{Baud Rate} = \frac{\text{SCI Module Clock}}{16 \times \text{SBR}}$
Note: The baud rate generator is disabled until the TE or RE bit in register SCICR is set for the first time after reset.	

SCI Baud Rate Register (SCIBR) SCI_BASE+$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	SBR												
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Reserved Bits

SCI

SCI Control Register (SCICR)

This register is continued on the following page.

arch.h: ArchIO.Sci *?*.ControlReg
registers.h: ArchIO_Sci *?*_ControlReg

Where *?* = 0 or 1 (826)
Where *?* = 0, 1 or 2 (827)

Loop Select	LOOP
Normal operation enabled	0
Loop operation enabled. See Notes below.	1

Stop in Wait Mode	SWAI
SCI enabled in Wait mode	0
SCI disabled in Wait mode	1

Receiver Source	RSRC
Receiver input internally connected to transmitter output	0
Receiver input connected to TXD pin	1

Data Format Mode	M
Eight data characters, one start bit, one STOP bit	0
Nine data characters, one start bit, one STOP bit	1

WAKE	Wake-Up Condition
0	Idle Line Wake-Up
1	Address Mark Wake-Up

POL	Polarity Bit
0	Do not invert transmit and receive data bits (Normal mode)
1	Invert transmit and receive data bits (Inverted mode)

PE	Parity Enable Bit
0	Parity function disabled
1	Parity function enabled

PT	Parity Type Bit
0	Even parity
1	Odd parity

SCI Control Register (SCICR) SCI_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	RIE	REIE	TE	RE	RWU	SBK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: To use the internal loop function, the transmitter and receiver must be enabled: RE = 1, TE = 1.

Note: When LOOP = 1, the RSRC bit determines the internal feedback path for the receiver. (See the Loop Functions table at the right for details.)

LOOP	RSRC	Loop Functions
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

SCI

SCI Control Register (SCICR) Continued

arch.h: ArchIO.Sci ?.ControlReg
registers.h: ArchIO_Sci ?_ControlReg

Where ? = 0 or 1 (826)
Where ? = 0, 1 or 2 (827)

Transmitter Empty Interrupt Enable	TEIE
TDRE interrupt requests disabled	0
TDRE interrupt requests enabled	1

Transmitter Idle Interrupt Enable	TIIE
TI interrupt requests disabled	0
TI interrupt requests enabled	1

Receive Full Interrupt Enable	RIE
RDRF and OR interrupt requests disabled	0
RDRF and OR interrupt requests enabled	1

Receive Error Interrupt Enable	REIE
Error interrupt requests disabled	0
Error interrupt requests enabled	1

TE	Transmitter Enable
0	Transmitter disabled—TXD pin is in high-impedance state
1	Transmitter enabled

RE	Receiver Enable
0	Receiver disabled
1	Receiver enabled

RWU	Receiver Wake-Up
0	Normal operation
1	Transmit break characters

SBK	Send Break
0	No break characters transmitted
1	Transmit break characters

SCI Control Register (SCICR) SCI_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	RIE	REIE	TE	RE	RWU	SBK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SCI

SCI Status Register (SCISR)

arch.h: ArchIO.Sci ?.StatusReg
registers.h: ArchIO_Sci ?_StatusReg

Where ? = 0 or 1 (826)
Where ? = 0, 1 or 2 (827)

Transmit Data Register Empty Flag	TDRE
No character transferred to transmit shift register	0
Character transferred to transmit shift register; transmit data register empty	1

Transmitter Idle Flag	TIDLE
Transmission in progress	0
No transmission in progress	1

Receive Data Register Full Flag	RDRF
Data not available in SCI data register	0
Received data available in SCI data register	1

Receiver Idle Line Flag	RIDLE
Receiver input is either active now or has never become active since the RIDLE flag was last cleared	0
Receiver input has become idle (after receiving a valid frame)	1

Receive Data Register Full Flag	RDRF
Data not available in SCI data register	0
Received data available in SCI data register	1

OR	Overrun Flag
0	No overrun
1	Overrun

NF	Noise Flag
0	No Noise
1	Noise

FE	Framing Error Flag
0	No framing error
1	Framing error

PF	Parity Error Flag
0	No parity error
1	Parity error

RAF	Receiver Active Flag
0	No reception in progress
1	Reception in progress

NF	Noise Flag
0	No Noise
1	Noise

SCI Status Register (SCISR) SCI_BASE+$2	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	0	0	0	RAF
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SCI

SCI Data Register (SCIDR)

arch.h: ArchIO.Sci **?**.DataReg
registers.h: ArchIO_Sci **?**_DataReg

Where **?** = 0 or 1 (826)
Where **?** = 0, 1 or 2 (827)

Nine bits of data in the SCIDR register can be read at any time, and can also be written to at any time.

During a read, 9 bits of received data may be accessed.

During a write, 9 bits of data to be transmitted may be accessed.

SCI Data Register (SCIDR) SCI_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	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: _____

SPI

SPI Status and Control Register (SPSCR)

This register is continued on the following page.

arch.h: ArchIO.Spi ?.ControlReg
registers.h: ArchIO_Spi ?_ControlReg

Where ? = 0 or 1

Data Shift Order	DSO
MSB transmitted first (MSB ≥ LSB)	0
LSB transmitted first (LSB ≥ MSB)	1
Note: LSB is always at location 0, and MSB is at the correct bit position when reading/writing registers SPDRR, or SPDTR.	

SPI Receiver Full	SPRF
Receive Data Register not full	0
Receive Data Register full	1
Note: SPRF clears automatically after register SPDRR is read.	
Note: SPRF generates an interrupt request if bit SPRIE is set.	

Error Interrupt Enable	ERRIE
MODF and OVRF cannot generate interrupt requests	0
MODF and OVRF can generate interrupt requests	1

Overflow	OVRF
No overflow	0
Overflow	1

MODF	Mode Fault
0	$\overline{SS}$ pin at appropriate logic level
1	$\overline{SS}$ pin at inappropriate logic level
Slave SPI: MODF is set if the $\overline{SS}$ pin goes high during a transmission with MODFEN bit set.	
Master SPI: MODF is set if the $\overline{SS}$ pin goes low at any time with the MODFEN bit set.	

SPTE	SPI Transmitter Empty
0	Transmit data register not empty
1	Transmit data register empty
Note: SPTE generates an interrupt request if the SPTIE bit is set.	

MODFEN	Mode Fault Enable
0	Slave SPI: Prevents the MODF flag from being set for an enabled SPI Master SPI: The $\overline{SS}$ pin level does not affect the operation of an enabled SPI.
1	Allows the MODF flag to be set Clearing MODFEN does not clear the MODF flag.

SPI Status and Control Reg. (SPSCR) SPI_BASE+$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	DSO	SPRF	EERIE	OVRF	MODF	SPTE	MOD FEN	SPR1	SPR0	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SPI

SPI Status and Control Register (SPSCR) Continued

arch.h: ArchIO.Spi ?.ControlReg
registers.h: ArchIO_Spi ?_ControlReg

Where ? = 0 or 1

SPI Baud Rate Select			
Slave Mode	SPRx has no effect		
Master Mode	SPRx bits select one of four baud rates listed in the table below		
	SPR1	SPR0	BD
	0	0	2
	0	1	8
	1	0	16
	1	1	32
Baud Rate = (clk / BD) where clk = IPBus Clock BD = Baud Rate Divisor			

SPI Receiver Interrupt Enable	SPRIE
SPRF interrupt requests disabled	0
SPRF interrupt requests enabled	1
Note: Bit SPRF is set when a full data length transfers from the shift register to the SPDRR register.	

SPI Master Bit	SPMSTR
Slave mode operation selected	0
Master mode operation selected	1

Clock Polarity Bit	CPOL
Rising edge of SCLK starts transmission	0
Falling edge of SCLK starts transmission	1
Note: To transmit data between SPI modules, the SPI modules must have identical CPOL values.	

CPHA	Clock Phase	
	CPHA controls the timing relationship between the serial clock and SPI data. The SPI modules must have identical CPHA values to transmit data.	
	SPI Configured as a Slave	**SPI Configured as a Master**
0	A falling edge on the $\overline{SS}$ pin starts the slave data transmission. The $\overline{SS}$ pin of the slave must be toggled high and back to low between each full length data transmission.	The SCLK signal remains inactive for the first half period.
1	The first SCLK edge starts a transmission. The $\overline{SS}$ pin can remain low between transmissions.	The first SCLK cycle begins with an edge on the SCLK line from its inactive to active level.

SPE	SPI Enable
0	SPI module disabled
1	SPI module enabled
Note: Clearing the SPE causes a partial reset of the SPI.	

SPTIE	SPI Transmit Interrupt Enable
0	SPTE interrupt requests disabled
1	SPTE interrupt requests enabled
Note: SPTE is set when a full data length transfers from the SPDTR register to the shift register.	

SPI Status and Control Reg. (SPSCR) SPI_BASE+$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	DSO	SPRF	EERIE	OVRF	MODF	SPTE	MOD FEN	SPR1	SPR0	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SPI

SPI Data Size Register (SPDSR)

arch.h: ArchIO.Spi ?.DataSizeReg
registers.h: ArchIO_Spi ?_DataSizeReg

Where ? = 0 or 1

DS3–DS0	Data Size— Data Length for Each Transmission
0000	Not Allowed
0001	2 bits
0010	3 bits
0011	4 bits
0100	5 bits
0101	6 bits
0110	7 bits
0111	8 bits
1000	9 bits
1001	10 bits
1010	11 bits
1011	12 bits
1100	13 bits
1101	14 bits
1110	15 bits
1111	16 bits

Note 1: The master and slave must transfer the same data length on each transmission.

Note 2: To cause a new value to take effect, disable and then enable the SPE bit in the SPSCR register.

SPI Data Size Register (SPDSR) SPI_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DS												DS3	DS2	DS1	DS0
	Wrte																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI

SPI Data Receive Register (SPDRR)

arch.h: ArchIO.Spi **?**.DataRxReg
registers.h: ArchIO_Spi **?**_DataRxReg

Where **?** = 0 or 1

Receive
Data read from SPDRR Data Receive **(R15–R0)** Register shows the last full data received after a complete transmission.
Note: The SPRF bit in the SPI Status Control Register (SFSCR) clears automatically after reading SPDRR.
Note: The SPRF bit in the SPI Status Control Register (SPSCR) will set when new data has been transferred to this register.
Note: The SPI Transmitter Empty Bit (SPTE) in the SPSCR register indicates when the next write to register SPDRR can occur.

SPI Data Receive Register (SPDRR) SPI_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

SPI

SPI Data Transfer Register (SPDTR)

arch.h: ArchIO.Spi **?**.DataTxReg
registers.h: ArchIO_Spi **?**_DataTxReg

Where **?** = 0 or 1

Transmit	
Data written to the SPD register is written to the transmit **(T15 -T0)** data buffer.	
Master Mode:	If new data is not written while in master mode, a new transaction will not begin until this register is written.
Slave Mode:	In slave mode, old data will be re-transmitted.
Note: Write all data with the LSB at bit 0. **Note:** Write new data to this register only when the SPTE bet in register SPSCR is set, otherwise, data may be lost. **Note:** Register SPDTR can only be written when the SPI is enabled, SPE=1.	

SPI Data Transmit Register (SPDTR) SPI_BASE+$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read																
	Write	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

PCS

PCS Base Address Registers (PCSBAD0-7)

arch.h: ArchIO.Pcs.PcsBar *?*
registers.h: ArchIO_Pcs_PcsBar *?*

Where *?* = 0, 1, 2, 3, 4, 5, 6, or 7

PCSBARn Registers PCS_BASE+$0 & PCS_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

PCSBARn Registers PCS_BASE+$2 to PCS_BASE+$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	0	0	0	0	0	0	BLKSZ		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

PCS

PCS Optional Registers (PCSOR0-7)

arch.h: ArchIO.Pcs.PcsOr *?*
registers.h: ArchIO_Pcs_PcsOr *?*

Where *?* = 0, 1, 2, 3, 4, 5, 6, or 7

PCSORn[15:0] Registers PCS_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	PDSEN	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

PCSORn Registers PCS_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	0	0	0	PDSEN	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PCSORn Registers PCS_BASE+$2 to PCS_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	0	0	0	0	PDSEN	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

Sheet 1 of 9

SSI

SSI Transmit Register (STX)

arch.h: ArchIO.Ssi.TransmitReg
registers.h: ArchIO_Ssi_TransmitReg

Transmit
Data to be transmitted is written to the SSI Transmit Data register (STX).
If the transmit FIFO is used, data is transferred from this register to the Transmit FIFO register when it becomes empty. Otherwise, data written to this register is transferred to the Transmit Shift Register (TXSR) when shifting of previous data is completed.
Note: Enable SSI (SSIEN=1) before writing to SSI Transmit Register.

SSI Transmit Register (STX) SSI_BASE+$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HIGH BYTE								LOW BYTE							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSI

SSI Receive Register (SRX)

arch.h: ArchIO.Ssi.ReceiveReg
registers.h: ArchIO_Ssi_ReceiveReg

Receive
Data received to be written to the SSI Receive Data register (SRX).
The register always accepts data from the Receiver Shift Register as it becomes full. If FIFO is enabled, the Receive Data Register then receives its values from this FIFO register after SSI Receive Data (SRX) register is full.

SSI Receive Register (SRX) SSI_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HIGH BYTE								LOW BYTE							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SSI

SSI Control/Status Register (SCSR)

arch.h: ArchIO.Ssi.ControlStatusReg
registers.h: ArchIO_Ssi_ControlStatusReg

Divider 4 Disable	DIVADIS
Fix_clk is equal to the mcu_clk	1
Fix_clk is equal to the mcu_clk/4	0

Receive Shift Direction	RSHFD
LSB bit is first received	1
MSB bit is first received	0

Receive Clock Polarity	RSCKP
Rising edge of clk ID used to capture data	1
Falling edge of clk used to capture data	0

Receive DMA Enable	RDMAE
There is no DMA on this part. This bit should always be cleared. It is considered a reserved bit.	

Transmit DMA Enable	TDMAE
There is no DMA on this part. This bit should always be cleared. It is considered a reserved bit.	

Receive Frame Sync Invert	RFSI
Receive frame sync in active low	1
Receive frame sync is active high	0

Receive Frame Sync Length	RFSL
One clock-bit-long frame sync	1
word-long frame sync	0

Receive Early Frame Sync	REFS
Frame sync initiated one bit before data received	1
Frame sync initiated as first bit of data received	0

Receive Data Ready	RDR
SSI Receive Data (SRX) Register or Receive FIFO loaded with a new value	1
	0

Transmit Data Register Empty	TDE
No data waiting to be transferred to STX	1
Data written to the STX or the STSR	0

ROE	Receive Overrun Error
1	Receive Shift Register (RXSR) filled, ready to transfer to the SRX of receive FIFO register. These registers are already filled.
0	Power on, or SSI reset, is also cleared by reading the SCSR with the ROE bit set, followed by reading the SRX.

TUE	Transmitter Underrun Error
1	When TXSR is empty and a transmit time slot occurs.
0	Frame sync occurred during power-on, or SSI reset and is cleared by reading SCSR with TUE bit set followed by writing to the STX register or to the STSR.

TFS	Transmitter Frame Sync
1	Frame sync occurred during transmission of last word written to STX register.
0	During power-on, SSI reset, or when starting transmission of next slot in network mode.

RFS	Receive Frame Sync
1	During receiving of next word into SRX
0	During power-on, SSI reset, or next slot of frame begins to receive in network mode.

RFF	Receive FIFO Full
1	Data level in Receive FIFO reaches selected Receive FIFO Watermark (RFWM) threshold.
0	Power-on reset and ssi is disabled receive FIFO has fewer than threshold values.

TFE	Transmit FIFO Empty
1	Data level in the Transmit FIFO falls below the selected Transmit Watermark (RFWM) threshold.
0	Power-on reset when SSI is disabled transmit FIFO has more than threshold values.

SSI Control/Status Register (SCSR) SSI_BASE+$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DIV4 DIS	RSHFD	RSCKP	RDMAE	TDMAE	RFSI	RFSL	REFS	RDR	TDE	ROE	TUE	TFS	RFS	RFF	TFE
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

SSI

SSI Control Register 2 (SCSR2)

arch.h: ArchIO.Ssi.Control2Reg
registers.h: ArchIO_Ssi_Control2Reg

Receive Interrupt Enable	RIE
Allows interrupt of program controller if ROE or RFF/RDR bit is set	1
Disables the receive interrupt	0

Transmit Interrupt Enable	TIE
Allows interrupt of program controller in TUE or TFE/TDE bit is set.	1
Disables the transmit interrupt	0

Receive Enable	RE
Receiver is enabled	1
Receiver is disabled	0

Transmit Enable	TE
Transmitter is enabled	1
Transmitter is disabled	0

Receive FIFO Enable	RFEN
Enable receive FIFO	1
Disables receive FIFO	0

Transmit FIFO Enable	TFEN
Enable transmit FIFO	1
Disable transmit FIFO	0

Receive Clock Direction	RXDIR
Clock is generated internally	1
Clock source is external	0

Transmit Clock Direction	TXDIR
Clock is generated internally	1
Clock source is external	0

SYN	Synchronous Mode
1	Synchronous mode
0	Other mode

TSHFD	Transmit Shift Direction
1	LSB is conveyed first for the transmit section.
0	MSB is conveyed first for the transmit section.

TSCKP	Transmit Clock Polarity
1	Falling edge of the bit clock is used to clock the data out.
0	Rising edge of the bit clock is used to clock the data out

SSIEN	SSI Enable
1	SSI is enabled
0	SSI is disabled

NET	Network Mode
1	Network mode of operation selected
0	Normal mode selected

TFSI	Transmit Frame Sync Invert
1	Frame sync is active low
0	Frame sync is active high

TFSL	Transmit Frame Sync Length
1	One clock bit-long frame sync
0	One word long frame sync

TEFS	Transmit Early Frame Sync
1	Frame sync is initiated one bit before the data is transmitted.
0	Frame sync is initiated as the first bit of data is transmitted.

SSI Control Register (SCR2) SSI_BASE+$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RIE	TIE	RE	TE	RFEN	TFEN	RXDIR	TXDIR	SYN	TSHFD	TSCKP	SSIEN	NET	TFSI	TFSL	TEFS
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSI

SSI Transmit Control Register (STXCR)

arch.h: ArchIO.Ssi.TxControlReg
registers.h: ArchIO_Ssi_TxControlReg

Prescaler Range	PSR
Divide-by-eight prescaler is operational	1
Prescaler is bypassed	0

Frame Rate Divider Control DC

Control the divide ratio for programmable frame rate dividers. The divide ratio ranges from 1 to 32 in normal mode and from 2 to 32 in Network mode.

WL

Used to select the length of the data words*
*See Table below.

Prescale Module Select PM

Specify the divide ratio of the prescale divider in the SSI clock generator. A divide ratio from 1 to 256 (PM[7:0] = $00 to $FF) can be selected.

SSI Transmit Control Register (STXCR) SSI_BASE+$0x04	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WL0	WL1	Number of Bits/Word
0	0	8
1	0	10
0	1	12
1	1	16

SSI

SSI Receive Control Register (SRXCR)

arch.h: ArchIO.Ssi.RxControlReg
registers.h: ArchIO_Ssi_RxControlReg

Prescaler Range	PSR
Fixed divide-by-eight prescaler is operational	1
Fixed prescaler is bypassed	0

Frame Rate Divider Control DC
Control the divide ratio for programmable frame rate dividers. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode.

WL
Used to select the length of the data words* *See Table below.

Prescale Module Select PM
Specify the divide ratio of the prescale divider in the SSI clock generator. A divide ratio from 1 to 256 (PM[7:0] = $00 to $FF) can be selected.

SSI Receive Control Register (SRXCR) SSI_BASE+$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WL0	WL1	Number of Bits/Word
0	0	8
1	0	10
0	1	12
1	1	16

SSI

SSI Time Slot Register (STSR)

arch.h: ArchIO.Ssi.TimeSlotReg
registers.h: ArchIO_Ssi_TimeSlotReg

Time Slot Register STSR[15:0]
Used when data is not to be transmitted in an available transmit time slot. The time slot register is a write-only register. It behaves like an alternate transmit data register.

SSI Time Slot Register (STSR) SSI_BASE+$6	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Write	DUMMY REGISTER, WRITTEN DURING INACTIVE TIME SLOTS (NETWORK MODE)															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSI

SSI FIFO Control/Status Register (SFCSR)

arch.h: ArchIO.Ssi.FifoCntlStatReg
registers.h: ArchIO_Ssi_FifoCntlStatReg

Receive FIFO Counter RFCNT
Indicates the number of words in the Receive FIFO See Table 12-7

Receive FIFO Full WaterMark RFWM
Controls the receive FIFO full flag threshold. See Table 12-9.

Transmit FIFO Counter TFCNT
Indicates the number of words in the Transmit FIFO. SeeTable 12-8.

Transmit FIFO Full WaterMark TFWM
Controls the threshold setting of the transmit FIFO empty flag. See Table 12-10.

SSI FIFO Control/Status Register (SFCSR) SSI_BASE+$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RFCNT				TFCNT				RFWM				TFWM			
	Write																
	Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Reserved Bits

SSI

SSI Option Register (SOR)

arch.h: ArchIO.Ssi.OptionReg
registers.h: ArchIO_Ssi_OptionReg

Receive Frame Direction	RFDIR
Frame sync is generated internally	1
Frame sync is external	0

TFDIR	Transmit Frame Direction
1	Frame sync is generated internally
0	Frame sync is external

SSI Option Register (SOR) SSI_BASE+$9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	15	14	13	12
	Read	0	0	0	0	0	0	0	0	0	0	RFDIR	TFDIR	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: _____

TMR

TMR Control Registers (CTRL)

This register is continued on the following page.

arch.h: ArchIO.TimerA.Channel ***i***.ControlReg
registers.h: ArchIO_TimerA_Channel ***i***_ControlReg

Where ***i*** = 0, 1, 2, 3, 4

Count Mode	Count Mode
No operation	000
Counts rising edges of primary source **Note:** Rising edges are counted only when IPS = 0. Falling edges are counted when IPS = 1.	001
Counts rising and falling edges of primary source	010
Counts rising edges of primary source while secondary input high active.	011
Quadrature count mode; uses primary and secondary sources	100
Counts primary source rising edges; secondary source specifies direction. **Note:** Rising Edges are counted only when IPS = 0. Falling edges are counted when IPS = 1.	101
Edge of secondary source triggers primary count until compare.	110
Cascaded Counter mode (up/down). **Note:** Primary count source must be set to 1 of the counter outputs.	111

Primary Count Source	Primary Count Source
The PRIMARY COUNT SOURCE bits select the primary count source.	
0000	Counter #0 input pin
0001	Counter #1 input pin
0010	Counter #2 input pin
0011	Counter #3 input pin
0100	Counter #0 output
0101	Counter #1 output
0110	Counter #2 output
0111	Counter #3 output
1000	Prescaler (IPBus clock divide by 1)
1001	Prescaler (IPBus clock divide by 2)
1010	Prescaler (IPBus clock divide by 4)
1011	Prescaler (IPBus clock divide by 8)
1100	Prescaler (IPBus clock divide by 16)
1101	Prescaler (IPBus clock divide by 32)
1110	Prescaler (IPBus clock divide by 64)
1111	Prescaler (IPBus clock divide by 128)
Note: A timer selecting its own output for input is not a legal choice. The result is no counting.	

TMR Control Register (CTRL)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COUNT MODE			PRIMARY COUNT SOURCE				SECONDARY SOURCE		ONCE	LENGTH	DIR	CO INIT	OUTPUT MODE		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE +$6, $E, $16, or $1E

TMRA0_CTRL (Timer A, Channel 0 Control)—Address: TMRA_BASE + $6
TMRA1_CTRL (Timer A, Channel 1 Control)—Address: TMRA_BASE + $E 56F826 – $16 56F827
TMRA2_CTRL (Timer A, Channel 2 Control)—Address: TMRA_BASE + $16 56F826 – $26 56F827
TMRA3_CTRL (Timer A, Channel 3 Control)—Address: TMRA_BASE + $1E 56F826 – $36 56F827

TMR

TMR Control Registers (CTRL) Continued

This register is continued on the following page.

arch.h: ArchIO.TimerA.Channel ***i***.ControlReg
registers.h: ArchIO_TimerA_Channel ***i***_ControlReg

Where ***i*** = 0, 1, 2, 3, 4

Secondary Count Source	SECOND-ARY SOURCE
Counter #0 input pin	00
Counter #1 input pin	01
Counter #2 input pin	10
Counter #3 input pin	11

Secondary Count Once	ONCE
Count repeatedly	0
Count until compare and then stop. If counting up, successful compare occurs when counter reaches CMP1 value. If counting down, successful compare occurs when counter reaches CMP2 value.	1

LENGTH	Count LENGTH
0	Roll Over
1	Count until compare, then reinitialilze. If counting up, successful compare occurs when counter reaches CMP1 value. If counting down, successful compare occures when counter reaches CMP2 value. **Note:** When output mode $4 is used, alternating values of CMP1 and CMP2 are used to generate successful compares. For example, when output mode is $4, the counter counts until CMP1 value is reached, reinitializes, then counts until CMP2 value is reached, reintiallizes, then counts until CMP1 value is reached, and so on.

TMR Control Register (CTRL)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	COUNT MODE			PRIMARY COUNT SOURCE				SECONDARY SOURCE		ONCE	LENGTH	DIR	CO INIT	OUTPUT MODE		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE +$6, $E, $16, or $1E

TMRA0_CTRL (Timer A, Channel 0 Control)—Address: TMRA_BASE + $6
TMRA1_CTRL (Timer A, Channel 1 Control)—Address: TMRA_BASE + $E 56F826 – $16 56F827
TMRA2_CTRL (Timer A, Channel 2 Control)—Address: TMRA_BASE + $16 56F826 – $26 56F827
TMRA3_CTRL (Timer A, Channel 3 Control)—Address: TMRA_BASE + $1E 56F826 – $36 56F827

TMR

TMR Control Registers (CTRL) Continued

arch.h: ArchIO.TimerA.Channel ***i***.ControlReg
registers.h: ArchIO_TimerA_Channel ***i***_ControlReg

Where ***i*** = 0, 1, 2, 3, 4

Count Direction	DIR
Count up	0
Count down	1

C0-Channel Initialization	Co INIT
Co-channel counter/timers can not force a reinitialization of this counter/timer	0
Co-channel counter/timers may force a reinitialization of this counter/timer	1

OUTPUT MODE	Output Mode
000	Asserted while counter is active
001	Clear OFLAG output on successful compare
010	Set OFLAG output on successful compare
011	Toggle OFLAG output on successful compare
100	Toggle PFLAG output using alternating compare registers
101	Set on compare, cleared on secondary source input edge
110	Set on compare, cleared on counter rollover
111	Enable Gated Clock output while counter is active **Note:** The Primary count source must be set to one of the counter outputs.

TMR Control Register (CTRL)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read / Write	COUNT MODE			PRIMARY COUNT SOURCE				SECONDARY SOURCE		ONCE	LENGTH	DIR	CO INIT	OUTPUT MODE		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE +$6, $E, $16, or $1E

TMRA0_CTRL (Timer A, Channel 0 Control)—Address: TMRA_BASE + $6
TMRA1_CTRL (Timer A, Channel 1 Control)—Address: TMRA_BASE + $E 56F826 – $16 56F827
TMRA2_CTRL (Timer A, Channel 2 Control)—Address: TMRA_BASE + $16 56F826 – $26 56F827
TMRA3_CTRL (Timer A, Channel 3 Control)—Address: TMRA_BASE + $1E 56F826 – $36 56F827

TMR

TMR Status and Control Registers (SCR)

This register is continued on the following page.

arch.h: ArchIO.TimerA.Channel ***i***.StatusControlReg
registers.h: ArchIO_TimerA_Channel ***i***_StatusControlReg

Where ***i*** = 0, 1, 2, or 3

Timer Compare Flag	TCF
Compare has not yet occurred	0
A successful compare occurred	1

Timer Compare Flag Interrupt Enable	TCFIE
Timer compare interrrupt is disabled	0
Interrupts enabled if the Timer Compare Flag, TCF, is set	1

Timer Overflow Flag	TOF
Timer overflow has not occurred	0
The timer/counter rolled over to its maximum value $FFFF or $0000 (depending on count direction)	1

Timer Overflow Flag Interrupt Enable	TOFIE
Timer overflow interrupt disabled	0
Timer overflow interrupt enabled if the Timer Overflow Flag, TOF, is also set	1

IEF	Input Edge Flag
0	An input edge transition has not occurred
1	A positive input edge transition occurred **Note:** Set the IPS bit to enable negative input edge transition detection. **Note:** The Secondary Count Source in the Control REgister, CTRL, determines which external input pin is monitored.

IEFIE	Input Edge Flag Interrupt Enable
0	Input edge interrupts disabled
1	Input edge interrupts enabled when the IEF bit is set

IPS	Input Polarity Select
0	Input signal polarity is unchanged
1	Input signal polarity is inverted

TMR Status and Control Registers (SCR)	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

TMR_BASE +$7, $F, $17, or $1F

TMRA0_SCR (Timer A, Channel 0 Status & Control)—Address: TMRA_BASE + $7
TMRA1_SCR (Timer A, Channel 1 Status & Control)—Address: TMRA_BASE + $F 56F826 – $17 56F827
TMRA2_SCR (Timer A, Channel 2 Status & Control)—Address: TMRA_BASE + $17 56F826 – $27 56F827
TMRA3_SCR (Timer A, Channel 3 Status & Control)—Address: TMRA_BASE + $1F 56F826 – $37 56F827

Reserved Bits

TMR

TMR Status and Control Registers (SCR) Continued

arch.h: ArchIO.TimerA.Channel ***i***.StatusControlReg
registers.h: ArchIO_TimerA_Channel ***i***_StatusControlReg

Where ***i*** = 0, 1, 2, or 3

External Input Signal	INPUT
External input pin polarity is unchanged	0
External input pin polarity is inverted	1

Input Capture Mode	CAPTURE MODE
Capture Mode specifies the operation of the Capture Register, CAP, and the Input Edge Flag, IEF.	
The Capture function is disabled, and the Input Edge Flag interrupts are disabled	00
Load the Capture register on a rising input edge	01
Load the Capture register on a falling input edge	10
Load the Capture register on both input edges	11

Master Mode	MSTR
Disabled	0
Enabled	1

Enable External OFLAF Force	EEOF
Compare disabled	0
Enables a compare from another counter/timer	1

VAL	Forced OFLAG Value
0	VAL is disabled
1	VAL initiates a read of teh OFLAG output signal value when a FORCE command is triggered by software or when a FORCE command is issued by a counter/timer set as a master.

FORCE	Force the OFLAG Output
0	This bit always reads as zero.
1	Forces the current value of the VAL bit to be written to the OFLAG output. (FORCE and VAL bits can be written in a single write operation.) **Note:** Write to the FORCE bit only if the counter is disabled. Setting this bit while the counter is enabled may yield unpredicable results.

OPS	Output Polarity Select
0	True polarity
1	Inverted polarity

OEN	Output Enable
0	OFLAG output signal is disabled
1	Enable the OFLAG output signal to be put on the external pin. Also connects a timer's output pin to its input.

TMR Status and Control Registers (SCR)	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

TMR_BASE +$7, $F, $17, or $1F

TMRA0_SCR (Timer A, Channel 0 Status & Control)—Address: TMRA_BASE + $7
TMRA1_SCR (Timer A, Channel 1 Status & Control)—Address: TMRA_BASE + $F 56F826 – $17 56F827
TMRA2_SCR (Timer A, Channel 2 Status & Control)—Address: TMRA_BASE + $17 56F826 – $27 56F827
TMRA3_SCR (Timer A, Channel 3 Status & Control)—Address: TMRA_BASE + $1F 56F826 – $37 56F827

Reserved Bits

TMR

TMR Compare Register 1 (CMP1)

arch.h: ArchIO.TimerA.Channel ***i***.CompareReg1
registers.h: ArchIO_TimerA_Channel ***i***_CompareReg1

Where ***i*** = 0, 1, 2, or 3

Compare Register 1
CMP1 stores the value used for comparison with the counter value

TMR Compare Register 1 (CMP1)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARISON VALUE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $0, $8, $10, or $18

TMRA0_CMP1 (Timer A, Channel 0 Compare 1)—Address: TMRA_BASE + $0
TMRA1_CMP1 (Timer A, Channel 1 Compare 1)—Address: TMRA_BASE + $8 56F826 – $10 56F827
TMRA2_CMP1 (Timer A, Channel 2 Compare 1)—Address: TMRA_BASE + $10 56F826 – $20 56F827
TMRA3_CMP1 (Timer A, Channel 3 Compare 1)—Address: TMRA_BASE + $18 56F826 – $30 56F827

TMR

TMR Compare Register 2 (CMP2)

arch.h: ArchIO.TimerA.Channel ***i***.CompareReg2
registers.h: ArchIO_TimerA_Channel ***i***_CompareReg2

Where ***i*** = 0, 1, 2, or 3

Compare Register 2
CMP2 stores the value used for comparison with the counter value.

TMR Compare Register 2 (CMP2)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARISON VALUE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $1, $9, $11, or $19

TMRA0_CMP2 (Timer A, Channel 0 Compare 2)—Address: TMRA_BASE + $1
TMRA1_CMP2 (Timer A, Channel 1 Compare 2)—Address: TMRA_BASE + $9 56F826 – $11 56F827
TMRA2_CMP2 (Timer A, Channel 2 Compare 2)—Address: TMRA_BASE + $11 56F826 – $21 56F827
TMRA3_CMP2 (Timer A, Channel 3 Compare 2)—Address: TMRA_BASE + $19 56F826 – $31 56F827

TMR

TMR Capture Register (CAP)

arch.h: ArchIO.TimerA.Channel ***i***.CaptureReg
registers.h: ArchIO_TimerA_Channel ***i***_CaptureReg

Where ***i*** = 0, 1, 2, or 3

Capture Register (CAP)
CAP stores the value captured from the counter.

TMR Capture Register (CAP)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CAPTURE VALUE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $2, $A, $12, or $1A

TMRA0_CAP (Timer A, Channel 0 Capture)—Address: TMRA_BASE + $2
TMRA1_CAP (Timer A, Channel 1 Capture)—Address: TMRA_BASE + $A 56F826 – $12 56F827
TMRA2_CAP (Timer A, Channel 2 Capture)—Address: TMRA_BASE + $12 56F826 – $22 56F827
TMRA3_CAP (Timer A, Channel 3 Capture)—Address: TMRA_BASE + $1A 56F826 – $3**2** 56F827

TMR

TMR Load Register (LOAD)

arch.h: ArchIO.TimerA.Channel ***i***.LoadReg
registers.h: ArchIO_TimerA_Channel ***i***_LoadReg

Where ***i*** = 0, 1, 2, or 3

Load Register (LOAD)
LOAD stores the value used to initialize the counter/timer.

TMR Load Register (LOAD)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOAD VALUE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $3, $B, $13, or $1B

TMRA0_LOAD (Timer A, Channel 0 Load)—Address: TMRA_BASE + $3
TMRA1_LOAD (Timer A, Channel 1 Load)—Address: TMRA_BASE + $B 56F826 – $13 56F827
TMRA2_LOAD (Timer A, Channel 2 Load)—Address: TMRA_BASE + $13 56F826 – $23 56F827
TMRA3_LOAD (Timer A, Channel 3 Load)—Address: TMRA_BASE + $1B 56F826 – $33 56F827

TMR

TMR Hold Register (HOLD)

arch.h: ArchIO.TimerA.Channel ***i***.HoldReg
registers.h: ArchIO_TimerA_Channel ***i***_HoldReg

Where ***i*** = 0, 1, 2, or 3

Hold Register (HOLD)
HOLD stores the specific channel's counter value when reading any of the four counters within a module.

TMR Hold Register (HOLD)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HOLD VALUE															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $4, $C, $14, or $1C

TMRA0_HOLD (Timer A, Channel 0 Hold)—Address: TMRA_BASE + $4
TMRA1_HOLD (Timer A, Channel 1 Hold)—Address: TMRA_BASE + $C 56F826 – $14 56F827
TMRA2_HOLD (Timer A, Channel 2 Hold)—Address: TMRA_BASE + $14 56F826 – $24 56F827
TMRA3_HOLD (Timer A, Channel 3 Hold)—Address: TMRA_BASE + $1C 56F826 – $34 56F827

Application: _____ Date: _____

_____ Programmer: _____

TMR

TMR Counter Register (CNTR)

arch.h: ArchIO.TimerA.Channel ***i***.CounterReg
registers.h: ArchIO_TimerA_Channel ***i***_CounterReg

Where ***i*** = 0, 1, 2, or 3

Counter Register (CNTR)
CNTR is the counter for the corresponding channel in a timer module.

TMR Counter Register (CNTR)	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

TMR_BASE + $5, $D, $15, or $1D

TMRA0_CNTR (Timer A, Channel 0 Counter)—Address:TMRA_BASE + $5
TMRA1_CNTR (Timer A, Channel 1 Counter)—Address: TMRA_BASE + $D 56F826 – $15 56F827
TMRA2_CNTR (Timer A, Channel 2 Counter)—Address: TMRA_BASE + $15 56F826 – $25 56F827
TMRA3_CNTR (Timer A, Channel 3 Counter)—Address: TMRA_BASE + $1D 56F826 – $35 56F827

TMR

TMR Comparator Load Register 1 (CMP1)

arch.h: ArchIO.TimerA.Channel ***i***.ComparatorLoadReg1 (827 only)
registers.h: ArchIO_TimerA_Channel ***i***_ComparatorLoadReg1 (827 only)

Where ***i*** = 0, 1, 2, or 3

Compare Load Register 1 (CMP1)
CMP1 stores the value used for comparison with the counter value. Available only on 56F827.

TMR Comparator Load Register1 (CMP1)	**Bits**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Read	COMPARE LOAD1															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $8, $18, $28, $38

TMRA0_CMP1 (Timer A, Channel 0 Compare1)—Address:TMRA_BASE + $8
TMRA1_CMP1 (Timer A, Channel 1 Compare1)—Address: TMRA_BASE + $18 56F827 Only
TMRA2_CMP1 (Timer A, Channel 2 Compare1)—Address: TMRA_BASE + $28 56F827 Only
TMRA3_CMP1 (Timer A, Channel 3 Compare1)—Address: TMRA_BASE + $38 56F827 Only

TMR

TMR Comparator Load Register 2 (CMP2)

arch.h: ArchIO.TimerA.Channel ***i***.ComparatorLoadReg2 (827 only)
registers.h: ArchIO_TimerA_Channel ***i***_ComparatorLoadReg2 (827 only)

Where ***i*** = 0, 1, 2, or 3

Compare Load Register Two (CMP2)
CMP2 stores the value used for comparison with the counter value. Available on the 56F827 only.

TMR Comparator Load Register 2 (CMP2)	**Bits**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Read	COMPARE LOAD 2															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $9, $19, $29, $39

TMRA0_CMP2 (Timer A, Channel 0 Compare2)—Address: TMRA_BASE + $9
TMRA1_CMP2 (Timer A, Channel 1 Compare2)—Address: TMRA_BASE + $19 56F827 Only
TMRA2_CMP2 (Timer A, Channel 2 Compare2)—Address: TMRA_BASE + $29 56F827 Only
TMRA3_CMP2 (Timer A, Channel 3 Compare2)—Address: TMRA_BASE + $39 56F827 Only

TMR

TMR Comparator Status and Control Register (COMSCR)

arch.h: ArchIO.TimerA.Channel ***i***.ComparatorControlReg (827 only)
registers.h: ArchIO_TimerA_Channel ***i***_ComparatorControlReg (827 only)

Where ***i*** = 0, 1, 2, or 3

Timer Compare Two Interrupt Enable (TCF2EN)
An interrupt is ussued when this bit and the TDF2 bit are both set.

Timer Compare One Interrupt Enable (TCF1EN)
An interrupt is issued when this bit and the TCF1 bit bit are both set.

Timer Compare Two Interrupt Source	(TCF2)
Clears the bit location	1
Has no effect	0

Timer Compare One Interrupt Source	(TCF1)
Clears the bit location	1
Has no effect	0

CL2	Compare Load Control Two
00	Never preload
01	Load upon successful compare with the value in CMP1
10	Load upon successful compare with the value in CMP1
11	Reserved

CL1	Compare Load Control One
00	Never preload
01	Load upon successful compare with the value in CMP1
10	Load upon successful compare with the value in CMP2
11	Reserved

TMR Comparator Status and Control Register (COMSCR)	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	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_BASE + $A, $1A, $2A, or $3A

TMRA0_COMSCR (Timer A, Channel 0 COMSCR)—Address: TMRA_BASE + $A 56F827 Only
TMRA1_COMSCR (Timer A, Channel 1 COMSCR)—Address: TMRA_BASE + $1A 56F827 Only
TMRA2_COMSCR (Timer A, Channel 2 COMSCR)—Address: TMRA_BASE + $2A 56F827 Only
TMRA3_COMSCR (Timer A, Channel 3 COMSCR)—Address: TMRA_BASE + $3A 56F827 Only

Reserved Bits

TOD

TOD Control Status Register (TODCS)

arch.h: ArchIO.Tod.ControlReg
registers.h: ArchIO_Tod_ControlReg

TOD One Second Interrupt Occurred Flag	TODSIO
No one second interrupt occurred	0
TOD one second interrupt occurred **Note:** The ISR should clear this bit to clear the interrupt	1

TOD Alarm Interrupt	TODDAL
No TOD alarm occurred flag	0
TOD alarm interrupt occurred **Note:** The ISR should clear this bit to clear the interrupt.	1

TOD Days Alarm	TODDA
Days alarm register not used in alarm determination	0
Days alarm register is used in alarm determination	1

TOD Enable Hours Alarm	TODHA
Hours alarm register not used in alarm determination	0
Hours alarm register used in alarm determination	1

TODMA	TOD Enable Minutes Alarm
0	Minutes alarm register not used in alarm determination
1	Minutes alarm register used in alarm determination

TODSA	TOD Enable Second Alarm
0	Seconds alarm register not used in alarm determination
1	Seconds alarm register used in alarm determination

TODSEN	TOD Second Alarm Enable
0	TOD One second IRQ disable
1	TOD One second IRQ enable

TODAEN	TOD Alarm Enable
0	TOD alarm IRQ disabled
1	TOD alarm IRQ enabled

TODEN	TOD Enable
0	Allow write to TOD
1	Enable TOD operation, write to TOD counter registers disable.

TOD Clock Control/Status (TODCS) TOD_BASE+$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TODSIO	TODAL	0	0	0	0	TEST		TODDA	TODHA	TODMA	TODSA	TODSEN	TODAEN	0	TODEN
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Clock Scaler Register (TODCSL)

arch.h: ArchIO.Tod.ClockScaleReg
registers.h: ArchIO_Tod_ClockScaleReg

TODSCL Register
All bits can be written when TODEN (bit 0 of TODCS) is low. Decimal range is 0-65535, Bits 0-15.

TOD Clock Scaler (TODCSL) TOD_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TIME OF DAY CLOCK SCALER															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TOD

TOD Seconds Register (TODSEC)

arch.h: ArchIO.Tod.SecondsReg
registers.h: ArchIO_Tod_SecondsReg

TOD Second Register (TODSEC)
Can be written only when TODEN is low. Decimal range is 0-59. Bits 0-5.

TOD Seconds (TODSEC) TOD_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	0	0	0	0	0	TODSEC					
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Seconds Alarm Register (TODSAL)

arch.h: ArchIO.Tod.SecondsAlarmReg
registers.h: ArchIO_Tod_SecondsAlarmReg

TOD Seconds Alarm Register
Can always be modified. Bits 0-5.

TOD Seconds Alarm (TODSAL) TOD_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	TODSAL					
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Minutes Register (TODMIN)

arch.h: ArchIO.Tod.MinutesReg
registers.h: ArchIO_Tod_MinutesReg

TOD Minutes Register
Can always be modified when TODEN is low. Decimal range is 0-59. Other bits are reserved. Bits 0-5.

TOD Minutes (TODMIN) TOD_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	TODMIN					
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Minutes Alarm Register (TODMAL)

arch.h: ArchIO.Tod.MinutesAlarmReg
registers.h: ArchIO_Tod_MinutesAlarmReg

TOD Minutes Alarm Register
Can always be modified. Decimal range is 0-59. Bits 0-5.

TOD Minutes Alarm (TODMAL) TOD_BASE+$5	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	TODMAL					
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Hours Register (TODHR)

arch.h: ArchIO.Tod.HoursReg
registers.h: ArchIO_Tod_HoursReg

TOD Hour Register
Can be modified only when TODEN is low. Decimal range is 0-23. Bits 0-4.

TOD Hours (TODHR) TOD_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	0	0	0	0	TODHR				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Hours Alarm Register (TODHAL)

arch.h: ArchIO.Tod.HoursAlarmReg
registers.h: ArchIO_Tod_HoursAlarmReg

TOD Hour Alarm Register
Can always be modified. Decimal range is 0-23. Bits 0-4.

TOD Hours Alarm (TODHAL) TOD_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	0	TODHAL				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Days Register (TODDAY)

arch.h: ArchIO.Tod.DaysReg
registers.h: ArchIO_Tod_DaysReg

TOD Day Register
Can be modified only when TODEN is low. Decimal range is 0-65535. Bits 0-15.

TOD Days Register (TODDAY) TOD_BASE+$8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TODDAY															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

TOD

TOD Days Alarm Register (TODDAL)

arch.h: ArchIO.Tod.DaysAlarmReg
registers.h: ArchIO_Tod_DaysAlarmReg

TOD Day Alarm Register
Can always be modified. Decimal range is 0-65535. Bits 0-15.

TOD Days Alarm Register (TODDAL) TOD_BASE+$9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TODDAL															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

OCCS

PLL Control Register (PLLCR)

arch.h: ArchIO.Pll.ControlReg
registers.h: ArchIO_Pll_ControlReg

PLL Interrupt Enable 1	PLLIE1
Disable interrupt	00
Enable interrupt on any rising edge of LCK1	01
Enable interrupt on falling edge of LCK1	10
Enable interrupt on any edge change of LCK1	11

CHPMPTRI	Charge Pump Tri-state
0	Normal chip operation
1	In the event of *Loss of Reference* (Clock: FREF), set bit CHPMPTRI to 1. **Note:** Activating this bit renders the PLL inoperable and should not be done during standard chip operation.

PLL Interrupt Enable 0	PLLIE0
Disable interrupt	00
Enable interrupt on any rising edge of LCK0	01
Enable interrupt on falling edge of LCK0	10
Enable interrupt on any edge change of LCK0	11

PLLPD	PLL Power Down
0	PLL enabled
1	PLL powered down **Note:** When PLL is powered down, ZSRC[2:0] is switched to this value to prevent a loss of clock to the core.

Loss of Clock Interrupt Enable	LOCIE
Interrupt disabled	0
Interrupt enabled	1

ZSRC	ZCLOCK Source
01	Prescaler output **Note:** ZSRC is automatically set to this value when PLLPD is set or when PLL enters STOP_MODE preventing loss of clock to the core.
10	Postscaler output
00, 11	Reserved

Lock Detector On	LCKON
Lock detector disabled	0
Lock detector enabled	1

PLL Control Register (PLLCR) CLKGEN_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	0	ZSRC	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

Sheet 2 of 5

OCCS

PLL Divide-By Register (PLLDB)

arch.h: ArchIO.Pll.DivideReg
registers.h: ArchIO_Pll_DivideReg

PLL Divide-By	PLLDB
PLL Output Frequency = Fout fout = $\frac{FREF \times (n+1)}{2}$ where FREF = Reference Frequency = (XTAL Clock) / PLLCID[1:0] n + 1 = Scaling Value n = value of PLLDB[6:0] $0 \le n \le 127$	0 – 127
Note: Before changing the divide-by value, it is recommended that the core clock be switched to the Prescaler Clock. **Note:** The value for n should be programmed so fout is in the normal operating range, 80 - 160MHz.	

PLLCO	PLL Clock Out Divide (Postscaler)
00	Divide by 1
01	Divide by 2
10	Divide by 4
11	Divide by 8

LORTP	Loss of Reference Timer Period
0 – 15	t = LORTP x 10 x (PLL-Clock-Time-Period) where t = Time to Generate Loss of Reference Interrupt

PLLCI	PLL Clock In Divide (Prescaler)
00	Divide by 1
01	Divide by 2
10	Divide by 4
11	Divide by 8

PLL Divide-By Register (PLLDB) CLKGEN_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LORTP				PLLCOD		PLLCID		0	PLLDB						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Reserved Bits

OCCS

PLL Status Register (PLLSR)

arch.h: ArchIO.Pll.StatusReg
registers.h: ArchIO_Pll_StatusReg

PLL Loss of Lock Interrupt 1	LOLI1
No interrupt pending [2:0]	0
Interrupt pending (See also PLLIE1[2:0])	1
Note: Write a one to LOLI1 to clear this bit.	

PLL Loss of Lock Interrupt 0	LOLIO
No interrupt pending [2:0]]	0
Interrupt pending (See also PLLIE1[2:0])	1
Note: Write a one to LOLI0 to clear this bit.	

Loss of Clock	LOCI
PLL reference clock normal	0
Lost PLL reference clock	1

Loss of Lock 1	LCK1
PLL is unlocked (fine)	0
PLL is locked (fine)	1

Loss of Lock 0	LCK0
PLL is unlocked (course)	0
PLL is locked (course)	1

PLLPDN	PLL Power Down
0	PLL not powered down
1	PLL powered down
Note: The PLL power-down status is delayed by four IPBus Clocks from the PLLPD bit in the PLLCR.	

PRECSS	Prescaler Clock Select Status
0	Relaxation oscillator clock
1	Crystal oscillator clock
Note: PRECSS takes more than one IPBus clock to indicate a changeover to a new clock.	

ZSRC	ZCLOCK Source
00	Synchronizing in progress
01	Prescaler output
10	Postscaler output
11	Synchronizing in progress
Note: ZSRC takes more than one IPBus clock to indicate a new selection.	

PLL Status Register (PLLSR) CLKGEN_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	PRECSS	ZSRC	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Reserved Bits

OCCS

PLL Select Register (CLKOSR)

arch.h: ArchIO.Pll.SelectReg
registers.h: ArchIO_Pll_SelectReg

CLKOSEL	CLKO Select
Selects clock to be “multiplexed out” on the CLKO pin.	
10000	No Clock
00000	ZCLK (Default)
00001 – 01111	Reserved for factory test

CLKO Select Register (CLKOSR) CLKGEN_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	CLKOSEL				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

COP

COP Control Register (COPCTL)

arch.h: ArchIO.Cop.ControlReg
registers.h: ArchIO_Cop_ControlReg

Stop Enable	CSEN
COP counter stops in Stop mode	0
COP counter runs in Stop mode	1
Note: This bit can only be changed when the CWP bit is set to zero. For the COP to run in Stop mode, the CEN bit must also be set.	

CEN	COP Enable
0	COP is disabled
1	COP is enabled
Note: This bit can only be changed when CWP is set to zero. Once this bit has been written to a 1, it cannot be changed back to 0 without resetting the COP module.	

COP Wait Enable	CWEN
COP counter will Stop in Wait mode	0
COP counter will run in Wait mode	1
Note: This bit can only be changed when the CWP bit is set to zero. For the COP to run in Stop mode, the CEN bit must also be set.	

CWP	COP Write Protect
0	COPCTL, COPTO registers are readable and writeable
1	COPCTL, COPTO registers are read-only

Control Register (COPCTL) SYS_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	CSEN	CWEN	CEN	CWP
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

COP

COP Timeout Register (COPTO)

arch.h: ArchIO.Cop.TimeoutReg
registers.h: ArchIO_Cop_TimeoutReg

CT	COP Timeout
0 – 4095	COP Timeout Period = [16384 x (CT[11:0] + 1) clock cycles Write the CT[11:0] value before the COP is enabled. (The COP is enabled in the COPCTL register.)
Note:	Changing CT[11:0] while the COP is enabled will result in a timeout period that differs from the expected value given by the formula above.
Note:	These bits can only be changed when the CWP bit in COPCTL is set to zero.

COP Timeout Register (COPTO) SYS_BASE+$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	COP TIMEOUT											
	Write																
	Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Reserved Bits

Application: _____ Date: _____

_____ Programmer: _____

COP

COP Service Register (COPSRV)

arch.h: ArchIO.Cop.ServiceReg
registers.h: ArchIO_Cop_ServiceReg

Service Sequence Write Values	COP Service Register
Write $5555 then Write $AAAA	COPSRV performs a periodic service sequence to clear the COP counter and prevent a reset. To perform a service sequence: 1. Write the value $5555 to COPSRV. An indefinite number of instructions may be executed before the second write. 2. Write the value $AAAA to COPSRV before the selected time-out period (as set in the COPTO register) expires.
Note: The writes to COPSRV must be performed in the correct order BEFORE the counter times out.	

Service Register (COPSRV) SYS_BASE+$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	COUNT											
	Write	COP SERVICE REGISTER															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: _____ Date: _____

_____ Programmer: _____

Sheet 1 of 5

SIM

SIM System Control Register (SYS_CNTL)

This register is continued on the following page.

arch.h: ArchIO.Sim.ControlReg
registers.h: ArchIO_Sim_ControlReg

827 SPI PD	SPI PD
When set, these pull-up pins are enabled	0
When set, these pull-up pins are disabled	1

826 SCI PD	SCI PD
When set, these pull-up pins are enabled	0
When set, these pull-up pins are disabled	1

Control Signal Pull-Up Disable	CTRL PD
Pull-ups for the $\overline{DS}$, $\overline{PS}$, $\overline{RD}$, and the $\overline{WR}$ I/O pins are enabled	0
Pull-ups for the $\overline{DS}$, $\overline{PS}$, $\overline{RD}$, and the $\overline{WR}$ I/O pins are disabled	1

SCI_SEL	Serial Communications Interface Select
Note: This is Bit 9 of the 56F826	
0	When this bit is enabled, SCI is selected.
1	When this bit is disabled, SPI is selected.

SPI_SEL	Serial Peripheral Interface Select
Note: This is Bit 9 of the 56F827	
0	When this bit is set, SPI is selected.
1	When this bit is cleared, SCI is selected.

DATA PD	826 Data Bus I/O Pull-up Disable
0	Pull-ups for the data bus I/O pins are enabled
1	Pull-ups for the data bus I/O pins are disabled

Note: Bits 8 and 9 are different for the **56F826** and **56F827**. Please use accordingly.

826 System Control Register (SYS_CNTL) SYS_BASE +$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	SCI PD	CTRL PD	SCI_SEL	DATA PD	0	0	0	BOOT MAP_B	LVIE27	LVIE22	PD	RPD
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56F826

827 System Control Register (SYS_CNTL) SYS_BASE +$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	SPI PD	CTRL PD	SPI_SEL	0	0	0	0	BOOT MAP_B	LVIE27	LVIE22	PD	RPD
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56F827

Reserved Bits

SIM

SIM System Control Register (SYS_CNTL) Continued

arch.h: ArchIO.Sim.ControlReg
registers.h: ArchIO_Sim_ControlReg

Note: Bits 8 and 9 are different for the **56F826** and **56F827**. Please use accordingly.

Bootmap (BOOTMAP_B)				
The Bootmap bit and the MA and MB bits in the OMR register determine memory mapping for program memory.				
Mode No.	Bit Values			Description
	OMR Register		SYS_CNTL Register	
	MA	MB	BOOTMAP_B	
0A	0	0	0	Boot Mode
0B	0	0	1	1st 32K memory is internal 2nd 32K memory is external
1	0	1	x	Reserved
2	1	0	x	Reserved
3	1	1	x	Development–64 K Ext ROM
Note: Modes 0A and 0B are sub-modes of mode 0.				
Note: For details, see Table 3-39, "Program Memory Chip Operating Modes," on page 3-28.				

LVIE27	2.7 V Low Voltage Interrupt Enable
0	2.7 V low voltage interrupt disabled
1	2.7 V low voltage interrupt enabled

LVIE22	2.2 V Low Voltage Interrupt Enable
000	2.2 V low voltage interrupt disabled
001	2.2 V low voltage interrupt enabled

PD	Permanent STOP/WAIT Disable
0	STOP and WAIT instructions enabled
1	STOP and WAIT instructions act as no operations (NOPs)
Note: The part must be reset to clear this bit	

RPD	Re-programmable STOP/WAIT
0	STOP and WAIT instructions enabled
1	STOP and WAIT instructions act as no operations (NOPs)
Note: Software sets and clears this bit	

826 System Control Register (SYS_CNTL) SYS_BASE +$0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	SCI PD	CTRL PD	SCI_SEL	DATA PD	0	0	0	BOOT MAP_B	LVIE27	LVIE22	PD	RPD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56F826

827 System Control Register (SYS_CNTL) SYS_BASE +$0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	SPI PD	CTRL PD	SPI_SEL	0	0	0	0	BOOT MAP_B	LVIE27	LVIE22	PD	RPD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56F827

Reserved Bits

SIM

SIM System Status Register (SYS_STS)

arch.h: ArchIO.Sim.StatusReg
registers.h: ArchIO_Sim_StatuslReg

COP Reset	COPR
A COP timer generated system reset did not occur	0
A COP timer generated system reset has occurred	1
Note: A Power On Reset will clear this bit. To clear this bit with software, write a one to this bit. (To set, this bit, write a zero to this bit. Setting this bit will not cause a reset operation.)	

External Reset	EXTR
An external system reset did not occur.	0
An external system reset has occurredd. The previous system reset was caused by external RESET pin being asserted low.	1
Note: A Power-On Reset will clear this bit. To clear this bit with software, write one to this bit. (To set this bit, write zero to this bit. Setting this bit will not cause a reset operation.)	

POR	Power On Reset
0	A Power On Reset did not occur.
1	A Power On Reset occurred some time in the past
Note: To clear, write a one this bit. To set, write zero to this bit. Setting this bit will not cause a reset operation.	

LVIS27	2.7 V Low Voltage Interrupt Source
0	A Low Voltage Interrupt has not occurred.
1	A 2.7 V Low Voltage Interrupt is active, and the chip voltage has dropped below 2.7 V.
Note: To clear, write a one to this bit. To set, write a zero to this bit. *Setting this bit will cause an interrupt if the corresponding interrupt enable bit is set.*	

LVIS22	2.2 Low Voltage Interrupt Source
0	A Low Voltage Interrupt has not occurred.
1	A 2.2 V Low Voltage Interrupt is active, and the chip voltage has dropped below 2.2 V.

System Status Register (SYS_STS) SYS_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	COPR	EXTR	POR	LVIS27	LVIS22
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	—	—	—	0	0

Reserved Bits

SIM

SIM Most/Significant Half of JTAG Registers (MSH_ID_LSH_ID)

Part Number		
Binary Value	Decimal Equivalent	Part Number
1100111010	826	56F826

Representative Values Shown In Registers
The representative values shown in the MSH_ID and LSH_ID registers are for the 56F805 chip, version1.

Bits 15-12 Version Number

Bits 11-6 Design Center

Most Significant Half of JTAG_ID (MSH_ID) SYS_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	1
	Write																
	Reset:	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1

Bits 11-1 Manufacturer ID = 14 (Constant Value)

Bit 0 IEEE Requirement—Always = 1

Least Significant Half of JTAG_ID (LSH_ID) SYS_BASE +$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1
	Write																
	Reset	0	1	0	10	0	0	0	0	0	0	0	1	1	1	0	1

Reserved Bits

SIM

SIM Safe Storage Registers (SSREG)

SSREG0-4	Safe Storage Registers
0 - 65535	These registers are available to hold data.
Note: **SSREG0-4 regosters are not affected by internal or external resets.**	
Note: On power-up, the value in these registers is unknown.	

SIM Safe Storage Registers SYS_BASE +$18-$1C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read																
	Write																
	Reset	These registers are not reset, therefore they have no reset value.															

Safe Storage Register 0 (SSREG0)—Address: SYS_BASE + $18
Safe Storage Register 1 (SSREG1)—Address: SYS_BASE + $19
Safe Storage Register 2 (SSREG2)—Address: SYS_BASE + $1A
Safe Storage Register 3 (SSREG3)—Address: SYS_BASE + $1B
Safe Storage Register 4 (SSREG4)—Address: SYS_BASE + $1C

Application: _____ Date: _____

_____ Programmer: _____

Sheet

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

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

DSP56F826-827UM
Rev. 3

9/2005