Rev. 0

DSP56309

24-Bit Digital Signal Processor User's Manual

Motorola, Incorporated
Semiconductor Products Sector
DSP Division
6501 William Cannon Drive West
Austin, TX 78735-8598

This document (and other documents) can be viewed on the World Wide Web at http://www.mot.com/SPS/DSP/documentation/

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.

Order this document by **DSP56309UM/D**

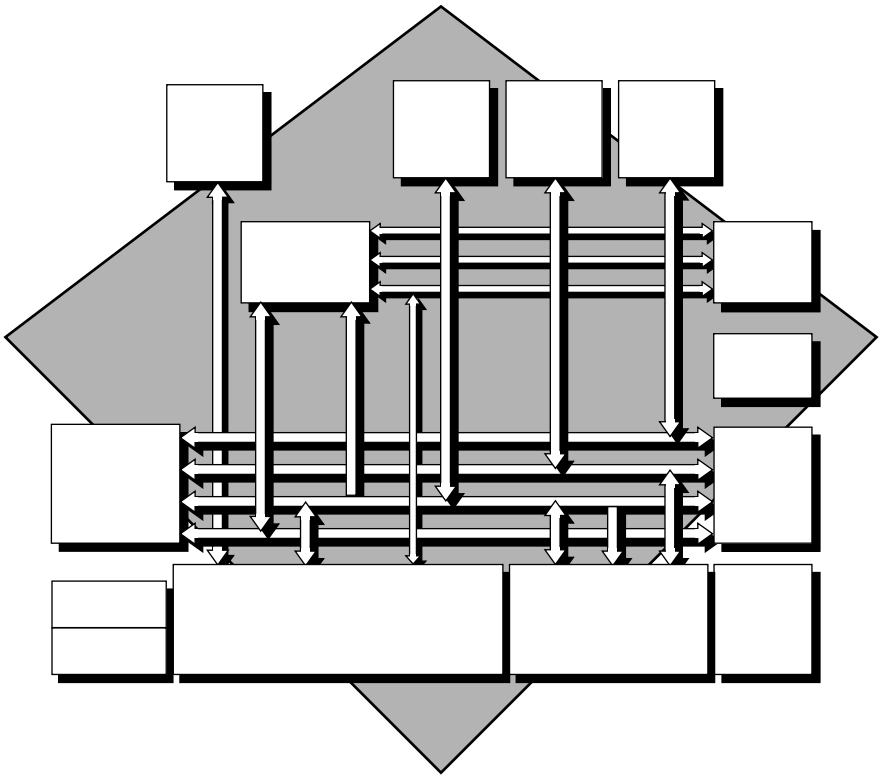
TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1

DSP56309 OVERVIEW



1.1 INTRODUCTION

This manual describes the DSP56309 24-bit digital signal processor (DSP), its memory, operating modes, and peripheral modules. The DSP56309 is an implementation of the DSP56300 core with a unique configuration of on-chip memory, cache, and peripherals.

This manual is intended to be used with the DSP56300 Family Manual (DSP56300FM/AD), which describes the central processing unit (CPU), core programming models, and instruction set details. DSP56309 Technical Data (DSP56309/D) provides electrical specifications, timing, pinout, and packaging descriptions of the DSP56309.

You can obtain these documents, as well as Motorola's DSP development tools, through a local Motorola Semiconductor Sales Office or authorized distributor.

To receive the latest information about this DSP, access the Motorola DSP home page at the address on the back cover of this document.

1.2 MANUAL ORGANIZATION

This manual contains the following sections and appendices:

Section 1—DSP56309 Overview

- Features list and block diagram
- Related documentation needed to use this chip
- The organization of this manual

Section 2—Signal/Connection Descriptions

- Signals on the DSP56309 pins and their functional groupings

Section 3—Memory Configuration

- DSP56309 memory spaces, RAM configuration, memory configuration bit settings, memory configurations, and memory maps

Section 4—Core Configuration

- Registers for configuring the DSP56300 core to program the DSP56309, in particular the interrupt vector locations and the operation of the interrupt priority registers
- Operating modes and how they affect the processor's program and data memories

Section 5—General-Purpose I/O

– DSP56309 general-purpose input/output (GPIO) capability and the programming model for the GPIO signals (operation, registers, and control)

Section 6—Host Interface (HI08)

– 8-bit host interface (HI08), including a quick reference to the HI08 programming model

Section 7—Enhanced Synchronous Serial Interface (ESSI)

– 24-bit ESSI, which provides two identical full duplex UART-style serial ports for communications with devices such as codecs, DSPs, microprocessors, and peripherals implementing the Motorola serial peripheral interface (SPI)

Section 8—Serial Communication Interface (SCI)

– 24-bit SCI, a full duplex serial port for serial communication to DSPs, microcontrollers, or other peripherals (such as modems or other RS-232 devices)

Section 9—Triple Timer Module

– The three identical internal timers/event counter devices

Section 10—On-Chip Emulation Module

– The On-Chip Emulation (OnCE™) module, which is accessed through the JTAG port

Section 11—JTAG Port

– Specifics of the Joint Test Action Group (JTAG) port on the DSP56309

Appendix A—Bootstrap Programs

– Bootstrap code used for the DSP56309

Appendix B—Equates

– Equates (I/O, HI08, SCI, ESSI, exception processing, timer, DMA, PLL, BIU, and interrupts) for the DSP56309

Appendix C—DSP56309 BSDL Listing

– BSDL listing for the DSP56309

Appendix D—Programming Reference

– Peripheral addresses, interrupt addresses, and interrupt priorities for the DSP56309

– Programming sheets listing the contents of the major DSP56309 registers for programmer's reference

1.3 MANUAL CONVENTIONS

This manual uses the following conventions:

- Bits within registers are always listed from most significant bit (MSB) to least significant bit (LSB).
- Bits within a register are indicated AA[n:m], n>m, 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 1. When a bit is described as "cleared," its value is 0.
- The word "assert" means that a high true (active high) signal is pulled high to V_{CC} or that a low true (active low) signal is pulled low to ground. The word "deassert" means that a high true signal is pulled low to ground or that a low true signal is pulled high to V_{CC}. See **Table 1-1**.

Table 1-1 High True/Low True Signal Conventions

Signal/Symbol	Logic State	Signal State	Voltage
$\overline{\text{PIN}}$[1]	True	Asserted	Ground[2]
$\overline{\text{PIN}}$	False	Deasserted	V_{CC}[3]
PIN	True	Asserted	V_{CC}
PIN	False	Deasserted	Ground

1. PIN is a generic term for any pin on the chip.
2. Ground is an acceptable low voltage level. See the appropriate data sheet for the range of acceptable low voltage levels (typically a TTL logic low).
3. V_{CC} is an acceptable high voltage level. See the appropriate data sheet for the range of acceptable high voltage levels (typically a TTL logic high).

- Pins or signals that are asserted low (made active when pulled to ground)
  - In text, have an overbar. For example, $\overline{\text{RESET}}$ is asserted low.
  - In code examples, have a tilde in front of their names. In **Example 1-1**, line 3 refers to the $\overline{\text{SS0}}$ signal (shown as `~SS0`).

- Sets of signals are indicated by the first and last signals in the set, for instance HA1–HA8.
- Code examples are displayed in a monospaced font, as shown in **Example 1-1**.

Example 1-1 Sample Code Listing

```
BFSET     #$0007,X:PCC; Configure:                    line 1
           ;  MISO0, MOSI0, SCK0 for SPI master       line 2
           ; ~SS0 as PC3 for GPIO                      line 3
```

- Hex values are indicated with a dollar sign ($) preceding the hex value. For example, $FFFFFF is the X memory address for the core interrupt priority register (IPR-C).
- The word ‘reset’ is used in four different contexts in this manual:
  - the reset signal, written as $\overline{\text{RESET}}$;
  - the reset instruction, written as RESET;
  - the reset operating state, written as Reset; and
  - the reset function, written as reset.

1.4 DSP56309 FEATURES

The DSP56309 is a member of the DSP56300 family of programmable CMOS DSPs. The DSP56309 uses the DSP56300 core, a high-performance engine with a single clock cycle per instruction. The DSP56300 core provides up to twice the performance of Motorola's popular DSP56000 core family, while retaining code compatibility.

The DSP56300 core family offers a new level of performance in speed and power provided by its rich instruction set and low-power dissipation, enabling a new generation of wireless, telecommunications, and multimedia products. The DSP56300 core is composed of the data arithmetic logic unit (Data ALU), address generation unit (AGU), program controller (PC), instruction cache controller, bus interface unit, direct memory access (DMA) controller, On-Chip Emulation (OnCE) module, and a PLL-based clock oscillator. Significant architectural enhancements to the DSP56300 core family include a barrel shifter, 24-bit addressing, an instruction cache, and DMA.

The DSP56300 core family members contain the DSP56300 core and additional modules. The modules are chosen from a library of standard pre-designed elements, such as memories and peripherals. New modules can be added to the library to meet customer

specifications. A standard interface between the DSP56300 core and the on-chip memory and peripherals supports a wide variety of memory and peripheral configurations.

The DSP56309 targets telecommunications applications, such as multi-line voice/data/fax processing, video conferencing, audio applications, control, and general digital signal processing.

1.5 DSP56309 CORE DESCRIPTION

The DSP56300 Family Manual fully describes core features; this manual describes pinout, memory, and peripheral features.

1.5.1 General Features

- 80/100 million instructions per second (MIPS) with a 80/100 MHz clock at 3.0 - 3.6 V
- Object-code compatible with the DSP56000 core
- Highly parallel instruction set

1.5.2 Hardware Debugging Support

- On-Chip Emulation (OnCE™) module
- Joint Test Action Group (JTAG) test access port (TAP)
- Address trace mode reflects internal program RAM accesses at the external port

1.5.3 Reduced Power Dissipation

- Very low-power CMOS design
- Wait and stop low-power standby modes
- Fully-static logic, operation frequency down to 0 Hz (dc)
- Optimized cycle-by-cycle power management circuitry (instruction-dependent, peripheral-dependent, and mode-dependent)

1.6 DSP56300 CORE FUNCTIONAL BLOCKS

The DSP56300 core provides the following functional blocks:

- Data ALU
- AGU
- PCU
- PLL and Clock Oscillator
- JTAG TAP and OnCE module
- Memory

In addition, the DSP56309 provides a set of on-chip peripherals, described in **Section 1.10**.

1.6.1 Data ALU

The Data ALU performs all the arithmetic and logical operations on data operands in the DSP56300 core. The components of the Data ALU are as follows:

- Fully pipelined 24 × 24-bit parallel multiplier-accumulator (MAC)
- Bit field unit, comprising a 56-bit parallel barrel shifter (fast shift and normalization; bit stream generation and parsing)
- Conditional ALU instructions
- 24-bit or 16-bit arithmetic support under software control
- Four 24-bit input general-purpose registers: X1, X0, Y1, and Y0
- Six Data ALU registers (A2, A1, A0, B2, B1, and B0) that are concatenated into two general-purpose, 56-bit accumulators, A and B, accumulator shifters
- Two data bus shifter/limiter circuits

1.6.1.1 Data ALU Registers

The Data ALU registers can be read or written over the X data bus (XDB) and the Y data bus (YDB) as 16- or 32-bit operands. The source operands for the Data ALU, which can be 16, 32, or 40 bits, always originate from Data ALU registers. The results of all Data ALU operations are stored in an accumulator.

All the Data ALU operations are performed in two clock cycles in pipeline fashion so that a new instruction can be initiated in every clock, yielding an effective execution rate of one instruction per clock cycle. The destination of every arithmetic operation can be used as a source operand for the immediately following operation without penalty.

1.6.1.2 Multiplier-Accumulator (MAC)

The MAC unit comprises the main arithmetic processing unit of the DSP56300 core and performs all of the calculations on data operands. For arithmetic instructions, the unit accepts as many as three input operands and outputs one 56-bit result of the following form, Extension:Most Significant Product:Least Significant Product (EXT:MSP:LSP).

The multiplier executes 24-bit × 24-bit, parallel, fractional multiplies between twos-complement signed, unsigned, or mixed operands. The 48-bit product is right-justified and added to the 56-bit contents of either the A or B accumulator. A 56-bit result can be stored as a 24-bit operand. The LSP can either be truncated or rounded into the MSP. Rounding is performed if specified.

1.6.2 Address Generation Unit (AGU)

The AGU performs the effective address calculations using integer arithmetic necessary to address data operands in memory and contains the registers that generate the addresses. It implements four types of arithmetic: linear, modulo, multiple wrap-around modulo, and reverse-carry. The AGU operates in parallel with other chip resources to minimize address-generation overhead.

The AGU is divided into two halves, each with its own address arithmetic logic unit (Address ALU). Each Address ALU has four sets of register triplets, and each register triplet is composed of an address register, an offset register, and a modifier register. The two Address ALUs are identical. Each contains a 16-bit full adder (called an offset adder).

A second full adder (called a modulo adder) adds the summed result of the first full adder to a modulo value that is stored in its respective modifier register. A third full adder (called a reverse-carry adder) is also provided.

The offset adder and the reverse-carry adder are in parallel and share common inputs. The only difference between them is that they carry propagates in opposite directions. Test logic determines which of the three summed results of the full adders is output.

Each Address ALU can update one address register from its respective address register file during one instruction cycle. The contents of the associated modifier register

specifies the type of arithmetic to be used in the address register update calculation. The modifier value is decoded in the Address ALU.

1.6.3 Program Control Unit (PCU)

The PCU performs instruction prefetch, instruction decoding, hardware DO loop control, and exception processing. The PCU implements a seven-stage pipeline and controls the different processing states of the DSP56300 core. The PCU consists of three hardware blocks:

- Program decode controller (PDC)
- Program address generator (PAG)
- Program interrupt controller

The PDC decodes the 24-bit instruction loaded into the instruction latch and generates all signals necessary for pipeline control. The PAG contains all the hardware needed for program address generation, system stack, and loop control. The PIC arbitrates among all interrupt requests (internal interrupts, as well as the five external requests $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$, $\overline{\text{IRQD}}$, and $\overline{\text{NMI}}$) and generates the appropriate interrupt vector address.

PCU features include the following:

- Position Independent Code (PIC) support
- Addressing modes optimized for DSP applications (including immediate offsets)
- On-chip instruction cache controller
- On-chip memory-expandable hardware stack
- Nested hardware DO loops
- Fast auto-return interrupts

The PCU implements its functions using the following registers:

- PC—program counter register
- SR—status register
- LA—loop address register
- LC—loop counter register
- VBA—vector base address register
- SZ—size register

- SP—stack pointer
- OMR—operating mode register
- SC—stack counter register

The PCU also includes a hardware system stack (SS).

1.6.4 PLL and Clock Oscillator

The clock generator in the DSP56300 core is composed of two main blocks: the PLL, which performs clock input division, frequency multiplication, and skew elimination; and the clock generator (CLKGEN), which performs low-power division and clock pulse generation.

- Allows change of low-power divide factor (DF) without loss of lock
- Output clock with skew elimination

The PLL allows the processor to operate at a high internal clock frequency using a low-frequency clock input, a feature that offers two immediate benefits:

- A lower-frequency clock input reduces the overall electromagnetic interference generated by a system.
- The ability to oscillate at different frequencies reduces costs by eliminating the need to add additional oscillators to a system.

1.6.5 JTAG TAP and OnCE Module

The DSP56300 core provides a dedicated user-accessible Test Access Port (TAP) that is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*. Problems associated with testing high density circuit boards have led to development of this standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The DSP56300 core implementation supports circuit-board test strategies based on this standard.

The test logic includes a TAP consisting of four dedicated signals, a 16-state controller, and three test data registers. A boundary scan register links all device signals into a single shift register. The test logic, implemented utilizing static logic design, is independent of the device system logic. More information on the JTAG port is provided in **Section 11—JTAG Port.**

The OnCE module provides a means of interacting with the DSP56300 core and its peripherals non-intrusively so that a user can examine registers, memory, or on-chip peripherals. This facilitates hardware and software development on the DSP56300 core processor. OnCE module functions are provided through the JTAG TAP signals. More information about the OnCE module is provided in **Section 10—On-Chip Emulation Module.**

1.6.6 On-Chip Memory

The memory space of the DSP56300 core is partitioned into program memory space, X data memory space, and Y data memory space. The data memory space is divided into X data memory and Y data memory in order to work with the two Address ALUs and to feed two operands simultaneously to the Data ALU. Memory space includes internal RAM and ROM and can be expanded off-chip under software control. More information about the internal memory is provided in **Section 3—Memory Configuration**.

Program RAM, instruction cache, X data RAM, and Y data RAM size are programmable, as indicated in **Table 1-2**.

Table 1-2 On Chip Memory

Instruction Cache	Switch Mode	Program RAM Size	Instruction Cache Size	X Data RAM Size	Y Data RAM Size
disabled	disabled	20K × 24-bit	0	7K × 24-bit	7K × 24-bit
enabled	disabled	19K× 24-bit	1K × 24-bit	7K × 24-bit	7K × 24-bit
disabled	enabled	24K × 24-bit	0	5K × 24-bit	5K × 24-bit
enabled	enabled	23K × 24-bit	1K × 24-bit	5K × 24-bit	5K × 24-bit

There is an on-chip 192 x 24-bit bootstrap ROM.

1.6.7 Off-Chip Memory Expansion

Memory can be expanded off-chip to do the following:

- Data memory expansion to two 256K × 24-bit word memory spaces (or up to two4 M × 24-bit word memory spaces by using the address attribute AA0–AA3 signals)
- Program memory expansion to one 256K × 24-bit words memory space (or up to one 4 M × 24-bit word memory space by using the address attribute AA0–AA3 signals)

Additional features of off-chip memory include the following:

- External memory expansion port
- Simultaneous glueless interface to static random access memory (SRAM) and dynamic random access memory (DRAM)
- Supports interleaved, non-interfering access to both types of memory without losing in-page DRAM access, including DMA-driven access

1.7 INTERNAL BUSES

The following buses provide data exchange between the functional blocks of the core:

- Peripheral I/O expansion bus (PIO_EB) to peripherals
- Program memory expansion bus (PM_EB) to Program RAM
- X memory expansion bus (XM_EB) to X memory
- Y memory expansion bus (YM_EB) to Y memory
- Global data bus (GDB) between PCU and other core structures
- Program data bus (PDB) for carrying program data throughout the core
- X memory data bus (XDB) for carrying X data throughout the core
- Y memory data bus (YDB) for carrying Y data throughout the core
- Program address bus (PAB) for carrying program memory addresses throughout the core
- X memory address bus (XAB) for carrying X memory addresses throughout the core
- Y memory address bus (YAB) for carrying Y memory addresses throughout the core

All internal buses on the DSP56300 family members are 16-bit buses except the PDB, which is a 24-bit bus. **Figure 1-1** shows a block diagram of the DSP56309.

1.8 DSP56309 BLOCK DIAGRAM

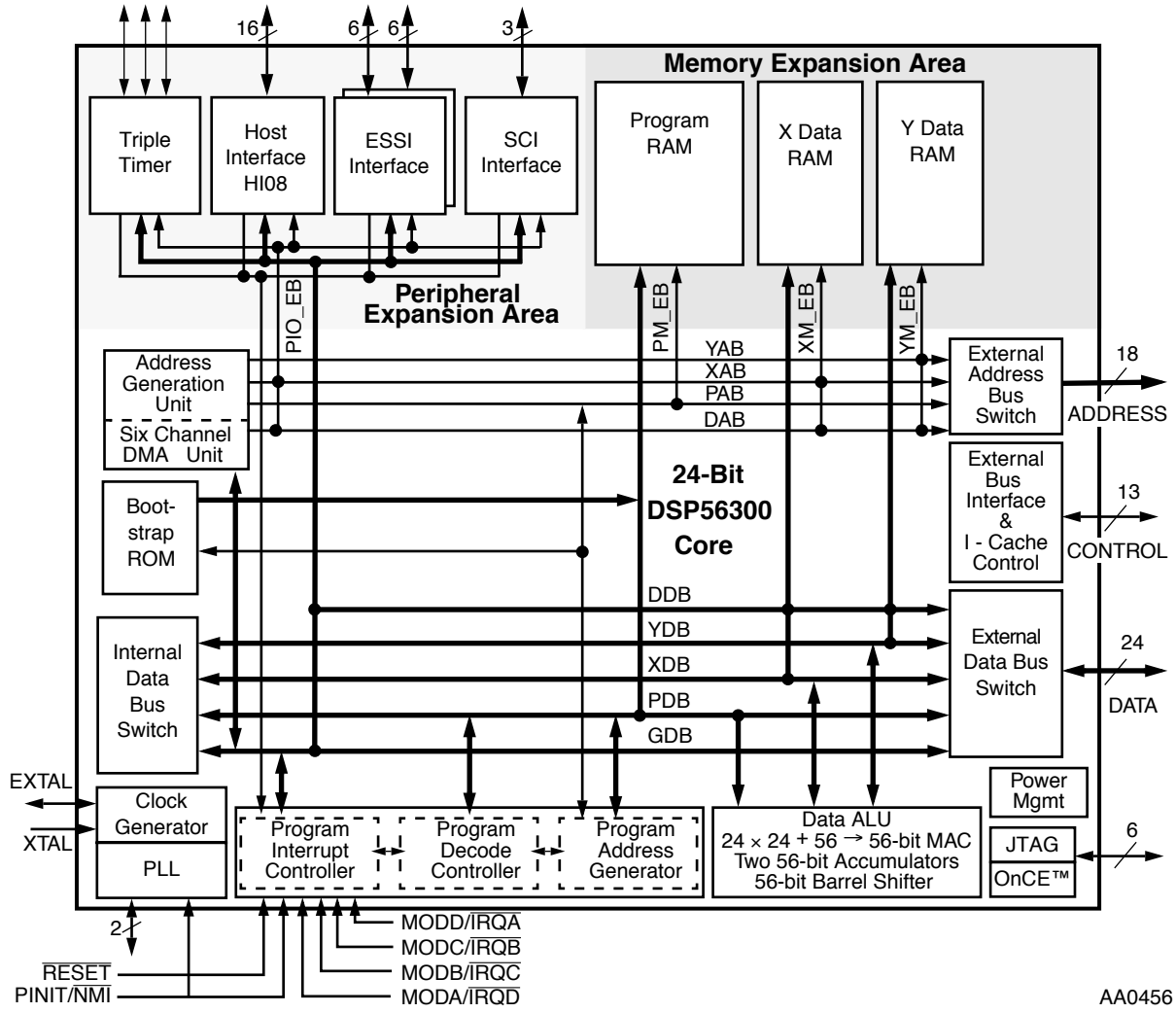


Figure 1-1 DSP56309 Block Diagram

Note: See **Section 1.6.6 On-Chip Memory** on page 1-12 for details on memory size.

1.9 DIRECT MEMORY ACCESS (DMA)

The DMA block has the following features:

- Six DMA channels supporting internal and external accesses
- One-, two-, and three-dimensional transfers (including circular buffering)
- End-of-block-transfer interrupts
- Triggering from interrupt lines, all peripherals, and DMA channels

1.10 DSP56309 ARCHITECTURE OVERVIEW

The DSP56309 performs a wide variety of fixed-point digital signal processing functions. In addition to the core features previously discussed, the DSP56309 provides the following peripherals:

- Enhanced DSP56000-like 8-bit parallel host interface (HI08) supports a variety of buses (e.g., industry standard architecture) and provides glueless connection to a number of industry standard microcomputers, microprocessors, and DSPs
- Two enhanced synchronous serial interfaces (ESSI0 and ESSI1), each with one receiver and three transmitters (allows six-channel home theater)
- Serial communications interface (SCI) with baud rate generator
- Triple timer module
- Up to 34 programmable general purpose input/output (GPIO) pins, depending on which peripherals are enabled

1.10.1 GPIO Functionality

The GPIO port consists of as many as thirty-four programmable signals, all of which are also used by the peripherals (HI08, ESSI, SCI, and timer). There are no dedicated GPIO signals. Peripheral pins are configured as GPIO inputs after any reset. (Data in the port data register is not affected by a reset.) The GPIO functionality for each peripheral is controlled by three memory-mapped registers per peripheral. The techniques for register programming for all GPIO functionality is very similar between these interfaces.

1.10.2 Host Interface (HI08)

The HI08 is a byte-wide, full-duplex, double-buffered, parallel port that can connect directly to the data bus of a host processor. The HI08 supports a variety of buses and connects to a number of industry-standard DSPs, microcomputers, and microprocessors without requiring additional logic.

The DSP core treats the HI08 as a memory-mapped peripheral occupying eight 24-bit words in data memory space. The DSP can use the HI08 as a memory-mapped peripheral, using either standard polled or interrupt programming techniques. Separate transmit and receive data registers are double-buffered to allow the DSP and host processor to transfer data efficiently at high speed. Memory mapping allows DSP core communication with the HI08 registers using standard instructions and addressing modes.

1.10.3 Enhanced Synchronous Serial Interface (ESSI)

On the DSP56309 are two independent and identical ESSIs. Each ESSI has a full-duplex serial port for communication with a variety of serial devices, including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals that implement the Motorola SPI. The ESSI consists of independent transmitter and receiver sections and a common ESSI clock generator.

The capabilities of the ESSI include the following:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs
- Normal mode operation using frame sync
- Network mode operation with as many as 32 time slots
- Programmable word length (8, 12, or 16 bits)
- Program options for frame synchronization and clock generation
- One receiver and three transmitters per ESSI allows six-channel home theater

1.10.4 Serial Communications Interface (SCI)

The DSP56309's SCI provides a full-duplex port for serial communication with other DSPs, microprocessors, or peripherals such as modems. The SCI interfaces without

additional logic to peripherals that use TTL-level signals. With a small amount of additional logic, the SCI can connect to peripheral interfaces that have non-TTL level signals, such as the RS-232C, RS-422, etc.

This interface uses three dedicated signals: transmit data (TXD), receive data (RXD), and SCI serial clock (SCLK). It supports industry-standard asynchronous bit rates and protocols, as well as high-speed synchronous data transmission (up to 8.25 Mbps for a 66 MHz clock). The asynchronous protocols supported by the SCI include a multidrop mode for master/slave operation with wakeup on idle line and wakeup on address bit capability. This mode allows the DSP56309 to share a single serial line efficiently with other peripherals.

The SCI consists of separate transmit and receive sections that can operate asynchronously with respect to each other. A programmable baud-rate generator provides the transmit and receive clocks. An enable vector and an interrupt vector have been included so that the baud-rate generator can function as a general purpose timer when it is not being used by the SCI or when the interrupt timing is the same as that used by the SCI.

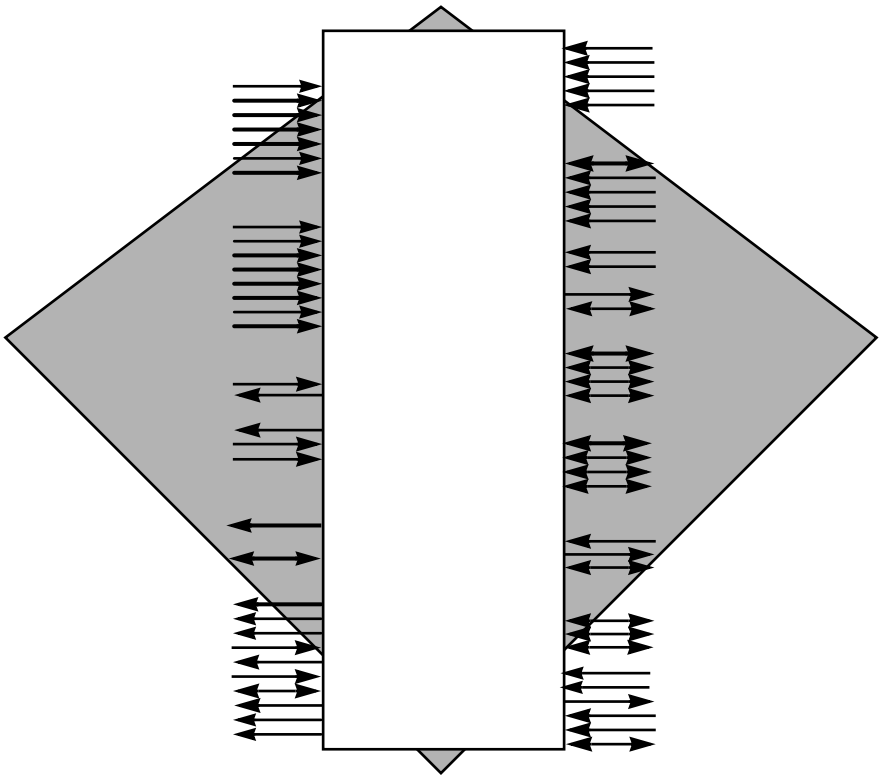
1.10.5 Timer Module

The triple timer module is composed of a common 21-bit prescaler and three independent and identical general-purpose 24-bit timer/event counters, each with its own memory-mapped register set.

Each timer has a single signal that can function as a GPIO signal or as a timer signal. Each timer can use internal or external clocking and can interrupt the DSP after a specified number of events (clocks) or can signal an external device after counting internal events. Each timer connects to the external world through one bidirectional signal. When this signal is configured as an input, the timer can function as an external event counter or measures external pulse width/signal period. When the signal is used as an output, the timer can function as either a timer, a watchdog, or a Pulse Width Modulator (PWM).

SECTION 2

SIGNAL/CONNECTION DESCRIPTIONS



2.1 SIGNAL GROUPINGS

The DSP56309 input and output signals are organized into functional groups, as shown in **Table 2-1** and as illustrated in **Figure 2-1**.

The DSP56309 is operated from a 3 V supply.

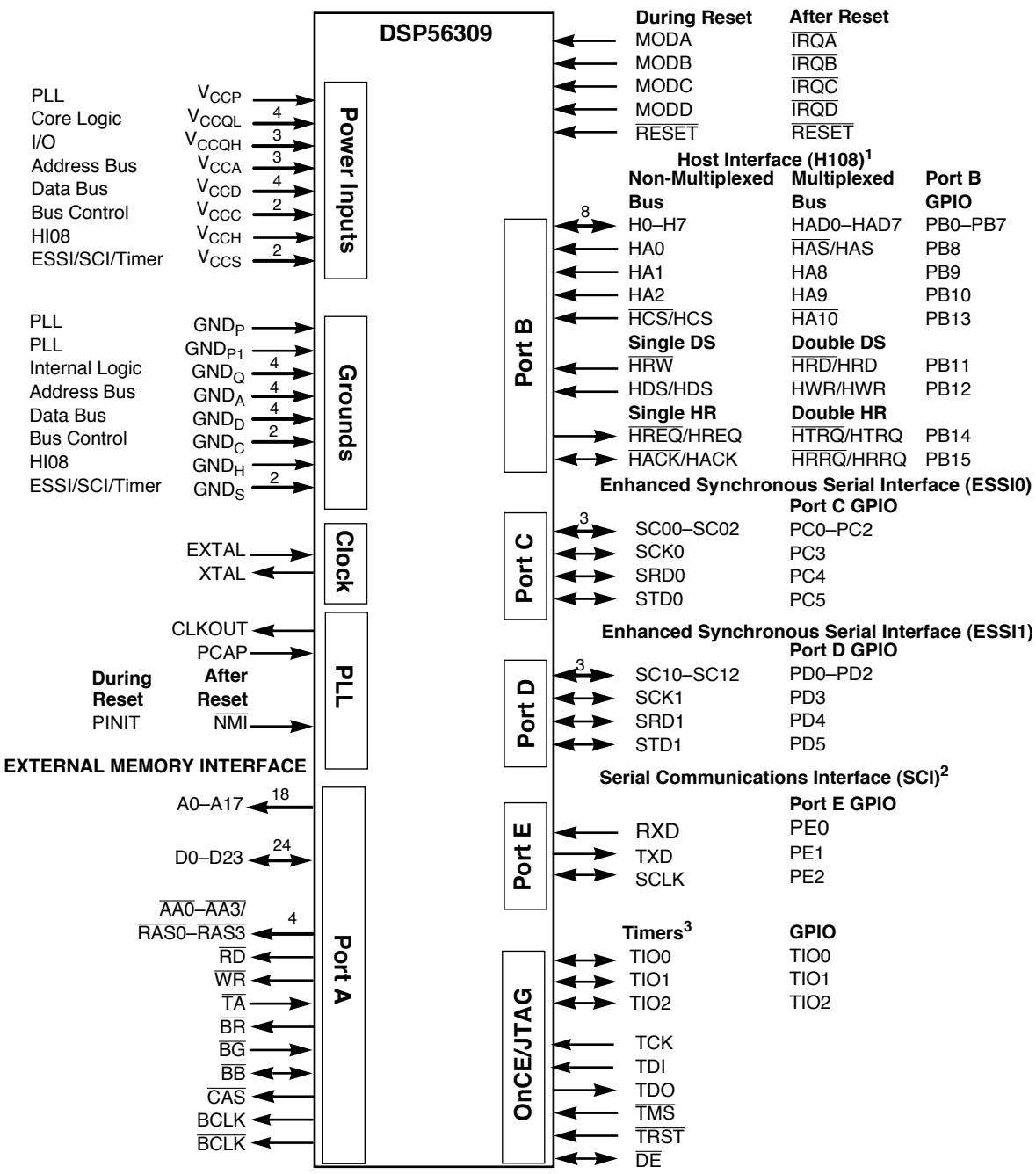
Table 2-1 DSP56309 Functional Signal Groupings

Functional Group		Number of Signals	Detailed Description
Power (V_{CC})		20	**Table 2-2**
Ground (GND)		19	**Table 2-3**
Clock		2	**Table 2-4**
PLL		3	**Table 2-5**
Address Bus	Port A[1]	18	**Table 2-6**
Data Bus		24	**Table 2-7**
Bus Control		13	**Table 2-8**
Interrupt and Mode Control		5	**Table 2-9**
Host Interface (HI08)	Port B[2]	16	**Table 2-11**
Enhanced Synchronous Serial Interface (ESSI)	Ports C and D[3]	12	**Table 2-12** and **Table 2-13**
Serial Communication Interface (SCI)	Port E[4]	3	**Table 2-14**
Timer		3	**Table 2-15**
OnCE/JTAG Port		6	**Table 2-16**

Note:
1. Port A signals define the external memory interface port, including the external address bus, data bus, and control signals.
2. Port B signals are the HI08 port signals multiplexed with the GPIO signals.
3. Port C and D signals are the two ESSI port signals multiplexed with the GPIO signals.
4. Port E signals are the SCI port signals multiplexed with the GPIO signals.

Figure 2-1 is a diagram of DSP56309 signals by functional group.

Signal Groupings



Notes:
1. The HI08 port supports a non-multiplexed or a multiplexed bus, single or double Data Strobe (DS), and single or double Host Request (HR) configurations. Since each of these modes is configured independently, any combination of these modes is possible. These HI08 signals can also be configured alternately as GPIO signals (PB0–PB15). Signals with dual designations (e.g., HAS/HAS) have configurable polarity.
2. The ESSI0, ESSI1, and SCI signals are multiplexed with the Port C GPIO signals (PC0–PC5), Port D GPIO signals (PD0–PD5), and Port E GPIO signals (PE0–PE2), respectively.
3. TIO0–TIO2 can be configured as GPIO signals.

AA0601

Figure 2-1 Signals Identified by Functional Group

2.2 POWER

Power input descriptions for the DSP56309 are listed in **Table 2-2**.

Table 2-2 Power Inputs

Power Name	Description
V_{CCP}	**PLL Power**—V_{CCP} is power dedicated for phase-locked loop (PLL) use. The voltage should be well regulated, and the input should be provided with an extremely low impedance path to the V_{CC} power rail. V_{CCP} should be bypassed to GND_P by a stabilizing capacitor located as close as possible to the chip package. There is one V_{CCP} input.
V_{CCQL} (4)	**Quiet Core (Low) Power**—V_{CCQL} is an isolated power for the core processing logic. This input must be isolated externally from all other chip power inputs. The user must provide adequate external decoupling capacitors. There are four V_{CCQL} inputs.
V_{CCQH} (3)	**Quiet External (High) Power**—V_{CCQH} is a quiet power source for I/O lines. This input must be tied externally to all other chip power inputs, *except V_{CCQL}*. The user must provide adequate external decoupling capacitors. There are three V_{CCQH} inputs.
V_{CCA} (3)	**Address Bus Power**—V_{CCA} is an isolated power for sections of the address bus I/O drivers. This input must be tied externally to all other chip power inputs *except V_{CCQL}*. The user must provide adequate external decoupling capacitors. There are three V_{CCA} inputs.
V_{CCD} (4)	**Data Bus Power**—V_{CCD} is an isolated power for sections of the data bus I/O drivers. This input must be tied externally to all other chip power inputs. The user must provide adequate external decoupling capacitors. There are four V_{CCD} inputs.
V_{CCC} (2)	**Bus Control Power**—V_{CCC} is an isolated power for the bus control I/O drivers. This input must be tied externally to all other chip power inputs *except V_{CCQL}*. The user must provide adequate external decoupling capacitors. There are two V_{CCC} inputs.
V_{CCH}	**Host Power**—V_{CCH} is an isolated power for the HI08 I/O drivers. This input must be tied externally to all other chip power inputs *except V_{CCQL}*. The user must provide adequate external decoupling capacitors. There is one V_{CCH} input.

Table 2-2 Power Inputs (Continued)

Power Name	Description
V_{CCS} (2)	**ESSI, SCI, and Timer Power**—V_{CCS} is an isolated power for the ESSI, SCI, and timer I/O drivers. This input must be tied externally to all other chip power inputs *except* V_{CCQL}. The user must provide adequate external decoupling capacitors. There are two V_{CCS} inputs.
Note: These designations are package-dependent. Some packages connect all V_{CC} inputs except V_{CCP} to each other internally. On those packages, all power input, except V_{CCP}, are labeled V_{CC}. The number of connections indicated in this table are minimum values; the total V_{CC} connections are package-dependent.	

2.3 GROUND

Ground descriptions for the DSP56309 are listed in **Table 2-3**.

Table 2-3 Grounds

Ground Name	Description
GND_P	**PLL Ground**—GND_P is a ground dedicated for PLL use. The connection should be provided with an extremely low-impedance path to ground. V_{CCP} should be bypassed to GND_P by a 0.47 μF capacitor located as close as possible to the chip package. There is one GND_P connection.
GND_{P1}	**PLL Ground 1**—GND_{P1} is a ground dedicated for PLL use. The connection should be provided with an extremely low-impedance path to ground. There is one GND_{P1} connection.
GND_Q (4)	**Quiet Ground**—GND_Q is an isolated ground for the internal processing logic. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors. There are four GND_Q connections.
GND_A (4)	**Address Bus Ground**—GND_A is an isolated ground for sections of the address bus I/O drivers. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors. There are four GND_A connections.

Table 2-3 Grounds (Continued)

Ground Name	Description
GND_D (4)	**Data Bus Ground**—GND_D is an isolated ground for sections of the data bus I/O drivers. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors. There are four GND_D connections.
GND_C (2)	**Bus Control Ground**—GND_C is an isolated ground for the bus control I/O drivers. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors. There are two GND_C connections.
GND_H	**Host Ground**—GND_H is an isolated ground for the HI08 I/O drivers. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors. There is one GND_H connection.
GND_S (2)	**ESSI, SCI, and Timer Ground**—GND_S is an isolated ground for the ESSI, SCI, and timer I/O drivers. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors. There are two GND_S connections.
Note: These designations are package-dependent. Some packages connect all GND inputs, except GND_P and GND_{P1}, to each other internally. On those packages, all ground connections, except GND_P and GND_{P1}, are labeled GND. The number of connections indicated in this table are minimum values; the total GND connections are package-dependent.	

2.4 CLOCK

Clock Signal descriptions for the DSP56309 are listed in **Table 2-4**.

Table 2-4 Clock Signals

Signal Name	Type	State During Reset	Signal Description
EXTAL	Input	Input	**External Clock/Crystal Input**—EXTAL interfaces the internal crystal oscillator input to an external crystal or an external clock.

Table 2-4 Clock Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
XTAL	Output	Chip-driven	**Crystal Output**—XTAL connects the internal crystal oscillator output to an external crystal. If an external clock is used, leave XTAL unconnected.

2.5 PHASE-LOCKED LOOP (PLL)

Phase-locked loop signal descriptions are listed in **Table 2-5**.

Table 2-5 Phase-Locked Loop Signals

Signal Name	Type	State During Reset	Signal Description
PCAP	Input	Input	**PLL Capacitor**—PCAP is an input connecting an off-chip capacitor to the PLL filter. Connect one capacitor terminal to PCAP and the other terminal to V_{CCP}. If the PLL is not used, PCAP can be tied to V_{CC}, tied to GND, or left floating.
CLKOUT	Output	Chip-driven	**Clock Output**—CLKOUT provides an output clock synchronized to the internal core clock phase. If the PLL is enabled and both the multiplication and division factors equal one, then CLKOUT is also synchronized to EXTAL. If the PLL is disabled, the CLKOUT frequency is half the frequency of EXTAL.

Table 2-5 Phase-Locked Loop Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
PINIT/$\overline{\text{NMI}}$	Input	Input	**PLL Initial/Non-Maskable Interrupt**—During assertion of $\overline{\text{RESET}}$, the value of PINIT/$\overline{\text{NMI}}$ is written into the PLL Enable (PEN) bit of the PLL control register, determining whether the PLL is enabled or disabled. After $\overline{\text{RESET}}$ deassertion and during normal instruction processing, the PINIT/$\overline{\text{NMI}}$ Schmitt-trigger input is a negative-edge-triggered Non-Maskable Interrupt (NMI) request internally synchronized to CLKOUT.

2.6 EXTERNAL MEMORY EXPANSION PORT (PORT A)

When the DSP56309 enters a low-power standby mode (stop or wait), it releases bus mastership and tri-states the relevant Port A signals: A0–A17, D0–D23, AA0/$\overline{\text{RAS0}}$–AA3/$\overline{\text{RAS3}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{BB}}$, $\overline{\text{CAS}}$, BCLK, $\overline{\text{BCLK}}$.

2.6.1 External Address Bus

External address bus signals for the DSP56309 are listed in **Table 2-6**.

Table 2-6 External Address Bus Signals

Signal Name	Type	State During Reset	Signal Description
A0–A17	Output	Tri-stated	**Address Bus**—When the DSP is the bus master, A0–A17 are active-high outputs that specify the address for external program and data memory accesses. Otherwise, the signals are tri-stated. To minimize power dissipation, A0–A17 do not change state when external memory spaces are not being accessed.

2.6.2 External Data Bus

External data bus signals for the DSP56309 are listed in **Table 2-7**.

Table 2-7 External Data Bus Signals

Signal Name	Type	State During Reset	Signal Description
D0–D23	Input/ Output	weakly driven by bus keeper	**Data Bus**—When the DSP is the bus master, D0–D23 are active-high, bidirectional input/outputs that provide the bidirectional data bus for external program and data memory accesses. Otherwise, D0–D23 are weakly driven by the bus keeper.

2.6.3 External Bus Control

External bus control signal descriptions for the DSP56309 are listed in **Table 2-8**.

Table 2-8 External Bus Control Signals

Signal Name	Type	State During Reset	Signal Description
AA0–AA3/ $\overline{\text{RAS0}}$–$\overline{\text{RAS3}}$	Output	Tri-stated	**Address Attribute or Row Address Strobe**—When defined as AA, these signals can be used as chip selects or additional address lines. When defined as $\overline{\text{RAS}}$, these signals can be used as $\overline{\text{RAS}}$ for DRAM interface. These signals are tri-statable outputs with programmable polarity.
$\overline{\text{RD}}$	Output	Tri-stated	**Read Enable**—When the DSP is the bus master, $\overline{\text{RD}}$ is an active-low output that is asserted to read external memory on the data bus (D0–D23). Otherwise, $\overline{\text{RD}}$ is tri-stated.
$\overline{\text{WR}}$	Output	Tri-stated	**Write Enable**—When the DSP is the bus master, $\overline{\text{WR}}$ is an active-low output that is asserted to write external memory on the data bus (D0–D23). Otherwise, the signals are tri-stated.

Table 2-8 External Bus Control Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{TA}$	Input	Ignored Input	**Transfer Acknowledge**—If the DSP56309 is the bus master and there is no external bus activity, or the DSP56309 is not the bus master, the $\overline{TA}$ input is ignored. The $\overline{TA}$ input is a data transfer acknowledge (DTACK) function that can extend an external bus cycle indefinitely. Any number of wait states (1, 2,..., infinity) can be added to the wait states inserted by the BCR by keeping $\overline{TA}$ deasserted. In typical operation, $\overline{TA}$ is deasserted at the start of a bus cycle, is asserted to enable completion of the bus cycle, and is deasserted before the next bus cycle. The current bus cycle completes one clock period after $\overline{TA}$ is asserted synchronous to CLKOUT. The number of wait states is determined by the $\overline{TA}$ input or by the BCR, whichever is longer. The BCR can be used to set the minimum number of wait states in external bus cycles. In order to use the $\overline{TA}$ functionality, the BCR must be programmed to at least one wait state. A zero wait state access cannot be extended by $\overline{TA}$ deassertion; otherwise, improper operation can result. $\overline{TA}$ can operate synchronously or asynchronously depending on the setting of the TAS bit in the OMR. You must not use $\overline{TA}$ functionality while performing DRAM type accesses; otherwise, improper operation can result.

Table 2-8 External Bus Control Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{BR}$	Output	Output (deasserted)	**Bus Request**—$\overline{BR}$ is an active-low output, never tri-stated. $\overline{BR}$ is asserted when the DSP requests bus mastership. $\overline{BR}$ is deasserted when the DSP no longer needs the bus. $\overline{BR}$ can be asserted or deasserted independent of whether the DSP56309 is a bus master or a bus slave. Bus "parking" allows $\overline{BR}$ to be deasserted even though the DSP56309 is the bus master; see the description of bus "parking" in the $\overline{BB}$ signal description. The bus request hole (BRH) bit in the BCR allows $\overline{BR}$ to be asserted under software control even though the DSP does not need the bus. $\overline{BR}$ is typically sent to an external bus arbitrator that controls the priority, parking, and tenure of each master on the same external bus. $\overline{BR}$ is only affected by DSP requests for the external bus, never for the internal bus. During hardware reset, $\overline{BR}$ is deasserted and the arbitration is reset to the bus slave state.
$\overline{BG}$	Input	Ignored Input	**Bus Grant**—$\overline{BG}$ is an active-low input. $\overline{BG}$ must be asserted/deasserted synchronous to CLKOUT for proper operation. $\overline{BG}$ is asserted by an external bus arbitration circuit when the DSP56309 becomes the next bus master. When $\overline{BG}$ is asserted, the DSP56309 must wait until $\overline{BB}$ is deasserted before taking bus mastership. When $\overline{BG}$ is deasserted, bus mastership is typically given up at the end of the current bus cycle. This can occur in the middle of an instruction that requires more than one external bus cycle for execution.

Table 2-8 External Bus Control Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{BB}$	Input/ Output	Input	**Bus Busy**—$\overline{BB}$ is a bidirectional active-low input/output and must be asserted and deasserted synchronous to CLKOUT. $\overline{BB}$ indicates that the bus is active. Only after $\overline{BB}$ is deasserted can the pending bus master become the bus master (and then assert the signal again). The bus master can keep $\overline{BB}$ asserted after ceasing bus activity regardless of whether $\overline{BR}$ is asserted or deasserted. This is called "bus parking" and allows the current bus master to reuse the bus without rearbitration until another device requires the bus. The deassertion of $\overline{BB}$ is done by an "active pull-up" method (i.e., $\overline{BB}$ is driven high and then released and held high by an external pull-up resistor). $\overline{BB}$ requires an external pull-up resistor.
$\overline{CAS}$	Output	Tri-stated	**Column Address Strobe**—When the DSP is the bus master, $\overline{CAS}$ is an active-low output used by DRAM to strobe the column address. Otherwise, if the Bus Mastership Enable (BME) bit in the DRAM Control Register is cleared, the signal is tri-stated.
BCLK	Output	Tri-stated	**Bus Clock**—When the DSP is the bus master, BCLK is an active-high output. BCLK is active as a sampling signal when the program address tracing mode is enabled (by setting the ATE bit in the OMR). When BCLK is active and synchronized to CLKOUT by the internal PLL, BCLK precedes CLKOUT by 1/4 of a clock cycle. The BCLK rising edge can be used to sample the internal program memory access on the A0–A23 address lines.
$\overline{BCLK}$	Output	Tri-stated	**Bus Clock Not**—When the DSP is the bus master, $\overline{BCLK}$ is an active-low output and is the inverse of the BCLK signal. Otherwise, the signal is tri-stated.

2.7 INTERRUPT AND MODE CONTROL

The interrupt and mode control signals select the chip's operating mode as it comes out of hardware reset. After $\overline{\text{RESET}}$ is deasserted, these inputs are hardware interrupt request lines.

Table 2-9 Interrupt and Mode Control

Signal Name	Type	State During Reset	Signal Description
$\overline{\text{RESET}}$	Input	Input	**Reset—**$\overline{\text{RESET}}$ is an active-low, Schmitt-trigger input. Deassertion of $\overline{\text{RESET}}$ is internally synchronized to the clock out (CLKOUT). When asserted, the chip is placed in the Reset state and the internal phase generator is reset. The Schmitt-trigger input allows a slowly rising input (such as a capacitor charging) to reset the chip reliably. If $\overline{\text{RESET}}$ is deasserted synchronous to CLKOUT, exact start-up timing is guaranteed, allowing multiple processors to start synchronously and operate together in lock-step. When the $\overline{\text{RESET}}$ signal is deasserted, the initial chip operating mode is latched from the MODA, MODB, MODC, and MODD inputs. The $\overline{\text{RESET}}$ signal must be asserted after power up.
MODA $\overline{\text{IRQA}}$	Input	Input	**Mode Select A**—MODA is an active-low Schmitt-trigger input, internally synchronized to CLKOUT. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into the OMR when the $\overline{\text{RESET}}$ signal is deasserted. External Interrupt Request A—After reset, this signal becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If $\overline{\text{IRQA}}$ is asserted synchronous to CLKOUT, multiple processors can be resynchronized using the WAIT instruction and asserting $\overline{\text{IRQA}}$ to exit the wait state. If the processor is in the stop standby state and $\overline{\text{IRQA}}$ is asserted, the processor exits the stop state.

Table 2-9 Interrupt and Mode Control (Continued)

Signal Name	Type	State During Reset	Signal Description
MODB	Input	Input	**Mode Select B—**MODB is an active-low Schmitt-trigger input, internally synchronized to CLKOUT. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into OMR when the $\overline{\text{RESET}}$ signal is deasserted.
$\overline{\text{IRQB}}$			**External Interrupt Request B**—After hardware reset, this signal becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If $\overline{\text{IRQB}}$ is asserted synchronous to CLKOUT, multiple processors can be resynchronized using the WAIT instruction and asserting $\overline{\text{IRQB}}$ to exit the wait state.
MODC	Input	Input	**Mode Select C—**MODC is an active-low Schmitt-trigger input, internally synchronized to CLKOUT. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into OMR when the $\overline{\text{RESET}}$ signal is deasserted.
$\overline{\text{IRQC}}$			**External Interrupt Request C**—After hardware reset, this signal becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If $\overline{\text{IRQC}}$ is asserted synchronous to CLKOUT, multiple processors can be resynchronized using the WAIT instruction and asserting $\overline{\text{IRQC}}$ to exit the wait state.

Table 2-9 Interrupt and Mode Control (Continued)

Signal Name	Type	State During Reset	Signal Description
MODD	Input	Input	**Mode Select D**—MODD is an active-low Schmitt-trigger input, internally synchronized to CLKOUT. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into OMR when the $\overline{\text{RESET}}$ signal is deasserted.
$\overline{\text{IRQD}}$			**External Interrupt Request D**—After hardware reset, this signal becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If $\overline{\text{IRQD}}$ is asserted synchronous to CLKOUT, multiple processors can be resynchronized using the WAIT instruction and asserting $\overline{\text{IRQD}}$ to exit the wait state.

2.8 HOST INTERFACE (HI08)

The HI08 provides a fast parallel 8-bit port, which can connect directly to the host bus.

The HI08 supports a variety of standard buses and can be directly connected to a number of industry standard microcomputers, microprocessors, DSPs, and DMA hardware.

2.8.1 Host Port Usage Considerations

When reading multiple-bit registers that are written by another asynchronous system, you must synchronize carefully. This problem commonly occurs when two asynchronous systems are connected (as they are in the host port). The considerations for proper operation are discussed in **Table 2-10**.

Table 2-10 Host Port Usage Considerations

Action	Description
Asynchronous read of receive byte registers	When reading the receive byte registers, receive register high (RXH), receive register middle (RXM), or receive register low (RXL), use interrupts or poll the receive register data full (RXDF) flag which indicates that data is available. This assures that the data in the receive byte registers is valid.
Asynchronous write to transmit byte registers	Do not write to the transmit byte registers, transmit register high (TXH), transmit register middle (TXM), or transmit register low (TXL), unless the transmit register data empty (TXDE) bit is set indicating that the transmit byte registers are empty. This guarantees that the transmit byte registers transfer valid data to the host receive (HRX) register.
Asynchronous write to host vector	Change the host vector (HV) register only when the host command bit (HC) is clear. This guarantees that the DSP interrupt control logic receives a stable vector.

2.8.2 Host Port Configuration

The functions of the signals associated with the HI08 vary according to the programmed configuration of the interface as determined by the HI08 Port Control Register (HPCR). Refer to **Section 6—Host Interface (HI08)** for detailed descriptions of this and the other configuration registers used with the HI08.

Host interface signal descriptions for the DSP56309 are listed in **Table 2-11**.

Table 2-11 Host Interface

Signal Name	Type	State During Reset	Signal Description
H0–H7	Input/ Output	Tri-stated	**Host Data**—When the HI08 is programmed to interface a non-multiplexed host bus and the HI function is selected, these signals are lines 0–7 of the data bidirectional, tri-state bus.
HAD0–HAD7	Input/ Output		**Host Address—**When HI08 is programmed to interface a multiplexed host bus and the HI function is selected, these signals are lines 0–7 of the address/data bidirectional, multiplexed, tri-state bus.
PB0–PB7	Input or Output		**Port B 0–7**—When the HI08 is configured as GPIO through the HPCR, these signals are individually programmed as inputs or outputs through the HI08 data direction register (HDDR).
HA0	Input	Input	**Host Address Input 0**—When the HI08 is programmed to interface a non-multiplexed host bus and the HI function is selected, this signal is line 0 of the host address input bus.
$\overline{\text{HAS}}$/HAS	Input		**Host Address Strobe—**When HI08 is programmed to interface a multiplexed host bus and the HI function is selected, this signal is the host address strobe (HAS) Schmitt-trigger input. The polarity of the address strobe is programmable but is configured active-low ($\overline{\text{HAS}}$) following reset.
PB8	Input or Output		**Port B 8**—When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.

Table 2-11 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
HA1	Input	Input	**Host Address Input 1**—When the HI08 is programmed to interface a non-multiplexed host bus and the HI function is selected, this signal is line 1 of the host address (HA1) input bus.
HA8	Input		**Host Address 8**—When HI08 is programmed to interface a multiplexed host bus and the HI function is selected, this signal is line 8 of the host address (HA8) input bus.
PB9	Input or Output		**Port B 9**—When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.
HA2	Input	Input	**Host Address Input 2**—When the HI08 is programmed to interface a non-multiplexed host bus and the HI function is selected, this signal is line 2 of the host address (HA2) input bus.
HA9	Input		**Host Address 9**—When HI08 is programmed to interface a multiplexed host bus and the HI function is selected, this signal is line 9 of the host address (HA9) input bus.
PB10	Input or Output		**Port B 10**—When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.

Table 2-11 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
HRW	Input	Input	**Host Read/Write**—When HI08 is programmed to interface a single-data-strobe host bus and the HI function is selected, this signal is the host read/$\overline{\text{write}}$ (HRW) input.
$\overline{\text{HRD}}$/HRD	Input		**Host Read Data**—When HI08 is programmed to interface a double-data-strobe host bus and the HI function is selected, this signal is the host read data strobe (HRD) Schmitt-trigger input. The polarity of the data strobe is programmable, but is configured as active-low ($\overline{\text{HRD}}$) after reset.
PB11	Input or Output		**Port B 11**—When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.
$\overline{\text{HDS}}$/HDS	Input	Input	**Host Data Strobe—**When HI08 is programmed to interface a single-data-strobe host bus and the HI function is selected, this signal is the host data strobe (HDS) Schmitt-trigger input. The polarity of the data strobe is programmable, but is configured as active-low ($\overline{\text{HDS}}$) following reset.
$\overline{\text{HWR}}$/ HWR	Input		**Host Write Data**—When HI08 is programmed to interface a double-data-strobe host bus and the HI function is selected, this signal is the host write data strobe (HWR) Schmitt-trigger input. The polarity of the data strobe is programmable, but is configured as active-low ($\overline{\text{HWR}}$) following reset.
PB12	Input or Output		**Port B 12**—When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.

Table 2-11 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
HCS	Input	Input	**Host Chip Select—**When HI08 is programmed to interface a non-multiplexed host bus and the HI function is selected, this signal is the host chip select (HCS) input. The polarity of the chip select is programmable, but is configured active-low ($\overline{HCS}$) after reset.
HA10	Input		**Host Address 10**—When HI08 is programmed to interface a multiplexed host bus and the HI function is selected, this signal is line 10 of the host address (HA10) input bus.
PB13	Input or Output		**Port B 13**—When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.

Table 2-11 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{\text{HREQ}}$/ HREQ	Output	Input	**Host Request**—When HI08 is programmed to interface a single host request host bus and the HI function is selected, this signal is the host request (HREQ) output. The polarity of the host request is programmable, but is configured as active-low ($\overline{\text{HREQ}}$) following reset. The host request can be programmed as a driven or open-drain output.
$\overline{\text{HTRQ}}$/ HTRQ	Output		**Transmit Host Request—**When HI08 is programmed to interface a double host request host bus and the HI function is selected, this signal is the transmit host request (HTRQ) output. The polarity of the host request is programmable, but is configured as active-low ($\overline{\text{HTRQ}}$) following reset. The host request can be programmed as a driven or open-drain output.
PB14	Input or Output		**Port B 14**—When the HI08 is programmed to interface a multiplexed host bus and the signal is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.

Table 2-11 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{\text{HACK}}$/ HACK	Input	Input	**Host Acknowledge**—When HI08 is programmed to interface a single host request host bus and the HI function is selected, this signal is the host acknowledge (HACK) Schmitt-trigger input. The polarity of the host acknowledge is programmable, but is configured as active-low ($\overline{\text{HACK}}$) after reset.
$\overline{\text{HRRQ}}$/ HRRQ	Output		**Receive Host Request**—When HI08 is programmed to interface a double host request host bus and the HI function is selected, this signal is the receive host request (HRRQ) output. The polarity of the host request is programmable, but is configured as active-low ($\overline{\text{HRRQ}}$) after reset. The host request can be programmed as a driven or open-drain output.
PB15	Input or Output		**Port B 15**—When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed as an input or output through the HDDR.

2.9 ENHANCED SYNCHRONOUS SERIAL INTERFACE

Two synchronous serial interfaces (ESSI0 and ESSI1) provide a full-duplex serial port for serial communication with a variety of serial devices, including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals which implement the Motorola SPI.

2.9.1 ESSI0

The ESSI0 signal descriptions for the DSP56309 are listed in **Table 2-12**.

Table 2-12 Enhanced Synchronous Serial Interface 0 (ESSI0)

Signal Name	Type	State During Reset	Signal Description
SC00	Input or Output	Input	**Serial Control 0**—The function of SC00 is determined by the selection of either synchronous or asynchronous mode. For asynchronous mode, this signal is used for the receive clock I/O (Schmitt-trigger input). For synchronous mode, this signal is used either for Transmitter 1 output or for Serial I/O Flag 0. This signal is driven by a weak keeper after reset.
PC0			**Port C 0**—The default configuration following reset is GPIO input PC0. When this port is configured as PC0, signal direction is controlled through the Port C direction register (PRR0). The signal can be configured as ESSI signal SC00 through the Port C control register (PCR0).

Table 2-12 Enhanced Synchronous Serial Interface 0 (ESSI0) (Continued)

Signal Name	Type	State During Reset	Signal Description
SC01	Input/ Output	Input	**Serial Control 1**—The function of this signal is determined by the selection of either synchronous or asynchronous mode. For Asynchronous mode, this signal is the receiver frame sync I/O. For synchronous mode, this signal is used either for Transmitter 2 output or for Serial I/O Flag 1. This signal is driven by a weak keeper after reset.
PC1	Input or Output		**Port C 1**—The default configuration following reset is GPIO input PC1. When this port is configured as PC1, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SC01 through PCR0.
SC02	Input/ Output	Input	**Serial Control Signal 2**—SC02 is used for frame sync I/O. SC02 is the frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode. When configured as an output, this signal is the internally generated frame sync signal. When configured as an input, this signal receives an external frame sync signal for the transmitter (and the receiver in synchronous operation). This signal is driven by a weak keeper after reset.
PC2	Input or Output		**Port C 2**—The default configuration following reset is GPIO input PC2. When this port is configured as PC2, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SC02 through PCR0.

Table 2-12 Enhanced Synchronous Serial Interface 0 (ESSI0) (Continued)

Signal Name	Type	State During Reset	Signal Description
SCK0	Input/ Output	Input	**Serial Clock**—SCK0 is a bidirectional Schmitt-trigger input signal providing the serial bit rate clock for the ESSI interface. The SCK0 is a clock input or output used by both the transmitter and receiver in synchronous modes or by the transmitter in asynchronous modes. Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 6 T (i.e., the system clock frequency must be at least three times the external ESSI clock frequency). The ESSI needs at least three DSP phases inside each half of the serial clock. This signal is driven by a weak keeper after reset.
PC3	Input or Output		**Port C 3**—The default configuration following reset is GPIO input PC3. When this port is configured as PC3, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SCK0 through PCR0.
SRD0	Input/ Output	Input	**Serial Receive Data**—SRD0 receives serial data and transfers the data to the ESSI receive shift register. SRD0 is an input when data is being received. This signal is driven by a weak keeper after reset.
PC4	Input or Output		**Port C 4**—The default configuration following reset is GPIO input PC4. When this port is configured as PC4, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SRD0 through PCR0.

Table 2-12 Enhanced Synchronous Serial Interface 0 (ESSI0) (Continued)

Signal Name	Type	State During Reset	Signal Description
STD0	Input/ Output	Input	**Serial Transmit Data**—STD0 is used for transmitting data from the serial Transmit shift register. STD0 is an output when data is being transmitted. This signal is driven by a weak keeper after reset.
PC5	Input or Output		**Port C 5**—The default configuration following reset is GPIO input PC5. When this port is configured as PC5, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal STD0 through PCR0.

2.9.2 ESSI1

The ESSI1 signal descriptions for the DSP56309 are listed in **Table 2-13**.

Table 2-13 Enhanced Synchronous Serial Interface 1 (ESSI1)

Signal Name	Type	State During Reset	Signal Description
SC10	Input or Output	Input	**Serial Control 0**—The function of SC10 is determined by the selection of either synchronous or asynchronous mode. For asynchronous mode, this signal is used for the receive clock I/O (Schmitt-trigger input). For synchronous mode, this signal is used either for Transmitter 1 output or for Serial I/O Flag 0. This signal is driven by a weak keeper after reset.
PD0			**Port D 0**—The default configuration following reset is GPIO input PD0. When this port is configured as PD0, signal direction is controlled through the Port D direction register (PRR1). The signal can be configured as an ESSI signal SC10 through the Port D control register (PCR1).
SC11	Input/ Output	Input	**Serial Control 1**—The function of this signal is determined by the selection of either synchronous or asynchronous mode. For asynchronous mode, this signal is the receiver frame sync I/O. For synchronous mode, this signal is used either for Transmitter 2 output or for Serial I/O Flag 1. This signal is driven by a weak keeper after reset.
PD1	Input or Output		**Port D 1**—The default configuration following reset is GPIO input PD1. When this port is configured as PD1, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SC11 through PCR1.

Table 2-13 Enhanced Synchronous Serial Interface 1 (ESSI1) (Continued)

Signal Name	Type	State During Reset	Signal Description
SC12	Input/ Output	Input	**Serial Control Signal 2**—SC12 is used for frame sync I/O. SC12 is the frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode. When configured as an output, this signal is the internally generated frame sync signal. When configured as an input, this signal receives an external frame sync signal for the transmitter. The receiver receives an external frame sync signal as well when in synchronous operation). This signal is driven by a weak keeper after reset.
PD2	Input or Output		**Port D 2**—The default configuration following reset is GPIO input PD2. When this port is configured as PD2, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SC12 through PCR1.

Table 2-13 Enhanced Synchronous Serial Interface 1 (ESSI1) (Continued)

Signal Name	Type	State During Reset	Signal Description
SCK1	Input/ Output	Input	**Serial Clock**—SCK1 is a bidirectional Schmitt-trigger input signal providing the serial bit rate clock for the ESSI interface. The SCK1 is a clock input or output used by both the transmitter and receiver in synchronous modes or by the transmitter in asynchronous modes. Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 6T (i.e., the system clock frequency must be at least three times the external ESSI clock frequency). The ESSI needs at least three DSP phases inside each half of the serial clock. This signal is driven by a weak keeper after reset.
PD3	Input or Output		**Port D 3**—The default configuration following reset is GPIO input PD3. When this port is configured as PD3, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SCK1 through PCR1.
SRD1	Input/ Output	Input	**Serial Receive Data**—SRD1 receives serial data and transfers the data to the ESSI receive shift register. SRD1 is an input when data is being received. This signal is driven by a weak keeper after reset.
PD4	Input or Output		**Port D 4**—The default configuration following reset is GPIO input PD4. When this port is configured as PD4, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SRD1 through PCR1.

Table 2-13 Enhanced Synchronous Serial Interface 1 (ESSI1) (Continued)

Signal Name	Type	State During Reset	Signal Description
STD1	Input/ Output	Input	**Serial Transmit Data**—STD1 is used for transmitting data from the serial transmit shift register. STD1 is an output when data is being transmitted. This signal is driven by a weak keeper after reset.
PD5	Input or Output		**Port D 5**—The default configuration following reset is GPIO input PD5. When this port is configured as PD5, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal STD1 through PCR1.

2.10 SERIAL COMMUNICATION INTERFACE (SCI)

SCI provides a full duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems. SCI signal descriptions are listed in **Table 2-14**.

Table 2-14 Serial Communication Interface (SCI)

Signal Name	Type	State During Reset	Signal Description
RXD	Input	Input	**Serial Receive Data**—This input receives byte oriented serial data and transfers it to the SCI receive shift register. This signal is driven by a weak keeper after reset.
PE0	Input or Output		**Port E 0**—The default configuration following reset is GPIO input PE0. When this port is configured as PE0, signal direction is controlled through the SCI Port E direction register (PRR). The signal can be configured as an SCI signal RXD through the SCI Port E control register (PCR).
TXD	Output	Input	**Serial Transmit Data**—This signal transmits data from SCI transmit data register. This signal is driven by a weak keeper after reset.
PE1	Input or Output		**Port E 1**—The default configuration following reset is GPIO input PE1. When this port is configured as PE1, signal direction is controlled through the SCI PRR. The signal can be configured as an SCI signal TXD through the SCI PCR.

Table 2-14 Serial Communication Interface (SCI) (Continued)

Signal Name	Type	State During Reset	Signal Description
SCLK	Input/ Output	Input	**Serial Clock**—This is the bidirectional Schmitt-trigger input signal providing the input or output clock used by the transmitter and/or the receiver. This signal is driven by a weak keeper after reset.
PE2	Input or Output		**Port E 2**—The default configuration following reset is GPIO input PE2. When this port is configured as PE2, signal direction is controlled through the SCI PRR. The signal can be configured as an SCI signal SCLK through the SCI PCR.

2.11 TIMERS

Three identical and independent timers are implemented in the DSP56309. Each timer can use internal or external clocking; each timer can interrupt the DSP56309 after a specified number of events (clocks) or can signal an external device after counting a specific number of internal events. Triple timer signal descriptions are listed in **Table 2-15**.

Table 2-15 Triple Timer Signals

Signal Name	Type	State During Reset	Signal Description
TIO0	Input or Output	Input	**Timer 0 Schmitt-Trigger Input/Output**—When timer 0 functions as an external event counter or in measurement mode, TIO0 is used as input. When timer 0 functions in watchdog, timer, or pulse modulation mode, TIO0 is used as output. This signal is driven by a weak keeper after reset. The default mode after reset is GPIO input. This can be changed to output or configured as a timer input/output through the timer 0 control/status register (TCSR0).
TIO1	Input or Output	Input	**Timer 1 Schmitt-Trigger Input/Output**—When Timer 1 functions as an external event counter or in measurement mode, TIO1 is used as input. When timer 1 functions in watchdog, timer, or pulse modulation mode, TIO1 is used as output. This signal is driven by a weak keeper after reset. The default mode after reset is GPIO input. This can be changed to output or configured as a timer input/output through the timer 1 control/status register (TCSR1).

Table 2-15 Triple Timer Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
TIO2	Input or Output	Input	**Timer 2 Schmitt-Trigger Input/Output**—When Timer 2 functions as an external event counter or in measurement mode, TIO2 is used as input. When timer 2 functions in watchdog, timer, or pulse modulation mode, TIO2 is used as output. This signal is driven by a weak keeper after reset. The default mode after reset is GPIO input. This can be changed to output or configured as a timer input/output through the timer 2 control/status register (TCSR2).

2.12 OnCE/JTAG INTERFACE

OnCE/JTAG interface signal descriptions are listed in **Table 2-16**.

Table 2-16 OnCE/JTAG Interface

Signal Name	Type	State During Reset	Signal Description
TCK	Input	Input	**Test Clock**—TCK is a test clock input signal used to synchronize the JTAG test logic. Its pin has a pull-up resistor.
TDI	Input	Input	**Test Data Input**—TDI is a test data serial input signal used for test instructions and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.

Table 2-16 OnCE/JTAG Interface (Continued)

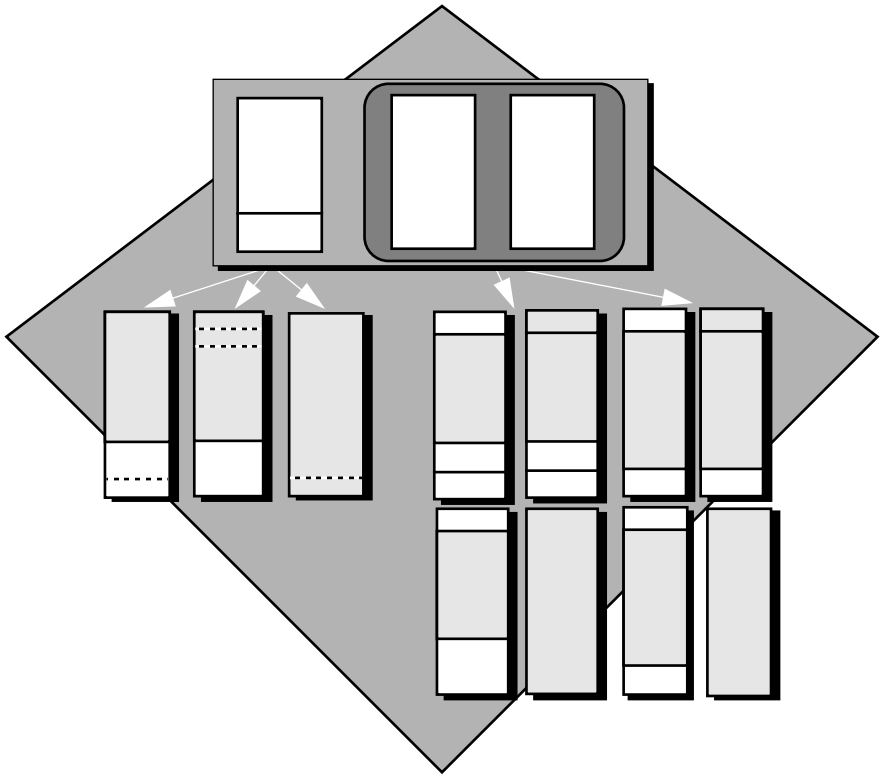
Signal Name	Type	State During Reset	Signal Description
TDO	Output	Tri-stated	**Test Data Output**—TDO is a test data serial output signal used for test instructions and data. TDO is tri-statable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TMS	Input	Input	**Test Mode Select**—TMS is an input signal used to sequence the test controller's state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
$\overline{\text{TRST}}$	Input	Input	**Test Reset**—$\overline{\text{TRST}}$ is an active-low Schmitt-trigger input signal used to asynchronously initialize the test controller. $\overline{\text{TRST}}$ has an internal pull-up resistor. $\overline{\text{TRST}}$ must be asserted after power up.

Table 2-16 OnCE/JTAG Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{DE}$	Input/ Output	Input	**Debug Event**—$\overline{DE}$ is an open-drain, bidirectional, active-low signal providing, as an input, a means of entering debug mode of operation from an external command controller, and as an output, a means of acknowledging that the chip has entered debug mode. This signal, when asserted as an input, causes the DSP56300 core to finish the current instruction being executed, save the instruction pipeline information, enter debug mode, and wait for commands to be entered from the debug serial input line. This signal is asserted as an output for three clock cycles when the chip enters debug mode as a result of a debug request or as a result of meeting a breakpoint condition. The $\overline{DE}$ has an internal pull-up resistor. This is not a standard part of the JTAG TAP controller. The signal connects directly to the OnCE module to initiate debug mode directly or to provide a direct external indication that the chip has entered debug mode. All other interfacing with the OnCE module must occur through the JTAG port.

SECTION 3

MEMORY CONFIGURATION



3.1 MEMORY SPACES

The DSP56309 provides three independent memory spaces:

- Program
- X data
- Y data

Each memory space uses 24-bit addressing by default. The program and data word length is 24 bits. Moreover, this device supports remapping address attribute registers "on the fly," thus allowing access to 16 M of memory.

The DSP56309 provides a sixteen-bit compatibility mode that effectively uses 16-bit addressing for each memory space, allowing access to 64K each of memory. This mode puts zeroes in the most significant byte of the usual (24-bit) program and data word; it ignores the zeroed byte, thus effectively using 16-bit program and data words. The sixteen-bit compatibility mode allows the DSP56309 to use 56000 object code without change, thus minimizing system cost for applications that use the smaller address space. See the DSP56300 Family Manual for further information.

3.1.1 Program Memory Space

Program memory space consists of the following:

- Internal program memory (Program RAM, 20K by default)
- Bootstrap Program ROM (192 x 24-bit)
- (Optionally) off-chip memory expansion (as much as 16 M in 24-bit mode and 64K in 16-bit mode)
- (Optionally) instruction cache (1K) formed from Program RAM

Program memory space at locations $FF00C0 to $FFFFFF is reserved and should not be accessed.

3.1.2 Data Memory Spaces

Data memory space is divided into X data memory and Y data memory to match the natural partitioning of DSP algorithms. The data memory partitioning allows the

DSP56309 to feed two operands to the Data ALU simultaneously, enabling it to perform a multiply-accumulate operation in one clock cycle.

X and Y data memory are identical in structure and functionality except for the upper 128 words of each space. The upper 128 words of X data memory are reserved for internal I/O. We recommend that the programmer reserve the upper 128 words of Y data memory for external I/O. (For further information, see **Section 3.1.2.1 X Data Memory Space** and **Section 3.1.2.2 Y Data Memory Space**.)

X and Y data memory space each consist of the following:

- Internal data memory (X data RAM and Y data RAM, the default size of each is 7K, but they can be switched to 5K each)
- (Optionally) Off-chip memory expansion (up to 16 M in the 24-bit address mode and 64K in the 16-bit address mode)

3.1.2.1 X Data Memory Space

The on-chip peripheral registers and some of the DSP56309 core registers occupy the top 128 locations of X data memory ($FFFF80–$FFFFFF in the 24-bit Address mode or $FF80–$FFFF in the 16-bit Address mode). This area is called X-I/O space, and it can be accessed by MOVE and MOVEP instructions and by bit oriented instructions (BCHG, BCLR, BSET, BTST, BRCLR, BRSET, BSCLR, BSSET, JCLR, JSET, JSCLR, and JSSET). For a listing of the contents of this area, see the programming sheets in **Appendix D—Programming Reference**.

The X memory space at locations $FF0000 to $FFEFFF is reserved and should not be accessed by the programmer.

3.1.2.2 Y Data Memory Space

The off-chip peripheral registers should be mapped into the top 128 locations of Y data memory ($FFFF80–$FFFFFF in the 24-bit address mode or $FF80–$FFFF in the 16-bit address mode) to take advantage of the move peripheral data (MOVEP) instruction and the bit oriented instructions (BCHG, BCLR, BSET, BTST, BRCLR, BRSET, BSCLR, BSSET, JCLR, JSET, JSCLR, and JSSET).

The Y memory space at locations $FF0000 to $FFEFFF is reserved and should not be accessed by the programmer.

3.1.3 Memory Space Configuration

Memory space addressing is 24-bit by default. The DSP56309 switches to sixteen-bit address compatibility mode by setting the sixteen-bit compatibility (SC) bit in the Status Register (SR).

Table 3-1 Memory Space Configuration Bit Settings for the DSP56309

Bit Abbreviation	Bit Name	Bit Location	Cleared = 0 Effect (Default)	Set = 1 Effect
SC	Sixteen-bit Compatibility	SR 13	16M word address space (24-bit address)	64K word address space (16-bit address)

Memory maps for the different configurations are shown in **Figure 3-1** through **Figure 3-8**.

3.2 RAM CONFIGURATION

The DSP56309 contains 34K of RAM, divided by default into the following:

- Program RAM (20K)
- X data RAM (7K)
- Y data RAM (7K)

RAM configuration depends on two bits: the Cache Enable (CE) of the SR and the Memory Select (MS) of the Operating Mode Register (OMR).

Table 3-2 RAM Configuration Bit Settings for the DSP56309

Bit Abbreviation	Bit Name	Bit Location	Cleared = 0 Effect (Default)	Set = 1 Effect
CE	Cache Enable	SR 19	Cache Disabled	Cache Enabled 1K
MS	Memory Switch	OMR 7	Program RAM 20K X data RAM 7K Y data RAM 7K	Program RAM 24K X data RAM 5K Y data RAM 5K

Memory maps for the different configurations are shown in **Figure 3-1** through **Figure 3-8**.

Note: The MS bit cannot be changed when CE is set. The instruction cache occupies the top 1K of what would otherwise be Program RAM; if you switch memory into or out of Program RAM when the cache is enabled, the switch causes conflicts. To change the MS bit when CE is set, do the following:

1. Clear CE.
2. Change MS.
3. Set CE.

3.2.1 On-Chip Program Memory (Program RAM)

The on-chip Program RAM consists of 24-bit wide, high-speed, internal Static RAM occupying the lowest 20K (default), 23K, 24K, or 19K locations in the program memory space (depending on the settings of the MS and CE bits). The Program RAM default organization is 80 banks of 256 24-bit words (20K). The upper eight banks of both X data RAM and Y data RAM can be configured as Program RAM by setting the MS bit. When the CE is set, the upper 1K of Program RAM is used as an internal Instruction Cache.

CAUTION

While the contents of Program RAM are unaffected by toggling the MS bit, the location of program data placed in the Program RAM/Instruction Cache area changes after the MS bit is toggled, since the cache always occupies the top-most 1K Program RAM addresses. To preserve program data integrity, do not set or clear the MS bit when the CE bit is set. See Section 3.2 on page 3-5 for the correct procedure.

3.2.2 On-Chip X Data Memory (X Data RAM)

The on-chip X data RAM consists of 24-bit wide, high-speed, internal Static RAM occupying the lowest 7K (default) or 5K locations in the X memory space. The size of the X data RAM depends on the setting of the MS bit (default: MS is cleared). The X data RAM default organization is 28 banks of 256 (7K) 24-bit words. Eight banks of RAM can be switched from the X data RAM to the Program RAM by setting the MS bit (leaving 5K of X data RAM).

3.2.3 On-Chip Y Data Memory (Y Data RAM)

The on-chip Y data RAM consists of 24-bit wide, high-speed, internal Static RAM occupying the lowest 7K (default) or 5K locations in the Y memory space. The size of the Y data RAM is dependent on the setting of the MS bit (default: MS is cleared). The Y data RAM default organization is 28 banks of 256 (7K) 24-bit words. Eight banks of RAM can be switched from the Y data RAM to the Program RAM by setting the MS bit (leaving 5K of Y data RAM).

3.2.4 Bootstrap ROM

The bootstrap code is accessed at addresses $FF0000 to $FFF0BF (192 words) in program memory space. The bootstrap ROM cannot be accessed in 16-bit address compatibility mode. See **Appendix A—Bootstrap Programs** for a complete listing of the bootstrap code.

3.3 MEMORY CONFIGURATIONS

Memory configuration determines the size and address range for addressable memory, as well as the amount of memory allocated to Program RAM, data RAM, and the instruction cache.

3.3.1 Memory Space Configurations

The memory space configurations are listed in **Table 3-3**.

Table 3-3 Memory Space Configurations for the DSP56309

SC Bit Setting	Addressable Memory Size	Address Range	Number of Address Bits
0	16M words	$000000–$FFFFFF	24
1	64K words	$0000–$FFFF	16

3.3.2 RAM Configurations

The RAM configurations for the DSP56309 appear in **Table 3-4**.

Table 3-4 RAM Configurations for the DSP56309

Bit Settings		Memory Sizes (in K)			
MS	CE	Program RAM	X data RAM	Y data RAM	Cache
0	0	20	7	7	0
0	1	19	7	7	1
1	0	24	5	5	0
1	1	23	5	5	1

The actual memory locations for Program RAM and the instruction cache in the Program memory space are determined by the MS and CE bits. Their addresses appear in **Table 3-5**.

Table 3-5 Memory Locations for Program RAM and Instruction Cache

MS	CE	Program RAM Location	Cache Location
0	0	$0000–$4FFF	N/A
0	1	$0000–$4BFF	$4C00–$4FFF
1	0	$0000–$5FFF	N/A
1	1	$0000–$5BFF	$5C00–$5FFF

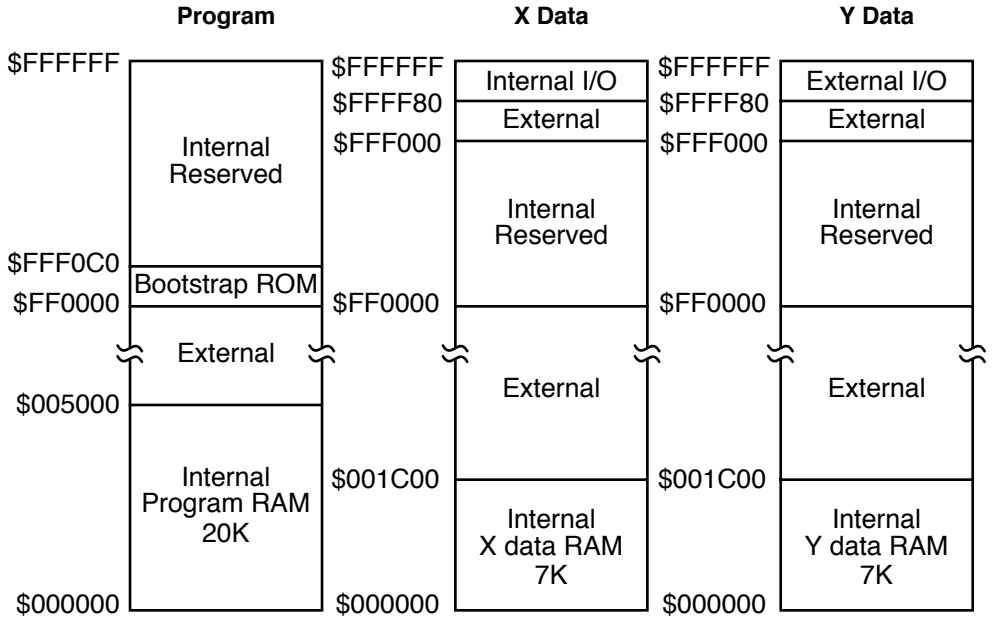
The actual memory locations for both X and Y data RAM in their own memory space are determined by the MS bit. Their addresses appear in **Table 3-6**.

Table 3-6 Memory Locations for Data RAM

MS	Data RAM Location
0	$0000–$1BFF
1	$0000–$13FF

3.4 MEMORY MAPS

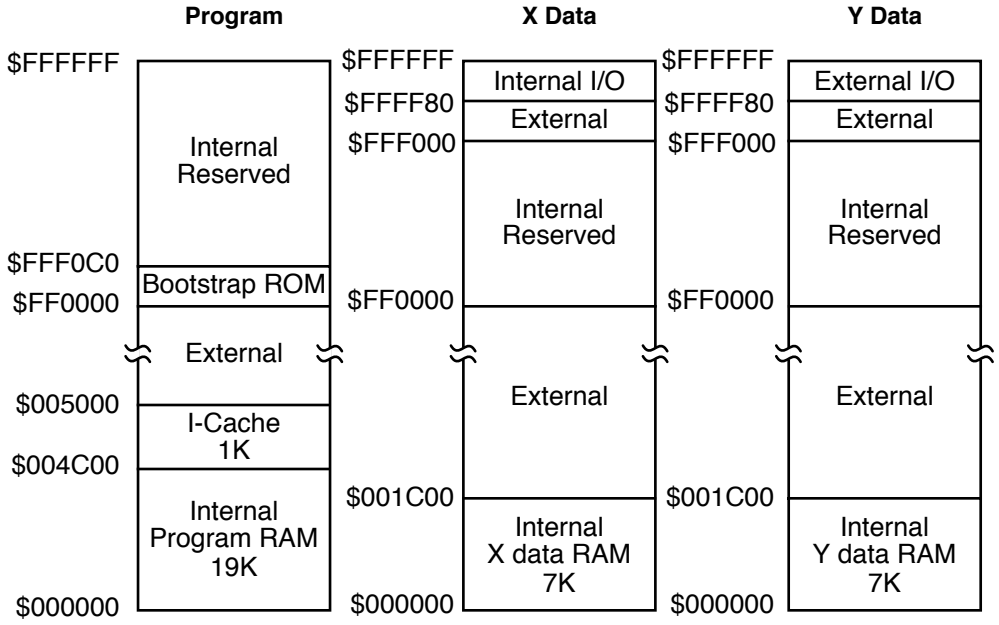
Figure 3-1 through **Figure 3-8** illustrate each of the memory space and RAM configurations defined by the settings of the SC, MS, and CE bits. The figures show the configuration, and the accompanying tables show the bit settings, memory sizes, and memory locations.



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	0	20K $0000–$4FFF	7K $0000–$1BFF	7K $0000–$1BFF	None	16M

AA0557

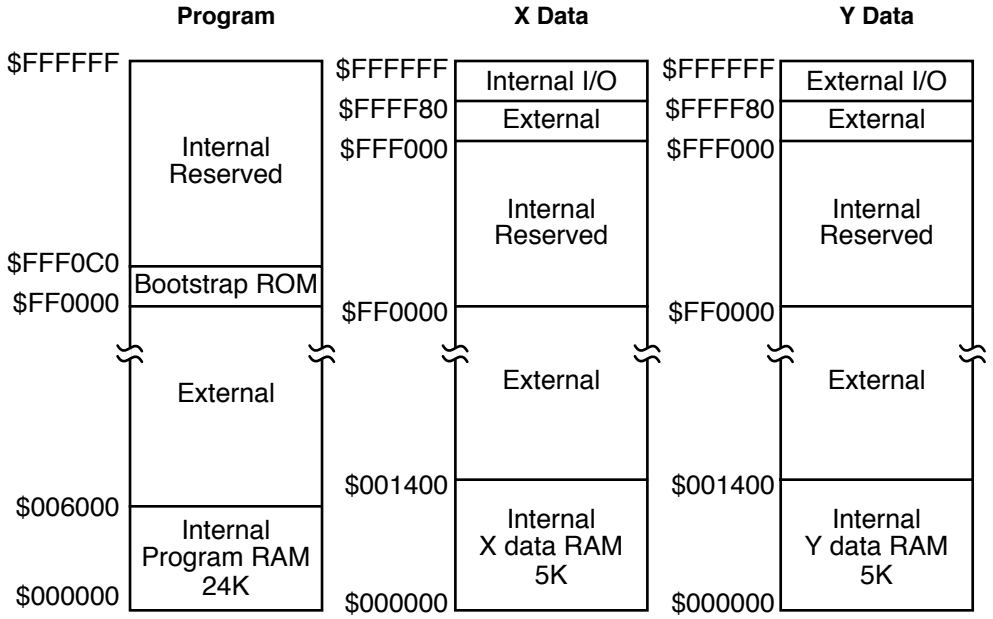
Figure 3-1 Default Settings (0, 0, 0)



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	1	19K $0000–$4BFF	7K $0000–$1BFF	7K $0000–$1BFF	1K $4C00–$4FFF	16 M

AA0561

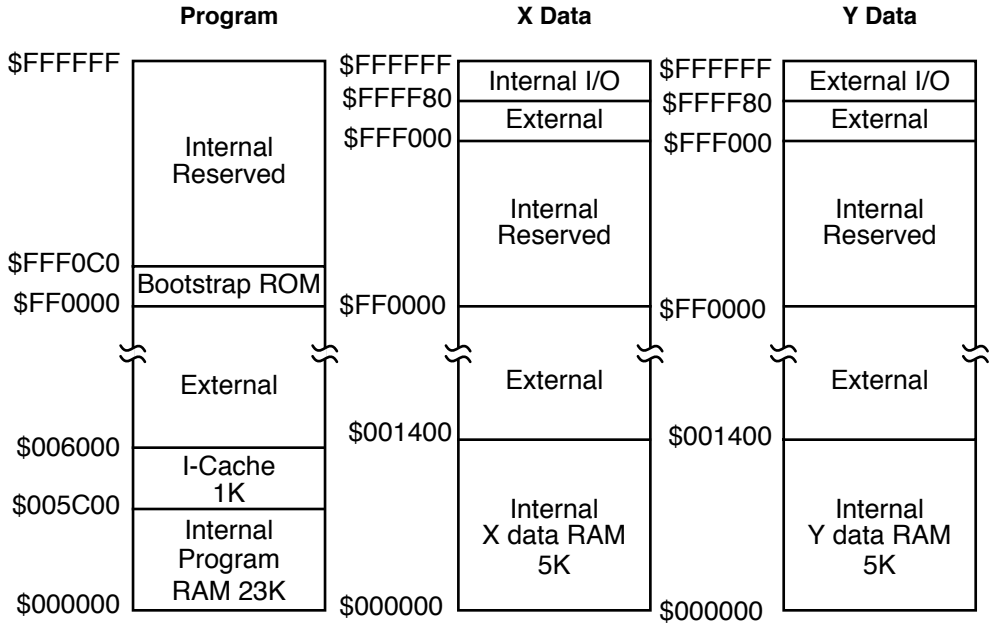
Figure 3-2 Instruction Cache Enabled (0, 0, 1)



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	0	24K $0000–$5FFF	5K $0000–$13FF	5K $0000–$13FF	None	16 M

AA0559

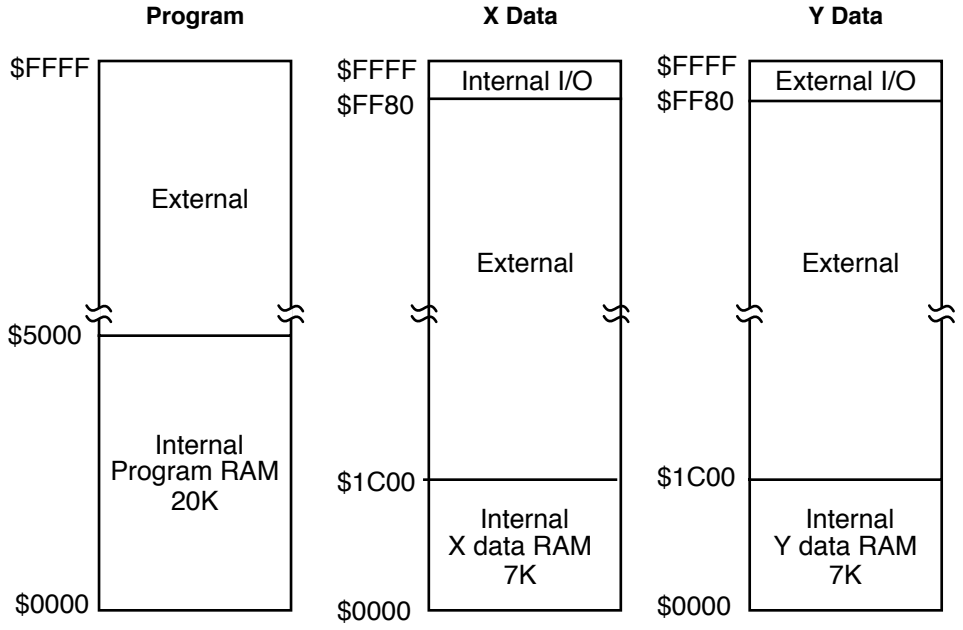
Figure 3-3 Switched Program RAM (0, 1, 0)



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	1	23K $0000–$5BFF	5K $0000–$13FF	5K $0000–$13FF	1K $5C00–$5FFF	16 M

AA0563

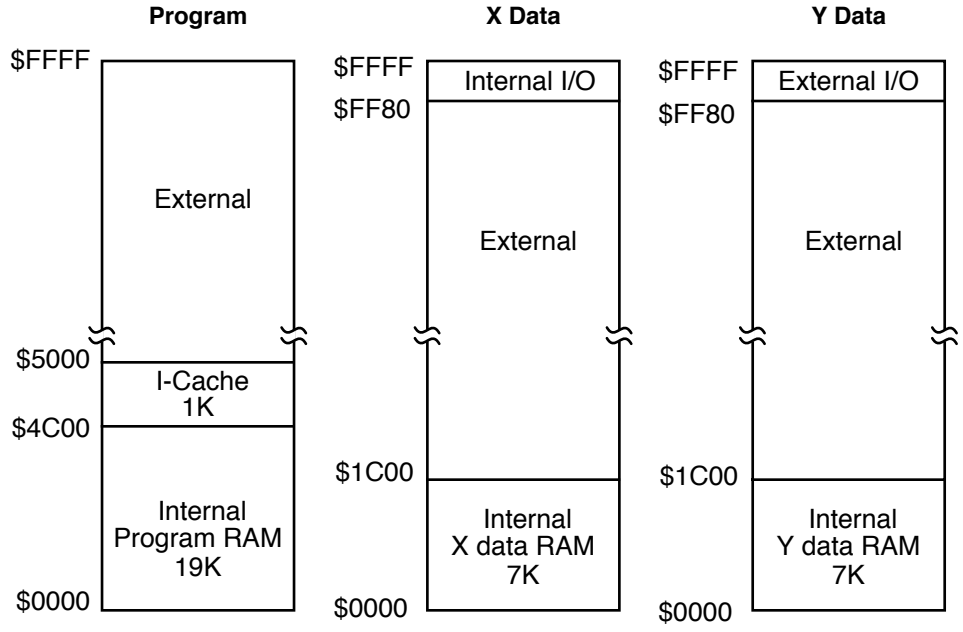
Figure 3-4 Switched Program RAM and Instruction Cache Enabled (0, 1, 1)



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	0	20K $0000–$4FFF	7K $0000–$1BFF	7K $0000–$1BFF	None	64K

AA0558

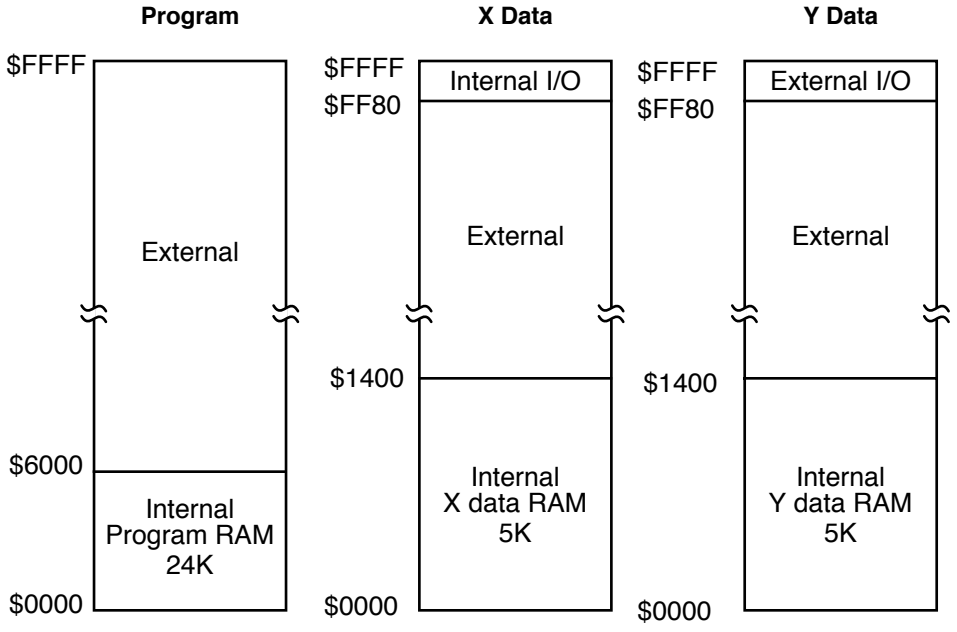
Figure 3-5 16-bit Space with Default RAM (1, 0, 0)



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	1	19K $0000–$4BFF	7K $0000–$1BFF	7K $0000–$1BFF	1K $4C00–$4FFF	64K

AA0562

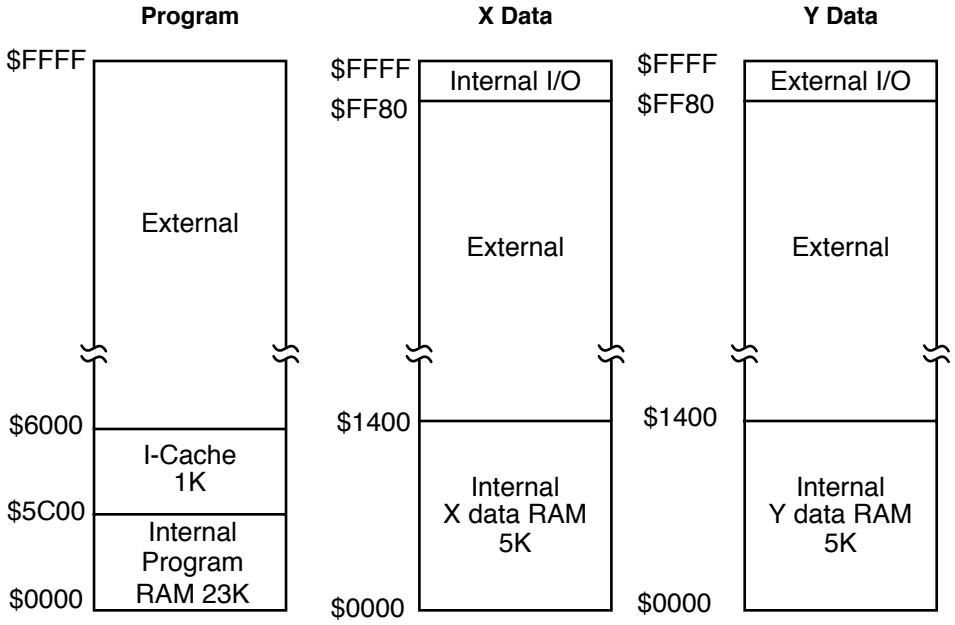
Figure 3-6 16-bit Space with Instruction Cache Enabled (1, 0, 1)



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	0	24K $0000–$5FFF	5K $0000–$13FF	5K $0000–$13FF	None	64K

AA0560

Figure 3-7 16-bit Space with Switched Program RAM (1, 1, 0)



Bit Settings			Memory Configuration				
SC	MS	CE	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	1	23K $0000–$5FFF	5K $0000–$13FF	5K $0000–$13FF	1K $5C00–$5FFF	64K

AA0564

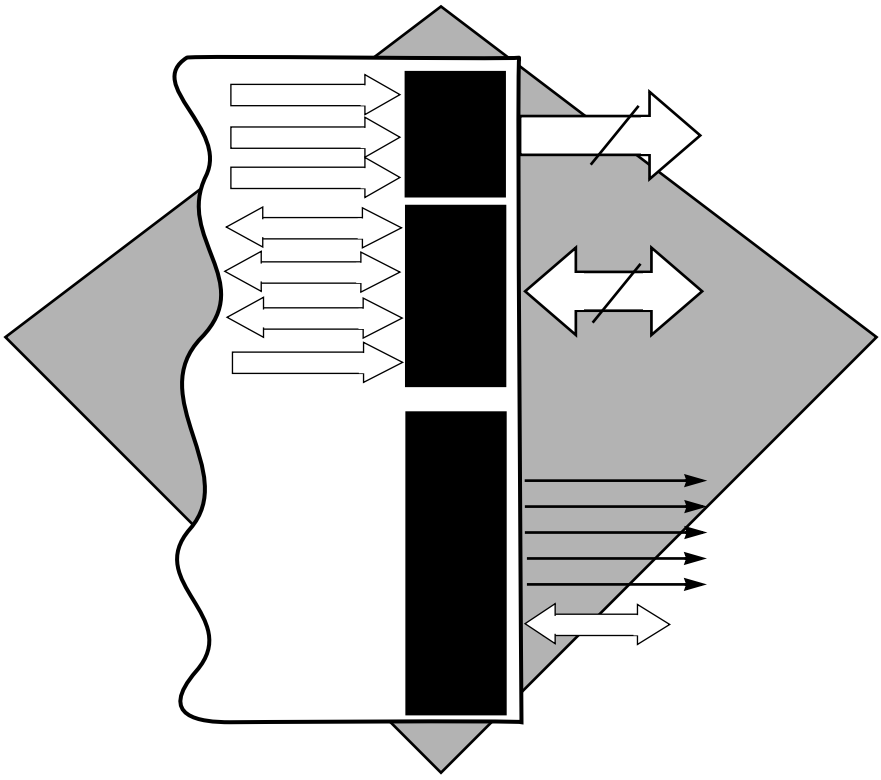
Figure 3-8 16-bit Space, Switched Program RAM, Instruction Cache Enabled (1, 1, 1)

3.5 INTERNAL I/O MEMORY MAP

The DSP56309 internal X-I/O space (the top 128 locations of the X data memory space) is listed in **Table D-2** on page D-11 of **Appendix D—Interrupt Sources**.

SECTION 4

CORE CONFIGURATION



4.1 INTRODUCTION

This chapter presents details on core configuration specific to the DSP56309. These configuration details include the following:

- Operating modes
- Bootstrap program
- Interrupt sources and priorities
- DMA request sources
- Operating Mode Register
- PLL control register
- AA control registers
- JTAG Boundary Scan Register

For information on specific registers or modules in the DSP56300 core, refer to the DSP56300 Family Manual (DSP56300FM/AD).

4.2 OPERATING MODES

The DSP56309 begins operation by leaving Reset state and going into one of eight operating modes. As the DSP56309 exits the Reset state, it loads the values of MODA, MODB, MODC, and MODD into bits MA, MB, MC, and MD of the Operating Mode Register (OMR). These bit settings select the operating mode, which determines the bootstrap program option the microprocessor uses to start up.

The MA–MD bits of the OMR can also be set directly by software. A jump directly to the bootstrap program entry point ($FF0000) after the OMR bits are set causes the DSP56309 to execute the specified bootstrap program option (except modes 0 and 8).

Table 4-1 shows the DSP56309 bootstrap operation modes, the corresponding settings of the external operational mode signal lines (the mode bits MA–MD in the OMR), and the reset vector address to which the DSP56309 jumps once it leaves the Reset state.

4.3 BOOTSTRAP PROGRAM

Bootstrap operating mode descriptions for the DSP56309 are listed in **Table 4-1**.

Table 4-1 DSP56309 Operating Modes

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
0	0	0	0	0	$C00000	Expanded mode: address $C00000 is reflected as $00000 on Port A signals A0-A17
1	0	0	0	1	$FF0000	Reserved
2	0	0	1	0	$FF0000	Reserved
3	0	0	1	1	$FF0000	Reserved
4	0	1	0	0	$FF0000	Reserved
5	0	1	0	1	$FF0000	Reserved
6	0	1	1	0	$FF0000	Reserved
7	0	1	1	1	$FF0000	Reserved
8	1	0	0	0	$008000	Expanded mode
9	1	0	0	1	$FF0000	Boot from byte-wide memory at $D00000
A	1	0	1	0	$FF0000	Boot through SCI
B	1	0	1	1	$FF0000	Reserved
C	1	1	0	0	$FF0000	HI08 boot in ISA mode
D	1	1	0	1	$FF0000	HI08 boot in HC11 non-multiplexed mode
E	1	1	1	0	$FF0000	HI08 boot in 8051 multiplexed bus mode
F	1	1	1	1	$FF0000	HI08 boot in MC68302 mode

The bootstrap program is factory-programmed in an internal, 192-word by 24-bit bootstrap ROM located in program memory space at locations $FF0000–$FF00BF. The bootstrap program can load any Program RAM segment from an external byte-wide EPROM, the SCI, or the host port. The bootstrap program code is listed in **Appendix A—Bootstrap Programs**.

On exiting the reset state, the DSP56309 does the following:

1. Samples the MODA, MODB, MODC, and MODD signal lines.
2. Loads their values into bits MA, MB, MC, and MD in the OMR.

The contents of the MA, MB, MC, and MD bits determine which bootstrap mode the DSP56309 enters:

1. If MA, MB, MC, and MD are all cleared (Bootstrap mode 0), the program bypasses the bootstrap ROM, and the DSP56309 starts loading instructions from external program memory location $C00000.
2. If MA, MB, and MC are cleared and MD is set (Bootstrap mode 8), the program bypasses the bootstrap ROM, and the DSP56309 starts loading in instruction values from external program memory location $008000.
3. Otherwise (Bootstrap modes 1–7), the DSP56309 jumps to the bootstrap program entry point at $FF0000.

If the bootstrap program is loading via the host interface (HI08), setting the HF0 bit in the host status register (HSR) causes the DSP56309 to stop loading and begin executing the loaded program at the specified start address.

See **Table 4-1** for a tabular description of the mode bit settings for the operating modes.

The bootstrap program options (except modes 0 and 8) can be invoked at any time by setting the MA, MB, MC, and MD bits in the OMR and jumping to the bootstrap program entry point, $FF0000. Software can directly set the mode selection bits in the OMR.

Bootstrap modes 0 and 8 are the normal functioning modes for the DSP56309. Bootstrap modes 1–7 are the bootstrap modes proper.

Bootstrap modes 9, A, C, D, E, F select different, specific devices for loading the bootstrap source. In those bootstrap modes, the bootstrap program expects the following data sequence when downloading the user program through an external port:

1. Three bytes defining the number of (24-bit) program words to be loaded
2. Three bytes defining the (24-bit) start address to which the user program loads in the DSP56309 program memory
3. The user program (three bytes for each 24-bit program word)

The three bytes for each data sequence must be loaded with the least significant byte first. Once the bootstrap program completes loading the specified number of words, it jumps to the specified starting address and executes the loaded program.

4.3.1 Mode 0: Expanded Mode

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
0	0	0	0	0	$C00000	Expanded mode

The bootstrap ROM is bypassed and the DSP56309 starts fetching instructions beginning at address $C00000. Memory accesses are performed using SRAM memory access type with 31 wait states and no address attributes selected (by default).

4.3.2 Modes 1 to 7: Reserved

These modes are reserved for future use.

4.3.3 Mode 8: Expanded Mode

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
8	1	0	0	0	$008000	Expanded mode

The bootstrap ROM is bypassed and the DSP56309 starts fetching instructions beginning at address $008000. Memory accesses are performed using SRAM memory access type with 31 wait states and no address attributes selected.

4.3.4 Mode 9: Boot from Byte-Wide External Memory

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
9	1	0	0	1	$FF0000	Boot from byte-wide memory (at $D00000)

The bootstrap program loads instructions through Port A from external byte-wide memory, starting at P:$D00000. The SRAM memory access type is selected by the values in address attribute register 1 (AAR1). Thirty-one wait states are inserted between each memory access. Address $D00000 is reflected as address $00000 on Port A signals HA0-HA17.

4.3.5 Mode A: Boot from SCI

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
A	1	0	1	0	$FF0000	Boot through SCI

Instructions are loaded through the SCI. The bootstrap program sets the SCI to operate in 10-bit asynchronous mode, with one start bit, eight data bits, one stop bit and no parity. Data is received in this order; start bit, eight data bits (LSB first), and one stop bit. Data is aligned in the SCI receive data register with the LSB of the least significant byte of the received data appearing at bit 0. The user must provide an external clock source with a frequency at least 16 times the transmission data rate. Each byte received by the SCI is echoed back through the SCI transmitter to the external transmitter.

4.3.6 Mode B: Reserved

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
B	1	0	1	1	$FF0000	Reserved

This mode is reserved for future use.

4.3.7 Modes C, D, E, F: Boot from HI08

Modes C, D, E, and F enable the programmer to boot through the host interface (HI08) in various ways.

4.3.7.1 Mode C: In ISA/DSP5630X Mode (8-Bit Bus)

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
C	1	1	0	0	$FF0000	HI08 Bootstrap in ISA/DSP5630X

In mode C: boot from HI08 in ISA/DSP5630x with an 8-bit wide bus, the HI08 is configured to interface with an ISA bus or with the memory expansion port of a master DSP5630*n* processor.

If the host processor sets host flag 0 (HF0) in the HI08 interface control register (HCR) while writing the initialization program, the bootstrap program stops loading instructions, jumps to the starting address specified, and executes the loaded program.

4.3.7.2 Mode D: In HC11 Non-multiplexed Mode

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
D	1	1	0	1	$FF0000	HI08 Bootstrap in HC11 non-multiplexed

In mode D: boot from HI08 in HC11 non-multiplexed mode, the bootstrap program sets the host interface to interface with the Motorola HC11 microcontroller.

If the host processor sets host flag 0 (HF0) in the HCR while writing the initialization program, the bootstrap program stops loading instructions, jumps to the starting address specified, and executes the loaded program.

4.3.7.3 Mode E: In 8051 Multiplexed Bus Mode

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
E	1	1	1	0	$FF0000	HI08 Bootstrap in 8051 multiplexed bus

In mode E: boot from HI08 in 8051 multiplexed bus mode, the bootstrap program sets the host interface to interface with the Intel® 8051 bus.

If the host processor sets host flag 0 (HF0) in the HCR while writing the initialization program, the bootstrap program stops loading instructions, jumps to the starting address specified, and executes the loaded program.

4.3.7.4 Mode F: In 68302/68360 Bus Mode

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
F	1	1	1	1	$FF0000	HI08 Bootstrap in 68302 bus

In mode F: boot from HI08 in 68302/68360 Bus mode, the bootstrap program sets the host interface to interface with the Motorola 68302 or 68360 bus.

If the host processor sets host flag 0 (HF0) in the HCR while writing the initialization program, the bootstrap program stops loading instructions, jumps to the starting address specified, and executes the loaded program.

4.4 INTERRUPT SOURCES AND PRIORITIES

DSP56309 interrupt handling, like that of all DSP56300 family members, has been optimized for DSP applications. Refer to Section 7 of the DSP56300 Family Manual. The interrupt table is located in the 256 locations of program memory to which the vector base address (VBA) register in the program control unit (PCU) points.

4.4.1 Interrupt Sources

Each interrupt is allocated two instructions in the table, so there are 128 table entries for interrupt handling. **Table 4-2** shows the table entry address for each interrupt source.

The DSP56309 initialization program loads the table entry for each interrupt serviced with two interrupt servicing instructions.

In the DSP56309, only 46 of the 128 vector addresses are used for specific interrupt sources. The remaining 82 are reserved. If you know that certain interrupts will not be used, those interrupt vector locations can be used for program or data storage.

Table 4-2 Interrupt Sources

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$00	3	Hardware $\overline{\text{RESET}}$
VBA:$02	3	Stack Error
VBA:$04	3	Illegal Instruction
VBA:$06	3	Debug Request Interrupt
VBA:$08	3	Trap
VBA:$0A	3	Non-Maskable Interrupt ($\overline{\text{NMI}}$)
VBA:$0C	3	Reserved
VBA:$0E	3	Reserved
VBA:$10	0–2	$\overline{\text{IRQA}}$
VBA:$12	0–2	$\overline{\text{IRQB}}$
VBA:$14	0–2	$\overline{\text{IRQC}}$
VBA:$16	0–2	$\overline{\text{IRQD}}$
VBA:$18	0–2	DMA Channel 0
VBA:$1A	0–2	DMA Channel 1
VBA:$1C	0–2	DMA Channel 2
VBA:$1E	0–2	DMA Channel 3
VBA:$20	0–2	DMA Channel 4
VBA:$22	0–2	DMA Channel 5
VBA:$24	0–2	TIMER 0 Compare
VBA:$26	0–2	TIMER 0 Overflow
VBA:$28	0–2	TIMER 1 Compare
VBA:$2A	0–2	TIMER 1 Overflow
VBA:$2C	0–2	TIMER 2 Compare
VBA:$2E	0–2	TIMER 2 Overflow

Table 4-2 Interrupt Sources (Continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$30	0–2	ESSI0 Receive Data
VBA:$32	0–2	ESSI0 Receive Data With Exception Status
VBA:$34	0–2	ESSI0 Receive Last Slot
VBA:$36	0–2	ESSI0 Transmit Data
VBA:$38	0–2	ESSI0 Transmit Data With Exception Status
VBA:$3A	0–2	ESSI0 Transmit Last Slot
VBA:$3C	0–2	Reserved
VBA:$3E	0–2	Reserved
VBA:$40	0–2	ESSI1 Receive Data
VBA:$42	0–2	ESSI1 Receive Data With Exception Status
VBA:$44	0–2	ESSI1 Receive Last Slot
VBA:$46	0–2	ESSI1 Transmit Data
VBA:$48	0–2	ESSI1 Transmit Data With Exception Status
VBA:$4A	0–2	ESSI1 Transmit Last Slot
VBA:$4C	0–2	Reserved
VBA:$4E	0–2	Reserved
VBA:$50	0–2	SCI Receive Data
VBA:$52	0–2	SCI Receive Data With Exception Status
VBA:$54	0–2	SCI Transmit Data
VBA:$56	0–2	SCI Idle Line
VBA:$58	0–2	SCI Timer
VBA:$5A	0–2	Reserved
VBA:$5C	0–2	Reserved
VBA:$5E	0–2	Reserved
VBA:$60	0–2	Host Receive Data Full
VBA:$62	0–2	Host Transmit Data Empty
VBA:$64	0–2	Host Command (Default)
VBA:$66	0–2	Reserved
:	:	:
VBA:$FE	0–2	Reserved

4.4.2 Interrupt Priority Levels

The DSP56309 has a four-level interrupt priority structure. Each interrupt has two interrupt priority level bits (IPL[1:0]) that determine its interrupt priority level. Level 0 is the lowest priority; Level 3 is the highest-level priority and is non-maskable. **Table 4-3** defines the IPL bits.

Table 4-3 Interrupt Priority Level Bits

IPL bits		Interrupts Enabled	Interrupts Masked	Interrupt Priority Level
xxL1	xxL0			
0	0	No	—	0
0	1	Yes	0	1
1	0	Yes	0, 1	2
1	1	Yes	0, 1, 2	3

There are two interrupt priority registers in the DSP56309. The IPR–C is dedicated to DSP56300 core interrupt sources, and IPR–P is dedicated to DSP56309 peripheral interrupt sources. IPR–C is shown in **Figure 4-1** and IPR–P is shown in **Figure 4-2**.

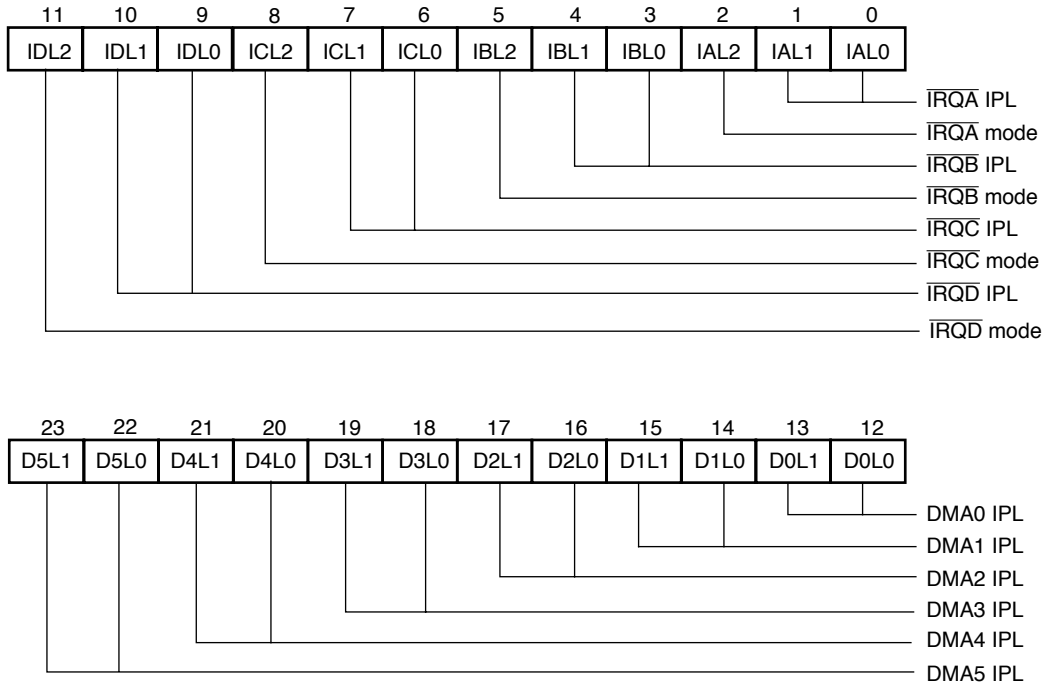


Figure 4-1 Interrupt Priority Register C (IPR-C) (X:$FFFFFF)

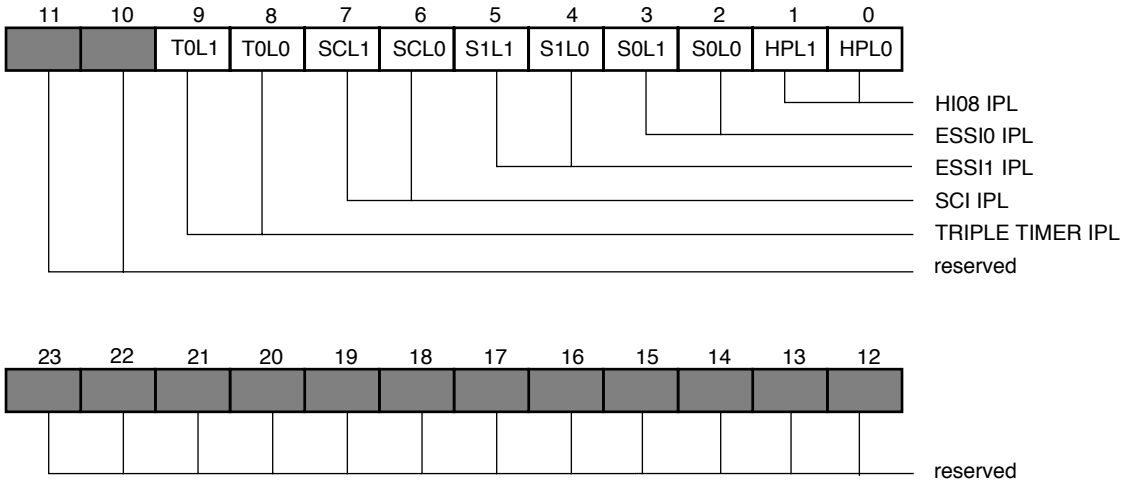


Figure 4-2 Interrupt Priority Register P (IPR-P) (X:$FFFFFE)

4.4.3 Interrupt Source Priorities Within an IPL

If more than one interrupt request is pending when an instruction executes, the interrupt source with the highest IPL is serviced first. When several interrupt requests having the same IPL are pending, another fixed-priority structure within that IPL determines which interrupt source is serviced first. This fixed priority list of interrupt sources within an IPL is shown in **Table 4-4**.

Table 4-4 Interrupt Source Priorities within an IPL

Priority	Interrupt Source
Level 3 (Nonmaskable)	
Highest	Hardware $\overline{\text{RESET}}$
—	Stack Error
—	Illegal Instruction
—	Debug Request Interrupt
—	Trap
Lowest	Non-Maskable Interrupt
Levels 0, 1, 2 (Maskable)	
Highest	$\overline{\text{IRQA}}$ (External Interrupt)
—	$\overline{\text{IRQB}}$ (External Interrupt)
—	$\overline{\text{IRQC}}$ (External Interrupt)
—	$\overline{\text{IRQD}}$ (External Interrupt)
—	DMA Channel 0 Interrupt
—	DMA Channel 1 Interrupt
—	DMA Channel 2 Interrupt
—	DMA Channel 3 Interrupt
—	DMA Channel 4 Interrupt
—	DMA Channel 5 Interrupt

Table 4-4 Interrupt Source Priorities within an IPL (Continued)

Priority	Interrupt Source
—	Host Command Interrupt
—	Host Transmit Data Empty
—	Host Receive Data Full
—	ESSI0 RX Data with Exception Interrupt
—	ESSI0 RX Data Interrupt
—	ESSI0 Receive Last Slot Interrupt
—	ESSI0 TX Data With Exception Interrupt
—	ESSI0 Transmit Last Slot Interrupt
—	ESSI0 TX Data Interrupt
—	ESSI1 RX Data With Exception Interrupt
—	ESSI1 RX Data Interrupt
—	ESSI1 Receive Last Slot Interrupt
—	ESSI1 TX Data With Exception Interrupt
—	ESSI1 Transmit Last Slot Interrupt
—	ESSI1 TX Data Interrupt
—	SCI Receive Data With Exception Interrupt
—	SCI Receive Data
—	SCI Transmit Data
—	SCI Idle Line
—	SCI Timer
—	TIMER0 Overflow Interrupt
—	TIMER0 Compare Interrupt
—	TIMER1 Overflow Interrupt
—	TIMER1 Compare Interrupt

Table 4-4 Interrupt Source Priorities within an IPL (Continued)

Priority	Interrupt Source
—	TIMER2 Overflow Interrupt
Lowest	TIMER2 Compare Interrupt

4.5 DMA REQUEST SOURCES

The DMA request source bits (DRS[4:0]) in the DMA control/status registers) encode the source of DMA requests used to trigger DMA transfers. The DMA request sources can be internal peripherals or external devices requesting service through the $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$, or $\overline{\text{IRQD}}$ signals. **Table 4-5** describes the meanings of the DRS bits.

Table 4-5 DMA Request Sources

DMA Request Source Bits DRS4... DRS0	Requesting Device
00000	External ($\overline{\text{IRQA}}$ signal)
00001	External ($\overline{\text{IRQB}}$ signal)
00010	External ($\overline{\text{IRQC}}$ signal)
00011	External ($\overline{\text{IRQD}}$ signal)
00100	Transfer done from DMA channel 0
00101	Transfer done from DMA channel 1
00110	Transfer done from DMA channel 2
00111	Transfer done from DMA channel 3
01000	Transfer done from DMA channel 4
01001	Transfer done from DMA channel 5
01010	ESSI0 Receive Data (RDF0 = 1)
01011	ESSI0 Transmit Data (TDE0 = 1)
01100	ESSI1 Receive Data (RDF1 = 1)

Table 4-5 DMA Request Sources (Continued)

DMA Request Source Bits DRS4... DRS0	Requesting Device
01101	ESSI1 Transmit Data (TDE1 = 1)
01110	SCI Receive Data (RDRF = 1)
01111	SCI Transmit Data (TDRE = 1)
10000	Timer0 (TCF0 = 1)
10001	Timer1 (TCF1 = 1)
10010	Timer2 (TCF2 = 1)
10011	Host Receive Data Full (HRDF = 1)
10100	Host Transmit Data Empty (HTDE = 1)
10101–11111	Reserved

4.6 OPERATING MODE REGISTER (OMR)

The OMR is a 24-bit, read/write register divided into three byte-sized units. The first two bytes (COM and EOM) control the chip's operating mode. The third byte (SCS) controls and monitors the stack extension. The OMR control bits are shown in **Figure 4-3**. Refer to the DSP56300 Family Manual for a complete description of the OMR.

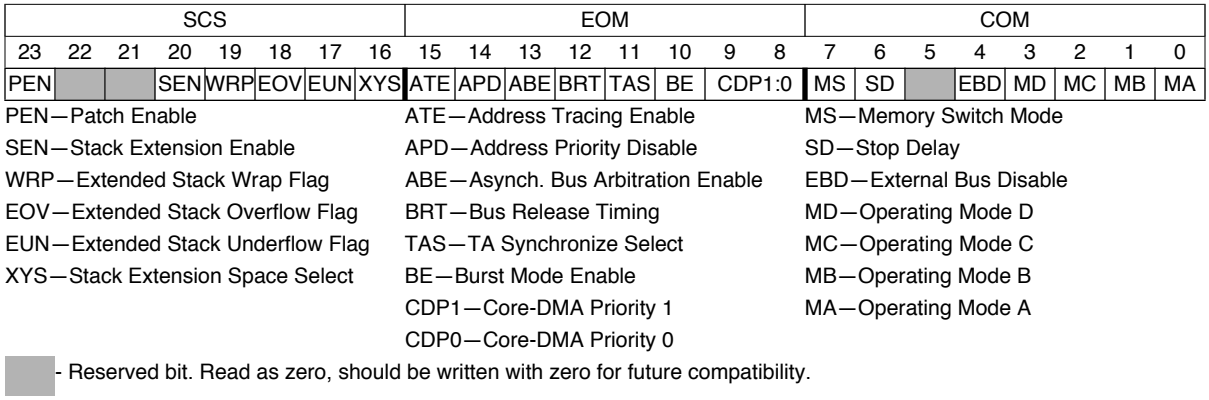


Figure 4-3 DSP56309 Operating Mode Register (OMR)

4.7 PLL CONTROL REGISTER

The PLL Control (PCTL) register is an X-I/O mapped, 24-bit, read/write register that directs the operation of the on-chip PLL. The PCTL control bits are shown in **Figure 4-4**. Refer to the DSP56300 Family Manual for a full description of the PCTL.

11	10	9	8	7	6	5	4	3	2	1	0
MF11	MF10	MF9	MF8	MF7	MF6	MF5	MF4	MF3	MF2	MF1	MF0
23	22	21	20	19	18	17	16	15	14	13	12
PD3	PD2	PD1	PD0	COD	PEN	PSTP	XTLD	XTLR	DF2	DF1	DF0

AA0852

Figure 4-4 PLL Control (PCTL) Register

4.7.1 PCTL PLL Multiplication Factor Bits 0–11

The multiplication factor bits (MF[11:0]) define the Multiplication Factor (MF) that is applied to the PLL input frequency. The MF bits are cleared during a DSP56309 hardware reset, which corresponds to an MF of one.

4.7.2 PCTL XTAL Disable Bit (XTLD) Bit 16

The XTAL disable bit (XTLD) controls the on-chip crystal oscillator XTAL output. The XTLD bit is cleared during a DSP56309 hardware reset, which means that the XTAL output signal is active, permitting normal operation of the crystal oscillator.

4.7.3 PCTL Predivider Factor Bits (PD0–PD3) Bits 20–23

The predivider factor bits (PD0–PD3) define the predivision factor (PDF) to be applied to the PLL input frequency. The PD0–PD3 bits are cleared during a DSP56309 hardware reset, which corresponds to a PDF of one.

4.8 DEVICE IDENTIFICATION REGISTER (IDR)

The device identification register (IDR) is a 24-bit, read-only factory programmed register that identifies DSP56300 family members. It specifies the derivative number and

revision number of the device. This information can be used in testing or by software. shows the contents of the IDR.

Revision numbers are assigned as follows: $0 is revision 0, $1 is revision A, and so on. Because the DSP56309 is based on the DSP56302, its identification number is based on the derivative number and revision number of that device.

23 16	15 12	11 0
Reserved	Revision Number	Derivative Number
$00	$2	$302

Figure 4-5 Identification Register Configuration (Revision 0)

4.9 AA CONTROL REGISTERS (AAR0–AAR3)

The address attribute register (AAR) appears in **Figure 4-6**. There are four of these registers in the DSP56309 (AAR0–AAR3), one for each AA signal.

For a full description of the address attribute registers see the DSP56300 Family Manual. Address multiplexing is not supported by the DSP56309. Bit 6 (BAM) of the AARs is reserved and should have only 0 written to it.

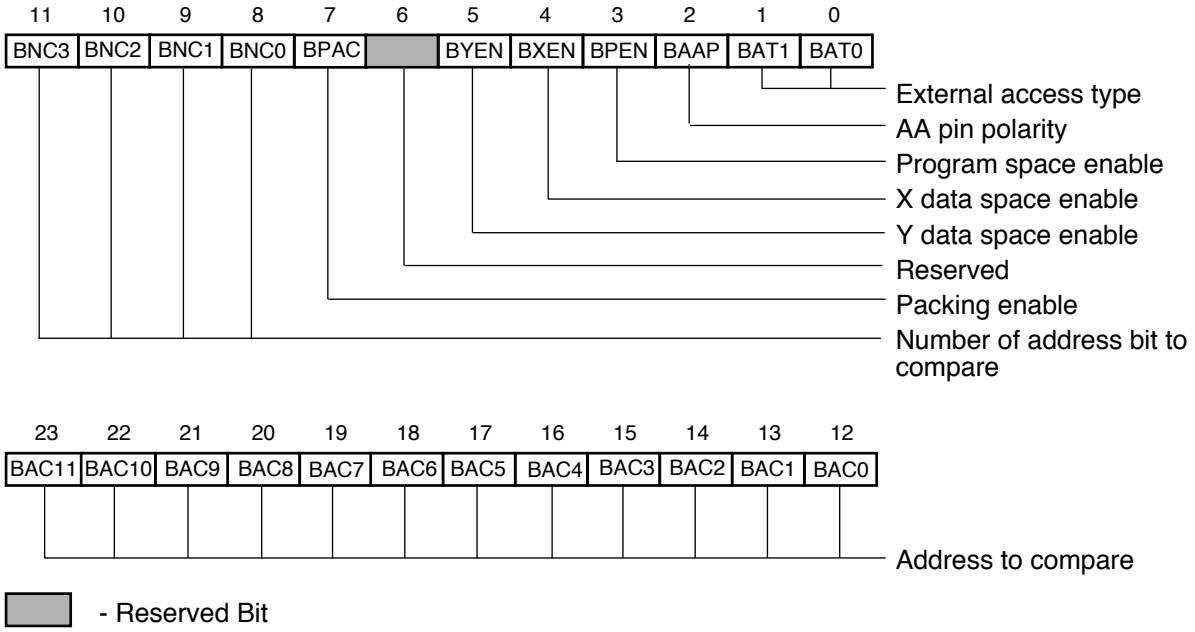


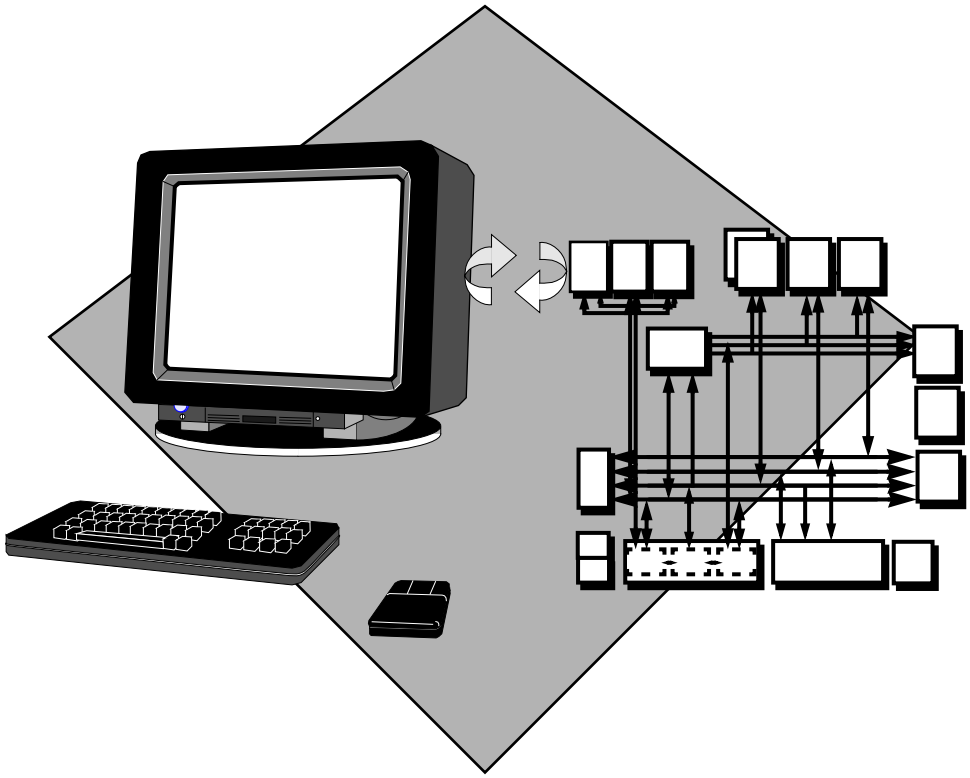
Figure 4-6 Address Attribute Registers (AAR0–AAR3)
(X:$FFFFF9–$FFFFF6)

4.10 JTAG BOUNDARY SCAN REGISTER (BSR)

The BSR in the DSP56309 JTAG implementation contains bits for all device signal and clock pins and associated control signals. All DSP56309 bidirectional pins have a corresponding register bit in the BSR for pin data and are controlled by an associated control bit in the BSR. The BSR is documented in **Section 11.5—DSP56309 Boundary Scan Register** on page 11-13. The JTAG code is listed in **Appendix C—DSP56309 BSDL Listing**.

SECTION 5

GENERAL-PURPOSE I/O



5.1 INTRODUCTION

The DSP56309 provides thirty-four bidirectional signals that can be configured as GPIO signals or as dedicated peripheral signals. No dedicated GPIO signals are provided. All of these signals are GPIO by default after reset. The control register settings of the DSP56309's peripherals determine whether these signals function as GPIO or as dedicated peripheral signals. This section describes how signals can function as GPIO.

5.2 PROGRAMMING MODEL

Section 2—Signal/Connection Descriptions of this manual documents the special uses of these signals in detail. There are five groups of these signals. They can be controlled separately or as groups. The groups include the following signals:

- Port B: sixteen GPIO signals (shared with the HI08 signals)
- Port C: six GPIO signals (shared with the ESSI0 signals)
- Port D: six GPIO signals (shared with the ESSI1 signals)
- Port E: three GPIO signals (shared with the SCI signals)
- Timers: three GPIO signals (shared with the Triple Timer signals)

5.2.1 Port B Signals and Registers

Each of the 16 Port B signals not used as a HI08 signal can be configured as a GPIO signal. The GPIO functionality of Port B is controlled by three registers: host control register (HCR), host port GPIO data register (HDR), and host port GPIO direction register (HDDR). These registers are documented in **Section 6—Host Interface (HI08)** of this manual.

5.2.2 Port C Signals and Registers

Each of the six Port C signals not used as an ESSI0 signal can be configured as a GPIO signal. The GPIO functionality of Port C is controlled by three registers: Port C control register (PCRC), Port C direction register (PRRC), and Port C data register (PDRC). These registers are documented in **Section 7—Enhanced Synchronous Serial Interface (ESSI)** of this manual.

5.2.3 Port D Signals and Registers

Each of the six Port D signals not used as a ESSI1 signal can be configured as a GPIO signal. The GPIO functionality of Port D is controlled by three registers: Port D control register (PCRD), Port D direction register (PRRD) and Port D data register (PDRD). These registers are also documented in **Section 7—Enhanced Synchronous Serial Interface (ESSI)** of this manual.

5.2.4 Port E Signals and Registers

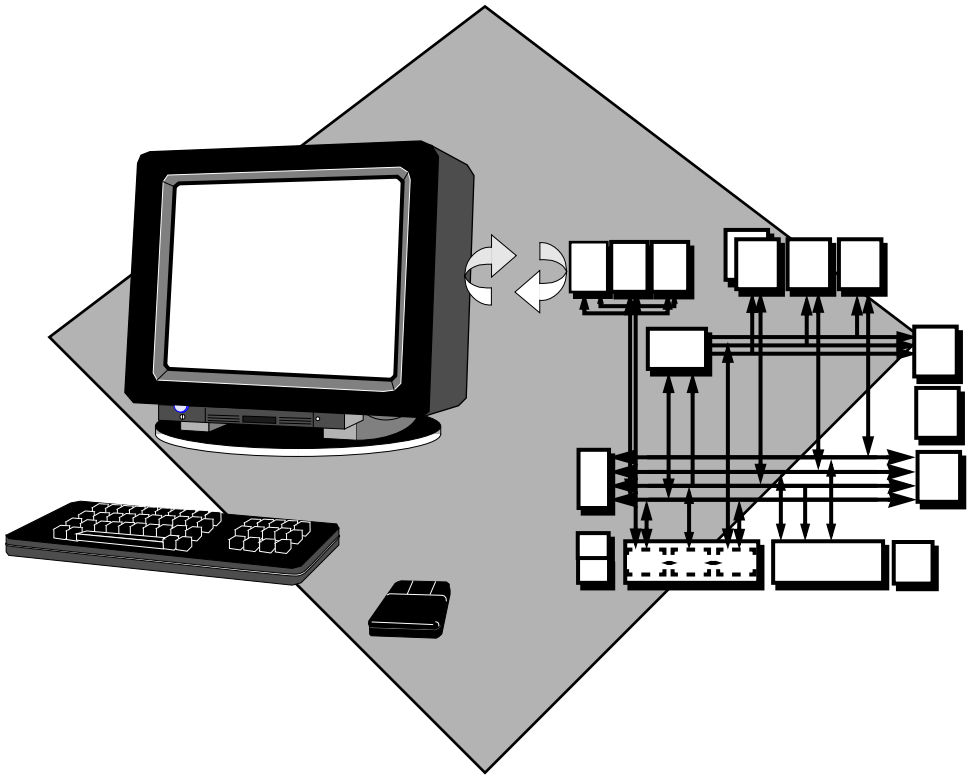
Each of the three Port E signals not used as a SCI signal can be configured as a GPIO signal. The GPIO functionality of Port E is controlled by three registers: Port E control register (PCRE), Port E direction register (PRRE) and Port E data register (PDRE). These registers are documented in **Section 8—Serial Communication Interface (SCI)** of this manual.

5.2.5 Triple Timer Signals

Each of the three triple timer interface signals (TIO0–TIO2) not used as a timer signal can be configured as a GPIO signal. Each signal is controlled by the appropriate timer control status register (TCSR0–TCSR2). These registers are documented in **Section 9—Triple Timer Module** of this manual.

SECTION 6

HOST INTERFACE (HI08)



6.1 INTRODUCTION

The Host Interface (HI08) is a byte-wide, full-duplex, double-buffered parallel port that can connect directly to the data bus of a host processor. The HI08 supports a variety of buses and provides glueless connection with a number of industry-standard microcomputers, microprocessors, and DSPs.

The host bus can operate asynchronously to the DSP core clock, so the HI08 registers are divided into two banks. The host register bank is accessible to the external host and the DSP register bank is accessible to the DSP core.

The HI08 supports two classes of interfaces:

- Host Processor/Microcontroller (MCU) connection interface
- GPIO port

Signals not used as HI08 port signals can be configured as GPIO signals, up to a total of 16.

6.2 HI08 FEATURES

This section lists the features of the host-to-DSP and DSP-to-host interfaces. Further details are given in **Section 6.5—HI08 DSP Side Programmer's Model** and **Section 6.8—HI08 Programming Model Quick Reference**. Also, see **Table 6-13** on page 6-34.

6.2.1 Host to DSP Core Interface

- Mapping:
  - Registers are directly mapped into eight internal X data memory locations
- Data word:
  - DSP56309 24-bit (native) data words are supported, as are 8-bit and 16-bit words
- Transfer modes:
  - DSP-to-host
  - Host-to-DSP

  - Host command
- Handshaking protocols:
  - Software polled
  - Interrupt driven
  - Core DMA accesses
- Instructions:
  - Memory-mapped registers allow the standard MOVE instruction to be used to transfer data between the DSP56309 and external hosts.
  - Special MOVEP instruction provides for I/O service capability using fast interrupts.
  - Bit addressing instructions (e.g., BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, JSSET) simplify I/O service routines.

6.2.2 HI08-to-Host Processor Interface

- Sixteen signals support non-multiplexed or multiplexed buses:
  - H0–H7/HAD0–HAD7 host data bus (H0–H7) or host multiplexed address/data bus (HAD0–HAD7)
  - HAS/HA0 address strobe (HAS) or host address line (HA0)
  - HA8/HA1 host address line (HA8) or host address line (HA1)
  - HA9/HA2 host address line (HA9) or host address line (HA2)
  - HRW/HRD read/write select (HRW) or read strobe (HRD)
  - HDS/HWR data strobe (HDS) or write strobe (HWR)
  - HCS/$\overline{\text{HA10}}$ host chip select (HCS) or host address line ($\overline{\text{HA10}}$)
  - HREQ/HTRQ host request (HREQ) or host transmit request (HTRQ)
  - HACK/HRRQ host acknowledge (HACK) or host receive request (HRRQ)
- Mapping:
  - HI08 registers are mapped into eight consecutive locations in external bus address space.
  - The HI08 acts as a memory or I/O-mapped peripheral for microprocessors, microcontrollers, etc.

- Data word: 8-bit
- Transfer modes:
  - Mixed 8-bit, 16-bit, and 24-bit data transfers
    - DSP-to-host
    - Host-to-DSP
  - Host command
- Handshaking protocols:
  - Software polled
  - Interrupt-driven (Interrupts are compatible with most processors, including the MC68000, 8051, HC11, and Hitachi H8.)
- Dedicated interrupts:
  - Separate interrupt lines for each interrupt source
  - Special host commands force DSP core interrupts under host processor control. These commands are useful for these purposes:
    - Real-time production diagnostics
    - Creating a debugging window for program development
    - Host control protocols
- Interface capabilities:
  - Glueless interface (no external logic required) to these devices:
    - Motorola HC11
    - Hitachi H8
    - 8051 family
    - Thomson P6 family
  - Minimal glue-logic (pullups, pulldowns) required to interface these devices:
    - ISA bus
    - Motorola 68K family
    - Intel® X86 family

6.3 HI08 HOST PORT SIGNALS

The host port signals are documented in **Section 2.8—Host Interface (HI08)**. Each host port signal can be programmed as a host port signal or as a GPIO signal, PB0–PB15, as in **Table 6-1** through **Table 6-3**.

Table 6-1 HI08 Signal Definitions for Various Operational Modes

HI08 Port Signal	Multiplexed Address/Data Bus Mode	Non-Multiplexed Bus Mode	GPIO Mode
HAD0–HAD7	HAD0–HAD7	H0–H7	PB0–PB7
HAS/HA0	$\overline{\text{HAS}}$/HAS	HA0	PB8
HA8/HA1	HA8	HA1	PB9
HA9/HA2	HA9	HA2	PB10
HCS/$\overline{\text{HA10}}$	$\overline{\text{HA10}}$	$\overline{\text{HCS}}$/HCS	PB13

Table 6-2 HI08 Data Strobe Signals

HI08 Port Signal	Single Strobe Bus	Dual Strobe Bus	GPIO Mode
HRW/HRD	HRW	$\overline{\text{HRD}}$/HRD	PB11
HDS/HWR	$\overline{\text{HDS}}$/HDS	$\overline{\text{HWR}}$/HWR	PB12

Table 6-3 HI08 Host Request Signals

HI08 Port Signal	Vector Required	No Vector Required	GPIO Mode
HREQ/ HTRQ	$\overline{\text{HREQ}}$/HREQ	$\overline{\text{HTRQ}}$/HTRQ	PB14
HACK/ HRRQ	$\overline{\text{HACK}}$/HACK	$\overline{\text{HRRQ}}$/HRRQ	PB15

6.4 HI08 BLOCK DIAGRAM

Figure 6-1 shows the HI08 registers. The top row of registers are for access by the DSP core. The bottom row of registers are for access by the host processor.

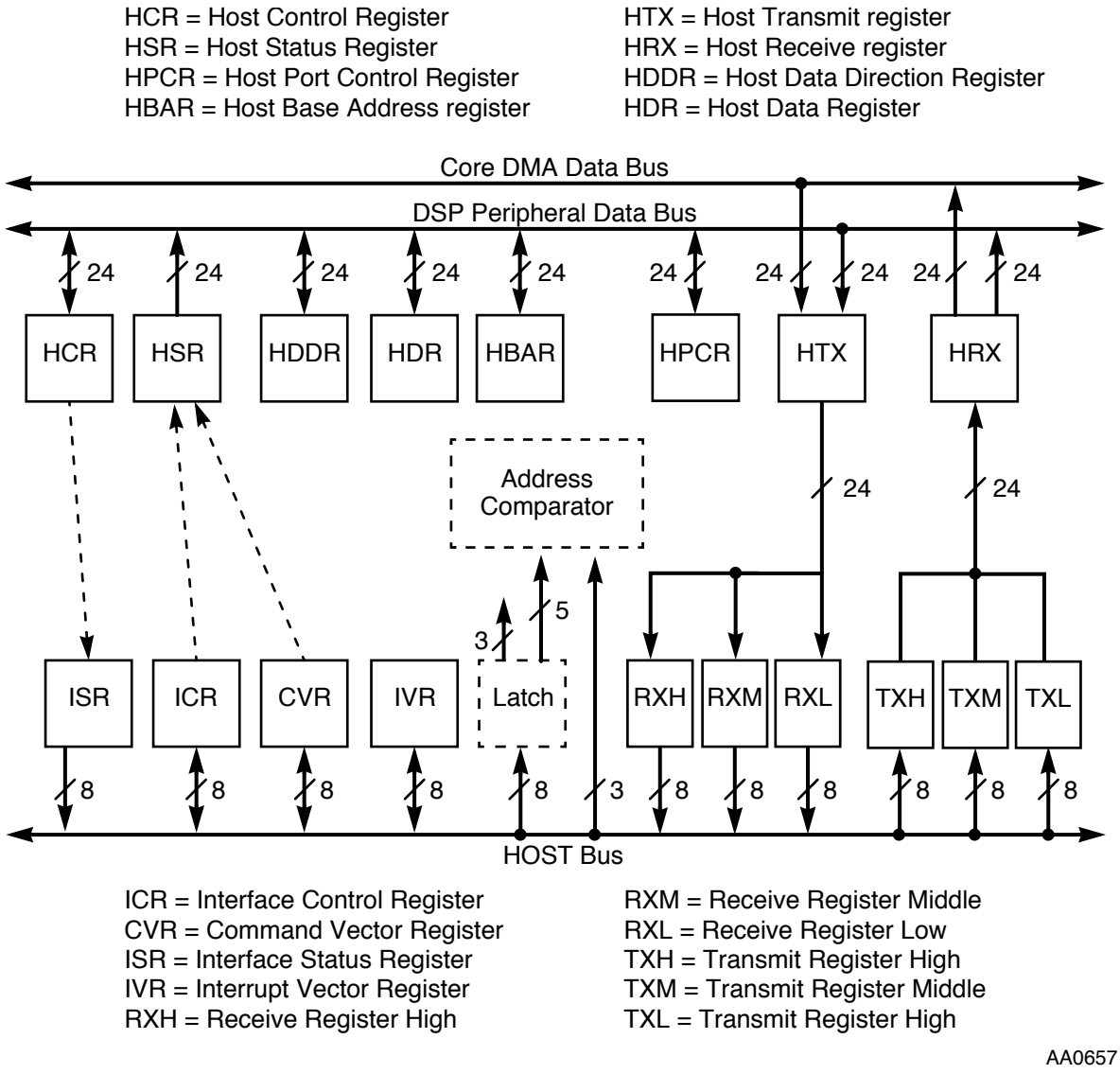


Figure 6-1 HI08 Block Diagram

6.5 HI08 DSP SIDE PROGRAMMER'S MODEL

The DSP56309 core treats the HI08 as a memory-mapped peripheral occupying eight 24-bit words in X data memory space. The DSP treats the HI08 as a normal memory-mapped peripheral, employing either standard polled or interrupt-driven programming techniques. Separate transmit and receive data registers are double-buffered to allow the DSP and host processor to transfer data efficiently at high speed. Direct memory mapping allows the DSP56309 core to communicate with the HI08 registers using standard instructions and addressing modes. In addition, the MOVEP instruction allows direct data transfers between DSP56309 internal memory and the HI08 registers or *vice versa*.

There are two kinds of host processor registers, data and control, with eight registers in all. All eight registers can be accessed by the DSP core but not by the external host.

Data registers are 24-bit registers used for high-speed data transfer to and from the DSP.

- Host data receive register (HRX)
- Host data transmit register (HTX)

The DSP side control registers are 16-bit registers that control DSP functions. The eight MSBs in the DSP side control registers are read by the DSP56309 as 0. Those registers are as follows:

- Host control register (HCR)
- Host status register (HSR)
- Host base address register (HBAR)
- Host port control register (HPCR)
- Host GPIO data direction register (HDDR)
- Host GPIO data register (HDR)

Both hardware $\overline{\text{RESET}}$ signals and software RESET instructions disable the HI08. After reset, the HI08 signals are configured to GPIO and disconnected from the DSP56309 core (i.e., the signals are left floating).

6.5.1 Host Receive Data Register (HRX)

The HRX register handles host-to-DSP data transfers. The DSP56309 views it as a 24-bit read-only register. Its address is X:$FFFFC6. It is loaded with 24-bit data from the

transmit data registers (TXH:TXM:TXL on the host side) when both the host's transmit data register empty (ISR:TXDE) bit and the DSP's host receive data full (HSR:HRDF) bits are cleared. The transfer operation sets both the TXDE and HRDF bits. When the HRDF bit is set, the HRX register contains valid data. The DSP56309 sets the HRIE bit (HCR, bit 0) to cause a host receive data interrupt when HRDF is set. When the DSP56309 reads the HRX register, the HRDF bit is cleared.

6.5.2 Host Transmit Data Register (HTX)

The HTX register handles for DSP-to-host data transfers. The DSP56309 views it as a 24-bit write-only register. Its address is X:$FFFFC7. Writing to the HTX register clears the DSP's host transfer data empty (HSR:HTDE) bit. The contents of the HTX register are transferred as 24-bit data to the receive byte registers (RXH:RXM:RXL) when both the HTDE and the host's receive data full (ISR:RXDF) bits are cleared. This transfer operation sets the HTDE and RXDF bits. The DSP56309 sets the HTIE bit to cause a host transmit data interrupt when HTDE is set. To prevent the previous data from being overwritten, data should not be written to the HTX until the HTDE bit is set.

Note: During data writes to a peripheral device, there is a two-cycle pipeline delay until any status bits affected by this operation are updated. If you read any of those status bits within the next two cycles, the bit does not reflect its current status. See the DSP56300 Family Manual, Appendix B, Polling a Peripheral Device for Write for further details.

6.5.3 Host Control Register (HCR)

The HCR is a 16-bit, read/write control register by which the DSP core controls the HI08 operating mode. Initialization values for HCR bits are documented in **Section 6.5.9—DSP Side Registers After Reset**. Reserved bits are read as 0 and should be written with 0 for future compatibility.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											HF3	HF2	HCIE	HTIE	HRIE

— Reserved bit, read as 0, should be written with 0 for future compatibility.

AA0658

Figure 6-2 Host Control Register (HCR) (X:$FFFFC2)

6.5.3.1 HCR Host Receive Interrupt Enable (HRIE) Bit 0

The HRIE bit generates a host receive data interrupt request if the host receive data full (HRDF) bit in the host status register (HSR, Bit 0), is set. The HRDF bit is set when data is written to the HRX. If HRIE is cleared, HRDF interrupts are disabled.

6.5.3.2 HCR Host Transmit Interrupt Enable (HTIE) Bit 1

The HTIE bit generates a host transmit data interrupt request if the host transmit data empty (HTDE) bit in the HSR is set. The HTDE bit is set when data is read from the HTX. If HTIE is cleared, HTDE interrupts are disabled.

6.5.3.3 HCR Host Command Interrupt Enable (HCIE) Bit 2

The HCIE bit generates a host command interrupt request if the host command pending (HCP) status bit in the HSR is set. If HCIE is cleared, HCP interrupts are disabled. The interrupt address is determined by the host command vector register (CVR).

Note: If more than one interrupt request source is asserted and enabled (e.g., HRDF is set, HCP is set, HRIE is set, and HCIE is set), the HI08 generates interrupt requests according to priorities shown in **Table 6-4**.

Table 6-4 Host Command Interrupt Priority List

Priority	Interrupt Source
Highest	Host Command (HCP = 1)
	Transmit Data (HTDE = 1)
Lowest	Receive Data (HRDF = 1)

6.5.3.4 HCR Host Flags 2,3 (HF[3:2]) Bits 3, 4

HF[3:2] bits are general-purpose flags for DSP-to-host communication. The DSP core sets and clears them. The values of HF[3:2] are reflected in the interface status register (ISR); that is, if they are modified by the DSP software, the host processor can read the modified values by reading the ISR.

These two flags can be used individually or as encoded pairs in a simple DSP-to-host communication protocol, implemented in both the DSP and the host processor software.

6.5.3.5 HCR Reserved Bits 5-15

These bits are reserved. They are read as 0 and should be written with 0.

6.5.4 Host Status Register (HSR)

The HSR is a 16-bit, read-only status register by which the DSP reads the HI08 status and flags. The host processor cannot access it directly. Reserved bits are read as 0 and should be written with 0. The initialization values for the HSR bits are described in **Section 6.5.9—DSP Side Registers After Reset** on page 6-18.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											HF1	HF0	HCP	HTDE	HRDF

— Reserved bit, read as 0, should be written with 0 for future compatibility.

AA0659

Figure 6-3 Host Status Register (HSR) (X:$FFFFC3)

6.5.4.1 HSR Host Receive Data Full (HRDF) Bit 0

The HRDF bit indicates that the host receive data register (HRX) contains data from the host processor. HRDF is set when data is transferred from the TXH:TXM:TXL registers to the HRX register. If HRDF is set, the HI08 generates a receive data full DMA request. HRDF is cleared when the DSP core reads the HRX. HRDF is also cleared when the host processor uses the initialize function.

6.5.4.2 HSR Host Transmit Data Empty (HTDE) Bit 1

The HTDE bit indicates that the host transmit data register (HTX) is empty and can be written by the DSP core. HTDE is set when the HTX register is transferred to the RXH:RXM:RXL registers. HTDE is also set when the host processor uses the initialize function. If HTDE is set, the HI08 generates a transmit data full DMA request. HTDE is cleared when HTX is written by the DSP core.

6.5.4.3 HSR Host Command Pending (HCP) Bit 2

The HCP bit indicates that the host has set the HC bit and that a host command interrupt is pending. The HCP bit reflects the status of the HC bit in the CVR. HC and HCP are cleared by the HI08 hardware when the interrupt request is serviced by the DSP core. If the host clears HC, HCP is also cleared.

6.5.4.4 HSR Host Flags 0, 1 (HF[1:0]) Bits 3, 4

HF[1:0] bits are used as general-purpose flags for host-to-DSP communication. HF[1:0] can be set or cleared by the host. These bits reflect the status of host flags HF[1:0] in the ICR on the host side. They can be used individually or as encoded pairs in a simple host-to-DSP communication protocol implemented in both the DSP and the host processor software.

6.5.4.5 HSR Reserved Bits 5-15

These bits are reserved. They are read as 0 and should be written with 0.

6.5.5 Host Base Address Register (HBAR)

The HBAR is used in multiplexed bus modes. This register, illustrated in **Figure 6-4**, selects the base address where the host side registers are mapped into the bus address space. The address from the host bus is compared with the base address as programmed in the base address register. If the addresses match, an internal chip select is generated. The use of this register by the chip select logic is described in **Figure 6-5**.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3

AA0665

Figure 6-4 Host Base Address Register (HBAR) (X:$FFFFC5)

6.5.5.1 HBAR Base Address (BA[10:3]) Bits 0-7

These bits reflect the base address where the host-side registers are mapped into the bus address space.

6.5.5.2 HBAR Reserved Bits 8-15

These bits are reserved. They are read as 0 and should be written with 0.

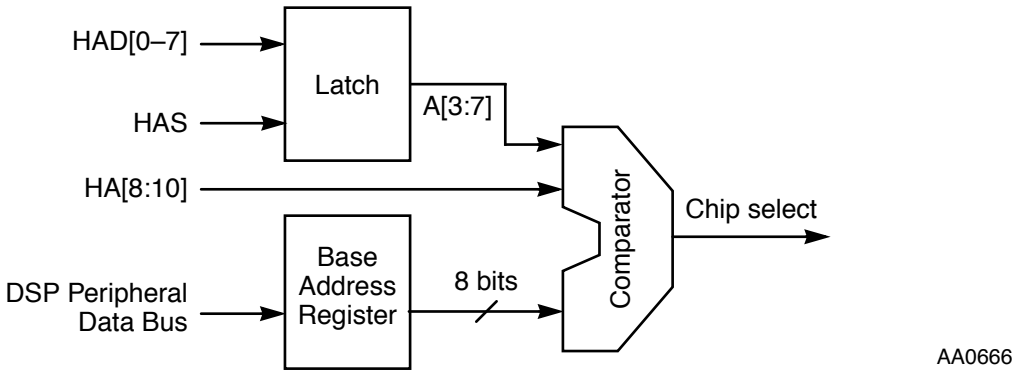


Figure 6-5 Self Chip Select Logic

6.5.6 Host Port Control Register (HPCR)

The HPCR is a 16-bit, read/write control register by which the DSP controls the HI08 operating mode. Reserved bits are read as 0 and should be written with 0 for future compatibility. The initialization values for the HPCR bits are described in **Section 6.5.9—DSP Side Registers After Reset**. The HPCR bits are illustrated in **Figure 6-6**.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAP	HRP	HCSP	HDDS	HMUX	HASP	HDSP	HROD		HEN	HAEN	HREN	HCSEN	HA9EN	HA8EN	HGEN

—Reserved bit, read as 0, should be written with 0 for future compatibility.

AA0660

Figure 6-6 Host Port Control Register (HPCR) (X:$FFFFC4)

Note: To assure proper operation of the DSP56309, the HPCR bits HAP, HRP, HCSP, HDDS, HMUX, HASP, HDSP, HROD, HAEN, and HREN should be changed only if HEN is cleared.

To assure proper operation of the DSP56309, the HPCR bits HAP, HRP, HCSP, HDDS, HMUX, HASP, HDSP, HROD, HAEN, HREN, HCSEN, HA9EN, and HA8EN should not be set when HEN is set or simultaneously with setting HEN.

6.5.6.1 HPCR Host GPIO Port Enable (HGEN) Bit 0

If HGEN is set, signals configured as GPIO are enabled. If this bit is cleared, signals configured as GPIO are disconnected; outputs are high impedance, and inputs are electrically disconnected. Signals configured as HI08 are not affected by the value of HGEN.

6.5.6.2 HPCR Host Address Line 8 Enable (HA8EN) Bit 1

If HA8EN is set and the HI08 is in multiplexed bus mode, then HA8/A1 acts as host address line 8 (HA8). If this bit is cleared and the HI08 is in multiplexed bus mode, then HA8/HA1 acts as a GPIO signal according to the value of the HDDR and HDR.

Note: HA8EN is ignored when the HI08 is not in the multiplexed bus mode (HMUX is cleared).

6.5.6.3 HPCR Host Address Line 9 Enable (HA9EN) Bit 2

If HA9EN is set and the HI08 is in multiplexed bus mode, then HA9/HA2 acts as host address line 9 (HA9). If this bit is cleared, and the HI08 is in multiplexed bus mode, then HA9/HA2 is configured as a GPIO signal according to the value of the HDDR and HDR.

Note: HA9EN is ignored when the HI08 is not in the multiplexed bus mode (HMUX is cleared).

6.5.6.4 HPCR Host Chip Select Enable (HCSEN) Bit 3

If the HCSEN bit is set, then HCS/HA10 is used as host chip select (HCS) in the non-multiplexed bus mode (HMUX is cleared), and as host address line 10 (HA10) in the multiplexed bus mode (HMUX is set). If this bit is cleared, then HCS/HA10 is configured as a GPIO signal according to the value of the HDDR and HDR.

6.5.6.5 HPCR Host Request Enable (HREN) Bit 4

The HREN bit controls the host request signals. If HREN is set and the HI08 is in the single host request mode (HDRQ is cleared in the host interface control register (ICR)), HREQ/HTRQ is configured as the host request (HREQ) output. If HREN is cleared, HREQ/HTRQ and HACK/HRRQ are configured as GPIO signals according to the value of the HDDR and HDR.

If HREN is set in the double host request mode (HDRQ is set in the ICR), HREQ/HTRQ is configured as the host transmit request (HTRQ) output and HACK/HRRQ as the host receive request (HRRQ) output. If HREN is cleared, HREQ/HTRQ and HACK/HRRQ are configured as GPIO signals according to the value of the HDDR and HDR.

6.5.6.6 HPCR Host Acknowledge Enable (HAEN) Bit 5

The HAEN bit controls the HACK signal. In the single host request mode (HDRQ is cleared in the ICR), if HAEN and HREN are both set, HACK/HRRQ is configured as the host acknowledge (HACK) input. If HAEN or HREN is cleared, HACK/HRRQ is configured as a GPIO signal according to the value of the HDDR and HDR. In the double host request mode (HDRQ is set in the ICR), HAEN is ignored.

6.5.6.7 HPCR Host Enable (HEN) Bit 6

If HEN is set, the HI08 operates as the host interface. If HEN is cleared, the HI08 is not active, and all the HI08 signals are configured as GPIO signals according to the value of the HDDR and HDR.

6.5.6.8 HPCR Reserved Bit 7

This bit is reserved. It is read as 0 and should be written as 0.

6.5.6.9 HPCR Host Request Open Drain (HROD) Bit 8

The HROD bit controls the output drive of the host request signals. In the single host request mode (HDRQ is cleared in ICR), if HROD is cleared and host requests are enabled (HREN is set and HEN is set in HPCR), the HREQ signal is always driven by the HI08. If HROD is set and host requests are enabled, the HREQ signal is an open drain output. In the double host request mode (HDRQ is set in the ICR), if HROD is cleared and host requests are enabled (HREN is set and HEN is set in the HPCR), the HTRQ and HRRQ signals are always driven. If HROD is set and host requests are enabled, the HTRQ and HRRQ signals are open drain outputs.

6.5.6.10 HPCR Host Data Strobe Polarity (HDSP) Bit 9

If HDSP is cleared, the data strobe signals are configured as active low inputs, and data is transferred when the data strobe is low. If HDSP is set, the data strobe signals are configured as active high inputs, and data is transferred when the data strobe is high. The data strobe signals are either HDS by itself or both HRD and HWR together.

6.5.6.11 HPCR Host Address Strobe Polarity (HASP) Bit 10

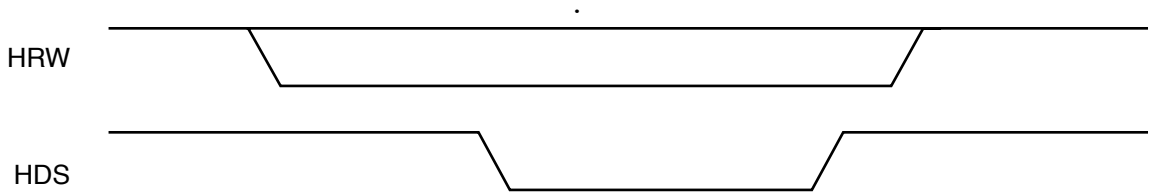
If HASP is cleared, the host address strobe (HAS) signal is an active low input, and the address on the host address/data bus is sampled when the HAS signal is low. If HASP is set, HAS is an active high address strobe input, and the address on the host address or data bus is sampled when the HAS signal is high.

6.5.6.12 HPCR Host Multiplexed Bus (HMUX) Bit 11

If HMUX is set, the HI08 latches the lower portion of a multiplexed address/data bus. In this mode the internal address line values of the host registers are taken from the internal latch. If HMUX is cleared, it indicates that the HI08 is connected to a non-multiplexed type of bus. The values of the address lines are then taken from the HI08 input signals.

6.5.6.13 HPCR Host Dual Data Strobe (HDDS) Bit 12

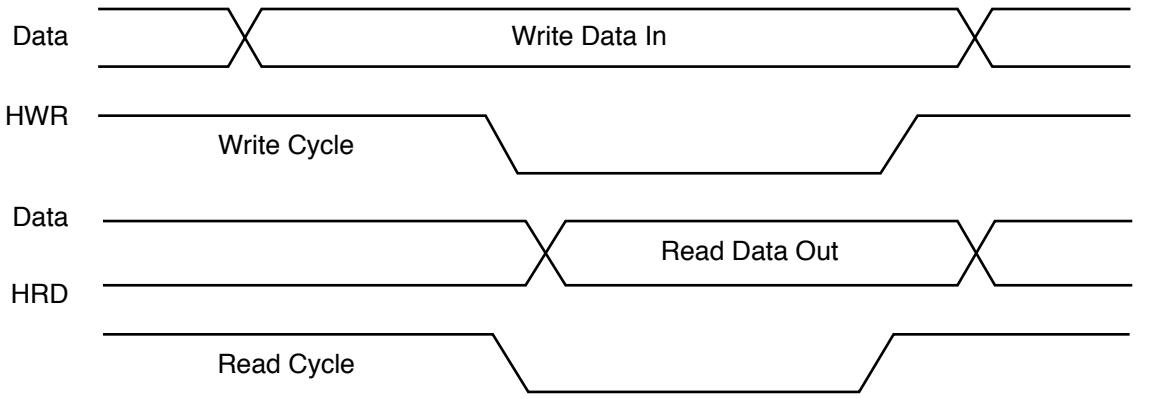
If the HDDS bit is cleared, the HI08 operates in the single-strobe bus mode. In this mode, the bus has a single data strobe signal for both reads and writes. If set, the HI08 operates in the dual-strobe bus mode. In this mode, the bus has two separate data strobes, one for data reads, the other for data writes. See **Figure 6-7** and **Figure 6-8** for more information on the two types of buses.



In a single-strobe bus, a DS (data strobe) signal qualifies the access, while a R/W (Read-Write) signal specifies the direction of the access.

AA0661

Figure 6-7 Single Strobe Bus



In dual strobe bus, there are separate HRD and HWR signals that specify the access as being a read or write access, respectively.

AA0662

Figure 6-8 Dual Strobe Bus

6.5.6.14 HPCR Host Chip Select Polarity (HCSP) Bit 13

If the HCSP bit is cleared, the host chip select (HCS) signal is configured as an active low input and the HI08 is selected when the HCS signal is low. If the HCSP signal is set, HCS is configured as an active high input, and the HI08 is selected when the HCS signal is high.

6.5.6.15 HPCR Host Request Polarity (HRP) Bit 14

The HRP bit controls the polarity of the host request signals. In the single-host request mode (HDRQ is cleared in the ICR), if HRP is cleared, and host requests are enabled (HREN is set and HEN is set), the HREQ signal is an active low output. If HRP is set and host requests are enabled, the HREQ signal is an active high output.

In the double-host request mode (HDRQ is set in the ICR), if HRP is cleared, and host requests are enabled (HREN is set and HEN is set), the HTRQ and HRRQ signals are active low outputs. If HRP is set, and host requests are enabled, the HTRQ and HRRQ signals are active high outputs.

6.5.6.16 HPCR Host Acknowledge Polarity (HAP) Bit 15

If the HAP bit is cleared, the host acknowledge (HACK) signal is configured as an active low input. The HI08 drives the contents of the IVR onto the host bus when the HACK signal is low. If the HAP bit is set, the HACK signal is configured as an active high input. The HI08 outputs the contents of the IVR when the HACK signal is high.

6.5.7 Host Data Direction Register (HDDR)

The HDDR controls the direction of the data flow for each of the HI08 signals configured as GPIO. It is illustrated in **Figure 6-9**. Even when the HI08 functions as the host interface, its unused signals can be configured as GPIO signals. For information on the HI08 GPIO configuration options, see **Section 6.6.8—General-Purpose I/O** on page 6-30. If bit DRxx is set, the corresponding HI08 signal is configured as an output signal. If bit DRxx is cleared, the corresponding HI08 signal is configured as an input signal.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

AA0663

Figure 6-9 Host Data Direction Register (HDDR) (X:$FFFFC8)

6.5.8 Host Data Register (HDR)

The HDR register holds the data value of the corresponding bits of the HI08 signals configured as GPIO signals. It is illustrated in **Figure 6-10**. The functionality of the Dxx bit depends on the corresponding HDDR bit (DRxx), as in **Table 6-5**. The HDR cannot be accessed by the host processor.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

AA0664

Figure 6-10 Host Data Register (HDR) (X:$FFFFC9)

Table 6-5 HDR and HDDR Functionality

HDDR	HDR	
DRxx	Dxx	
	Configured as GPIO signal	Configured as non-GPIO signal
0	Read only bit— The value read is the binary value of the signal. The corresponding signal is configured as an input.	Read only bit—Does not contain significant data.
1	Read/write bit— The value written is the value read. The corresponding signal is configured as an output, and is driven with the data written to Dxx.	Read/write bit— The value written is the value read.

6.5.9 DSP Side Registers After Reset

Table 6-6 shows the results of the four reset types on the bits in each of the HI08 registers accessible by the DSP56309. Reset types are as follows:

- Hardware reset (HW)—caused by the $\overline{\text{RESET}}$ signal
- Software reset (SW)—caused by executing the RESET instruction
- Individual reset (IR)—caused by clearing the HPCR:HEN
- Stop reset (ST)—caused by executing the STOP instruction.

Table 6-6 DSP Side Registers after Reset

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	IR Reset	ST Reset
HCR	All bits	0	0	bit value indeterminate after reset	—
HPCR	All bits	0	0	—	—
HSR	HF[1:0]	0	0	—	—
	HCP	0	0	0	0
	HTDE	1	1	1	1
	HRDF	0	0	0	0
HBAR	BA[10:3]	$80	$80	—	—
HDDR	DR[15:0]	0	0	—	—
HDR	D[15:0]	—	—	—	—
HRX	HRX [23:0]	empty	empty	empty	empty
HTX	HTX [23:0]	empty	empty	empty	empty

6.5.10 Host Interface DSP Core Interrupts

The HI08 can request interrupt service from either the DSP56309 or the host processor. The DSP56309 interrupts are internal and do not require the use of an external interrupt signal. When the appropriate interrupt enable bit in the HCR is set, an interrupt condition caused by the host processor sets the appropriate bit in the HSR, generating an interrupt request to the DSP56309. (See **Figure 6-11**.) The DSP56309 acknowledges interrupts caused by the host processor by jumping to the appropriate interrupt service routine. There are three possible interrupts:

- Host command
- Transmit data register empty
- Receive data register full

Although there is a set of vectors reserved for host command use, the host command can access any interrupt vector in the interrupt vector table. The DSP interrupt service

routine must read or write the appropriate HI08 register (e.g., clearing HRDF or HTDE) to clear the interrupt. For host command interrupts, the interrupt acknowledge from the DSP56309 program controller clears the pending interrupt condition.

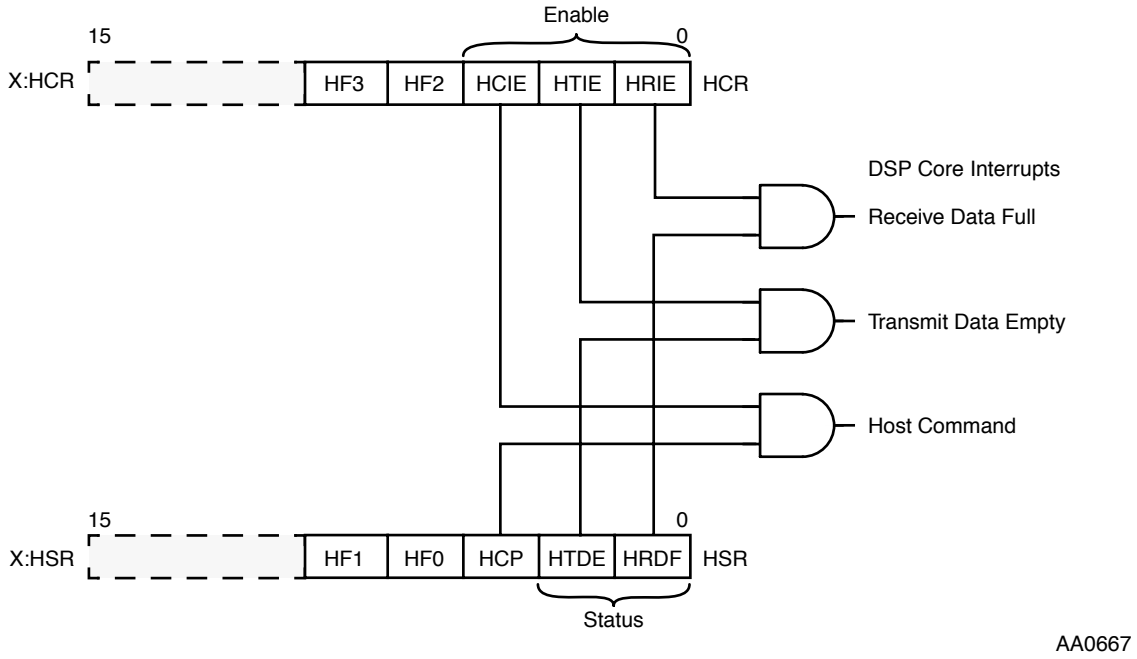


Figure 6-11 HSR-HCR Operation

6.6 HI08-EXTERNAL HOST PROGRAMMER'S MODEL

The HI08 is a simple, high speed interface to a host processor. To the host bus, the HI08 appears to be eight byte-wide registers. Separate transmit and receive data registers are double-buffered to allow the DSP core and host processor to transfer data efficiently at high speed. The host can access the HI08 asynchronously by using polling techniques or interrupt-based techniques.

The HI08 appears to the host processor as a memory-mapped peripheral occupying eight bytes in the host processor address space, as in **Table 6-7** on page 6-22. The eight HI08 registers include the following:

- A control register (ICR)
- A status register (ISR)
- Three data registers (RXH/TXH, RXM/TXM, and RXL/TXL)
- Two vector registers (IVR and CVR)

The CVR is a special command register by which the host processor issues commands to the DSP56309. Only the host processor can access this register.

Host processors can use standard host processor instructions (e.g., byte move) and addressing modes to communicate with the HI08 registers. The HI08 registers are aligned so that 8-bit host processors can use 8/16/24-bit load and store instructions for data transfers. The HREQ/HTRQ and HACK/HRRQ handshake flags are provided for polled or interrupt-driven data transfers with the host processor. Because of the speed of the DSP56309 interrupt response, most host microprocessors can load or store data at their maximum programmed I/O instruction rate without testing the handshake flags for each transfer. If full handshake is not needed, the host processor can treat the DSP56309 as a fast device, and data can be transferred between the host processor and the DSP56309 at the fastest host processor data rate.

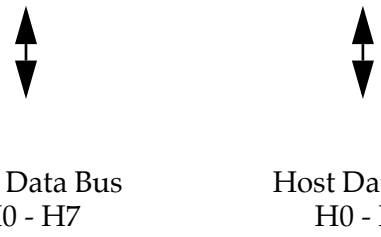
One of the most innovative features of the host interface is the host command feature. With this feature, the host processor can issue vectored interrupt requests to the DSP56309. The host can select any of 128 DSP interrupt routines for execution by writing a vector address register in the HI08. This flexibility allows the host processor to execute up to 128 pre-programmed functions inside the DSP56309. For example, use of the DSP56309 host interrupts can allow the host processor to read or write DSP registers (X, Y, or program memory locations), force interrupt handlers (e.g., SSI, SCI, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$ interrupt routines), and perform control and debugging operations.

Note: When the DSP enters stop mode, the HI08 signals are electrically disconnected internally, thus disabling the HI08 until the core leaves stop mode. While the HI08 configuration remains unchanged in stop mode, the core cannot be restarted via the HI08 interface.

Do not issue a STOP command to the DSP via the HI08 unless some other mechanism for exiting stop mode is provided.

Table 6-7 Host Side Register Map

Host Address	Big Endian HLEND = 0		Little Endian HLEND = 1	
0	ICR		ICR	Interface Control
1	CVR		CVR	Command Vector
2	ISR		ISR	Interface Status
3	IVR		IVR	Interrupt Vector
4	00000000		00000000	Unused
5	RXH/TXH		RXL/TXL	Receive/Transmit Bytes
6	RXM/TXM		RXM/TXM	
7	RXL/TXL		RXH/TXH	



6.6.1 Interface Control Register (ICR)

The ICR is an 8-bit, read/write control register by which the host processor controls the HI08 interrupts and flags. It is illustrated in **Figure 6-12**. The DSP core cannot access the ICR. The ICR is a read/write register, which allows the use of bit manipulation instructions on control register bits. The control bits are described in the following paragraphs.

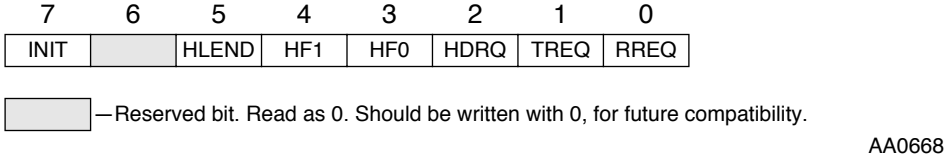


Figure 6-12 Interface Control Register

6.6.1.1 ICR Receive Request Enable (RREQ) Bit 0

The RREQ bit controls the HREQ signal for host receive data transfers. RREQ enables host requests via the host request (HREQ or HRRQ) signal when the receive data register full (RXDF) status bit in the ISR is set. If RREQ is cleared, RXDF interrupts are disabled. If RREQ and RXDF are set, the host request signal (HREQ or HRRQ) is asserted.

6.6.1.2 ICR Transmit Request Enable (TREQ) Bit 1

TREQ enables host requests via the host request (HREQ or HTRQ) signal when the transmit data register empty (TXDE) status bit in the ISR is set. If TREQ is cleared, TXDE interrupts are disabled. If TREQ and TXDE are set, the host request signal is asserted.

Table 6-8 and **Table 6-9** summarize the effect of RREQ and TREQ on the HREQ and HRRQ signals.

Table 6-8 TREQ and RREQ modes (HDRQ = 0)

TREQ	RREQ	HREQ Signal
0	0	No Interrupts (Polling)
0	1	RXDF Request (Interrupt)
1	0	TXDE Request (Interrupt)
1	1	RXDF and TXDE Request (Interrupts)

Table 6-9 TREQ and RREQ modes (HDRQ = 1)

TREQ	RREQ	HTRQ Single	HRRQ Signal
0	0	No Interrupts (Polling)	No Interrupts (Polling)
0	1	No Interrupts (Polling)	RXDF Request (Interrupt)
1	0	TXDE Request (Interrupt)	No Interrupts (Polling)
1	1	TXDE Request (Interrupt)	RXDF Request (Interrupt)

6.6.1.3 ICR Double Host Request (HDRQ) Bit 2

If cleared, the HDRQ bit configures HREQ/HTRQ and HACK/HRRQ as HREQ and HACK, respectively. If HDRQ is set, HREQ/HTRQ and HACK/HRRQ are configured as HTRQ and HRRQ, respectively.

6.6.1.4 ICR Host Flag 0 (HF0) Bit 3

The HF0 bit is a general-purpose flag for host-to-DSP communication. The host processor can set or clear HF0, and the DSP56309 cannot change this bit. HF0 is reflected in the HSR on the DSP side of the HI08.

6.6.1.5 ICR Host Flag 1 (HF1) Bit 4

The HF1 bit is a general-purpose flag for host-to-DSP communication. The host processor can set or clear HF1, and the DSP56309 cannot change this bit. HF1 is reflected in the HSR on the DSP side of the HI08.

6.6.1.6 ICR Host Little Endian (HLEND) Bit 5

If the HLEND bit is cleared, the host can access the HI08 in big endian byte order. If set, the host can access the HI08 in little endian byte order. If the HLEND bit is cleared the RXH/TXH register is located at address $5, the RXM/TXM register at $6, and the RXL/TXL register at $7. If the HLEND bit is set, the RXH/TXH register is located at address $7, the RXM/TXM register at $6, and the RXL/TXL register at $5.

6.6.1.7 ICR Reserved Bit 6

This bit is reserved. It is read as 0 and should be written with 0.

6.6.1.8 ICR Initialize Bit (INIT) Bit 7

The host processor uses the INIT bit to force initialization of the HI08 hardware. During initialization, the HI08 transmit and receive control bits are configured. Using the INIT bit to initialize the HI08 hardware may or may not be necessary, depending on the software design of the interface.

The t®ype of initialization done when the INIT bit is set depends on the state of TREQ and RREQ in the HI08. The INIT command, which is local to the HI08, can conveniently configure the HI08 into the desired data transfer mode. The effect of the INIT command is described in **Table 6-10**. When the host sets the INIT bit, the HI08 hardware executes the INIT command. The interface hardware clears the INIT bit after the command has been executed.

Table 6-10 INIT Command Effects

TREQ	RREQ	After INIT Execution	Transfer Direction Initialized
0	0	INIT = 0	None
0	1	INIT = 0; RXDF = 0; HTDE = 1	DSP to Host
1	0	INIT = 0; TXDE = 1; HRDF = 0	Host to DSP
1	1	INIT = 0; RXDF = 0; HTDE = 1; TXDE = 1; HRDF = 0	Host to/from DSP

6.6.2 Command Vector Register (CVR)

The host processor uses the CVR to cause the DSP56309 to execute an interrupt. The host command feature is independent of any of the data transfer mechanisms in the HI08. It can cause any of the 128 possible interrupt routines in the DSP core to be executed. This register is illustrated in **Figure 6-13**.

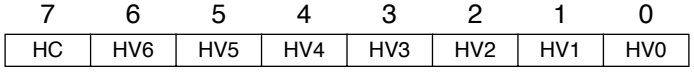


Figure 6-13 Command Vector Register (CVR)

6.6.2.1 CVR Host Vector (HV[6:0]) Bits 0–6

The seven HV bits select the host command interrupt address to be used by the host command interrupt logic. When the host command interrupt is recognized by the DSP interrupt control logic, the address of the interrupt routine taken is 2 × HV. The host can write HC and HV in the same write cycle.

The host processor can select any of the 128 possible interrupt routine starting addresses in the DSP by writing the interrupt routine address divided by two into the HV bits. This means that the host processor can force any of the existing interrupt handlers (SSI, SCI,

IRQA, IRQB, etc.) and can use any of the reserved or otherwise unused addresses (provided they have been pre-programmed in the DSP). HV is set to $32 (vector location $0064) by a hardware $\overline{\text{RESET}}$ signal, software RESET instruction, individual reset, or a STOP instruction.

6.6.2.2 CVR Host Command Bit (HC) Bit 7

The host processor uses the HC bit to handshake the execution of host command interrupts. Normally, the host processor sets HC to request a host command interrupt from the DSP56309. When the DSP56309 acknowledges the host command interrupt, the HI08 hardware clears the HC bit. The host processor can read the state of HC to determine when the host command has been accepted. After setting HC, the host must not write to the CVR again until the HI08 hardware clears HC. Setting the HC bit causes host command pending (HCP) to be set in the HSR. The host can write to the HC and HV bits in the same write cycle.

6.6.3 Interface Status Register (ISR)

The interface status register (ISR) is an 8-bit, read-only status register used by the host processor to interrogate the status and flags of the HI08. The host processor can write to this address without affecting the internal state of the HI08. The DSP core cannot access the ISR. The ISR bits are described in the following paragraphs. This register is illustrated in **Figure 6-14**.

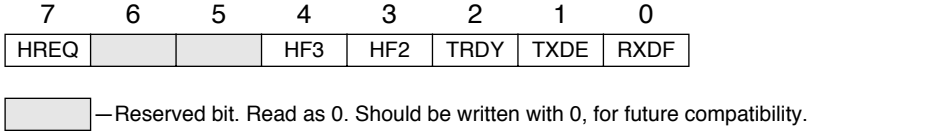


Figure 6-14 Interface Status Register

6.6.3.1 ISR Receive Data Register Full (RXDF) Bit 0

The RXDF bit indicates that the receive byte registers (RXH:RXM:RXL) contain data from the DSP56309 and can be read by the host processor. RXDF is set when the HTX is transferred to the receive byte registers. RXDF is cleared when the receive data (RXL or RXH according to HLEND bit) register is read by the host processor. RXDF can be cleared by the host processor using the initialize function. RXDF can assert the external HREQ signal if the RREQ bit is set. Regardless of whether the RXDF interrupt is enabled, RXDF indicates whether the RX registers are full and data can be latched out so that the host processor can use polling techniques.

6.6.3.2 ISR Transmit Data Register Empty (TXDE) Bit 1

The TXDE bit indicates that the transmit byte registers (TXH:TXM:TXL) are empty and can be written by the host processor. TXDE is set when the contents of the transmit byte registers are transferred to the HRX register. TXDE is cleared when the transmit (TXL or TXH according to HLEND bit) register is written by the host processor. The host processor can set TXDE using the initialize function. TXDE can assert the external HTRQ signal if the TREQ bit is set. Regardless of whether the TXDE interrupt is enabled, TXDE indicates whether the TX registers are full and data can be latched in so that the host processor can use polling techniques.

6.6.3.3 ISR Transmitter Ready (TRDY) Bit 2

The TRDY status bit indicates that TXH:TXM:TXL, and the HRX registers are empty.

TRDY = TXDE and $\overline{\text{HRDF}}$

If TRDY is set, the data that the host processor writes to TXH:TXM:TXL is immediately transferred to the DSP side of the HI08. This feature has many applications. For example, if the host processor issues a host command which causes the DSP56309 to read the HRX, the host processor can be guaranteed that the data it just transferred to the HI08 is that being received by the DSP56309.

6.6.3.4 ISR Host Flag 2 (HF2) Bit 3

HF2 indicates the state of host flag 2 in the HCR on the DSP side. HF2 can be changed only by the DSP56309, as documented in **Section 6.5.3.4—HCR Host Flags 2,3 (HF[3:2]) Bits 3, 4** on page 6-10.

6.6.3.5 ISR Host Flag 3 (HF3) Bit 4

HF3 indicates the state of Host Flag 3 in the HCR on the DSP side. HF3 can be changed only by the DSP56309, as documented in **Section 6.5.3.4—HCR Host Flags 2,3 (HF[3:2]) Bits 3, 4** on page 6-10.

6.6.3.6 ISR Reserved Bits 5, 6

These bits are reserved. They are read as 0 and should be written with 0.

6.6.3.7 ISR Host Request (HREQ) Bit 7

HREQ indicates the status of the external transmit and receive request output signals (HTRQ and HRRQ) if HDRQ is set. If HDRQ is cleared, it indicates the status of the external host request output signal (HREQ). **Table 6-11** shows possible settings of HRDQ and HDEQ and their effects.

Table 6-11 HREQ and HDRQ Settings

HDRQ	HREQ	Effect
0	0	HREQ is cleared; no host processor interrupts are requested.
0	1	HREQ is set; an interrupt is requested.
1	0	HTRQ and HRRQ are cleared, no host processor interrupts are requested.
1	1	HTRQ or HRRQ are set; an interrupt is requested.

The HREQ is set from either or both of two conditions—either the receive byte registers are full or the transmit byte registers are empty. These conditions are indicated by the ISR RXDF and TXDE status bits, respectively. If the interrupt source has been enabled by the associated request enable bit in the ICR, HREQ is set if one or more of the two enabled interrupt sources is set.

6.6.4 Interrupt Vector Register (IVR)

The IVR is an 8-bit, read/write register which typically contains the interrupt vector number used with MC68000 family processor vectored interrupts. Only the host processor can read and write this register. The contents of the IVR are placed on the host data bus, H[7:0], when both the HREQ and HACK signals are asserted. The contents of this register are initialized to $0F by a hardware $\overline{\text{RESET}}$ signal or software RESET instruction. This value corresponds to the uninitialized interrupt vector in the MC68000 family. This register is illustrated in **Figure 6-15**.

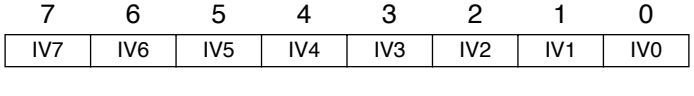


Figure 6-15 Interrupt Vector Register (IVR)

6.6.5 Receive Byte Registers (RXH: RXM: RXL)

The receive byte registers are viewed by the host processor as three 8-bit, read-only registers. These registers are the receive high register (RXH), the receive middle register (RXM), and the receive low register (RXL). They receive data from the high, middle, and

low bytes, respectively, of the HTX register and are selected by the external host address inputs (HA[2:0]) during a host processor read operation.

The memory address of the receive byte registers is set by the HLEND bit in the ICR. If the HLEND bit is set, the RXH is located at address $7, RXM at $6, and RXL at $5. If the HLEND bit is cleared, the RXH is located at address $5, RXM at $6, and RXL at $7.

When data is written to the receive byte register at host address $7, the receive data register full (RXDF) bit is set. The host processor can program the RREQ bit to assert the external HREQ signal when RXDF is set. This indicates that the HI08 has a full word (either 8, 16, or 24 bits) for the host processor. The host processor can program the RREQ bit to assert the external HREQ signal when RXDF is set. Asserting the HREQ signal informs the host processor that the receive byte registers have data to be read. When the host reads the receive byte register at host address $7, the RXDF bit is cleared.

6.6.6 Transmit Byte Registers (TXH:TXM:TXL)

The host processor views the transmit byte registers as three 8-bit, write-only registers. These registers are the transmit high register (TXH), the transmit middle register (TXM), and the transmit low register (TXL). These registers send data to the high, middle, and low bytes, respectively, of the HRX register and are selected by the external host address inputs, HA[2:0], during a host processor write operation.

If the HLEND bit in the ICR is set, the TXH register is located at address $7, the TXM register at $6 and the TXL register at $5. If the HLEND bit in the ICR is cleared, the TXH register is located at address $5, the TXM register at $6, and the TXL register at $7.

Data is written into the transmit byte registers when the transmit data register empty (TXDE) bit is set. The host processor programs the TREQ bit to assert the external HREQ/HTRQ signal when TXDE is set. This informs the host processor that the transmit byte registers are empty. Writing to the data register at host address $7 clears the TXDE bit. The contents of the transmit byte registers are transferred as 24-bit data to the HRX register when both the TXDE and the HRDF bit are cleared. This transfer operation sets TXDE and HRDF.

Note: When data is written to a peripheral device, there is a two-cycle pipeline delay until any status bits affected by this operation are updated. If you read any of those status bits within the next two cycles, the bit does not reflect its current status. See the DSP56300 Family Manual, Appendix B, Polling a Peripheral Device for Write for further details.

6.6.7 Host Side Registers After Reset

Table 6-12 shows the result of the four kinds of reset on bits in each of the HI08 registers seen by the host processor. The hardware reset is caused by asserting the $\overline{\text{RESET}}$ signal. The software reset is caused by executing the RESET instruction. The individual reset is caused by clearing the HEN bit in the HPCR. The stop reset is caused by executing the STOP instruction.

Table 6-12 Host Side Registers After Reset

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	IR Reset	ST Reset
ICR	All Bits	0	0	—	—
CVR	HC	0	0	0	0
	HV[0:6]	$32	$32	—	—
ISR	HREQ	0	0	1 if TREQ is set; 0 otherwise	1 if TREQ is set; 0 otherwise
	HF3 -HF2	0	0	—	—
	TRDY	1	1	1	1
	TXDE	1	1	1	1
	RXDF	0	0	0	0
IVR	IV[0:7]	$0F	$0F	—	—
RX	RXH: RXM:RXL	empty	empty	empty	empty
TX	TXH: TXM:TXL	empty	empty	empty	empty

6.6.8 General-Purpose I/O

When configured as GPIO, the HI08 is viewed by the DSP56309 as memory-mapped registers, as documented in **Section 6.5—HI08 DSP Side Programmer's Model** on page 6-8. Those memory-mapped registers control up to 16 I/O signals. Software RESET instructions and hardware $\overline{\text{RESET}}$ signals clear all DSP side control registers and configure the HI08 as GPIO with all 16 signals disconnected. External circuitry

connected to the HI08 may need external pull-up/pull-down resistors until the signals are configured for operation. The registers cleared are the HPCR, HDDR, and HDR. Selection between GPIO and HI08 is made by clearing HPCR bits 6 through 1 for GPIO or setting these bits for HI08 functionality. If the HI08 is in GPIO mode, the HDDR configures each corresponding signal in the HDR as an input signal if the HDDR bit is cleared or as an output signal if the HDDR bit is set. (See **Section 6.5.7—Host Data Direction Register (HDDR)** on page 6-17 and **Section 6.5.8—Host Data Register (HDR)** also on page 6-17.)

6.7 SERVICING THE HOST INTERFACE

The HI08 can be serviced by using one of the following protocols:

- Polling
- Interrupts

The host processor writes to the appropriate HI08 register to reset the control bits and configure the HI08 for proper operation.

6.7.1 HI08 Host Processor Data Transfer

To the host processor, the HI08 looks like a contiguous block of Static RAM. To transfer data between itself and the HI08, the host processor performs the following steps:

1. asserts the HI08 address to select the register to be read or written
2. selects the direction of the data transfer
   (If it is writing, the host processor sources the data on the bus.)
3. strobes the data transfer

6.7.2 Polling

In polling mode, the HREQ/HTRQ signal is not connected to the host processor and HACK must be deasserted to insure IVR data is not being driven on H[7:0] when other registers are being polled. (If the HACK function is not needed, the HACK signal can be configured as a GPIO signal, as documented in **Section 6.5.6—Host Port Control Register (HPCR)** on page 6-12.)

The host processor first performs a data read transfer to read the ISR, as in **Figure 6-16**. This convention allows the host processor to assess the status of the HI08 and perform the appropriate actions.

Generally, after the appropriate data transfer has been made, the corresponding status bit is updated to reflect the transfer.

- If RXDF is set, the receive data register is full, and the host processor can perform a data read.
- If TXDE is set, the transmit data register is empty, and the host processor can perform a data write.
- If TRDY is set, the transmit data register is empty. This implies that the receive data register on the DSP side is also empty. Data written by the host processor to the HI08 is transferred directly to the DSP side.
- If (HF2 and HF3) ≠ 0, depending on how the host flags have been used, this may indicate that an application-specific state within the DSP56309 has been reached. Intervention by the host processor may be required.
- If HREQ is set, the HREQ/TRQ signal has been asserted, and the DSP56309 is requesting the attention of the host processor. One of the previous four conditions exists.

After the appropriate data transfer has been made, the corresponding status bit is updated to reflect the transfer.

If the host processor has issued a command to the DSP56309 by writing to the CVR and setting the HC bit, it can read the HC bit in the CVR to determine whether the command has been accepted by the interrupt controller in the DSP core. When the command has been accepted for execution, the HC bit is cleared by the interrupt controller in the DSP core.

6.7.3 Servicing Interrupts

If either HREQ/HTRQ or the HRRQ signal or both are connected to the host processor's interrupt input, the HI08 can request service from the host processor by asserting one of these signals. The HREQ/HTRQ and/or the HRRQ signal is asserted when TXDE is set and/or RXDF is set and the corresponding enable bit (TREQ or RREQ, respectively) is set. This situation appears in **Figure 6-16**.

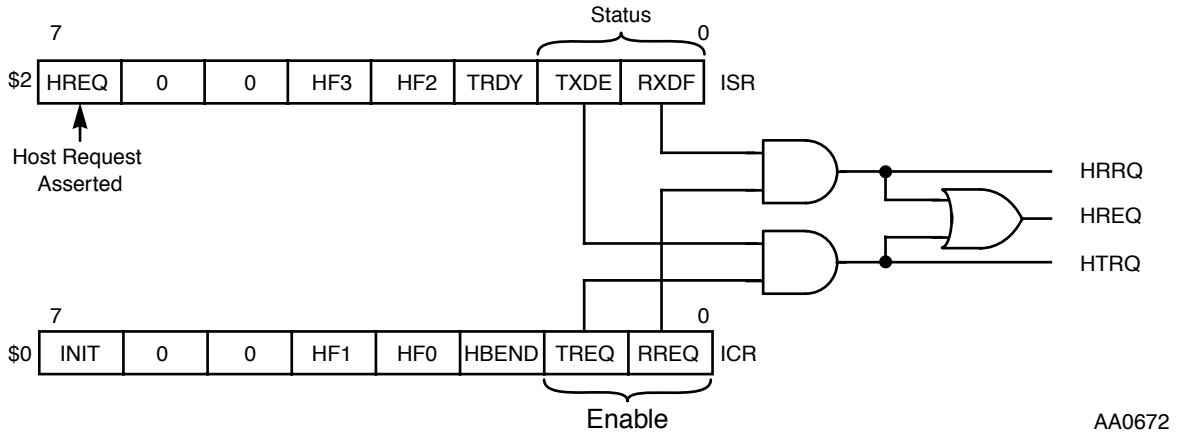


Figure 6-16 HI08 Host Request Structure

HREQ is normally connected to the maskable interrupt input of the host processor. The host processor acknowledges host interrupts by executing an interrupt service routine. The host processor can test the two LSBs (RXDF and TXDE) of the ISR register to determine the interrupt source, as in **Figure 6-16**. The host processor interrupt service routine must read or write the appropriate HI08 data register to clear the interrupt. HREQ/HTRQ and/or HRRQ is deasserted under either of the following conditions:

- The enabled request is cleared or masked.
- The DSP is reset.

If the host processor is a member of the MC68000 family, there is no need for the additional step when the host processor reads the ISR to determine how to respond to an interrupt generated by the DSP56309. Instead, the DSP56309 automatically sources the contents of the IVR on the data bus when the host processor acknowledges the interrupt by asserting HACK. The contents of the IVR are placed on the host data bus while HREQ/TRQ (or HRRQ) and HACK are simultaneously asserted. The IVR data tells the MC680XX host processor which interrupt routine to execute to service the DSP56309.

Table 6-13 shows the HI08 programming model.

6.8 HI08 PROGRAMMING MODEL QUICK REFERENCE

Table 6-13 HI08 Programming Model

Reg	Bit					Comments	Reset Type		
	#	Mnemonic	Name	Value	Function		HW/ SW	I R	S T
DSP SIDE									
HCR	0	HRIE	Receive Interrupt Enable	0 1	HRRQ interrupt disabled HRRQ interrupt enabled	—	0	—	—
	1	HTIE	Transmit Interrupt Enable	0 1	HTRQ interrupt disabled HTRQ interrupt enabled	—	0	—	—
	2	HCIE	Host Command Interrupt Enable	0 1	HCP interrupt disabled HCP interrupt enabled	—	0	—	—
	3	HF2	Host Flag 2	—	—	—	0	—	—
	4	HF3	Host Flag 3	—	—	—	0	—	—
HPCR	0	HGEN	Host GPIO Enable	0 1	GPIO signal disconnected GPIO signals active	—	0	—	—
	1	HA8EN	Host Address Line 8 Enable	0 1	HA8/A1 = GPIO HA8/A1 = HA8	This bit is treated as 1 if HMUX = 0. This bit is treated as 0 if HEN = 0.	0	—	—
	2	HA9EN	Host Address Line 9 Enable	0 1	HA9/A2 = GPIO HA9/A2 = HA9	This bit is treated as 1 if HMUX = 0. This bit is treated as 0 if HEN = 0.	0	—	—
	3	HCSEN	Host Chip Select Enable	0 1	HCS/A10 = GPIO HCS/A10 = HCS	This bit is treated as 0 if HEN = 0.	0	—	—

Table 6-13 HI08 Programming Model (Continued)

Reg	Bit					Comments	Reset Type		
	#	Mnemonic	Name	Value	Function		HW/ SW	I R	S T
HPCR	4	HREN	Host Request Enable	0 1	HDRQ = 0 HDRQ = 1 HREQ/HTRQ = GPIO HREQ/HTRQ HACK/HRRQ = GPIO HREQ/HTRQ = HREQ,HREQ/HTRQ HACK/HRRQ = HTRQ, HRRQ	—	0	—	—
	5	HAEN	Host Acknowledge Enable	0 1	HDRQ = 0 HDRQ=1 HACK/HRRQ = GPIO HREQ/HTRQ HACK/HRRQ = GPIO HACK/HRRQ = HACK HREQ/HTRQ HACK/HRRQ = HTRQ, HRRQ	This bit is ignored if HDRQ = 1. This bit is treated as 0 if HREN = 0. This bit is treated as 0 if HEN = 0.	0	—	—
	6	HEN	Host Enable	0 1	Host Port = GPIO Host Port Active	—	0	—	—
	8	HROD	Host Request Open Drain	0 1	HREQ/HTRQ/HRRQ = driven HREQ/HTRQ/HRRQ = open drain	This bit is ignored if HEN = 0.	0		
	9	HDSP	Host Data Strobe Polarity	0 1	HDS/HRD/HWR active low HDS/HRD/HWR active high	This bit is ignored if HEN = 0.	0	—	—
	10	HASP	Host Address Strobe Polarity	0 1	HAS active low HAS active high	This bit is ignored if HEN = 0.	0	—	—
	11	HMUX	Host Multiplexed Bus	0 1	Separate address and data lines Multiplexed address/data	This bit is ignored if HEN = 0.	0	—	—
	12	HDDS	Host Dual Data Strobe	0 1	Single Data Strobe (HDS) Double Data Strobe (HWR, HRD)	This bit is ignored if HEN = 0.	0	—	—

Table 6-13 HI08 Programming Model (Continued)

Reg	Bit					Comments	Reset Type		
	#	Mnemonic	Name	Value	Function		HW/ SW	I R	S T
HPCR	13	HCSP	Host Chip Select Polarity	0 1	HCS active low HCS active high	This bit is ignored if HEN = 0.	0	—	—
	14	HRP	Host Request polarity	0 1	HREQ/HTRQ/HRRQ active low HREQ/HTRQ/HRRQ active high	This bit is ignored if HEN = 0.	0	—	—
	15	HAP	Host Acknowledge Polarity	0 1	HACK active low HACK active high	This bit is ignored if HEN = 0.	0	—	—
HSR	0	HRDF	Host Receive Data Full	0 1	no receive data to be read Receive Data Register is full	—	0	0	0
	1	HTDE	Host Transmit Data Empty	1 0	The Transmit Data Register is empty. The Transmit Data Register is not empty.	—	1	1	1
	2	HCP	Host Command Pending	0 1	no host command pending host command pending	—	0	0	0
	3	HF0	Host Flag 0	—	—	—	0	—	—
	4	HF1	Host Flag 1	—	—	—	0	—	—
HBAR	7-0	BA10-BA3	Host Base Address Register	—	—	—	$80	—	—
HRX	23-0	—	DSP Receive Data Register	—	—	—	empty	—	—
HTX	23-0	—	DSP Transmit Data Register	—	—	—	empty	—	—
HDR	16-0	D16-D0	GPIO signal Data	—	—	—	$0000	—	—
HDRR	16-0	DR16-DR0	GPIO signal Direction	[0] [1]	Input Output	—	$0000	—	—

Table 6-13 HI08 Programming Model (Continued)

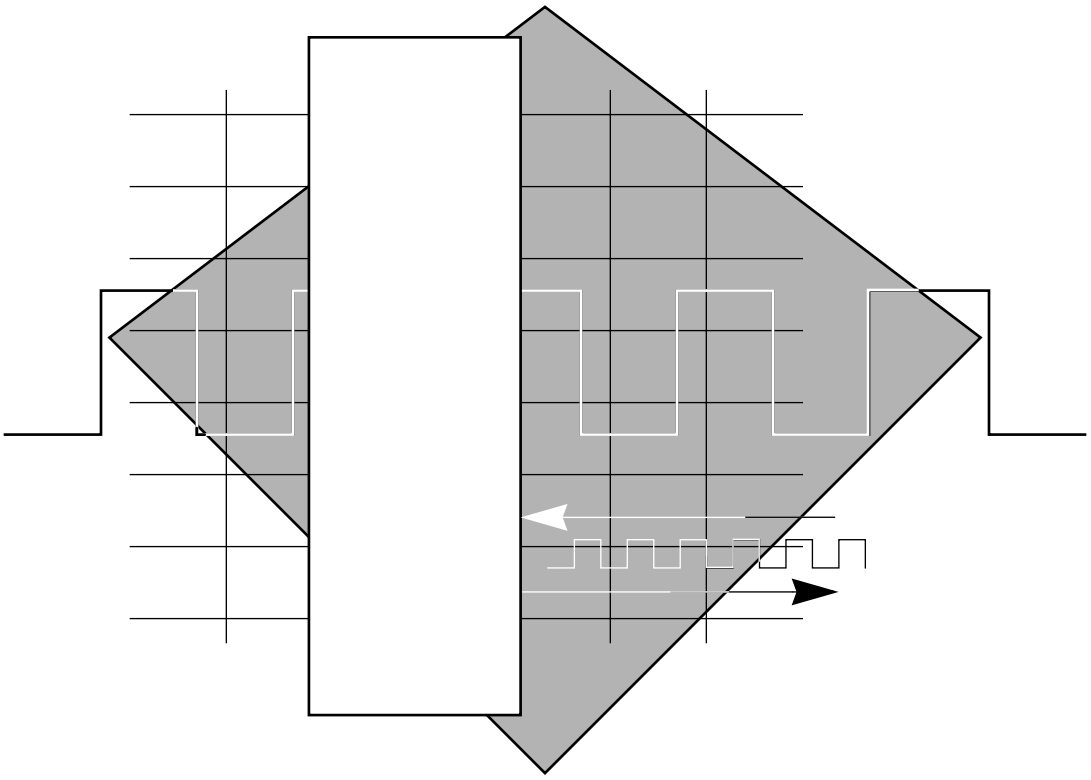
Reg	Bit					Comments	Reset Type		
	#	Mnemonic	Name	Value	Function		HW/ SW	I R	S T
HOST SIDE									
ICR	0	RREQ	Receive Request Enable	0 1	HRRQ interrupt disabled HRRQ interrupt enabled	—	0	—	—
	1	TREQ	Transmit Request Enable	0 1	HTRQ interrupt disabled HTRQ interrupt enabled	—	0	—	—
	2	HDRQ	Double Host Request	0 1	HREQ/HTRQ = HREQ, HACK/HRRQ = HACK HREQ/HTRQ = HTRQ, HACK/HRRQ = HRRQ	—	0	—	—
	3	HF0	Host Flag 0	—	—	—	0	—	—
	4	HF1	Host Flag 1	—	—	—	0	—	—
	5	HLEND	Host Little Endian	0 1	Big Endian order Little Endian order	—	0	—	—
	7	INIT	Initialize	1	Reset data paths according to TREQ and RREQ	cleared by HI08 hardware	0	—	—

Table 6-13 HI08 Programming Model (Continued)

Reg	Bit					Comments	Reset Type		
	#	Mnemonic	Name	Value	Function		HW/ SW	I R	S T
ISR	0	RXDF	Receive Data Register Full	0 1	Host Receive Register is empty Host Receive Register is full	—	0	0	0
	1	TXDE	Transmit Data Register Empty	1 0	Host Transmit Register is empty Host Transmit Register is full	—	1	1	1
	2	TRDY	Transmitter Ready	1 0	transmit FIFO (6 deep) is empty transmit FIFO is not empty	—	1	1	1
	3	HF2	Host Flag 2	—	—	—	0	—	—
	4	HF3	Host Flag 3	—	—	—	0	—	—
	7	HREQ	Host Request	0 1	$\overline{\text{HREQ}}$ signal is deasserted $\overline{\text{HREQ}}$ signal is asserted (if enabled)	—	0	0	0
CVR	6-0	HV6-HV0	Host Command Vector	—	—	default vector via programmable	$32	—	—
CVR	7	HC	Host Command	0 1	no host command pending host command pending	cleared by HI08 hardware when the HC interrupt request is serviced	0	0	0
RXH/M/L	7-0	—	Host Receive Data Register	—	—	—	empty	—	—
TXH/M/L	7-0	—	Host Transmit Data Register	—	—	—	empty	—	—
IVR	7-0	IV7-IV0	Interrupt Register	—	68000 family vector register	—	$0F	—	—

SECTION 7

ENHANCED SYNCHRONOUS SERIAL INTERFACE (ESSI)



7.1 INTRODUCTION

The Enhanced Synchronous Serial Interface (ESSI) provides a full-duplex serial port for serial communication with a variety of serial devices, including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals that implement the Motorola Serial Peripheral Interface (SPI). The ESSI consists of independent transmitter and receiver sections and a common ESSI clock generator.

There are two independent and identical ESSIs in the DSP56309: ESSI0 and ESSI1. For the sake of simplicity, a single generic ESSI is described.

The ESSI block diagram appears in **Figure 7-1** on page 7-5. This interface is synchronous because all serial transfers are synchronized to a clock.

Note: This should not be confused with the asynchronous mode of the ESSI, in which separate clocks are used for the receiver and transmitter. In this mode, the ESSI is still a synchronous device, because all transfers are synchronized to these clocks.

Additional synchronization signals are used to delineate the word frames. Normal mode is used to transfer data at a periodic rate, one word per period. Network mode is similar in that it is also intended for periodic transfers; however, it supports up to 32 words (time slots) per period. Network mode can be used to build time division multiplexed (TDM) networks. In contrast, on-demand mode is intended for non-periodic transfers of data. This mode can be used to transfer data serially at high speed when the data become available. This mode offers a subset of the SPI protocol.

Since each ESSI unit can be configured with one receiver and three transmitters, the two units can be used together for surround sound applications (which need two digital input channels and six digital output channels).

7.2 ENHANCEMENTS TO THE ESSI

The synchronous serial interface (SSI) used in the DSP56000 family has been enhanced in the following ways to make the ESSI:

- Network enhancements
  - Time slot mask registers (receive and transmit) added
  - End-of-frame interrupt added
  - Drive enable signal added (to be used with transmitter 0)

- Audio enhancements
  - Three transmitters per ESSI (for six-channel surround sound)
- General enhancements
  - Can trigger DMA interrupts (receive or transmit)
  - Separate exception enable bits
- Other Changes
  - One divide by 2 removed from the internal clock source chain
  - CRA (PSR) bit definition is reversed
  - Gated clock mode not available

7.3 ESSI DATA AND CONTROL SIGNALS

Three to six signals are required for ESSI operation, depending on the operating mode selected. The serial transmit data (STD) signal and serial control (SC0 and SC1) signals are fully synchronized to the clock if they are programmed as transmit-data signals.

7.3.1 Serial Transmit Data (STD) Signal

The STD signal is used for transmitting data from the TX0 serial transmit shift register. STD is an output when data is being transmitted from the TX0 shift register. With an internally generated bit clock, the STD signal becomes a high impedance output signal for a full clock period after the last data bit has been transmitted. If sequential data words are being transmitted, the STD signal does not assume a high-impedance state. The STD signal can be programmed as a GPIO signal (P5) when the ESSI STD function is not being used.

7.3.2 Serial Receive Data Signal (SRD)

The SRD signal receives serial data and transfers the data to the ESSI receive shift register. SRD can be programmed as a GPIO signal (P4) when the ESSI SRD function is not being used.

The ESSI block diagram is shown in **Figure 7-1**.

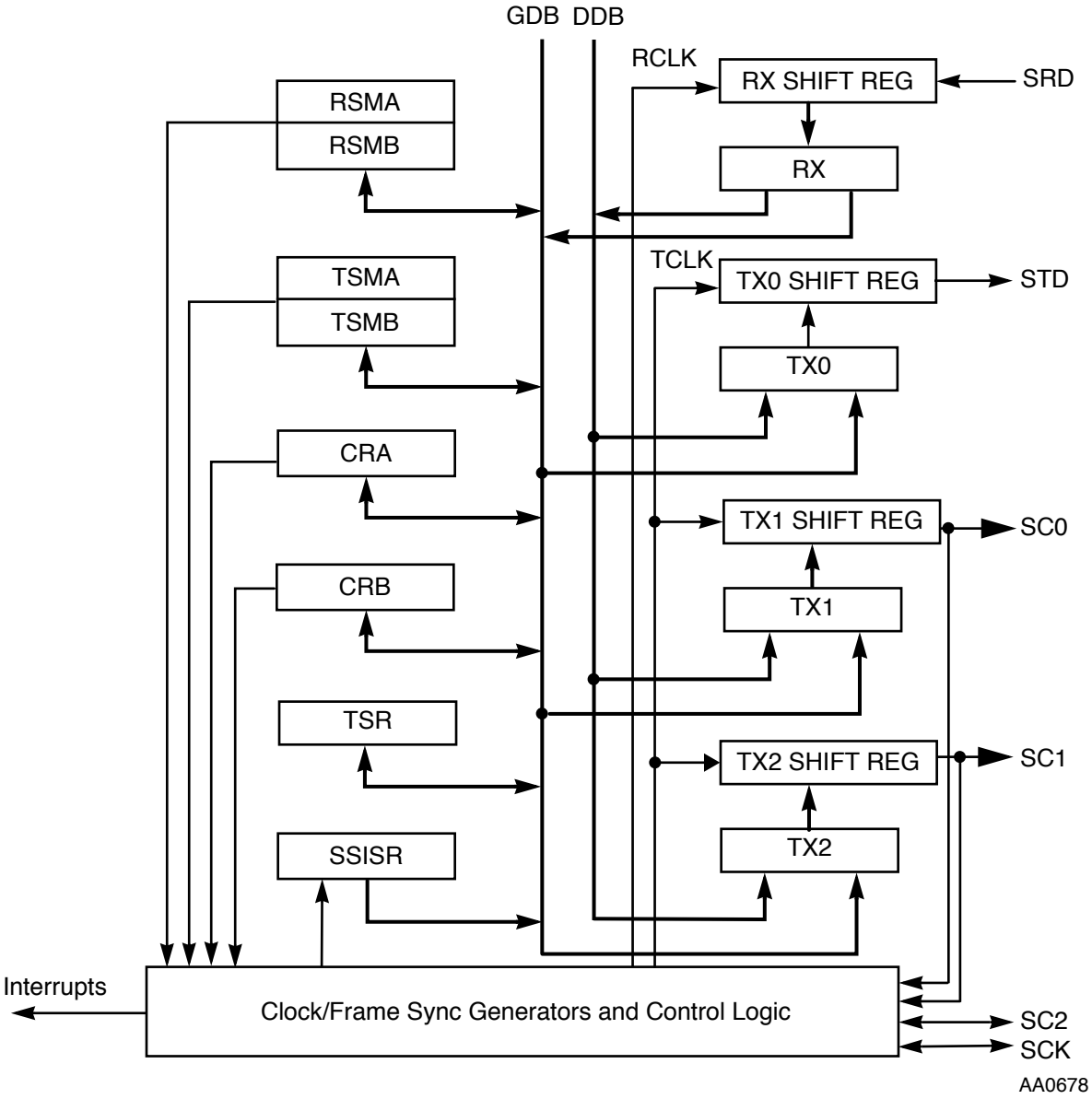


Figure 7-1 ESSI Block Diagram

7.3.3 Serial Clock (SCK)

The SCK signal is a bidirectional signal providing the serial bit rate clock for the ESSI interface. The SCK signal is a clock input or output used by all the enabled transmitters and receiver in synchronous modes or by all the enabled transmitters in asynchronous

modes; see **Table 7-1** on page 7-8. SCK can be programmed as a GPIO signal (P3) when the ESSI SCK function is not being used.

Notes:
1. Although an external serial clock can be independent of and asynchronous to the DSP system clock, the external ESSI clock frequency must not exceed $F_{core}/3$, and each ESSI phase must exceed the minimum of 1.5 CLKOUT cycles.
2. The internally sourced ESSI clock frequency must not exceed $F_{core}/4$.

7.3.4 Serial Control Signal (SC0)

SC00 is a serial control signal for ESSI0, and SC10 is a serial control signal for ESSI1. They are referred to collectively as SC0.

The function of this signal is determined by selecting either synchronous or asynchronous mode; see **Table 7-4** on page 7-24. In asynchronous mode, this signal is used for the receive clock I/O. In synchronous mode, this signal is used as the transmitter data out signal for transmit shift register 1 or for serial flag I/O. A typical application of serial flag I/O would be multiple device selection for addressing in codec systems.

If SC0 is configured as a serial flag signal, its direction is determined by the serial control direction 0 (SCD0) bit in the ESSI control register B (CRB). When configured as an output, its value is determined by the value of the serial output flag 0 (OF0) bit in the CRB. When configured as an input, SC0 controls the state of serial input flag 0 (IF0) bit in the ESSI status register (SSISR).

When SC0 is configured as a transmit data signal, it is always an output signal regardless of the SCD0 bit value. SC0 is fully synchronized with the other transmit data signals (STD and SC1).

In asynchronous mode, SC0 is configured as the receive clock. The direction of the SC0 in this mode is also determined by SCD0.

SC0 can be programmed as a GPIO signal (P0) when the ESSI SC0 function is not being used.

Note: The ESSI can operate with more than one active transmitter only in synchronous mode.

7.3.5 Serial Control Signal (SC1)

SC01 is a serial control signal for ESSI0, and SC11 is a serial control signal for ESSI1. They are referred to collectively as SC1.

The function of this signal is determined by selecting either synchronous or asynchronous mode; see **Table 7-4** on page 7-24. In asynchronous mode (such as a single codec with asynchronous transmit and receive), SC1 is the receiver frame sync I/O. In synchronous mode, SC1 is used for the transmitter data out signal of transmit shift register TX2, for the drive enable transmitter 0 signal, or for serial flag SC1.

When used as a serial flag signal, SC1 operates like the previously described SC0. SC0 and SC1 are independent flags but can be used together for multiple serial device selection. SC0 and SC1 can be used unencoded to select up to two codecs or can be decoded externally to select up to four codecs.

If SC1 is configured as a serial flag signal, its direction is determined by the SCD1 bit in the CRB. When configured as an output, its value is determined by the value of the serial output flag1 (OF1) bit in the CRB. When configured as an input, SC0 controls the stated serial input flag 1 (IF1) bits in SSISR.

When SC1 is configured as a transmit data signal, it is always an output signal regardless of the SCD1 bit value. As an output, it is fully synchronized with the other ESSI transmit data signals (STD and SC0).

In asynchronous mode, SC1 is configured as the receive frame sync. The direction of SC1 in this mode is determined by SCD1.

SC1 can be programmed as a GPIO signal (P1) when the ESSI SC1 function is not being used.

Table 7-1 summarizes ESSI clock sources, whether synchronous or asynchronous, and shows the bit settings for the signals involved.

Table 7-1 ESSI Clock Sources

SYN	SCKD	SCD0	R Clock Source	RX Clock Out	T Clock Source	TX Clock Out
Asynchronous						
0	0	0	EXT, SC0	—	EXT, SCK	—
0	0	1	INT	SC0	EXT, SCK	—
0	1	0	EXT, SC0	—	INT	SCK
0	1	1	INT	SC0	INT	SCK
Synchronous						
1	0	0/1	EXT, SCK	—	EXT, SCK	—
1	1	0/1	INT	SCK	INT	SCK

7.3.6 Serial Control Signal (SC2)

SC02 is a serial control signal for ESSI0, and SC12 is a serial control signal for ESSI1. They are referred to collectively as SC2.

This signal is used for frame sync I/O. SC2 is the frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode. The direction of this signal is determined by the SCD2 bit in the CRB. When configured as an output, this signal outputs the internally generated frame sync signal. When configured as an input, this signal receives an external frame sync signal for the transmitter in asynchronous mode and for the receiver when in synchronous mode. SC2 can be programmed as a GPIO signal (P2) when the ESSI SC2 function is not being used.

7.4 ESSI PROGRAMMING MODEL

The ESSI includes the following registers:

- Two control registers (CRA, CRB) illustrated in **Figure 7-2** and **Figure 7-3**
- One status register (SSISR) illustrated in **Figure 7-4**

- Three transmit data registers (TX0, TX1, TX2)
- One receive data register (RX)
- Two transmit slot mask registers (TSMA, TSMB) illustrated in **Figure 7-5** and **Figure 7-6**
- Two receive slot mask registers (RSMA, RSMB) illustrated in **Figure 7-7** and **Figure 7-8**
- One special-purpose time slot register (TSR)

The following paragraphs give detailed descriptions and operations of each of the bits in the ESSI registers. The GPIO functionality of the ESSI is documented in **Section 7.6—GPIO Signals and Registers** of this manual.

11	10	9	8	7	6	5	4	3	2	1	0
PSR				PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

23	22	21	20	19	18	17	16	15	14	13	12
	SSC1	WL2	WL1	WL0	ALC		DC4	DC3	DC2	DC1	DC0

AA0857

Figure 7-2 ESSI Control Register A (CRA)

(ESSI0 X:$FFFFB5, ESSI1 X:$FFFFA5)

11	10	9	8	7	6	5	4	3	2	1	0
CKP	FSP	FSR	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0

23	22	21	20	19	18	17	16	15	14	13	12
REIE	TEIE	RLIE	TLIE	RIE	TIE	RE	TE0	TE1	TE2	MOD	SYN

AA0858

Figure 7-3 ESSI Control Register B (CRB)

(ESSI0 X:$FFFFB6, ESSI1 X:$FFFFA6)

11	10	9	8	7	6	5	4	3	2	1	0
				RDF	TDE	ROE	TUE	RFS	TFS	IF1	IF0

23	22	21	20	19	18	17	16	15	14	13	12

AA0859

Figure 7-4 ESSI Status Register (SSISR)

(ESSI0 X:$FFFFB7, ESSI1 X:$FFFFA7)

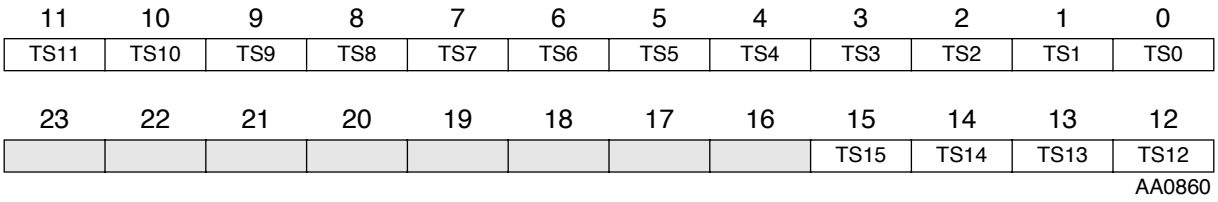


Figure 7-5 ESSI Transmit Slot Mask Register A (TSMA)

(ESSI0 X:$FFFFB4, ESSI1 X:$FFFFA4)

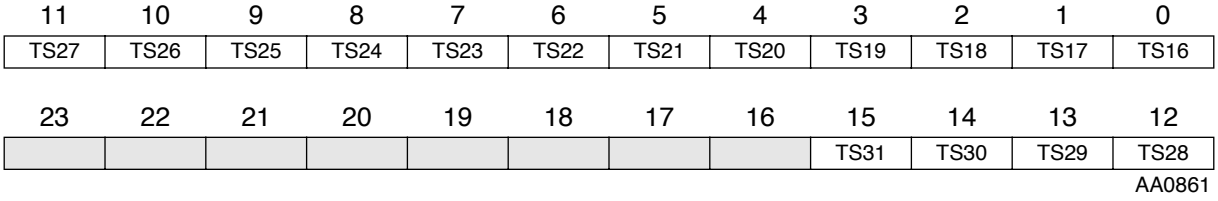


Figure 7-6 ESSI Transmit Slot Mask Register B (TSMB)

(ESSI0 X:$FFFFB3, ESSI1 X:$FFFFA3)

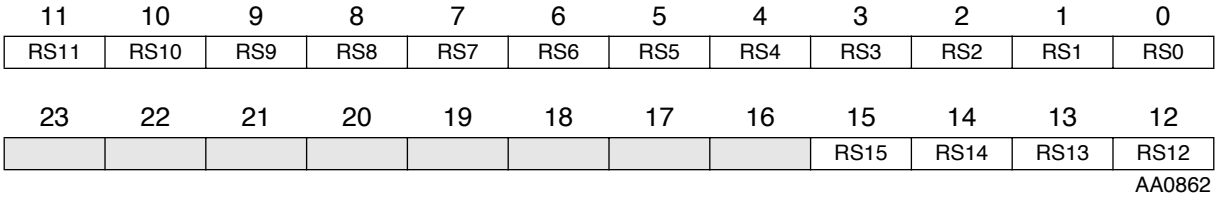


Figure 7-7 ESSI Receive Slot Mask Register A (RSMA)

(ESSI0 X:$FFFFB2, ESSI1 X:$FFFFA2)

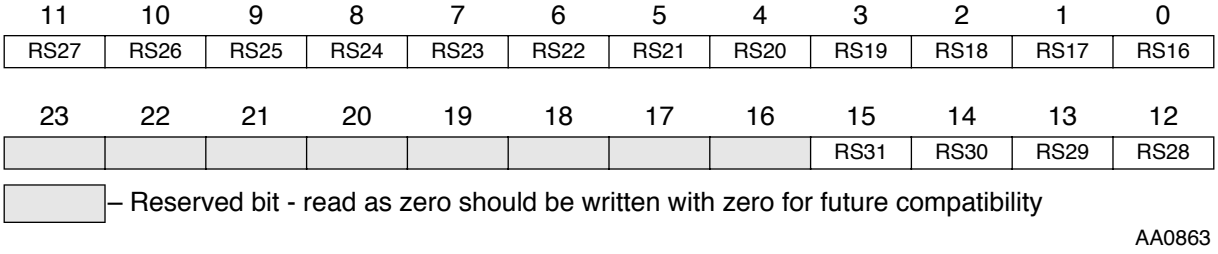


Figure 7-8 ESSI Receive Slot Mask Register B (RSMB)

(ESSI0 X:$FFFFB1, ESSI1 X:$FFFFA1)

7.4.1 ESSI Control Register A (CRA)

The ESSI control register A (CRA) is one of two 24-bit, read/write control registers used to direct the operation of the ESSI. The CRA controls the ESSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The CRA control bits are described in the following paragraphs; see also **Figure 7-2** on page 7-9.

7.4.1.1 CRA Prescale Modulus Select PM[7:0] Bits 7–0

The PM[7:0] bits specify the divide ratio of the prescale divider in the ESSI clock generator. A divide ratio from 1 to 256 (PM = $0 to $FF) can be selected. The bit clock output is available at the transmit clock signal (SCK) and/or the receive clock (SC0) signal of the DSP. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. The ESSI clock generator functional diagram is shown in **Figure 7-9**. F_{core} is the DSP56309 core clock frequency (the same frequency as the CLKOUT signal, when that signal is enabled). Careful choice of the crystal oscillator frequency and the prescaler modulus allows generation of the industry-standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz. Both the hardware $\overline{\text{RESET}}$ signal and the software RESET instruction clear PM[7:0].

7.4.1.2 CRA Reserved Bits 8–10

These bits are reserved. They are read as 0 and should be written with 0.

7.4.1.3 CRA Prescaler Range (PSR) Bit 11

The PSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When PSR is set, the fixed prescaler is bypassed. When PSR is cleared, the fixed divide-by-eight prescaler is operational; see **Figure 7-9** on page 7-12.

Note: This definition is reversed from that of the 560xx SSI.

The maximum allowed internally generated bit clock frequency is the internal DSP56309 clock frequency divided by 4; the minimum possible internally generated bit clock frequency is the DSP56309 internal clock frequency divided by 4096. Both the hardware $\overline{\text{RESET}}$ signal and the software RESET instruction clear PSR.

Note: The combination PSR = 1 and PM[7:0] = $00 (dividing F_{core} by 2) can cause synchronization problems and should not be used.

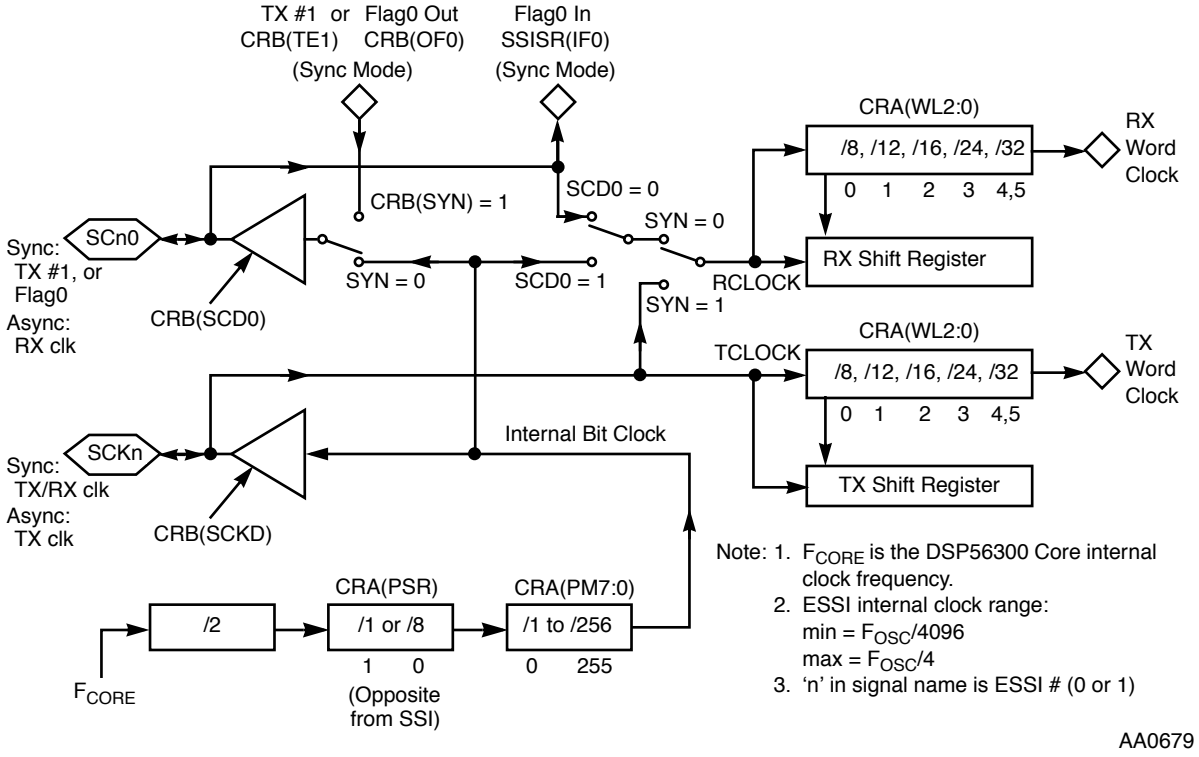


Figure 7-9 ESSI Clock Generator Functional Block Diagram

7.4.1.4 CRA Frame Rate Divider Control DC[4:0] Bits 16–12

The values of the DC[4:0] bits control the divide ratio for the programmable frame rate dividers used to generate the frame clocks. In network mode, this ratio can be interpreted as the number of words per frame minus one. In normal mode, this ratio determines the word transfer rate.

The divide ratio can range from 1 to 32 (DC = 00000 to 11111) for normal mode and 2 to 32 (DC = 00001 to 11111) for network mode. A divide ratio of one (DC = 00000) in network mode is a special case known as on-demand mode. In normal mode, a divide ratio of one (DC = 00000) provides continuous periodic data word transfers. A bit-length frame sync must be used in this case and is selected by setting the FSL[1:0] bits in the CRA to (01). Both the hardware $\overline{\text{RESET}}$ signal and the software RESET instruction clear DC[4:0].

The ESSI frame sync generator functional diagram is shown in **Figure 7-10**.

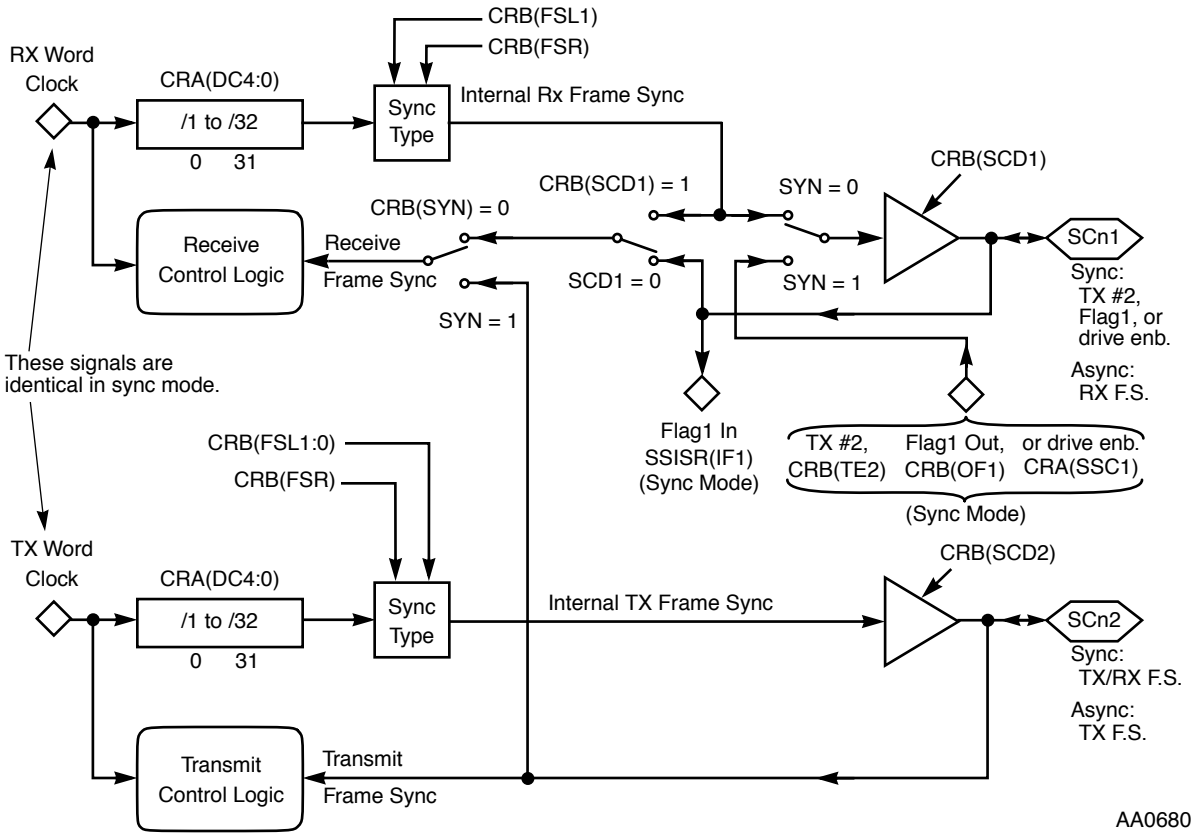


Figure 7-10 ESSI Frame Sync Generator Functional Block Diagram

7.4.1.5 CRA Reserved Bit 17

This bit is reserved. It is read as 0 and should be written with 0.

7.4.1.6 CRA Alignment Control (ALC) Bit 18

The ESSI is designed for 24-bit fractional data. Shorter data words are left aligned to the MSB, Bit 23. For applications that use 16 bit fractional data, shorter data words are left aligned to bit 15. The ALC bit supports shorter data words. If ALC is set, received words are left aligned to bit 15 in the receive shift register. Transmitted words must be left aligned to bit 15 in the transmit shift register. If the ALC bit is cleared, received words are left aligned to bit 23 in the receive shift register. Transmitted words must be left aligned to bit 23 in the transmit shift register. The ALC bit is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction**.**

Note: If the ALC bit is set, only 8-, 12-, or 16-bit words should be used. The use of 24- or 32-bit words leads to unpredictable results.

7.4.1.7 CRA Word-length Control (WL[2:0]) Bits 21–19

The WL[2:0] bits are used to select the length of the data words being transferred via the ESSI. Word lengths of 8-, 12-, 16-, 24-, or 32- bits can be selected, as in **Table 7-2**. The ESSI data path programming model in **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32 has additional information about selecting different length data words. The ESSI data registers are 24 bits long. The ESSI transmits 32-bit words either by duplicating the last bit eight times when WL[2:0] = 100, or by duplicating the first bit eight times when WL[2:0] = 101. The WL[2:0] bits are cleared by a hardware $\overline{\text{RESET}}$ signal or by a software RESET instruction.

Table 7-2 ESSI Word Length Selection

WL2	WL1	WL0	Number of Bits/Word
0	0	0	8
0	0	1	12
0	1	0	16
0	1	1	24
1	0	0	32 (valid data in the first 24 bits)
1	0	1	32 (valid data in the last 24 bits)
1	1	0	Reserved
1	1	1	Reserved

7.4.1.8 CRA Select SC1 (SSC1) Bit 22

The SSC1 bit controls the functionality of the SC1 signal. This bit is only valid when the ESSI is configured in synchronous mode (i.e., if the CRB synchronous/asynchronous bit (SYN) is set), and transmitter 2 is disabled (i.e., if transmit enable (TE2) = 0). If SSC1 is set and SC1 is configured as an output (SCD1 = 1), then the SC1 signal acts as the driver enabled signal of transmitter 0. This enables an external buffer for the transmitter 0 output.

If SSC1 is cleared, SC1 acts as the serial I/O flag.

7.4.1.9 CRA Reserved Bit 23

This bit is reserved. It is read as 0 and should be written with 0.

7.4.2 ESSI Control Register B (CRB)

The CRB is one of two 24-bit, read/write control registers used to direct the operation of the ESSI; see **Figure 7-3** on page 7-9. CRB controls the ESSI multifunction signals, SC[2:0], which can be used as clock inputs or outputs, frame synchronization signals, transmit data signals, or serial I/O flag signals.

The serial output flag control bits and the direction control bits for the serial control signals are in the ESSI CRB. Interrupt enable bits for the receiver and the transmitter are also in the CRB. The bit setting of the CRB also determines how many transmitters are enabled (0, 1, 2, or 3 transmitters can be enabled). The CRB settings also determine the ESSI operating mode.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all the bits in the CRB.

The relationship between the ESSI signals SC[2:0], SCK, and the CRB bits is summarized in **Table 7-4** on page 7-24. The ESSI CRB bits are described in the following paragraphs.

7.4.2.1 CRB Serial Output Flags (OF0, OF1) Bits 0, 1

The ESSI has two serial output flag bits, OF1 and OF0. The normal sequence for setting output flags when transmitting data (by transmitter 0 through the STD signal only) consists of these steps:

1. Wait for TDE (TX0 empty) to be set.
2. Write the flags.
3. Write the transmit data to the TX register.

Bits OF0 and OF1 are double-buffered so that the flag states appear on the signals when the TX data is transferred to the transmit shift register. The flag bits values are synchronized with the data transfer.

Note: The timing of the optional serial output signals SC[2:0] is controlled by the frame timing and is not affected by the settings of TE2, TE1, TE0, or the receive enable (RE) bit of the CRB.

7.4.2.1.1 CRB Serial Output Flag 0 (OF0) Bit 0

When the ESSI is in synchronous mode and transmitter 1 is disabled (TE1 = 0), the SC0 signal is configured as ESSI flag 0. If the serial control direction bit (SCD0) is set, the SC0 signal is an output. Data present in bit OF0 is written to SC0 at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.

Bit OF0 is cleared by a hardware $\overline{\text{RESET}}$ signal or by a software RESET instruction.

7.4.2.1.2 CRB Serial Output Flag 1 (OF1) Bit 1

When the ESSI is in synchronous mode and transmitter 2 is disabled (TE2 = 0), the SC1 signal is configured as ESSI flag 1. If the serial control direction bit (SCD1) is set, the SC1 signal is an output. Data present in bit OF1 is written to SC1 at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.

Bit OF1 is cleared by a hardware $\overline{\text{RESET}}$ signal or by a software RESET instruction.

7.4.2.2 CRB Serial Control Direction 0 (SCD0) Bit 2

In synchronous mode (SYN = 1) when transmitter 1 is disabled (TE1 = 0), or in asynchronous mode (SYN = 0), SCD0 controls the direction of the SC0 I/O signal. When SCD0 is set, SC0 is an output; when SCD0 is cleared, SC0 is an input.

When TE1 is set, the value of SCD0 is ignored, and the SC0 signal is always an output.

Bit SCD0 is cleared by a hardware $\overline{\text{RESET}}$ signal or by a software RESET instruction.

7.4.2.3 CRB Serial Control Direction 1 (SCD1) Bit 3

In synchronous mode (SYN = 1) when transmitter 2 is disabled (TE2 = 0), or in asynchronous mode (SYN = 0), SCD1 controls the direction of the SC1 I/O signal. When SCD1 is set, SC1 is an output; when SCD1 is cleared, SC1 is an input.

When TE2 is set, the value of SCD1 is ignored, and the SC1 signal is always an output. Bit SCD1 is cleared by a hardware $\overline{\text{RESET}}$ signal or by a software RESET instruction.

7.4.2.4 CRB Serial Control Direction 2 (SCD2) Bit 4

SCD2 controls the direction of the SC2 I/O signal. When SCD2 is set, SC2 is an output; when SCD2 is cleared, SC2 is an input. SCD2 is cleared by a hardware $\overline{\text{RESET}}$ signal or by a software RESET instruction.

7.4.2.5 CRB Clock Source Direction (SCKD) Bit 5

SCKD selects the source of the clock signal. If SCKD is set and the ESSI is in synchronous mode, the internal clock is the source of the clock signal used for all the transmit shift registers and the receive shift register. If SCKD is set and the ESSI is in asynchronous mode, the internal clock source becomes the bit clock for the transmit shift register and word length divider. The internal clock is output on the SCK signal.

When SCKD is cleared, the external clock source is selected. The internal clock generator is disconnected from the SCK signal, and an external clock source can drive this signal.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears SCKD.

7.4.2.6 CRB Shift Direction (SHFD) Bit 6

The setting of the SHFD bit determines the shift direction of the transmit or receive shift register. If SHFD is set, data is shifted out with the LSB first. If SHFD is cleared, data is shifted out MSB first; see **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32. Received data is shifted in LSB first when SHFD is set or MSB first when SHFD is cleared.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears SHFD.

7.4.2.7 CRB Frame Sync Length FSL[1:0] Bits 7 and 8

These bits select the length of frame sync to be generated or recognized; see **Figure 7-11** on page 7-19, **Figure 7-14** on page 7-22, and **Figure 7-15** on page 7-23. The values of FSL[1:0] are documented in **Table 7-3**.

Table 7-3 FSL1 and FSL0 Encoding

FSL1	FSL0	Frame Sync Length	
		RX	TX
0	0	word	word
0	1	word	bit
1	0	bit	bit
1	1	bit	word

The word length is defined by WL[2:0].

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears FSL[1:0].

7.4.2.8 CRB Frame Sync Relative Timing (FSR) Bit 9

The FSR bit determines the relative timing of the receive and transmit frame sync signal in reference to the serial data lines, for word length frame sync only. When FSR is cleared, the word length frame sync occurs together with the first bit of the data word of the first slot. When FSR is set, the word length frame sync occurs one serial clock cycle earlier (i.e., simultaneously with the last bit of the previous data word).

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears FSR.

7.4.2.9 CRB Frame Sync Polarity (FSP) Bit 10

The FSP bit determines the polarity of the receive and transmit frame sync signals. When FSP is cleared, the frame sync signal polarity is positive (i.e., the frame start is indicated

by the frame sync signal going high). When FSP is set, the frame sync signal polarity is negative (i.e., the frame start is indicated by the frame sync signal going low).

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears FRB.

7.4.2.10 CRB Clock Polarity (CKP) Bit 11

The CKP bit controls on which bit clock edge data and frame sync are clocked out and latched in. If CKP is cleared, the data and the frame sync are clocked out on the rising edge of the transmit bit clock and latched in on the falling edge of the receive bit clock. If CKP is set, the data and the frame sync are clocked out on the falling edge of the transmit bit clock and latched in on the rising edge of the receive bit clock.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears CKP.

7.4.2.11 CRB Synchronous /Asynchronous (SYN) Bit 12

SYN controls whether the receive and transmit functions of the ESSI occur synchronously or asynchronously with respect to each other; see **Figure 7-12** on page 7-20. When SYN is cleared, the ESSI is in asynchronous mode, and separate clock and frame sync signals are used for the transmit and receive sections. When SYN is set, the ESSI is in synchronous mode and the transmit and receive sections use common clock and frame sync signals. Only in synchronous mode can more than one transmitter be enabled.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears SYN.

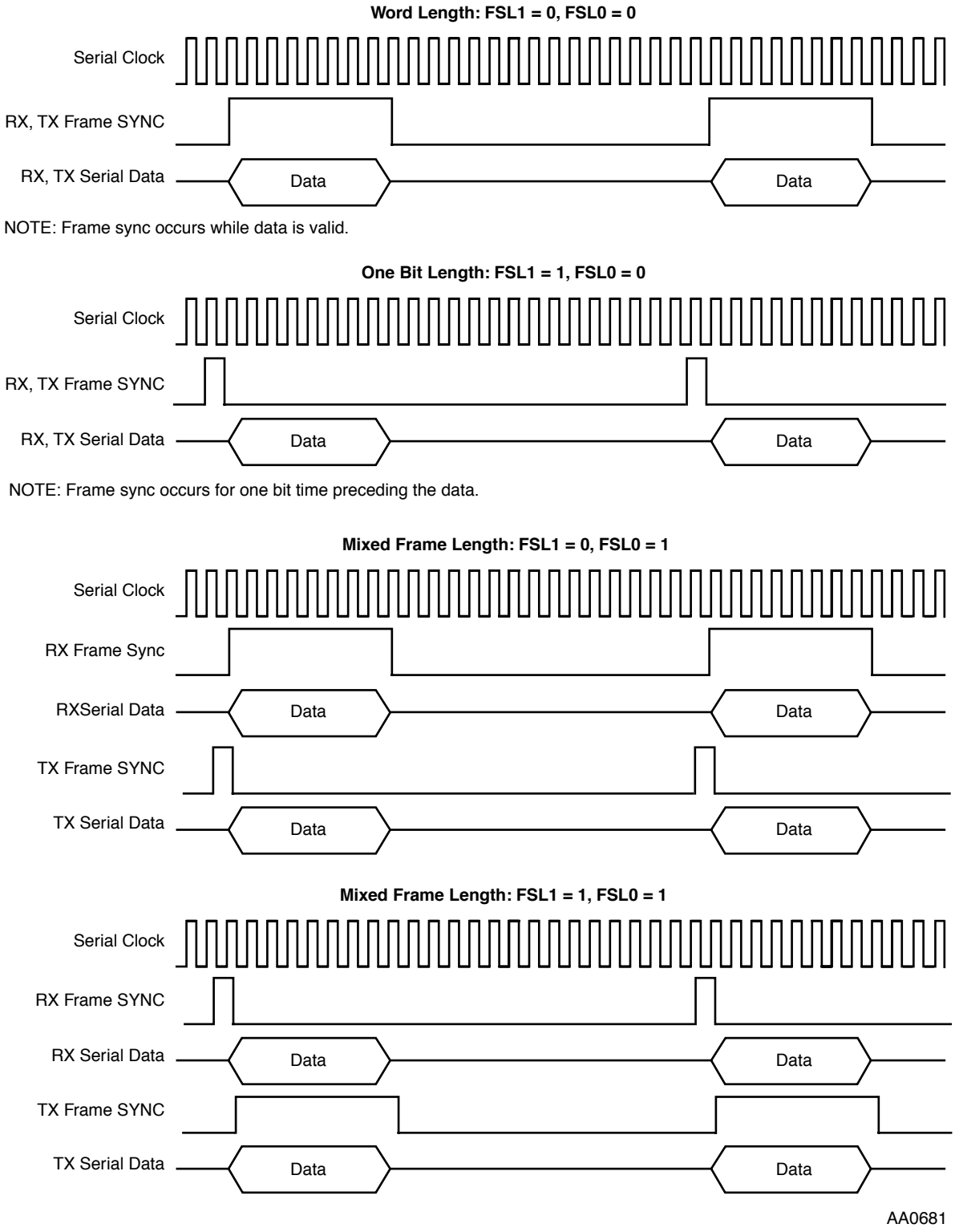


Figure 7-11 CRB FSL0 and FSL1 Bit Operation (FSR = 0)

7.4.2.12 CRB ESSI Mode Select (MOD) Bit 13

MOD selects the operational mode of the ESSI; see **Figure 7-13** on page 7-21, **Figure 7-14** on page 7-22, and **Figure 7-15** on page 7-23. When MOD is cleared, normal mode is selected; when MOD is set, network mode is selected. In normal mode, the frame rate divider determines the word transfer rate: one word is transferred per frame sync during the frame sync time slot. In network mode, a word can be transferred every time slot. For more details, see **Section 7.5—Operating Modes**. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears MOD.

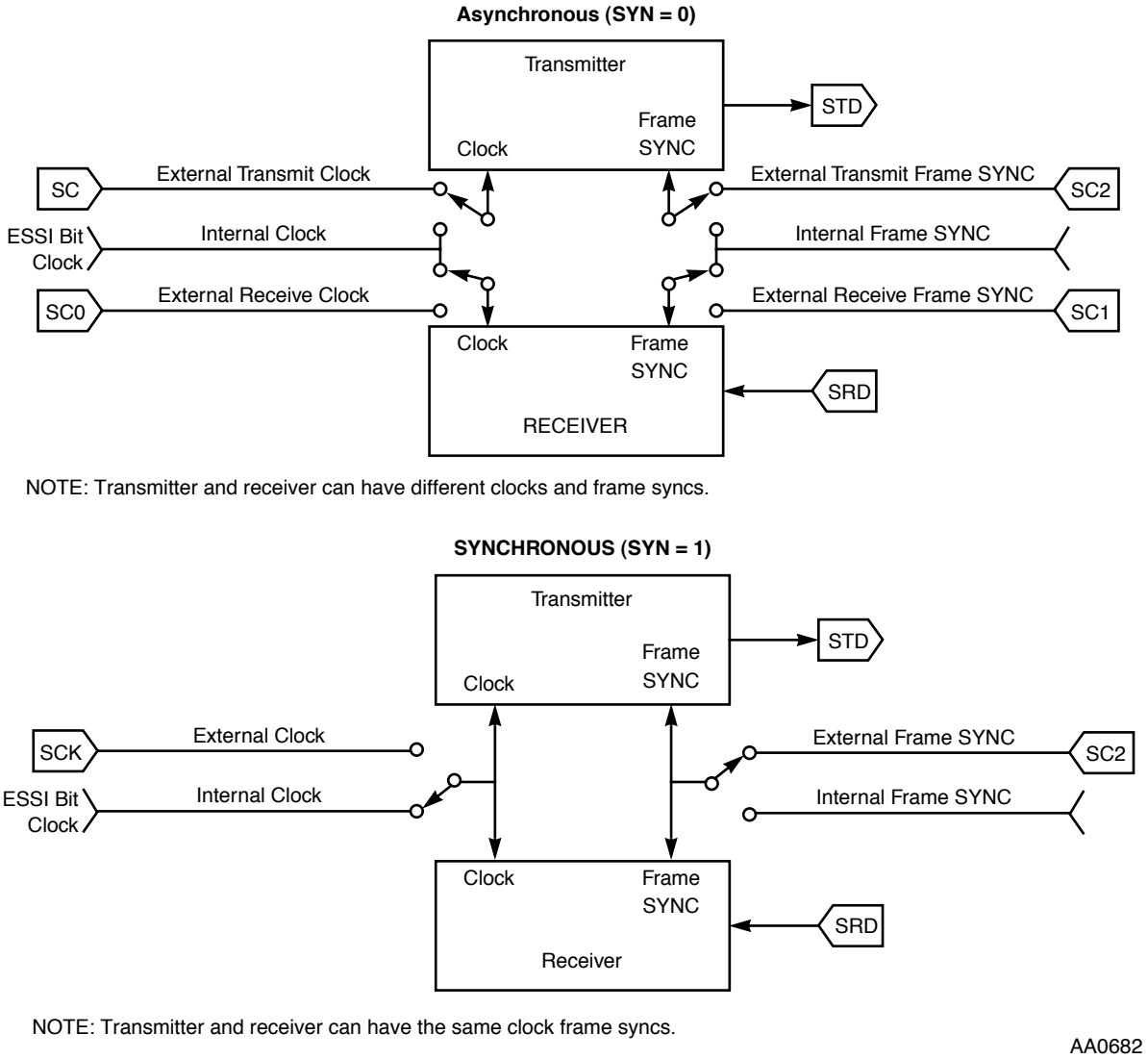


Figure 7-12 CRB SYN Bit Operation

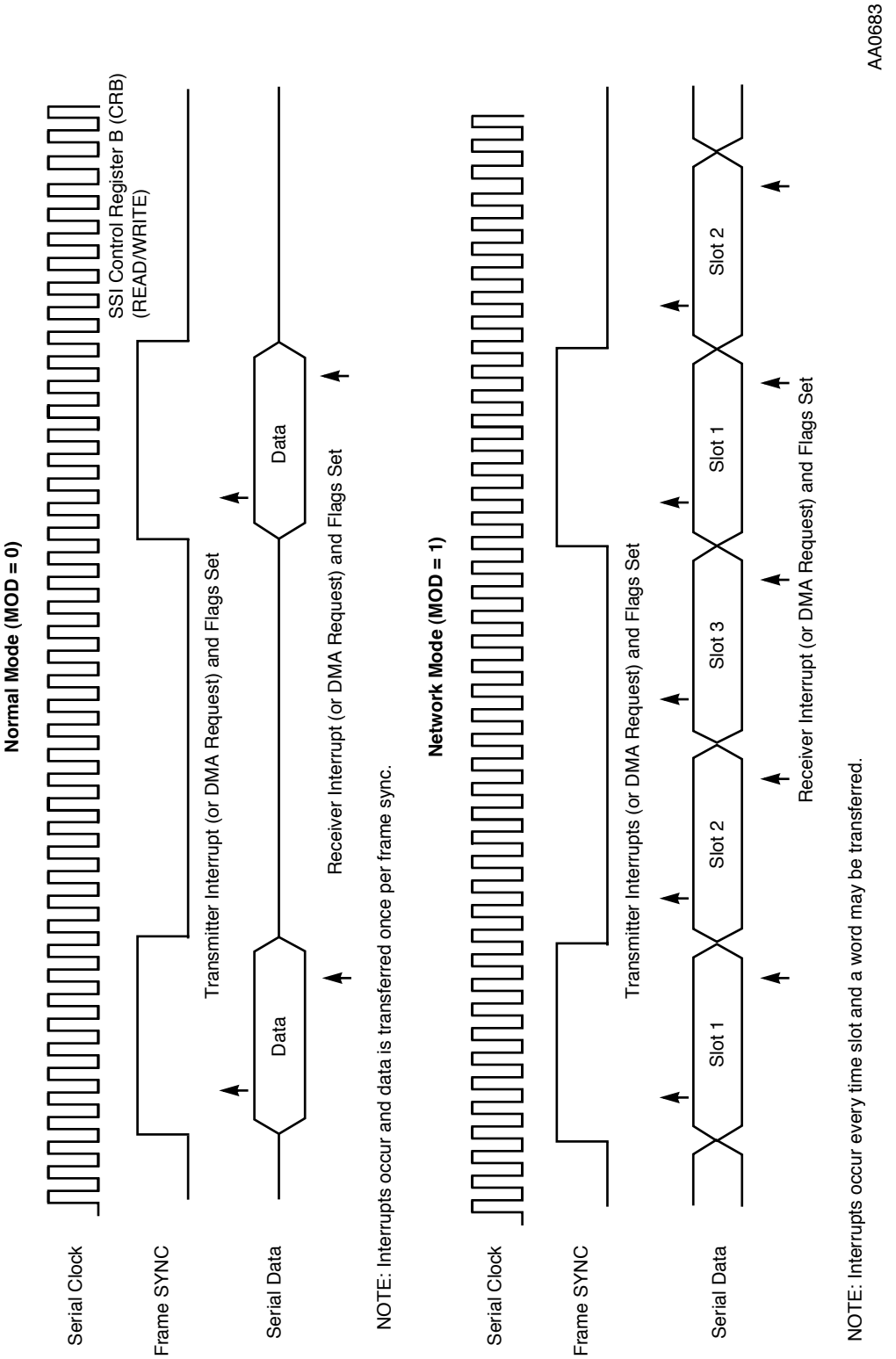


Figure 7-13 CRB MOD Bit Operation

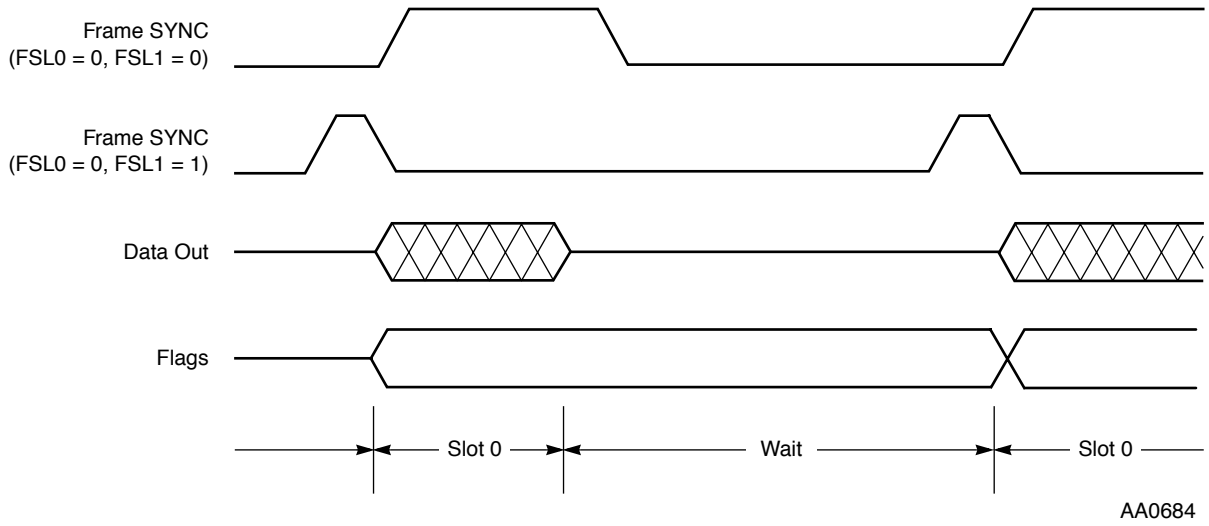


Figure 7-14 Normal Mode, External Frame Sync (8 Bit, 1 Word in Frame)

7.4.2.13 Enabling, Disabling ESSI Data Transmission

The ESSI has three transmit enable bits (TE[2:0]), one for each data transmitter. The process of transmitting data from TX1 and TX2 is the same. TX0 can also operate in asynchronous mode. The normal transmit enable sequence is to write data to one or more transmit data registers (or the time slot register (TSR)) before setting the TE bit. The normal transmit disable sequence is to clear the TE, transmit interrupt enable (TIE), and transmit exception interrupt enable (TEIE) bits after the transmit data empty (TDE) bit is set. In network mode, clearing the appropriate TE bit and setting it again disables the corresponding transmitter (0, 1, or 2) after transmission of the current data word. The transmitter remains disabled until the beginning of the next frame. During that time period, the corresponding SC (or STD in the case of TX0) signal remains in the high-impedance state.

7.4.2.14 CRB ESSI Transmit 2 Enable (TE2) Bit 14

The TE2 bit enables the transfer of data from TX2 to transmit shift register 2. TE2 is functional only when the ESSI is in synchronous mode and is ignored when the ESSI is in asynchronous mode.

When TE2 is set and a frame sync is detected, transmitter 2 is enabled for that frame.

When TE2 is cleared, transmitter 2 is disabled after completing transmission of data currently in the ESSI transmit shift register. Any data present in TX2 is not transmitted. If TE2 is cleared, data can be written to TX2; the TDE bit is cleared, but data is not transferred to transmit shift register 2.

Keeping the TE2 bit cleared until the start of the next frame causes the SC1 signal to act as serial I/O flag from the start of the frame, in both normal and network mode. The on-demand mode transmit enable sequence can be the same as normal mode, or the TE2 bit can be left enabled.

The TE2 bit is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Note: The setting of the TE2 bit does not affect the generation of frame sync or output flags.

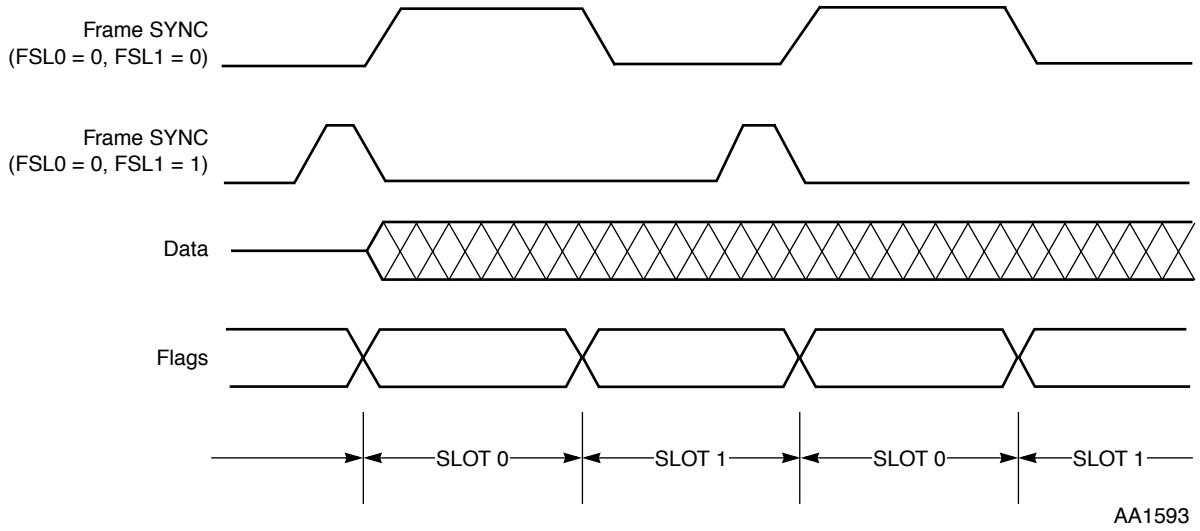


Figure 7-15 Network Mode, External Frame Sync (8 Bit, 2 Words in Frame)

7.4.2.15 CRB ESSI Transmit 1 Enable (TE1) Bit 15

The TE1 bit enables the transfer of data from TX1 to transmit shift register 1. TE1 is functional only when the ESSI is in synchronous mode and is ignored when the ESSI is in asynchronous mode.

When TE1 is set and a frame sync is detected, the transmitter 1 is enabled for that frame.

When TE1 is cleared, transmitter 1 is disabled after completing transmission of data currently in the ESSI transmit shift register. Any data present in TX1 is not transmitted. If TE1 is cleared, data can be written to TX1; the TDE bit is cleared, but data is not transferred to transmit shift register 1.

Keeping the TE1 bit cleared until the start of the next frame causes the SC0 signal to act as serial I/O flag from the start of the frame, in both normal and network mode. The transmit enable sequence for on-demand mode can be the same as for normal mode, or the TE1 bit can be left enabled.

The TE1 bit is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Note: The setting of the TE1 bit does not affect the generation of frame sync or output flags.

7.4.2.16 CRB ESSI Transmit 0 Enable (TE0) Bit 16

The TE0 bit enables the transfer of data from TX0 to transmit shift register 0. TE0 is functional when the ESSI is in either synchronous or asynchronous mode.

When TE0 is set and a frame sync is detected, the transmitter 0 is enabled for that frame.

When TE0 is cleared, transmitter 0 is disabled after completing transmission of data currently in the ESSI transmit shift register. The STD output is tri-stated, and any data present in TX0 is not transmitted (i.e., data can be written to TX0 with TE0 cleared; the TDE bit is cleared, but data is not transferred to the transmit shift register 0).

The TE0 bit is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

The transmit enable sequence for on-demand mode can be the same as for normal mode, or TE0 can be left enabled.

Note: Transmitter 0 is the only transmitter that can operate in asynchronous mode (SYN = 0). TE0 does not affect the generation of frame sync or output flags.

Table 7-4 summarizes the preceding sections; it shows possible settings of control bits and their associated signals.

Table 7-4 Mode and Signal Definition Table

Control Bits					ESSI Signals					
SYN	**TE0**	**TE1**	**TE2**	**RE**	**SC0**	**SC1**	**SC2**	**SCK**	**STD**	**SRD**
0	0	X	X	0	U	U	U	U	U	U
0	0	X	X	1	RXC	FSR	U	U	U	RD
0	1	X	X	0	U	U	FST	TXC	TD0	U
0	1	X	X	1	RXC	FSR	FST	TXC	TD0	RD
1	0	0	0	0	U	U	U	U	U	U

Table 7-4 Mode and Signal Definition Table (Continued)

Control Bits					ESSI Signals					
SYN	**TE0**	**TE1**	**TE2**	**RE**	**SC0**	**SC1**	**SC2**	**SCK**	**STD**	**SRD**
1	0	0	0	1	F0/U	F1/T0D/U	FS	XC	U	RD
1	0	0	1	0	F0/U	TD2	FS	XC	U	U
1	0	0	1	1	F0/U	TD2	FS	XC	U	RD
1	0	1	0	0	TD1	F1/T0D/U	FS	XC	U	U
1	0	1	0	1	TD1	F1/T0D/U	FS	XC	U	RD
1	0	1	1	0	TD1	TD2	FS	XC	U	U
1	0	1	1	1	TD1	TD2	FS	XC	U	RD
1	1	0	0	0	F0/U	F1/T0D/U	FS	XC	TD0	U
1	1	0	0	1	F0/U	F1/T0D/U	FS	XC	TD0	RD
1	1	0	1	0	F0/U	TD2	FS	XC	TD0	U
1	1	0	1	1	F0/U	TD2	FS	XC	TD0	RD
1	1	1	0	0	TD1	F1/T0D/U	FS	XC	TD0	U
1	1	1	0	1	TD1	F1/T0D/U	FS	XC	TD0	RD
1	1	1	1	0	TD1	TD2	FS	XC	TD0	U
1	1	1	1	1	TD1	TD2	FS	XC	TD0	RD

Note: TXC = Transmitter Clock
Note: RXC = Receiver Clock
Note: XC = Transmitter/Receiver Clock (Synchronous Operation)
Note: FST = Transmitter Frame Sync
Note: FSR = Receiver Frame Sync
Note: FS = Transmitter/Receiver Frame Sync (Synchronous Operation)
Note: TD0 = Transmit Data signal 0
Note: TD1 = Transmit Data signal 1
Note: TD2 = Transmit Data signal 2
Note: T0D = Transmitter 0 drive enable if SSC1 = 1 & SCD1 = 1
Note: RD = Receive Data
Note: F0 = Flag 0
Note: F1 = Flag 1 if SSC1 = 0
Note: U = Unused (can be used as GPIO signal)
Note: X = Indeterminate

7.4.2.17 CRB ESSI Receive Enable (RE) Bit 17

When the RE bit is set, the receive portion of the ESSI is enabled. When this bit is cleared, the receiver is disabled by inhibiting data transfer into RX. If data is being received while this bit is cleared, the remainder of the word is shifted in and transferred to the ESSI receive data register.

RE must be set in both the normal and on-demand modes for the ESSI to receive data. In network mode, clearing RE and setting it again disables the receiver after reception of the current data word. The receiver remains disabled until the beginning of the next data frame.

RE is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Note: The setting of the RE bit does not affect the generation of a frame sync.

7.4.2.18 CRB ESSI Transmit Interrupt Enable (TIE) Bit 18

Setting the TIE bit enables a DSP transmit interrupt, which is generated when both the TIE and the TDE bits in the ESSI status register are set. When TIE is cleared, the transmit interrupt is disabled. The use of the transmit interrupt is described in **Section 7.5.3**. Writing data to the data registers of the enabled transmitters or to the TSR clears TDE and also clears the interrupt. Transmit interrupts with exception conditions have higher priority than normal transmit data interrupts. If the transmitter underrun error (TUE) bit is set, signaling that an exception has occurred, and the TEIE bit is set, the ESSI requests an SSI transmit data with exception interrupt from the interrupt controller.

TIE is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

7.4.2.19 CRB ESSI Receive Interrupt Enable (RIE) Bit 19

Setting the RIE enables a DSP receive data interrupt, which is generated when both the RIE and receive data register full (RDF) bit in the SSISR are set. When RIE is cleared, this interrupt is disabled. The use of the receive interrupt is described in **Section 7.5.3**. Reading the receive data register clears RDF and the pending interrupt. Receive interrupts with exception have higher priority than normal receive data interrupts. If the receiver overrun error (ROE) bit is set, signaling that an exception has occurred, and the REIE bit is set, the ESSI requests an SSI receive data with exception interrupt from the interrupt controller.

RIE is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

7.4.2.20 Transmit Last Slot Interrupt Enable (TLIE) Bit 20

Setting the TLIE bit enables an interrupt at the beginning of the last slot of a frame when the ESSI is in network mode. When TLIE is set, the DSP is interrupted at the start of the last slot in a frame regardless of the transmit mask register setting. When TLIE is cleared,

the transmit last slot interrupt is disabled. The use of the transmit last slot interrupt is described in **Section 7.5.3—ESSI Exceptions**.

TLIE is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction. TLIE is disabled when the ESSI is in on-demand mode (DC = $0).

7.4.2.21 Receive Last Slot Interrupt Enable (RLIE) Bit 21

Setting the RLIE bit enables an interrupt after the last slot of a frame ends when the ESSI is in network mode. When RLIE is set, the DSP is interrupted after the last slot in a frame ends regardless of the receive mask register setting. When RLIE is cleared, the receive last slot interrupt is disabled. The use of the receive last slot interrupt is described in **Section 7.5.3—ESSI Exceptions**.

RLIE is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction. RLIE is disabled when the ESSI is in on-demand mode (DC = $0).

7.4.2.22 Transmit Exception Interrupt Enable (TEIE) Bit 22

When the TEIE bit is set, the DSP is interrupted when both TDE and TUE in the ESSI Status Register are set. When TEIE is cleared, this interrupt is disabled. The use of the transmit interrupt is described in **Section 7.5.3—ESSI Exceptions**. Reading the status register, followed by writing to all the data registers of the enabled transmitters, clears both TUE and the pending interrupt.

TEIE is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

7.4.2.23 Receive Exception Interrupt Enable (REIE) Bit 23

When the REIE bit is set, the DSP is interrupted when both RDF and ROE in the ESSI status register are set. When REIE is cleared, this interrupt is disabled. The use of the receive interrupt is described in **Section 7.5.3—ESSI Exceptions**. Reading the status register followed by reading the receive data register clears both ROE and the pending interrupt.

REIE is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

7.4.3 ESSI Status Register (SSISR)

The SSISR (in **Figure 7-4** on page 7-9) is a 24-bit, read-only status register used by the DSP to read the status and serial input flags of the ESSI. The SSISR bits are documented in the following paragraphs.

7.4.3.1 SSISR Serial Input Flag 0 (IF0) Bit 0

The IF0 bit is enabled only when SC0 is an input flag and synchronous mode is selected (i.e., when the SYN bit is set, and the TE1 and SCD0 bits are cleared).

The ESSI latches data present on the SC0 signal during reception of the first received bit after the frame sync is detected. The IF0 bit is updated with this data when the data in the receive shift register is transferred into the receive data register.

If it is not enabled, the IF0 bit is cleared.

A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears the IF0 bit.

7.4.3.2 SSISR Serial Input Flag 1 (IF1) Bit 1

The IF1 bit is enabled only when SC1 is an input flag and synchronous mode is selected, the SYN bit is set, and the TE2 and SCD1 bits are cleared.

The ESSI latches data present on the SC1 signal during reception of the first received bit after the frame sync is detected. The IF1 bit is updated with this data when the data in the receive shift register is transferred into the receive data register.

If it is not enabled, the IF1 bit is cleared.

A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears the IF1 bit.

7.4.3.3 SSISR Transmit Frame Sync Flag (TFS) Bit 2

When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. If the transmitter is enabled, data written to a transmit data register during the time slot when TFS is set is transmitted (in network mode) during the second time slot in the frame. TFS is useful in network mode to identify the start of a frame. TFS is valid only if at least one transmitter is enabled (TE0, TE1 or TE2 are set).

A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears TFS.

Note: In normal mode, TFS is always read as 1 when transmitting data because there is only one time slot per frame, the 'frame sync' time slot.

7.4.3.4 SSISR Receive Frame Sync Flag (RFS) Bit 3

When set, the RFS bit indicates that a receive frame sync occurred during the reception of a word in the serial receive data register. This means that the data word is from the

first time slot in the frame. When the RFS bit is cleared and a word is received, it indicates (only in Network mode) that the frame sync did not occur during reception of that word. RFS is valid only if the receiver is enabled (i.e., the RE bit is set).

A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears RFS.

Note: In normal mode, RFS is always read as 1 when reading data because there is only one time slot per frame, the frame sync time slot.

7.4.3.5 SSISR Transmitter Underrun Error Flag (TUE) Bit 4

The TUE bit is set when at least one of the enabled serial transmit shift registers is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TX registers that were not written) is retransmitted. In normal mode, there is only one transmit time slot per frame. In network mode, there can be up to thirty-two transmit time slots per frame. If the TEIE bit is set, a DSP transmit underrun error interrupt request is issued when the TUE bit is set.

A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears TUE. TUE can also be cleared by first reading the SSISR with the TUE bit set, then writing to all the enabled transmit data registers or to the TSR.

7.4.3.6 SSISR Receiver Overrun Error Flag (ROE) Bit 5

The ROE bit is set when the serial receive shift register is filled and ready to transfer to the receive data register (RX), but RX is already full (i.e., the RDF bit is set). If the REIE bit is set, a DSP receiver overrun error interrupt request is issued when the ROE bit is set.

A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears ROE. ROE can also be cleared by reading the SSISR with the ROE bit set and then reading the RX.

7.4.3.7 ESSI Transmit Data Register Empty (TDE) Bit 6

The TDE bit is set when the contents of the transmit data register of every enabled transmitter are transferred to the transmit shift register. It is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, the TDE bit indicates that data should be written to all the TX registers of the enabled transmitters or to the TSR. The TDE bit is cleared when the DSP56309 writes to all the transmit data registers of the enabled transmitters or when the DSP writes to the TSR to disable transmission of the next time slot. If the TIE bit is set, a DSP transmit data interrupt request is issued when TDE is set. A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears the TDE bit.

7.4.3.8 ESSI Receive Data Register Full (RDF) Bit 7

The RDF bit is set when the contents of the receive shift register are transferred to the receive data register. The RDF bit is cleared when the DSP reads the receive data register. If RIE is set, a DSP receive data interrupt request is issued when RDF is set. A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction clears the RDF bit.

The ESSI data path programming models are shown in **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32.

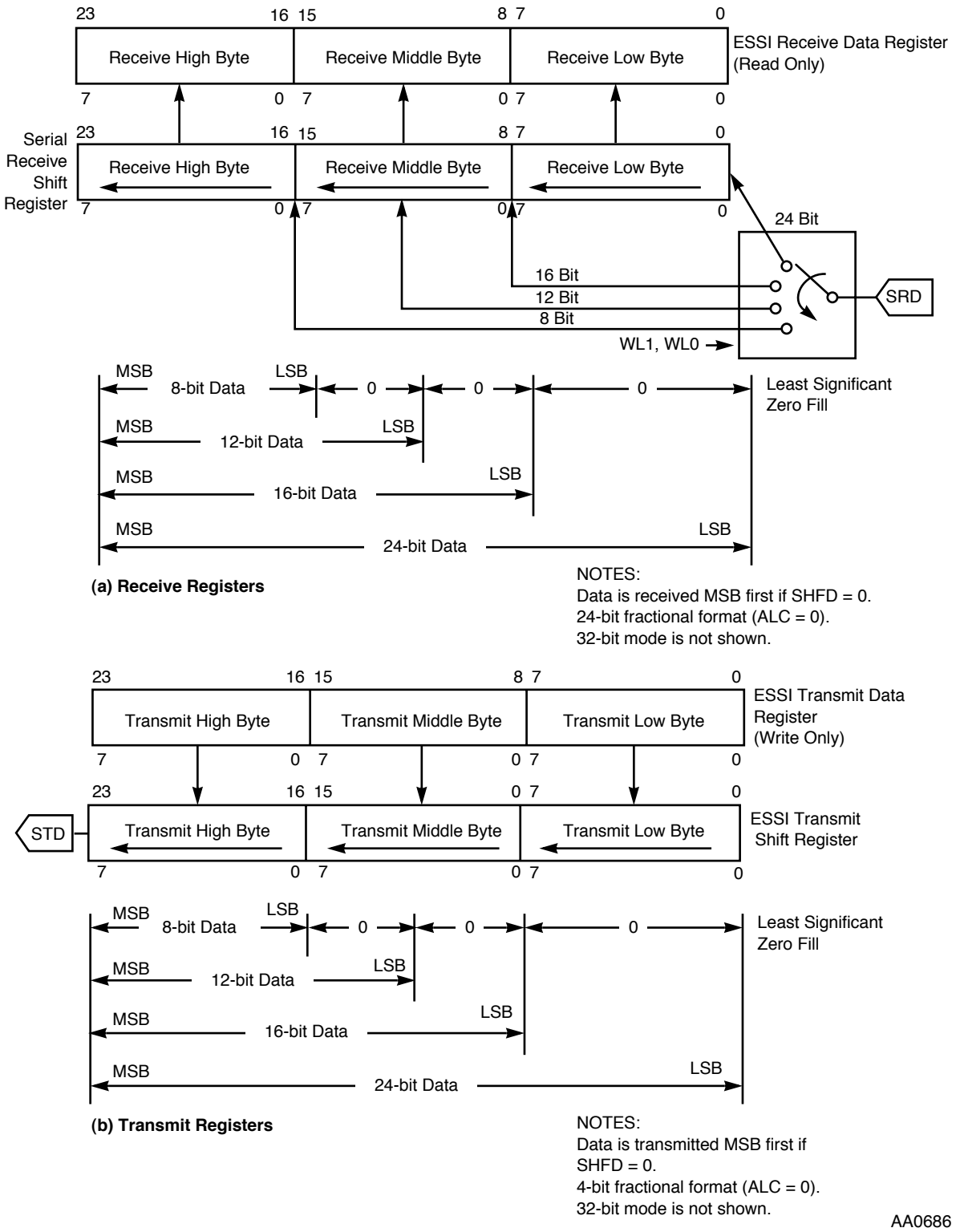


Figure 7-16 ESSI Data Path Programming Model (SHFD = 0)

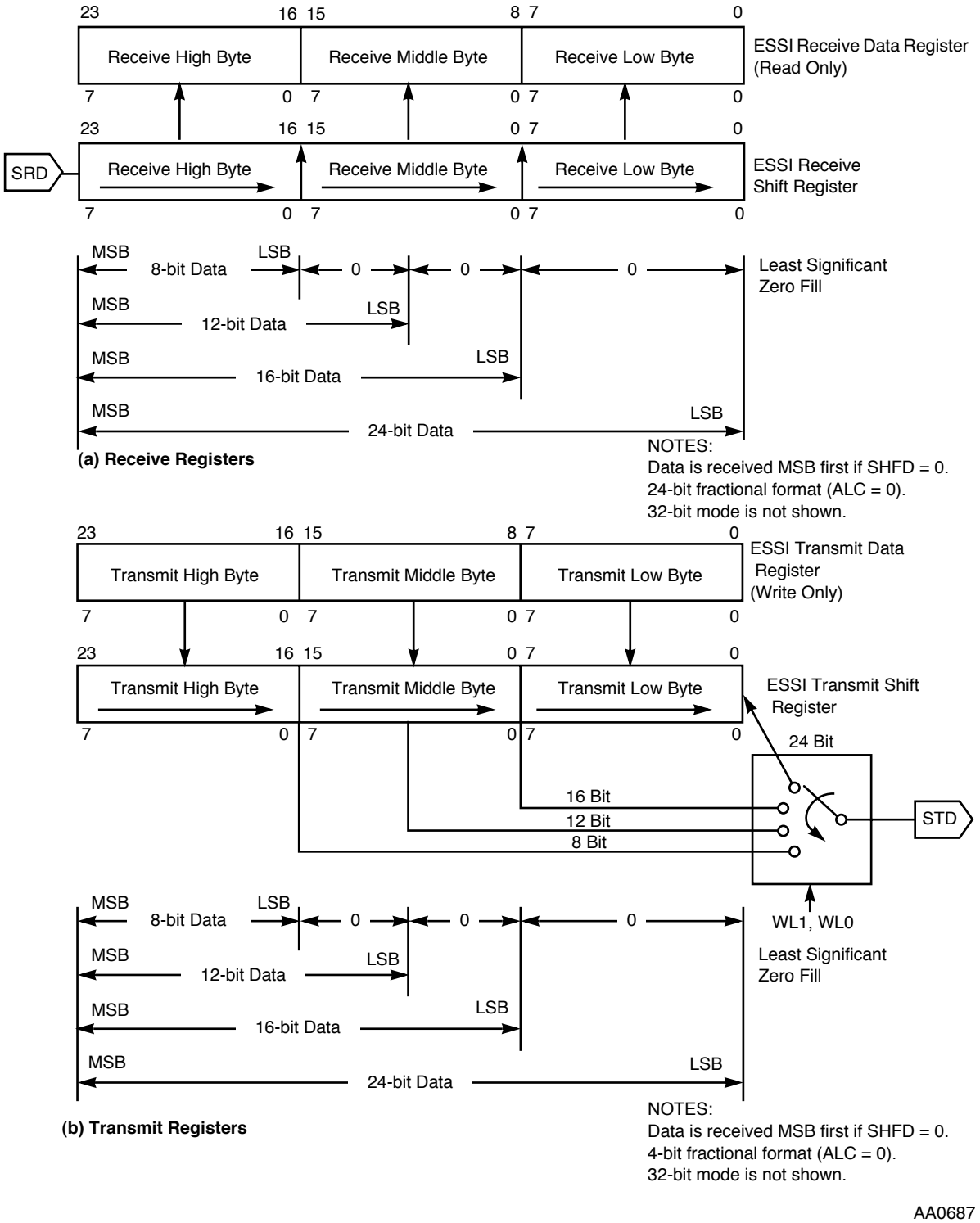


Figure 7-17 ESSI Data Path Programming Model (SHFD = 1)

7.4.4 ESSI Receive Shift Register

The 24-bit receive shift register (in **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32) receives the incoming data from the serial receive data signal. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. It is assumed that data is received MSB first if SHFD is cleared and LSB first if SHFD is set. Data is transferred to the ESSI receive data register after 8, 12, 16, 24, or 32 serial clock cycles are counted, depending on the word-length control bits in the CRA.

7.4.5 ESSI Receive Data Register (RX)

The receive data register (RX) is a 24-bit, read-only register that accepts data from the receive shift register as it becomes full; see **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32. The data read is aligned according to the value of the ALC bit. When the ALC bit is cleared, the MSB is bit 23 and the least significant byte is unused. When the ALC bit is set, the MSB is bit 15 and the most significant byte is unused. Unused bits are read as 0s. If the associated interrupt is enabled, the DSP is interrupted whenever the RX register becomes full.

7.4.6 ESSI Transmit Shift Registers

The three 24-bit transmit shift registers contain the data being transmitted; see **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32. Data is shifted out to the serial transmit data signals by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. The word-length control bits in the CRA determine the number of bits that must be shifted out before the shift registers are considered empty and can be written to again. Depending on the setting of the CRA, the number of bits to be shifted out can be 8, 12, 16, 24, or 32 bits.

The data transmitted is aligned according to the value of the ALC bit. When the ALC bit is cleared, the MSB is bit 23 and the least significant byte is unused. When ALC is set, the MSB is bit 15 and the most significant byte is unused. Unused bits are read as 0s. Data is shifted out of these registers MSB first if the SHFD bit is cleared and LSB first if the SHFD bit is set.

7.4.7 ESSI Transmit Data Registers (TX0-2)

TX20, TX10, and TX00 are transmit data registers for ESSI0. TX21, TX11, and TX01 are transmit data registers for ESSI1. TX20 and TX21 are known as TX2. TX10 and TX11 are known as TX1. TX00 and TX01 are known as TX0.

TX2, TX1, and TX0 are 24-bit, write-only registers. Data to be transmitted is written into these registers and automatically transferred to the transmit shift registers; see **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32. The data transmitted (8, 12, 16, or 24 bits) is aligned according to the value of the ALC bit. When the ALC bit is cleared, the MSB is Bit 23. When ALC is set, the MSB is Bit 15. If the transmit data register empty interrupt has been enabled, the DSP is interrupted whenever a transmit data register becomes empty.

Note: When data is written to a peripheral device, there is a two cycle pipeline delay until any status bits affected by this operation are updated. If you read any of those status bits within the next two cycles, the bit does not reflect its current status. See the DSP56300 Family Manual, Appendix B, Polling a Peripheral Device for Write for further details.

7.4.8 ESSI Time Slot Register (TSR)

TSR is effectively a write-only null data register that is used to prevent data transmission in the current transmit time slot. For the purposes of timing, TSR is a write-only register that behaves like an alternative transmit data register, except that, rather than transmitting data, the transmit data signals of all the enabled transmitters are in the high-impedance state for the current time slot.

7.4.9 Transmit Slot Mask Registers (TSMA, TSMB)

The transmit slot mask Registers are two 16-bit, read/write registers. When the TSMA or TSMB is read to the internal data bus, the register contents occupy the two low-order bytes of the data bus, and the high-order byte is zero-filled. In network mode, these registers are used by the transmitter(s) to determine what action to take in the current transmission slot. Depending on the setting of the bits, the transmitter(s) either tri-state the transmitter(s) data signal(s) or transmit a data word and generate a transmitter empty condition.

TSMA and TSMB (in **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32) can be seen as a single 32-bit register, TSM. Bit n in TSM (TSn) is an enable/disable control bit for transmission in slot number N. When TSn is cleared, all the transmit data signals of the enabled transmitters are tri-stated during transmit time slot number N. The data is still transferred from the enabled transmit data register(s) to the transmit shift register. However, the TDE and the TUE flags are not set. This means that during a disabled slot, no transmitter empty interrupt is generated. The DSP is interrupted only for enabled slots. Data written to the transmit data register when servicing the transmitter empty interrupt request is transmitted in the next enabled transmit time slot.

When TSn is set, the transmit sequence proceeds normally. Data is transferred from the TX register to the shift register during slot number N and the TDE flag is set.

Using the TSM slot mask does not conflict with using the TSR. Even if a slot is enabled in the TSM, you can write to the TSR to tri-state the signals of the enabled transmitters during the next transmission slot. Setting the bits in the TSM affects the next frame transmission. The frame currently being transmitted is not affected by the new TSM setting. If the TSM is read, it shows the current setting.

After a hardware $\overline{\text{RESET}}$ signal or software RESET instruction, the TSM register is reset to $FFFFFFFF; this setting enables all thirty-two slots for data transmission.

7.4.10 Receive Slot Mask Registers (RSMA, RSMB)

The receive slot mask registers are two 16-bit, read/write registers. In network mode, these registers are used by the receiver(s) to determine what action to take in the current time slot. Depending on the setting of the bits, the receiver(s) either tri-state the receiver(s) data signal(s) or receive a data word and generate a receiver full condition.

RSMA and RSMB (in **Figure 7-16** on page 7-31 and **Figure 7-17** on page 7-32) can be seen as one 32-bit register, RSM. Bit n in RSM (RSn) is an enable/disable control bit for time slot number N. When RSn is cleared, all the data signals of the enabled receivers are tri-stated during time slot number N. Data is transferred from the receive data register(s) to the receive shift register(s) and the RDF and ROE flags are not set. During a disabled slot, no receiver full interrupt is generated. The DSP is interrupted only for enabled slots.

When RSn is set, the receive sequence proceeds normally. Data is received during slot number N, and the RDF flag is set.

Setting the bits in the RSM affects the next frame transmission. The frame currently being transmitted is not affected by the new RSM setting. If the RSM is read, it shows the current setting.

When the RSMA or RSMB register are read by the internal data bus, the register contents occupy the two low-order bytes of the data bus, and the high-order byte is zero-filled.

After a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction, the RSM register is reset to $FFFFFFFF; this setting enables all thirty-two time slots for data transmission.

7.5 OPERATING MODES

The ESSI operating modes are selected by the ESSI control registers (CRA and CRB). The operating modes are described in the following paragraphs.

7.5.1 ESSI After Reset

A hardware $\overline{\text{RESET}}$ signal or software RESET instruction clears the port control register and the port direction control register. This situation configures all the ESSI signals as GPIO. The ESSI is in the reset state while all ESSI signals are programmed as GPIO; the ESSI is active only if at least one of the ESSI I/O signals is programmed as an ESSI signal.

7.5.2 ESSI Initialization

To initialize the ESSI, do the following:

1. Send a reset: a hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction.
2. Program the ESSI control and time slot registers.
3. Write data to all the enabled transmitters.
4. Configure at least one signal as an ESSI signal.
5. If an external frame sync is used, from the moment the ESSI is activated, at least five serial clocks are needed before the first external frame sync is supplied. Otherwise, improper operation can result.

Clearing the PC[5:0] bits in the GPIO PCR during program execution causes the ESSI to stop serial activity and enter the individual reset state. All status bits of the interface are set to their reset state. The contents of CRA and CRB are not affected. The ESSI individual reset allows a program to reset each interface separately from the other

internal peripherals. During ESSI individual reset, internal DMA accesses to the data registers of the ESSI are not valid and data read is undefined.

To insure proper operation of the ESSI, use an ESSI individual reset when changing the ESSI control registers (except for bits TEIE, REIE, TLIE, RLIE, TIE, RIE, TE2, TE1, TE0, and RE).

Here is an example of initializing the ESSI.

1. Put the ESSI in its individual reset state by clearing the PCR bits.
2. Configure the control registers (CRA, CRB) to set the operating mode. Disable the transmitters and receiver by clearing the TE[2:0] and RE bits. Set the interrupt enable bits for the operating mode chosen.
3. Enable the ESSI by setting the PCR bits to activate the input/output signals to be used.
4. Write initial data to the transmitters that are used during operation. This step is needed even if DMA is used to service the transmitters.
5. Enable the transmitters and receiver to be used.

Now the ESSI can be serviced by polling, interrupts, or DMA.

Once the ESSI has been enabled (Step 3), operation starts as follows:

- For internally generated clock and frame sync, these signals start activity immediately after the ESSI is enabled.
- Data is received by the ESSI after the occurrence of a frame sync signal (either internally or externally generated) only when the receive enable (RE) bit is set.
- Data is transmitted after the occurrence of a frame sync signal (either internally or externally generated) only when the transmitter enable (TE[2:0]) bit is set.

7.5.3 ESSI Exceptions

The ESSI can generate six different exceptions. They are discussed in the following paragraphs (ordered from the highest to the lowest exception priority)**:**

1. ESSI receive data with exception status:
   Occurs when the receive exception interrupt is enabled, the receive data register is full, and a receiver overrun error has occurred. This exception sets the ROE bit. The ROE bit is cleared by first reading the SSISR and then reading RX.

2. ESSI receive data:
   Occurs when the receive interrupt is enabled, the receive data register is full, and no receive error conditions exist. Reading RX clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead.
3. ESSI receive last slot interrupt:
   Occurs when the ESSI is in network mode and the last slot of the frame has ended. This interrupt is generated regardless of the receive mask register setting. The receive last slot interrupt can be used to signal that the receive mask slot register can be reset, the DMA channels can be reconfigured, and data memory pointers can be reassigned. Using the receive last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame is serviced with the new setting without synchronization problems.

Note: The maximum time it takes to service a receive last slot interrupt should not exceed N – 1 ESSI bits service time (where N is the number of bits the ESSI can transmit per time slot).

4. ESSI transmit data with exception status:
   Occurs when the transmit exception interrupt is enabled, at least one transmit data register of the enabled transmitters is empty, and a transmitter underrun error has occurred. This exception sets the TUE bit. The TUE bit is cleared by first reading the SSISR and then writing to all the transmit data registers of the enabled transmitters or by writing to the TSR to clear the pending interrupt.
5. ESSI transmit last slot interrupt:
   Occurs when the ESSI is in network mode at the start of the last slot of the frame. This exception occurs regardless of the transmit mask register setting. The transmit last slot interrupt can be used to signal that the transmit mask slot register can be reset, the DMA channels can be reconfigured, and data memory pointers can be reassigned. Using the transmit last slot interrupt guarantees that the previous frame was serviced with the previous setting, and the new frame is serviced with the new setting without synchronization problems.

Note: The maximum transmit last slot interrupt service time should not exceed N – 1 ESSI bits service time (where N is the number of bits in a slot).

6. ESSI transmit data:
   Occurs when the transmit interrupt is enabled, at least one of the enabled transmit data registers is empty, and no transmitter error conditions exist. Writing to all the enabled TX registers or to the TSR clears this interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead (if no more than two transmitters are used).

To configure an ESSI exception, perform the following steps:

1. Configure interrupt service routine (ISR)
   a. Load vector base address register. `VBA (b23:8)`
   b. Define I_VEC to be equal to the VBA value (if that is nonzero). If it is defined, I_VEC must be defined for the assembler before the interrupt equate file is included.
   c. Load the exception vector table entry: two-word fast interrupt or jump/branch to subroutine (long interrupt). `p:I_SI0TD`
2. Configure interrupt trigger/preload transmit data
   a. Enable and prioritize overall peripheral interrupt functionality. `IPRP (S0L1:0)`
   b. Enable peripheral and associated signals. `PCRC (PC5:0)`
   c. Write data to all enabled transmit registers. `TX00`
   d. Enable peripheral interrupt-generating function. `CRB (TE0)`
   e. Enable specific peripheral interrupt. `CRB0 (TIE)`
   f. Unmask interrupts at global level. `SR (I1:0)`

Notes:
1. The example material to the right of the steps above shows register settings for configuring an ESSI0 transmit interrupt using transmitter 0.

2. The order of the steps is optional except that the interrupt trigger configuration must not be completed until the ISR configuration has been completed. Since 2d can cause an immediate transmit without generating an interrupt, perform the transmit data preload in 2c before 2d to insure valid data is sent in the first transmission.

3. After the first transmit, subsequent transmit values are typically loaded into TXnn by the ISR (one value per register per interrupt). Therefore, if N items are to be sent from a particular TXnn, the ISR will need to load the transmit register (N – 1) times.

4. Steps d and e can be performed using a single instruction.

5. If an interrupt trigger event occurs at a time when not all interrupt trigger configuration steps have been performed, the event is ignored forever (the event is not queued in this case).

6. If interrupts derived from the core or other peripherals need to be enabled at the same time as ESSI interrupts, step f should be done last.

7.5.4 Operating Modes: Normal, Network, and On-Demand

The ESSI has three basic operating modes and several data/operation formats. These modes can be programmed using the ESSI control registers. The data/operation formats available to the ESSI are selected by setting or clearing control bits in the CRA and CRB. These control bits are WL[2:1], MOD, SYN, FSL[1:0], FSR, FSP, CKP, and SHFD.

7.5.4.1 Normal/Network/On-Demand Mode Selection

To select normal mode (or network mode), clear (or set) the MOD bit in the CRB. In normal mode, the ESSI sends or receives one data word per frame (per enabled receiver or transmitter). In network mode, two to thirty-two time slots per frame can be selected. During each frame, zero to thirty-two data words can be received or transmitted (from each enabled receiver or transmitter). In either case, the transfers are periodic.

Normal mode is typically used to transfer data to or from a single device. Network mode is typically used in TDM networks of codecs or DSPs with multiple words per frame.

Network mode has a sub-mode called on-demand mode. Setting the MOD bit in the CRB for network mode, and setting the frame rate divider to 0 (DC = $00000) selects the on-demand mode. This sub-mode does not generate a periodic frame sync. A frame sync pulse is generated only when data is available to transmit. The frame sync signal indicates the first time slot in the frame. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). For simplex operation, synchronous mode could be used; however, for full-duplex operation, asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into each TX. Although the ESSI is double-buffered, only one word can be written to each TX, even if the transmit shift register is empty. The receive and transmit interrupts function normally, using TDE and RDF; however, transmit underruns are impossible for ‘on- demand’ transmission and are disabled. This mode is useful for interfacing to codecs requiring a continuous clock.

7.5.4.2 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of the ESSI interface can be synchronous or asynchronous. The transmitter and receiver use common clock and synchronization signals in synchronous mode; they use separate clock and sync signals in asynchronous mode. The SYN bit in CRB selects synchronous or asynchronous operation. When the SYN bit is cleared, the ESSI TX and RX clocks and frame sync sources are independent. If the SYN bit is set, the ESSI TX and RX clocks and frame sync are driven by the same source (either external or internal). Since the ESSI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

Transmitter 1 and transmitter 2 operate only in synchronous mode. Data clock and frame sync signals can be generated internally by the DSP or can be obtained from external sources. If clocks are internally generated, the ESSI clock generator derives bit clock and frame sync signals from the DSP internal system clock. The ESSI clock generator consists of a selectable fixed prescaler with a programmable prescaler for bit rate clock generation and a programmable frame-rate divider with a word-length divider for frame-rate sync-signal generation.

7.5.4.3 Frame Sync Selection

The transmitter and receiver can operate independently. The transmitter can have either a bit-long or word-long frame-sync signal format, and the receiver can have the same or another format. The selection is made by programming FSL[1:0], FSR, and FSP bits in the CRB.

7.5.4.3.1 Frame Sync Signal Format

FSL1 controls the frame-sync signal format.

- If the FSL1 bit is cleared, the RX frame sync is asserted during the entire data transfer period. This frame sync length is compatible with Motorola codecs, serial peripherals that conform to the Motorola SPI, serial A/D and D/A converters, shift registers, and telecommunication pulse code modulation (PCM) serial I/O.
- If the FSL1 bit is set, the RX frame sync pulses active for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs, and telecommunication PCM serial I/O.

7.5.4.3.2 Frame Sync Length for Multiple Devices

The ability to mix frame sync lengths is useful in configuring systems in which data is received from one type of device (e.g., codec) and transmitted to a different type of device. FSL0 controls whether RX and TX have the same frame sync length.

- If the FSL0 bit is cleared, both RX and TX have the same frame sync length.
- If the FSL0 bit is set, RX and TX have different frame sync lengths.

FSL0 is ignored when the SYN bit is set.

7.5.4.3.3 Word-Length Frame Sync and Data-Word Timing

The FSR bit controls the relative timing of the word-length frame sync relative to the data word timing.

- When the FSR bit is cleared, the word length frame sync is generated (or expected) with the first bit of the data word.

- When the FSR bit is set, the word length frame sync is generated (or expected) with the last bit of the previous word.

FSR is ignored when a bit length frame sync is selected.

7.5.4.3.4 Frame Sync Polarity

The FSP bit controls the polarity of the frame sync.

- When the FSP bit is cleared, the polarity of the frame sync is positive (i.e., the frame sync signal is asserted high). The ESSI synchronizes on the leading edge of the frame sync signal.
- When the FSP bit is set, the polarity of the frame sync is negative (i.e., the frame sync is asserted low). The ESSI synchronizes on the trailing edge of the frame sync signal.

The ESSI receiver looks for a receive frame sync edge (leading edge if FSP is cleared, trailing edge if FSP is set) only when the previous frame is completed. If the frame sync is asserted before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync or a word length frame sync with FSR set), the current frame sync is not recognized, and the receiver is internally disabled until the next frame sync.

Frames do not have to be adjacent, that is, a new frame sync does not have to follow immediately the previous frame. Gaps of arbitrary periods can occur between frames. All the enabled transmitters are tri-stated during these gaps.

7.5.4.4 Byte Format (LSB/MSB) for the Transmitter

Some devices, such as codecs, require a MSB-first data format. Other devices, such as those that use the AES-EBU digital audio format, require the LSB first. To be compatible with all formats, the shift registers in the ESSI are bidirectional. The MSB/LSB selection is made by programming the SHFD bit in the CRB.

- If the SHFD bit is cleared, data is shifted into the receive shift register MSB first and shifted out of the transmit shift register MSB first.
- If the SHFD bit is set, data is shifted into the receive shift register LSB first and shifted out of the transmit shift register LSB first.

7.5.5 Flags

Two ESSI signals (SC[1:0]) are available for use as serial I/O flags. Their operation is controlled by the SYN, SCD[1:0], SSC1, and TE[2:1] bits in the CRB/CRA.The control bits OF[1:0] and status bits IF[1:0] are double-buffered to/from SC[1:0]. Double-buffering the flags keeps the flags in sync with TX and RX.

The SC[1:0] flags are available in synchronous mode only. Each flag can be separately programmed.

Flag SC0 is enabled when transmitter 1 is disabled (TE1 = 0). The flag's direction is selected by the SCD0 bit. When SCD0 is set, SC0 is configured as output. When SCD0 is cleared, SC0 is configured as input.

Similarly, the SC1 flag is enabled when transmitter 2 is disabled (TE2 = 0) and the SC1 signal is not configured as transmitter drive enable (Bit SSC1 = 0). SC1's direction is selected by the SCD1 bit. When SCD1 is set, SC1 is an output flag. When SCD1 is cleared, SC1 is an input flag.

When programmed as input flags, the value of the SC[1:0] bits are latched at the same time as the first bit of the receive data word is sampled. Once the input has been latched, the signal on the input flag signal (SC0 and SC1) can change without affecting the input flag. The value of SC[1:0] does not change until the first bit of the next data word is received. When the received data word is latched by RX, the latched values of SC[1:0] are latched by the respective SSISR IF[1:0] bits and can be read by software.

When programed as output flags, the value of the SC[1:0] bits is taken from the value of the OF[1:0] bits. The value of the OF[1:0] bits is latched when the contents of TX are transferred to the transmit shift register. The value on SC[1:0] is stable from the time the first bit of the transmit data word is transmitted until the first bit of the next transmit data word is transmitted. The OF[1:0] values can be set directly by software. This allows the DSP56309 to control data transmission by indirectly controlling the value of the SC[1:0] flags.

7.6 GPIO SIGNALS AND REGISTERS

The GPIO functionality of an ESSI port (C, D) is controlled by three registers: port control register (PCRC, PCRD), port direction register (PRRC, PRRD) and port data register (PDRC, PDRD).

7.6.1 Port Control Register (PCR)

The read/write, 24-bit PCR controls the functionality of the ESSI GPIO signals. Each of PC[5:0] bits controls the functionality of the corresponding port signal. When a PC[i] bit is set, the corresponding port signal is configured as a ESSI signal. When a PC[i] bit is cleared, the corresponding port signal is configured as a GPIO signal. Either a hardware

$\overline{\text{RESET}}$ signal or a software RESET instruction clears all PCR bits. **Figure 7-18** shows the PCR bits.

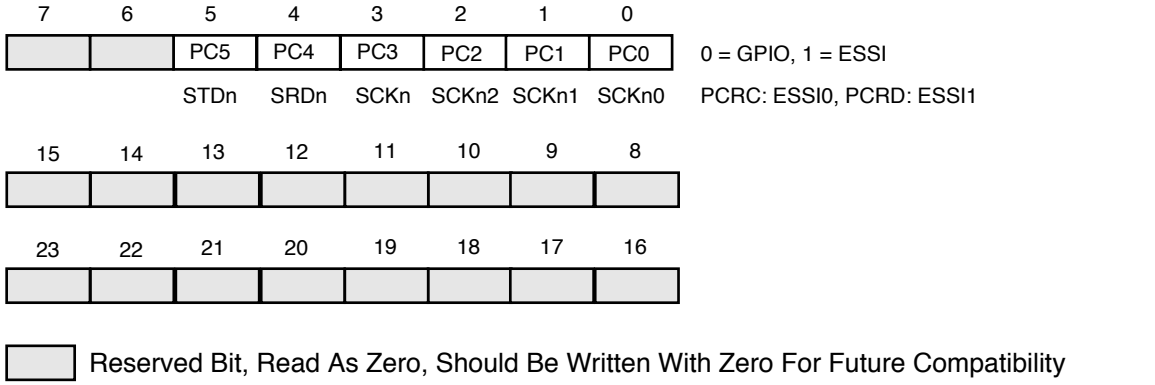


Figure 7-18 Port Control Register (PCR) (PCRC X:$FFFFBF)

(PCRD X:$FFFFAF)

7.6.2 Port Direction Register (PRR)

The read/write, 24-bit PRR controls the data direction of the ESSI GPIO signals. When PRR[i] is set, the corresponding signal is an output signal. When PRR[i] is cleared, the corresponding signal is an input signal. **Figure 7-19** shows the PRR bits.

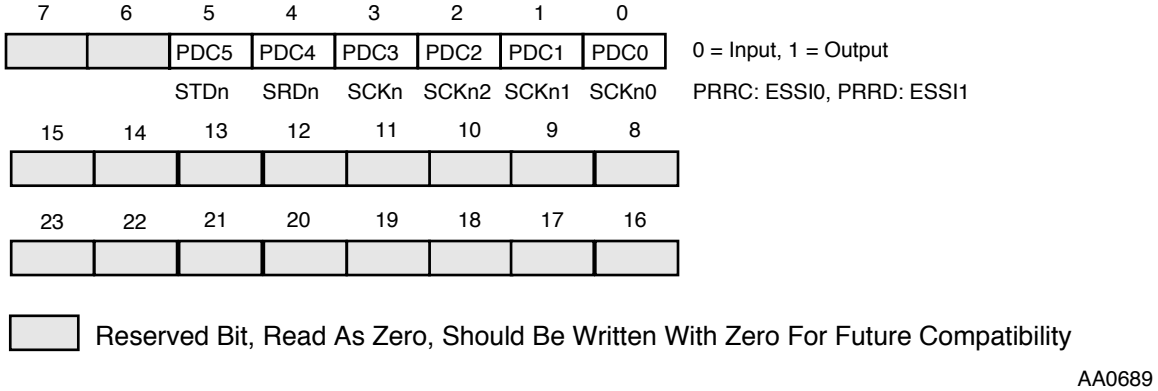


Figure 7-19 Port Direction Register (PRR)(PRRC X:$FFFFBE)

(PRRD X:$FFFFAE)

Note: Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PRR bits.

Table 7-5 shows the port signal configurations.

Table 7-5 Port Control Register and Port Direction Register Bits

PC[i]	PDC[i]	Port Signal[i] Function
1	X	ESSI
0	0	GPIO input
0	1	GPIO output
Note: X: The signal setting is irrelevant to port signal [i] function.		

7.6.3 Port Data Register (PDR)

The read/write, 24-bit PDR is used to read or write data to and from the ESSI GPIO signals. The PD[5:0] bits are used to read or write data from and to the corresponding port signals if they are configured as GPIO signals. If a port signal [i] is configured as a GPIO input, then the corresponding PD[i] bit reflects the value present on this signal. If a port signal [i] is configured as a GPIO output, then the value written into the corresponding PD[i] bit is reflected on the this signal. **Figure 7-20** shows the PDR bits.

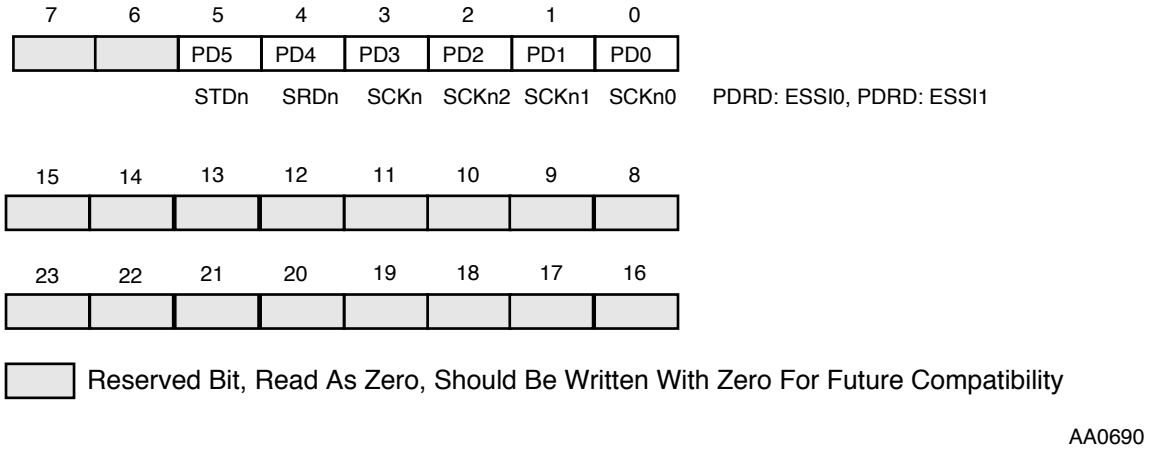


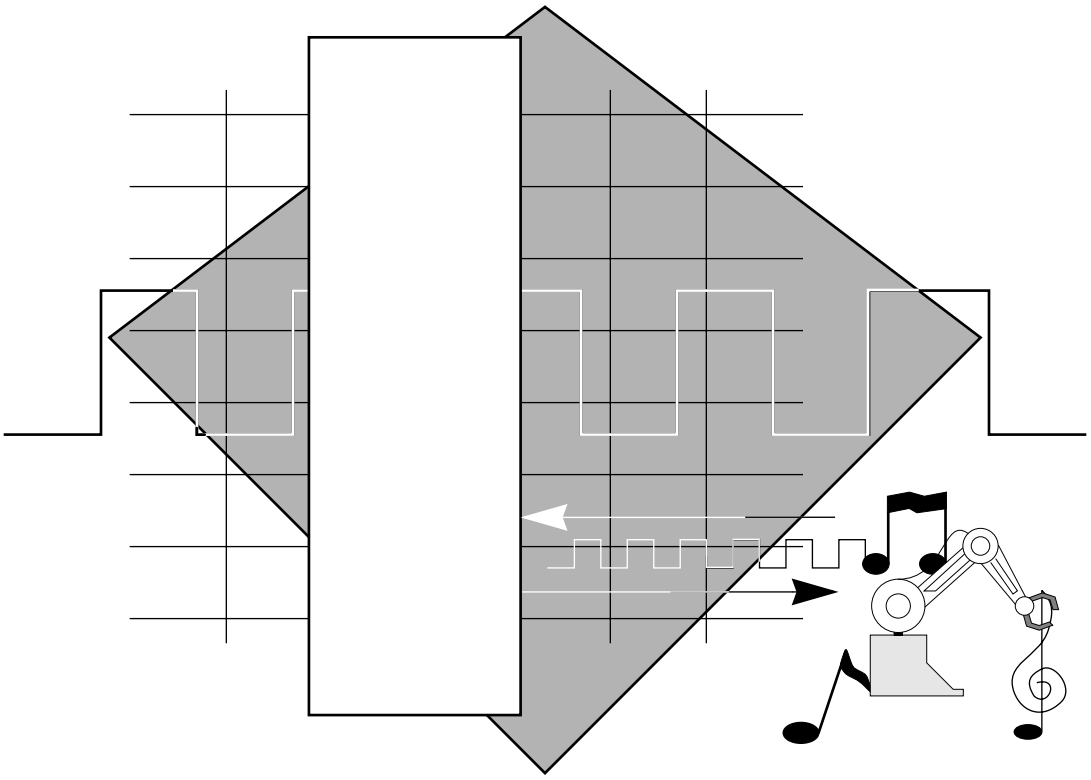
Figure 7-20 Port Data Register (PDR) (PDRC X:$FFFFBD)

(PDRD X:$FFFFAD)

Note: Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PDR bits.

SECTION 8

SERIAL COMMUNICATION INTERFACE (SCI)



8.1 INTRODUCTION

The DSP56309 serial communication interface (SCI) provides a full-duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems. The SCI interfaces without additional logic to peripherals that use TTL-level signals. With a small amount of additional logic, the SCI can connect to peripheral interfaces that have non-TTL level signals, such as the RS232C, RS422, etc.

This interface uses three dedicated signals: transmit data (TXD), receive data (RXD), and SCI serial clock (SCLK). It supports industry-standard asynchronous bit rates and protocols, as well as high-speed synchronous data transmission. The asynchronous protocols supported by the SCI include a multidrop mode for master/slave operation with wakeup on idle line and wakeup on address bit capability. This mode allows the DSP56309 to share a single serial line efficiently with other peripherals.

The SCI consists of separate transmit and receive sections that can operate asynchronously with respect to each other. A programmable baud-rate generator provides the transmit and receive clocks. An enable vector and an interrupt vector have been included so that the baud-rate generator can function as a general purpose timer when it is not being used by the SCI, or when the interrupt timing is the same as that used by the SCI.

8.2 SCI I/O SIGNALS

Each of the three SCI signals (RXD, TXD, and SCLK) can be configured as either a GPIO signal or as a specific SCI signal. Each signal is independent of the others. For example, if only the TXD signal is needed, the RXD and SCLK signals can be programmed for GPIO. However, at least one of the three signals must be selected as an SCI signal to release the SCI from reset.

SCI interrupts can be enabled by programming the SCI control registers before any of the SCI signals are programmed as SCI functions. In this case, only one transmit interrupt can be generated because the transmit data register is empty. The timer and timer interrupt operate regardless of how the SCI pins are configured—either as SCI or GPIO.

8.2.1 Receive Data (RXD)

This input signal receives byte-oriented serial data and transfers the data to the SCI receive shift register. Asynchronous input data is sampled on the positive edge of the receive clock (1 × SCLK) if SCKP is cleared. RXD can be configured as a GPIO signal (PE0) when the SCI RXD function is not being used.

8.2.2 Transmit Data (TXD)

This output signal transmits serial data from the SCI transmit shift register. Data changes on the negative edge of the asynchronous transmit clock (SCLK) if SCKP is cleared. This output is stable on the positive edge of the transmit clock. TXD can be programmed as a GPIO signal (PE1) when the SCI TXD function is not being used.

8.2.3 SCI Serial Clock (SCLK)

This bidirectional signal provides an input or output clock from which the transmit and/or receive baud rate is derived in asynchronous mode and from which data is transferred in synchronous mode. SCLK can be programmed as a GPIO signal (PE2) when the SCI SCLK function is not being used. This signal can be programmed as PE2 when data is being transmitted on TXD, since the clock does not need to be transmitted in asynchronous mode. Because SCLK is independent of SCI data I/O, there is no connection between programming the PE2 signal as SCLK and data coming out the TXD signal.

8.3 SCI PROGRAMMING MODEL

The SCI programming model can be viewed as three types of registers:

- Control
  - SCI control register (SCR) in **Figure 8-1**
  - SCI clock control register (SCCR) in **Figure 8-3**
- Status
  - SCI status register (SSR) in **Figure 8-2**
- Data transfer

– SCI Receive Data Registers (SRX) in **Figure 8-7** on page 8-19

– SCI Transmit Data Registers (STX) in **Figure 8-7**

– SCI Transmit Data Address Register (STXA) in **Figure 8-7**

The SCI also supports the GPIO functions documented in **Section 8—GPIO Signals and Registers** on page 8-27. The following paragraphs describe each bit in the programming model. Beginning on page 8-6, **Figure 8-4** shows the formats of data words.

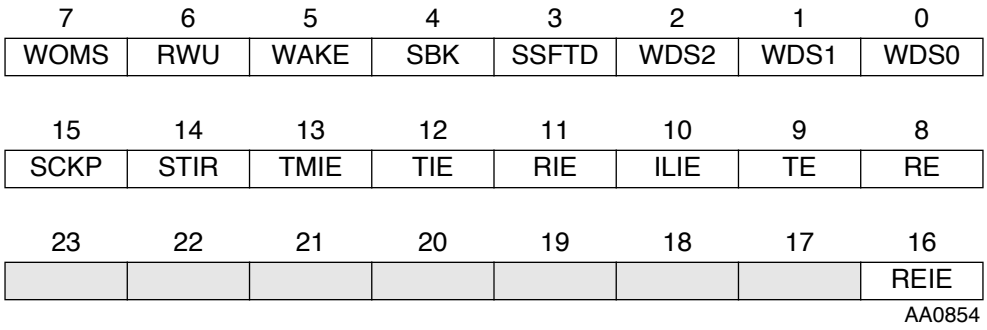


Figure 8-1 SCI Control Register (SCR)

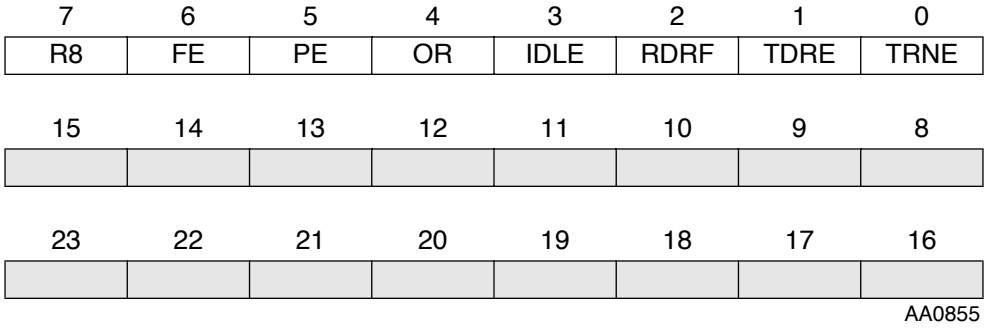


Figure 8-2 SCI Status Register (SSR)

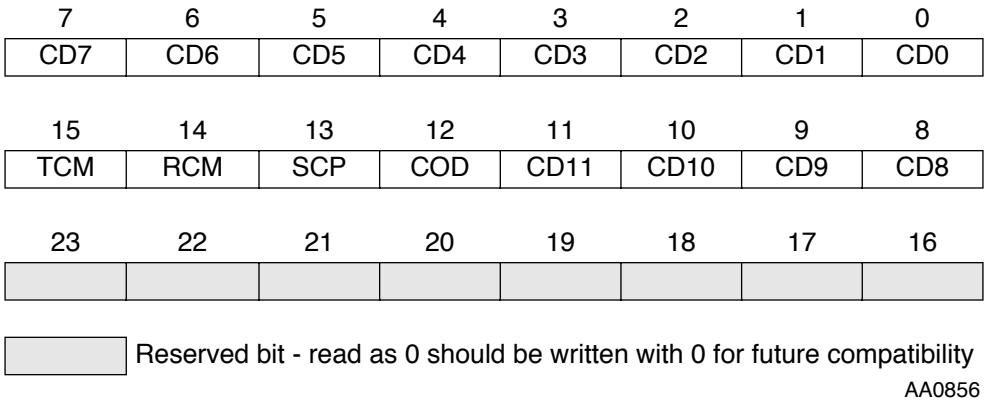


Figure 8-3 SCI Clock Control Register (SCCR)

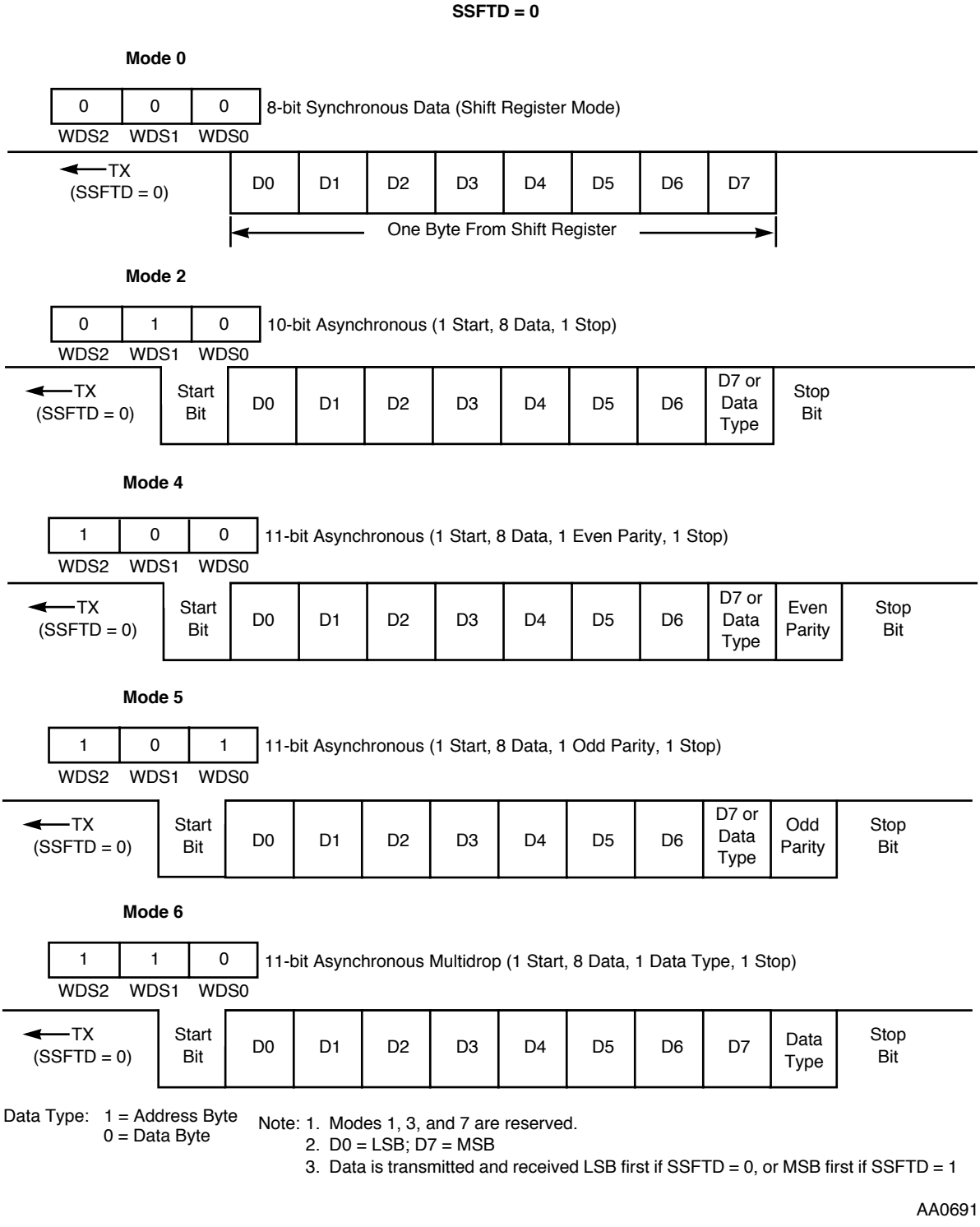


Figure 8-4 SCI Data Word Formats

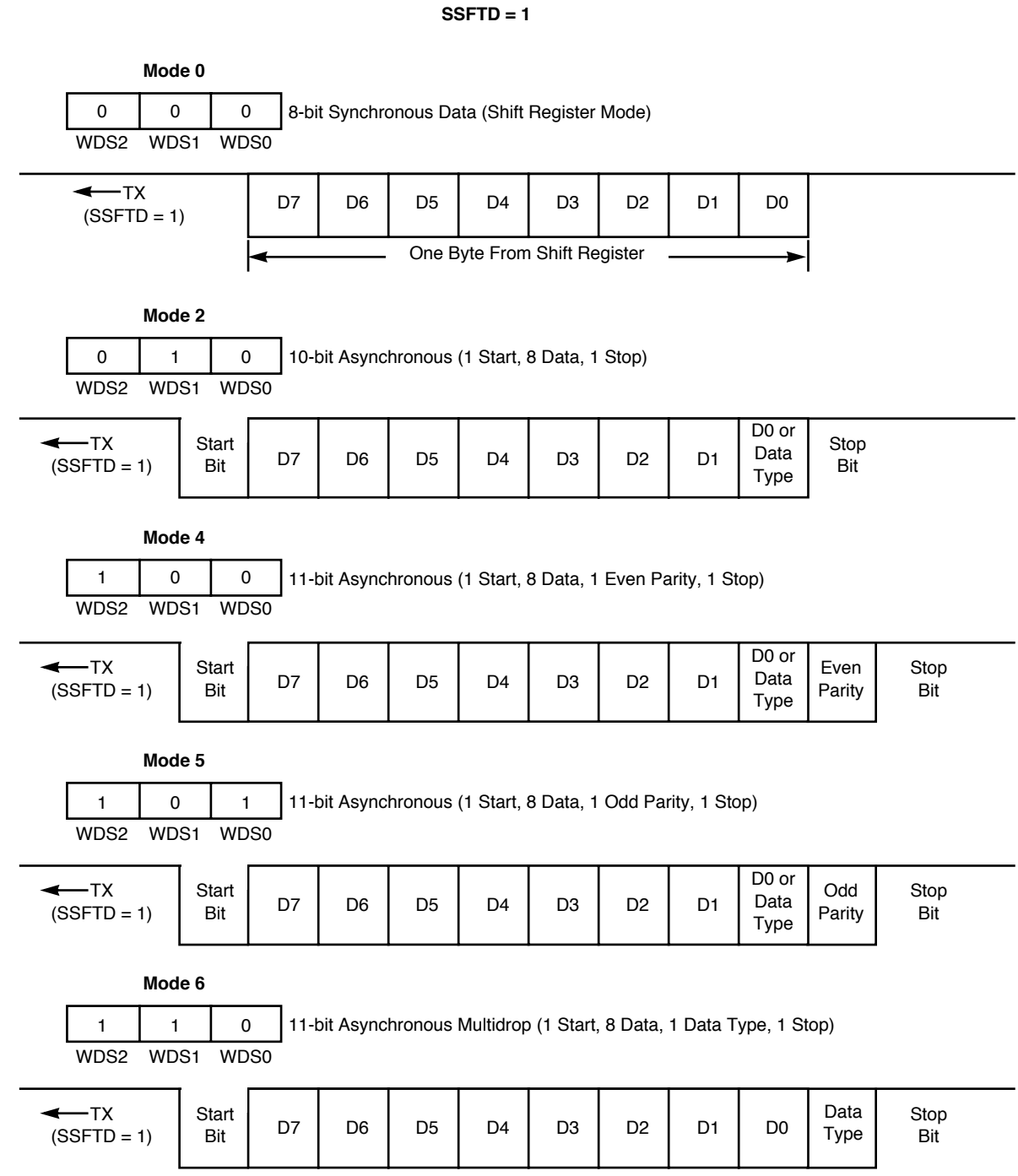


Figure 8-4 SCI Data Word Formats (Continued)

8.3.1 SCI Control Register (SCR)

The SCR is a 24-bit, read/write register that controls the serial interface operation. Seventeen of the twenty-four bits are currently defined. Each bit is described in the following paragraphs.

8.3.1.1 SCR Word Select (WDS[0:2]) Bits 0–2

The word select WDS[0:2] bits select the format of transmitted and received data. Format modes are listed in **Table 8-1** below and shown in **Figure 8-4** on page 8-6.

Table 8-1 Word Formats

WDS2	WDS1	WDS0	Mode	Word Formats
0	0	0	0	8-bit synchronous data (shift register mode)
0	0	1	1	Reserved
0	1	0	2	10-bit asynchronous (1 start, 8 data, 1 stop)
0	1	1	3	Reserved
1	0	0	4	11-bit asynchronous (1 start, 8 data, 1 even parity, 1 stop)
1	0	1	5	11-bit asynchronous (1 start, 8 data, 1 odd parity, 1 stop)
1	1	0	6	11-bit multidrop asynchronous (1 start, 8 data, 1 data type, 1 stop)
1	1	1	7	Reserved

Asynchronous modes are compatible with most UART-type serial devices and support standard RS232C communication links. Multidrop asynchronous mode is compatible with the MC68681 DUART, the M68HC11 SCI interface, and the Intel 8051 serial interface. Synchronous data mode is essentially a high-speed shift register used for I/O expansion and stream-mode channel interfaces. A gated transmit and receive clock compatible with the Intel 8051 serial interface mode 0 makes it possible for you to synchronize data.

When odd parity is selected, the transmitter counts the number of 1s in the data word. If the total is not an odd number, the parity bit is set, thus producing an odd number. If the receiver counts an even number of 1s, an error in transmission has occurred. When even parity is selected, an even number must result from the calculation performed at both ends of the line, or an error in transmission has occurred.

The word select bits are cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

8.3.1.2 SCR SCI Shift Direction (SSFTD) Bit 3

The SSFTD bit determines the order in which the SCI data shift registers shift data in or out: MSB first when set, LSB first when cleared. The parity and data type bits do not change their position in the frame; they remain adjacent to the stop bit. SSFTD is cleared by either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

8.3.1.3 SCR Send Break (SBK) Bit 4

A break is an all-zero word frame—a start bit 0, characters of all 0s (including any parity), and a stop bit 0 (i.e., ten or eleven 0s, depending on the mode selected). If SBK is set and then cleared, the transmitter completes transmission of the current frame, sends ten or eleven 0s (depending on WDS mode), and reverts to idle or sending data. If SBK remains set, the transmitter continually sends whole frames of 0s (ten or eleven bits with no stop bit). At the completion of the break code, the transmitter sends at least one high (set) bit before transmitting any data to guarantee recognition of a valid start bit. Break can be used to signal an unusual condition, message, etc. by forcing a frame error, which is caused by a missing stop bit. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears SBK.

8.3.1.4 SCR Wakeup Mode Select (WAKE) Bit 5

When WAKE is cleared, the wakeup on idle line mode is selected. In the wakeup on idle line mode, the SCI receiver is reenabled by an idle string of at least ten or eleven (depending on WDS mode) consecutive 1s. The transmitter's software must provide this idle string between consecutive messages. The idle string cannot occur within a valid message because each word frame contains a start bit that is 0.

When WAKE is set, the wakeup on address bit mode is selected. In the wakeup on address bit mode, the SCI receiver is reenabled when the last (eighth or ninth) data bit received in a character (frame) is 1. The ninth data bit is the address bit (R8) in the 11-bit multidrop mode; the eighth data bit is the address bit in the 10-bit asynchronous and 11-bit asynchronous with parity modes. Thus, the received character is an address that has to be processed by all sleeping processors—that is, each processor has to compare the received character with its own address and decide whether to receive or ignore all following characters. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears WAKE.

8.3.1.5 SCR Receiver Wakeup Enable (RWU) Bit 6

When RWU is set and the SCI is in an asynchronous mode, the wakeup function is enabled—that is, the SCI is asleep, and can be awakened by the event defined by the WAKE bit. In the sleep state, all interrupts and all receive flags except IDLE are disabled.

When the receiver wakes up, RWU is cleared by the wakeup hardware. The programmer can also clear the RWU bit to wake up the receiver.

RWU can be used by the programmer to ignore messages that are for other devices on a multidrop serial network. Wakeup on idle line (WAKE is cleared) or wakeup on address bit (WAKE is set) must be chosen.

1. When WAKE is cleared and RWU is set, the receiver does not respond to data on the data line until an idle line is detected.
2. When WAKE is set and RWU is set, the receiver does not respond to data on the data line until a data frame with the address bit set is detected.

When the receiver wakes up, the RWU bit is cleared, and the first frame of data is received. If interrupts are enabled, the CPU is interrupted and the interrupt routine reads the message header to determine if the message is intended for this DSP.

1. If the message is for this DSP, the message is received, and RWU is set to wait for the next message.
2. If the message is not for this DSP, the DSP immediately sets RWU. Setting RWU causes the DSP to ignore the remainder of the message and wait for the next message.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears RWU. RWU is ignored in synchronous mode.

8.3.1.6 SCR Wired-OR Mode Select (WOMS) Bit 7

When the WOMS bit is set, the SCI TXD driver is programmed to function as an open-drain output and can be wired together with other TXD signals in an appropriate bus configuration, such as a master-slave multidrop configuration. An external pullup resistor is required on the bus. When the WOMS is cleared, the TXD signal uses an active internal pullup. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears WOMS.

8.3.1.7 SCR Receiver Enable (RE) Bit 8

When RE is set, the receiver is enabled. When RE is cleared, the receiver is disabled, and data transfer from the receive shift register to the receive data register (SRX) is inhibited. If RE is cleared while a character is being received, the reception of the character is completed before the receiver is disabled. RE does not inhibit RDRF or receive interrupts. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears RE.

8.3.1.8 SCR Transmitter Enable (TE) Bit 9

When TE is set, the transmitter is enabled. When TE is cleared, the transmitter completes transmission of data in the SCI Transmit Data Shift Register, then the serial output is

forced high (i.e., idle). Data present in the SCI transmit data register (STX) is not transmitted. STX can be written and TDRE cleared, but the data is not transferred into the shift register. TE does not inhibit TDRE or transmit interrupts. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears TE.

Setting TE causes the transmitter to send a preamble of ten or eleven consecutive 1s (depending on WDS). This procedure gives the programmer a convenient way to insure that the line goes idle before starting a new message. To force this separation of messages by the minimum idle line time, the following sequence is recommended:

1. Write the last byte of the first message to STX.
2. Wait for TDRE to go high, indicating the last byte has been transferred to the transmit shift register.
3. Clear TE and set TE. This queues an idle line preamble to follow immediately the transmission of the last character of the message (including the stop bit).
4. Write the first byte of the second message to STX.

In this sequence, if the first byte of the second message is not transferred to STX prior to the finish of the preamble transmission, the transmit data line marks idle until STX is finally written.

8.3.1.9 SCR Idle Line Interrupt Enable (ILIE) Bit 10

When ILIE is set, the SCI interrupt occurs when IDLE (SCI status register bit 3) is set. When ILIE is cleared, the IDLE interrupt is disabled. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears ILIE.

An internal flag, the shift register idle interrupt (SRIINT) flag, is the interrupt request to the interrupt controller. SRIINT is not directly accessible to the user.

When a valid start bit has been received, an idle interrupt is generated if both IDLE and ILIE are set. The idle interrupt acknowledge from the interrupt controller clears this interrupt request. The idle interrupt is not asserted again until at least one character has been received. The results are as follows:

1. The IDLE bit shows the real status of the receive line at all times.
2. An idle interrupt is generated once for each idle state, no matter how long the idle state lasts.

8.3.1.10 SCR SCI Receive Interrupt Enable (RIE) Bit 11

The RIE bit is set to enable the SCI Receive Data interrupt. If RIE is cleared, the Receive Data interrupt is disabled, and then the RDRF bit in the SCI Status Register must be

polled to determine if the receive data register is full. If both RIE and RDRF are set, the SCI requests an SCI receive data interrupt from the interrupt controller.

Receive interrupts with exception have higher priority than normal receive data interrupts. Therefore, if an exception occurs (i.e., if PE, FE, or OR are set) and REIE is set, the SCI requests an SCI receive data with exception interrupt from the interrupt controller. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears RIE.

8.3.1.11 SCR SCI Transmit Interrupt Enable (TIE) Bit 12

The TIE bit is set to enable the SCI transmit data interrupt. If TIE is cleared, transmit data interrupts are disabled, and the transmit data register empty (TDRE) bit in the SCI status register must be polled to determine if the transmit data register is empty. If both TIE and TDRE are set, the SCI requests an SCI transmit data interrupt from the interrupt controller. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears TIE.

8.3.1.12 SCR Timer Interrupt Enable (TMIE) Bit 13

The TMIE bit is set to enable the SCI timer interrupt. If TMIE is set, timer interrupt requests are sent to the interrupt controller at the rate set by the SCI clock register. The timer interrupt is automatically cleared by the timer interrupt acknowledge from the interrupt controller. This feature allows DSP programmers to use the SCI baud rate generator as a simple periodic interrupt generator if the SCI is not in use, if external clocks are used for the SCI, or if periodic interrupts are needed at the SCI baud rate. The SCI internal clock is divided by 16 (to match the 1 × SCI baud rate) for timer interrupt generation. This timer does not require that any SCI signals be configured for SCI use to operate. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears TMIE.

8.3.1.13 SCR Timer Interrupt Rate (STIR) Bit 14

The STIR bit controls a divide-by-32 in the SCI Timer interrupt generator. When STIR is cleared, the divide-by-32 is inserted in the chain. When STIR is set, the divide-by-32 is bypassed, thereby increasing timer resolution by a factor of 32. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears this bit. To insure proper operation of the timer, STIR must not be changed during timer operation (i.e., if TMIE = 1).

8.3.1.14 SCR SCI Clock Polarity (SCKP) Bit 15

The SCKP bit controls the clock polarity sourced or received on the clock signal (SCLK), eliminating the need for an external inverter. When SCKP is cleared, the clock polarity is positive. When SCKP is set, the clock polarity is negative. In synchronous mode, positive polarity means that the clock is normally positive and transitions negative during valid data. Negative polarity means that the clock is normally negative and transitions positive during valid data. In asynchronous mode, positive polarity means that the rising edge of the clock occurs in the center of the period that data is valid. Negative polarity means that the falling edge of the clock occurs during the center of the period

that data is valid. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears SCKP.

8.3.1.15 Receive with Exception Interrupt Enable (REIE) Bit 16

The REIE bit is set to enable the SCI receive data with exception interrupt. If REIE is cleared, the receive data with exception interrupt is disabled. If both REIE and RDRF are set, and PE, FE, and OR are not all cleared, the SCI requests an SCI receive data with exception interrupt from the interrupt controller. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears REIE.

8.3.2 SCI Status Register (SSR)

The SSR is a 24-bit, read-only register used by the DSP to determine the status of the SCI. The status bits are described in the following paragraphs.

8.3.2.1 SSR Transmitter Empty (TRNE) Bit 0

The TRNE flag bit is set when both the transmit shift register and transmit data register (STX) are empty to indicate that there is no data in the transmitter. When TRNE is set, data written to one of the three STX locations or to the transmit data address register (STXA) is transferred to the transmit shift register and is the first data transmitted. TRNE is cleared when TDRE is cleared by writing data into the STX or the STXA, or when an idle, preamble, or break is transmitted. This bit, when set, indicates that the transmitter is empty; therefore, the data written to STX or STXA is transmitted next. That is, there is no word in the transmit shift register presently being transmitted. This procedure is useful when initiating the transfer of a message (i.e., a string of characters). TRNE is set by a hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction.

8.3.2.2 SSR Transmit Data Register Empty (TDRE) Bit 1

The TDRE flag bit is set when the SCI transmit data register is empty. When TDRE is set, new data can be written to one of the SCI transmit data registers (STX) or the transmit data address register (STXA). TDRE is cleared when the SCI transmit data register is written. TDRE is set by the hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction.

In synchronous mode, when using the internal SCI clock, there is a delay of up to 5.5 serial clock cycles between the time that STX is written until TDRE is set, indicating the data has been transferred from the STX to the transmit shift register. There is a 2 to 4 serial clock cycle delay between writing STX and loading the transmit shift register; in addition, TDRE is set in the middle of transmitting the second bit. When using an external serial transmit clock, if the clock stops, the SCI transmitter stops. TDRE is not set

until the middle of the second bit transmitted after the external clock starts. Gating the external clock off after the first bit has been transmitted delays TDRE indefinitely.

In asynchronous mode, the TDRE flag is not set immediately after a word is transferred from the STX or STXA to the transmit shift register nor when the word first begins to be shifted out. TDRE is set two cycles of the 16 × clock after the start bit— that is, two 16 × clock cycles into the transmission time of the first data bit.

8.3.2.3 SSR Receive Data Register Full (RDRF) Bit 2

The RDRF bit is set when a valid character is transferred to the SCI Receive Data Register from the SCI receive shift register (regardless of the error bits condition). RDRF is cleared when the SCI receive data register is read or by a hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction.

8.3.2.4 SSR Idle Line Flag (IDLE) Bit 3

IDLE is set when 10 (or 11) consecutive 1s are received. IDLE is cleared by a start-bit detection. The IDLE status bit represents the status of the receive line. The transition of IDLE from 0 to 1 can cause an IDLE interrupt (ILIE). IDLE is cleared by a hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction.

8.3.2.5 SSR Overrun Error Flag (OR) Bit 4

The OR flag bit is set when a byte is ready to be transferred from the receive shift register to the receive data register (SRX) that is already full (RDRF = 1). The receive shift register data is not transferred to the SRX. The OR flag indicates that character(s) in the received data stream may have been lost. The only valid data is located in the SRX. OR is cleared when the SCI Status Register is read, followed by a read of SRX. The OR bit clears the FE and PE bits—that is, an overrun error has higher priority than FE or PE. OR is cleared by a hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction.

8.3.2.6 SSR Parity Error (PE) Bit 5

In the 11-bit asynchronous modes, the PE bit is set when an incorrect parity bit has been detected in the received character. It is set simultaneously with RDRF for the byte which contains the parity error—that is, when the received word is transferred to the SRX. If PE is set, further data transfer into the SRX is not inhibited. PE is cleared when the SCI status register is read, followed by a read of SRX. PE is also cleared by a hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction. In 10-bit asynchronous mode, 11-bit multidrop mode, and 8-bit synchronous mode, the PE bit is always cleared since there is no parity bit in these modes. If the byte received causes both parity and overrun errors, the SCI receiver recognizes only the overrun error.

8.3.2.7 SSR Framing Error Flag (FE) Bit 6

The FE bit is set in asynchronous modes when no stop bit is detected in the data string received. FE and RDRE are set simultaneously when the received word is transferred to the SRX. However, the FE flag inhibits further transfer of data into the SRX until it is cleared. FE is cleared when the SCI status register is read followed by reading the SRX. A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction also clears FE. In 8-bit synchronous mode, FE is always cleared. If the byte received causes both framing and overrun errors, the SCI receiver recognizes only the overrun error.

8.3.2.8 SSR Received Bit 8 (R8) Address Bit 7

In 11-bit asynchronous multidrop mode, the R8 bit is used to indicate whether the received byte is an address or data. R8 is set for addresses and is cleared for data. R8 is not affected by reading the SRX or SCI status register. A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, SCI individual reset, or STOP instruction also clears R8.

8.3.3 SCI Clock Control Register (SCCR)

The SCCR is a 24-bit, read/write register that controls the selection of the clock modes and baud rates for the transmit and receive sections of the SCI interface. The control bits are described in the following paragraphs. The SCCR is cleared by a hardware $\overline{\text{RESET}}$ signal. The basic features of the clock generator (as in **Figure 8-5** on page 8-16 and **Figure 8-6** on page 8-18) are these:

- The SCI logic always uses a 16 × internal clock in asynchronous modes and always uses a 2 × internal clock in synchronous mode. The maximum internal clock available to the SCI peripheral block is the oscillator frequency divided by 4.
- The 16 × clock is necessary for asynchronous modes to synchronize the SCI to the incoming data, as in **Figure 8-5**.
- For asynchronous modes, the user must provide a 16 × clock to use an external baud rate generator (i.e., SCLK input).
- For asynchronous modes, the user can select either 1 × or 16 × for the output clock when using internal TX and RX clocks (TCM = 0 and RCM = 0).
- When SCKP is cleared, the transmitted data on the TXD signal changes on the negative edge of the 1 × serial clock and is stable on the positive edge. When SCKP is set, the data changes on the positive edge and is stable on the negative edge.
- The received data on the RXD signal is sampled on the positive edge (if SCKP = 0) or on the negative edge (if SCKP = 1) of the 1 × serial clock.

- For asynchronous mode, the output clock is continuous.
- For synchronous mode, a 1 × clock is used for the output or input baud rate. The maximum 1 × clock is the crystal frequency divided by 8.
- For synchronous mode, the clock is gated.
- For synchronous mode, the transmitter and receiver are synchronous with each other.

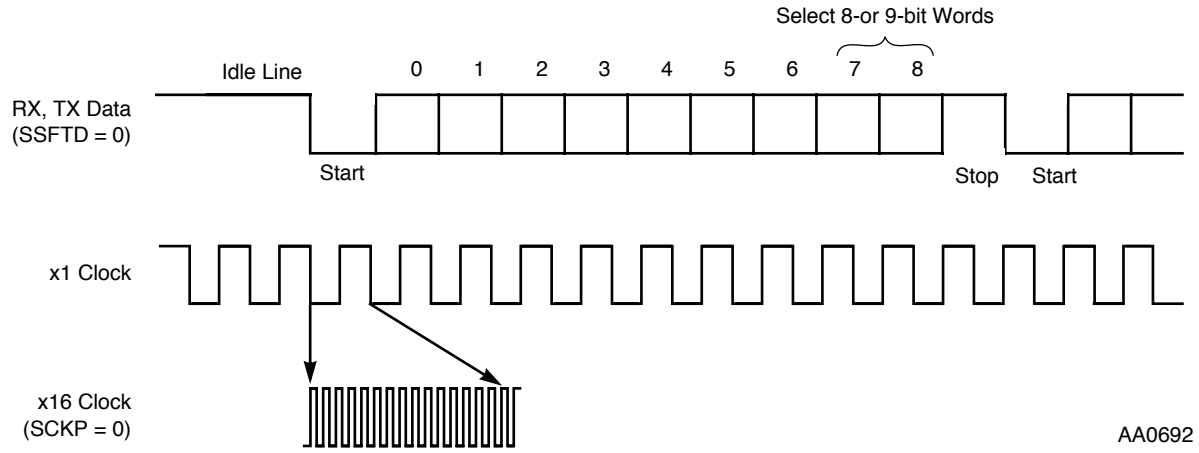


Figure 8-5 16 x Serial Clock

8.3.3.1 SCCR Clock Divider (CD[11:0]) Bits 11–0

The CD[11:0] bits specify the divide ratio of the prescale divider in the SCI clock generator. A divide ratio from 1 to 4096 (CD[11:0] = $000 to $FFF) can be selected. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears CD11–CD0.

8.3.3.2 SCCR Clock Out Divider (COD) Bit 12

The clock output divider is controlled by COD and SCI mode. If SCI mode is synchronous, the output divider is fixed at divide by 2.

If SCI mode is asynchronous, then one of the following conditions occurs:

- If COD is cleared and SCLK is an output (i.e., TCM and RCM are both cleared), the SCI clock is divided by 16 before being output to the SCLK signal. Thus, the SCLK output is a 1 × clock.
- If COD is set and SCLK is an output, the SCI clock is fed directly out to the SCLK signal. Thus, the SCLK output is a 16 × baud clock.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears COD.

8.3.3.3 SCCR SCI Clock Prescaler (SCP) Bit 13

The SCP bit selects a divide by 1 (SCP is cleared) or divide by 8 (SCP is set) prescaler for the clock divider. The output of the prescaler is further divided by 2 to form the SCI clock. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears SCP.

8.3.3.4 SCCR Receive Clock Mode Source (RCM) Bit 14

RCM selects whether an internal or external clock is used for the receiver. If RCM is cleared, the internal clock is used. If RCM is set, the external clock (from the SCLK signal) is used. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears RCM.

Table 8-2 TCM and RCM Bit Configuration

TCM	RCM	TX Clock	RX Clock	SCLK Signal	Mode
0	0	Internal	Internal	Output	Synchronous/Asynchronous
0	1	Internal	External	Input	Asynchronous Only
1	0	External	Internal	Input	Asynchronous Only
1	1	External	External	Input	Synchronous/Asynchronous

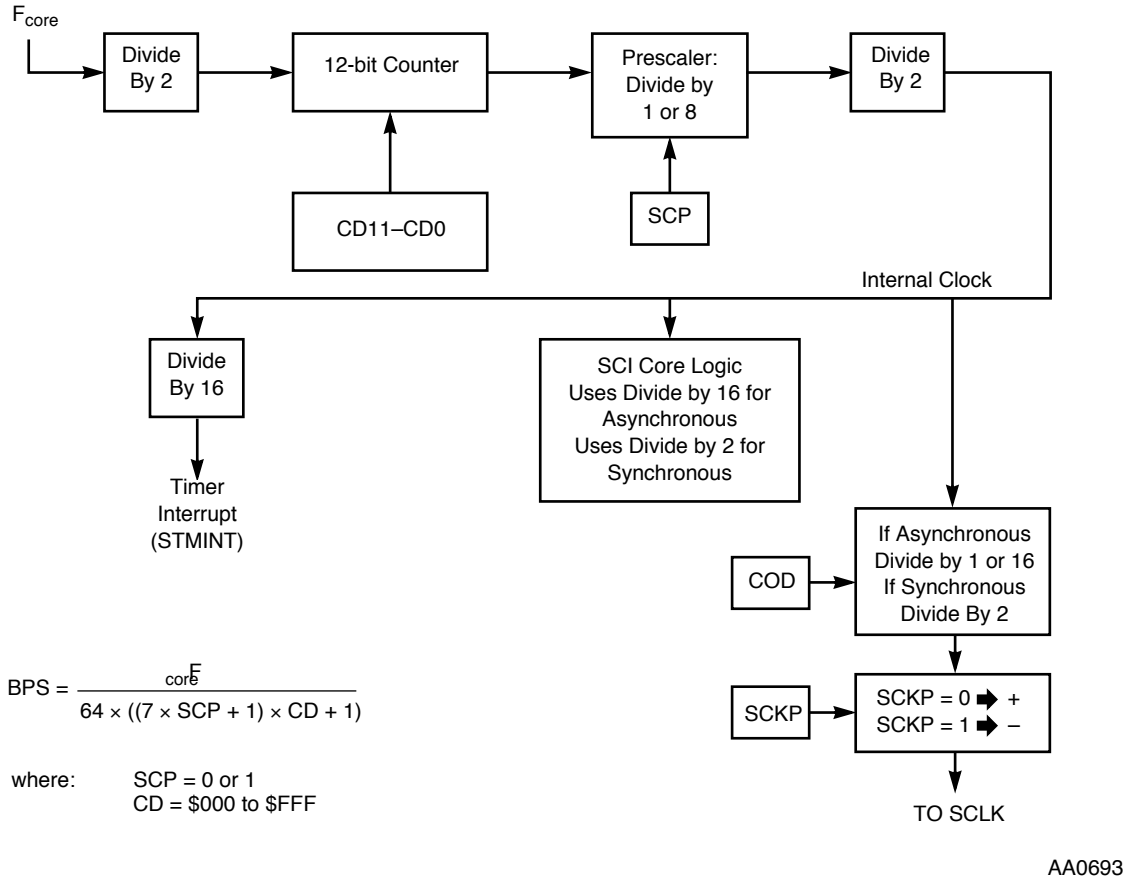


Figure 8-6 SCI Baud Rate Generator

8.3.3.5 SCCR Transmit Clock Source Bit (TCM) Bit 15

TCM selects whether an internal or external clock is used for the transmitter. If TCM is cleared, the internal clock is used. If TCM is set, the external clock (from the SCLK signal) is used. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears TCM.

8.3.4 SCI Data Registers

The SCI data registers are divided into two groups: receive and transmit (as in **Figure 8-7**). There are two receive registers—a receive data register (SRX) and a serial-to-parallel receive shift register. There are also two transmit registers—a transmit data register (called either STX or STXA) and a parallel-to-serial transmit shift register.

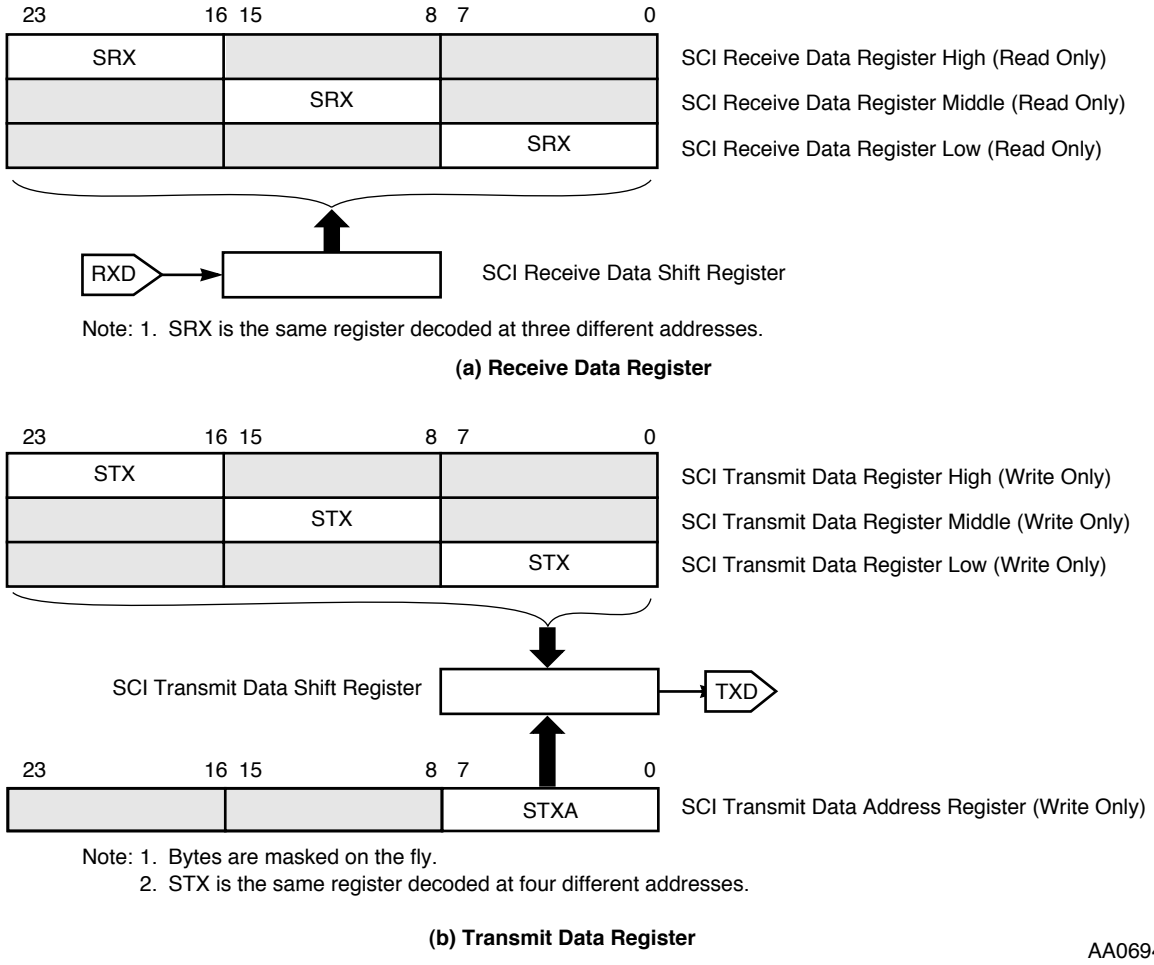


Figure 8-7 SCI Programming Model Data Registers

8.3.4.1 SCI Receive Registers (SRX)

Data bits received on the RXD signal are shifted into the SCI receive shift register. When a complete word has been received, the data portion of the word is transferred to the byte-wide SRX. This process converts the serial data to parallel data and provides double-buffering. Double-buffering provides flexibility to the programmer and increased throughput since the programmer can save (and process) the previous word while the current word is being received.

The SRX can be read at three locations as SRXL, SRXM, and SRXH. When SRXL is read, the contents of the SRX are placed in the lower byte of the data bus and the remaining bits on the data bus are read as 0s. Similarly, when SRXM is read, the contents of SRX are placed in the middle byte of the bus, and when SRXH is read, the contents of SRX are placed in the high byte with the remaining bits read as 0s. Mapping SRX as described allows three bytes to be efficiently packed into one 24-bit word by ORing three data bytes read from the three addresses.

The length and format of the serial word are defined by the WDS0, WDS1, and WDS2 control bits in the SCR. The clock source is defined by the receive clock mode (RCM) select bit in the SCR.

In synchronous mode, the start bit, the eight data bits, the address/data indicator bit and/or the parity bit, and the stop bit are received in that order. Data bits are sent LSB first if SSFTD is cleared, and MSB first if SSFTD is set. In synchronous mode, the synchronization is provided by gating the clock.

In either synchronous or asynchronous mode, when a complete word has been clocked in, the contents of the shift register can be transferred to the SRX and the flags; RDRF, FE, PE, and OR are changed appropriately. Because the operation of the receive shift register is transparent to the DSP, the contents of this register are not directly accessible to the programmer.

8.3.4.2 SCI Transmit Registers

The transmit data register is a one byte-wide register mapped into four addresses as STXL, STXM, STXH, and STXA. In asynchronous mode, when data is to be transmitted, STXL, STXM, and STXH are used. When STXL is written, the low byte on the data bus is transferred to the STX. When STXM is written, the middle byte is transferred to the STX. When STXH is written, the high byte is transferred to the STX. This structure makes it easy for the programmer to unpack the bytes in a 24-bit word for transmission. TDXA should be written in the 11-bit asynchronous multidrop mode when the data is an address and it is desired that the ninth bit (the address bit) be set. When STXA is written, the data from the low byte on the data bus is stored in it. The address data bit is cleared in the 11-bit asynchronous multidrop mode when any of STXL, STXM or STXH is written. When either STX (STXL, STXM, or STXH) or STXA is written, TDRE is cleared.

The transfer from either STX or STXA to the transmit shift register occurs automatically, but not immediately, when the last bit from the previous word has been shifted out; that is, the transmit shift register is empty. Like the receiver, the transmitter is double-buffered. However, a 2 to 4 serial clock cycle delay occurs between when the data is transferred from either STX or STXA to the transmit shift register and when the first bit appears on the TXD signal. (A serial clock cycle is the time required to transmit one data bit). The transmit shift register is not directly addressable, and a dedicated flag for this register does not exist. Because of this fact and the 2 to 4 cycle delay, two bytes cannot be written consecutively to STX or STXA without polling, as the second byte might overwrite the first byte. The TDRE flag should always be polled prior to writing STX or STXA to prevent overruns unless transmit interrupts have been enabled. Either STX or STXA is usually written as part of the interrupt service routine. An interrupt is generated only if TDRE is set. The transmit shift register is indirectly visible via the TRNE bit in the SSR.

In synchronous mode, data is synchronized with the transmit clock, which can have either an internal or external source, as defined by the TCM bit in the SCCR. The length and format of the serial word is defined by the WDS0, WDS1, and WDS2 control bits in the SCR. In asynchronous modes, the start bit, the eight data bits (with the LSB first if SSFTD = 0 and the MSB first if SSFTD = 1), the address/data indicator bit or parity bit, and the stop bit are transmitted in that order.

The data to be transmitted can be written to any one of the three STX addresses. If SCKP is set and SSHTD is set, the SCI synchronous mode is equivalent to the SSI operation in the 8-bit data on-demand mode.

Note: When writing data to a peripheral device, there is a two cycle pipeline delay until any status bits affected by this operation are updated. If you read any of those status bits within the next two cycles, the bit does not reflect its current status. See the DSP56300 Family Manual, Appendix B, Polling a Peripheral Device for Write for further details.

8.4 OPERATING MODES

The operating modes for the DSP56309 SCI are these:

- 8-bit synchronous (shift register mode)
- 10-bit asynchronous (1 start, 8 data, 1 stop)
- 11-bit asynchronous (1 start, 8 data, 1 even parity, 1 stop)
- 11-bit asynchronous (1 start, 8 data, 1 odd parity, 1 stop)
- 11-bit multidrop asynchronous (1 start, 8 data, 1 data type, 1 stop)
  This mode is used for master/slave operation with wakeup on idle line and wakeup on address bit capability. It allows the DSP56309 to share a single serial line efficiently with other peripherals.

These modes are selected using the WD[0:2] bits in the SCR.

The synchronous data mode is essentially a high-speed shift register used for I/O expansion and stream-mode channel interfaces. Data synchronization is accomplished by the use of a gated transmit and receive clock that is compatible with the Intel® 8051 serial interface mode 0.

Asynchronous modes are compatible with most UART-type serial devices. Standard RS232C communication links are supported by these modes.

The multidrop asynchronous modes are compatible with the MC68681 DUART, the M68HC11 SCI interface, and the Intel 8051 serial interface.

8.4.1 SCI After Reset

There are four different methods of resetting the SCI.

1. Hardware $\overline{\text{RESET}}$ signal
2. Software RESET instruction:
   Both hardware and software resets clear the port control register bits, which configure all I/O as GPIO input. The SCI remains in the reset state as long as all SCI signals are programmed as GPIO (PC2, PC1, and PC0 all are cleared); the SCI becomes active only when at least one of the SCI I/O signals is not programmed as GPIO.
3. Individual reset:
   During program execution, the PC2, PC1, and PC0 bits can be cleared (individual reset), which causes the SCI to stop serial activity and enter the reset state. All SCI status bits are set to their reset state. However, the contents of the SCR are not affected, allowing the DSP program to reset the SCI separately from the other internal peripherals. During individual reset, internal DMA accesses to the data registers of the SCI are not valid and the data read is unknown.
4. Stop processing state reset:
   Executing the STOP instruction halts operation of the SCI until the DSP is restarted, causing the SSR to be reset. No other SCI registers are affected by the STOP instruction.

Table 8-3 on page 8-23 illustrates how each type of reset affects each register in the SCI.

Table 8-3 SCI Registers after Reset

Register Bit	Bit Mnemonic	Bit Number	Reset Type			
			HW Reset	SW Reset	IR Reset	ST Reset
SCR	REIE	16	0	0	—	—
	SCKP	15	0	0	—	—
	STIR	14	0	0	—	—
	TMIE	13	0	0	—	—
	TIE	12	0	0	—	—
	RIE	11	0	0	—	—
	ILIE	10	0	0	—	—
	TE	9	0	0	—	—
	RE	8	0	0	—	—
	WOMS	7	0	0	—	—
	RWU	6	0	0	—	—
	WAKE	5	0	0	—	—
	SBK	4	0	0	—	—
	SSFTD	3	0	0	—	—
	WDS[2:0]	2–0	0	0	—	—
SSR	R8	7	0	0	0	0
	FE	6	0	0	0	0
	PE	5	0	0	0	0
	OR	4	0	0	0	0
	IDLE	3	0	0	0	0
	RDRF	2	0	0	0	0
	TDRE	1	1	1	1	1

Table 8-3 SCI Registers after Reset (Continued)

Register Bit	Bit Mnemonic	Bit Number	Reset Type			
			HW Reset	SW Reset	IR Reset	ST Reset
	TRNE	0	1	1	1	1
SCCR	TCM	15	0	0	—	—
	RCM	14	0	0	—	—
	SCP	13	0	0	—	—
	COD	12	0	0	—	—
	CD[11:0]	11–0	0	0	—	—
SRX	SRX [23:0]	23–16, 15–8, 7–0	—	—	—	—
STX	STX[23:0]	23–0	—	—	—	—
SRSH	SRS[8:0]	8–0	—	—	—	—
STSH	STS[8:0]	8–0	—	—	—	–

Note: SRSH—SCI Receive Shift Register, STSH — SCI Transmit Shift Register
HW—Hardware reset is caused by asserting the external $\overline{\text{RESET}}$ signal.
SW—Software reset is caused by executing the RESET instruction.
IR—Individual reset is caused by clearing PCRE (bits 0–2) (configured for GPIO).
ST—Stop reset is caused by executing the STOP instruction.
1—The bit is set during this reset.
0—The bit is cleared during this reset.
— — The bit is not changed during this reset

8.4.2 SCI Initialization

The correct way to initialize the SCI is as follows:

1. Send a hardware $\overline{\text{RESET}}$ signal or software RESET instruction.
2. Program SCI control registers.
3. Configure at least one SCI signal as other than GPIO.

If interrupts are to be used, the signals must be selected, and interrupts must be enabled and unmasked before the SCI can operate. The order does not matter; any one of these three requirements for interrupts can be used to enable the SCI.

Synchronous applications usually require exact frequencies, which require that the crystal frequency be chosen carefully. An alternative to selecting the system clock to accommodate the SCI requirements is to provide an external clock to the SCI.

8.4.3 SCI Initialization Example

One way to initialize the SCI is described here as an example.

1. The SCI should be in its individual reset state (PCR = $0).
2. Configure the control registers (SCR, SCCR) according to the operating mode, but do not enable either transmitter (TE = 0) or receiver (RE = 0).

   It is possible to set the interrupt enable bits that would be in use during the operation (no interrupt occurs).
3. Enable the SCI by setting the PCR bits according to which signals will be in use during operation.
4. If transmit interrupt is not used, write data to the transmitter.

   If transmitter interrupt enable is set, an interrupt is issued and the interrupt handler should write data into the transmitter.

   SCI transmit request is serviced by DMA channel if it is programmed to service the SCI transmitter.
5. Enable transmitters (TE = 1) and receiver (RE = 1), according to usage.

Operation starts as follows:

- For an internally generated clock, the SCLK signal starts operation immediately after the SCI is enabled (Step 3 above) for asynchronous modes. In synchronous mode, the SCLK signal is active only while transmitting (gated clock).
- Data is received only when the receiver is enabled (RE = 1) and after the occurrence of the SCI receive sequence on the RXD signal, as defined by the operating mode (i.e., idle line sequence).
- Data is transmitted only after the transmitter is enabled (TE = 1), and after transmitting the initialization sequence depending on the operating mode.

8.4.4 Preamble, Break, and Data Transmission Priority

Two or three transmission commands can be set simultaneously:

1. A preamble (TE is set.)
2. A break (SBK is set or is cleared.)
3. There is data for transmission (TDRE is cleared.)

After the current character transmission, if two or more of these commands are set, the transmitter executes them in the following order:

1. Preamble
2. Break
3. Data

8.4.5 SCI Exceptions

The SCI can cause five different exceptions in the DSP. These exceptions are as follows (ordered from the highest to the lowest priority):

1. SCI receive data with exception status is caused by receive data register full with a receiver error (parity, framing, or overrun error). Clearing the pending interrupt is done by reading the SCI status register, followed by a read of SRX. A long interrupt service routine should be used to handle the error condition. This interrupt is enabled by SCR bit 16 (REIE).
2. SCI receive data is caused by receive data register full. Reading SRX clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead. This interrupt is enabled by SCR bit 11 (RIE).
3. SCI transmit data is caused by transmit data register empty. Writing STX clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead. This interrupt is enabled by SCR bit 12 (TIE).
4. SCI idle line is caused by the receive line entering the idle state (10 or 11 bits of 1s). This interrupt is latched and then automatically reset when the interrupt is accepted. This interrupt is enabled by SCR bit 10 (ILIE).
5. SCI timer is caused by the baud rate counter reaching zero. This interrupt is automatically reset when the interrupt is accepted. This interrupt is enabled by SCR bit 13 (TMIE).

8.5 GPIO SIGNALS AND REGISTERS

The GPIO functionality of port SCI is controlled by three registers: Port E control register (PCRE), Port E direction register (PRRE) and Port E data register (PDRE).

8.5.1 Port E Control Register (PCRE)

The read/write, 24-bit PCRE controls the functionality of SCI GPIO signals. Each of the PC[2:0] bits controls the functionality of the corresponding port signal. When a PC[i] bit is set, the corresponding port signal is configured as a SCI signal. When a PC[i] bit is cleared, the corresponding port signal is configured as a GPIO signal. Bits in the Port E control register appear in **Figure 8-8**.

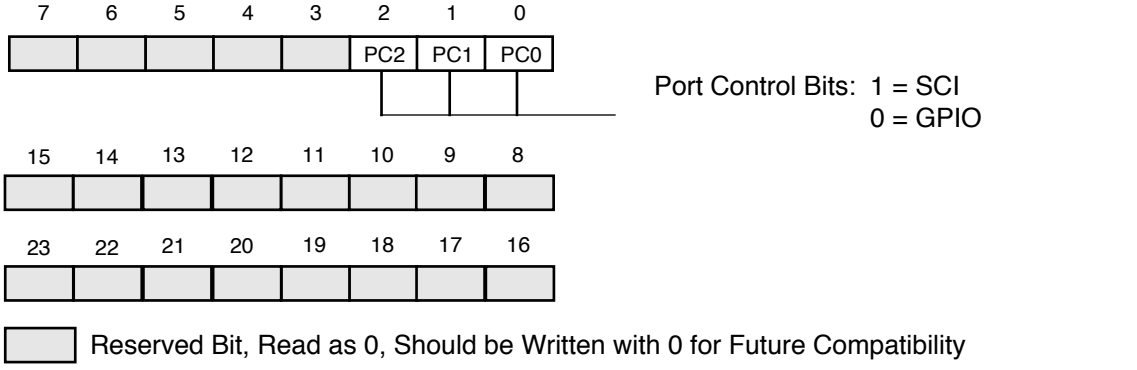


Figure 8-8 Port E Control Register (PCRE)

Note: A hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PCR bits.

8.5.2 Port E Direction Register (PRRE)

The read/write, 24-bit PRRE controls the direction of SCI GPIO signals. When port signal[i] is configured as GPIO, PDC[i] controls the port signal direction. When PDC[i] is set, the GPIO port signal[i] is configured as output. When PDC[i] is cleared, the GPIO port signal[i] is configured as input. Bits in the Port E direction register appear in **Figure 8-9**.

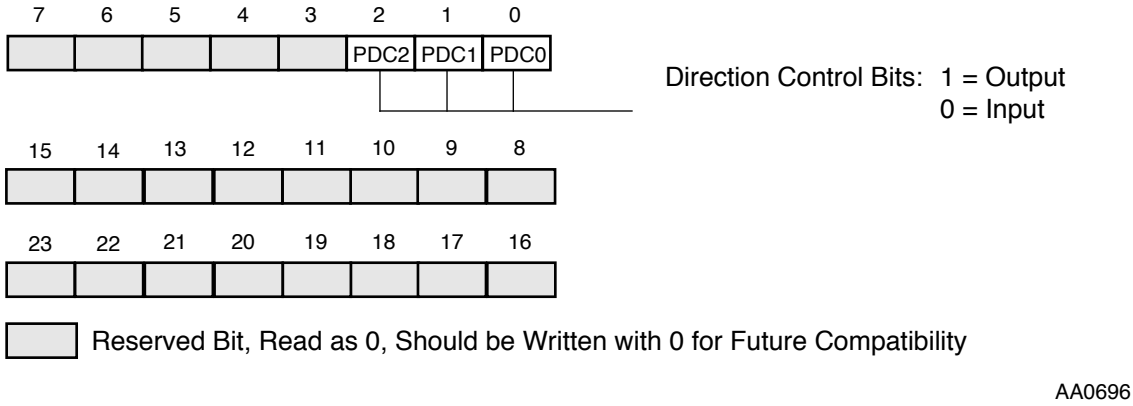


Figure 8-9 Port E Direction Register (PRRE)

Note: A hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PRR bits.

Table 8-4 shows the port signal configurations.

Table 8-4 Port Control Register and Port Direction Register Bits

PC[i]	PDC[i]	Port Signal[i] Function
1	1 or 0	SCI
0	0	GPIO input
0	1	GPIO output

8.5.3 Port E Data Register (PDRE)

The read/write, 24-bit PDRE is used to read or write data to or from SCI GPIO signals. Bits PD[2:0] are used to read or write data from or to the corresponding port signals if they are configured as GPIO. If a port signal[i] is configured as a GPIO input, then the corresponding PD[i] bit reflects the value of this signal. If a port signal[i] is configured as a GPIO output, then the value of the corresponding PD[i] bit is reflected on this signal. Bits of the Port E data register appear in **Figure 8-10**.

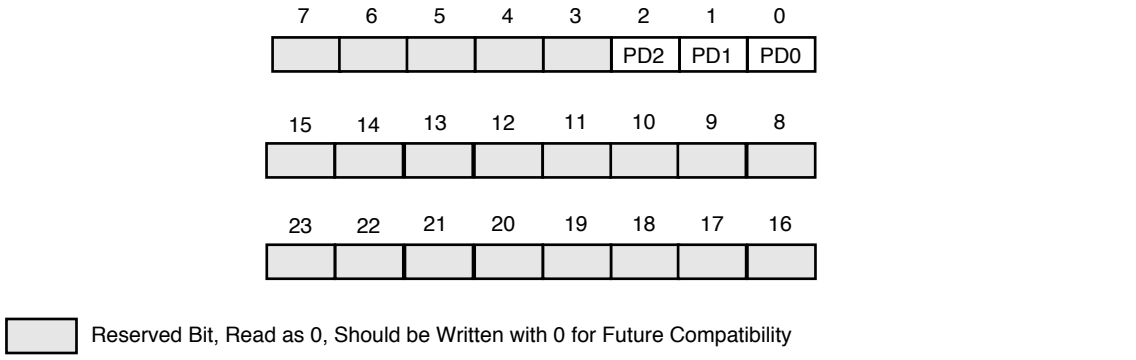
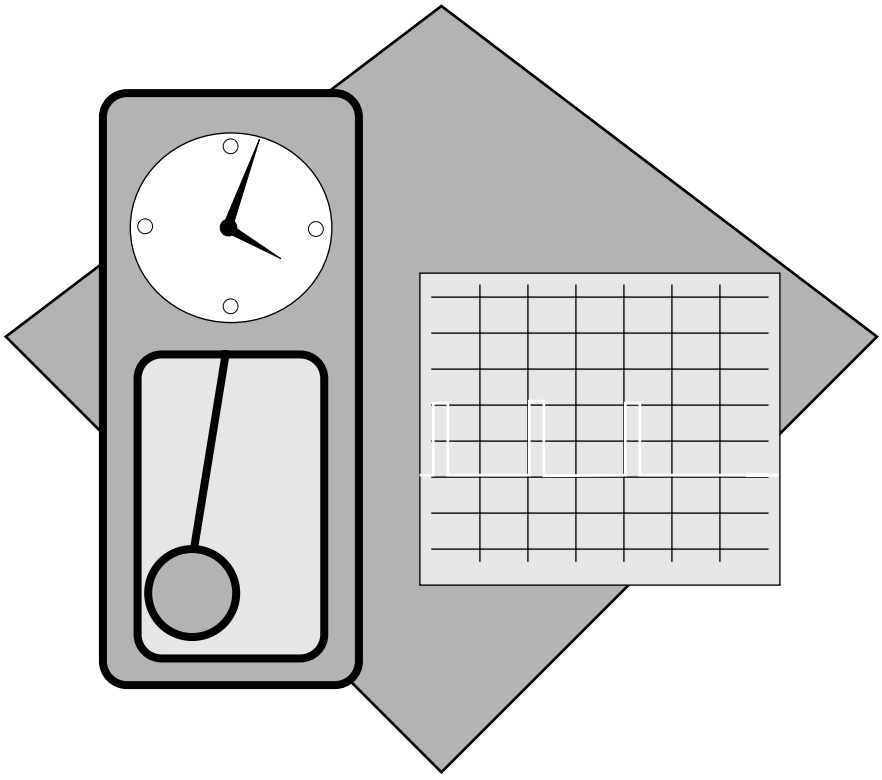


Figure 8-10 Port E Data Register (PDRE)

Note: A hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PDRE bits.

SECTION 9

TRIPLE TIMER MODULE



9.1 INTRODUCTION

This section describes the internal triple timer module in the DSP56309. Each timer has a single signal that can be used as a GPIO signal or as a timer signal. These three timers can be used to generate timed pulses or as pulse width modulators. They can also be used as an event counter, to capture an event, or to measure the width or period of a signal.

9.2 TRIPLE TIMER MODULE ARCHITECTURE

The timer module is composed of a common 21-bit prescaler and three independent and identical general purpose 24-bit timer/event counters, each having its own register set. Each timer can use internal or external clocking and can interrupt the DSP56309 after a specified number of events (clocks) or can signal an external device after counting internal events. Each timer can also be used to trigger DMA transfers after a specified number of events (clocks) has occurred. Each timer connects to the external world through one bidirectional signal, designated TIO0–TIO2 for Timers 0–2, respectively.

When the TIO signal is configured as input, the timer functions as an external event counter or measures external pulse width/signal period. When the TIO signal is used as output, the timer functions as a timer, a watchdog timer, or a pulse width modulator. When the TIO signal is not used by the timer, it can be used as a GPIO signal (also called TIO0–TIO2).

9.2.1 Triple Timer Module Block Diagram

Figure 9-1 shows a block diagram of the triple timer module. This module includes a 24-bit timer prescaler load register (TPLR), a 24-bit timer prescaler count register (TPCR), a 21-bit prescaler clock counter, and three timers. Each of the three timers can use the prescaler clock as its clock source.

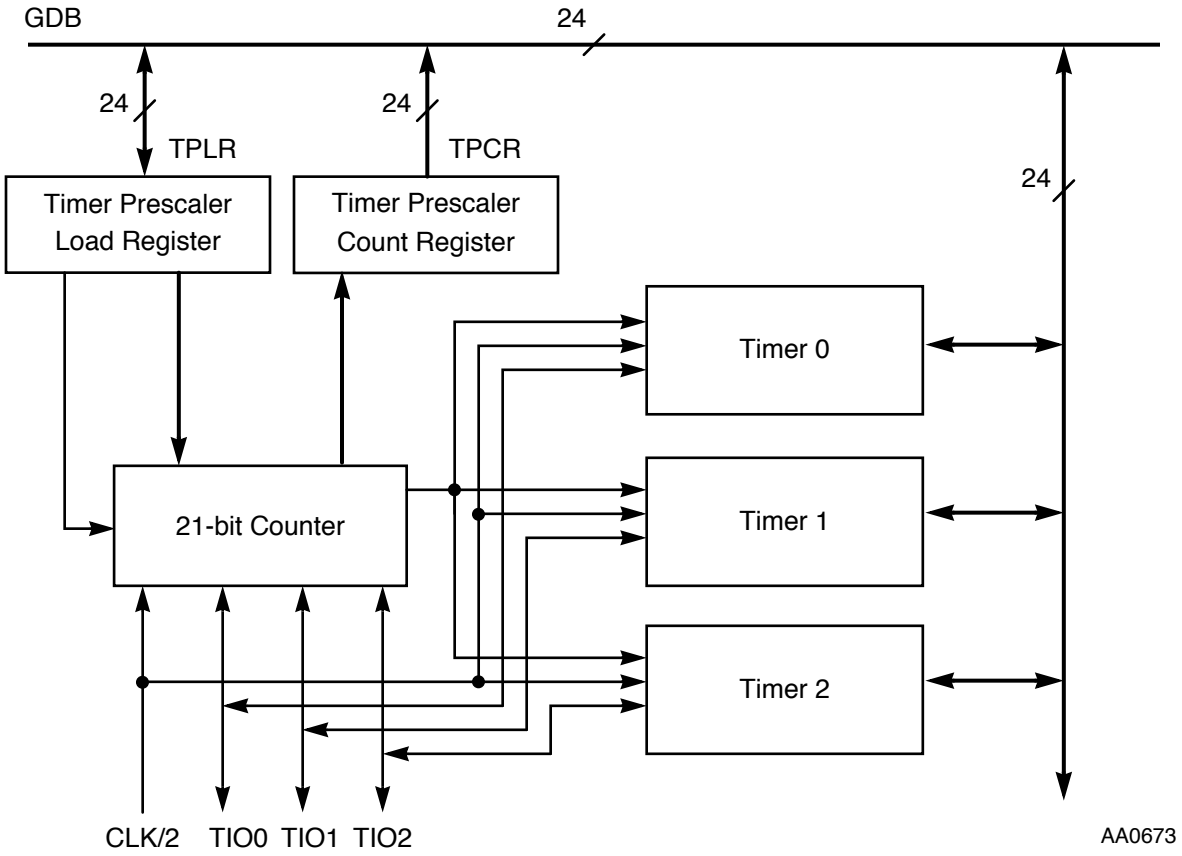


Figure 9-1 Triple Timer Module Block Diagram

9.2.2 Timer Block Diagram

The timer block diagram (in **Figure 9-2**) shows the structure of a timer module. The timer programmer's model (in **Figure 9-3** on page 9-6) shows the structure of the timer registers. The three timers are identical in structure and function. A generic timer is discussed in this section.

The timer includes a 24-bit counter, a 24-bit read/write timer control and status register (TCSR), a 24-bit read-only timer count register (TCR), a 24-bit write-only timer load register (TLR), a 24-bit read/write timer compare register (TCPR), and logic for clock selection and interrupt/DMA trigger generation.

The timer mode is controlled by the TC[3:0] bits of the timer control/status register (TCSR). For a listing of the timer modes, see **Section 9—Timer Operational Modes**. For a description of their operation, see **Section 9.4.1—Timing Modes**.

The DSP56309 views each timer as a memory-mapped peripheral with four registers occupying four 24-bit words in the X data memory space. Either standard polled or interrupt programming techniques can be used to service the timers. The timer programming model is shown in **Figure 9-3** on page 9-6.

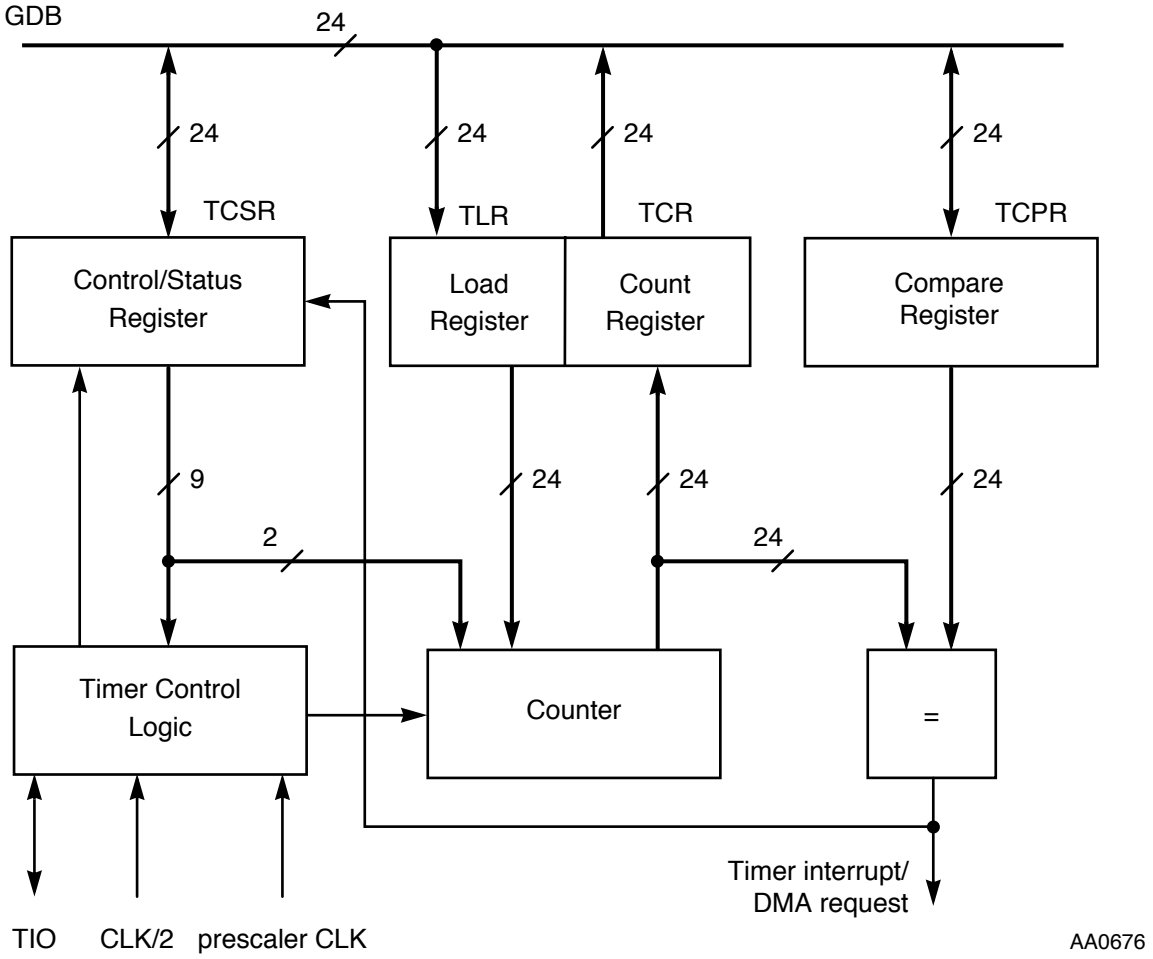


Figure 9-2 Timer Module Block Diagram

9.3 TRIPLE TIMER MODULE PROGRAMMING MODEL

The programming model for the triple timer module appears in **Figure 9-3** on page 9-6.

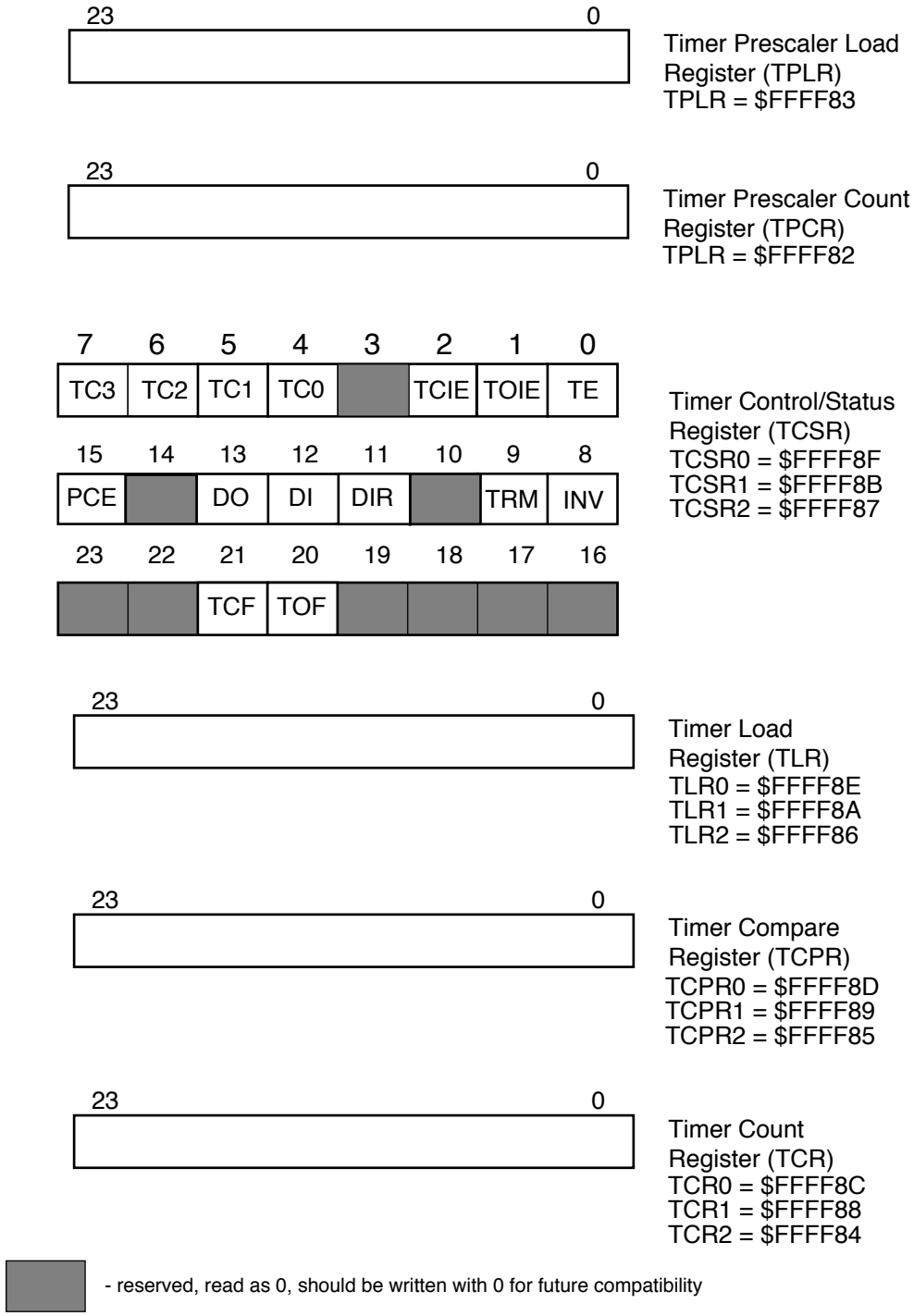


Figure 9-3 Timer Module Programmer's Model

9.3.1 Prescaler Counter

The prescaler counter is a 21-bit counter that is decremented on the rising edge of the prescaler input clock. The counter is enabled when at least one of the three timers is enabled (i.e., one or more of the timer enable (TE) bits are set) and is using the prescaler output as its source (i.e., one or more of the PCE bits are set).

9.3.2 Timer Prescaler Load Register (TPLR)

The TPLR is a 24-bit, read/write register that controls the prescaler divide factor (i. e., the number that the prescaler counter loads and begins counting from) and the source for the prescaler input clock. The control bits are shown below in **Figure 9-4**.

23	22	21	20	19	18	17	16	15	14	13	12
	PS1	PS0	PL20	PL19	PL18	PL17	PL16	PL15	PL14	PL13	PL12

11	10	9	8	7	6	5	4	3	2	1	0
PL11	PL10	PL9	PL8	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0

— reserved, read as 0, should be written with 0 for future compatibility

Figure 9-4 Timer Prescaler Load Register (TPLR)

9.3.2.1 TPLR Prescaler Preload Value (PL[20:0]) Bits 20-0

These 21 bits contain the prescaler preload value. This value is loaded into the prescaler counter when the counter value reaches 0, or the counter switches state from disabled to enabled.

If PL[20:0] = N, then the prescaler counts N + 1 source clock cycles before generating a prescaler clock pulse. Therefore, the prescaler divide factor = (preload value) + 1.

The PL[20:0] bits are cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

9.3.2.2 TPLR Prescaler Source (PS[1:0]) Bits 22-21

The two PS bits control the source of the prescaler clock. **Table 9-1** summarizes PS bit functionality. The prescaler's use of a TIO signal is not affected by the TCSR settings of the timer corresponding to the TIO signal being used.

If the prescaler source clock is external, the prescaler counter is incremented by signal transitions on the TIO signal. The external clock is internally synchronized to the internal clock. The external clock frequency must be lower than the DSP56309 internal operating frequency divided by 4 (CLK/4).

The PS[1:0] bits are cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Note: To insure proper operation, change the PS[1:0] bits only when the prescaler counter is disabled. Disable the prescaler counter by clearing the TE bit in the TCSR of each of three timers.

Table 9-1 Prescaler Source Selection

PS1	PS0	Prescaler Clock Source
0	0	Internal CLK/2
0	1	TIO0
1	0	TIO1
1	1	TIO2

9.3.2.3 TPLR Reserved Bit 23

This reserved bit is read as 0 and should be written with 0 for future compatibility.

9.3.3 Timer Prescaler Count Register (TPCR)

The TPCR is a 24-bit, read-only register that reflects the current value in the prescaler counter. The register bits are shown in **Figure 9-5**.

23	22	21	20	19	18	17	16	15	14	13	12
			PC20	PC19	PC18	PC17	PC16	PC15	PC14	PC13	PC12

11	10	9	8	7	6	5	4	3	2	1	0
PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

— reserved, read as 0, should be written with 0 for future compatibility

Figure 9-5 Timer Prescaler Count Register (TPCR)

9.3.3.1 TPCR Prescaler Counter Value (PC[20:0]) Bits 20-0

These 21 bits contain the current value of the prescaler counter.

9.3.3.2 TPCR Reserved Bits 23-21

These reserved bits are read as 0 and should be written with 0 for future compatibility.

9.3.4 Timer Control/Status Register (TCSR)

The TCSR is a 24-bit, read/write register controlling the timer and reflecting its status. The register bits are shown in **Figure 9-6**. The control and status bits are documented in **Table 9-2** on page 9-10.

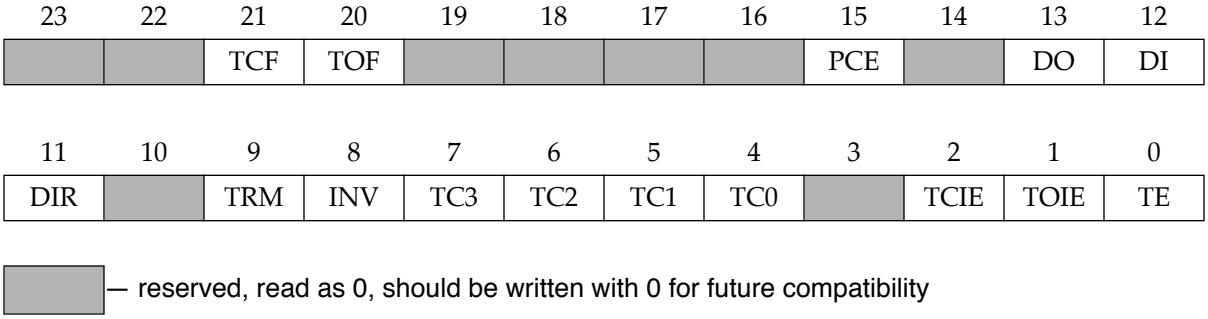


Figure 9-6 Timer Control/Status Register

9.3.4.1 Timer Enable (TE) Bit 0

The TE bit is used to enable or disable the timer. Setting TE enables the timer and clears the timer counter. The counter starts counting according to the mode selected by the timer control (TC[3:0]) bit values.

Clearing the TE bit disables the timer. The TE bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Note: When all three timers are disabled and the signals are not in GPIO mode, all three TIO signals are tri-stated. To prevent undesired spikes on the TIO signals when switching from tri-state into active state, these signals should be tied to the high or low signal state by the use of pull-up or pull-down resistors.

9.3.4.2 Timer Overflow Interrupt Enable (TOIE) Bit 1

The TOIE bit is used to enable the timer overflow interrupts. Setting TOIE enables overflow interrupt generation. The timer counter can hold a maximum value of $FFFFFF. When the counter value is at the maximum value and a new event causes the counter to be incremented to $000000, the timer generates an overflow interrupt.

Clearing the TOIE bit disables overflow interrupt generation. The TOIE bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

9.3.4.3 Timer Compare Interrupt Enable (TCIE) Bit 2

The TCIE bit is used to enable or disable the timer compare interrupts. Setting TCIE enables the compare interrupts. In the timer, pulse width modulation (PWM), or watchdog modes, a compare interrupt is generated after the counter value matches the value of the TCPR. The counter starts counting up from the number loaded from the TLR and if the TCPR value is N, an interrupt occurs after (N – M + 1) events, where M is the value of TLR.

Clearing the TCIE bit disables the compare interrupts. The TCIE bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

9.3.4.4 Timer Control (TC[3:0]) Bits 4-7

The four TC bits control the source of the timer clock, the behavior of the TIO signal, and timer mode. **Table 9-2** summarizes the TC bit functionality. There is a detailed description of the timer operating modes in **Section 9.4—Timer Operational Modes**.

The TC bits are cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Note: If the clock is external, the counter is incremented by the transitions on the TIO signal. The external clock is internally synchronized to the internal clock, and its frequency should be lower than the internal operating frequency divided by 4 (CLK/4).

Note: To insure proper operation, the TC[3:0] bits should be changed only when the timer is disabled (i.e., when the TE bit in the TCSR has been cleared).

Table 9-2 Timer Control Bits

Bit Settings				Mode Characteristics			
TC3	TC2	TC1	TC0	Mode Number	Mode Function	TIO	Clock
0	0	0	0	0	Timer and GPIO	GPIO [1]	Internal
0	0	0	1	1	Timer Pulse	Output	Internal
0	0	1	0	2	Timer Toggle	Output	Internal
0	0	1	1	3	Event Counter	Input	External

Table 9-2 Timer Control Bits (Continued)

Bit Settings				Mode Characteristics			
TC3	TC2	TC1	TC0	Mode Number	Mode Function	TIO	Clock
0	1	0	0	4	Input Width Measurement	Input	Internal
0	1	0	1	5	Input Period Measurement	Input	Internal
0	1	1	0	6	Capture Event	Input	Internal
0	1	1	1	7	Pulse Width Modulation (PWM)	Output	Internal
1	0	0	0	8	Reserved	—	—
1	0	0	1	9	Watchdog Pulse	Output	Internal
1	0	1	0	10	Watchdog Toggle	Output	Internal
1	0	1	1	11	Reserved	—	—
1	1	0	0	12	Reserved	—	—
1	1	0	1	13	Reserved	—	—
1	1	1	0	14	Reserved	—	—
1	1	1	1	15	Reserved	—	—

Note 1: The GPIO function is enabled only if all of the TC[3:0] bits are 0.

9.3.4.5 Inverter (INV) Bit 8

The Inverter (INV) bit affects the polarity definition of the incoming signal on the TIO signal when TIO is programmed as input. It also affects the polarity of the output pulse generated on the TIO signal when TIO is programmed as output.

The inverter bit operation is described below in **Table 9-3**.

Table 9-3 Inverter (INV) Bit Operation

Mode	TIO Programmed as Input		TIO Programmed as Output	
	INV = 0	INV = 1	INV = 0	INV = 1
0	GPIO signal on the TIO signal read directly	GPIO signal on the TIO signal inverted	Bit written to GPIO put on TIO signal directly	Bit written to GPIO inverted and put on TIO signal
1	Counter is incremented on the **rising** edge of the signal from the TIO signal	Counter is incremented on the **falling** edge of the signal from the TIO signal	—	—
2	Counter is incremented on the **rising** edge of the signal from the TIO signal	Counter is incremented on the **falling** edge of the signal from the TIO signal	TCRx output put on TIO signal directly	TCRx output inverted and put on TIO signal
3	Counter is incremented on the **rising** edge of the signal from the TIO signal	Counter is incremented on the **falling** edge of the signal from the TIO signal	—	—
4	Width of the **high** input pulse is measured	Width of the **low** input pulse is measured	—	—
5	Period is measured between the **rising** edges of the input signal	Period is measured between the **falling** edges of the input signal	—	—
6	Event is captured on the **rising** edge of the signal from the TIO signal	Event is captured on the **falling** edge of the signal from the TIO signal	—	—

Table 9-3 Inverter (INV) Bit Operation (Continued)

Mode	TIO Programmed as Input		TIO Programmed as Output	
	INV = 0	INV = 1	INV = 0	INV = 1
7	—	—	Pulse generated by the timer has **positive** polarity	Pulse generated by the timer has **negative** polarity
9	—	—	Pulse generated by the timer has **positive** polarity	Pulse generated by the timer has **negative** polarity
10	—	—	Pulse generated by the timer has **positive** polarity.	Pulse generated by the timer has **negative** polarity

The INV bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

The INV bit affects both the timer and GPIO modes. To insure correct operation, this bit should be changed only when one or both of the following conditions is true:

- The timer has been disabled by clearing the TE bit in the TCSR.
- The timer is in GPIO mode.

The INV bit does not affect the polarity of the prescaler source when the TIO is used as input to the prescaler.

9.3.4.6 Timer Reload Mode (TRM) Bit 9

The TRM bit controls the counter preload operation.

In timer (0–3) and watchdog (9–10) modes, the counter is preloaded with the TLR value after the TE bit is set and the first internal or external clock signal is received. If the TRM bit is set, the counter is reloaded each time after it reaches the value contained by the TCR. In PWM mode (7), the counter is reloaded each time counter overflow occurs. In measurement (4–5) modes, if the TRM and the TE bits are set, the counter is preloaded with the TLR value on each appropriate edge of the input signal.

If the TRM bit is cleared, the counter operates as a free running counter and is incremented on each incoming event. The TRM bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

9.3.4.7 Direction (DIR) Bit 11

The DIR bit determines the behavior of the TIO signal when it is used as a GPIO signal. When the DIR bit is set, the TIO signal is an output; when the DIR bit is cleared, the TIO signal is an input. The TIO signal can be used as a GPIO signal only when all of the TC[3:0] bits are cleared. If any of the TC[3:0] bits are set, then the GPIO function is disabled and the DIR bit has no effect.

The DIR bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

9.3.4.8 Data Input (DI) Bit 12

The DI bit reflects the value of the TIO signal. If the INV bit is set, the value of the TIO signal is inverted before it is written to the DI bit. If the INV bit is cleared, the value of the TIO signal is written directly to the DI bit.

DI is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

9.3.4.9 Data Output (DO) Bit 13

The DO bit is the source of the TIO value when it is a data output signal. The TIO signal is data output when GPIO mode is enabled and DIR is set. A value written to the DO bit is written to the TIO signal. If the INV bit is set, the value of the DO bit is inverted when written to the TIO signal. When the INV bit is cleared, the value of the DO bit is written directly to the TIO signal. When GPIO mode is disabled, writing the DO bit has no effect.

The DO bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

9.3.4.10 Prescaler Clock Enable (PCE) Bit 15

The PCE bit is used to select the prescaler clock as the timer source clock. When the PCE bit is cleared, the timer uses either an internal (CLK/2) signal or an external (TIO) signal as its source clock. When the PCE bit is set, the prescaler output is used as the timer source clock for the counter regardless of the timer operating mode. To insure proper operation, the PCE bit should be changed only when the timer is disabled (when the TE bit is cleared). Which source clock is used for the prescaler is determined by the PS[1:0] bits of the TPLR. A timer can be clocked by a prescaler clock that is derived from the TIO of another timer.

9.3.4.11 Timer Overflow Flag (TOF) Bit 20

The TOF bit is set to indicate that counter overflow has occurred. This bit is cleared by writing a 1 to the TOF bit. Writing a 0 to the TOF bit has no effect. The bit is also cleared when the timer overflow interrupt is serviced.

The TOF bit is cleared by a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, the STOP instruction, or by clearing the TE bit to disable the timer.

9.3.4.12 Timer Compare Flag (TCF) Bit 21

The TCF bit is set to indicate that the event count is complete. In the timer, PWM, and watchdog modes, the TCF bit is set when (N – M + 1) events have been counted. (N is the value in the compare register and M is the TLR value.) In measurement modes, the TCF bit is set when the measurement is completed.

Writing a 1 into the TCF bit clears this bit. Writing a 0 into the TCF bit has no effect. The bit is also cleared when the timer compare interrupt is serviced.

The TCF bit is cleared by a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, the STOP instruction, or by clearing the TE bit to disable the timer.

Note: The TOF and TCF bits are cleared by writing a 1 to the specific bit. In order to insure that only the desired bit is cleared, do not use the BSET command. The proper way to clear these bits is to write (using a MOVEP instruction) a 1 to the flag to be cleared and a 0 to the other flag.

9.3.4.13 TCSR Reserved Bits 3, 10, 14, 16-19, 22, 23

These reserved bits are read as 0 and should be written with 0 for future compatibility.

9.3.5 Timer Load Register (TLR)

The TLR is a 24-bit, write-only register. In all modes, the counter is preloaded with the TLR value after the TE bit in the TCSR is set and a first event occurs.

- In timer modes, if the timer reload mode (TRM) bit in the TCSR is set, the counter is reloaded each time after it has reached the value contained by the timer compare register (TCPR) and the new event occurs.
- In measurement modes, if the TRM bit in the TCSR is set and the TE bit in the TCSR is set, the counter is reloaded with the value in the TLR on each appropriate edge of the input signal.
- In PWM modes, if the TRM bit in the TCSR is set, the counter is reloaded each time after it has overflowed and the new event occurs.
- In watchdog modes, if the TRM bit in the TCSR is set, the counter is reloaded each time after it has reached the value contained by the TCPR and the new event occurs. In this mode, the counter is also reloaded whenever the TLR is written with a new value while the TE bit in the TCSR is set.
- In all modes, if the TRM bit in the TCSR is cleared (TRM = 0), the counter operates as a free-running counter.

9.3.6 Timer Compare Register (TCPR)

The TCPR is a 24-bit, read/write register that contains the value to be compared to the counter value. These two values are compared every timer clock after the TE bit in the TCSR is set. When the values match, the timer compare flag (TCF) bit is set and an interrupt is generated if interrupts are enabled (if the timer compare interrupt enable (TCIE) bit in the TCSR is set). The TCPR is ignored in measurement modes.

9.3.7 Timer Count Register (TCR)

The Timer Count Register (TCR) is a 24-bit, read-only register. In timer and watchdog modes, the counter's contents can be read at any time by reading the TCR register. In Measurement modes, the TCR is loaded with the current value of the counter on the appropriate edge of the input signal, and its value can be read to determine the width, period, or delay of the leading edge of the input signal. When the timer is in measurement modes, the TIO signal is used for the input signal.

9.4 TIMER OPERATIONAL MODES

Each timer has these operational modes to meet a variety of system requirements:

- Timer
  - GPIO, mode 0: Internal timer interrupt generated by the internal clock
  - Pulse, mode 1: External timer pulse generated by the internal clock
  - Toggle, mode 2: Output timing signal toggled by the internal clock
  - Event counter, mode 3: Internal timer interrupt generated by an external clock
- Measurement
  - Input width, mode 4: Input pulse width measurement
  - Input pulse, mode 5: Input signal period measurement
  - Capture, mode 6: Capture external signal
- PWM, mode 7: Pulse width modulation
- Watchdog
  - Pulse, mode 9: Output pulse, internal clock

– Toggle, mode 10: Output toggle, internal clock

These modes are described in detail below. Timer modes are selected by setting the TC[3:0] bits in the TCSR. **Table 9-2** on page 9-10 shows how the different timer modes are selected by setting the bits in the TCSR. The table also shows the TIO signal direction and the clock source for each timer mode. That table summarizes these modes, and the following paragraphs describe these modes in detail.

Note: To insure proper operation, the TC[3:0] bits should be changed only when the timer is disabled (i.e., when the TE bit in the TCSR is cleared).

9.4.1 Timing Modes

The following timing modes are provided:

- Timer GPIO
- Timer pulse
- Timer toggle
- Event counter

9.4.1.1 Timer GPIO (Mode 0)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	TIO	Clock	#	Function	Name
0	0	0	0	GPIO	Internal	0	Timer	GPIO

In this mode, the timer generates an internal interrupt when a counter value is reached (if the timer compare interrupt is enabled).

Set the TE bit to clear the counter and enable the timer. Load the value the timer is to count into the TCPR. The counter is loaded with the TLR value when the first timer clock signal is received. The timer clock can be taken from either the DSP56309 clock divided by two (CLK/2) or from the prescaler clock output. Each subsequent clock signal increments the counter.

When the counter equals the TCPR value, the TCF bit in TCSR is set, and a compare interrupt is generated if the TCIE bit is set. If the TRM bit in the TCSR is set, the counter

is reloaded with the TLR value at the next timer clock and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each timer clock signal.

This process is repeated until the timer is disabled (i.e., TE is cleared).

If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated.

The counter contents can be read at any time by reading the TCR.

9.4.1.2 Timer Pulse (Mode 1)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	TIO	Clock	#	Function	Name
0	0	0	1	Output	Internal	1	Timer	Pulse

In this mode, the timer generates an external pulse on its TIO signal when the timer count reaches a pre-set value.

Set the TE bit to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TCPR. The counter is loaded with the TLR value when the first timer clock signal is received. The TIO signal is loaded with the value of the INV bit. The timer clock signal can be taken from either the DSP56309 clock divided by two (CLK/2) or from the prescaler clock output. Each subsequent clock signal increments the counter.

When the counter matches the TCPR value, the TCF bit in TCSR is set and a compare interrupt is generated if the TCIE bit is set. The polarity of the TIO signal is inverted for one timer clock period.

If the TRM bit is set, the counter is loaded with the TLR value on the next timer clock and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each timer clock.

This process is repeated until the TE bit is cleared (disabling the timer). The counter contents can be read at any time by reading TCR.

The value of the TLR sets the delay between starting the timer and the generation of the output pulse. To generate successive output pulses with a delay of X clocks between signals, the TLR value should be set to X/2 and the TRM bit should be set.

This process is repeated until the timer is disabled (i.e., TE is cleared).

If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated.

The counter contents can be read at any time by reading the TCR.

9.4.1.3 Timer Toggle (Mode 2)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	TIO	Clock	#	Function	Name
0	0	1	0	Output	Internal	0	Timer	Toggle

In this mode, the timer periodically toggles the polarity of the TIO signal.

Set the TE bit in the TCR to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TPCR. The counter is loaded with the TLR value when the first timer clock signal is received. The TIO signal is loaded with the value of the INV bit. The timer clock signal can be taken from either the DSP56309 clock divided by two (CLK/2) or from the prescaler clock output. Each subsequent clock signal increments the counter.

When the counter value matches the value in the TCPR, the polarity of the TIO output signal is inverted. The TCF bit in the TCSR is set and a compare interrupt is generated if the TCIE bit is set.

If the TRM bit is set, the counter is loaded with the value of the TLR when the next timer clock is received, and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each timer clock.

This process is repeated until the TE bit is cleared, disabling the timer. The counter contents can be read at any time by reading the TCR.

The TLR value in the TCPR sets the delay between starting the timer and toggling the TIO signal. To generate output signals with a delay of X clock cycles between toggles, the TLR value should be set to X/2, and the TRM bit should be set.

This process is repeated until the timer is disabled (i.e., TE is cleared). If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated.

The counter contents can be read at any time by reading the TCR.

9.4.1.4 Timer Event Counter (Mode 3)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	TIO	Clock	#	Function	Name
0	0	1	1	Input	External	3	Timer	Event Counter

In this mode, the timer counts external events and issues an interrupt when a preset number of events is counted.

Set the TE bit to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TPCR. The counter is loaded with the TLR value when the first timer clock signal is received. The timer clock signal can be taken from either the TIO input signal or the prescaler clock output. Each subsequent clock signal increments the counter. If an external clock is used, it must be internally synchronized to the internal clock, and its frequency must be less than the DSP56309 internal operating frequency divided by 4.

The value of the INV bit in the TCSR determines whether low-to-high (0 to 1) transitions or high-to-low (1 to 0) transitions increment the counter. If the INV bit is set, high-to-low transitions increment the counter. If the INV bit is cleared, low-to-high transitions increment the counter.

When the counter matches the value contained in the TCPR, the TCF bit in the TCSR is set, and a compare interrupt is generated if the TCIE bit is set. If the TRM bit is set, the counter is loaded with the value of the TLR when the next timer clock is received, and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each timer clock.

This process is repeated until the timer is disabled (i.e., TE is cleared). If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated.

The counter contents can be read at any time by reading the TCR.

9.4.2 Signal Measurement Modes

The following Signal Measurement modes are provided:

- Measurement input width

- Measurement input period
- Measurement capture

9.4.2.1 Measurement Accuracy

The external signal is synchronized with the internal clock used to increment the counter. This synchronization process can cause the number of clocks measured for the selected signal value to vary from the actual signal value by plus or minus one counter clock cycle.

9.4.2.2 Measurement Input Width (Mode 4)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	0	0	4	Input Width	Measurement	Input	Internal

In this mode, the timer counts the number of clocks that occur between opposite edges of an input signal.

Set the TE bit to clear the counter and enable the timer. Load the timer's count value into the TLR. After the first appropriate transition (as determined by the INV bit) occurs on the TIO input signal, the counter is loaded with the TLR value on the first timer clock signal received either from the DSP56309 clock divided by two (CLK/2) or from the prescaler clock input. Each subsequent clock signal increments the counter.

If the INV bit is set, the timer starts on the first high-to-low (1 to 0) signal transition on the TIO signal. If the INV bit is cleared, the timer starts on the first low-to-high (0 to 1) transition on the TIO signal.

When the first transition opposite in polarity to the INV bit setting occurs on the TIO signal, the counter stops. The TCF bit in the TCSR is set and a compare interrupt is generated if the TCIE bit is set. The value of the counter (which measures the width of the TIO pulse) is loaded into the TCR. The TCR can be read to determine the external signal pulse width.

If the TRM bit is set, the counter is loaded with the TLR value on the first timer clock received following the next valid transition occurring on the TIO input signal and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each timer clock.

This process is repeated until the timer is disabled (i.e., TE is cleared). If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated. The counter contents can be read at any time by reading the TCR.

9.4.2.3 Measurement Input Period (Mode 5)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	0	1	5	Input Period	Measurement	Input	Internal

In this mode, the timer counts the period between the reception of signal edges of the same polarity across the TIO signal.

Set the TE bit to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TLR. The value of the INV bit determines whether the period is measured between consecutive low-to-high (0 to 1) transitions of TIO or between consecutive high-to-low (1 to 0) transitions of TIO. If INV is set, high-to-low signal transitions are selected. If INV is cleared, low-to-high signal transitions are selected.

After the first appropriate transition occurs on the TIO input signal, the counter is loaded with the TLR value on the first timer clock signal received from either the DSP56309 clock divided by two (CLK/2) or the prescaler clock output. Each subsequent clock signal increments the counter.

On the next signal transition of the same polarity that occurs on TIO, the TCF bit in the TCSR is set and a compare interrupt is generated if the TCIE bit is set. The contents of the counter are loaded into the TCR. The TCR then contains the value of the time that elapsed between the two signal transitions on the TIO signal.

After the second signal transition, if the TRM bit is set, the TE bit is set to clear the counter and enable the timer. The counter is loaded with the TLR value on the first timer clock signal. Each subsequent clock signal increments the counter.

After the second signal transition, if the TRM bit is set, the TE bit is set to clear if the TRM bit is cleared, the counter continues to be incremented on each timer clock. This process is repeated until the timer is disabled (i.e., TE is cleared). If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated. The counter contents can be read at any time by reading the TCR.

9.4.2.4 Measurement Capture (Mode 6)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	1	0	6	Capture	Measurement	Input	Internal

In this mode, the timer counts the number of clocks that elapse between starting the timer and receiving an external signal.

Set the TE bit to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TLR. When the first timer clock signal is received, the counter is loaded with the TLR value. The timer clock signal can be taken from either the DSP56309 clock divided by two (CLK/2) or from the prescaler clock output. Each subsequent clock signal increments the counter.

At the first appropriate transition of the external clock detected on the TIO signal, the TCF bit in the TCSR is set and, if the TCIE bit is set, a compare interrupt is generated, the counter halts, and the contents of the counter are loaded into the TCR. The value of the TCR represents the delay between the setting of the TE bit and the detection of the first clock edge signal on the TIO signal.

The value of the INV bit determines whether a high-to-low (1 to 0) or low-to-high (0 to 1) transition of the external clock signals the end of the timing period. If the INV bit is set, a high-to-low transition signals the end of the timing period. If INV is cleared, a low-to-high transition signals the end of the timing period.

If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated. The counter contents can be read at any time by reading the TCR.

9.4.3 Pulse Width Modulation (PWM, Mode 7)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	1	1	7	Pulse Width Modulation	PWM	Output	Internal

In this mode, the timer generates periodic pulses of a preset width.

Set the TE bit to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TPCR. When first timer clock is received from either the DSP56309 internal clock divided by two (CLK/2) or the prescaler clock output, the counter is loaded with the TLR value. Each subsequent timer clock increments the counter.

When the counter equals the value in the TCPR, the TIO output signal is toggled and the TCF bit in the TCSR is set. The contents of the counter are placed into the TCR. If the TCIE bit is set, a compare interrupt is generated. The counter continues to be incremented on each timer clock.

If counter overflow has occurred, the TIO output signal is toggled, the TOF bit in TCSR is set, and an overflow interrupt is generated if the TOIE bit is set. If the TRM bit is set, the counter is loaded with the TLR value on the next timer clock and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each timer clock.

This process is repeated until the timer is disabled by clearing the TE bit.

TIO signal polarity is determined by the value of the INV bit. When the counter is started by setting the TE bit, the TIO signal assumes the value of the INV bit. On each subsequent toggling of the TIO signal, the polarity of the TIO signal is reversed. For example, if the INV bit is set, the TIO signal generates the following signal: 1010. If the INV bit is cleared, the TIO signal generates the following signal: 0101.

The counter contents can be read at any time by reading the TCR.

The value of the TLR determines the output period ($FFFFFF – TLR + 1). The timer counter increments the initial TLR value and toggles the TIO signal when the counter value exceeds $FFFFFF.

The duty cycle of the TIO signal is determined by the value in the TCPR. When the value in the TLR is incremented to a value equal to the value in the TCPR, the TIO signal is toggled. The duty cycle is equal to ($FFFFFF – TCPR) divided by ($FFFFFF – TLR + 1). For a 50% duty cycle, the value of TCPR is equal to ($FFFFFF + TLR + 1) / 2.

Note: The value in TCPR must be greater than the value in TLR.

9.4.4 Watchdog Modes

The following watchdog timer modes are provided:

- Watchdog pulse
- Watchdog toggle

9.4.4.1 Watchdog Pulse (Mode 9)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
1	0	0	1	9	Pulse	Watchdog	Output	Internal

In this mode, the timer generates an external signal at a preset rate. The signal period is equal to the period of one timer clock.

Set the TE bit to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TCPR. The counter is loaded with the TLR value on the first timer clock received from either the DSP56309 internal clock divided by two (CLK/2) or the prescaler clock output. Each subsequent timer clock increments the counter.

When the counter matches the value of the TCPR, the TCF bit in the TCSR is set and a compare interrupt is generated if the TCIE bit is also set.

If the TRM bit is set, the counter is loaded with the TLR value on the next timer clock and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each subsequent timer clock.

This process is repeated until the timer is disabled (i.e., TE is cleared).

If the counter overflows, the TOF bit is set, and if TOIE is set, an overflow interrupt is generated. At the same time, a pulse is output on the TIO signal with a pulse width equal to the timer clock period. The pulse polarity is determined by the value of the INV bit. If the INV bit is set, the pulse polarity is high (logical 1). If the INV bit is cleared, the pulse polarity is low (logical 0).

The counter contents can be read at any time by reading the TCR. The counter is reloaded whenever the TLR is written with a new value while the TE bit is set.

Note: In this mode, internal logic preserves the TIO value and direction for an additional 2.5 internal clock cycles after a DSP56309 hardware $\overline{\text{RESET}}$ signal is asserted. This insures that a valid $\overline{\text{RESET}}$ signal is generated when the TIO signal is used to reset the DSP56309.

9.4.4.2 Watchdog Toggle (Mode 10)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	NAME	Function	TIO	Clock
1	0	1	0	10	Toggle	Watchdog	Output	Internal

In this mode, the timer toggles an external signal after preset period.

Set the TE bit to clear the counter and enable the timer. The value to which the timer is to count is loaded into the TPCR. The counter is loaded with the TLR value on the first timer clock received from either the DSP56309 internal clock divided by two (CLK/2) or the prescaler clock output. Each subsequent timer clock increments the counter. The TIO signal is set to the value of the INV bit.

When the counter equals the value in the TCPR, the TCF bit in the TCSR is set, and a compare interrupt is generated if the TCIE bit is also set. If the TRM bit is set, the counter is loaded with the TLR value on the next timer clock and the count is resumed. If the TRM bit is cleared, the counter continues to be incremented on each subsequent timer clock.

When counter overflow has occurred, the polarity of the TIO output signal is inverted, the TOF bit in the TCSR is set, and an overflow interrupt is generated if the TOIE bit is also set. The TIO polarity is determined by the INV bit.

The counter is reloaded whenever the TLR is written with a new value while the TE bit is set. This process is repeated until the timer is disabled by clearing the TE bit. The counter contents can be read at any time by reading the TCR register.

Note: In this mode, internal logic preserves the TIO value and direction for an additional 2.5 internal clock cycles after a DSP56309 hardware $\overline{\text{RESET}}$ signal is asserted. This insures that a valid reset signal is generated when the TIO signal is used to reset the DSP56309.

9.4.5 Reserved Modes

Modes 8, 11, 12, 13, 14, and 15 are reserved.

9.4.6 Special Cases

The following special cases apply during wait and stop state.

9.4.6.1 Timer Behavior during Wait

Timer clocks are active during the execution of the WAIT instruction and timer activity is undisturbed. If a timer interrupt is generated, the DSP56309 leaves the wait state and services the interrupt.

9.4.6.2 Timer Behavior during Stop

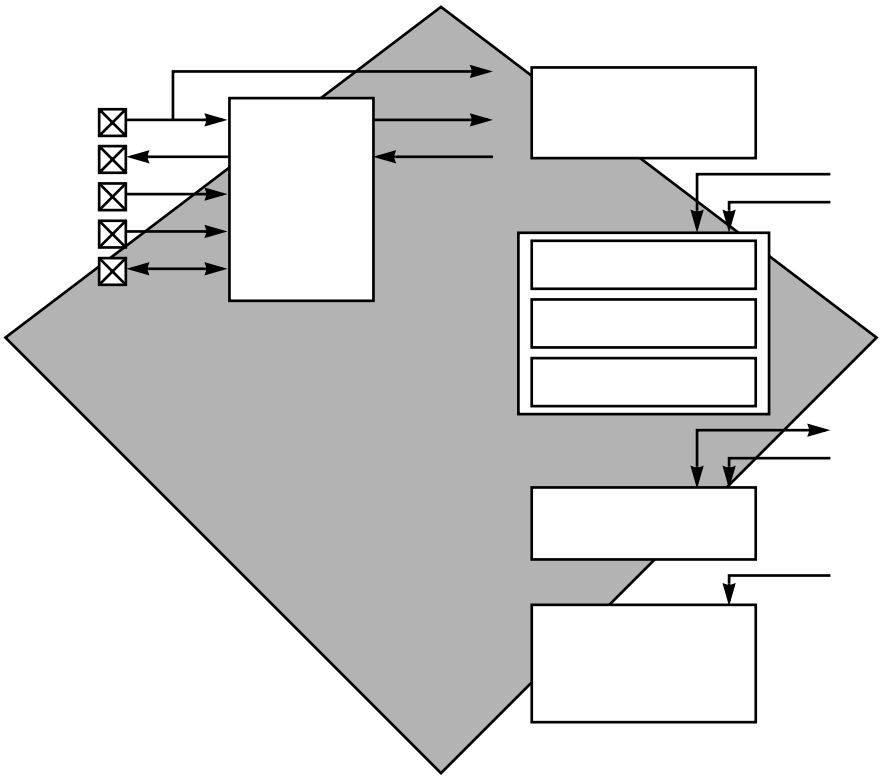
During the execution of the STOP instruction, the timer clocks are disabled, timer activity is stopped, and the TIO signals are disconnected. Any external changes that happen to the TIO signals are ignored when the DSP56309 is in the stop state. To insure correct operation, the timers should be disabled before the DSP56309 is placed into the stop state.

9.4.7 DMA Trigger

Each timer can also be used to trigger DMA transfers. For this to occur, a DMA channel must be programmed to be triggered by a timer event. The timer issues a DMA trigger on every event in all modes of operation. The DMA channel does not have the capability to save multiple DMA triggers generated by the timer. To insure that all DMA triggers are serviced, the user must provide for the preceding DMA trigger to be serviced before the next trigger is received by the DMA channel.

SECTION 10

ON-CHIP EMULATION MODULE



10.1 INTRODUCTION

The DSP56300 core On-Chip Emulation (OnCE™) module provides a means of interacting with the DSP56300 core and its peripherals nonintrusively so that a user can examine registers, memory, or on-chip peripherals, thus facilitating hardware and software development on the DSP56300 core processor. To achieve this, special circuits and dedicated signals on the DSP56300 core are defined to avoid sacrificing any user-accessible on-chip resource. The OnCE module resources can be accessed only after executing the JTAG instruction ENABLE_ONCE. These resources are accessible even when the chip is operating in normal mode. See **Section 11—JTAG Port** for a description of the JTAG functionality and its relation to the OnCE module. **Figure 10-1** shows the block diagram of the OnCE module.

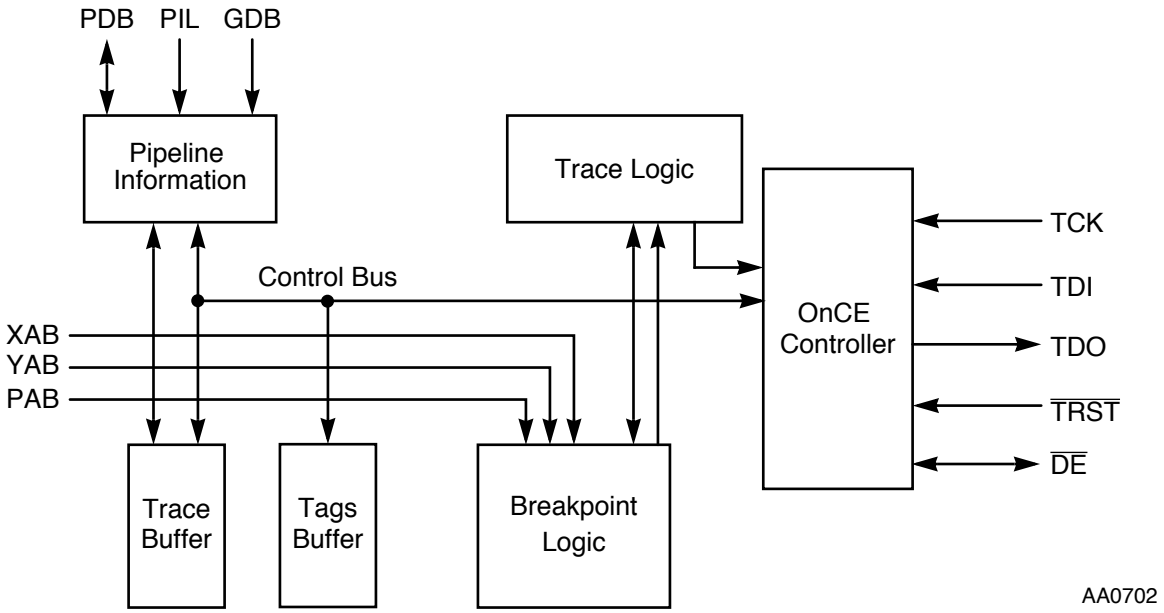


Figure 10-1 OnCE Module Block Diagram

10.2 OnCE MODULE SIGNALS

The OnCE module controller functionality is accessed through the JTAG port. There are no dedicated OnCE module signals for the clock, data in, or data out. The JTAG signals TCK, TDI, and TDO are used to shift in and out data and instructions. See **Section 11.2—JTAG Signals** on page 11-4 for the description of the JTAG signals. To facilitate emulation-specific functions, one additional signal, called $\overline{DE}$, is provided on the DSP56309.

10.3 DEBUG EVENT ($\overline{DE}$)

The bidirectional open drain debug event signal ($\overline{DE}$) provides a fast means of entering debug mode from an external command controller (when input), and a fast means of acknowledging the entering of debug mode to an external command controller (when output). The assertion of this signal by a command controller causes the DSP56300 core to finish the current instruction being executed, save the instruction pipeline information, enter debug mode, and wait for commands to be entered from the TDI line. If the $\overline{DE}$ signal is used to enter debug mode, then it must be deasserted after the OnCE port responds with an acknowledge and before sending the first OnCE command. The assertion of this signal by the DSP56300 core indicates that the DSP has entered debug mode and is waiting for commands to be entered from the TDI line. The $\overline{DE}$ signal also facilitates multiple processor connections, as shown in **Figure 10-2**.

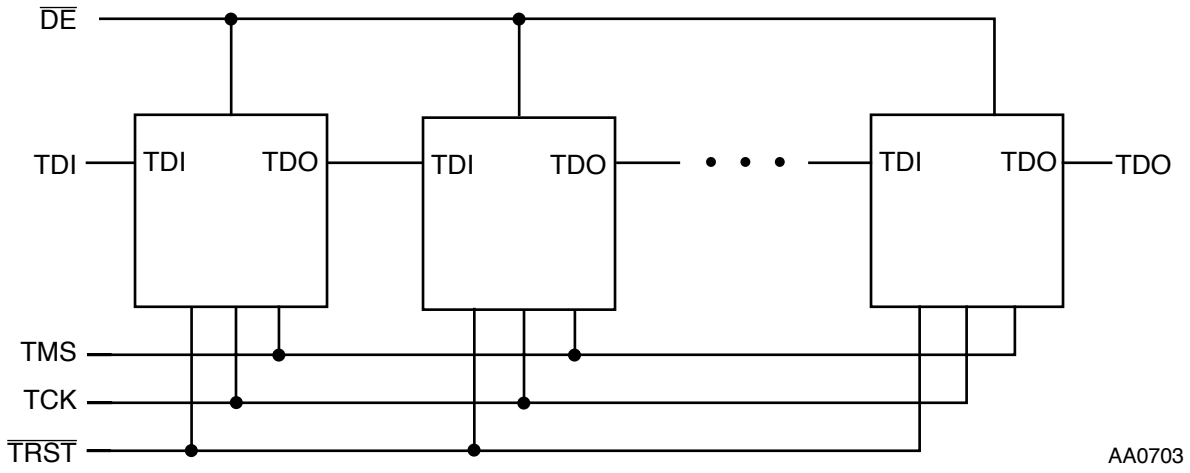


Figure 10-2 OnCE Module Multiprocessor Configuration

In this way, the user can stop all the devices in the system when one of the devices enters debug mode. The user can also stop all the devices synchronously by asserting the $\overline{DE}$ line.

10.4 OnCE CONTROLLER

The OnCE controller contains the following blocks: OnCE Command Register (OCR), OnCE Decoder, and the status/control register. **Figure 10-3** illustrates a block diagram of the OnCE controller.

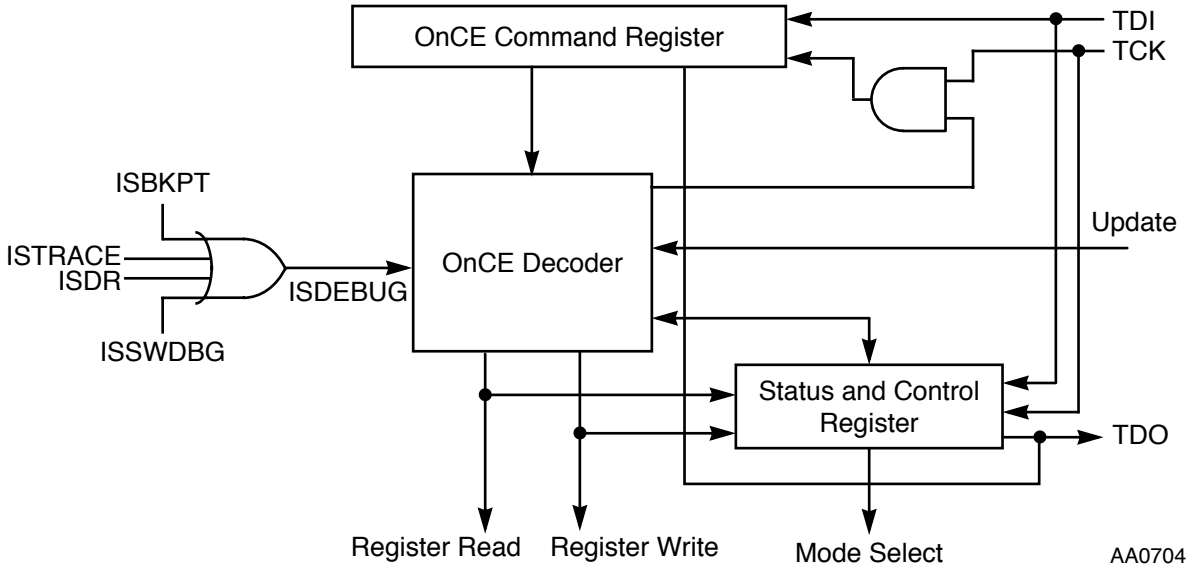


Figure 10-3 OnCE Controller Block Diagram

10.4.1 OnCE Command Register (OCR)

The OCR is an 8-bit shift register that receives its serial data from the TDI signal. It holds the 8-bit commands to be used as input for the OnCE decoder. The OCR is shown in **Figure 10-4**.

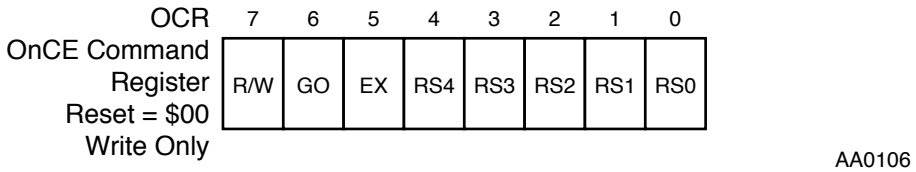


Figure 10-4 OnCE Command Register

10.4.1.1 Register Select (RS4–RS0) Bits 0–4

The register select bits define which register is source/destination for the read/write operation. See **Table 10-4** for the OnCE register select encoding.

10.4.1.2 Exit Command (EX) Bit 5

If the EX bit is set, leave debug mode and resume normal operation. The EXIT command is executed only if the GO command is issued, and the operation is write to OPDBR or

read/write to "No Register Selected". Otherwise the EX bit is ignored. **Table 10-1** shows the definition of the EX bit.

Table 10-1 EX Bit Definition

EX	Action
0	Remain in Debug mode
1	Leave Debug mode

10.4.1.3 GO Command (GO) Bit 6

If the GO bit is set, execute the instruction that resides in the PIL register. To execute the instruction, the core leaves debug mode. The core returns to debug mode immediately after executing the instruction if the EX bit is cleared. The core goes on to normal operation if the EX bit is set. The GO command is executed only if the operation is write to OPDBR or read/write to "No Register Selected". Otherwise the GO bit is ignored. **Table 10-2** shows the definition of the GO bit.

Table 10-2 GO Bit Definition

GO	Action
0	Inactive—no action taken
1	Execute instruction in PIL

10.4.1.4 Read/Write Command (R/$\overline{W}$) Bit 7

The R/$\overline{W}$ bit, as shown in **Table 10-3**, specifies the direction of data transfer. **Table 10-4** shows how to encode OnCE register selections.

Table 10-3 R/$\overline{W}$ Bit Definition

R/$\overline{W}$	Action
0	Write the data associated with the command into the register specified by RS4–RS0.
1	Read the data contained in the register specified by RS4–RS0.

Table 10-4 OnCE Register Select Encoding

RS[4:0]	Register Selected
00000	OnCE Status and Control Register (OSCR)

Table 10-4 OnCE Register Select Encoding (Continued)

RS[4:0]	Register Selected
00001	Memory Breakpoint Counter (OMBC)
00010	Breakpoint Control Register (OBCR)
00011	Reserved Address
00100	Reserved Address
00101	Memory Limit Register 0 (OMLR0)
00110	Memory Limit Register 1 (OMLR1)
00111	Reserved Address
01000	Reserved Address
01001	GDB Register (OGDBR)
01010	PDB Register (OPDBR)
01011	PIL Register (OPILR)
01100	PDB GO-TO Register (for GO TO command)
01101	Trace Counter (OTC)
01110	Reserved Address
01111	PAB Register for Fetch (OPABFR)
10000	PAB Register for Decode (OPABDR)
10001	PAB Register for Execute (OPABEX)
10010	Trace Buffer and Increment Pointer
10011	Reserved Address
101xx	Reserved Address
11xx0	Reserved Address
11x0x	Reserved Address
110xx	Reserved Address
11111	No Register Selected

10.4.2 OnCE Decoder (ODEC)

The ODEC supervises the entire OnCE module activity. It receives as input the 8-bit command from the OCR, a signal from JTAG Controller (indicating that 8 or 24 bits have been received and update of the selected data register must be performed), and a signal indicating that the core was halted. The ODEC generates all the strobes required for reading and writing the selected OnCE registers.

10.4.3 OnCE Status and Control Register (OSCR)

The OSCR is a 24-bit register used to enable trace mode and to indicate the cause of entering debug mode. The control bits are read/write while the status bits are read-only. The OSCR bits are cleared by a hardware $\overline{\text{RESET}}$ signal. The OSCR is shown in **Figure 10-5**.

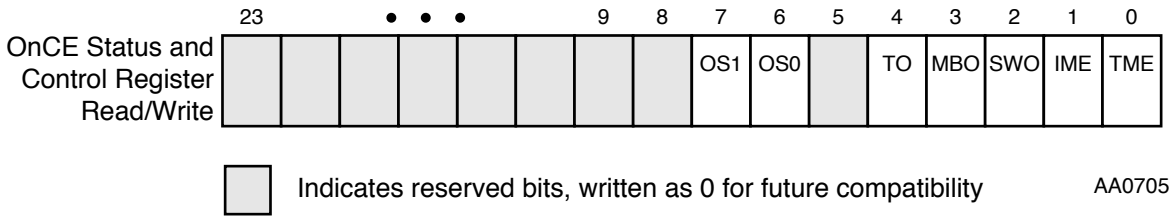


Figure 10-5 OnCE Status and Control Register (OSCR)

10.4.3.1 Trace Mode Enable (TME) Bit 0

The TME control bit, when set, enables trace mode.

10.4.3.2 Interrupt Mode Enable (IME) Bit 1

The IME control bit, when set, causes the chip to execute a vectored interrupt to the address VBA:$06 instead of entering debug mode.

10.4.3.3 Software Debug Occurrence (SWO) Bit 2

The SWO bit is a read-only status bit that is set when debug mode is entered because of the execution of the DEBUG or DEBUGcc instruction with condition true. This bit is cleared when leaving debug mode.

10.4.3.4 Memory Breakpoint Occurrence (MBO) Bit 3

The MBO bit is a read-only status bit that is set when debug mode is entered because a memory breakpoint has been encountered. This bit is cleared when leaving debug mode.

10.4.3.5 Trace Occurrence (TO) Bit 4

The TO bit is a read-only status bit that is set when debug mode is entered when the trace counter is zero while trace mode is enabled. This bit is cleared when leaving debug mode.

10.4.3.6 Reserved OCSR Bit 5

Bit 5 is reserved for future use. It is read as 0 and should be written with 0 for future compatibility.

10.4.3.7 Core Status (OS0, OS1) Bits 6-7

The OS0, OS1 bits are read-only status bits that provide core status information. By examining the status bits, the user can determine whether the chip has entered debug mode. Examining SWO, MBO, and TO identifies the cause of entering debug mode. The user can also examine these bits and determine the cause why the chip has not entered debug mode after debug event ($\overline{DE}$) assertion or as a result of the execution of the JTAG debug request instruction (core waiting for the bus, STOP or WAIT instruction, etc.). These bits are also reflected in the JTAG instruction shift register, which allows the polling of the core status information at the JTAG level. This is useful when the DSP56300 core executes the STOP instruction (and therefore there are no clocks) to allow the reading of OSCR. See **Table 10-5** for the definition of the OS0–OS1 bits.

Table 10-5 Core Status Bits Description

OS1	OS0	Description
0	0	DSP56300 core is executing instructions
0	1	DSP56300 core is in wait or stop
1	0	DSP56300 core is waiting for bus
1	1	DSP56300 core is in debug mode

10.4.3.8 Reserved Bits 8-23

Bits 8–23 are reserved for future use. They are read as 0 and should be written with 0 for future compatibility.

10.5 OnCE MEMORY BREAKPOINT LOGIC

Memory breakpoints can be set on program memory or data memory locations. In addition, the breakpoint does not have to be in a specific memory address, but within an approximate address range where the program may be executing. This significantly increases the programmer's ability to monitor what the program is doing in real time.

The breakpoint logic, described in **Figure 10-6**, contains a latch for the addresses, which are registers that store the upper and lower address limit, address comparators, and a breakpoint counter.

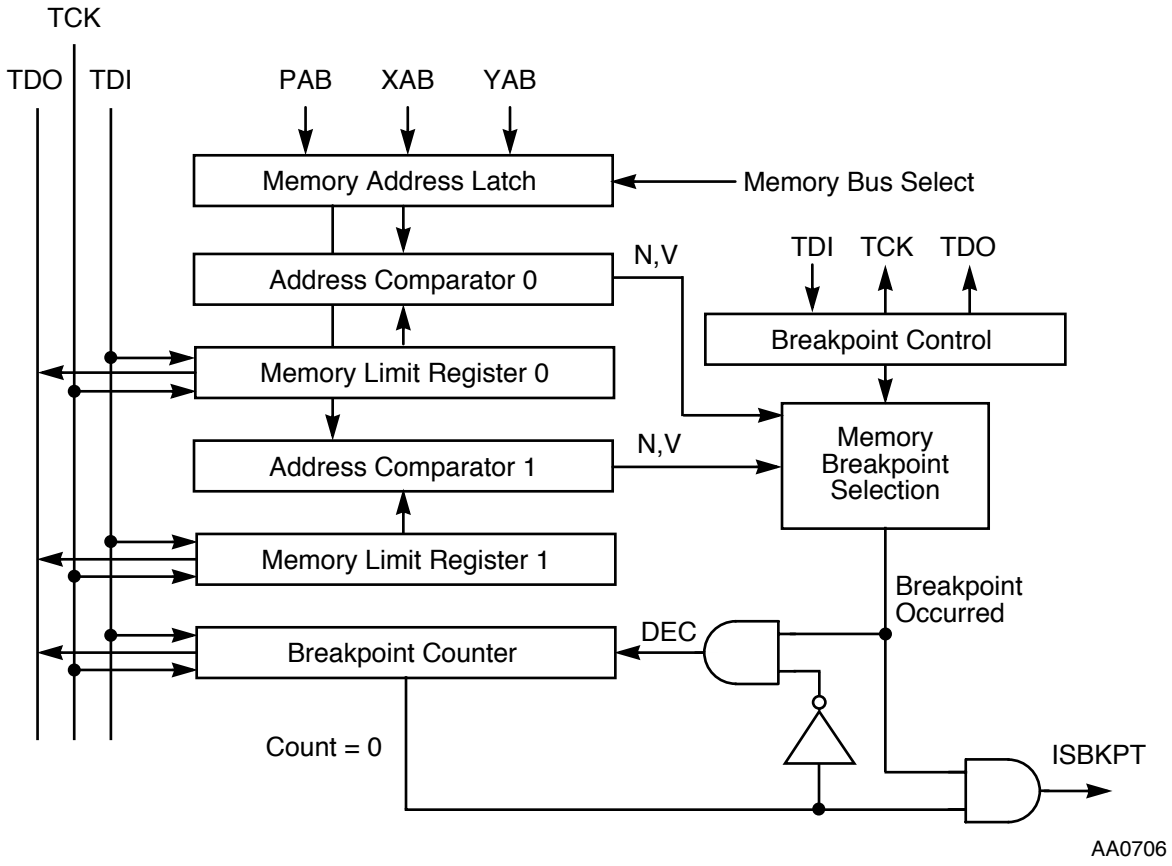


Figure 10-6 OnCE Memory Breakpoint Logic 0

Address comparators are useful in determining where a program may be getting lost or when data is being written where it should not be written. They are also useful in halting a program at a specific point to examine/change registers or memory. Using address comparators to set breakpoints enables the user to set breakpoints in RAM or ROM and while in any operating mode. Memory accesses are monitored according to the contents of the OBCR as specified in **Section 10.5.6—OnCE Breakpoint Control Register (OBCR)**.

10.5.1 OnCE Memory Address Latch (OMAL)

The OMAL is a 16-bit register that latches the PAB, XAB or YAB on every instruction cycle according to the MBS1–MBS0 bits in OBCR.

10.5.2 OnCE Memory Limit Register 0 (OMLR0)

The OMLR0 is a 16-bit register that stores the memory breakpoint limit. OMLR0 can be read or written through the JTAG port. Before enabling breakpoints, OMLR0 must be loaded by the external command controller.

10.5.3 OnCE Memory Address Comparator 0 (OMAC0)

The OMAC0 compares the current memory address (stored in OMAL0) with the OMLR0 contents.

10.5.4 OnCE Memory Limit Register 1 (OMLR1)

The OMLR1 is a 16-bit register that stores the memory breakpoint limit. OMLR1 can be read or written through the JTAG port. Before enabling breakpoints, OMLR1 must be loaded by the external command controller.

10.5.5 OnCE Memory Address Comparator 1 (OMAC1)

The OMAC1 compares the current memory address (stored in OMAL0) with the OMLR1 contents.

10.5.6 OnCE Breakpoint Control Register (OBCR)

The OBCR is a 16-bit register used to define the memory breakpoint events. OBCR can be read or written through the JTAG port. All the bits of the OBCR are cleared by a hardware $\overline{\text{RESET}}$ signal. The OBCR appears in **Figure 10-7**.

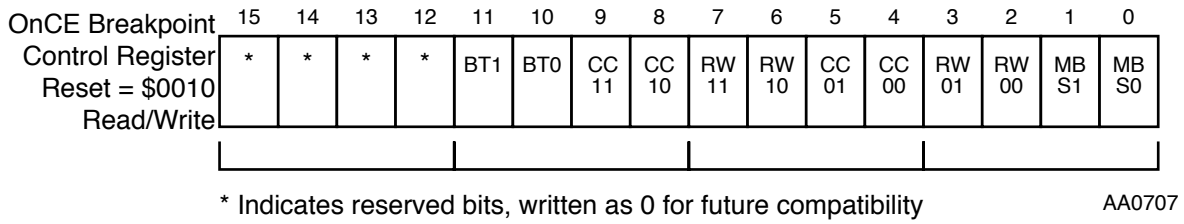


Figure 10-7 OnCE Breakpoint Control Register (OBCR)

10.5.6.1 Memory Breakpoint Select (MBS0–MBS1) Bits 0–1

The MBS0–MBS1 bits enable memory breakpoints 0 and 1, allowing them to occur when a memory access is performed on P, X, or Y space. See **Table 10-6** for the definition of the MBS0–MBS1 bits.

Table 10-6 Memory Breakpoint 0 and 1 Select Table

MBS1	MBS0	Description
0	0	Reserved
0	1	Breakpoint on P access
1	0	Breakpoint on X access
1	1	Breakpoint on Y access

10.5.6.2 Breakpoint 0 Read/Write Select (RW00–RW01) Bits 2–3

The RW00–RW01 bits define the memory breakpoint 0 to occur when a memory address access is performed for read, write, or both. See **Table 10-7** for the definition of the RW00–RW01 bits.

Table 10-7 Breakpoint 0 Read/Write Select Table

RW01	RW00	Description
0	0	Breakpoint disabled
0	1	Breakpoint on write access
1	0	Breakpoint on read access
1	1	Breakpoint on read or write access

10.5.6.3 Breakpoint 0 Condition Code Select (CC00–CC01) Bits 4–5

The CC00–CC01 bits define the condition of the comparison between the current memory address (OMAL0) and the memory limit register 0 (OMLR0). See **Table 10-8** for the definition of the CC00–CC01 bits.

Table 10-8 Breakpoint 0 Condition Select Table

CC01	CC00	Description
0	0	Breakpoint on not equal
0	1	Breakpoint on equal
1	0	Breakpoint on less than
1	1	Breakpoint on greater than

10.5.6.4 Breakpoint 1 Read/Write Select (RW10–RW11) Bits 6–7

The RW10–RW11 bits control define memory breakpoint 1 to occur when a memory address access is performed for read, write, or both. See **Table 10-9** for the definition of the RW10–RW11 bits.

Table 10-9 Breakpoint 1 Read/Write Select Table

RW11	RW10	Description
0	0	Breakpoint disabled
0	1	Breakpoint on write access
1	0	Breakpoint on read access
1	1	Breakpoint read or write access

10.5.6.5 Breakpoint 1 Condition Code Select (CC10–CC11) Bits 8–9

The CC10–CC11 bits define the condition of the comparison between the current memory address (OMAL0) and the OnCE Memory Limit Register 1 (OMLR1). See **Table 10-10** for the definition of the CC10–CC11 bits.

Table 10-10 Breakpoint 1 Condition Select Table

CC11	CC10	Description
0	0	Breakpoint on not equal
0	1	Breakpoint on equal
1	0	Breakpoint on less than
1	1	Breakpoint on greater than

10.5.6.6 Breakpoint 0 and 1 Event Select (BT0–BT1) Bits 10–11

The BT0–BT1 bits define the sequence between breakpoint 0 and 1. If the condition defined by BT0–BT1 is met, then the OnCE Breakpoint Counter (OMBC) is decremented. See **Table 10-11** for the definition of the BT0–BT1 bits.

Table 10-11 Breakpoint 0 and 1 Event Select Table

BT1	BT0	Description
0	0	Breakpoint 0 and Breakpoint 1
0	1	Breakpoint 0 or Breakpoint 1
1	0	Breakpoint 1 after Breakpoint 0
1	1	Breakpoint 0 after Breakpoint 1

10.5.6.7 OnCE Memory Breakpoint Counter (OMBC)

The OMBC is a 16-bit counter that is loaded with a value equal to the number of times minus one that a memory access event should occur before a memory breakpoint is declared. The memory access event is specified by the OBCR and by the memory limit registers. On each occurrence of the memory access event, the breakpoint counter is decremented. When the counter reaches 0 and a new occurrence takes place, the chip enters debug mode. The OMBC can be read or written through the JTAG port. Every time that the limit register is changed or a different breakpoint event is selected in the OBCR, the breakpoint counter must be written afterwards. This insures that the OnCE

breakpoint logic is reset and that no previous events can affect the new breakpoint event selected. The breakpoint counter is cleared by a hardware $\overline{\text{RESET}}$ signal.

10.5.6.8 Reserved Bits 12-15

Bits 12–15 are reserved for future use. They are read as 0 and should be written with 0 for future compatibility.

10.6 OnCE TRACE LOGIC

Using the OnCE trace logic, execution of instructions in single or multiple steps is possible. The OnCE trace logic causes the chip to enter debug mode after the execution of one or more instructions and wait for OnCE commands from the debug serial port. The OnCE trace logic block diagram is shown in **Figure 10-8**.

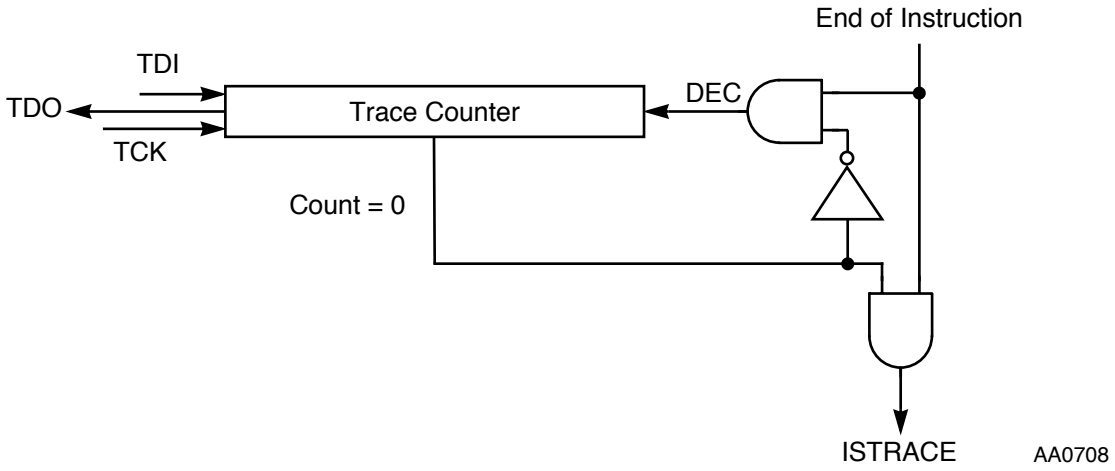


Figure 10-8 OnCE Trace Logic Block Diagram

Trace mode has a counter associated with it so that more than one instruction can be executed before returning back to debug mode. The objective of the counter is to allow the user to take multiple instruction steps real time before entering debug mode. This feature helps the software developer debug sections of code that do not have a normal flow or are getting hung up in infinite loops. The OTC also enables the user to count the number of instructions executed in a code segment.

To enable trace mode, the counter is loaded with a value, the program counter is set to the start location of the instruction(s) to be executed real time, the TME bit is set in the

OSCR, and the DSP56300 core exits debug mode by executing the appropriate command issued by the external command controller.

Upon exiting debug mode, the counter is decremented after each execution of an instruction. Interrupts are serviceable then. Moreover, all executed instructions, including fast interrupt services and the execution of each repeated instruction, cause the OTC to be decremented. Upon decrementing to 0, the DSP56300 core reenters debug mode, the trace occurrence bit (TO) in the OSCR register is set, the core status bits OS[1:0] are set to 11, and the $\overline{DE}$ signal is asserted to indicate that the DSP56300 core has entered debug mode and is requesting service.

The OnCE Trace Counter (OTC) is a 16-bit counter that can be read or written through the JTAG port. If N instructions are to be executed before entering debug mode, the OTC should be loaded with N – 1. The OTC is cleared by a hardware $\overline{RESET}$ signal.

10.7 METHODS OF ENTERING DEBUG MODE

Entering debug mode is acknowledged by the chip by setting the core status bits OS1 and OS0 and asserting the $\overline{DE}$ line. This informs the external command controller that the chip has entered debug mode and is waiting for commands. The DSP56300 core can disable the OnCE module if the ROM Security option is implemented. If the ROM security is implemented, the OnCE module remains inactive until a write operation to the OGDBR is executed by the DSP56300 core.

10.7.1 External Debug Request During RESET Assertion

Holding the $\overline{DE}$ line asserted during the assertion of $\overline{RESET}$ causes the chip to enter debug mode. After receiving the acknowledge, the external command controller must negate the $\overline{DE}$ line before sending the first command.

Note: In this case, the chip does not execute any instruction before entering debug mode.

10.7.2 External Debug Request During Normal Activity

Holding the $\overline{DE}$ line asserted during normal chip activity causes the chip to finish the execution of the current instruction and then enter Debug mode. After receiving the acknowledge, the external command controller must negate the $\overline{DE}$ line before sending

the first command. This process is the same for any newly fetched instruction, including instructions fetched by the interrupt processing or instructions that will be aborted by the interrupt processing.

Note: In this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the instruction latch.

10.7.3 Executing the JTAG DEBUG_REQUEST Instruction

Executing the JTAG instruction DEBUG_REQUEST asserts an internal debug request signal. Consequently, the chip finishes the execution of the current instruction and stops after the newly fetched instruction enters the instruction latch. After entering debug mode, the core status bits OS1 and OS0 are set and the $\overline{DE}$ line is asserted, thus acknowledging the external command controller that debug mode has been entered.

10.7.4 External Debug Request During Stop

Executing the JTAG instruction DEBUG_REQUEST (or asserting $\overline{DE}$) while the chip is in the stop state (i. e., has executed a STOP instruction) causes the chip to exit the stop state and enter debug mode. After receiving the acknowledge, the external command controller must negate $\overline{DE}$ before sending the first command.

Note: In this case, the chip completes the execution of the STOP instruction and halts after the next instruction enters the instruction latch.

10.7.5 External Debug Request During Wait

Executing the JTAG instruction DEBUG_REQUEST (or asserting $\overline{DE}$) while the chip is in the wait state (i. e., has executed a WAIT instruction) causes the chip to exit the wait state and enter debug mode. After receiving the acknowledge, the external command controller must negate $\overline{DE}$ before sending the first command.

Note: In this case, the chip completes the execution of the WAIT instruction and halts after the next instruction enters the instruction latch.

10.7.6 Software Request During Normal Activity

Upon executing the DSP56300 core instruction DEBUG (or DEBUGcc when the specified condition is true), the chip enters debug mode after the instruction following the DEBUG instruction has entered the instruction latch.

10.7.7 Enabling Trace Mode

When Trace mode is enabled and the OTC is greater than zero, the OTC is decremented after each instruction execution. Execution of an instruction when the value in the OTC is 0 causes the chip to enter debug mode after completing the execution of the instruction. Only instructions actually executed cause the OTC to decrement. An aborted instruction does not decrement the OTC and does not cause the chip to enter debug mode.

10.7.8 Enabling Memory Breakpoints

When the memory breakpoint mechanism is enabled with a breakpoint counter value of 0, the chip enters debug mode after completing the execution of the instruction that caused the memory breakpoint to occur. In case of breakpoints on executed program memory fetches, the breakpoint is acknowledged immediately after the execution of the fetched instruction. In case of breakpoints on accesses to X, Y or program memory spaces by MOVE instructions, the breakpoint is acknowledged after the completion of the instruction following the instruction that accessed the specified address.

10.8 PIPELINE INFORMATION AND OGDB REGISTER

To restore the pipeline and to resume normal chip activity upon returning from debug mode, a number of on-chip registers store the chip pipeline status. **Figure 10-9** shows the block diagram of the pipeline information registers, with the exception of the PAB registers, which appear in **Figure 10-10** on page 10-22.

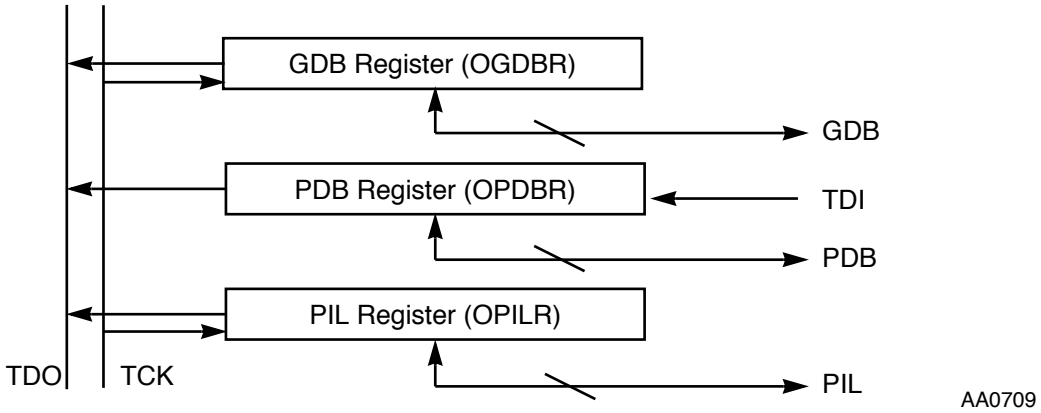


Figure 10-9 OnCE Pipeline Information and GDB Registers

10.8.1 OnCE PDB Register (OPDBR)

The OPDBR is a 24-bit latch that stores the value of the program data bus generated by the last program memory access of the core before debug mode is entered. The OPDBR register can be read or written through the JTAG port. This register is affected by the operations performed during debug mode and must be restored by the external command controller when returning to normal mode.

10.8.2 OnCE PIL Register (OPILR)

The OPILR is a 24-bit latch that stores the value of the instruction latch before debug mode is entered. OPILR can only be read through the JTAG port.

Note: Since the instruction latch is affected by the operations performed during debug mode, it must be restored by the external command controller when returning to normal mode. Since there is no direct write access to the instruction latch, the task of restoring is accomplished by writing to OPDBR with no-GO and no-EX. In this case the data written on PDB is transferred into the instruction latch.

10.8.3 OnCE GDB Register (OGDBR)

The OGDBR is a 16-bit latch that can only be read through the JTAG port. The OGDBR is not actually required for restoring the pipeline status but is required as a means of passing information between the chip and the external command controller. The OGDBR is mapped on the X internal I/O space at address $FFFC. Whenever the external command controller needs the contents of a register or memory location, it forces the chip to execute an instruction that brings that information to the OGDBR. Then the contents of the OGDBR are delivered serially to the external command controller by the command `READ GDB REGISTER`.

10.9 DEBUGGING RESOURCES

To ease debugging activity and keep track of program flow, the DSP56300 core provides a number of on-chip dedicated resources. There are three read-only PAB registers that give pipeline information when debug mode is entered, and a trace buffer that stores the address of the last instruction that was executed, as well as the addresses of the last 12 change-of-flow instructions.

10.9.1 OnCE PAB Register for Fetch (OPABFR)

The OPABFR is a 16-bit register that stores the address of the last instruction whose fetch was started before debug mode was entered. The OPABFR can only be read through the JTAG port. This register is not affected by the operations performed during debug mode.

10.9.2 PAB Register for Decode (OPABDR)

The OPABDR is a 16-bit register that stores the address of the instruction currently on the PDB. This is the instruction whose fetch was completed before the chip has entered debug mode. The OPABDR can only be read through the JTAG port. This register is not affected by the operations performed during debug mode.

10.9.3 OnCE PAB Register for Execute (OPABEX)

The OPABEX is a 16-bit register that stores the address of the instruction currently in the instruction latch. This is the instruction that would have been decoded and executed if

the chip would not have entered debug mode. The OPABEX register can only be read through the JTAG port. This register is not affected by the operations performed during debug mode.

10.9.4 Trace Buffer

The trace buffer stores the addresses of the last 12 change-of-flow instructions that were executed, as well as the address of the last executed instruction. The trace buffer is implemented as a circular buffer containing 12 17-bit registers and one 4-bit counter. All the registers have the same address, but any read access to the trace buffer address causes the counter to increment, thus pointing to the next trace buffer register. The registers are serially available to the external command controller through their common trace buffer address. **Figure 10-10** on page 10-22 shows the block diagram of the trace buffer. The trace buffer is not affected by the operations performed during debug mode except for the trace buffer pointer increment when reading the trace buffer. When entering debug mode, the trace buffer counter is pointing to the trace buffer register containing the address of the last executed instructions. The first trace buffer read obtains the oldest address and the following trace buffer reads get the other addresses from the oldest to the newest, in order of execution.

Notes:

1. To insure trace buffer coherence, a complete set of 12 reads of the Trace buffer must be performed. This is necessary due to the fact that each read increments the trace buffer pointer, thus pointing to the next location. After 12 reads, the pointer indicates the same location as before starting the read procedure.

2. On any change of flow instruction, the trace buffer stores both the address of the change of flow instruction, as well as the address of the target of the change of flow instruction. In the case of conditional change of flows, the address of the change of flow instruction is always stored (regardless of the fact that the change of flow is true or false), but if the conditional change of flow is false (i.e., not taken) the address of the target is not stored. In order to facilitate the program trace reconstruction, every trace buffer location has an additional 'invalid bit' (bit 24). If a conditional change of flow instruction has a 'condition false', the invalid bit is set, thus marking this instruction as not taken. Therefore, it is imperative to read 17 bits of data when reading the 12 trace buffer registers. Since data is read LSB first, the invalid bit is the first bit to be read.

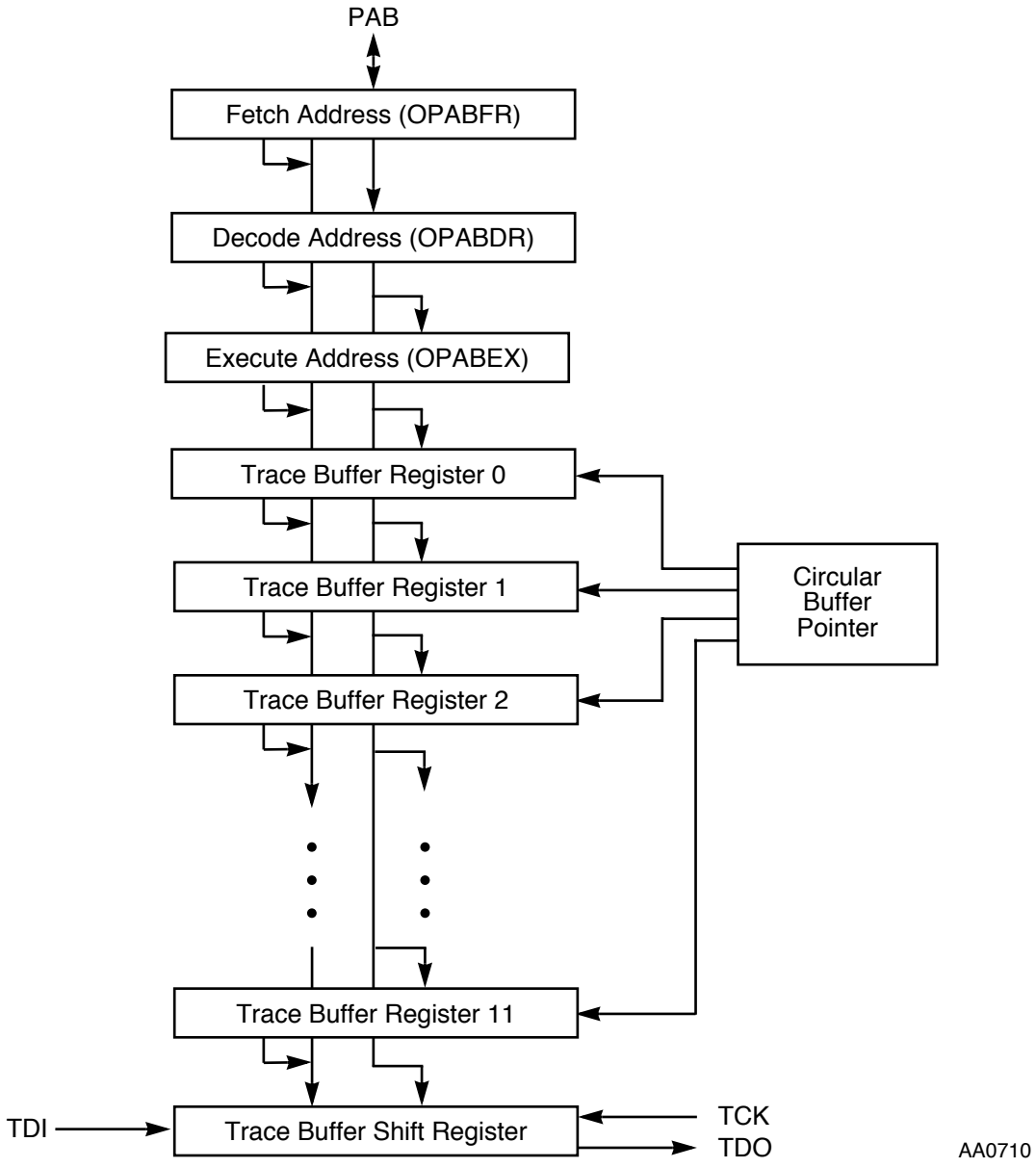


Figure 10-10 OnCE Trace Buffer

10.10 SERIAL PROTOCOL DESCRIPTION

To permit an efficient means of communication between the external command controller and the DSP56300 core chip, the following protocol is adopted. Before starting any debugging activity, the external command controller has to wait for an acknowledge

on the $\overline{DE}$ line indicating that the chip has entered debug mode. Optionally the external command controller can poll the OS1 and OS0 bits in the JTAG instruction shift register. The external command controller communicates with the chip by sending 8-bit commands that can be accompanied by 24 bits of data. Both commands and data are sent or received LSB first. After sending a command, the external command controller should wait for the DSP56300 core chip to acknowledge execution of the command. The external command controller can send a new command only after the chip has acknowledged execution of the previous command.

The OnCE commands are classified as follows:

- Read commands (when the chip delivers the required data)
- Write commands (when the chip receives data and writes the data in one of the OnCE registers)
- Commands that do not have data transfers associated with them

The commands are 8 bits long. The command formats are shown in **Figure 10-4** on page 10-5.

10.11 TARGET SITE DEBUG SYSTEM REQUIREMENTS

A typical debug environment consists of a target system where the DSP56300 core-based device resides in the user defined hardware. The JTAG port interfaces to the external command controller over a 8-wire link consisting of the five JTAG port wires, one OnCE module wire, a ground, and a reset wire. The reset wire is optional and is only used to reset the DSP56300 core-based device and its associated circuitry.

The external command controller acts as the medium between the DSP56300 core target system and a host computer. The external command controller circuit acts as a JTAG port driver and host computer command interpreter. The controller issues commands based on the host computer inputs from a user interface program that communicates with the user.

10.12 OnCE MODULE EXAMPLES

Following are some examples of debugging procedures. All these examples assume that the DSP is the only device in the JTAG chain. If there is more than one device in the chain (additional DSPs or other devices), the other devices can be forced to execute the JTAG BYPASS instruction such as their effect in the serial stream will be one bit per additional

device. The events such as select-DR, select-IR, update-DR, and shift-DR refer to bringing the JTAG TAP in the corresponding state. For a detailed description of the JTAG protocol, see **Section 11—JTAG Port**.

10.12.1 Checking Whether the Chip Has Entered Debug Mode

There are two methods to verify that the chip has entered debug mode:

- Every time the chip enters debug mode, a pulse is generated on the $\overline{DE}$ signal. A pulse is also generated every time the chip acknowledges the execution of an instruction while in debug mode. An external command controller can connect the $\overline{DE}$ line to an interrupt signal in order to sense the acknowledge.
- An external command controller can poll the JTAG instruction shift register for the status bits OS[1:0]. When the chip is in Debug mode, these bits are set to the value 11.

Note: In the following paragraphs, the ACK notation denotes the operation performed by the command controller to check whether debug mode has been entered (either by sensing $\overline{DE}$ or by polling JTAG instruction shift register).

10.12.2 Polling the JTAG Instruction Shift Register

In order to poll the core status bits in the JTAG instruction shift register the following sequence must be performed:

1. Select shift-IR. Passing through capture-IR loads the core status bits into the instruction shift register.
2. Shift in ENABLE_ONCE. While shifting-in the new instruction, the captured status information is shifted-out. Pass through update-IR.
3. Return to Run-Test/Idle.

The external command controller can analyze the information shifted out and detect whether the chip has entered debug mode.

10.12.3 Saving Pipeline Information

The debugging activity is accomplished by means of DSP56300 core instructions supplied from the external command controller. Therefore, the current state of the DSP56300 core pipeline must be saved prior to starting the debug activity and the state must be restored prior to returning to normal mode. Here is the description of the save procedure (it assumes that ENABLE_ONCE has been executed and Debug mode has been entered and verified, as described in **Section 10.12.1—Checking Whether the Chip Has Entered Debug Mode**):

1. Select shift-DR. Shift in the "Read PDB". Pass through update-DR.
2. Select shift-DR. Shift out the 24 bit OPDB register. Pass through update-DR.
3. Select shift-DR. Shift in the "Read PIL". Pass through update-DR.
4. Select shift-DR. Shift out the 24 bit OPILR register. Pass through update-DR.

Note that there is no need to verify acknowledge between steps 1 and 2, as well as 3 and 4, because completion is guaranteed by design.

10.12.4 Reading the Trace Buffer

An optional step during debugging activity is reading the information associated with the trace buffer in order to enable an external program to reconstruct the full trace of the executed program. In the following description of the read trace buffer procedure, it is assumed that all actions described in **Saving Pipeline Information** have been executed.

1. Select shift-DR. Shift in the "Read PABFR". Pass through update-DR.
2. Select shift-DR. Shift out the 16 bit OPABFR register. Pass through update-DR.
3. Select shift-DR. Shift in the "Read PABDR". Pass through update-DR.
4. Select shift-DR. Shift out the 16 bit OPABDR register. Pass through update-DR.
5. Select shift-DR. Shift in the "Read PABEX". Pass through update-DR.
6. Select shift-DR. Shift out the 16 bit OPABEX register. Pass through update-DR.
7. Select shift-DR. Shift in the "Read FIFO". Pass through update-DR.
8. Select shift-DR. Shift out the 17 bit FIFO register. Pass through update-DR.
9. Repeat steps 7 and 8 for the entire FIFO (12 times).

Note: The user must read the entire FIFO, since each read increments the FIFO pointer, thus pointing to the next FIFO location. At the end of this procedure, the FIFO pointer points back to the beginning of the FIFO.

The information that has been read by the external command controller now contains the address of the newly fetched instruction, the address of the instruction currently on the PDB, the address of the instruction currently on the instruction latch, as well as the addresses of the last 12 instructions that have been executed and are change of flow. A user program can now reconstruct the flow of a full trace based on this information and on the original source code of the currently running program.

10.12.5 Displaying a Specified Register

To display a specified register, the DSP56300 must be in debug mode and all actions described in **Section 10.12.3—Saving Pipeline Information** have been executed. The sequence of actions is as follows:

1. Select shift-DR. Shift in the "Write PDB with GO no-EX". Pass through update-DR.
2. Select shift-DR. Shift in the 24-bit opcode: "MOVE reg, X:OGDB". Pass through update-DR to actually write OPDBR and thus begin executing the MOVE instruction.
3. Wait for DSP to reenter debug mode (wait for $\overline{DE}$ or poll core status).
4. Select shift-DR and shift in "READ GDB REGISTER". Pass through update-DR. This step selects OGDBR as the data register for the read.
5. Select shift-DR. Shift out the OGDBR contents. Pass through update-DR. Wait for next command.

10.12.6 Displaying X Memory Area Starting at Address $xxxx

The DSP56309 must be in debug mode and all actions described in **Section 10.12.3—Saving Pipeline Information** must have been executed. Since R0 is used as pointer for the memory, R0 is saved first. The sequence of actions is as follows:

1. Select shift-DR. Shift in the "Write PDB with GO no-EX". Pass through update-DR.

2. Select shift-DR. Shift in the 24-bit opcode: “MOVE R0, X:OGDB”. Pass through update-DR to actually write OPDBR and thus begin executing the MOVE instruction.
3. Wait for DSP to reenter debug mode (wait for $\overline{DE}$ or poll core status).
4. Select shift-DR and shift in “READ GDB REGISTER”. Pass through update-DR. (This selects OGDBR as the data register for read.)
5. Select shift-DR. Shift out the OGDBR contents. Pass through update-DR. R0 is now saved.
6. Select shift-DR. Shift in the “Write PDB with no-GO no-EX”. Pass through update-DR.
7. Select shift-DR. Shift in the 24 bit opcode: “MOVE #$xxxx,R0”. Pass through update-DR to actually write OPDBR.
8. Select shift-DR. Shift in the “Write PDB with GO no-EX”. Pass through update-DR.
9. Select shift-DR. Shift in the second word of the 24 bit opcode: “MOVE #$xxxx,R0” (the $xxxx field). Pass through update-DR to actually write OPDBR and execute the instruction. R0 is loaded with the base address of the memory block to be read.
10. Wait for DSP to reenter debug mode. (Wait for $\overline{DE}$ or poll core status.)
11. Select shift-DR. Shift in the “Write PDB with GO no-EX”. Pass through update-DR.
12. Select shift-DR. Shift in the 24-bit opcode: “MOVE X:(R0)+, X:OGDB”. Pass through update-DR to actually write OPDBR and thus begin executing the MOVE instruction.
13. Wait for DSP to reenter debug mode. (Wait for $\overline{DE}$ or poll core status.)
14. Select shift-DR and shift in “READ GDB REGISTER”. Pass through update-DR. (This selects OGDBR as the data register for read.)
15. Select shift-DR. Shift out the OGDBR contents. Pass through update-DR. The memory contents of address $xxxx have been read.
16. Select shift-DR. Shift in the “NO SELECT with GO no-EX”. Pass through update-DR. This reexecutes the same “MOVE X:(R0)+, X:OGDB” instruction.
17. Repeat from step 14 to complete the reading of the entire block. When finished, restore the original value of R0.

10.12.7 Returning from Debug to Normal Mode (Same Program)

In this case, you have finished examining the current state of the machine, changed some of the registers, and wish to return and continue execution of the same program from the point where it stopped. Therefore, you must restore the pipeline of the machine and enable normal instruction execution. The sequence of actions to do so is listed below:

1. Select shift-DR. Shift in the "Write PDB with no-GO no-EX". Pass through update-DR.
2. Select shift-DR. Shift in the 24 bits of saved PIL (instruction latch value). Pass through update-DR to actually write the instruction latch.
3. Select shift-DR. Shift in the "Write PDB with GO and EX". Pass through update-DR.
4. Select shift-DR. Shift in the 24 bits of saved PDB. Pass through update-DR to actually write the PDB. At the same time the internally saved value of the PAB is driven back from the PABFR register onto the PAB, the ODEC releases the chip from debug mode, and the normal flow of execution is continued.

10.12.8 Returning from Debug to Normal Mode (New Program)

In this case, you have finished examining the current state of the machine, changed some of the registers, and wish to start the execution of a new program (the GOTO command). Therefore, you must force a "change-of-flow" to the starting address of the new program ($xxxx). The sequence of actions to do so is listed below:

1. Select shift-DR. Shift in the "Write PDB with no-GO no-EX". Pass through update-DR.
1. Select shift-DR. Shift in the 24-bit "$0AF080" which is the opcode of the JUMP instruction. Pass through update-DR to actually write the instruction latch.
2. Select shift-DR. Shift in the "Write PDB-GO-TO with GO and EX". Pass through update-DR.
3. Select shift-DR. Shift in the 16 bit of "$xxxx". Pass through update-DR to actually write the PDB. At this time the ODEC releases the chip from debug mode and the execution is started from the address $xxxx.

Note: If the device enters debug mode during a DO LOOP, REP instruction, or other special cases such as interrupt processing, STOP, WAIT, or conditional branching, you must first reset the DSP56300 and then proceed with the execution of the new program.

10.13 JTAG PORT/OnCE MODULE INTERACTION

This subsection lists the details of the JTAG port/OnCE module interaction and TMS sequencing required in order to achieve the communication described in **Section 10.12—OnCE Module Examples**.

The external command controller can force the DSP56300 into debug mode by executing the JTAG instruction DEBUG_REQUEST. In order to check that the DSP56300 has entered debug mode, the external command controller must poll the status by reading the OS[1:0] bits in the JTAG instruction shift register. The TMS sequencing appears in **Table 10-12**.

The sequence to enable the OnCE module appears in **Table 10-13**.

After executing the JTAG instructions DEBUG_REQUEST and ENABLE_ONCE and after the core status was polled to verify that the chip is in debug mode, the pipeline saving procedure must take place. The TMS sequencing for this procedure is depicted in **Table 10-12**.

Table 10-12 TMS Sequencing for DEBUG_REQUEST

Step	TMS	JTAG Port	OnCE Module	Note
a	0	Run-Test/Idle	Idle	—
b	1	Select-DR-Scan	Idle	—
c	1	Select-IR-Scan	Idle	—
d	0	Capture-IR	Idle	The status is sampled in the shifter.
e	0	Shift-IR	Idle	The four bits of the JTAG DEBUG_REQUEST (0111) are shifted in while status is shifted out.
	.....			
e	0	Shift-IR	Idle	
f	1	Exit1-IR	Idle	—
g	1	Update-IR	Idle	The debug request is generated.
h	1	Select-DR-Scan	Idle	—
i	1	Select-IR-Scan	Idle	
j	0	Capture-IR	Idle	The status is sampled in the shifter.

Table 10-12 TMS Sequencing for DEBUG_REQUEST (Continued)

Step	TMS	JTAG Port	OnCE Module	Note
k	0	Shift-IR	Idle	The four bits of the JTAG DEBUG_REQUEST (0111) are shifted in while status is shifted out.
	.....			
k	0	Shift-IR	Idle	
l	1	Exit1-IR	Idle	
m	1	Update-IR	Idle	
n	0	Run-Test/Idle	Idle	This step is repeated, enabling an external command controller to poll the status.
.....				
n	0	Run-Test/Idle	Idle	

In "step n" the external command controller verifies that the OS[1:0] bits have the value 11, indicating that the chip has entered debug mode. If the chip has not yet entered debug mode, the external command controller goes to "step b", "step c" etc. until debug mode is acknowledged.

Table 10-13 TMS Sequencing for ENABLE_ONCE

Step	TMS	JTAG Port	OnCE Module	Note
a	1	Test-Logic-Reset	Idle	—
b	0	Run-Test/Idle	Idle	—
c	1	Select-DR-Scan	Idle	—
d	1	Select-IR-Scan	Idle	—
e	0	Capture-IR	Idle	The core status bits are captured.
f	0	Shift-IR	Idle	The four bits of the JTAG ENABLE_ONCE instruction (0110) are shifted into the JTAG instruction register while status is shifted out.
g	0	Shift-IR	Idle	
h	0	Shift-IR	Idle	
i	0	Shift-IR	Idle	
j	1	Exit1-IR	Idle	—
k	1	Update-IR	Idle	The OnCE module is enabled.

Table 10-13 TMS Sequencing for ENABLE_ONCE (Continued)

Step	TMS	JTAG Port	OnCE Module	Note
l	0	Run-Test/Idle	Idle	This step can be repeated, enabling an external command controller to poll the status.
.....				
l	0	Run-Test/Idle	Idle	

Table 10-14 TMS Sequencing for Reading Pipeline Registers

Step	TMS	JTAG Port	OnCE Module	Note
a	0	Run-Test/Idle	Idle	—
b	1	Select-DR-Scan	Idle	—
c	0	Capture-DR	Idle	—
d	0	Shift-DR	Idle	The eight bits of the OnCE command "Read PIL" (10001011) are shifted in.
.....				
d	0	Shift-DR	Idle	
e	1	Exit1-DR	Idle	—
f	1	Update-DR	Execute "Read PIL"	The PIL value is loaded in the shifter.
g	1	Select-DR-Scan	Idle	—
h	0	Capture-DR	Idle	—
i	0	Shift-DR	Idle	The 24 bits of the PIL are shifted out (24 steps).
.....				
i	0	Shift-DR	Idle	
j	1	Exit1-DR	Idle	—
k	1	Update-DR	Idle	—
l	1	Select-DR-Scan	Idle	—
m	0	Capture-DR	Idle	—

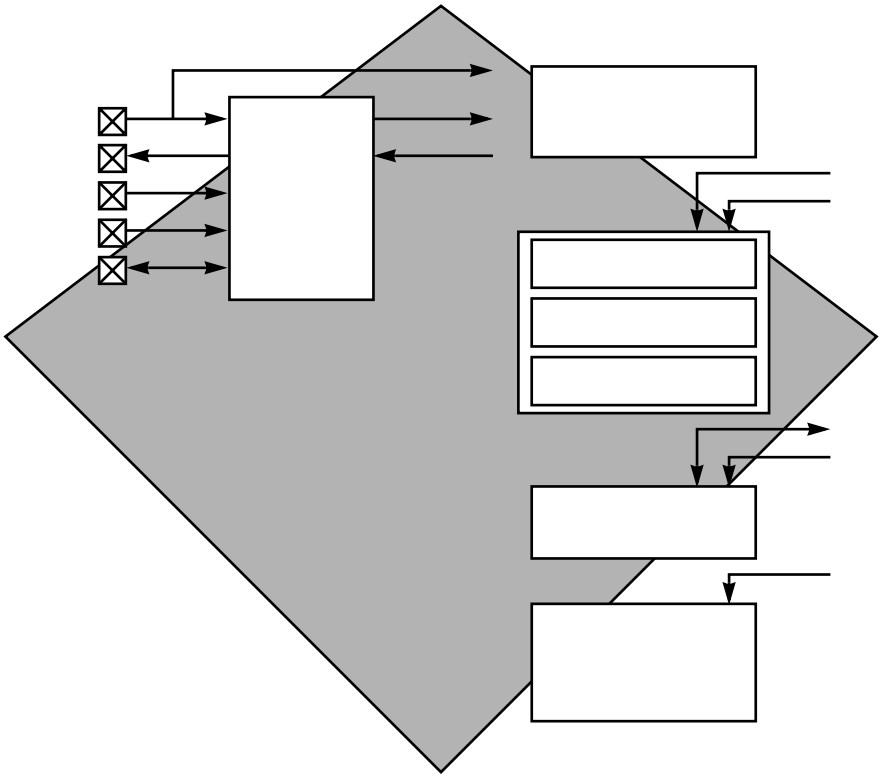
Table 10-14 TMS Sequencing for Reading Pipeline Registers (Continued)

Step	TMS	JTAG Port	OnCE Module	Note
n	0	Shift-DR	Idle	The eight bits of the OnCE command "Read PDB" (10001010) are shifted in.
.....				
n	0	Shift-DR	Idle	
o	1	Exit1-DR	Idle	—
p	1	Update-DR	Execute "Read PDB"	PDB value is loaded in shifter.
q	1	Select-DR-Scan	Idle	—
r	0	Capture-DR	Idle	—
s	0	Shift-DR	Idle	The 24 bits of the PDB are shifted out (24 steps).
.....				
s	0	Shift-DR	Idle	
t	1	Exit1-DR	Idle	—
u	1	Update-DR	Idle	—
v	0	Run-Test/Idle	Idle	This step can be repeated, enabling an external command controller to analyze the information.
.....				
v	0	Run-Test/Idle	Idle	

During "step v" the external command controller stores the pipeline information. Afterwards, it can proceed with the debug activities as requested by the user.

SECTION 11

JTAG PORT



11.1 INTRODUCTION

The DSP56300 core provides a dedicated user-accessible test access port (TAP) that is fully compatible with the *I*EEE 1149.1 Standard Test Access Port and Boundary Scan Architecture. Problems associated with testing high density circuit boards have led to development of this proposed standard under the sponsorship of the Test Technology Committee of IEEE and the JTAG. The DSP56300 core implementation supports circuit-board test strategies based on this standard.

The test logic includes a TAP that consists of five dedicated signals, a 16-state controller, and three test data registers. A Boundary Scan Register (BSR) links all device signals into a single shift register. The test logic, implemented utilizing static logic design, is independent of the device system logic. The DSP56300 core implementation provides the following capabilities:

- Performs boundary scan operations to test circuit-board electrical continuity (EXTEST)
- Bypasses the DSP56300 core for a given circuit-board test by effectively reducing the BSR to a single cell (BYPASS)
- Samples the DSP56300 core-based device system signals during operation and transparently shifts out the result in the BSR
- Preloads values to output signals prior to invoking the EXTEST instruction (SAMPLE/PRELOAD)
- Disables the output drive to signals during circuit-board testing (HI-Z)
- Provides a means of accessing the OnCE controller and circuits to control a target system (ENABLE_ONCE)
- Provides a means of entering debug Mode (DEBUG_REQUEST)
- Queries identification information (manufacturer, part number and version) from a DSP56300 core-based device (IDCODE)
- Forces test data onto the outputs of a DSP56300 core-based device while replacing its boundary scan register in the serial data path with a single bit register (CLAMP)

This section, which includes aspects of the JTAG implementation that are specific to the DSP56300 core, is intended to be used with the supporting IEEE 1149.1 document. The discussion includes those items required by the standard to be defined and, in certain cases, provides additional information specific to the DSP56300 core implementation. For internal details and applications of the standard, refer to the IEEE 1149.1 document. **Figure 11-1** shows a block diagram of the TAP port.

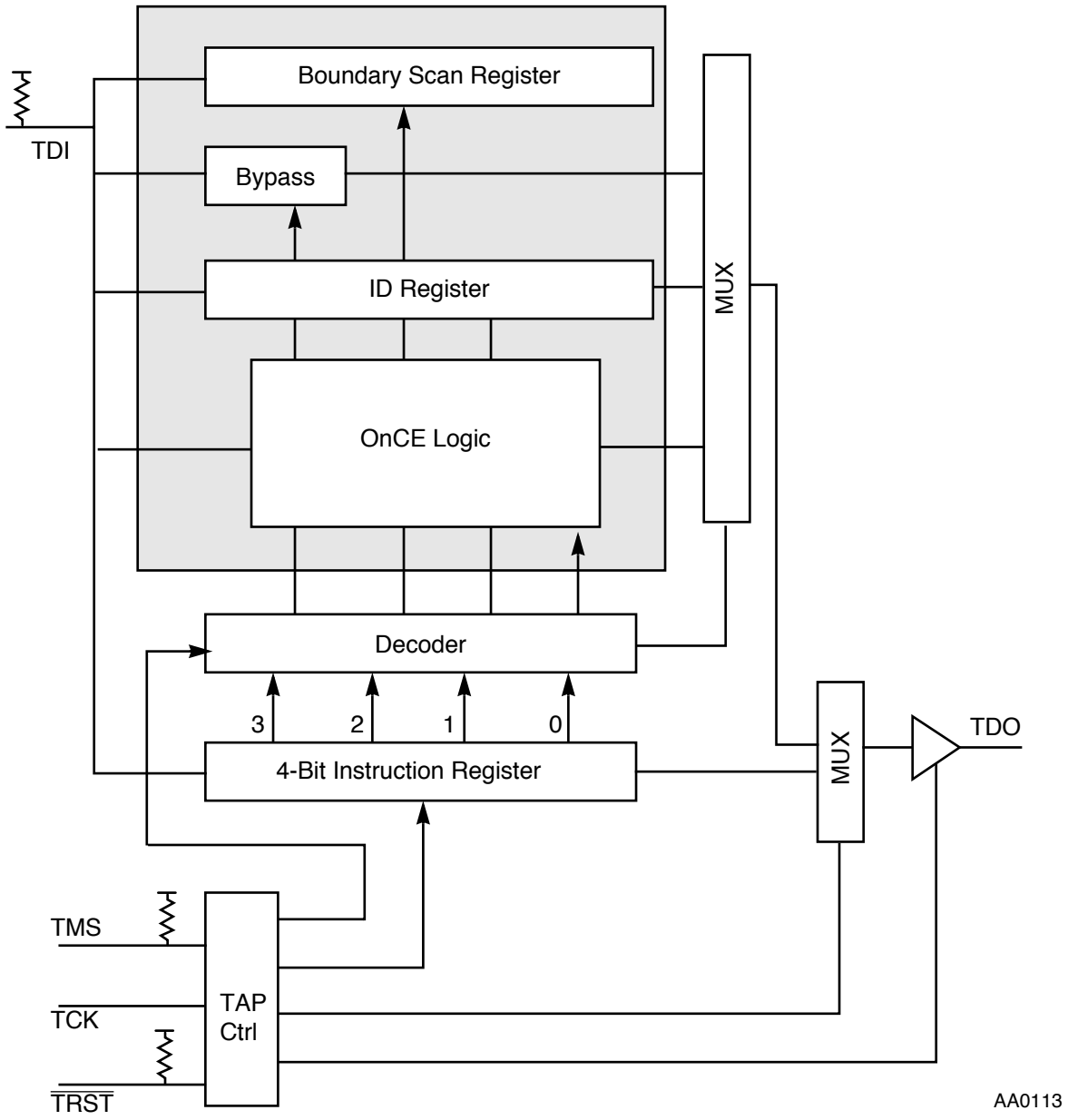


Figure 11-1 TAP Block Diagram

11.2 JTAG SIGNALS

As described in the IEEE 1149.1 document, the JTAG port requires a minimum of four signals to support TDI, TDO, TCK, and TMS signals. The DSP56300 family also provides

the optional $\overline{\text{TRST}}$ signal. On the DSP56309, the debug event ($\overline{\text{DE}}$) signal is provided for use by the OnCE module; it is documented in **Section 10—On-Chip Emulation Module**. The signal functions are described in the following paragraphs.

11.2.1 Test Clock (TCK)

The TCK signal is used to synchronize the test logic.

11.2.2 Test Mode Select (TMS)

The TMS signal is used to sequence the test controller's state machine. The TMS is sampled on the rising edge of TCK, and it has an internal pull-up resistor.

11.2.3 Test Data Input (TDI)

Serial test instruction and data are received through the Test Data Input (TDI) signal. TDI is sampled on the rising edge of TCK, and it has an internal pull-up resistor.

11.2.4 Test Data Output (TDO)

The TDO signal is the serial output for test instructions and data. TDO is tri-stateable and is actively driven in the Shift-IR and Shift-DR controller states. TDO changes on the falling edge of TCK.

11.2.5 Test Reset ($\overline{\text{TRST}}$)

The $\overline{\text{TRST}}$ signal is used to asynchronously initialize the test controller. The $\overline{\text{TRST}}$ signal has an internal pullup resistor.

11.3 TAP CONTROLLER

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The state machine is shown in **Figure 11-2**. 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 represents the value of the TMS signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, refer to the IEEE 1149.1 document.

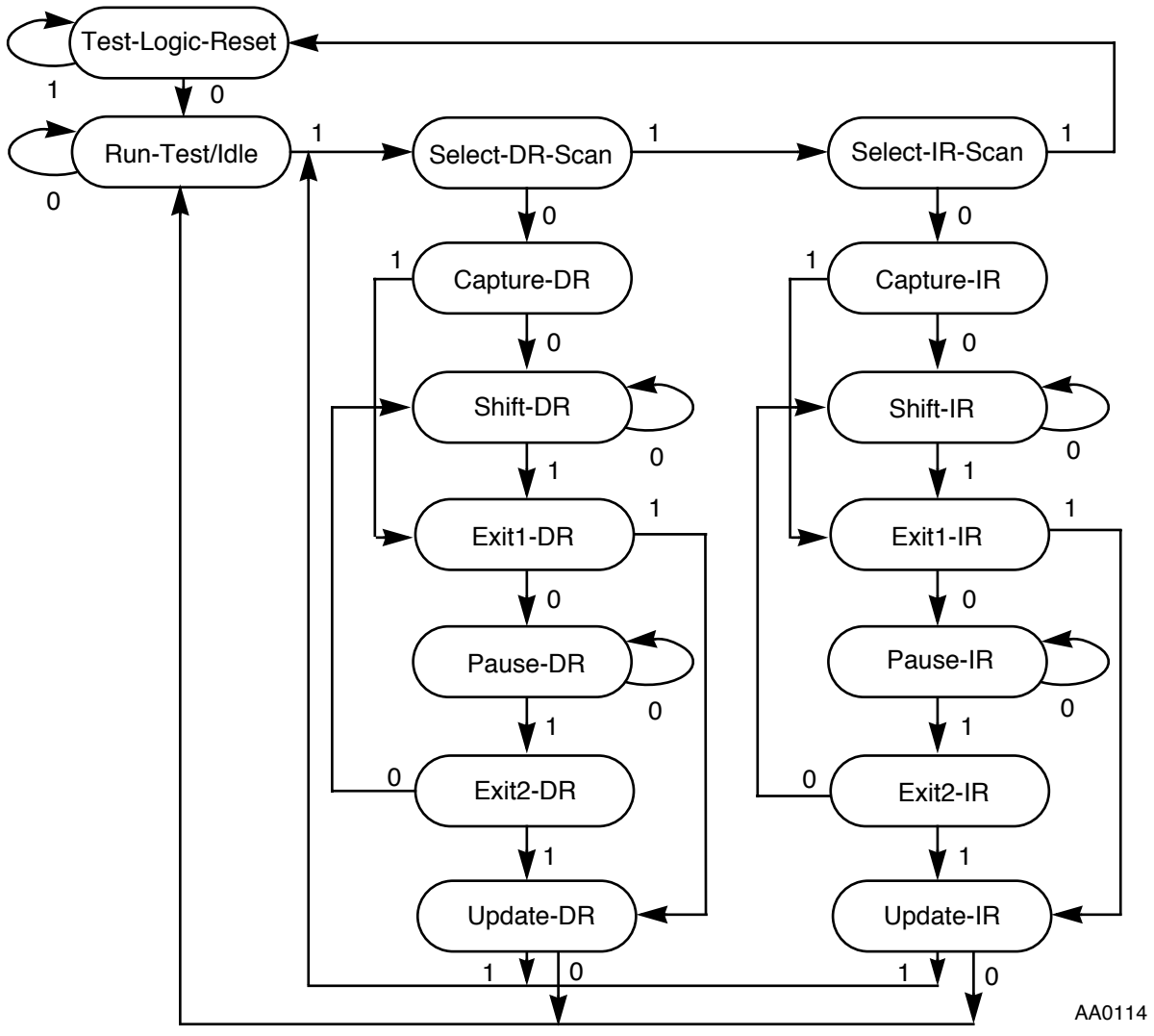


Figure 11-2 TAP Controller State Machine

11.3.1 Boundary Scan Register (BSR)

The BSR in the DSP56309 JTAG implementation contains bits for all device signal and clock signals and associated control signals. All DSP56309 bidirectional signals have a single register bit in the BSR for signal data; each such signal is controlled by an associated control bit in the BSR. The DSP56309 BSR bit definitions are described in **Table 11-2** on page 11-13.

11.3.2 Instruction Register

The DSP56309 JTAG implementation includes the three mandatory public instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS), and also supports the optional CLAMP instruction defined by IEEE 1149.1. The HI-Z public instruction provides the capability for disabling all device output drivers. The ENABLE_ONCE public instruction enables the JTAG port to communicate with the OnCE circuitry. The DEBUG_REQUEST public instruction enables the JTAG port to force the DSP56300 core into debug mode. The DSP56300 core includes a 4-bit instruction register without parity consisting of a shift register with four parallel outputs. Data is transferred from the shift register to the parallel outputs during the Update-IR controller state. **Figure 11-3** shows the JTAG instruction register.

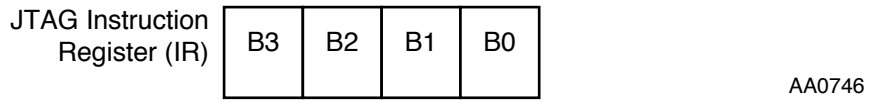


Figure 11-3 JTAG Instruction Register

The four bits are used to decode the eight unique instructions shown in **Table 11-1**. All other encodings are reserved for future enhancements and are decoded as BYPASS.

Table 11-1 JTAG Instructions

Code				Instruction
B3	B2	B1	B0	
0	0	0	0	EXTEST
0	0	0	1	SAMPLE/PRELOAD
0	0	1	0	IDCODE
0	0	1	1	CLAMP
0	1	0	0	HI-Z
0	1	0	1	RESERVED
0	1	1	0	ENABLE_ONCE
0	1	1	1	DEBUG_REQUEST
1	0	x	x	RESERVED
1	1	0	x	RESERVED
1	1	1	0	RESERVED
1	1	1	1	BYPASS

The parallel output of the instruction register is reset to 0010 in the Test-Logic-Reset controller state, which is equivalent to the IDCODE instruction.

During the Capture-IR controller state, the parallel inputs to the instruction shift register are loaded with 01 in the LSBs as required by the standard. The two MSBs are loaded with the values of the core status bits OS1 and OS0 from the OnCE controller. See **Section 10—On-Chip Emulation Module** for a description of the status bits.

11.3.2.1 EXTEST (B[3:0] = 0000)

The external test (EXTEST) instruction selects the BSR. EXTEST also asserts internal reset for the DSP56300 core system logic to force a predictable internal state while performing external boundary scan operations.

By using the TAP, the BSR is capable of the following:

- Scanning user-defined values into the output buffers
- Capturing values presented to input signals

- Controlling the direction of bidirectional signals
- Controlling the output drive of tri-stateable output signals

For more details on the function and use of the EXTEST instruction, please refer to the IEEE 1149.1 document.

11.3.2.2 SAMPLE/PRELOAD (B[3:0] = 0001)

The SAMPLE/PRELOAD instruction provides two separate functions. First, it provides a means to obtain a snapshot of system data and control signals. 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.

Note: Because there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

The second function of the SAMPLE/PRELOAD instruction is to initialize the BSR output cells prior to selection of EXTEST. This initialization insures that known data appears on the outputs when entering the EXTEST instruction.

11.3.2.3 IDCODE (B[3:0] = 0010)

The IDCODE instruction selects the ID register. This instruction is provided as a public instruction to allow the manufacturer, part number, and version of a component to be determined through the TAP. **Figure 11-4** shows the ID register configuration.

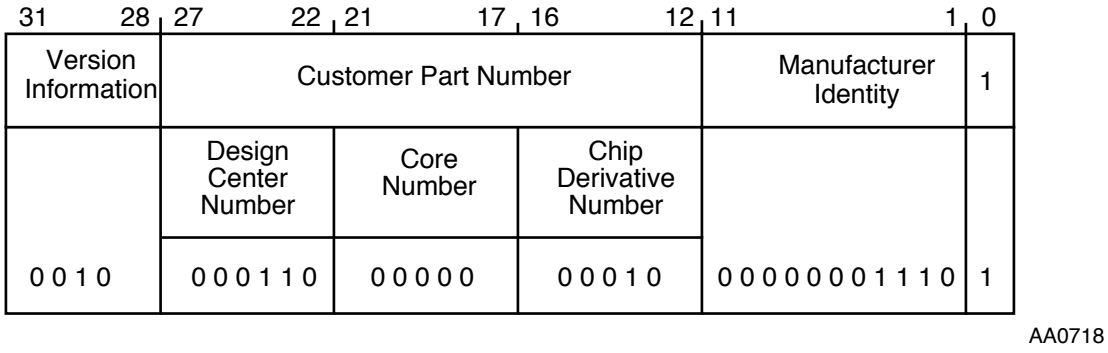


Figure 11-4 JTAG ID Register

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to the IEEE 1149.1 standard, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design in order to determine the type of each component in

each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Motorola's manufacturer identity is 00000001110. The customer part number consists of two parts: Motorola design center number (bits 27:22) and a sequence number (bits 21:12). The sequence number is divided into two parts: core number (bits 21:17) and chip derivative number (bits 16:12). Motorola Semiconductor Israel (MSIL) design center number is 000110 and DSP56300 core number is 00001.

Once the IDCODE instruction is decoded, it selects the ID register, which is a 32-bit data register. Since the Bypass register loads a logical 0 at the start of a scan cycle, whereas the ID register loads a logical 1 into its LSB, examination of the first bit of data shifted out of a component during a test data scan sequence immediately following exit from Test-Logic-Reset controller state shows whether such a register 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 by the IEEE 1149.1 standard.

11.3.2.4 CLAMP (B[3:0] = 0011)

The CLAMP instruction is not included in the IEEE 1149.1 standard. It is provided as a public instruction that selects the 1-bit bypass register as the serial path between TDI and TDO while allowing signals driven from the component signals to be determined from the BSR. During testing of ICs on PCB, it may be necessary to place static guarding values on signals that control operation of logic not involved in the test. The EXTEST instruction could be used for this purpose, but because it selects the BSR, the required guarding signals would be loaded as part of the complete serial data stream shifted in, both at the start of the test and each time a new test pattern is entered. Since the CLAMP instruction allows guarding values to be applied using the BSR of the appropriate ICs while selecting their bypass registers, it allows much faster testing than does the EXTEST instruction. 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. The CLAMP instruction also asserts internal reset for the DSP56300 core system logic to force a predictable internal state while performing external boundary scan operations.

11.3.2.5 HI-Z (B[3:0] = 0100)

The HI-Z instruction is not included in the IEEE 1149.1 standard. It is provided as a manufacturer's optional public instruction to prevent having to backdrive the output signals during circuit-board testing. When HI-Z is invoked, all output drivers, including the two-state drivers, are turned off (i.e., high impedance). The instruction selects the bypass register. The HI-Z instruction also asserts internal reset for the DSP56300 core system logic to force a predictable internal state while performing external boundary scan operations.

11.3.2.6 ENABLE_ONCE(B[3:0] = 0110)

The ENABLE_ONCE instruction is not included in the IEEE 1149.1 standard. It is provided as a public instruction to allow you to perform system debug functions. When the ENABLE_ONCE instruction is decoded the TDI and TDO signals are connected directly to the OnCE registers. The particular OnCE register connected between TDI and TDO at a given time is selected by the OnCE controller depending on the OnCE instruction being currently executed. All communication with the OnCE controller is done through the Select-DR-Scan path of the JTAG TAP controller. See **Section 10—On-Chip Emulation Module** for more information.

11.3.2.7 DEBUG_REQUEST(B[3:0] = 0111)

The DEBUG_REQUEST instruction is not included in the IEEE 1149.1 standard. It is provided as a public instruction to allow you to generate a debug request signal to the DSP56300 core. When the DEBUG_REQUEST instruction is decoded, the TDI and TDO signals are connected to the instruction registers. Due to the fact that in the Capture-IR state of the TAP the OnCE status bits are captured in the Instruction shift register, the external JTAG controller must continue to shift-in the DEBUG_REQUEST instruction while polling the status bits that are shifted-out until debug mode is entered (acknowledged by the combination 11 on OS1–OS0). After the acknowledgment of debug mode is received, the external JTAG controller must issue the ENABLE_ONCE instruction to allow the user to perform system debug functions.

11.3.2.8 BYPASS (B[3:0] = 1111)

The BYPASS instruction selects the single-bit bypass register, as shown in **Figure 11-5**. This choice creates a shift-register path from TDI to the bypass register, and finally to TDO, circumventing the BSR. This instruction is used to enhance test efficiency when a component other than the DSP56300 core-based device becomes the device under test. When the bypass register is selected by the current instruction, the shift-register stage is set to a logical 0 on the rising edge of TCK in the Capture-DR controller state. Therefore, the first bit shifted out after selecting the bypass register is always a logical 0.

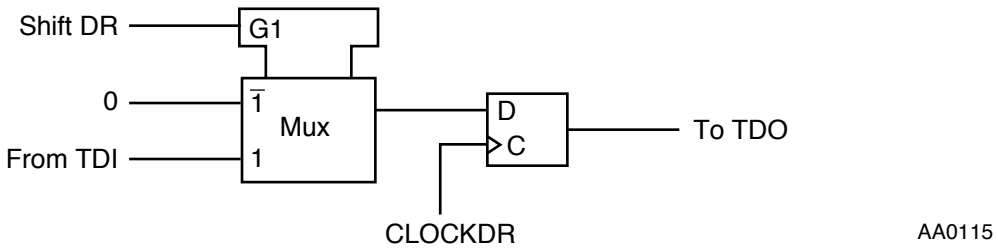


Figure 11-5 Bypass Register

11.4 DSP56300 RESTRICTIONS

The control afforded by the output enable signals using the BSR and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. You must avoid situations in which the DSP56300 core output drivers are enabled into actively driven networks. In addition, the EXTEST instruction can be performed only after power-up or a regular hardware $\overline{\text{RESET}}$ signal while EXTAL was provided. Then during the execution of EXTEST, EXTAL can remain inactive.

There are two constraints related to the JTAG interface. First, the TCK input does not include an internal pullup resistor and should not be left unconnected. The second constraint is to insure that the JTAG test logic is kept transparent to the system logic by forcing the TAP into the Test-Logic-Reset controller state, using either of two methods. During power-up, $\overline{\text{TRST}}$ must be externally asserted to force the TAP controller into this state. After power-up is concluded, TMS must be sampled as a logical 1 for five consecutive TCK rising edges. If TMS either remains unconnected or is connected to V_{CC}, then the TAP controller cannot leave the Test-Logic-Reset state, regardless of the state of TCK.

The DSP56300 core features a low-power stop mode, which is invoked using the STOP instruction. The interaction of the JTAG interface 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 the low-power 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_{CC} or GND.
3. The TMS and TDI signals include on-chip pullup resistors. In low-power stop mode, these two signals should remain either unconnected or connected to V_{CC} to achieve minimal power consumption.

Since during stop mode all DSP56309 core clocks are disabled, the JTAG interface provides the means of polling the device status (sampled in the Capture-IR state).

11.5 DSP56309 BOUNDARY SCAN REGISTER

Table 11-2 provides a listing of the contents of the BSR for the DSP56309.

Table 11-2 DSP56309 BSR Bit Definitions

Bit #	Cell Type	Signal Name	Signal Type	BSR Cell Type
0	BC_1	MODA	Input	Data
1	BC_1	MODB	Input	Data
2	BC_1	MODC	Input	Data
3	BC_1	MODD	Input	Data
4	BC_6	D23	Input/Output	Data
5	BC_6	D22	Input/Output	Data
6	BC_6	D21	Input/Output	Data
7	BC_6	D20	Input/Output	Data
8	BC_6	D19	Input/Output	Data
9	BC_6	D18	Input/Output	Data
10	BC_6	D17	Input/Output	Data
11	BC_6	D16	Input/Output	Data
12	BC_6	D15	Input/Output	Data
13	BC_1	D[23:12]	—	Control
14	BC_6	D14	Input/Output	Data
15	BC_6	D13	Input/Output	Data
16	BC_6	D12	Input/Output	Data
17	BC_6	D11	Input/Output	Data
18	BC_6	D10	Input/Output	Data
19	BC_6	D9	Input/Output	Data
20	BC_6	D8	Input/Output	Data
21	BC_6	D7	Input/Output	Data

Table 11-2 DSP56309 BSR Bit Definitions (Continued)

Bit #	Cell Type	Signal Name	Signal Type	BSR Cell Type
22	BC_6	D6	Input/Output	Data
23	BC_6	D5	Input/Output	Data
24	BC_6	D4	Input/Output	Data
25	BC_6	D3	Input/Output	Data
26	BC_1	D[11:0]	—	Control
27	BC_6	D2	Input/Output	Data
28	BC_6	D1	Input/Output	Data
29	BC_6	D0	Input/Output	Data
30	BC_2	A15	Output 2	Data
31	BC_2	A14	Output 2	Data
32	BC_2	A13	Output 2	Data
33	BC_2	A12	Output 2	Data
34	BC_2	A11	Output 2	Data
35	BC_2	A10	Output 2	Data
36	BC_2	A9	Output 2	Data
37	BC_2	A8	Output 2	Data
38	BC_2	A7	Output 2	Data
39	BC_2	A6	Output 2	Data
40	BC_2	A5	Output 2	Data
41	BC_2	A4	Output 2	Data
42	BC_2	A3	Output 2	Data
43	BC_2	A2	Output 2	Data
44	BC_2	A1	Output 2	Data
45	BC_2	A0	Output 2	Data

Table 11-2 DSP56309 BSR Bit Definitions (Continued)

Bit #	Cell Type	Signal Name	Signal Type	BSR Cell Type
46	BC_2	$\overline{\text{MCS}}$	Output	Data
47	BC_2	$\overline{\text{RD}}$	Output	Data
48	BC_2	$\overline{\text{WR}}$	Output	Data
49	BC_2	$\overline{\text{AT}}$	Output	Data
50	BC_2	CLKOUT	Output	Data
51	BC_1	EXTAL	Input	Data
52	BC_1	$\overline{\text{RESET}}$	Input	Data
53	BC_1	HAD0	—	Control
54	BC_6	HAD0	Input/Output	Data
55	BC_1	HAD1	—	Control
56	BC_6	HAD1	Input/Output	Data
57	BC_1	HAD2	—	Control
58	BC_6	HAD2	Input/Output	Data
59	BC_1	HAD3	—	Control
60	BC_6	HAD3	Input/Output	Data
61	BC_1	HAD4	—	Control
62	BC_6	HAD4	Input/Output	Data
63	BC_1	HAD5	—	Control
64	BC_6	HAD5	Input/Output	Data
65	BC_1	HAD6	—	Control
66	BC_6	HAD6	Input/Output	Data
67	BC_1	HAD7	—	Control
68	BC_6	HAD7	Input/Output	Data
69	BC_1	HAS/A0	—	Control

Table 11-2 DSP56309 BSR Bit Definitions (Continued)

Bit #	Cell Type	Signal Name	Signal Type	BSR Cell Type
70	BC_6	HAS/A0	Input/Output	Data
71	BC_1	HA8/A1	—	Control
72	BC_6	HA8/A1	Input/Output	Data
73	BC_1	HA9/A2	—	Control
74	BC_6	HA9/A2	Input/Output	Data
75	BC_1	HCS/A10	—	Control
76	BC_6	HCS/A10	Input/Output	Data
77	BC_1	TIO0	—	Control
78	BC_6	TIO0	Input/Output	Data
79	BC_1	TIO1	—	Control
80	BC_6	TIO1	Input/Output	Data
81	BC_1	TIO2	—-	Control
82	BC_6	TIO2	Input/Output	Data
83	BC_1	HREQ/TRQ	—	Control
84	BC_6	HREQ/TRQ	Input/Output	Data
85	BC_1	HACK/RRQ	—	Control
86	BC_6	HACK/RRQ	Input/Output	Data
87	BC_1	HRW/RD	—	Control
88	BC_6	HRW/RD	Input/Output	Data
89	BC_1	HDS/WR	—	Control
90	BC_6	HDS/WR	Input/Output	Data
91	BC_1	SCK0	—	Control
92	BC_6	SCK0	Input/Output	Data
93	BC_1	SCK1	—	Control

Table 11-2 DSP56309 BSR Bit Definitions (Continued)

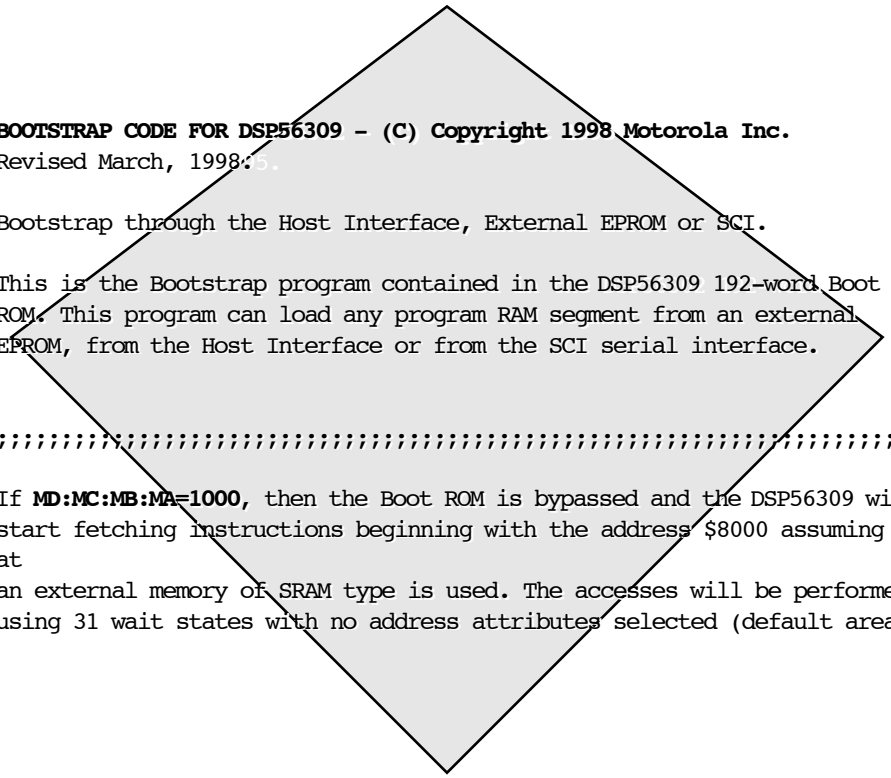
Bit #	Cell Type	Signal Name	Signal Type	BSR Cell Type
94	BC_6	SCK1	Input/Output	Data
95	BC_1	GPIO2	—	Control
96	BC_6	GPIO2	Input/Output	Data
97	BC_1	GPIO1	—	Control
98	BC_6	GPIO1	Input/Output	Data
99	BC_1	GPIO0	—	Control
100	BC_6	GPIO0	Input/Output	Data
101	BC_1	SC00	—	Control
102	BC_6	SC00	Input/Output	Data
103	BC_1	SC10	—	Control
104	BC_6	SC10	Input/Output	Data
105	BC_1	STD0	—	Control
106	BC_6	STD0	Input/Output	Data
107	BC_1	SRD0	—	Control
108	BC_6	SRD0	Input/Output	Data
109	BC_1	PINIT	—	Control
110	BC_6	PINIT	Input/Output	Data
111	BC_1	$\overline{DE}$	—	Control
112	BC_6	$\overline{DE}$	Input/Output	Data
113	BC_1	SC01	—	Control
114	BC_6	SC01	Input/Output	Data
115	BC_1	SC02	—	Control
116	BC_6	SC02	Input/Output	Data
117	BC_1	STD1	—	Control

Table 11-2 DSP56309 BSR Bit Definitions (Continued)

Bit #	Cell Type	Signal Name	Signal Type	BSR Cell Type
118	BC_6	STD1	Input/Output	Data
119	BC_1	SRD1	—	Control
120	BC_6	SRD1	Input/Output	Data
121	BC_1	SC11	—	Control
122	BC_6	SC11	Input/Output	Data
123	BC_1	SC12	—	Control

APPENDIX A

BOOTSTRAP PROGRAMS



```
; BOOTSTRAP CODE FOR DSP56309 - (C) Copyright 1998 Motorola Inc.
; Revised March, 1998.
;
; Bootstrap through the Host Interface, External EPROM or SCI.
;
; This is the Bootstrap program contained in the DSP56309 192-word Boot
; ROM. This program can load any program RAM segment from an external
; EPROM, from the Host Interface or from the SCI serial interface.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; If MD:MC:MB:MA=1000, then the Boot ROM is bypassed and the DSP56309 will
; start fetching instructions beginning with the address $8000 assuming
that
; an external memory of SRAM type is used. The accesses will be performed
; using 31 wait states with no address attributes selected (default area).
```

```
; BOOTSTRAP CODE FOR DSP56309 - (C) Copyright 1997 Motorola Inc.
; Revised March, 18 1997.
;
; Bootstrap through the Host Interface, External EPROM or SCI.
;
; This is the Bootstrap program contained in the DSP56309 192-word Boot
; ROM. This program can load any program RAM segment from an external
; EPROM, from the Host Interface or from the SCI serial interface.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=x000, then the Boot ROM is bypassed and the DSP56309
; will start fetching instructions beginning with address $C00000 (MD=0)
; or $008000 (MD=1) assuming that an external memory of SRAM type is
; used. The accesses will be performed using 31 wait states with no
; address attributes selected (default area).
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Operation modes MD:MC:MB:MA=0001-0111 are reserved.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1001, then it loads a program RAM segment from consecutive
; byte-wide P memory locations, starting at P:$D00000 (bits 7-0).
; The memory is selected by the Address Attribute AA1 and is accessed with
; 31 wait states.
; The EPROM bootstrap code expects to read 3 bytes
; specifying the number of program words, 3 bytes specifying the address
; to start loading the program words and then 3 bytes for each program
; word to be loaded. The number of words, the starting address and the
; program words are read least significant byte first followed by the
; mid and then by the most significant byte.
; The program words will be condensed into 24-bit words and stored in
; contiguous PRAM memory locations starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1010, then it loads the program RAM from the SCI interface.
; The number of program words to be loaded and the starting address must
; be specified. The SCI bootstrap code expects to receive 3 bytes
; specifying the number of program words, 3 bytes specifying the address
; to start loading the program words and then 3 bytes for each program
; word to be loaded. The number of words, the starting address and the
; program words are received least significant byte first followed by the
; mid and then by the most significant byte. After receiving the
; program words, program execution starts in the same address where
; loading started. The SCI is programmed to work in asynchronous mode
; with 8 data bits, 1 stop bit and no parity. The clock source is
; external and the clock frequency must be 16x the baud rate.
; After each byte is received, it is echoed back through the SCI
```

```
; transmitter.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Operation mode MD:MC:MB:MA=1011 is reserved.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1100, then it loads the program RAM from the Host
; Interface programmed to operate in the ISA mode.
; The HOST ISA bootstrap code expects to read a 24-bit word
; specifying the number of program words, a 24-bit word specifying the address
; to start loading the program words and then a 24-bit word for each program
; word to be loaded. The program words  will be stored in
; contiguous PRAM memory locations starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by
; setting the Host Flag 0 (HF0). This will start execution of the loaded
; program from the specified starting address.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1101, then it loads the program RAM from the Host
; Interface programmed to operate in the HC11 non multiplexed mode.
;
; The HOST HC11  bootstrap code expects to read a 24-bit word
; specifying the number of program words, a 24-bit word specifying the address
; to start loading the program words and then a 24-bit word for each program
; word to be loaded. The program words  will be stored in
; contiguous PRAM memory locations starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by
; setting the Host Flag 0 (HF0). This will start execution of the loaded
; program from the specified starting address.
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1110, then it loads the program RAM from the Host
; Interface programmed to operate in the 8051 multiplexed bus mode,
; in double-strob pin configuration.
; The HOST 8051 bootstrap code expects accesses that are byte wide.
; The HOST 8051 bootstrap code expects to read 3 bytes forming a 24-bit word
; specifying the number of program words, 3 bytes forming a 24-bit word
; specifying the address to start loading the program words and then 3 bytes
; forming 24-bit words for each program word to be loaded.
; The program words  will be stored in contiguous PRAM memory locations
; starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0). This will start execution of the loaded program from
```

```
; the specified starting address.
;
; The base address of the HI08 in multiplexed mode is 0x80 and is not
; modified by the bootstrap code. All the address lines are enabled
; and should be connected accordingly.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1111, then it loads the program RAM from the Host
; Interface programmed to operate in the MC68302 (IMP) bus mode,
; in single-strob pin configuration.
; The HOST MC68302 bootstrap code expects accesses that are byte wide.
; The HOST MC68302 bootstrap code expects to read 3 bytes forming a 24-bit word
; specifying the number of program words, 3 bytes forming a 24-bit word
; specifying the address to start loading the program words and then 3 bytes
; forming 24-bit words for each program word to be loaded.
; The program words  will be stored in contiguous PRAM memory locations
; starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0). This will start execution of the loaded program from
; the specified starting address.
;

;;;;;;;;;;;;;;;;;;; MEMORY EQUATES ;;;;;;;;;;;;;;;;;;;;;;;;
        page    132,55,0,0,0
                opt     mex


EQUALDATA       equ     1       ;; 1 if xram and yram are of equal
                                ;; size and addresses, 0 otherwise.

        if      (EQUALDATA)
start_dram      equ     0       ;; 24k X and Y RAM
length_dram     equ     $1c00   ;; same addresses
        else
start_xram      equ     0       ;; 7k XRAM
length_xram     equ     $1c00
start_yram      equ     0       ;; 7k YRAM
length_yram     equ     $1c00
        endif

start_pram      equ     0       ;; 20k PRAM
length_pram     equ     $5000

;;
;;;;;;;;;;;;;;;;;;; GENERAL EQUATES ;;;;;;;;;;;;;;;;;;;;;;;;
;;

BOOT    equ     $D00000         ; this is the location in P memory
                                ; on the external memory bus
                                ; where the external byte-wide
                                ; EPROM would be located
```

```
AARV    equ     $D00409         ; AAR1 selects the EPROM as CE~
                                ; mapped as P from $D00000 to
                                ; $DFFFFF, active low

;;
;;;;;;;;;;;;;;;;;;; DSP I/O REGISTERS ;;;;;;;;;;;;;;;;;;;;;;;;
;;

M_PDRC  EQU     $FFFFBD         ;; Port C GPIO Data Register
M_PRRC  EQU     $FFFFBE         ;; Port C Direction Register
M_SSR   EQU     $FFFF93         ; SCI Status Register
M_STXL  EQU     $FFFF95         ; SCI Transmit Data Register (low)
M_SRXL  EQU     $FFFF98         ; SCI Receive Data Register (low)
M_SCCR  EQU     $FFFF9B         ; SCI Clock Control Register
M_SCR   EQU     $FFFF9C         ; SCI Control Register
M_PCRE  EQU     $FFFF9F         ; Port E Control register
M_AAR1  EQU     $FFFFF8         ; Address Attribute Register 1
M_HPCR  EQU     $FFFFC4         ; Host Polarity Control Register
M_HSR   EQU     $FFFFC3         ; Host Status Rgister
M_HRX   EQU     $FFFFC6         ; Host Recceive Register
HRDF    EQU     $0              ; Host Receive Data Full
HF0     EQU     $3              ; Host Flag 0
HEN     EQU     $6              ; Host Enable
SCK0    EQU     $3              ;; SCK0 is bit #3 as GPIO



        ORG PL:$ff0000,PL:$ff0000    ; bootstrap code starts at $ff0000

START
        clr a                   ; clear a
        jclr #3,omr,OMR0XXX     ; If MD:MC:MB:MA=0xxx, go to OMR0XXX
        jclr #2,omr,EPRSCILD    ; If MD:MC:MB:MA=10xx, load from EPROM/SCI
        jclr #1,omr,OMR1IS0     ; IF MD:MC:MB:MA=110x, look for ISA/HC11
        jclr #0,omr,I8051HOSTLD ; If MD:MC:MB:MA=1110, load from 8051 Host
                                ; If MD:MC:MB:MA=1111, load from MC68302 Host

;=====================================================================
; This is the routine which loads a program through the HI08 host port
; The program is downloaded from the host MCU with the following rules:
; 1) 3 bytes - Define the program length.
; 2) 3 bytes - Define the address to which to start loading the program to.
; 3) 3n bytes (while n is any integer number)
; The program words will be stroed in contiguous PRAM memory locations starting
; at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The host MCU may terminate the loading process by setting the HF1=0 and HF0=1.
; When the downloading is terminated, the program will start execution of the
; loaded program from the specified starting address.
; The HI08 boot ROM program enables the following busses to download programs
; through the HI08 port:
;
```

```
;  1 - ISA          - Dual strobes non-multiplexed bus with negative strobe
;                   pulses duale positive request
;  2 - HC11         - Single strobe non-multiplexed bus with positive strobe
;                   pulse single negative request.
;  4 - i8051        - Dual strobes multiplexed bus with negative strobe pulses
;                   dual negative request.
;  5 - MC68302      - Single strobe non-multiplexed bus with negative strobe
;                   pulse single negative request.
;=====================================================================

MC68302HOSTLD
        movep   #%0000000000111000,x:M_HPCR
                                ; Configure the following conditions:
                                ; HAP   = 0 Negative host acknowledge
                                ; HRP   = 0 Negative host request
                                ; HCSP  = 0 Negatice chip select input
                                ; HD/HS = 0 Single strobe bus (R/W~ and DS)
                                ; HMUX  = 0 Non multiplexed bus
                                ; HASP  = 0 (address strobe polarity has no
                                ;           meaning in non-multiplexed bus)
                                ; HDSP  = 0 Negative data stobes polarity
                                ; HROD  = 0 Host request is active when enabled
                                ; spare = 0 This bit should be set to 0 for
                                ;           future compatability
                                ; HEN   = 0 When the HPCR register is modified
                                ;           HEN should be cleared
                                ; HAEN  = 1 Host acknowledge is enabled
                                ; HREN  = 1 Host requests are enabled
                                ; HCSEN = 1 Host chip select input enabled
                                ; HA9EN = 0 (address 9 enable bit has no
                                ;           meaning in non-multiplexed bus)
                                ; HA8EN = 0 (address 8 enable bit has no
                                ;           meaning in non-multiplexed bus)
                                ; HGEN  = 0 Host GPIO pins are disabled
        bra     <HI08CONT
OMR1IS0
        jset #0,omr,HC11HOSTLD  ; If MD:MC:MB:MA=1101, go load from HC11 Host
                                ; If MD:MC:MB:MA=1100, go load from ISA HOST

ISAHOSTLD
        movep   #%0101000000011000,x:M_HPCR
                                ; Configure the following conditions:
                                ; HAP   = 0 Negative host acknowledge
                                ; HRP   = 1 Positive host request
                                ; HCSP  = 0 Negatice chip select input
                                ; HD/HS = 1 Dual strobes bus (RD and WR)
                                ; HMUX  = 0 Non multiplexed bus
                                ; HASP  = 0 (address strobe polarity has no
                                ;           meaning in non-multiplexed bus)
                                ; HDSP  = 0 Negative data stobes polarity
                                ; HROD  = 0 Host request is active when enabled
                                ; spare = 0 This bit should be set to 0 for
                                ;           future compatability
```

```
                                ; HEN   = 0 When the HPCR register is modified
                                ;           HEN should be cleared
                                ; HAEN  = 0 Host acknowledge is disabled
                                ; HREN  = 1 Host requests are enabled
                                ; HCSEN = 1 Host chip select input enabled
                                ; HA9EN = 0 (address 9 enable bit has no
                                ;           meaning in non-multiplexed bus)
                                ; HA8EN = 0 (address 8 enable bit has no
                                ;           meaning non-multiplexed bus)
                                ; HGEN  = 0 Host GPIO pins are disabled
        bra     <HI08CONT
HC11HOSTLD
        movep   #%0000001000011000,x:M_HPCR
                                ; Configure the following conditions:
                                ; HAP   = 0 Negative host acknowledge
                                ; HRP   = 0 Negative host request
                                ; HCSP  = 0 Negatice chip select input
                                ; HD/HS = 0 Single strobe bus (R/W~ and DS)
                                ; HMUX  = 0 Non multiplexed bus
                                ; HASP  = 0 (address strobe polarity has no
                                ;           meaning in non-multiplexed bus)
                                ; HDSP  = 1 Negative data stobes polarity
                                ; HROD  = 0 Host request is active when enabled
                                ; spare = 0 This bit should be set to 0 for
                                ;           future compatability
                                ; HEN   = 0 When the HPCR register is modified
                                ;           HEN should be cleared
                                ; HAEN  = 0 Host acknowledge is disabled
                                ; HREN  = 1 Host requests are enabled
                                ; HCSEN = 1 Host chip select input enabled
                                ; HA9EN = 0 (address 9 enable bit has no
                                ;           meaning in non-multiplexed bus)
                                ; HA8EN = 0 (address 8 enable bit has no
                                ;           meaning in non-multiplexed bus)
                                ; HGEN  = 0 Host GPIO pins are disabled
        bra     <HI08CONT
I8051HOSTLD
        movep   #%0001110000011110,x:M_HPCR
                                ; Configure the following conditions:
                                ; HAP   = 0 Negative host acknowledge
                                ; HRP   = 0 Negatice host request
                                ; HCSP  = 0 Negatice chip select input
                                ; HD/HS = 1 Dual strobes bus (RD and WR)
                                ; HMUX  = 1 Multiplexed bus
                                ; HASP  = 1 Positive address strobe polarity
                                ; HDSP  = 0 Negative data stobes polarity
                                ; HROD  = 0 Host request is active when enabled
                                ; spare = 0 This bit should be set to 0 for
                                ;           future compatability
                                ; HEN   = 0 When the HPCR register is modified
                                ;           HEN should be cleared
                                ; HAEN  = 0 Host acknowledge is disabled
                                ; HREN  = 1 Host requests are enabled
```

```
                                ; HCSEN = 1 Host chip select input enabled
                                ; HA9EN = 1 Enable address 9 input
                                ; HA8EN = 1 Enable address 8 input
                                ; HGEN  = 0 Host GPIO pins are disabled

HI08CONT
        bset    #HEN,x:M_HPCR           ; Enable the HI08 to operate as host
                                        ; interface (set HEN=1)
        jclr    #HRDF,x:M_HSR,*         ; wait for the program length to be
                                        ; written
        movep   x:M_HRX,a0
        jclr    #HRDF,x:M_HSR,*         ; wait for the program starting address
                                        ; to be written
        movep   x:M_HRX,r0
        move    r0,r1
        do      a0,HI08LOOP             ; set a loop with the downloaded length
HI08LL
        jset    #HRDF,x:M_HSR,HI08NW    ; If new word was loaded then jump to
                                        ; read that word
        jclr    #HF0,x:M_HSR,HI08LL     ; If HF0=0 then continue with the
                                        ; downloading
        enddo                           ; Must terminate the do loop
        bra     <HI08LOOP
HI08NW
        movep   x:M_HRX,p:(r0)+         ; Move the new word into its destination
                                        ; location in the program RAM
        nop                             ; pipeline delay
HI08LOOP
        bra     <FINISH
;=========================================================================
EPRSCILD
        jclr #1,omr,EPROMLD     ; If MD:MC:MB:MA=1001, go load from EPROM
;       jset #0,omr,SCILD       ; If MD:MC:MB:MA=1011, reserved, default to SCI

;=========================================================================
; This is the routine that loads from the SCI.
; MD:MC:MB:MA=1010 - external SCI clock

SCILD
        movep #$0302,X:M_SCR    ; Configure SCI Control Reg
        movep #$C000,X:M_SCCR   ; Configure SCI Clock Control Reg
        movep #7,X:M_PCRE       ; Configure SCLK, TXD and RXD

        do #6,_LOOP6            ; get 3 bytes for number of
                                ; program words and 3 bytes
                                ; for the starting address
        jclr #2,X:M_SSR,*       ; Wait for RDRF to go high
        movep X:M_SRXL,A2       ; Put 8 bits in A2
        jclr #1,X:M_SSR,*       ; Wait for TDRE to go high
        movep A2,X:M_STXL       ; echo the received byte
        asr #8,a,a
_LOOP6
        move a1,r0              ; starting address for load
```

```
        move a1,r1              ; save starting address

        do a0,_LOOP7            ; Receive program words
        do #3,_LOOP8
        jclr #2,X:M_SSR,*       ; Wait for RDRF to go high
        movep X:M_SRXL,A2       ; Put 8 bits in A2
        jclr #1,X:M_SSR,*       ; Wait for TDRE to go high
        movep a2,X:M_STXL       ; echo the received byte
        asr #8,a,a
_LOOP8
        movem a1,p:(r0)+        ; Store 24-bit result in P mem.
        nop                     ; pipeline delay
_LOOP7
        bra <FINISH             ; Boot from SCI done

;=====================================================================
; This is the routine that loads from external EPROM.
; MD:MC:MB:MA=1001

EPROMLD
        move #BOOT,r2           ; r2 = address of external EPROM
        movep #AARV,X:M_AAR1    ; aar1 configured for SRAM types of access


        do #6,_LOOP9            ; read number of words and starting address
        movem p:(r2)+,a2        ; Get the 8 LSB from ext. P mem.
        asr #8,a,a              ; Shift 8 bit data into A1
_LOOP9                          ;
        move a1,r0              ; starting address for load
        move a1,r1              ; save it in r1
                                ; a0 holds the number of words

        do a0,_LOOP10           ; read program words
        do #3,_LOOP11           ; Each instruction has 3 bytes
        movem p:(r2)+,a2        ; Get the 8 LSB from ext. P mem.
        asr #8,a,a              ; Shift 8 bit data into A1
_LOOP11                         ; Go get another byte.
        movem a1,p:(r0)+        ; Store 24-bit result in P mem.
        nop                     ; pipeline delay
_LOOP10                         ; and go get another 24-bit word.
                                ; Boot from EPROM done
;=====================================================================
FINISH

; This is the exit handler that returns execution to normal
; expanded mode and jumps to the RESET vector.

        andi #$0,ccr            ; Clear CCR as if RESET to 0.
        jmp (r1)                ; Then go to starting Prog addr.


;=====================================================================
; The following modes are reserved, some of which are used for internal testing
```

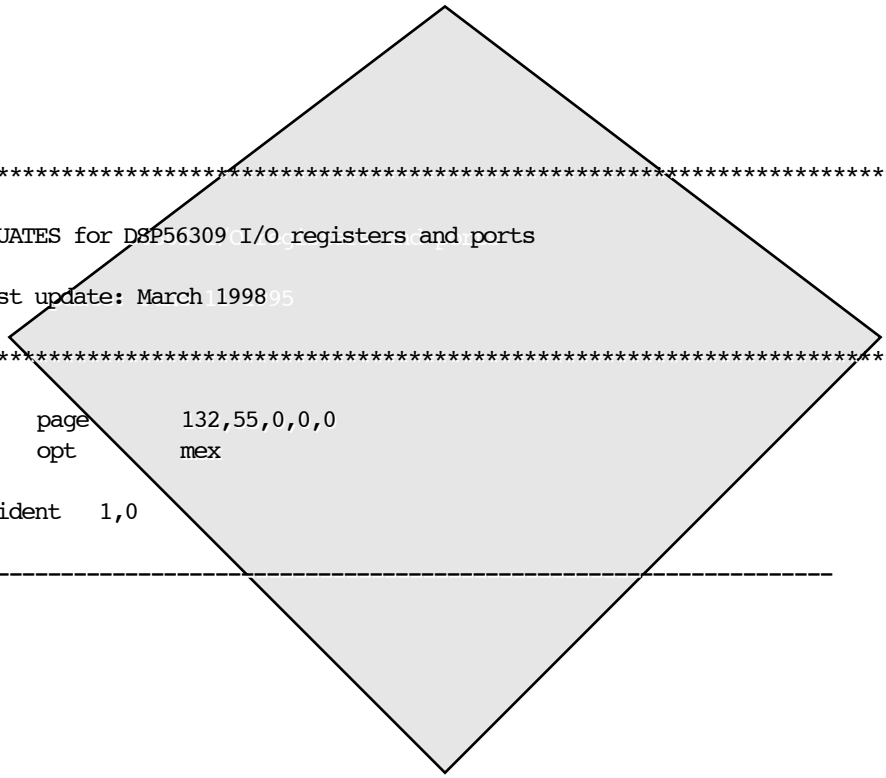
```
; Can be implemented in future.
;
OMR0XXX
        jclr #2,omr,EPRSCILD    ; If MD:MC:MB:MA=00xx, default to EPROM/SCI
        jclr #1,omr,OMR1IS0     ; IF MD:MC:MB:MA=010x, default to ISA/HC11
        jclr #0,omr,I8051HOSTLD ; If MD:MC:MB:MA=0110, default to 8051 Host
                                ; If MD:MC:MB:MA=0111, execute burnin test

;=====================================================================
; This mode is reserved for internal testing purposes
; MD:MC:MB:MA=0111

;;-------------------------------------------------------
;;
;;  burnin mode
;;
;;  Intended to be used for burn-in test. Wake up from reset
;;  with PINIT set for execution in maximum frequency.
;;  All RAM locations are validated, arithmetic/logic operations
;;  are validated (add, eor) and exercised (mac).
;;  While all tests pass, the SCK0 pin will continue to toggle.
;;  When the test fails the DSP enters DEBUG and stops execution.
;;
;;-------------------------------------------------------

                ;; get PATTERN pointer
                clr b   #PATTERNS,r6            ;; b is the error accumulator
                move    #<(NUM_PATTERNS-1),m6   ;; program runs forever in
                                                ;; cyclic form

                ;; configure SCK0 as gpio output. PRRC register is cleared at reset.
                movep   b,x:M_PDRC              ;; clear GPIO data register
                bset    #SCK0,x:M_PRRC          ;; Define SCK0 as output GPIO pin
                                                ;; SCK0 toggles means test pass

                do #9,burn1
                ;;---------------------------
                ;; test RAM
                ;;  each pass checks 1 pattern
                ;;---------------------------
                move    p:(r6)+,x1              ;; pattern for x memory
                move    p:(r6)+,x0              ;; pattern for y memory
                move    p:(r6)+,y0              ;; pattern for p memory

                ;; write pattern to all memory locations

        if      (EQUALDATA)                     ;; x/y ram symmetrical
                ;; write x and y memory
                clr a   #start_dram,r0          ;; start of x/y ram
                move    #>length_dram,n0        ;; length of x/y ram
                rep     n0
                mac x0,x1,a x,l:(r0)+           ;; exercise mac, write x/y ram
```

```
        else                                    ;; x/y ram not symmetrical

                ;; write x memory
                clr a   #start_xram,r0          ;; start of xram
                move    #>length_xram,n0        ;; length of xram
                rep     n0
                mac x0,y0,a x1,x:(r0)+          ;; exercise mac, write xram

                ;; write y memory
                clr a   #start_yram,r1          ;; start of yram
                move    #>length_yram,n1        ;; length of yram
                rep     n1
                mac x1,y0,a x0,y:(r1)+          ;; exercise mac, write yram

        endif

                ;; write p memory
                clr a   #start_pram,r2          ;; start of pram
                move    #>length_pram,n2        ;; length of pram
                rep     n2
                move    y0,p:(r2)+              ;; write pram

                ;; check memory contents

        if      (EQUALDATA)                     ;; x/y ram symmetrical

                ;; check dram
                clr a   #start_dram,r0          ;; restore pointer, clear a
                do      n0,_loopd
                move    x:(r0),a1               ;; a0=a2=0
                eor     x1,a
                add     a,b                     ;; accumulate error in b
                move    y:(r0)+,a1              ;; a0=a2=0
                eor     x0,a
                add     a,b                     ;; accumulate error in b
_loopd

        else                                    ;; x/y ram not symmetrical

                ;; check xram
                clr a   #start_xram,r0          ;; restore pointer, clear a
                do      n0,_loopx
                move    x:(r0)+,a1              ;; a0=a2=0
                eor     x1,a
                add     a,b                     ;; accumulate error in b
_loopx

                ;; check yram
                clr a   #start_yram,r1          ;; restore pointer, clear a
                do      n1,_loopy
                move    y:(r1)+,a1              ;; a0=a2=0
                eor     x0,a
                add     a,b                     ;; accumulate error in b
```

```
_loopy

        endif

                ;; check pram
                clr a   #start_pram,r2          ;; restore pointer, clear a
                do      n2,_loopp
                move    p:(r2)+,a1              ;; a0=a2=0
                eor     y0,a
                add     a,b                     ;; accumulate error in b
_loopp

                ;;-------------------------------------------------
                ;; toggle pin if no errors, stop execution otherwise.
                ;;-------------------------------------------------
                beq     label1
                bclr    #SCK0,x:M_PDRC          ;; clear sck0 if error,
                enddo                           ;; terminate the loop normally
                bra     <burn1                  ;; and stop execution
label1                                          ;; if no error
                bchg    #SCK0,x:M_PDRC          ;; toggle pin and keep on looping

burn1
                wait                            ;; enter wait after test completion

                ORG PL:,PL:
PATTERNS        dsm     4                       ;; align for correct modulo addressing
                ORG     PL:PATTERNS,PL:PATTERNS ;; Each value is written to all memories

                dc      $555555
                dc      $AAAAAA
                dc      $333333
                dc      $F0F0F0

NUM_PATTERNS    equ     *-PATTERNS

                end
```

APPENDIX B

EQUATES



```
;*****************************************************************************
;
;     EQUATES for DSP56309 I/O registers and ports
;
;     Last update: March 1998
;
;*****************************************************************************

            page        132,55,0,0,0
            opt         mex

ioequ   ident   1,0

;-----------------------------------------------------------------------
```

B.1 I/O EQUATES

```
;------------------------------------------------------------------------
;
;I/O Port Programming
;
;------------------------------------------------------------------------
;       Register Addresses

M_HDR    EQU     $FFFFC9         ; Host port GPIO data Register
M_HDDR   EQU     $FFFFC8         ; Host port GPIO direction Register
M_PCRC   EQU     $FFFFBF         ; Port C Control Register
M_PRRC   EQU     $FFFFBE         ; Port C Direction Register
M_PDRC   EQU     $FFFFBD         ; Port C GPIO Data Register
M_PCRD   EQU     $FFFFAF         ; Port D Control register
M_PRRD   EQU     $FFFFAE         ; Port D Direction Data Register
M_PDRD   EQU     $FFFFAD         ; Port D GPIO Data Register
M_PCRE   EQU     $FFFF9F         ; Port E Control register
M_PRRE   EQU     $FFFF9E         ; Port E Direction Register
M_PDRE   EQU     $FFFF9D         ; Port E Data Register
M_OGDB   EQU     $FFFFFC         ; OnCE GDB Register
```

B.2 HOST INTERFACE (HI08) EQUATES

```
;------------------------------------------------------------------------
Host Interface (HI08) Equates
;------------------------------------------------------------------------
;       Register Addresses

M_HCR    EQU     $FFFFC2          ; Host Control Register
M_HSR    EQU     $FFFFC3          ; Host Status Register
M_HPCR   EQU     $FFFFC4          ; Host Polarity Control Register
M_HBAR   EQU     $FFFFC5          ; Host Base Address Register
M_HRX    EQU     $FFFFC6          ; Host Receive Register
M_HTX    EQU     $FFFFC7          ; Host Transmit Register

;       HCR bits definition

M_HRIE   EQU    $0              ; Host Receive interrupts Enable
M_HTIE   EQU    $1              ; Host Transmit Interrupt Enable
M_HCIE   EQU    $2              ; Host Command Interrupt Enable
M_HF2    EQU    $3              ; Host Flag 2
M_HF3    EQU    $4              ; Host Flag 3

;       HSR bits definition

M_HRDF   EQU    $0              ; Host Receive Data Full
M_HTDE   EQU    $1              ; Host Receive Data Empty
M_HCP    EQU    $2              ; Host Command Pending
```

```
M_HF0    EQU    $3              ; Host Flag 0
M_HF1    EQU    $4              ; Host Flag 1

;       HPCR bits definition

M_HGEN   EQU    $0              ; Host Port GPIO Enable
M_HA8EN  EQU    $1              ; Host Address 8 Enable
M_HA9EN  EQU    $2              ; Host Address 9 Enable
M_HCSEN  EQU    $3              ; Host Chip Select Enable
M_HREN   EQU    $4              ; Host Request Enable
M_HAEN   EQU    $5              ; Host Acknowledge Enable
M_HEN    EQU    $6              ; Host Enable
M_HOD    EQU    $8              ; Host Request Open Drain mode
M_HDSP   EQU    $9              ; Host Data Strobe Polarity
M_HASP   EQU    $A              ; Host Address Strobe Polarity
M_HMUX   EQU    $B              ; Host Multiplexed bus select
M_HD_HS  EQU    $C              ; Host Double/Single Strobe select
M_HCSP   EQU    $D              ; Host Chip Select Polarity
M_HRP    EQU    $E              ; Host Request Polarity
M_HAP    EQU    $F              ; Host Acknowledge Polarity
```

B.3 SCI EQUATES

```
;------------------------------------------------------------------------

;Serial Communications Interface (SCI) Equates

;------------------------------------------------------------------------
;       Register Addresses

M_STXH   EQU     $FFFF97         ; SCI Transmit Data Register (high)
M_STXM   EQU     $FFFF96         ; SCI Transmit Data Register (middle)
M_STXL   EQU     $FFFF95         ; SCI Transmit Data Register (low)
M_SRXH   EQU     $FFFF9A         ; SCI Receive Data Register (high)
M_SRXM   EQU     $FFFF99         ; SCI Receive Data Register (middle)
M_SRXL   EQU     $FFFF98         ; SCI Receive Data Register (low)
M_STXA   EQU     $FFFF94         ; SCI Transmit Address Register
M_SCR    EQU     $FFFF9C         ; SCI Control Register
M_SSR    EQU     $FFFF93         ; SCI Status Register
M_SCCR   EQU     $FFFF9B         ; SCI Clock Control Register

;       SCI Control Register Bit Flags

M_WDS    EQU     $7              ; Word Select Mask (WDS0-WDS3)
M_WDS0   EQU     0               ; Word Select 0
M_WDS1   EQU     1               ; Word Select 1
M_WDS2   EQU     2               ; Word Select 2
M_SSFTD  EQU     3               ; SCI Shift Direction
M_SBK    EQU     4               ; Send Break
M_WAKE   EQU     5               ; Wakeup Mode Select
```

```
M_RWU    EQU     6               ; Receiver Wakeup Enable
M_WOMS   EQU     7               ; Wired-OR Mode Select
M_SCRE   EQU     8               ; SCI Receiver Enable
M_SCTE   EQU     9               ; SCI Transmitter Enable
M_ILIE   EQU     10              ; Idle Line Interrupt Enable
M_SCRIE  EQU     11              ; SCI Receive Interrupt Enable
M_SCTIE  EQU     12              ; SCI Transmit Interrupt Enable
M_TMIE   EQU     13              ; Timer Interrupt Enable
M_TIR    EQU     14              ; Timer Interrupt Rate
M_SCKP   EQU     15              ; SCI Clock Polarity
M_REIE   EQU     16              ; SCI Error Interrupt Enable (REIE)

;       SCI Status Register Bit Flags

M_TRNE   EQU     0               ; Transmitter Empty
M_TDRE   EQU     1               ; Transmit Data Register Empty
M_RDRF   EQU     2               ; Receive Data Register Full
M_IDLE   EQU     3               ; Idle Line Flag
M_OR     EQU     4               ; Overrun Error Flag
M_PE     EQU     5               ; Parity Error
M_FE     EQU     6               ; Framing Error Flag
M_R8     EQU     7               ; Received Bit 8 (R8) Address

;       SCI Clock Control Register

M_CD     EQU     $FFF            ; Clock Divider Mask (CD0-CD11)
M_COD    EQU     12              ; Clock Out Divider
M_SCP    EQU     13              ; Clock Prescaler
M_RCM    EQU     14              ; Receive Clock Mode Source Bit
M_TCM    EQU     15              ; Transmit Clock Source Bit
```

B.4 ESSI EQUATES

```
;------------------------------------------------------------------------

;Enhanced Synchronous Serial Interface (ESSI) Equates

;------------------------------------------------------------------------
;       Register Addresses Of SSI0

M_TX00   EQU     $FFFFBC         ; SSI0 Transmit Data Register 0
M_TX01   EQU     $FFFFBB         ; SSIO Transmit Data Register 1
M_TX02   EQU     $FFFFBA         ; SSIO Transmit Data Register 2
M_TSR0   EQU     $FFFFB9         ; SSI0 Time Slot Register
M_RX0    EQU     $FFFFB8         ; SSI0 Receive Data Register
M_SSISR0 EQU     $FFFFB7         ; SSI0 Status Register
M_CRB0   EQU     $FFFFB6         ; SSI0 Control Register B
M_CRA0   EQU     $FFFFB5         ; SSI0 Control Register A
M_TSMA0  EQU     $FFFFB4         ; SSI0 Transmit Slot Mask Register A
M_TSMB0  EQU     $FFFFB3         ; SSI0 Transmit Slot Mask Register B
```

```
M_RSMA0  EQU     $FFFFB2         ; SSI0 Receive Slot Mask Register A
M_RSMB0  EQU     $FFFFB1         ; SSI0 Receive Slot Mask Register B

;       Register Addresses Of SSI1

M_TX10   EQU     $FFFFAC         ; SSI1 Transmit Data Register 0
M_TX11   EQU     $FFFFAB         ; SSI1 Transmit Data Register 1
M_TX12   EQU     $FFFFAA         ; SSI1 Transmit Data Register 2
M_TSR1   EQU     $FFFFA9         ; SSI1 Time Slot Register
M_RX1    EQU     $FFFFA8         ; SSI1 Receive Data Register
M_SSISR1 EQU     $FFFFA7         ; SSI1 Status Register
M_CRB1   EQU     $FFFFA6         ; SSI1 Control Register B
M_CRA1   EQU     $FFFFA5         ; SSI1 Control Register A
M_TSMA1  EQU     $FFFFA4         ; SSI1 Transmit Slot Mask Register A
M_TSMB1  EQU     $FFFFA3         ; SSI1 Transmit Slot Mask Register B
M_RSMA1  EQU     $FFFFA2         ; SSI1 Receive Slot Mask Register A
M_RSMB1  EQU     $FFFFA1         ; SSI1 Receive Slot Mask Register B

;       SSI Control Register A Bit Flags

M_PM     EQU     $FF             ; Prescale Modulus Select Mask (PM0-PM7)
M_PSR    EQU     11              ; Prescaler Range
M_DC     EQU     $1F000          ; Frame Rate Divider Control Mask (DC0-DC7)
M_ALC    EQU     18              ; Alignment Control (ALC)
M_WL     EQU     $380000         ; Word Length Control Mask (WL0-WL7)
M_SSC1   EQU     22              ; Select SC1 as TR #0 drive enable (SSC1)

;       SSI Control Register B Bit Flags

M_OF     EQU     $3              ; Serial Output Flag Mask
M_OF0    EQU     0               ; Serial Output Flag 0
M_OF1    EQU     1               ; Serial Output Flag 1
M_SCD    EQU     $1C             ; Serial Control Direction Mask
M_SCD0   EQU     2               ; Serial Control 0 Direction
M_SCD1   EQU     3               ; Serial Control 1 Direction
M_SCD2   EQU     4               ; Serial Control 2 Direction
M_SCKD   EQU     5               ; Clock Source Direction
M_SHFD   EQU     6               ; Shift Direction
M_FSL    EQU     $180            ; Frame Sync Length Mask (FSL0-FSL1)
M_FSL0   EQU     7               ; Frame Sync Length 0
M_FSL1   EQU     8               ; Frame Sync Length 1
M_FSR    EQU     9               ; Frame Sync Relative Timing
M_FSP    EQU     10              ; Frame Sync Polarity
M_CKP    EQU     11              ; Clock Polarity
M_SYN    EQU     12              ; Sync/Async Control
M_MOD    EQU     13              ; SSI Mode Select
M_SSTE   EQU     $1C000          ; SSI Transmit enable Mask
M_SSTE2  EQU     14              ; SSI Transmit #2 Enable
M_SSTE1  EQU     15              ; SSI Transmit #1 Enable
M_SSTE0  EQU     16              ; SSI Transmit #0 Enable
M_SSRE   EQU     17              ; SSI Receive Enable
M_SSTIE  EQU     18              ; SSI Transmit Interrupt Enable
M_SSRIE  EQU     19              ; SSI Receive Interrupt Enable
```

```
M_STLIE  EQU     20              ; SSI Transmit Last Slot Interrupt Enable
M_SRLIE  EQU     21              ; SSI Receive Last Slot Interrupt Enable
M_STEIE  EQU     22              ; SSI Transmit Error Interrupt Enable
M_SREIE  EQU     23              ; SSI Receive Error Interrupt Enable

;       SSI Status Register Bit Flags

M_IF     EQU     $3              ; Serial Input Flag Mask
M_IF0    EQU     0               ; Serial Input Flag 0
M_IF1    EQU     1               ; Serial Input Flag 1
M_TFS    EQU     2               ; Transmit Frame Sync Flag
M_RFS    EQU     3               ; Receive Frame Sync Flag
M_TUE    EQU     4               ; Transmitter Underrun Error FLag
M_ROE    EQU     5               ; Receiver Overrun Error Flag
M_TDE    EQU     6               ; Transmit Data Register Empty
M_RDF    EQU     7               ; Receive Data Register Full

;       SSI Transmit Slot Mask Register A

M_SSTSA  EQU     $FFFF           ; SSI Transmit Slot Bits Mask A (TS0-TS15)

;       SSI Transmit Slot Mask Register B

M_SSTSB  EQU     $FFFF           ; SSI Transmit Slot Bits Mask B (TS16-TS31)

;       SSI Receive Slot Mask Register A

M_SSRSA  EQU     $FFFF           ; SSI Receive Slot Bits Mask A (RS0-RS15)

;       SSI Receive Slot Mask Register B

M_SSRSB  EQU     $FFFF           ; SSI Receive Slot Bits Mask B (RS16-RS31)
```

B.5 EXCEPTION PROCESSING EQUATES

```
;---------------------------------------------------------------------
Exception Processing Equates
;---------------------------------------------------------------------
;       Register Addresses

M_IPRC   EQU     $FFFFFF         ; Interrupt Priority Register Core
M_IPRP   EQU     $FFFFFE         ; Interrupt Priority Register Peripheral

;       Interrupt Priority Register Core (IPRC)

M_IAL    EQU     $7              ; IRQA Mode Mask
M_IAL0   EQU     0               ; IRQA Mode Interrupt Priority Level (low)
M_IAL1   EQU     1               ; IRQA Mode Interrupt Priority Level (high)
M_IAL2   EQU     2               ; IRQA Mode Trigger Mode
M_IBL    EQU     $38             ; IRQB Mode Mask
```

```
M_IBL0   EQU     3               ; IRQB Mode Interrupt Priority Level (low)
M_IBL1   EQU     4               ; IRQB Mode Interrupt Priority Level (high)
M_IBL2   EQU     5               ; IRQB Mode Trigger Mode
M_ICL    EQU     $1C0            ; IRQC Mode Mask
M_ICL0   EQU     6               ; IRQC Mode Interrupt Priority Level (low)
M_ICL1   EQU     7               ; IRQC Mode Interrupt Priority Level (high)
M_ICL2   EQU     8               ; IRQC Mode Trigger Mode
M_IDL    EQU     $E00            ; IRQD Mode Mask
M_IDL0   EQU     9               ; IRQD Mode Interrupt Priority Level
;(low)
M_IDL1   EQU     10              ; IRQD Mode Interrupt Priority Level
;(high)
M_IDL2   EQU     11              ; IRQD Mode Trigger Mode
M_D0L    EQU     $3000           ; DMA0 Interrupt priority Level Mask
M_D0L0   EQU     12              ; DMA0 Interrupt Priority Level (low)
M_D0L1   EQU     13              ; DMA0 Interrupt Priority Level (high)
M_D1L    EQU     $C000           ; DMA1 Interrupt Priority Level Mask
M_D1L0   EQU     14              ; DMA1 Interrupt Priority Level (low)
M_D1L1   EQU     15              ; DMA1 Interrupt Priority Level (high)
M_D2L    EQU     $30000          ; DMA2 Interrupt priority Level Mask
M_D2L0   EQU     16              ; DMA2 Interrupt Priority Level (low)
M_D2L1   EQU     17              ; DMA2 Interrupt Priority Level (high)
M_D3L    EQU     $C0000          ; DMA3 Interrupt Priority Level Mask
M_D3L0   EQU     18              ; DMA3 Interrupt Priority Level (low)
M_D3L1   EQU     19              ; DMA3 Interrupt Priority Level (high)
M_D4L    EQU     $300000         ; DMA4 Interrupt priority Level Mask
M_D4L0   EQU     20              ; DMA4 Interrupt Priority Level (low)
M_D4L1   EQU     21              ; DMA4 Interrupt Priority Level (high)
M_D5L    EQU     $C00000         ; DMA5 Interrupt priority Level Mask
M_D5L0   EQU     22              ; DMA5 Interrupt Priority Level (low)
M_D5L1   EQU     23              ; DMA5 Interrupt Priority Level (high)

;       Interrupt Priority Register Peripheral (IPRP)

M_HPL    EQU     $3              ; Host Interrupt Priority Level Mask
M_HPL0   EQU     0               ; Host Interrupt Priority Level (low)
M_HPL1   EQU     1               ; Host Interrupt Priority Level (high)
M_S0L    EQU     $C              ; SSI0 Interrupt Priority Level Mask
M_S0L0   EQU     2               ; SSI0 Interrupt Priority Level (low)
M_S0L1   EQU     3               ; SSI0 Interrupt Priority Level (high)
M_S1L    EQU     $30             ; SSI1 Interrupt Priority Level Mask
M_S1L0   EQU     4               ; SSI1 Interrupt Priority Level (low)
M_S1L1   EQU     5               ; SSI1 Interrupt Priority Level (high)
M_SCL    EQU     $C0             ; SCI  Interrupt Priority Level  Mask
M_SCL0   EQU     6               ; SCI  Interrupt Priority Level  (low)
M_SCL1   EQU     7               ; SCI  Interrupt Priority Level  (high)
M_T0L    EQU     $300            ; TIMER Interrupt Priority Level Mask
M_T0L0   EQU     8               ; TIMER Interrupt Priority Level (low)
M_T0L1   EQU     9               ; TIMER Interrupt Priority Level (high)
```

B.6 TIMER MODULE EQUATES

```
;---------------------------------------------------------------

;Timer Module Equates

;---------------------------------------------------------------
;       Register Addresses Of TIMER0

M_TCSR0  EQU     $FFFF8F         ; TIMER0 Control/Status Register
M_TLR0   EQU     $FFFF8E         ; TIMER0 Load Register
M_TCPR0  EQU     $FFFF8D         ; TIMER0 Compare Register
M_TCR0   EQU     $FFFF8C         ; TIMER0 Count Register

;       Register Addresses Of TIMER1

M_TCSR1  EQU     $FFFF8B         ; TIMER1 Control/Status Register
M_TLR1   EQU     $FFFF8A         ; TIMER1 Load Register
M_TCPR1  EQU     $FFFF89         ; TIMER1 Compare Register
M_TCR1   EQU     $FFFF88         ; TIMER1 Count Register

;       Register Addresses Of TIMER2

M_TCSR2  EQU     $FFFF87         ; TIMER2 Control/Status Register
M_TLR2   EQU     $FFFF86         ; TIMER2 Load Register
M_TCPR2  EQU     $FFFF85         ; TIMER2 Compare Register
M_TCR2   EQU     $FFFF84         ; TIMER2 Count Register
M_TPLR   EQU     $FFFF83         ; TIMER Prescaler Load Register
M_TPCR   EQU     $FFFF82         ; TIMER Prescaler Count Register

;       Timer Control/Status Register Bit Flags

M_TE     EQU     0               ; Timer Enable
M_TOIE   EQU     1               ; Timer Overflow Interrupt Enable
M_TCIE   EQU     2               ; Timer Compare Interrupt Enable
M_TC     EQU     $F0             ; Timer Control Mask TC(3:0)
M_INV    EQU     8               ; Inverter Bit
M_TRM    EQU     9               ; Timer Restart Mode
M_DIR    EQU     11              ; Direction Bit
M_DI     EQU     12              ; Data Input
M_DO     EQU     13              ; Data Output
M_PCE    EQU     15              ; Prescaled Clock Enable
M_TOF    EQU     20              ; Timer Overflow Flag
M_TCF    EQU     21              ; Timer Compare Flag

;       Timer Prescaler Register Bit Flags

M_PS     EQU     $600000        ; Prescaler Source Mask
M_PS0    EQU     21
M_PS1    EQU     22

;        Timer Control Bits
```

```
M_TC0    EQU     4               ; Timer Control 0
M_TC1    EQU     5               ; Timer Control 1
M_TC2    EQU     6               ; Timer Control 2
M_TC3    EQU     7               ; Timer Control 3
```

B.7 DIRECT MEMORY ACCESS (DMA) EQUATES

```
;------------------------------------------------------------------------

;Direct Memory Access (DMA) Equates

;------------------------------------------------------------------------
;       Register Addresses Of DMA

M_DSTR   EQU     $FFFFF4         ; DMA Status Register
M_DOR0   EQU     $FFFFF3         ; DMA Offset Register 0
M_DOR1   EQU     $FFFFF2         ; DMA Offset Register 1
M_DOR2   EQU     $FFFFF1         ; DMA Offset Register 2
M_DOR3   EQU     $FFFFF0         ; DMA Offset Register 3

;       Register Addresses Of DMA0

M_DSR0   EQU     $FFFFEF         ; DMA0 Source Address Register
M_DDR0   EQU     $FFFFEE         ; DMA0 Destination Address Register
M_DCO0   EQU     $FFFFED         ; DMA0 Counter
M_DCR0   EQU     $FFFFEC         ; DMA0 Control Register

;       Register Addresses Of DMA1

M_DSR1   EQU     $FFFFEB         ; DMA1 Source Address Register
M_DDR1   EQU     $FFFFEA         ; DMA1 Destination Address Register
M_DCO1   EQU     $FFFFE9         ; DMA1 Counter
M_DCR1   EQU     $FFFFE8         ; DMA1 Control Register

;       Register Addresses Of DMA2

M_DSR2   EQU     $FFFFE7       ; DMA2 Source Address Register
M_DDR2   EQU     $FFFFE6       ; DMA2 Destination Address Register
M_DCO2   EQU     $FFFFE5       ; DMA2 Counter
M_DCR2   EQU     $FFFFE4       ; DMA2 Control Register

;       Register Addresses Of DMA4

M_DSR3   EQU     $FFFFE3         ; DMA3 Source Address Register
M_DDR3   EQU     $FFFFE2         ; DMA3 Destination Address Register
M_DCO3   EQU     $FFFFE1         ; DMA3 Counter
M_DCR3   EQU     $FFFFE0         ; DMA3 Control Register

;       Register Addresses Of DMA4
```

```
M_DSR4   EQU     $FFFFDF         ; DMA4 Source Address Register
M_DDR4   EQU     $FFFFDE         ; DMA4 Destination Address Register
M_DCO4   EQU     $FFFFDD         ; DMA4 Counter
M_DCR4   EQU     $FFFFDC         ; DMA4 Control Register

;       Register Addresses Of DMA5

M_DSR5   EQU     $FFFFDB         ; DMA5 Source Address Register
M_DDR5   EQU     $FFFFDA         ; DMA5 Destination Address Register
M_DCO5   EQU     $FFFFD9         ; DMA5 Counter
M_DCR5   EQU     $FFFFD8         ; DMA5 Control Register


;       DMA Control Register

M_DSS    EQU     $3              ; DMA Source Space Mask

;(DSS0-Dss1)

M_DSS0   EQU    0                ; DMA Source Memory space 0
M_DSS1   EQU    1                ; DMA Source Memory space 1
M_DDS    EQU    $C               ; DMA Destination Space Mask

;(DDS-DDS1)

M_DDS0   EQU    2                ; DMA Destination Memory Space 0
M_DDS1   EQU    3                ; DMA Destination Memory Space 1
M_DAM    EQU    $3f0             ; DMA Address Mode Mask

;(DAM5-DAM0)

M_DAM0   EQU    4          ; DMA Address Mode 0
M_DAM1   EQU    5          ; DMA Address Mode 1
M_DAM2   EQU    6          ; DMA Address Mode 2
M_DAM3   EQU    7          ; DMA Address Mode 3
M_DAM4   EQU    8          ; DMA Address Mode 4
M_DAM5   EQU    9          ; DMA Address Mode 5
M_D3D    EQU    10         ; DMA Three Dimensional Mode
M_DRS    EQU    $F800      ; DMA Request Source Mask (DRS0-DRS4)
M_DCON   EQU    16         ; DMA Continuous Mode
M_DPR    EQU    $60000     ; DMA Channel Priority
M_DPR0   EQU    17         ; DMA Channel Priority Level (low)
M_DPR1   EQU    18         ; DMA Channel Priority Level (high)
M_DTM    EQU    $380000    ; DMA Transfer Mode Mask

;(DTM2-DTM0)

M_DTM0   EQU    19         ; DMA Transfer Mode 0
M_DTM1   EQU    20         ; DMA Transfer Mode 1
M_DTM2   EQU    21         ; DMA Transfer Mode 2
M_DIE    EQU    22         ; DMA Interrupt Enable bit
M_DE     EQU    23         ; DMA Channel Enable bit
```

```
;       DMA Status Register

M_DTD    EQU    $3F      ;Channel Transfer Done Status MASK
M_DTD0   EQU    0        ; DMA Channel Transfer Done Status 0
M_DTD1   EQU    1        ; DMA Channel Transfer Done Status 1
M_DTD2   EQU    2        ; DMA Channel Transfer Done Status 2
M_DTD3   EQU    3        ; DMA Channel Transfer Done Status 3
M_DTD4   EQU    4        ; DMA Channel Transfer Done Status 4
M_DTD5   EQU    5        ; DMA Channel Transfer Done Status 5
M_DACT   EQU    8        ; DMA Active State
M_DCH    EQU    $E00     ; DMA Active Channel Mask

;(DCH0DCH2)

M_DCH0   EQU    9        ; DMA Active Channel 0
M_DCH1   EQU    10       ; DMA Active Channel 1
M_DCH2   EQU    11       ; DMA Active Channel 2
```

B.8 PHASE-LOCKED LOOP (PLL) EQUATES

```
;-------------------------------------------------------------

;Phase Locked Loop (PLL) equates

;-------------------------------------------------------------
;       Register Addresses Of PLL

M_PCTL   EQU     $FFFFFD         ; PLL Control Register

;       PLL Control Register

M_MF     EQU     $FFF            ; Multiplication Factor Bit Mask (MF0-MF11)
M_DF     EQU     $7000           ; Division Factor Bit Mask (DF0-DF2)
M_XTLR   EQU     15              ; XTAL Range select bit
M_XTLD   EQU     16              ; XTAL Disable Bit
M_PSTP   EQU     17              ; STOP Processing State Bit
M_PEN    EQU     18              ; PLL Enable Bit
M_PCOD   EQU     19              ; PLL Clock Output Disable Bit
M_PD     EQU     $F00000         ; PreDivider Factor Bit Mask (PD0-PD3)
```

B.9 BUS INTERFACE UNIT (BIU) EQUATES

```
;---------------------------------------------------------------

;Bus Interface Unit (BIU) Equates

;---------------------------------------------------------------
;       Register Addresses Of BIU

M_BCR    EQU     $FFFFFB         ; Bus Control Register
M_DCR    EQU     $FFFFFA         ; DRAM Control Register
M_AAR0   EQU     $FFFFF9         ; Address Attribute Register 0
M_AAR1   EQU     $FFFFF8         ; Address Attribute Register 1
M_AAR2   EQU     $FFFFF7         ; Address Attribute Register 2
M_AAR3   EQU     $FFFFF6         ; Address Attribute Register 3
M_IDR    EQU     $FFFFF5         ; ID Register

;       Bus Control Register

M_BA0W   EQU     $1F             ; Area 0 Wait Control Mask (BA0W0-BA0W4)
M_BA1W   EQU     $3E0            ; Area 1 Wait Control Mask (BA1W0-BA14)
M_BA2W   EQU     $1C00           ; Area 2 Wait Control Mask (BA2W0-BA2W2)
M_BA3W   EQU     $E000           ; Area 3 Wait Control Mask (BA3W0-BA3W3)
M_BDFW   EQU     $1F0000         ; Default Area Wait Control Mask (BDFW0-BDFW4)
M_BBS    EQU     21              ; Bus State
M_BLH    EQU     22              ; Bus Lock Hold
M_BRH    EQU     23              ; Bus Request Hold

;       DRAM Control Register

M_BCW    EQU     $3              ; In Page Wait States Bit Mask (BCW0-BCW1)
M_BRW    EQU     $C              ; Out Of Page Wait States Bit Mask (BRW0-BRW1)
M_BPS    EQU     $300            ; DRAM Page Size Bit Mask (BPS0-BPS1)
M_BPLE   EQU     11              ; Page Logic Enable
M_BME    EQU     12              ; Mastership Enable
M_BRE    EQU     13              ; Refresh Enable
M_BSTR   EQU     14              ; Software Triggered Refresh
M_BRF    EQU     $7F8000         ; Refresh Rate Bits Mask (BRF0-BRF7)
M_BRP    EQU     23              ; Refresh prescaler

;       Address Attribute Registers

M_BAT    EQU     $3             ; External Access Type and Pin Definition Bits

;      Mask BAT(1:0)

M_BAAP   EQU     2               ; Address Attribute Pin Polarity
M_BPEN   EQU     3               ; Program Space Enable
M_BXEN   EQU     4               ; X Data Space Enable
M_BYEN   EQU     5               ; Y Data Space Enable
M_BAM    EQU     6               ; Address Muxing
M_BPAC   EQU     7               ; Packing Enable
```

```
M_BNC    EQU     $F00            ; Number of Address Bits to Compare Mask
M_BAC    EQU     $FFF000         ; Address to Compare Bits Mask BAC(11:0)

;       control and status bits in SR

M_CP     EQU     $c00000         ; mask for CORE-DMA priority bits in SR
M_CA     EQU     0               ; Carry
M_V      EQU     1               ; Overflow
M_Z      EQU     2               ; Zero
M_N      EQU     3               ; Negative
M_U      EQU     4               ; Unnormalized
M_E      EQU     5               ; Extension
M_L      EQU     6               ; Limit
M_S      EQU     7               ; Scaling Bit
M_I0     EQU     8               ; Interrupt Mask Bit 0
M_I1     EQU     9               ; Interrupt Mask Bit 1
M_S0     EQU     10              ; Scaling Mode Bit 0
M_S1     EQU     11              ; Scaling Mode Bit 1
M_SC     EQU     13              ; Sixteen_Bit Compatibility
M_DM     EQU     14              ; Double Precision Multiply
M_LF     EQU     15              ; DO-Loop Flag
M_FV     EQU     16              ; DO-Forever Flag
M_SA     EQU     17              ; Sixteen-Bit Arithmetic
M_CE     EQU     19              ; Instruction Cache Enable
M_SM     EQU     20              ; Arithmetic Saturation
M_RM     EQU     21              ; Rounding Mode
M_CP0    EQU     22              ; bit 0 of priority bits in SR
M_CP1    EQU     23              ; bit 1 of priority bits in SR

;       control and status bits in OMR

M_CDP    EQU     $300            ; mask for CORE-DMA priority bits in OMR
M_MA     EQU     0               ; Operating Mode A
M_MB     EQU     1               ; Operating Mode B
M_MC     EQU     2               ; Operating Mode C
M_MD     EQU     3               ; Operating Mode D
M_EBD    EQU     4               ; External Bus Disable bit in OMR
M_SD     EQU     6               ; Stop Delay
M_MS     EQU     7               ; Memory Switch bit in OMR
M_CDP0   EQU     8               ; bit 0 of priority bits in OMR
M_CDP1   EQU     9               ; bit 1 of priority bits in OMR
M_BEN    EQU     10              ; Burst Enable
M_TAS    EQU     11              ; TA Synchronize Select
M_BRT    EQU     12              ; Bus Release Timing
M_ATE    EQU     15              ; Address Tracing Enable bit in OMR.
M_XYS    EQU     16              ; Stack Extension space select bit in OMR.
M_EUN    EQU     17              ; Extended stack UNderflow flag in OMR.
M_EOV    EQU     18              ; Extended stack OVerflow flag in OMR.
M_WRP    EQU     19              ; Extended WRaP flag in OMR.
M_SEN    EQU     20              ; Stack Extension Enable bit in OMR.
```

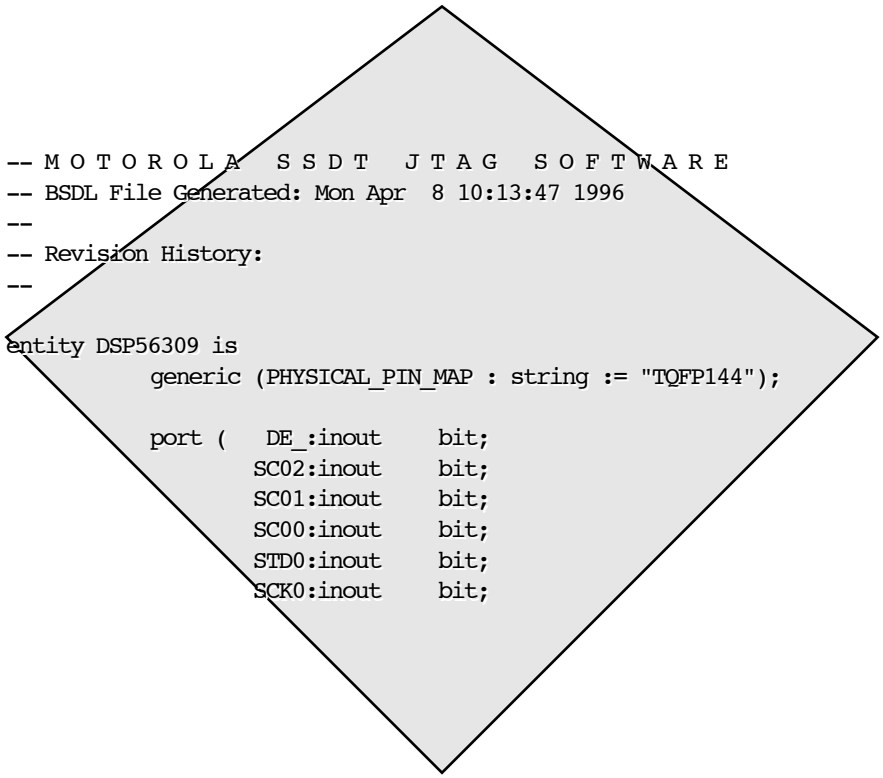
B.10 INTERRUPT EQUATES

```
;*************************************************************
;     EQUATES for DSP56309 interrupts

;*************************************************************

;-------------------------------------------------------------

; Non-Maskable interrupts

;-------------------------------------------------------------

I_RESET  EQU  I_VEC+$00   ; Hardware RESET
I_STACK  EQU  I_VEC+$02   ; Stack Error
I_ILL    EQU  I_VEC+$04   ; Illegal Instruction
I_DBG    EQU  I_VEC+$06   ; Debug Request
I_TRAP   EQU  I_VEC+$08   ; Trap
I_NMI    EQU  I_VEC+$0A   ; Non Maskable Interrupt
;-------------------------------------------------------------

; Interrupt Request Pins

;-------------------------------------------------------------
I_IRQA   EQU  I_VEC+$10   ; IRQA
I_IRQB   EQU  I_VEC+$12   ; IRQB
I_IRQC   EQU  I_VEC+$14   ; IRQC
I_IRQD   EQU  I_VEC+$16   ; IRQD
;-------------------------------------------------------------

; DMA Interrupts

;-------------------------------------------------------------
I_DMA0   EQU  I_VEC+$18   ; DMA Channel 0
I_DMA1   EQU  I_VEC+$1A   ; DMA Channel 1
I_DMA2   EQU  I_VEC+$1C   ; DMA Channel 2
I_DMA3   EQU  I_VEC+$1E   ; DMA Channel 3
I_DMA4   EQU  I_VEC+$20   ; DMA Channel 4
I_DMA5   EQU  I_VEC+$22   ; DMA Channel 5
;-------------------------------------------------------------

; Timer Interrupts

;-------------------------------------------------------------
I_TIM0C  EQU  I_VEC+$24   ; TIMER 0 compare
I_TIM0OF EQU  I_VEC+$26   ; TIMER 0 overflow
I_TIM1C  EQU  I_VEC+$28   ; TIMER 1 compare
I_TIM1OF EQU  I_VEC+$2A   ; TIMER 1 overflow
I_TIM2C  EQU  I_VEC+$2C   ; TIMER 2 compare
I_TIM2OF EQU  I_VEC+$2E   ; TIMER 2 overflow
;-------------------------------------------------------------
```

```
; ESSI Interrupts

;------------------------------------------------------------------
I_SI0RD  EQU  I_VEC+$30   ; ESSI0 Receive Data
I_SI0RDE EQU  I_VEC+$32   ; ESSI0 Receive Data With Exception Status
I_SI0RLS EQU  I_VEC+$34   ; ESSI0 Receive last slot
I_SI0TD  EQU  I_VEC+$36   ; ESSI0 Transmit data
I_SI0TDE EQU  I_VEC+$38   ; ESSI0 Transmit Data With Exception Status
I_SI0TLS EQU  I_VEC+$3A   ; ESSI0 Transmit last slot
I_SI1RD  EQU  I_VEC+$40   ; ESSI1 Receive Data
I_SI1RDE EQU  I_VEC+$42   ; ESSI1 Receive Data With Exception Status
I_SI1RLS EQU  I_VEC+$44   ; ESSI1 Receive last slot
I_SI1TD  EQU  I_VEC+$46   ; ESSI1 Transmit data
I_SI1TDE EQU  I_VEC+$48   ; ESSI1 Transmit Data With Exception Status
I_SI1TLS EQU  I_VEC+$4A   ; ESSI1 Transmit last slot
;------------------------------------------------------------------

; SCI Interrupts

;------------------------------------------------------------------
I_SCIRD  EQU  I_VEC+$50   ; SCI Receive Data
I_SCIRDE EQU  I_VEC+$52   ; SCI Receive Data With Exception Status
I_SCITD  EQU  I_VEC+$54   ; SCI Transmit Data
I_SCIIL  EQU  I_VEC+$56   ; SCI Idle Line
I_SCITM  EQU  I_VEC+$58   ; SCI Timer
;------------------------------------------------------------------

; HOST Interrupts

;------------------------------------------------------------------
I_HRDF   EQU     I_VEC+$60   ; Host Receive Data Full
I_HTDE   EQU     I_VEC+$62   ; Host Transmit Data Empty
I_HC     EQU     I_VEC+$64   ; Default Host Command
;------------------------------------------------------------------

; INTERRUPT ENDING ADDRESS

;------------------------------------------------------------------
I_INTEND EQU  I_VEC+$FF   ; last address of interrupt vector space
```

APPENDIX C

DSP56309 BSDL LISTING



```
-- M O T O R O L A   S S D T   J T A G   S O F T W A R E
-- BSDL File Generated: Mon Apr  8 10:13:47 1996
--
-- Revision History:
--

entity DSP56309 is
        generic (PHYSICAL_PIN_MAP : string := "TQFP144");

        port (   DE_:inout    bit;
                SC02:inout    bit;
                SC01:inout    bit;
                SC00:inout    bit;
                STD0:inout    bit;
                SCK0:inout    bit;
```

```
-- M O T O R O L A   S S D T   J T A G   S O F T W A R E
-- BSDL File Generated: Tue Mar  3 15:14:41 1998
--
-- Revision History:
--

entity DSP56309 is
          generic (PHYSICAL_PIN_MAP : string := "TQFP144");

          port (  DE_N:inout    bit;
                  SC02:inout    bit;
                  SC01:inout    bit;
                  SC00:inout    bit;
                  STD0:inout    bit;
                  SCK0:inout    bit;
                  SRD0:inout    bit;
                  SRD1:inout    bit;
                  SCK1:inout    bit;
                  STD1:inout    bit;
                  SC10:inout    bit;
                  SC11:inout    bit;
                  SC12:inout    bit;
                   TXD:inout    bit;
                  SCLK:inout    bit;
                   RXD:inout    bit;
                  TIO0:inout    bit;
                  TIO1:inout    bit;
                  TIO2:inout    bit;
                   HAD:inout    bit_vector(0 to 7);
                  HREQ:inout    bit;
                  MODD:in       bit;
                  MODC:in       bit;
                  MODB:in       bit;
                  MODA:in       bit;
                     D:inout    bit_vector(0 to 23);
                     A:out      bit_vector(0 to 17);
                 EXTAL:in       bit;
                  XTAL:linkage  bit;
                  RD_N:out      bit;
                  WR_N:out      bit;
                    AA:out      bit_vector(0 to 3);
                  BR_N:buffer   bit;
                  BG_N:in       bit;
                  BB_N:inout    bit;
                  PCAP:linkage  bit;
               RESET_N:in       bit;
                 PINIT:in       bit;
                  TA_N:in       bit;
                CAS_N:out       bit;
                  BCLK:out      bit;
               BCLK_N:out       bit;
               CLKOUT:buffer    bit;
               TRST_N:in        bit;
```

```
        TDO:out      bit;
        TDI:in       bit;
        TCK:in       bit;
        TMS:in       bit;
  RESERVED:linkage   bit_vector(0 to 1);
      SGND:linkage   bit_vector(0 to 1);
      SVCC:linkage   bit_vector(0 to 1);
      QGND:linkage   bit_vector(0 to 3);
      QVCC:linkage   bit_vector(0 to 3);
      HGND:linkage   bit;
      HVCC:linkage   bit;
      DGND:linkage   bit_vector(0 to 3);
      DVCC:linkage   bit_vector(0 to 3);
      AGND:linkage   bit_vector(0 to 3);
      AVCC:linkage   bit_vector(0 to 3);
      JVCC:linkage   bit;
     JGND1:linkage   bit;
      JGND:linkage   bit;
      HACK:inout     bit;
       HDS:inout     bit;
       HRW:inout     bit;
      CVCC:linkage   bit_vector(0 to 1);
      CGND:linkage   bit_vector(0 to 1);
       HCS:inout     bit;
       HA9:inout     bit;
       HA8:inout     bit;
       HAS:inout     bit);

use STD_1149_1_1994.all;

attribute COMPONENT_CONFORMANCE of DSP56309 : entity is "STD_1149_1_1993";

attribute PIN_MAP of DSP56309 : entity is PHYSICAL_PIN_MAP;

constant TQFP144 : PIN_MAP_STRING :=
"SRD1:       1, " &
"STD1:       2, " &
"SC02:       3, " &
"SC01:       4, " &
"DE_N:       5, " &
"PINIT:      6, " &
"SRD0:       7, " &
"SVCC:      (8, 25), " &
"SGND:      (9, 26), " &
"STD0:       10, " &
"SC10:       11, " &
"SC00:       12, " &
"RXD:        13, " &
"TXD:        14, " &
"SCLK:       15, " &
"SCK1:       16, " &
"SCK0:       17, " &
"QVCC:      (18, 56, 91, 126), " &
```

```
"QGND:      (19, 54, 90, 127), " &
"RESERVED:  (49, 20), " &
"HDS:        21, " &
"HRW:        22, " &
"HACK:       23, " &
"HREQ:       24, " &
"TIO2:       27, " &
"TIO1:       28, " &
"TIO0:       29, " &
"HCS:        30, " &
"HA9:        31, " &
"HA8:        32, " &
"HAS:        33, " &
"HAD:       (43, 42, 41, 40, 37, 36, 35, 34), " &
"HVCC:       38, " &
"HGND:       39, " &
"RESET_N:    44, " &
"JVCC:       45, " &
"PCAP:       46, " &
"JGND:       47, " &
"JGND1:      48, " &
"AA:        (70, 69, 51, 50), " &
"CAS_N:      52, " &
"XTAL:       53, " &
"EXTAL:      55, " &
"CVCC:      (57, 65), " &
"CGND:      (58, 66), " &
"CLKOUT:     59, " &
"BCLK:       60, " &
"BCLK_N:     61, " &
"TA_N:       62, " &
"BR_N:       63, " &
"BB_N:       64, " &
"WR_N:       67, " &
"RD_N:       68, " &
"BG_N:       71, " &
"A:         (72, 73, 76, 77, 78, 79, 82, 83, 84, 85, 88, 89, 92, 93, 94, 97, 98, 99), " &
"AVCC:      (74, 80, 86, 95), " &
"AGND:      (75, 81, 87, 96), " &
"D:         (100, 101, 102, 105, 106, 107, 108, 109, 110, 113, 114, 115, 116, 117, " &
                    " 118, 121, 122, 123, 124, 125, 128, 131, 132, 133), " &
"DVCC:      (103, 111, 119, 129), " &
"DGND:      (104, 112, 120, 130), " &
"MODD:       134, " &
"MODC:       135, " &
"MODB:       136, " &
"MODA:       137, " &
"TRST_N:     138, " &
"TDO:        139, " &
"TDI:        140, " &
"TCK:        141, " &
"TMS:        142, " &
"SC12:       143, " &
```

```
"SC11:      144 ";

attribute TAP_SCAN_IN    of     TDI : signal is true;
attribute TAP_SCAN_OUT   of     TDO : signal is true;
attribute TAP_SCAN_MODE  of     TMS : signal is true;
attribute TAP_SCAN_RESET of  TRST_N : signal is true;
attribute TAP_SCAN_CLOCK of     TCK : signal is (20.0e6, BOTH);

attribute INSTRUCTION_LENGTH of DSP56309 : entity is 4;

attribute INSTRUCTION_OPCODE of DSP56309 : entity is
   "EXTEST       (0000)," &
   "SAMPLE       (0001)," &
   "IDCODE       (0010)," &
   "CLAMP        (0101)," &
   "HIGHZ        (0100)," &
   "ENABLE_ONCE  (0110)," &
   "DEBUG_REQUEST(0111)," &
   "BYPASS       (1111)";

attribute INSTRUCTION_CAPTURE of DSP56309 : entity is "0001";
attribute IDCODE_REGISTER   of DSP56309 : entity is
   "0010"          & -- version
   "000110"        & -- manufacturer's use
   "0000000010"    & -- sequence number
   "00000001110"   & -- manufacturer identity
   "1";              -- 1149.1 requirement


attribute REGISTER_ACCESS of DSP56309 : entity is
   "ONCE[8]   (ENABLE_ONCE,DEBUG_REQUEST)" ;

attribute BOUNDARY_LENGTH of DSP56309 : entity is 144;

attribute BOUNDARY_REGISTER of DSP56309 : entity is
-- num    cell   port   func           safe [ccell  dis  rslt]
   "0     (BC_1, MODA,      input,         X)," &
   "1     (BC_1, MODB,      input,         X)," &
   "2     (BC_1, MODC,      input,         X)," &
   "3     (BC_1, MODD,      input,         X)," &
   "4     (BC_6, D(23),     bidir,         X,     13,   1,   Z)," &
   "5     (BC_6, D(22),     bidir,         X,     13,   1,   Z)," &
   "6     (BC_6, D(21),     bidir,         X,     13,   1,   Z)," &
   "7     (BC_6, D(20),     bidir,         X,     13,   1,   Z)," &
   "8     (BC_6, D(19),     bidir,         X,     13,   1,   Z)," &
   "9     (BC_6, D(18),     bidir,         X,     13,   1,   Z)," &
   "10    (BC_6, D(17),     bidir,         X,     13,   1,   Z)," &
   "11    (BC_6, D(16),     bidir,         X,     13,   1,   Z)," &
   "12    (BC_6, D(15),     bidir,         X,     13,   1,   Z)," &
   "13    (BC_1, *,         control,       1)," &
   "14    (BC_6, D(14),     bidir,         X,     13,   1,   Z)," &
   "15    (BC_6, D(13),     bidir,         X,     13,   1,   Z)," &
   "16    (BC_6, D(12),     bidir,         X,     13,   1,   Z)," &
```

```
   "17    (BC_6, D(11),     bidir,         X,     26,   1,   Z)," &
   "18    (BC_6, D(10),     bidir,         X,     26,   1,   Z)," &
   "19    (BC_6, D(9),      bidir,         X,     26,   1,   Z)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "20    (BC_6, D(8),      bidir,         X,     26,   1,   Z)," &
   "21    (BC_6, D(7),      bidir,         X,     26,   1,   Z)," &
   "22    (BC_6, D(6),      bidir,         X,     26,   1,   Z)," &
   "23    (BC_6, D(5),      bidir,         X,     26,   1,   Z)," &
   "24    (BC_6, D(4),      bidir,         X,     26,   1,   Z)," &
   "25    (BC_6, D(3),      bidir,         X,     26,   1,   Z)," &
   "26    (BC_1, *,         control,       1)," &
   "27    (BC_6, D(2),      bidir,         X,     26,   1,   Z)," &
   "28    (BC_6, D(1),      bidir,         X,     26,   1,   Z)," &
   "29    (BC_6, D(0),      bidir,         X,     26,   1,   Z)," &
   "30    (BC_1, A(17),     output3,       X,     33,   1,   Z)," &
   "31    (BC_1, A(16),     output3,       X,     33,   1,   Z)," &
   "32    (BC_1, A(15),     output3,       X,     33,   1,   Z)," &
   "33    (BC_1, *,         control,       1)," &
   "34    (BC_1, A(14),     output3,       X,     33,   1,   Z)," &
   "35    (BC_1, A(13),     output3,       X,     33,   1,   Z)," &
   "36    (BC_1, A(12),     output3,       X,     33,   1,   Z)," &
   "37    (BC_1, A(11),     output3,       X,     33,   1,   Z)," &
   "38    (BC_1, A(10),     output3,       X,     33,   1,   Z)," &
   "39    (BC_1, A(9),      output3,       X,     33,   1,   Z)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "40    (BC_1, A(8),      output3,       X,     43,   1,   Z)," &
   "41    (BC_1, A(7),      output3,       X,     43,   1,   Z)," &
   "42    (BC_1, A(6),      output3,       X,     43,   1,   Z)," &
   "43    (BC_1, *,         control,       1)," &
   "44    (BC_1, A(5),      output3,       X,     43,   1,   Z)," &
   "45    (BC_1, A(4),      output3,       X,     43,   1,   Z)," &
   "46    (BC_1, A(3),      output3,       X,     43,   1,   Z)," &
   "47    (BC_1, A(2),      output3,       X,     43,   1,   Z)," &
   "48    (BC_1, A(1),      output3,       X,     43,   1,   Z)," &
   "49    (BC_1, A(0),      output3,       X,     43,   1,   Z)," &
   "50    (BC_1, BG_N,      input,         X)," &
   "51    (BC_1, AA(0),     output3,       X,     55,   1,   Z)," &
   "52    (BC_1, AA(1),     output3,       X,     56,   1,   Z)," &
   "53    (BC_1, RD_N,      output3,       X,     64,   1,   Z)," &
   "54    (BC_1, WR_N,      output3,       X,     64,   1,   Z)," &
   "55    (BC_1, *,         control,       1)," &
   "56    (BC_1, *,         control,       1)," &
   "57    (BC_1, *,         control,       1)," &
   "58    (BC_6, BB_N,      bidir,         X,     57,   1,   Z)," &
   "59    (BC_1, BR_N,      output2,       X)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "60    (BC_1, TA_N,      input,         X)," &
   "61    (BC_1, BCLK_N,    output3,       X,     64,   1,   Z)," &
   "62    (BC_1, BCLK,      output3,       X,     64,   1,   Z)," &
   "63    (BC_1, CLKOUT,    output2,       X)," &
   "64    (BC_1, *,         control,       1)," &
   "65    (BC_1, *,         control,       1)," &
   "66    (BC_1, *,         control,       1)," &
```

```
   "67   (BC_1, *,         control,      1)," &
   "68   (BC_1, EXTAL,     input,        X)," &
   "69   (BC_1, CAS_N,     output3,      X,     65,   1,   Z)," &
   "70   (BC_1, AA(2),     output3,      X,     66,   1,   Z)," &
   "71   (BC_1, AA(3),     output3,      X,     67,   1,   Z)," &
   "72   (BC_1, RESET_N,   input,        X)," &
   "73   (BC_1, *,         control,      1)," &
   "74   (BC_6, HAD(0),    bidir,        X,     73,   1,   Z)," &
   "75   (BC_1, *,         control,      1)," &
   "76   (BC_6, HAD(1),    bidir,        X,     75,   1,   Z)," &
   "77   (BC_1, *,         control,      1)," &
   "78   (BC_6, HAD(2),    bidir,        X,     77,   1,   Z)," &
   "79   (BC_1, *,         control,      1)," &
-- num   cell   port   func          safe [ccell  dis  rslt]
   "80   (BC_6, HAD(3),    bidir,        X,     79,   1,   Z)," &
   "81   (BC_1, *,         control,      1)," &
   "82   (BC_6, HAD(4),    bidir,        X,     81,   1,   Z)," &
   "83   (BC_1, *,         control,      1)," &
   "84   (BC_6, HAD(5),    bidir,        X,     83,   1,   Z)," &
   "85   (BC_1, *,         control,      1)," &
   "86   (BC_6, HAD(6),    bidir,        X,     85,   1,   Z)," &
   "87   (BC_1, *,         control,      1)," &
   "88   (BC_6, HAD(7),    bidir,        X,     87,   1,   Z)," &
   "89   (BC_1, *,         control,      1)," &
   "90   (BC_6, HAS,       bidir,        X,     89,   1,   Z)," &
   "91   (BC_1, *,         control,      1)," &
   "92   (BC_6, HA8,       bidir,        X,     91,   1,   Z)," &
   "93   (BC_1, *,         control,      1)," &
   "94   (BC_6, HA9,       bidir,        X,     93,   1,   Z)," &
   "95   (BC_1, *,         control,      1)," &
   "96   (BC_6, HCS,       bidir,        X,     95,   1,   Z)," &
   "97   (BC_1, *,         control,      1)," &
   "98   (BC_6, TIO0,      bidir,        X,     97,   1,   Z)," &
   "99   (BC_1, *,         control,      1)," &
-- num   cell   port   func          safe [ccell  dis  rslt]
   "100  (BC_6, TIO1,      bidir,        X,     99,   1,   Z)," &
   "101  (BC_1, *,         control,      1)," &
   "102  (BC_6, TIO2,      bidir,        X,    101,   1,   Z)," &
   "103  (BC_1, *,         control,      1)," &
   "104  (BC_6, HREQ,      bidir,        X,    103,   1,   Z)," &
   "105  (BC_1, *,         control,      1)," &
   "106  (BC_6, HACK,      bidir,        X,    105,   1,   Z)," &
   "107  (BC_1, *,         control,      1)," &
   "108  (BC_6, HRW,       bidir,        X,    107,   1,   Z)," &
   "109  (BC_1, *,         control,      1)," &
   "110  (BC_6, HDS,       bidir,        X,    109,   1,   Z)," &
   "111  (BC_1, *,         control,      1)," &
   "112  (BC_6, SCK0,      bidir,        X,    111,   1,   Z)," &
   "113  (BC_1, *,         control,      1)," &
   "114  (BC_6, SCK1,      bidir,        X,    113,   1,   Z)," &
   "115  (BC_1, *,         control,      1)," &
   "116  (BC_6, SCLK,      bidir,        X,    115,   1,   Z)," &
   "117  (BC_1, *,         control,      1)," &
```

```
            "118   (BC_6, TXD,       bidir,         X,    117,   1,   Z)," &
            "119   (BC_1, *,         control,       1)," &
         -- num    cell   port   func           safe [ccell  dis  rslt]
            "120   (BC_6, RXD,       bidir,         X,    119,   1,   Z)," &
            "121   (BC_1, *,         control,       1)," &
            "122   (BC_6, SC00,      bidir,         X,    121,   1,   Z)," &
            "123   (BC_1, *,         control,       1)," &
            "124   (BC_6, SC10,      bidir,         X,    123,   1,   Z)," &
            "125   (BC_1, *,         control,       1)," &
            "126   (BC_6, STD0,      bidir,         X,    125,   1,   Z)," &
            "127   (BC_1, *,         control,       1)," &
            "128   (BC_6, SRD0,      bidir,         X,    127,   1,   Z)," &
            "129   (BC_1, PINIT,     input,         X)," &
            "130   (BC_1, *,         control,       1)," &
            "131   (BC_6, DE_N,      bidir,         X,    130,   1,   Z)," &
            "132   (BC_1, *,         control,       1)," &
            "133   (BC_6, SC01,      bidir,         X,    132,   1,   Z)," &
            "134   (BC_1, *,         control,       1)," &
            "135   (BC_6, SC02,      bidir,         X,    134,   1,   Z)," &
            "136   (BC_1, *,         control,       1)," &
            "137   (BC_6, STD1,      bidir,         X,    136,   1,   Z)," &
            "138   (BC_1, *,         control,       1)," &
            "139   (BC_6, SRD1,      bidir,         X,    138,   1,   Z)," &
         -- num    cell   port   func           safe [ccell  dis  rslt]
            "140   (BC_1, *,         control,       1)," &
            "141   (BC_6, SC11,      bidir,         X,    140,   1,   Z)," &
            "142   (BC_1, *,         control,       1)," &
            "143   (BC_6, SC12,      bidir,         X,    142,   1,   Z)";

end DSP56309 TQFP;


-----------------------------------------------------------------------
-- M O T O R O L A   S S D T   J T A G   S O F T W A R E
-- BSDL File Generated: Wed May 20 10:37:28 1998
--
-- Revision History:
--
-- 1) Date : Tue Mar  3 15:14:41 1998
--    Changes : Created for dsp56309 rev0, PBGA
--
-- 2) Date : Wed May 20 10:37:28 1998
--    Changes : Fix in definition of DE_N, it is Pull1 when disabled
--    Updated by Roman Sajman
--

entity DSP56309 is
        generic (PHYSICAL_PIN_MAP : string := "TQFP144");

        port (  DE_N:   inout   bit;
                SC02:   inout   bit;
                SC01:   inout   bit;
                SC00:   inout   bit;
```

```
      STD0:   inout   bit;
      SCK0:   inout   bit;
      SRD0:   inout   bit;
      SRD1:   inout   bit;
      SCK1:   inout   bit;
      STD1:   inout   bit;
      SC10:   inout   bit;
      SC11:   inout   bit;
      SC12:   inout   bit;
       TXD:   inout   bit;
      SCLK:   inout   bit;
       RXD:   inout   bit;
      TIO0:   inout   bit;
      TIO1:   inout   bit;
      TIO2:   inout   bit;
       HAD:   inout   bit_vector(0 to 7);
      HREQ:   inout   bit;
      MODD:   in      bit;
      MODC:   in      bit;
      MODB:   in      bit;
      MODA:   in      bit;
         D:   inout   bit_vector(0 to 23);
         A:   out     bit_vector(0 to 17);
     EXTAL:   in      bit;
      XTAL:   linkage bit;
      RD_N:   out     bit;
      WR_N:   out     bit;
        AA:   out     bit_vector(0 to 3);
      BR_N:   buffer  bit;
      BG_N:   in      bit;
      BB_N:   inout   bit;
      PCAP:   linkage bit;
   RESET_N:   in      bit;
     PINIT:   in      bit;
      TA_N:   in      bit;
     CAS_N:   out     bit;
      BCLK:   out     bit;
    BCLK_N:   out     bit;
   CLKOUT:   buffer  bit;
   TRST_N:   in      bit;
      TDO:   out     bit;
      TDI:   in      bit;
      TCK:   in      bit;
      TMS:   in      bit;
RESERVED:   linkage bit_vector(0 to 1);
     SGND:   linkage bit_vector(0 to 1);
     SVCC:   linkage bit_vector(0 to 1);
     QGND:   linkage bit_vector(0 to 3);
     QVCC:   linkage bit_vector(0 to 3);
     HGND:   linkage bit;
     HVCC:   linkage bit;
     DGND:   linkage bit_vector(0 to 3);
     DVCC:   linkage bit_vector(0 to 3);
```

```
        AGND:   linkage bit_vector(0 to 3);
        AVCC:   linkage bit_vector(0 to 3);
        JVCC:   linkage bit;
       JGND1:   linkage bit;
        JGND:   linkage bit;
        HACK:   inout   bit;
         HDS:   inout   bit;
         HRW:   inout   bit;
        CVCC:   linkage bit_vector(0 to 1);
        CGND:   linkage bit_vector(0 to 1);
         HCS:   inout   bit;
         HA9:   inout   bit;
         HA8:   inout   bit;
         HAS:   inout   bit);

use STD_1149_1_1994.all;

attribute COMPONENT_CONFORMANCE of DSP56309 : entity is "STD_1149_1_1993";

attribute PIN_MAP of DSP56309 : entity is PHYSICAL_PIN_MAP;

constant TQFP144 : PIN_MAP_STRING :=
        "SRD1:       1, " &
        "STD1:       2, " &
        "SC02:       3, " &
        "SC01:       4, " &
        "DE_N:       5, " &
        "PINIT:      6, " &
        "SRD0:       7, " &
        "SVCC:      (8, 25), " &
        "SGND:      (9, 26), " &
        "STD0:       10, " &
        "SC10:       11, " &
        "SC00:       12, " &
        "RXD:        13, " &
        "TXD:        14, " &
        "SCLK:       15, " &
        "SCK1:       16, " &
        "SCK0:       17, " &
        "QVCC:      (18, 56, 91, 126), " &
        "QGND:      (19, 54, 90, 127), " &
        "RESERVED:  (49, 20), " &
        "HDS:        21, " &
        "HRW:        22, " &
        "HACK:       23, " &
        "HREQ:       24, " &
        "TIO2:       27, " &
        "TIO1:       28, " &
        "TIO0:       29, " &
        "HCS:        30, " &
        "HA9:        31, " &
        "HA8:        32, " &
        "HAS:        33, " &
```

```
                "HAD:       (43, 42, 41, 40, 37, 36, 35, 34), " &
                "HVCC:       38, " &
                "HGND:       39, " &
                "RESET_N:    44, " &
                "JVCC:       45, " &
                "PCAP:       46, " &
                "JGND:       47, " &
                "JGND1:      48, " &
                "AA:        (70, 69, 51, 50), " &
                "CAS_N:      52, " &
                "XTAL:       53, " &
                "EXTAL:      55, " &
                "CVCC:      (57, 65), " &
                "CGND:      (58, 66), " &
                "CLKOUT:     59, " &
                "BCLK:       60, " &
                "BCLK_N:     61, " &
                "TA_N:       62, " &
                "BR_N:       63, " &
                "BB_N:       64, " &
                "WR_N:       67, " &
                "RD_N:       68, " &
                "BG_N:       71, " &
                "A:         (72, 73, 76, 77, 78, 79, 82, 83, 84, 85, 88, 89, 92, 93, 94, 97, 98,
99), " &
                "AVCC:      (74, 80, 86, 95), " &
                "AGND:      (75, 81, 87, 96), " &
               "D:         (100, 101, 102, 105, 106, 107, 108, 109, 110, 113, 114, 115, 116, 117,
118, 121,  " &
                        "122, 123, 124, 125, 128, 131, 132, 133), " &
                "DVCC:      (103, 111, 119, 129), " &
                "DGND:      (104, 112, 120, 130), " &
                "MODD:       134, " &
                "MODC:       135, " &
                "MODB:       136, " &
                "MODA:       137, " &
                "TRST_N:     138, " &
                "TDO:        139, " &
                "TDI:        140, " &
                "TCK:        141, " &
                "TMS:        142, " &
                "SC12:       143, " &
                "SC11:       144 ";

        attribute TAP_SCAN_IN    of     TDI : signal is true;
        attribute TAP_SCAN_OUT   of     TDO : signal is true;
        attribute TAP_SCAN_MODE  of     TMS : signal is true;
        attribute TAP_SCAN_RESET of  TRST_N : signal is true;
        attribute TAP_SCAN_CLOCK of     TCK : signal is (20.0e6, BOTH);

        attribute INSTRUCTION_LENGTH of DSP56309 : entity is 4;

        attribute INSTRUCTION_OPCODE of DSP56309 : entity is
```

```
   "EXTEST              (0000)," &
   "SAMPLE              (0001)," &
   "IDCODE              (0010)," &
   "CLAMP               (0101)," &
   "HIGHZ               (0100)," &
   "ENABLE_ONCE         (0110)," &
   "DEBUG_REQUEST       (0111)," &
   "BYPASS              (1111)";

attribute INSTRUCTION_CAPTURE of DSP56309 : entity is "0001";
attribute IDCODE_REGISTER   of DSP56309 : entity is
   "0010"          & -- version
   "000110"        & -- manufacturer's use
   "0000000010"    & -- sequence number
   "00000001110"   & -- manufacturer identity
   "1";              -- 1149.1 requirement

attribute REGISTER_ACCESS of DSP56309 : entity is
   "ONCE[8]   (ENABLE_ONCE,DEBUG_REQUEST)" ;

attribute BOUNDARY_LENGTH of DSP56309 : entity is 144;

attribute BOUNDARY_REGISTER of DSP56309 : entity is
-- num    cell   port   func          safe [ccell  dis  rslt]
   "0     (BC_1, MODA,      input,         X)," &
   "1     (BC_1, MODB,      input,         X)," &
   "2     (BC_1, MODC,      input,         X)," &
   "3     (BC_1, MODD,      input,         X)," &
   "4     (BC_6, D(23),     bidir,         X,     13,   1,   Z)," &
   "5     (BC_6, D(22),     bidir,         X,     13,   1,   Z)," &
   "6     (BC_6, D(21),     bidir,         X,     13,   1,   Z)," &
   "7     (BC_6, D(20),     bidir,         X,     13,   1,   Z)," &
   "8     (BC_6, D(19),     bidir,         X,     13,   1,   Z)," &
   "9     (BC_6, D(18),     bidir,         X,     13,   1,   Z)," &
   "10    (BC_6, D(17),     bidir,         X,     13,   1,   Z)," &
   "11    (BC_6, D(16),     bidir,         X,     13,   1,   Z)," &
   "12    (BC_6, D(15),     bidir,         X,     13,   1,   Z)," &
   "13    (BC_1, *,         control,       1)," &
   "14    (BC_6, D(14),     bidir,         X,     13,   1,   Z)," &
   "15    (BC_6, D(13),     bidir,         X,     13,   1,   Z)," &
   "16    (BC_6, D(12),     bidir,         X,     13,   1,   Z)," &
   "17    (BC_6, D(11),     bidir,         X,     26,   1,   Z)," &
   "18    (BC_6, D(10),     bidir,         X,     26,   1,   Z)," &
   "19    (BC_6, D(9),      bidir,         X,     26,   1,   Z)," &
-- num    cell   port   func          safe [ccell  dis  rslt]
   "20    (BC_6, D(8),      bidir,         X,     26,   1,   Z)," &
   "21    (BC_6, D(7),      bidir,         X,     26,   1,   Z)," &
   "22    (BC_6, D(6),      bidir,         X,     26,   1,   Z)," &
   "23    (BC_6, D(5),      bidir,         X,     26,   1,   Z)," &
   "24    (BC_6, D(4),      bidir,         X,     26,   1,   Z)," &
   "25    (BC_6, D(3),      bidir,         X,     26,   1,   Z)," &
   "26    (BC_1, *,         control,       1)," &
   "27    (BC_6, D(2),      bidir,         X,     26,   1,   Z)," &
```

```
    "28    (BC_6, D(1),      bidir,         X,     26,   1,   Z)," &
    "29    (BC_6, D(0),      bidir,         X,     26,   1,   Z)," &
    "30    (BC_1, A(17),     output3,       X,     33,   1,   Z)," &
    "31    (BC_1, A(16),     output3,       X,     33,   1,   Z)," &
    "32    (BC_1, A(15),     output3,       X,     33,   1,   Z)," &
    "33    (BC_1, *,         control,       1)," &
    "34    (BC_1, A(14),     output3,       X,     33,   1,   Z)," &
    "35    (BC_1, A(13),     output3,       X,     33,   1,   Z)," &
    "36    (BC_1, A(12),     output3,       X,     33,   1,   Z)," &
    "37    (BC_1, A(11),     output3,       X,     33,   1,   Z)," &
    "38    (BC_1, A(10),     output3,       X,     33,   1,   Z)," &
    "39    (BC_1, A(9),      output3,       X,     33,   1,   Z)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
    "40    (BC_1, A(8),      output3,       X,     43,   1,   Z)," &
    "41    (BC_1, A(7),      output3,       X,     43,   1,   Z)," &
    "42    (BC_1, A(6),      output3,       X,     43,   1,   Z)," &
    "43    (BC_1, *,         control,       1)," &
    "44    (BC_1, A(5),      output3,       X,     43,   1,   Z)," &
    "45    (BC_1, A(4),      output3,       X,     43,   1,   Z)," &
    "46    (BC_1, A(3),      output3,       X,     43,   1,   Z)," &
    "47    (BC_1, A(2),      output3,       X,     43,   1,   Z)," &
    "48    (BC_1, A(1),      output3,       X,     43,   1,   Z)," &
    "49    (BC_1, A(0),      output3,       X,     43,   1,   Z)," &
    "50    (BC_1, BG_N,      input,         X)," &
    "51    (BC_1, AA(0),     output3,       X,     55,   1,   Z)," &
    "52    (BC_1, AA(1),     output3,       X,     56,   1,   Z)," &
    "53    (BC_1, RD_N,      output3,       X,     64,   1,   Z)," &
    "54    (BC_1, WR_N,      output3,       X,     64,   1,   Z)," &
    "55    (BC_1, *,         control,       1)," &
    "56    (BC_1, *,         control,       1)," &
    "57    (BC_1, *,         control,       1)," &
    "58    (BC_6, BB_N,      bidir,         X,     57,   1,   Z)," &
    "59    (BC_1, BR_N,      output2,       X)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
    "60    (BC_1, TA_N,      input,         X)," &
    "61    (BC_1, BCLK_N,    output3,       X,     64,   1,   Z)," &
    "62    (BC_1, BCLK,      output3,       X,     64,   1,   Z)," &
    "63    (BC_1, CLKOUT,    output2,       X)," &
    "64    (BC_1, *,         control,       1)," &
    "65    (BC_1, *,         control,       1)," &
    "66    (BC_1, *,         control,       1)," &
    "67    (BC_1, *,         control,       1)," &
    "68    (BC_1, EXTAL,     input,         X)," &
    "69    (BC_1, CAS_N,     output3,       X,     65,   1,   Z)," &
    "70    (BC_1, AA(2),     output3,       X,     66,   1,   Z)," &
    "71    (BC_1, AA(3),     output3,       X,     67,   1,   Z)," &
    "72    (BC_1, RESET_N,   input,         X)," &
    "73    (BC_1, *,         control,       1)," &
    "74    (BC_6, HAD(0),    bidir,         X,     73,   1,   Z)," &
    "75    (BC_1, *,         control,       1)," &
    "76    (BC_6, HAD(1),    bidir,         X,     75,   1,   Z)," &
    "77    (BC_1, *,         control,       1)," &
    "78    (BC_6, HAD(2),    bidir,         X,     77,   1,   Z)," &
```

```
   "79    (BC_1, *,         control,       1)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "80    (BC_6, HAD(3),    bidir,         X,     79,   1,   Z)," &
   "81    (BC_1, *,         control,       1)," &
   "82    (BC_6, HAD(4),    bidir,         X,     81,   1,   Z)," &
   "83    (BC_1, *,         control,       1)," &
   "84    (BC_6, HAD(5),    bidir,         X,     83,   1,   Z)," &
   "85    (BC_1, *,         control,       1)," &
   "86    (BC_6, HAD(6),    bidir,         X,     85,   1,   Z)," &
   "87    (BC_1, *,         control,       1)," &
   "88    (BC_6, HAD(7),    bidir,         X,     87,   1,   Z)," &
   "89    (BC_1, *,         control,       1)," &
   "90    (BC_6, HAS,       bidir,         X,     89,   1,   Z)," &
   "91    (BC_1, *,         control,       1)," &
   "92    (BC_6, HA8,       bidir,         X,     91,   1,   Z)," &
   "93    (BC_1, *,         control,       1)," &
   "94    (BC_6, HA9,       bidir,         X,     93,   1,   Z)," &
   "95    (BC_1, *,         control,       1)," &
   "96    (BC_6, HCS,       bidir,         X,     95,   1,   Z)," &
   "97    (BC_1, *,         control,       1)," &
   "98    (BC_6, TIO0,      bidir,         X,     97,   1,   Z)," &
   "99    (BC_1, *,         control,       1)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "100   (BC_6, TIO1,      bidir,         X,     99,   1,   Z)," &
   "101   (BC_1, *,         control,       1)," &
   "102   (BC_6, TIO2,      bidir,         X,    101,   1,   Z)," &
   "103   (BC_1, *,         control,       1)," &
   "104   (BC_6, HREQ,      bidir,         X,    103,   1,   Z)," &
   "105   (BC_1, *,         control,       1)," &
   "106   (BC_6, HACK,      bidir,         X,    105,   1,   Z)," &
   "107   (BC_1, *,         control,       1)," &
   "108   (BC_6, HRW,       bidir,         X,    107,   1,   Z)," &
   "109   (BC_1, *,         control,       1)," &
   "110   (BC_6, HDS,       bidir,         X,    109,   1,   Z)," &
   "111   (BC_1, *,         control,       1)," &
   "112   (BC_6, SCK0,      bidir,         X,    111,   1,   Z)," &
   "113   (BC_1, *,         control,       1)," &
   "114   (BC_6, SCK1,      bidir,         X,    113,   1,   Z)," &
   "115   (BC_1, *,         control,       1)," &
   "116   (BC_6, SCLK,      bidir,         X,    115,   1,   Z)," &
   "117   (BC_1, *,         control,       1)," &
   "118   (BC_6, TXD,       bidir,         X,    117,   1,   Z)," &
   "119   (BC_1, *,         control,       1)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "120   (BC_6, RXD,       bidir,         X,    119,   1,   Z)," &
   "121   (BC_1, *,         control,       1)," &
   "122   (BC_6, SC00,      bidir,         X,    121,   1,   Z)," &
   "123   (BC_1, *,         control,       1)," &
   "124   (BC_6, SC10,      bidir,         X,    123,   1,   Z)," &
   "125   (BC_1, *,         control,       1)," &
   "126   (BC_6, STD0,      bidir,         X,    125,   1,   Z)," &
   "127   (BC_1, *,         control,       1)," &
   "128   (BC_6, SRD0,      bidir,         X,    127,   1,   Z)," &
```

```
          "129   (BC_1, PINIT,     input,         X)," &
          "130   (BC_1, *,         control,       1)," &
          "131   (BC_6, DE_N,      bidir,         X,    130,   1,   Pull1)," &
          "132   (BC_1, *,         control,       1)," &
          "133   (BC_6, SC01,      bidir,         X,    132,   1,   Z)," &
          "134   (BC_1, *,         control,       1)," &
          "135   (BC_6, SC02,      bidir,         X,    134,   1,   Z)," &
          "136   (BC_1, *,         control,       1)," &
          "137   (BC_6, STD1,      bidir,         X,    136,   1,   Z)," &
          "138   (BC_1, *,         control,       1)," &
          "139   (BC_6, SRD1,      bidir,         X,    138,   1,   Z)," &
       -- num    cell   port   func           safe [ccell  dis  rslt]
          "140   (BC_1, *,         control,       1)," &
          "141   (BC_6, SC11,      bidir,         X,    140,   1,   Z)," &
          "142   (BC_1, *,         control,       1)," &
          "143   (BC_6, SC12,      bidir,         X,    142,   1,   Z)";

end DSP56309;

    ---------------------------------------------------------------------
-- M O T O R O L A   S S D T   J T A G   S O F T W A R E
-- BSDL File Generated: Wed May 20 09:48:32 1998
--
-- Revision History:
--

entity DSP56309 is
       generic (PHYSICAL_PIN_MAP : string := "PBGA196");

       port (  DE_N:   inout   bit;
               SC02:   inout   bit;
               SC01:   inout   bit;
               SC00:   inout   bit;
               STD0:   inout   bit;
               SCK0:   inout   bit;
               SRD0:   inout   bit;
               SRD1:   inout   bit;
               SCK1:   inout   bit;
               STD1:   inout   bit;
               SC10:   inout   bit;
               SC11:   inout   bit;
               SC12:   inout   bit;
                TXD:   inout   bit;
               SCLK:   inout   bit;
                RXD:   inout   bit;
               TIO0:   inout   bit;
               TIO1:   inout   bit;
               TIO2:   inout   bit;
                HAD:   inout   bit_vector(0 to 7);
               HREQ:   inout   bit;
               MODD:   in      bit;
               MODC:   in      bit;
               MODB:   in      bit;
```

```
          MODA:   in      bit;
             D:   inout   bit_vector(0 to 23);
             A:   out     bit_vector(0 to 17);
         EXTAL:   in      bit;
          XTAL:   linkage bit;
          RD_N:   out     bit;
          WR_N:   out     bit;
            AA:   out     bit_vector(0 to 3);
          BR_N:   buffer  bit;
          BG_N:   in      bit;
          BB_N:   inout   bit;
          PCAP:   linkage bit;
       RESET_N:   in      bit;
         PINIT:   in      bit;
          TA_N:   in      bit;
         CAS_N:   out     bit;
          BCLK:   out     bit;
        BCLK_N:   out     bit;
        CLKOUT:   buffer  bit;
        TRST_N:   in      bit;
           TDO:   out     bit;
           TDI:   in      bit;
           TCK:   in      bit;
           TMS:   in      bit;
      RESERVED:   linkage bit_vector(0 to 4);
          SVCC:   linkage bit_vector(0 to 1);
          HVCC:   linkage bit;
          DVCC:   linkage bit_vector(0 to 3);
          AVCC:   linkage bit_vector(0 to 2);
          HACK:   inout   bit;
           HDS:   inout   bit;
           HRW:   inout   bit;
          CVCC:   linkage bit_vector(0 to 1);
           HCS:   inout   bit;
           HA9:   inout   bit;
           HA8:   inout   bit;
           HAS:   inout   bit;
           GND:   linkage bit_vector(0 to 63);
         QVCCL:   linkage bit_vector(0 to 3);
         QVCCH:   linkage bit_vector(0 to 2);
          PVCC:   linkage bit;
          PGND:   linkage bit;
         PGND1:   linkage bit);

use STD_1149_1_1994.all;

attribute COMPONENT_CONFORMANCE of DSP56309 : entity is “STD_1149_1_1993”;

attribute PIN_MAP of DSP56309 : entity is PHYSICAL_PIN_MAP;

constant PBGA196 : PIN_MAP_STRING :=
        “RESERVED:  (A1, A14, B14, P1, P14), “ &
        “SC11:       A2, “ &
```

```
                "TMS:        A3, " &
                "TDO:        A4, " &
                "MODB:       A5, " &
               "D:         (E14, D12, D13, C13, C14, B13, C12, A13, B12, A12, B11, A11, C10, B10,
A10, B9,  " &
                       "A9, B8, C8, A8, B7, B6, C6, A6), " &
                "DVCC:      (A7, C9, C11, D14), " &
                "SRD1:       B1, " &
                "SC12:       B2, " &
                "TDI:        B3, " &
                "TRST_N:     B4, " &
                "MODD:       B5, " &
                "SC02:       C1, " &
                "STD1:       C2, " &
                "TCK:        C3, " &
                "MODA:       C4, " &
                "MODC:       C5, " &
                "QVCCL:     (C7, G13, H2, N9), " &
                "PINIT:      D1, " &
                "SC01:       D2, " &
                "DE_N:       D3, " &
               "GND:       (E8, E9, E10, E11, F4, F5, F11, G4, G5, G6, G7, G8, G9, G10, G11, H4,
H5, H6,  " &
                       "H7, H8, H9, H10, H11, J4, J5, J6, J7, J8, J9, J10, J11, K4, K5, K6, K7,
K8, K9,  " &
                      "K10, K11, L4, L5, L6, L7, L8, L9, L10, L11, D4, D5, D6, D7, D8, D9, D10,
D11, E4,  " &
                       "E5, E6, E7, F6, F7, F8, F9, F10), " &
                "STD0:       E1, " &
                "SVCC:      (E2, K1), " &
                "SRD0:       E3, " &
               "A:         (N14, M13, M14, L13, L14, K13, K14, J13, J12, J14, H13, H14, G14, G12,
F13, F14,  " &
                       "E13, E12), " &
                "RXD:        F1, " &
                "SC10:       F2, " &
                "SC00:       F3, " &
                "QVCCH:     (F12, H1, M7), " &
                "SCK1:       G1, " &
                "SCLK:       G2, " &
                "TXD:        G3, " &
                "SCK0:       H3, " &
                "AVCC:      (H12, K12, L12), " &
                "HACK:       J1, " &
                "HRW:        J2, " &
                "HDS:        J3, " &
                "HREQ:       K2, " &
                "TIO2:       K3, " &
                "HCS:        L1, " &
                "TIO1:       L2, " &
                "TIO0:       L3, " &
                "HA8:        M1, " &
                "HA9:        M2, " &
```

```
        "HAS:        M3, " &
        "HVCC:       M4, " &
        "HAD:       (M5, P4, N4, P3, N3, P2, N1, N2), " &
        "PVCC:       M6, " &
        "EXTAL:      M8, " &
        "CLKOUT:     M9, " &
        "BCLK_N:     M10, " &
        "WR_N:       M11, " &
        "RD_N:       M12, " &
        "RESET_N:    N5, " &
        "PGND:       N6, " &
        "AA:        (N13, P12, P7, N7), " &
        "CAS_N:      N8, " &
        "BCLK:       N10, " &
        "BR_N:       N11, " &
        "CVCC:      (N12, P9), " &
        "PCAP:       P5, " &
        "PGND1:      P6, " &
        "XTAL:       P8, " &
        "TA_N:       P10, " &
        "BB_N:       P11, " &
        "BG_N:       P13 ";

attribute TAP_SCAN_IN    of     TDI : signal is true;
attribute TAP_SCAN_OUT   of     TDO : signal is true;
attribute TAP_SCAN_MODE  of     TMS : signal is true;
attribute TAP_SCAN_RESET of  TRST_N : signal is true;
attribute TAP_SCAN_CLOCK of     TCK : signal is (20.0e6, BOTH);

attribute INSTRUCTION_LENGTH of DSP56309 : entity is 4;

attribute INSTRUCTION_OPCODE of DSP56309 : entity is
   "EXTEST              (0000)," &
   "SAMPLE              (0001)," &
   "IDCODE              (0010)," &
   "CLAMP               (0101)," &
   "HIGHZ               (0100)," &
   "ENABLE_ONCE         (0110)," &
   "DEBUG_REQUEST       (0111)," &
   "BYPASS              (1111)";

attribute INSTRUCTION_CAPTURE of DSP56309 : entity is "0001";
attribute IDCODE_REGISTER   of DSP56309 : entity is
   "0010"          & -- version
   "000110"        & -- manufacturer's use
   "0000000010"    & -- sequence number
   "00000001110"   & -- manufacturer identity
   "1";              -- 1149.1 requirement

attribute REGISTER_ACCESS of DSP56309 : entity is
   "ONCE[8]   (ENABLE_ONCE,DEBUG_REQUEST)" ;

attribute BOUNDARY_LENGTH of DSP56309 : entity is 144;
```

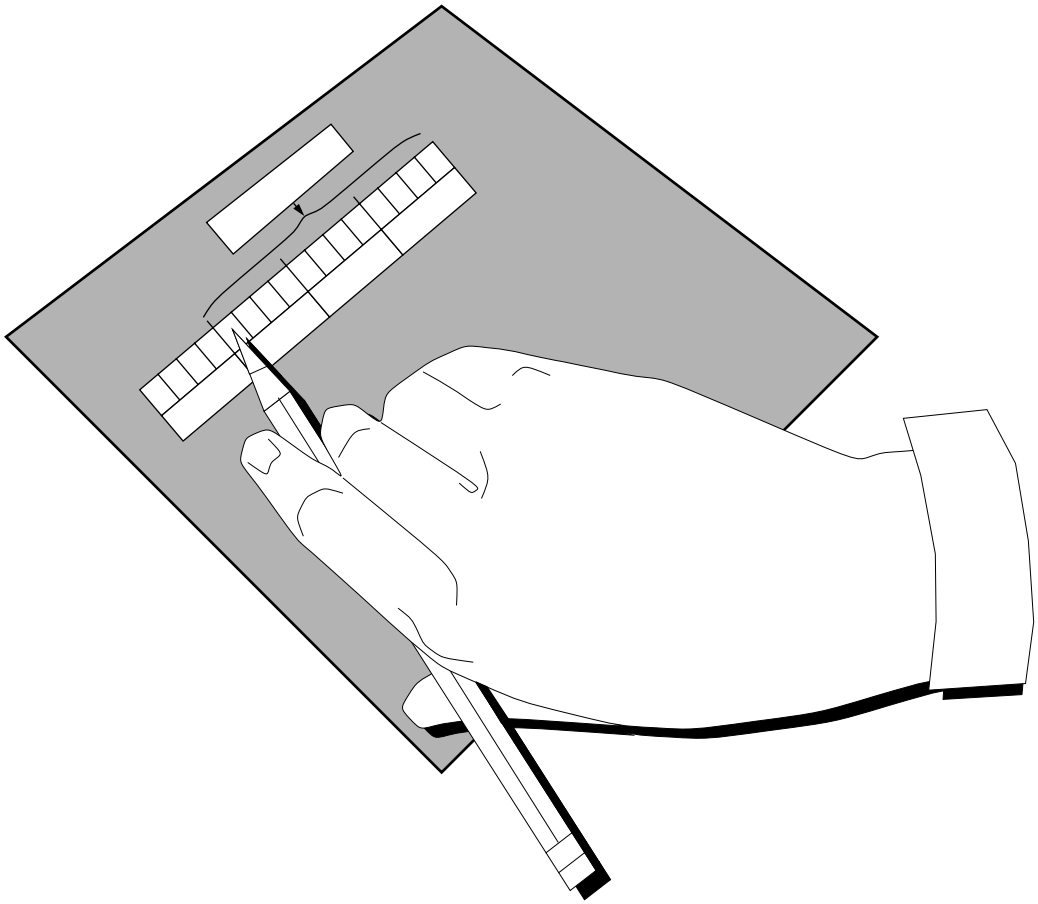
```
attribute BOUNDARY_REGISTER of DSP56309 : entity is
-- num    cell   port   func          safe [ccell  dis  rslt]
   "0     (BC_1, MODA,      input,         X)," &
   "1     (BC_1, MODB,      input,         X)," &
   "2     (BC_1, MODC,      input,         X)," &
   "3     (BC_1, MODD,      input,         X)," &
   "4     (BC_6, D(23),     bidir,         X,     13,   1,   Z)," &
   "5     (BC_6, D(22),     bidir,         X,     13,   1,   Z)," &
   "6     (BC_6, D(21),     bidir,         X,     13,   1,   Z)," &
   "7     (BC_6, D(20),     bidir,         X,     13,   1,   Z)," &
   "8     (BC_6, D(19),     bidir,         X,     13,   1,   Z)," &
   "9     (BC_6, D(18),     bidir,         X,     13,   1,   Z)," &
   "10    (BC_6, D(17),     bidir,         X,     13,   1,   Z)," &
   "11    (BC_6, D(16),     bidir,         X,     13,   1,   Z)," &
   "12    (BC_6, D(15),     bidir,         X,     13,   1,   Z)," &
   "13    (BC_1, *,         control,       1)," &
   "14    (BC_6, D(14),     bidir,         X,     13,   1,   Z)," &
   "15    (BC_6, D(13),     bidir,         X,     13,   1,   Z)," &
   "16    (BC_6, D(12),     bidir,         X,     13,   1,   Z)," &
   "17    (BC_6, D(11),     bidir,         X,     26,   1,   Z)," &
   "18    (BC_6, D(10),     bidir,         X,     26,   1,   Z)," &
   "19    (BC_6, D(9),      bidir,         X,     26,   1,   Z)," &
-- num    cell   port   func          safe [ccell  dis  rslt]
   "20    (BC_6, D(8),      bidir,         X,     26,   1,   Z)," &
   "21    (BC_6, D(7),      bidir,         X,     26,   1,   Z)," &
   "22    (BC_6, D(6),      bidir,         X,     26,   1,   Z)," &
   "23    (BC_6, D(5),      bidir,         X,     26,   1,   Z)," &
   "24    (BC_6, D(4),      bidir,         X,     26,   1,   Z)," &
   "25    (BC_6, D(3),      bidir,         X,     26,   1,   Z)," &
   "26    (BC_1, *,         control,       1)," &
   "27    (BC_6, D(2),      bidir,         X,     26,   1,   Z)," &
   "28    (BC_6, D(1),      bidir,         X,     26,   1,   Z)," &
   "29    (BC_6, D(0),      bidir,         X,     26,   1,   Z)," &
   "30    (BC_1, A(17),     output3,       X,     33,   1,   Z)," &
   "31    (BC_1, A(16),     output3,       X,     33,   1,   Z)," &
   "32    (BC_1, A(15),     output3,       X,     33,   1,   Z)," &
   "33    (BC_1, *,         control,       1)," &
   "34    (BC_1, A(14),     output3,       X,     33,   1,   Z)," &
   "35    (BC_1, A(13),     output3,       X,     33,   1,   Z)," &
   "36    (BC_1, A(12),     output3,       X,     33,   1,   Z)," &
   "37    (BC_1, A(11),     output3,       X,     33,   1,   Z)," &
   "38    (BC_1, A(10),     output3,       X,     33,   1,   Z)," &
   "39    (BC_1, A(9),      output3,       X,     33,   1,   Z)," &
-- num    cell   port   func          safe [ccell  dis  rslt]
   "40    (BC_1, A(8),      output3,       X,     43,   1,   Z)," &
   "41    (BC_1, A(7),      output3,       X,     43,   1,   Z)," &
   "42    (BC_1, A(6),      output3,       X,     43,   1,   Z)," &
   "43    (BC_1, *,         control,       1)," &
   "44    (BC_1, A(5),      output3,       X,     43,   1,   Z)," &
   "45    (BC_1, A(4),      output3,       X,     43,   1,   Z)," &
   "46    (BC_1, A(3),      output3,       X,     43,   1,   Z)," &
   "47    (BC_1, A(2),      output3,       X,     43,   1,   Z)," &
```

```
   "48    (BC_1, A(1),      output3,       X,     43,   1,   Z)," &
   "49    (BC_1, A(0),      output3,       X,     43,   1,   Z)," &
   "50    (BC_1, BG_N,      input,         X)," &
   "51    (BC_1, AA(0),     output3,       X,     55,   1,   Z)," &
   "52    (BC_1, AA(1),     output3,       X,     56,   1,   Z)," &
   "53    (BC_1, RD_N,      output3,       X,     64,   1,   Z)," &
   "54    (BC_1, WR_N,      output3,       X,     64,   1,   Z)," &
   "55    (BC_1, *,         control,       1)," &
   "56    (BC_1, *,         control,       1)," &
   "57    (BC_1, *,         control,       1)," &
   "58    (BC_6, BB_N,      bidir,         X,     57,   1,   Z)," &
   "59    (BC_1, BR_N,      output2,       X)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "60    (BC_1, TA_N,      input,         X)," &
   "61    (BC_1, BCLK_N,    output3,       X,     64,   1,   Z)," &
   "62    (BC_1, BCLK,      output3,       X,     64,   1,   Z)," &
   "63    (BC_1, CLKOUT,    output2,       X)," &
   "64    (BC_1, *,         control,       1)," &
   "65    (BC_1, *,         control,       1)," &
   "66    (BC_1, *,         control,       1)," &
   "67    (BC_1, *,         control,       1)," &
   "68    (BC_1, EXTAL,     input,         X)," &
   "69    (BC_1, CAS_N,     output3,       X,     65,   1,   Z)," &
   "70    (BC_1, AA(2),     output3,       X,     66,   1,   Z)," &
   "71    (BC_1, AA(3),     output3,       X,     67,   1,   Z)," &
   "72    (BC_1, RESET_N,   input,         X)," &
   "73    (BC_1, *,         control,       1)," &
   "74    (BC_6, HAD(0),    bidir,         X,     73,   1,   Z)," &
   "75    (BC_1, *,         control,       1)," &
   "76    (BC_6, HAD(1),    bidir,         X,     75,   1,   Z)," &
   "77    (BC_1, *,         control,       1)," &
   "78    (BC_6, HAD(2),    bidir,         X,     77,   1,   Z)," &
   "79    (BC_1, *,         control,       1)," &
-- num    cell   port   func           safe [ccell  dis  rslt]
   "80    (BC_6, HAD(3),    bidir,         X,     79,   1,   Z)," &
   "81    (BC_1, *,         control,       1)," &
   "82    (BC_6, HAD(4),    bidir,         X,     81,   1,   Z)," &
   "83    (BC_1, *,         control,       1)," &
   "84    (BC_6, HAD(5),    bidir,         X,     83,   1,   Z)," &
   "85    (BC_1, *,         control,       1)," &
   "86    (BC_6, HAD(6),    bidir,         X,     85,   1,   Z)," &
   "87    (BC_1, *,         control,       1)," &
   "88    (BC_6, HAD(7),    bidir,         X,     87,   1,   Z)," &
   "89    (BC_1, *,         control,       1)," &
   "90    (BC_6, HAS,       bidir,         X,     89,   1,   Z)," &
   "91    (BC_1, *,         control,       1)," &
   "92    (BC_6, HA8,       bidir,         X,     91,   1,   Z)," &
   "93    (BC_1, *,         control,       1)," &
   "94    (BC_6, HA9,       bidir,         X,     93,   1,   Z)," &
   "95    (BC_1, *,         control,       1)," &
   "96    (BC_6, HCS,       bidir,         X,     95,   1,   Z)," &
   "97    (BC_1, *,         control,       1)," &
   "98    (BC_6, TIO0,      bidir,         X,     97,   1,   Z)," &
```

```
      "99   (BC_1, *,        control,      1)," &
   -- num   cell   port   func          safe [ccell  dis  rslt]
      "100  (BC_6, TIO1,     bidir,        X,    99,   1,   Z)," &
      "101  (BC_1, *,        control,      1)," &
      "102  (BC_6, TIO2,     bidir,        X,    101,  1,   Z)," &
      "103  (BC_1, *,        control,      1)," &
      "104  (BC_6, HREQ,     bidir,        X,    103,  1,   Z)," &
      "105  (BC_1, *,        control,      1)," &
      "106  (BC_6, HACK,     bidir,        X,    105,  1,   Z)," &
      "107  (BC_1, *,        control,      1)," &
      "108  (BC_6, HRW,      bidir,        X,    107,  1,   Z)," &
      "109  (BC_1, *,        control,      1)," &
      "110  (BC_6, HDS,      bidir,        X,    109,  1,   Z)," &
      "111  (BC_1, *,        control,      1)," &
      "112  (BC_6, SCK0,     bidir,        X,    111,  1,   Z)," &
      "113  (BC_1, *,        control,      1)," &
      "114  (BC_6, SCK1,     bidir,        X,    113,  1,   Z)," &
      "115  (BC_1, *,        control,      1)," &
      "116  (BC_6, SCLK,     bidir,        X,    115,  1,   Z)," &
      "117  (BC_1, *,        control,      1)," &
      "118  (BC_6, TXD,      bidir,        X,    117,  1,   Z)," &
      "119  (BC_1, *,        control,      1)," &
   -- num   cell   port   func          safe [ccell  dis  rslt]
      "120  (BC_6, RXD,      bidir,        X,    119,  1,   Z)," &
      "121  (BC_1, *,        control,      1)," &
      "122  (BC_6, SC00,     bidir,        X,    121,  1,   Z)," &
      "123  (BC_1, *,        control,      1)," &
      "124  (BC_6, SC10,     bidir,        X,    123,  1,   Z)," &
      "125  (BC_1, *,        control,      1)," &
      "126  (BC_6, STD0,     bidir,        X,    125,  1,   Z)," &
      "127  (BC_1, *,        control,      1)," &
      "128  (BC_6, SRD0,     bidir,        X,    127,  1,   Z)," &
      "129  (BC_1, PINIT,    input,        X)," &
      "130  (BC_1, *,        control,      1)," &
      "131  (BC_6, DE_N,     bidir,        X,    130,  1,   Pull1)," &
      "132  (BC_1, *,        control,      1)," &
      "133  (BC_6, SC01,     bidir,        X,    132,  1,   Z)," &
      "134  (BC_1, *,        control,      1)," &
      "135  (BC_6, SC02,     bidir,        X,    134,  1,   Z)," &
      "136  (BC_1, *,        control,      1)," &
      "137  (BC_6, STD1,     bidir,        X,    136,  1,   Z)," &
      "138  (BC_1, *,        control,      1)," &
      "139  (BC_6, SRD1,     bidir,        X,    138,  1,   Z)," &
   -- num   cell   port   func          safe [ccell  dis  rslt]
      "140  (BC_1, *,        control,      1)," &
      "141  (BC_6, SC11,     bidir,        X,    140,  1,   Z)," &
      "142  (BC_1, *,        control,      1)," &
      "143  (BC_6, SC12,     bidir,        X,    142,  1,   Z)";

end DSP56309;
```

APPENDIX D

PROGRAMMING REFERENCE



D.1 INTRODUCTION

This section has been compiled as a reference for programmers. It contains a table showing the addresses of all the DSP's memory-mapped peripherals, an exception priority table, and programming sheets for the major programmable registers on the DSP. The programming sheets are grouped in the following order: central processor, Phase-Locked Loop (PLL), Host Interface (HI08), Enhanced Synchronous Serial Interface (ESSI), Serial Communication Interface (SCI), Timer, and GPIO. Each sheet provides room to write in the value of each bit and the hexadecimal value for each register. The programmer can photocopy these sheets and reuse them for each application development project. For details about the instruction set of the DSP56300 family chips, see the DSP56300 Family Manual.

D.1.1 Peripheral Addresses

Table D-1 lists the memory addresses of all on-chip peripherals.

D.1.2 Interrupt Addresses

Table D-2 on page -11 lists the interrupt starting addresses and sources.

D.1.3 Interrupt Priorities

Table D-3 on page -13 lists the priorities of specific interrupts within interrupt priority levels.

D.1.4 Programming Sheets

The remaining figures show the major programmable registers on the DSP56309.

D.2 INTERNAL I/O MEMORY MAP

Table D-1 Internal I/O Memory Map

Peripheral	16-Bit Address	24-Bit Address	Register Name
IPR	$FFFF	$FFFFFF	Interrupt Priority Register Core (IPR-C)
	$FFFE	$FFFFFE	Interrupt Priority Register Peripheral (IPR-P)
PLL	$FFFD	$FFFFFD	PLL Control Register (PCTL)
OnCE	$FFFC	$FFFFFC	OnCE GDB Register (OGDB)
BIU	$FFFB	$FFFFFB	Bus Control Register (BCR)
	$FFFA	$FFFFFA	DRAM Control Register (DCR)
	$FFF9	$FFFFF9	Address Attribute Register 0 (AAR0)
	$FFF8	$FFFFF8	Address Attribute Register 1 (AAR1)
	$FFF7	$FFFFF7	Address Attribute Register 2 (AAR2)
	$FFF6	$FFFFF6	Address Attribute Register 3 (AAR3)
	$FFF5	$FFFFF5	ID Register (IDR)
DMA	$FFF4	$FFFFF4	DMA Status Register (DSTR)
	$FFF3	$FFFFF3	DMA Offset Register 0 (DOR0)
	$FFF2	$FFFFF2	DMA Offset Register 1 (DOR1)
	$FFF1	$FFFFF1	DMA Offset Register 2 (DOR2)
	$FFF0	$FFFFF0	DMA Offset Register 3 (DOR3)
DMA0	$FFEF	$FFFFEF	DMA Source Address Register (DSR0)
	$FFEE	$FFFFEE	DMA Destination Address Register (DDR0)
	$FFED	$FFFFED	DMA Counter (DCO0)
	$FFEC	$FFFFEC	DMA Control Register (DCR0)

Table D-1 Internal I/O Memory Map (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
DMA1	$FFEB	$FFFFEB	DMA Source Address Register (DSR1)
	$FFEA	$FFFFEA	DMA Destination Address Register (DDR1)
	$FFE9	$FFFFE9	DMA Counter (DCO1)
	$FFE8	$FFFFE8	DMA Control Register (DCR1)
DMA2	$FFE7	$FFFFE7	DMA Source Address Register (DSR2)
	$FFE6	$FFFFE6	DMA Destination Address Register (DDR2)
	$FFE5	$FFFFE5	DMA Counter (DCO2)
	$FFE4	$FFFFE4	DMA Control Register (DCR2)
DMA3	$FFE3	$FFFFE3	DMA Source Address Register (DSR3)
	$FFE2	$FFFFE2	DMA Destination Address Register (DDR3)
	$FFE1	$FFFFE1	DMA Counter (DCO3)
	$FFE0	$FFFFE0	DMA Control Register (DCR3)
DMA4	$FFDF	$FFFFDF	DMA Source Address Register (DSR4)
	$FFDE	$FFFFDE	DMA Destination Address Register (DDR4)
	$FFDD	$FFFFDD	DMA Counter (DCO4)
	$FFDC	$FFFFDC	DMA Control Register (DCR4)
DMA5	$FFDB	$FFFFDB	DMA Source Address Register (DSR5)
	$FFDA	$FFFFDA	DMA Destination Address Register (DDR5)
	$FFD9	$FFFFD9	DMA Counter (DCO5)
	$FFD8	$FFFFD8	DMA Control Register (DCR5)

Table D-1 Internal I/O Memory Map (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
—	$FFD7	$FFFFD7	Reserved
	$FFD6	$FFFFD6	Reserved
	$FFD5	$FFFFD5	Reserved
	$FFD4	$FFFFD4	Reserved
	$FFD3	$FFFFD3	Reserved
	$FFD2	$FFFFD2	Reserved
	$FFD1	$FFFFD1	Reserved
	$FFD0	$FFFFD0	Reserved
—	$FFCF	$FFFFCF	Reserved
	$FFCE	$FFFFCE	Reserved
	$FFCD	$FFFFCD	Reserved
	$FFCC	$FFFFCC	Reserved
	$FFCB	$FFFFCB	Reserved
	$FFCA	$FFFFCA	Reserved
PORT B	$FFC9	$FFFFC9	Host Port GPIO Data Register (HDR)
	$FFC8	$FFFFC8	Host Port GPIO Direction Register (HDDR)
HI08	$FFC7	$FFFFC7	Host Transmit Register (HTX)
	$FFC6	$FFFFC6	Host Receive Register (HRX)
	$FFC5	$FFFFC5	Host Base Address Register (HBAR)
	$FFC4	$FFFFC4	Host Polarity Control Register (HPCR)
	$FFC3	$FFFFC3	Host Status Register (HSR)
	$FFC2	$FFFFC2	Host Control Register (HCR)
	$FFC1	$FFFFC1	Reserved
	$FFC0	$FFFFC0	Reserved

Table D-1 Internal I/O Memory Map (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
PORT C	$FFBF	$FFFFBF	Port C Control Register (PCRC)
	$FFBE	$FFFFBE	Port C Direction Register (PRRC)
	$FFBD	$FFFFBD	Port C GPIO Data Register (PDRC)
ESSI 0	$FFBC	$FFFFBC	ESSI 0 Transmit Data Register 0 (TX00)
	$FFBB	$FFFFBB	ESSI 0 Transmit Data Register 1 (TX01)
	$FFBA	$FFFFBA	ESSI 0 Transmit Data Register 2 (TX02)
	$FFB9	$FFFFB9	ESSI 0 Time Slot Register (TSR0)
	$FFB8	$FFFFB8	ESSI 0 Receive Data Register (RX0)
	$FFB7	$FFFFB7	ESSI 0 Status Register (SSISR0)
	$FFB6	$FFFFB6	ESSI 0 Control Register B (CRB0)
	$FFB5	$FFFFB5	ESSI 0 Control Register A (CRA0)
	$FFB4	$FFFFB4	ESSI 0 Transmit Slot Mask Register A (TSMA0)
	$FFB3	$FFFFB3	ESSI 0 Transmit Slot Mask Register B (TSMB0)
	$FFB2	$FFFFB2	ESSI 0 Receive Slot Mask Register A (RSMA0)
	$FFB1	$FFFFB1	ESSI 0 Receive Slot Mask Register B (RSMB0)
—	$FFB0	$FFFFB0	Reserved
PORT D	$FFAF	$FFFFAF	Port D Control Register (PCRD)
	$FFAE	$FFFFAE	Port D Direction Register (PRRD)
	$FFAD	$FFFFAD	Port C GPIO Data Register (PDRD)

Table D-1 Internal I/O Memory Map (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
ESSI 1	$FFAC	$FFFFAC	ESSI 1 Transmit Data Register 0 (TX10)
	$FFAB	$FFFFAB	ESSI 1 Transmit Data Register 1 (TX11)
	$FFAA	$FFFFAA	ESSI 1 Transmit Data Register 2 (TX12)
	$FFA9	$FFFFA9	ESSI 1 Time Slot Register (TSR1)
	$FFA8	$FFFFA8	ESSI 1 Receive Data Register (RX1)
	$FFA7	$FFFFA7	ESSI 1 Status Register (SSISR1)
	$FFA6	$FFFFA6	ESSI 1 Control Register B (CRB1)
	$FFA5	$FFFFA5	ESSI 1 Control Register A (CRA1)
	$FFA4	$FFFFA4	ESSI 1 Transmit Slot Mask Register A (TSMA1)
	$FFA3	$FFFFA3	ESSI 1 Transmit Slot Mask Register B (TSMB1)
	$FFA2	$FFFFA2	ESSI 1 Receive Slot Mask Register A (RSMA1)
	$FFA1	$FFFFA1	ESSI 1 Receive Slot Mask Register B (RSMB1)
—	$FFA0	$FFFFA0	Reserved
PORT E	$FF9F	$FFFF9F	Port E Control Register (PCRE)
	$FF9E	$FFFF9E	Port E Direction Register (PRRE)
	$FF9D	$FFFF9D	Port E GPIO Data Register (PDRE)

Table D-1 Internal I/O Memory Map (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
SCI	$FF9C	$FFFF9C	SCI Control Register (SCR)
	$FF9B	$FFFF9B	SCI Clock Control Register (SCCR)
	$FF9A	$FFFF9A	SCI Receive Data Register - High (SRXH)
	$FF99	$FFFF99	SCI Receive Data Register - Middle (SRXM)
	$FF98	$FFFF98	SCI Recieve Data Register - Low (SRXL)
	$FF97	$FFFF97	SCI Transmit Data Register - High (STXH)
	$FF96	$FFFF96	SCI Transmit Data Register - Middle (STXM)
	$FF95	$FFFF95	SCI Transmit Data Register - Low (STXL)
	$FF94	$FFFF94	SCI Transmit Address Register (STXA)
	$FF93	$FFFF93	SCI Status Register (SSR)
—	$FF92	$FFFF92	Reserved
	$FF91	$FFFF91	Reserved
	$FF90	$FFFF90	Reserved

Table D-1 Internal I/O Memory Map (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
TRIPLE TIMER	$FF8F	$FFFF8F	Timer 0 Control/Status Register (TCSR0)
	$FF8E	$FFFF8E	Timer 0 Load Register (TLR0)
	$FF8D	$FFFF8D	Timer 0 Compare Register (TCPR0)
	$FF8C	$FFFF8C	Timer 0 Count Register (TCR0)
	$FF8B	$FFFF8B	Timer 1 Control/Status Register (TCSR1)
	$FF8A	$FFFF8A	Timer 1 Load Register (TLR1)
	$FF89	$FFFF89	Timer 1 Compare Register (TCPR1)
	$FF88	$FFFF88	Timer 1 Count Register (TCR1)
	$FF87	$FFFF87	Timer 2 Control/Status Register (TCSR2)
	$FF86	$FFFF86	Timer 2 Load Register (TLR2)
	$FF85	$FFFF85	Timer 2 Compare Register (TCPR2)
	$FF84	$FFFF84	Timer 2 Count Register (TCR2)
	$FF83	$FFFF83	Timer Prescaler Load Register (TPLR)
	$FF82	$FFFF82	Timer Prescaler Count Register (TPCR)
—	$FF81	$FFFF81	Reserved
	$FF80	$FFFF80	Reserved

D.3 INTERRUPT ADDRESSES AND SOURCES

Table D-2 Interrupt Sources

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$00	3	Hardware $\overline{\text{RESET}}$
VBA:$02	3	Stack Error
VBA:$04	3	Illegal Instruction
VBA:$06	3	Debug Request Interrupt
VBA:$08	3	Trap
VBA:$0A	3	Non-Maskable Interrupt ($\overline{\text{NMI}}$)
VBA:$0C	3	Reserved
VBA:$0E	3	Reserved
VBA:$10	0–2	$\overline{\text{IRQA}}$
VBA:$12	0–2	$\overline{\text{IRQB}}$
VBA:$14	0–2	$\overline{\text{IRQC}}$
VBA:$16	0–2	$\overline{\text{IRQD}}$
VBA:$18	0–2	DMA Channel 0
VBA:$1A	0–2	DMA Channel 1
VBA:$1C	0–2	DMA Channel 2
VBA:$1E	0–2	DMA Channel 3
VBA:$20	0–2	DMA Channel 4
VBA:$22	0–2	DMA Channel 5
VBA:$24	0–2	TIMER 0 Compare
VBA:$26	0–2	TIMER 0 Overflow
VBA:$28	0–2	TIMER 1 Compare
VBA:$2A	0–2	TIMER 1 Overflow
VBA:$2C	0–2	TIMER 2 Compare
VBA:$2E	0–2	TIMER 2 Overflow
VBA:$30	0–2	ESSI0 Receive Data
VBA:$32	0–2	ESSI0 Receive Data With Exception Status
VBA:$34	0–2	ESSI0 Receive Last Slot
VBA:$36	0–2	ESSI0 Transmit Data

Table D-2 Interrupt Sources (Continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$38	0–2	ESSI0 Transmit Data With Exception Status
VBA:$3A	0–2	ESSI0 Transmit Last Slot
VBA:$3C	0–2	Reserved
VBA:$3E	0–2	Reserved
VBA:$40	0–2	ESSI1 Receive Data
VBA:$42	0–2	ESSI1 Receive Data With Exception Status
VBA:$44	0–2	ESSI1 Receive Last Slot
VBA:$46	0–2	ESSI1 Transmit Data
VBA:$48	0–2	ESSI1 Transmit Data With Exception Status
VBA:$4A	0–2	ESSI1 Transmit Last Slot
VBA:$4C	0–2	Reserved
VBA:$4E	0–2	Reserved
VBA:$50	0–2	SCI Receive Data
VBA:$52	0–2	SCI Receive Data With Exception Status
VBA:$54	0–2	SCI Transmit Data
VBA:$56	0–2	SCI Idle Line
VBA:$58	0–2	SCI Timer
VBA:$5A	0–2	Reserved
VBA:$5C	0–2	Reserved
VBA:$5E	0–2	Reserved
VBA:$60	0–2	Host Receive Data Full
VBA:$62	0–2	Host Transmit Data Empty
VBA:$64	0–2	Host Command (Default)
VBA:$66	0–2	Reserved
:	:	:
VBA:$FE	0–2	Reserved

D.4 INTERRUPT PRIORITIES

Table D-3 Interrupt Source Priorities within an IPL

Priority	Interrupt Source
Level 3 (Nonmaskable)	
Highest	Hardware $\overline{\text{RESET}}$
—	Stack Error
—	Illegal Instruction
—	Debug Request Interrupt
—	Trap
Lowest	Non-Maskable Interrupt
Levels 0, 1, 2 (Maskable)	
Highest	$\overline{\text{IRQA}}$ (External Interrupt)
—	$\overline{\text{IRQB}}$ (External Interrupt)
—	$\overline{\text{IRQC}}$ (External Interrupt)
—	$\overline{\text{IRQD}}$ (External Interrupt)
—	DMA Channel 0 Interrupt
—	DMA Channel 1 Interrupt
—	DMA Channel 2 Interrupt
—	DMA Channel 3 Interrupt
—	DMA Channel 4 Interrupt
—	DMA Channel 5 Interrupt
—	Host Command Interrupt
—	Host Transmit Data Empty
—	Host Receive Data Full
—	ESSI0 RX Data with Exception Interrupt

Table D-3 Interrupt Source Priorities within an IPL (Continued)

Priority	Interrupt Source
—	ESSI0 RX Data Interrupt
—	ESSI0 Receive Last Slot Interrupt
—	ESSI0 TX Data With Exception Interrupt
—	ESSI0 Transmit Last Slot Interrupt
—	ESSI0 TX Data Interrupt
—	ESSI1 RX Data With Exception Interrupt
—	ESSI1 RX Data Interrupt
—	ESSI1 Receive Last Slot Interrupt
—	ESSI1 TX Data With Exception Interrupt
—	ESSI1 Transmit Last Slot Interrupt
—	ESSI1 TX Data Interrupt
—	SCI Receive Data With Exception Interrupt
—	SCI Receive Data
—	SCI Transmit Data
—	SCI Idle Line
—	SCI Timer
—	TIMER0 Overflow Interrupt
—	TIMER0 Compare Interrupt
—	TIMER1 Overflow Interrupt
—	TIMER1 Compare Interrupt
—	TIMER2 Overflow Interrupt
Lowest	TIMER2 Compare Interrupt

Application:_____ Date:_____

_____ Programmer:_____

Sheet 1 of 5

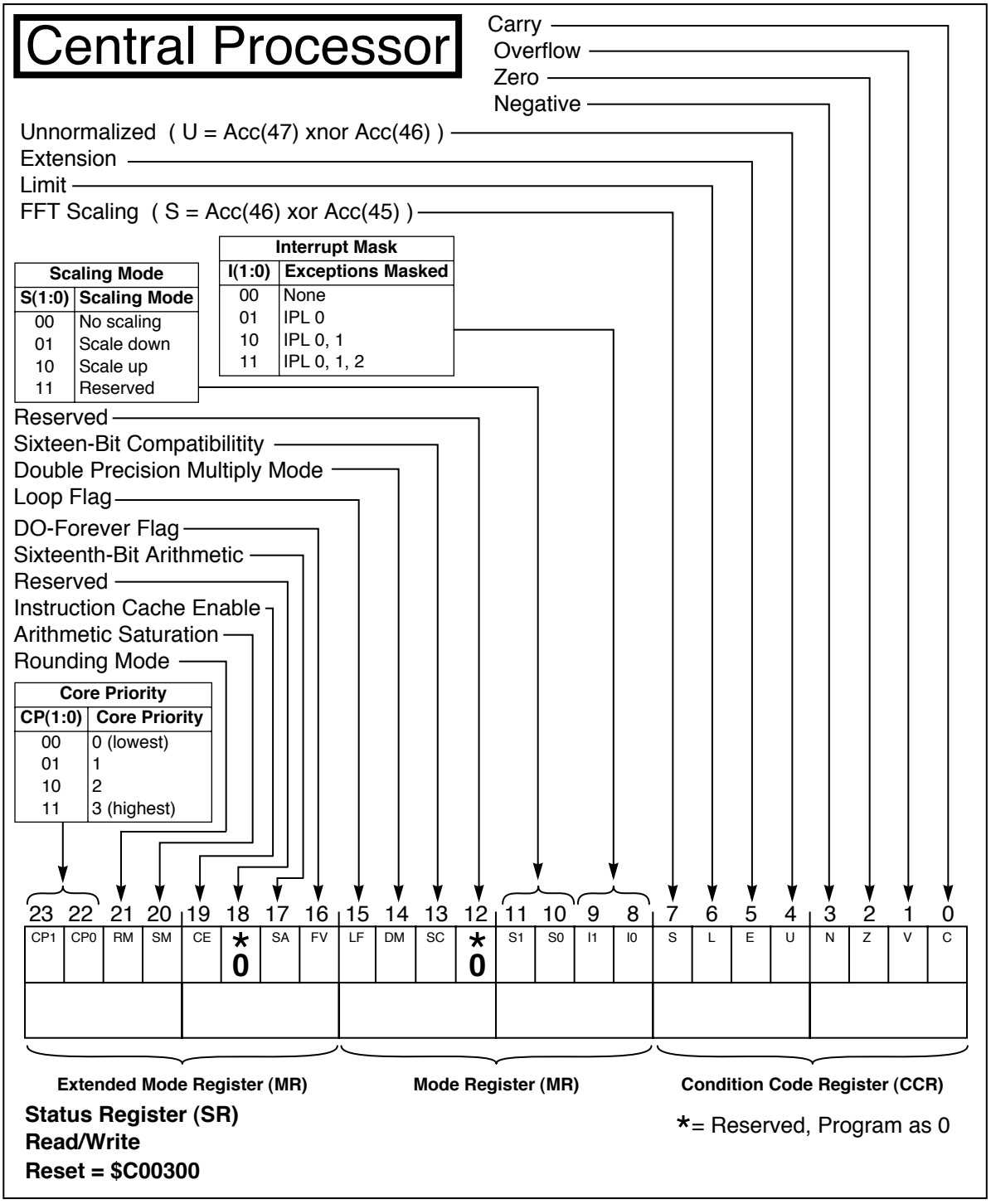


Figure D-1 Status Register (SR)

Application:_____ Date:_____

_____ Programmer:_____

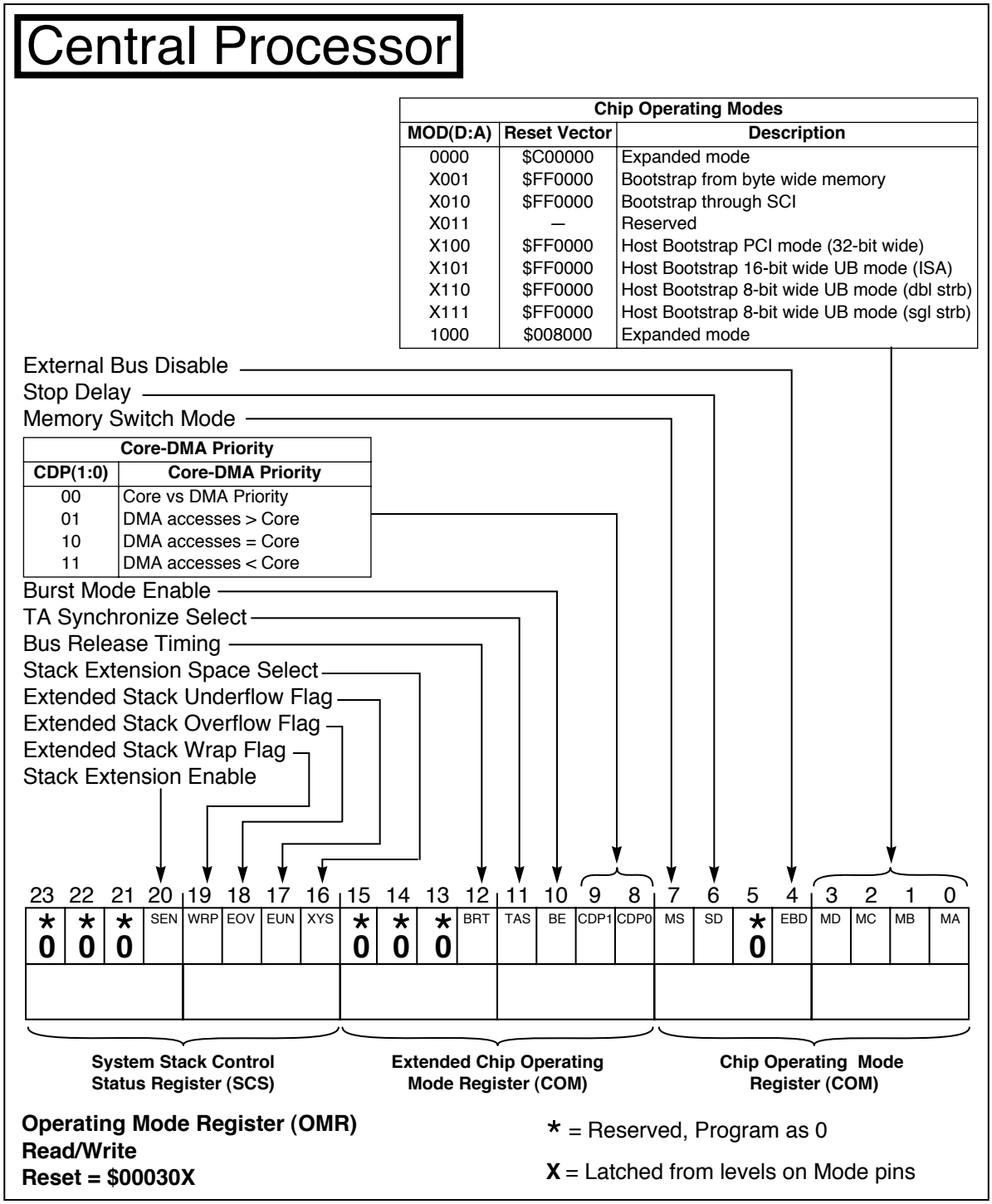


Figure D-2 Operating Mode Register (OMR)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 3 of 5

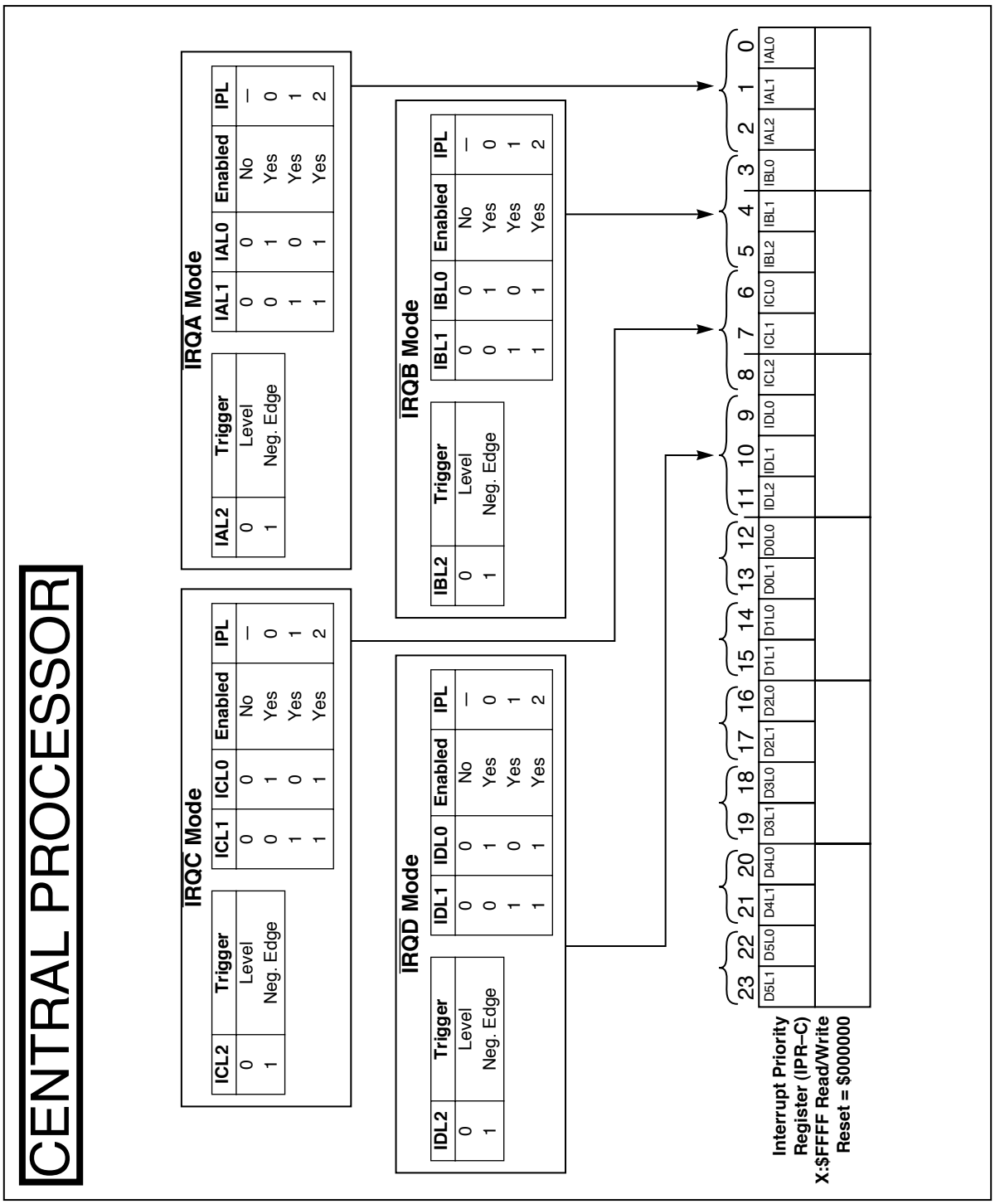


Figure D-3 Interrupt Priority Register–Core (IPR–C)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 4 of 5

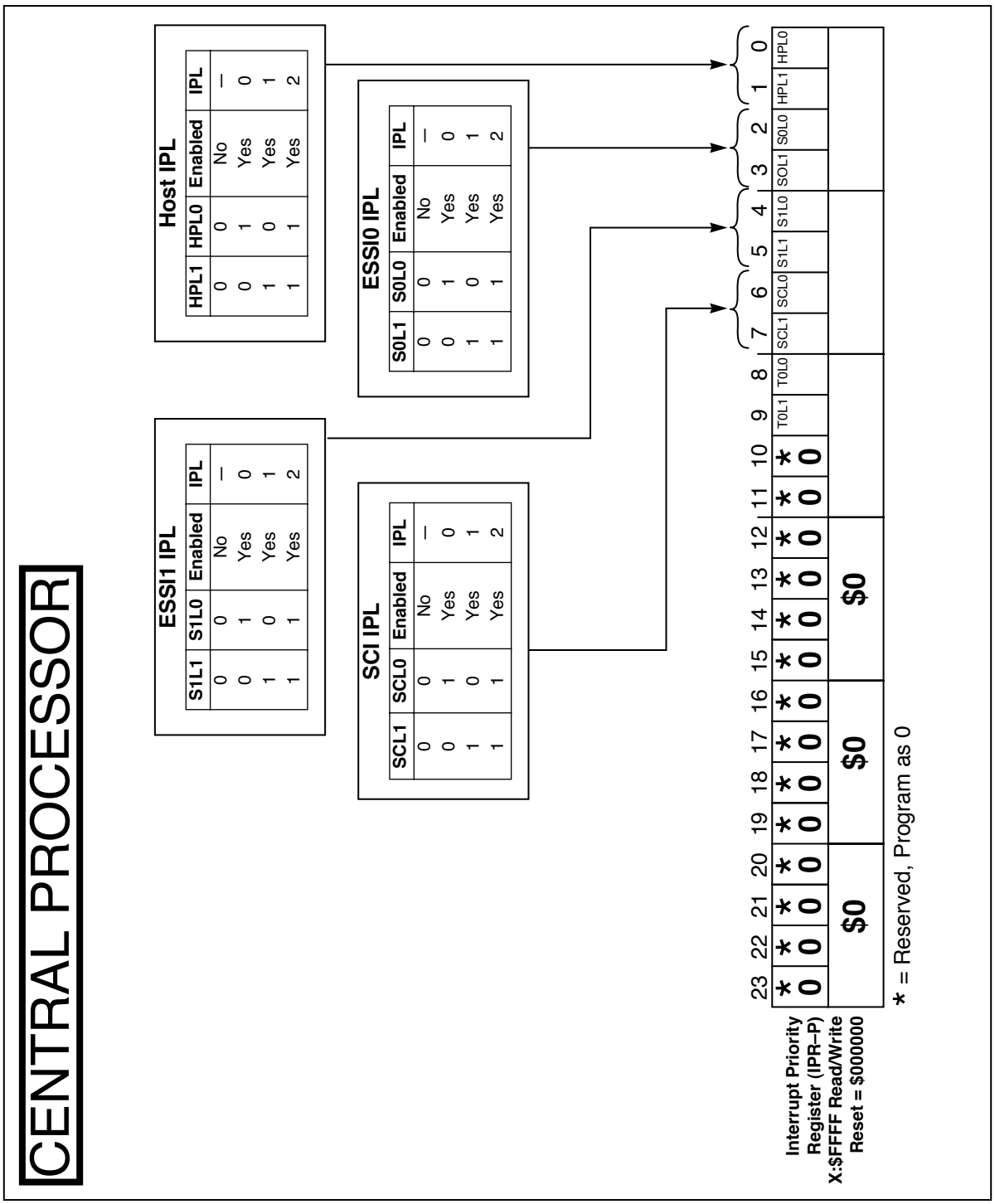


Figure D-4 Interrupt Priority Register – Peripherals (IPR–P)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 5 of 5

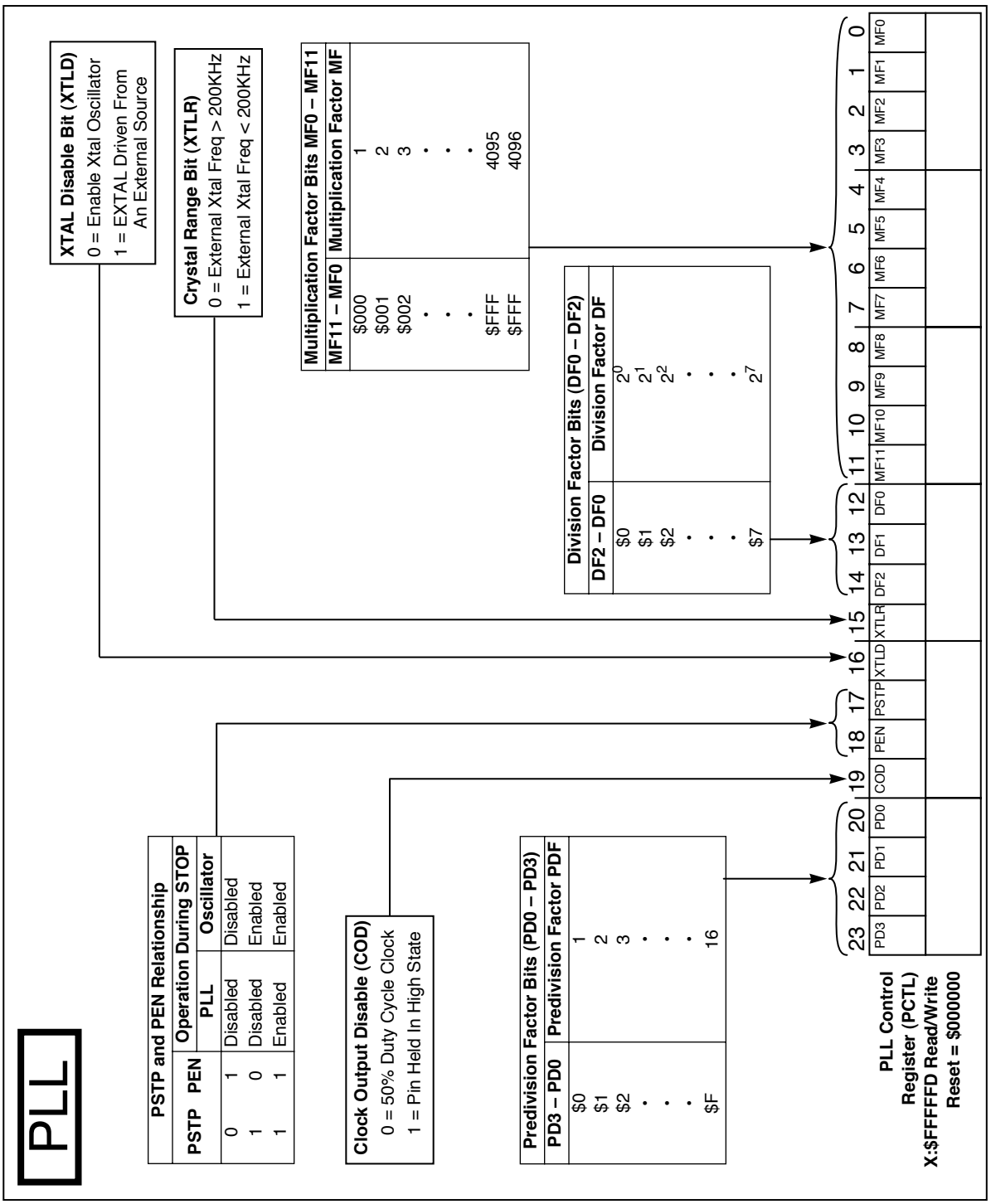


Figure D-5 Phase-Locked Loop Control Register (PCTL)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 1 of 6

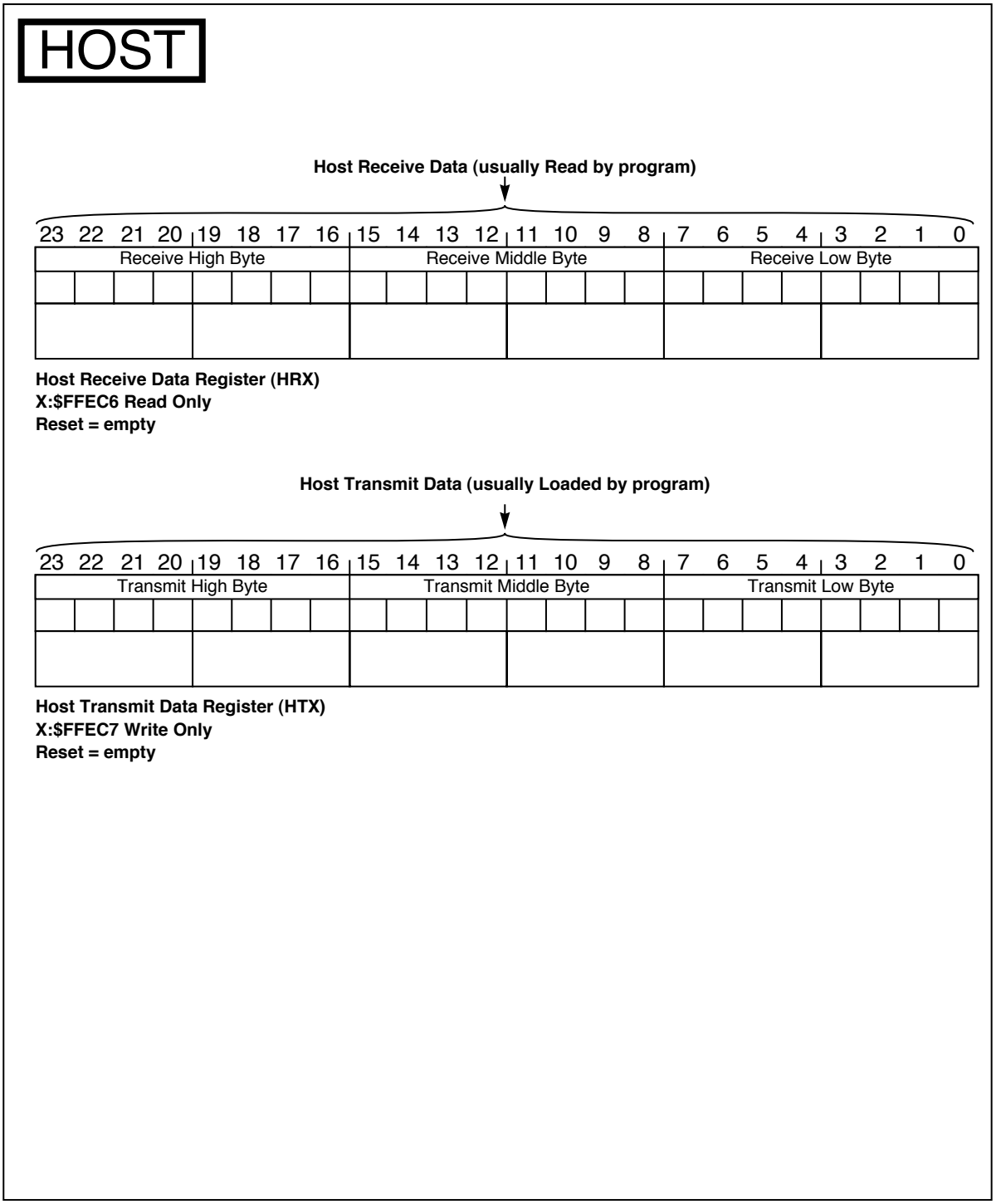


Figure D-6 Host Receive and Host Transmit Data Registers

Application:_____ Date:_____

_____ Programmer:_____

Sheet 2 of 6

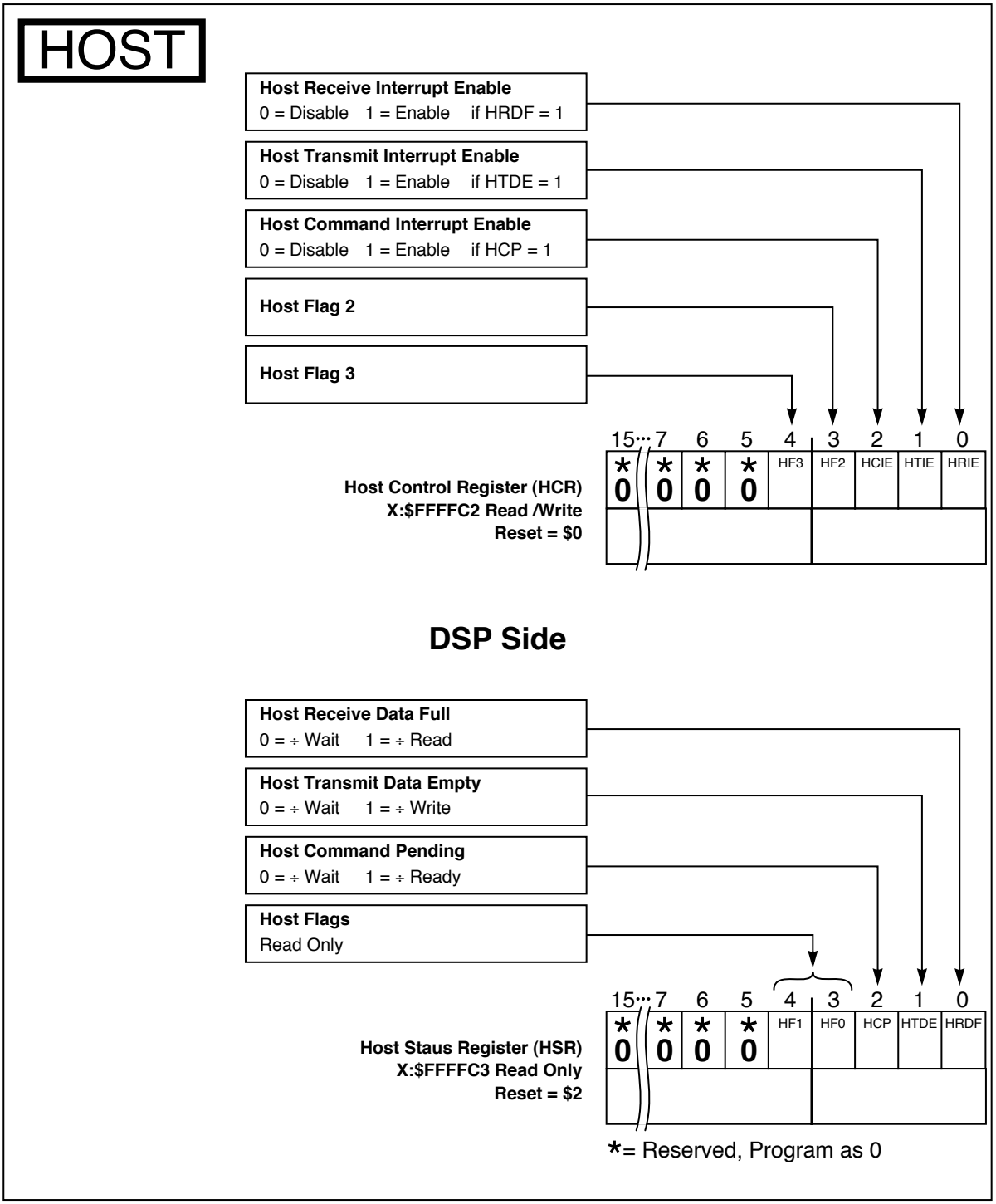


Figure D-7 Host Control and Host Status Registers

Application:_____ Date:_____

_____ Programmer:_____

Sheet 3 of 6

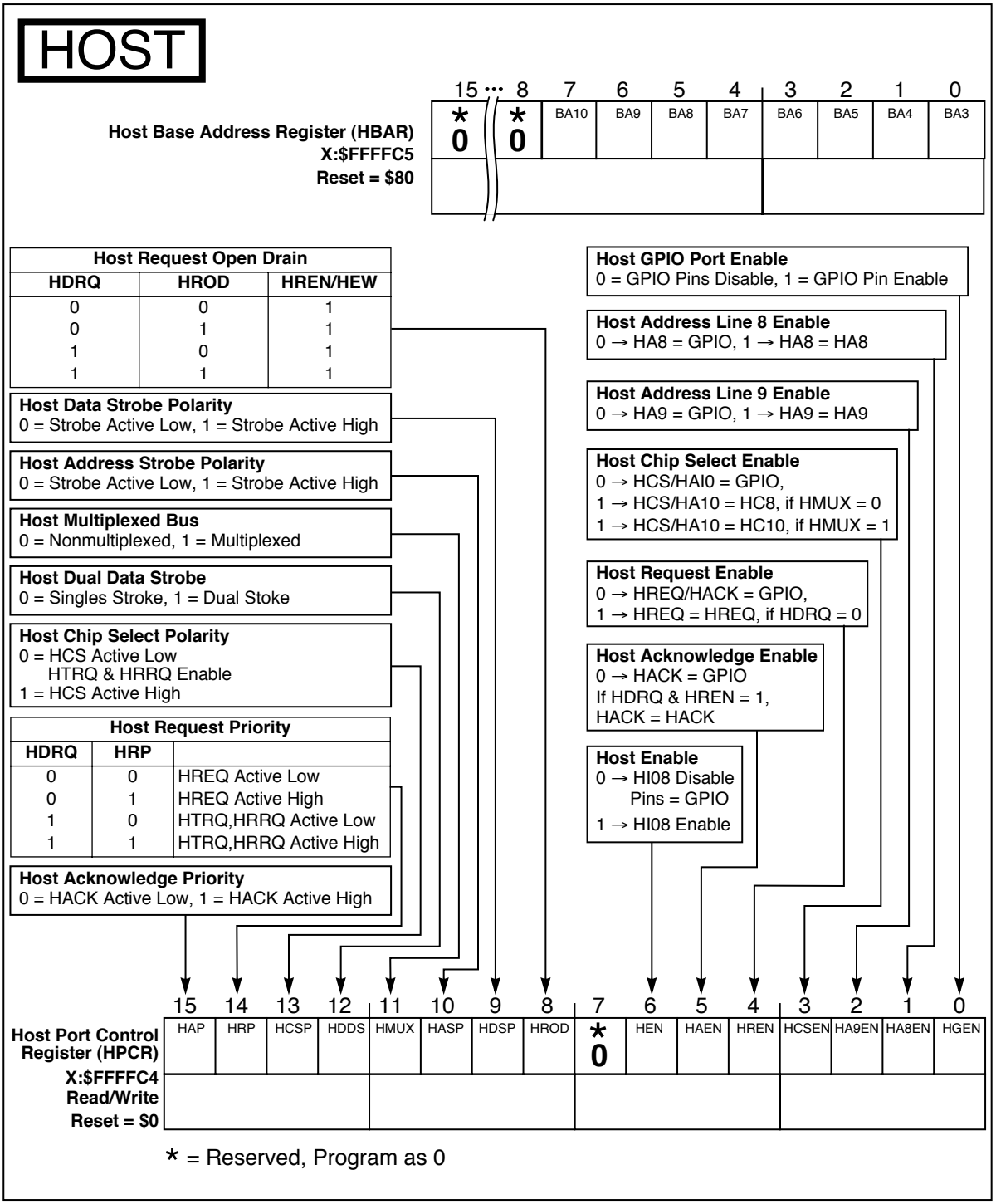


Figure D-8 Host Base Address and Host Port Control Registers

Application:_____ Date:_____

_____ Programmer:_____

Sheet 4 of 6

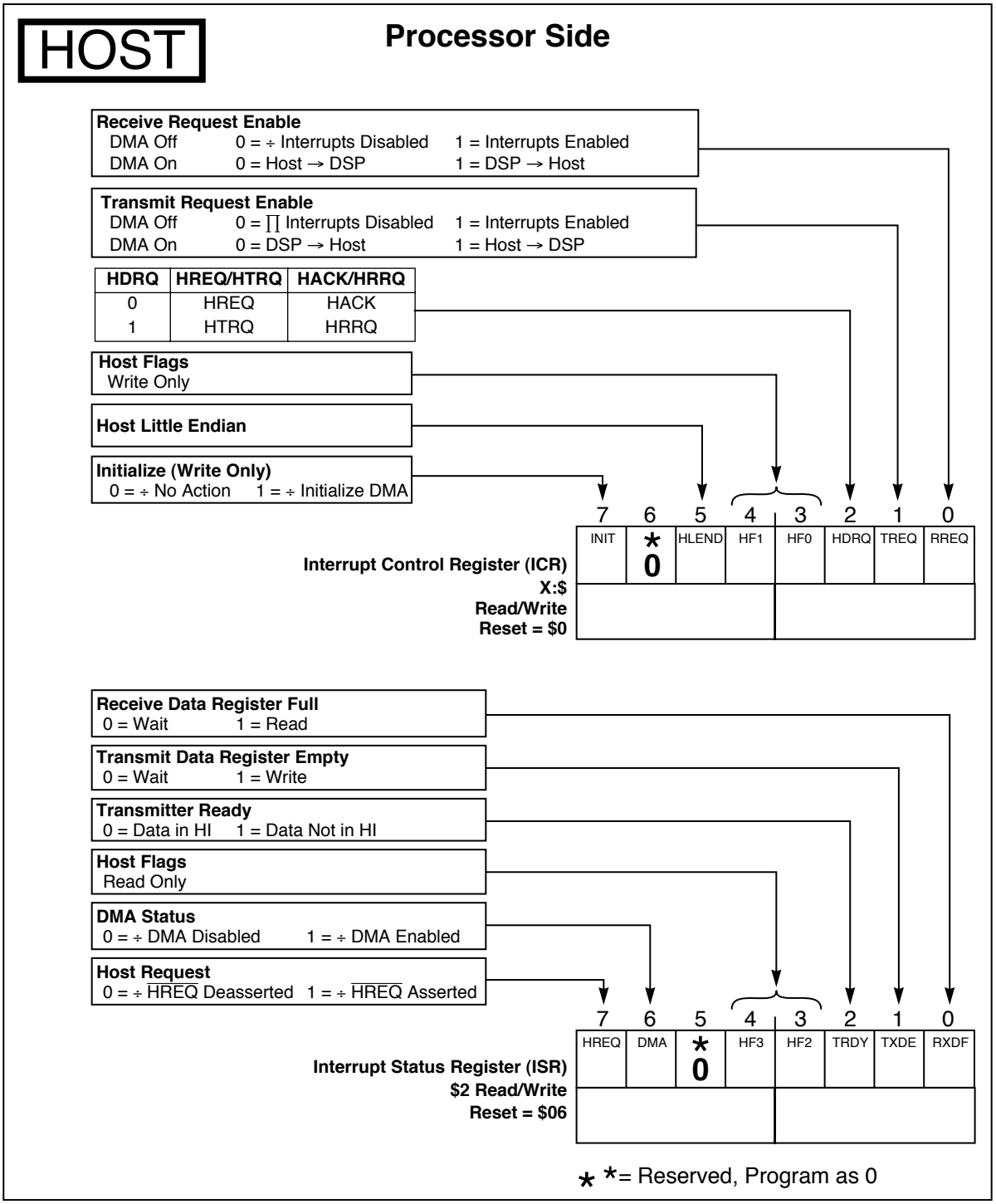


Figure D-9 Interrupt Control and Interrupt Status Registers

Application:_____ Date:_____

_____ Programmer:_____

Sheet 5 of 6

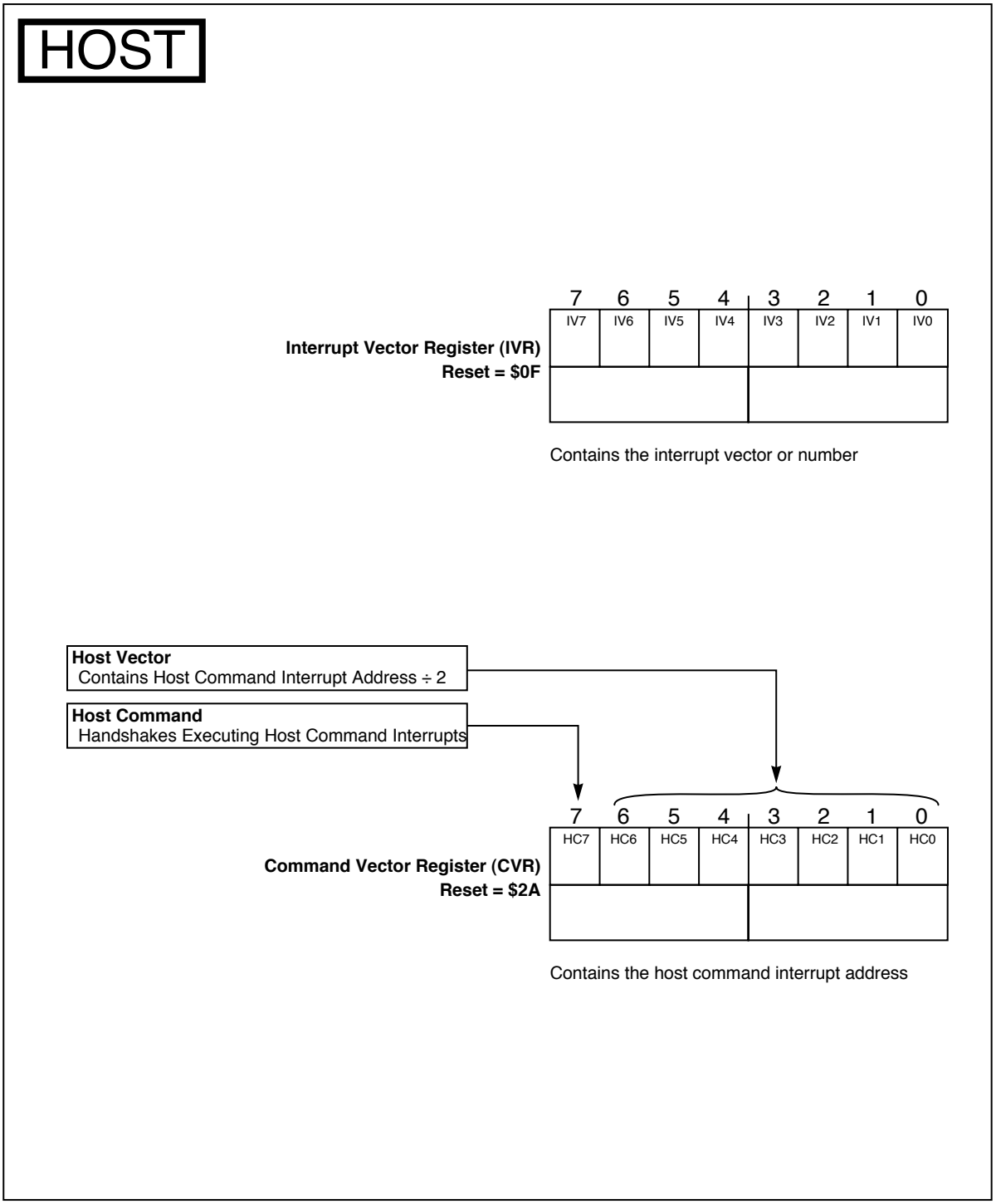


Figure D-10 Interrupt Vector and Command Vector Registers

Application:_____ Date:_____

_____ Programmer:_____

Sheet 6 of 6

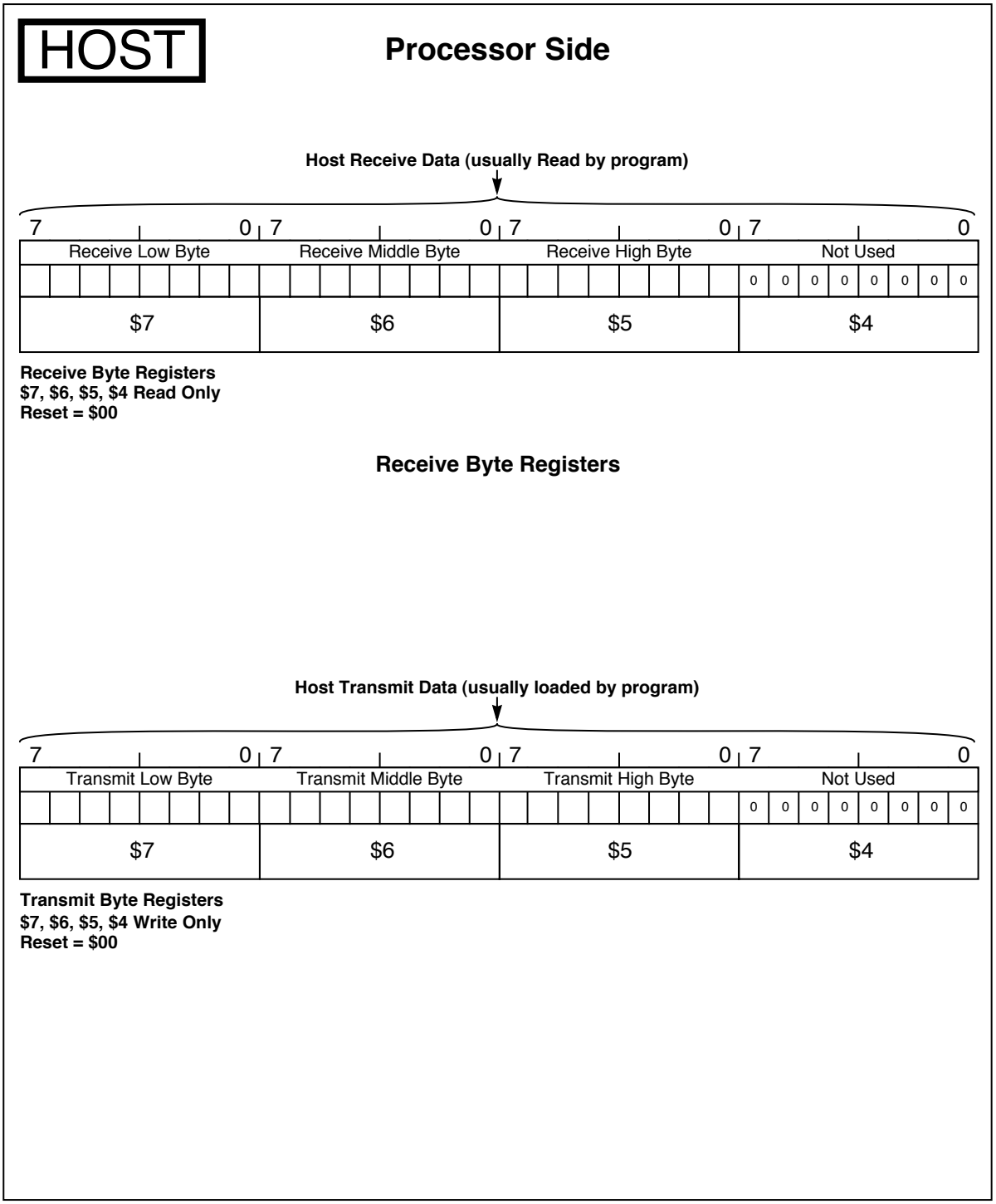


Figure D-11 Host Receive and Host Transmit Data Registers

Application:_____ Date:_____

_____ Programmer:_____

Sheet 1 of 4

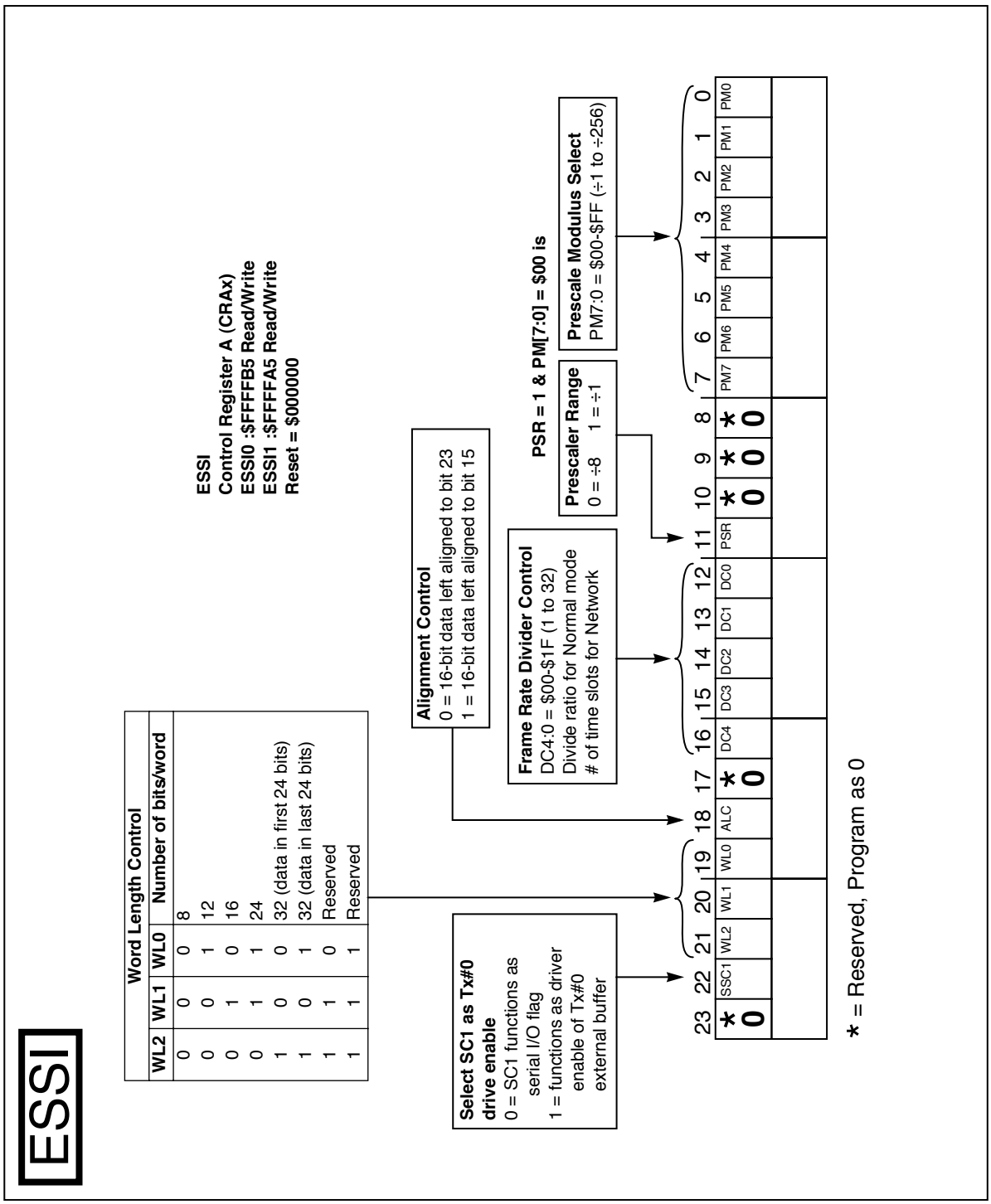


Figure D-12 ESSI Control Register A (CRA)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 2 of 4

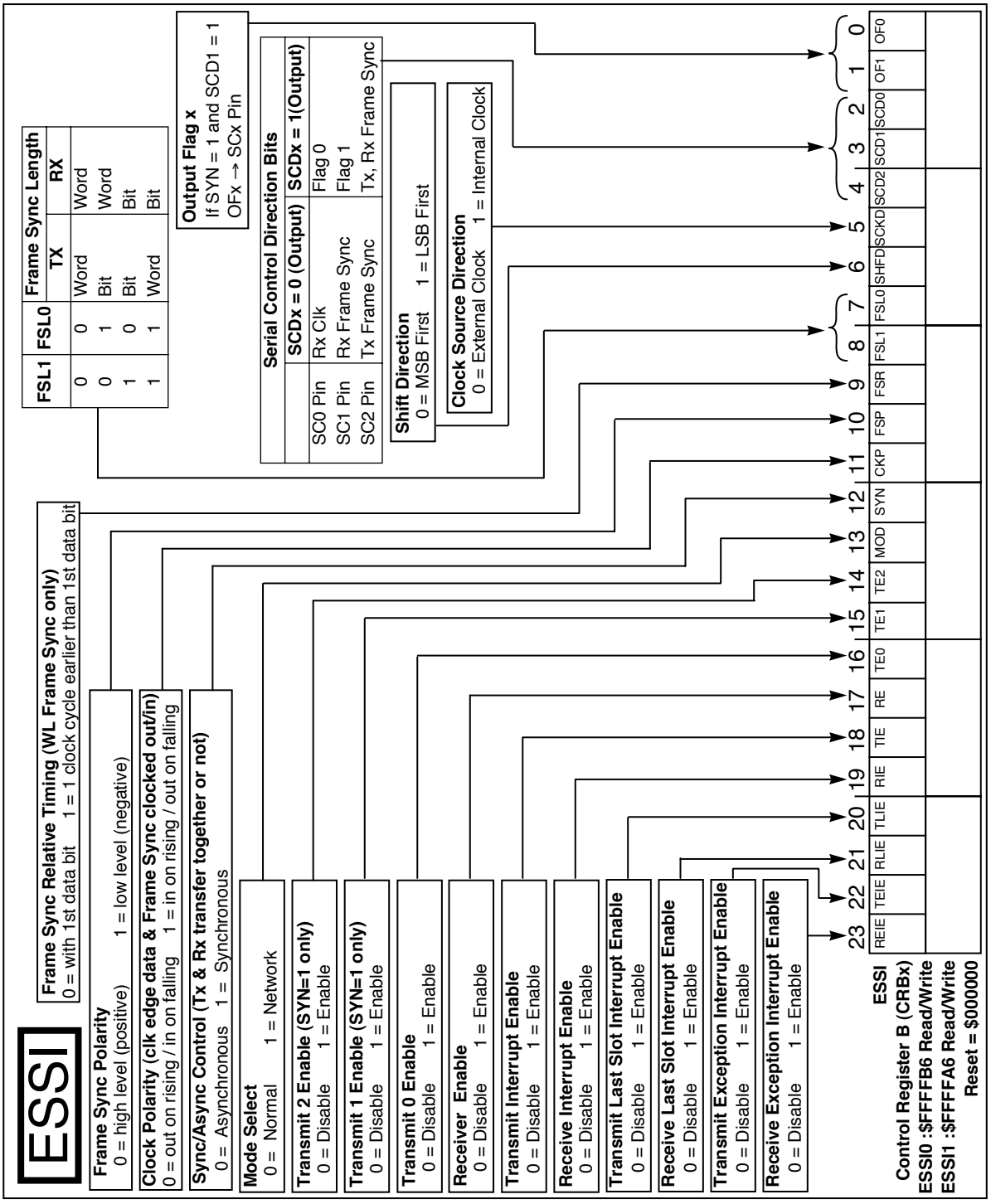


Figure D-13 ESSI Control Register B (CRB)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 3 of 4

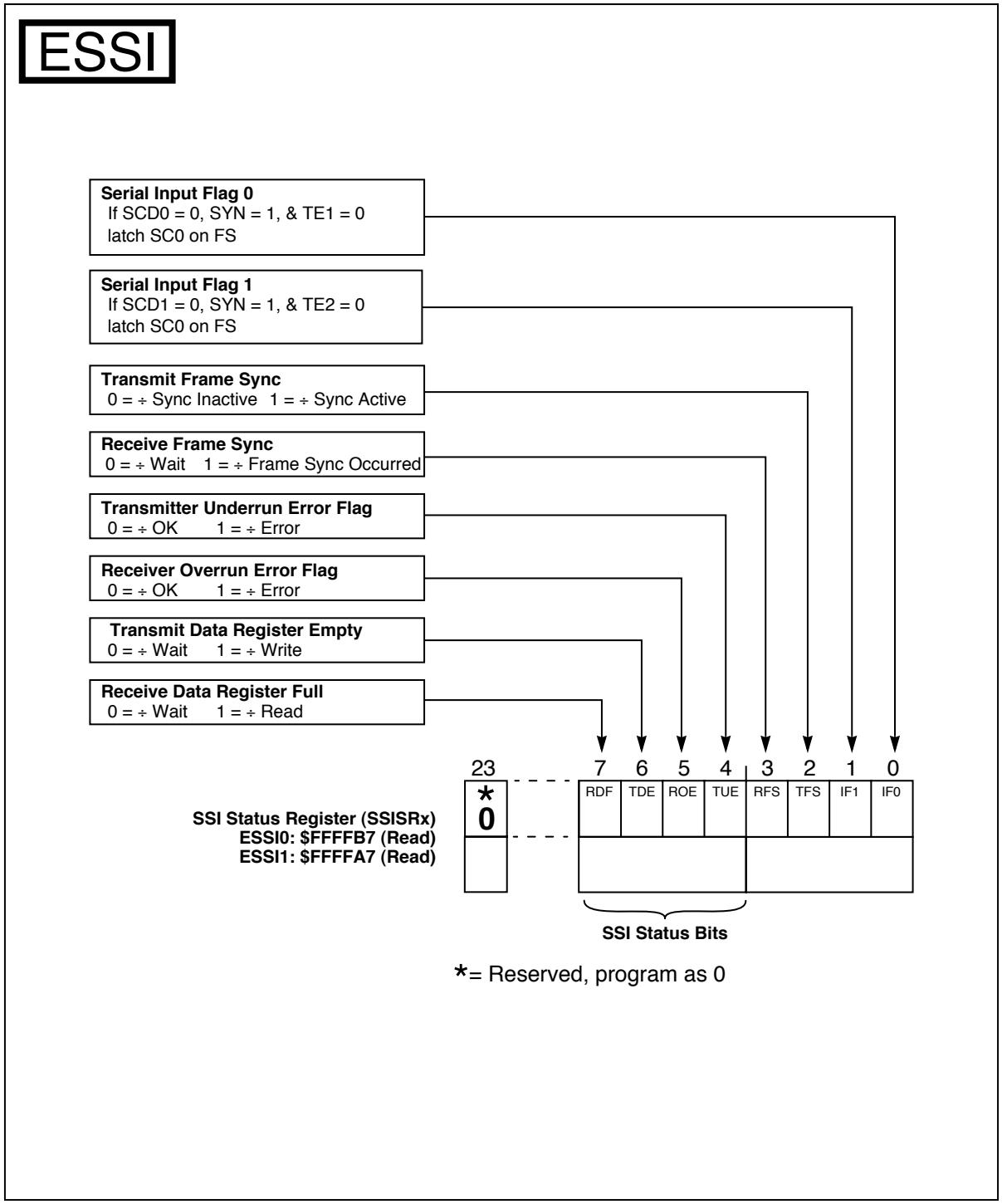


Figure D-14 ESSI Status Register (SSISR)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 4 of 4

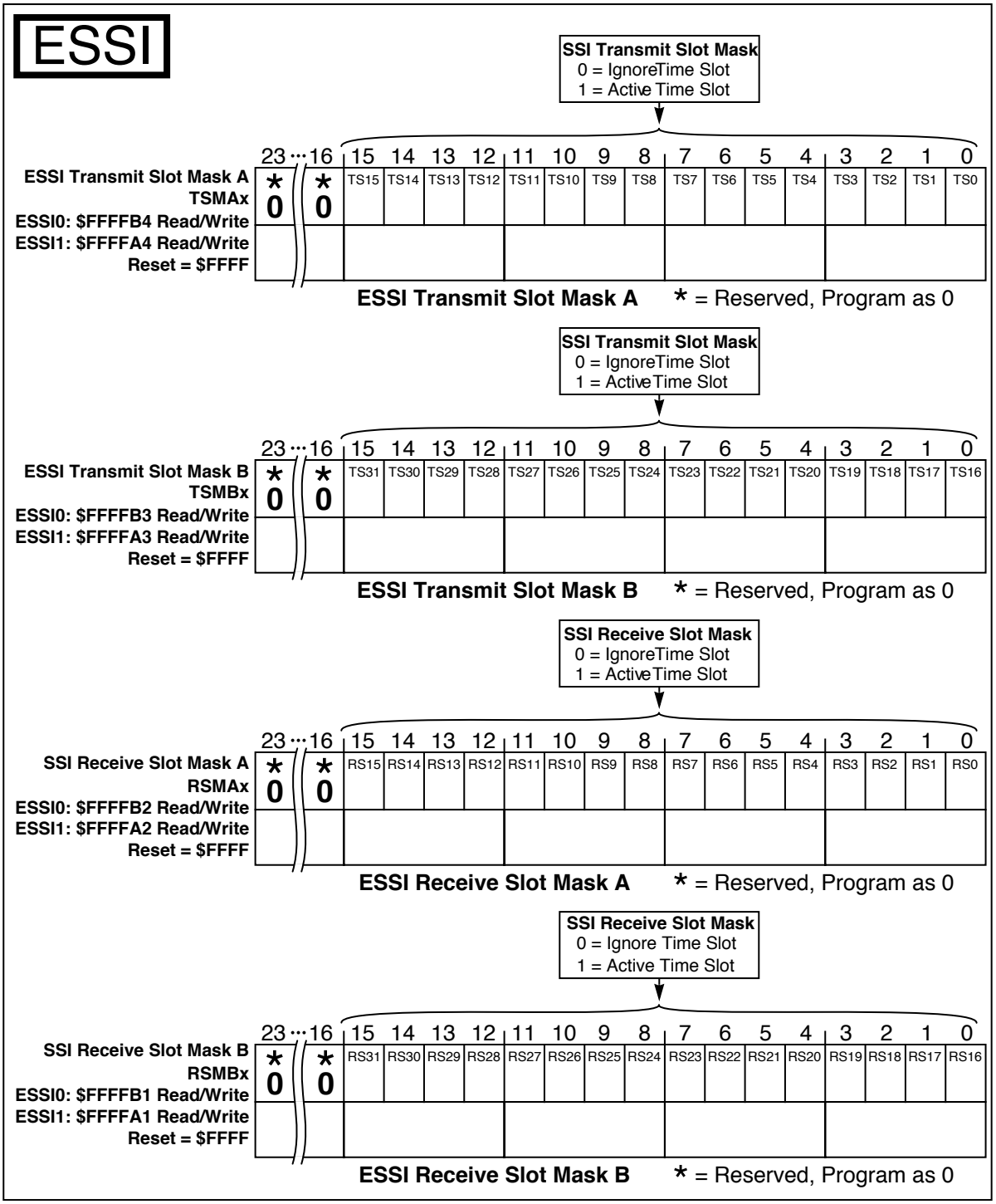
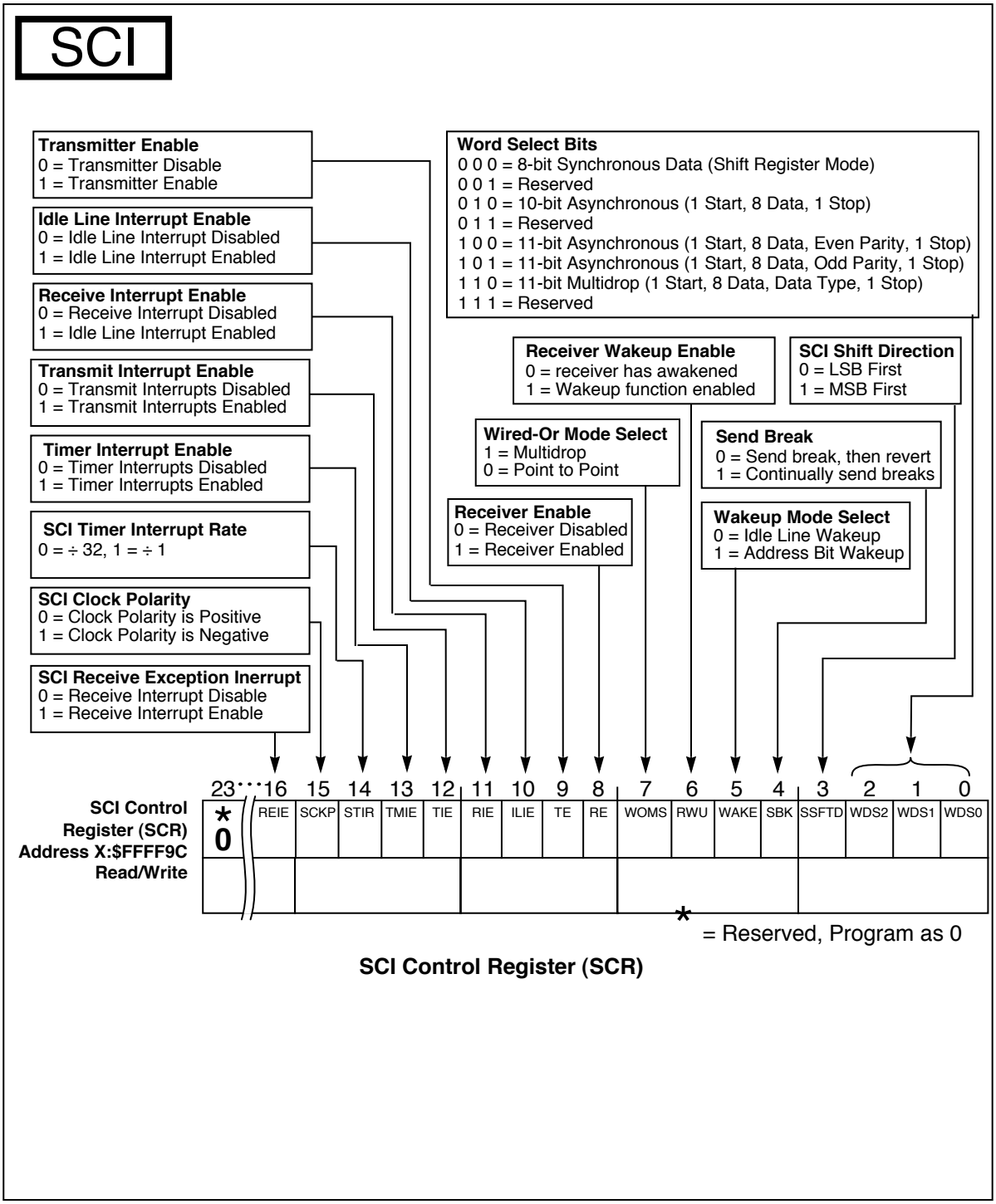


Figure D-15 ESSR Transmit and Receive Slot Mask Registers (TSM, RSM)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 1 of 3



SCI Control Register (SCR)

Figure D-16 SCI Control Register (SCR)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 2 of 3

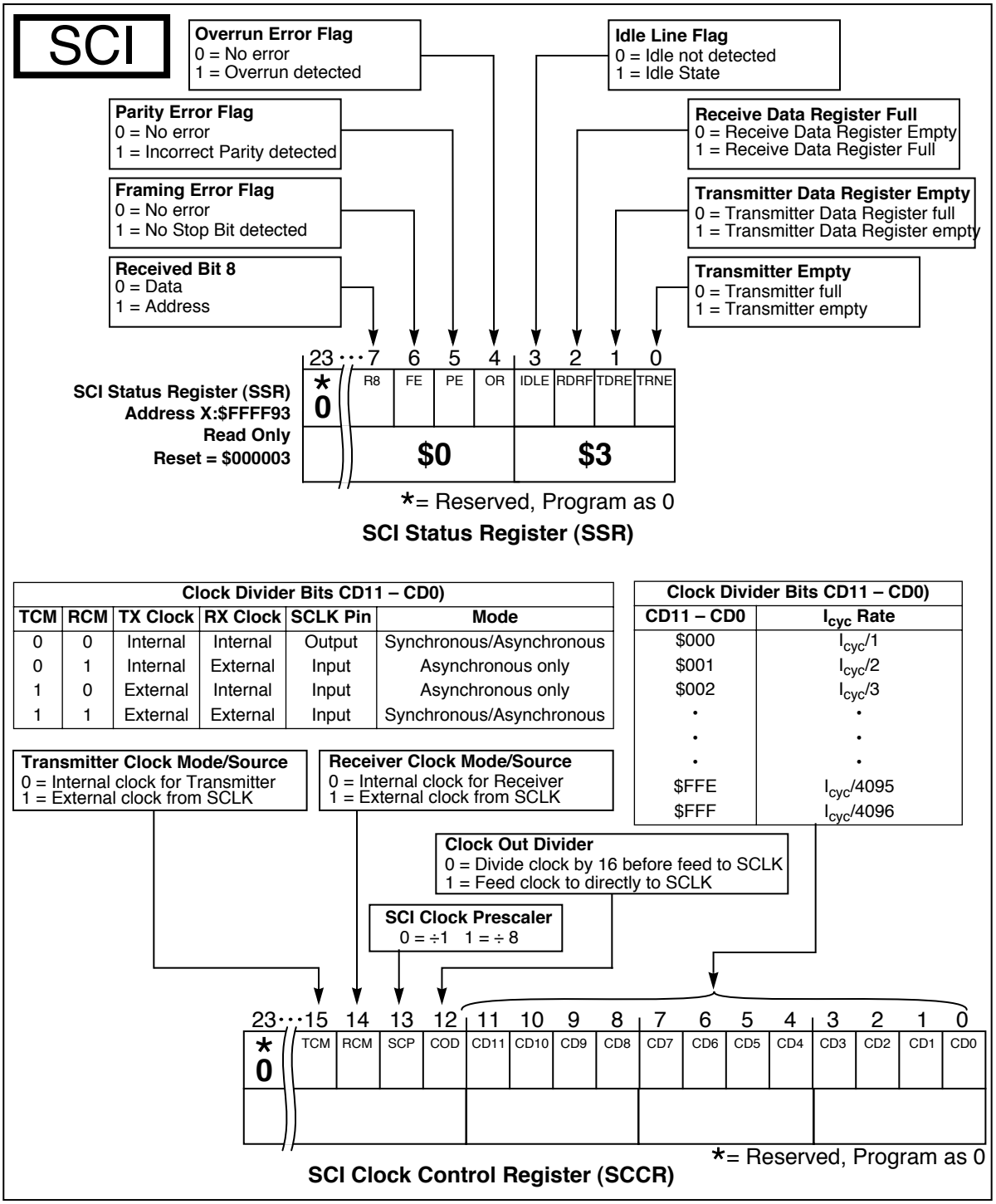


Figure D-17 SCI Status and Clock Control Registers (SSR, SCCR)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 3 of 3

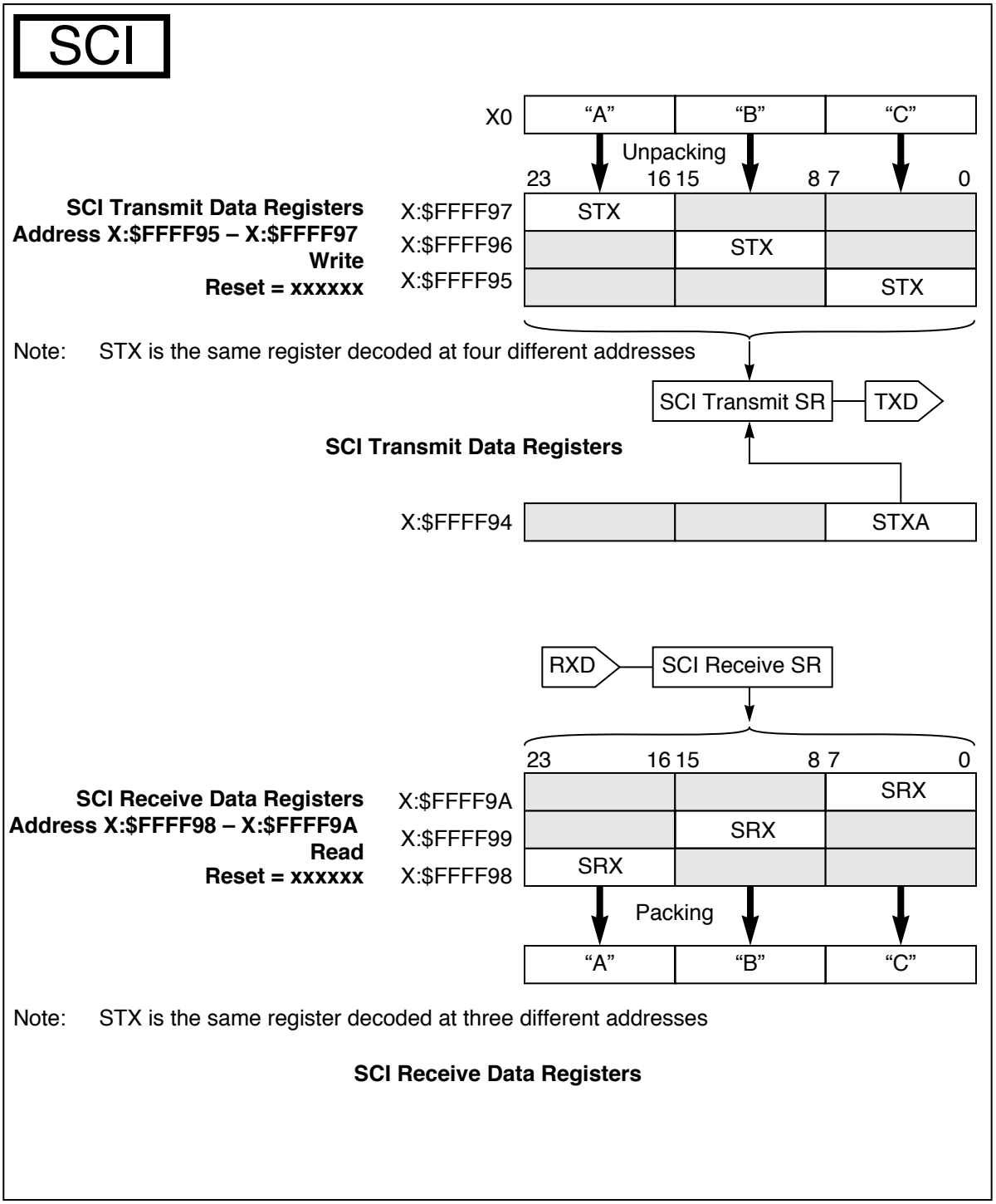


Figure D-18 SCI Receive and Transmit Data Registers (SRX, TRX)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 1 of 3

Timers

PS (1:0)	Prescaler Clock Source
00	Internal CLK/2
01	TIO0
10	TIO1
11	TIO2

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	PS1	PS0	Prescaler Preload Value (PL [0:20])																				

Timer Prescaler Load Register *= Reserved, Program as 0
TPLR:$FFFF83 Read/Write
Reset = $000000

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	Current Value of Prescaler Counter (PC [0:20])																				

Timer Prescaler Count Register *= Reserved, Program as 0
TPCR:$FFFF82 Read Only
Reset = $000000

Figure D-19 Timer Prescaler Load/Count Register (TPLR, TPCR)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 2 of 3

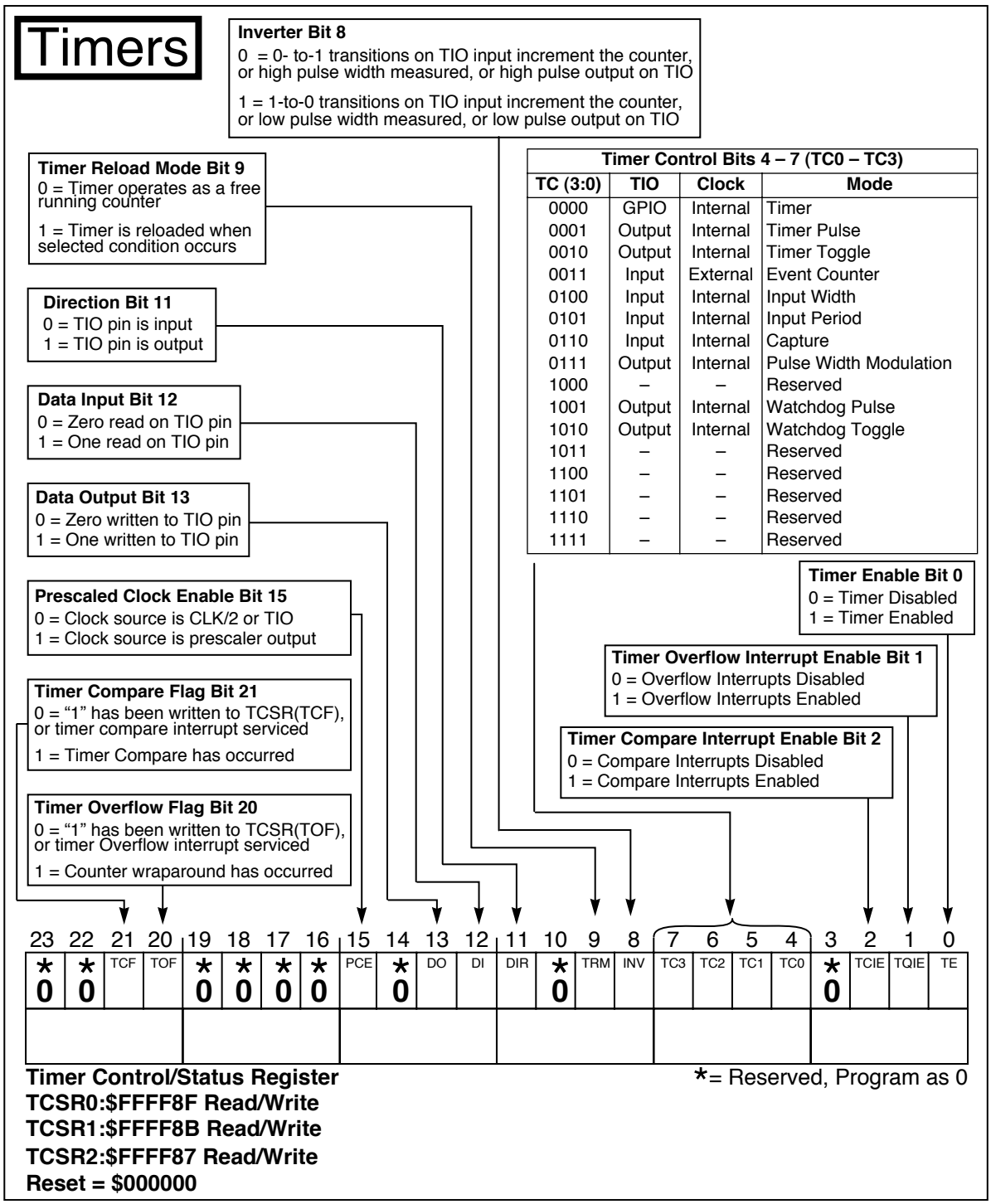


Figure D-20 Timer Control/Status Register (TCSR)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 3 of 3

Timers

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timer Reload Value																							

Timer Load Register
TLR0:$FFFF8E Write Only
TLR1:$FFFF8A Write Only
TLR2:$FFFF86 Write Only

Reset = $000000

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value Compared to Counter Value																							

Timer Compare Register
TCPR0:$FFFF8D Read/Write
TCPR1:$FFFF89 Read/Write
TCPR2:$FFFF85 Read/Write

Reset = $000000

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timer Count Value																							

Timer Count Register
TCR0:$FFFF8C Read Only
TCR1:$FFFF88 Read Only
TCR2:$FFFF84 Read Only

Reset = $000000

Figure D-21 Timer Load, Compare, Count Registers (TLR, TCPR, TCR)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 1 of 4

GPIO

Port B (HI08)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Host Data Direction Register (HDDR)	DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
X:$FFFFC8 Write																
Reset = $0																

DRx = 1 → HIx is Output DRx = 0 → HIx is Input

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Host Data Register (HDR)	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X:$FFFFC9 Write																
Reset = Undefined																

DRx holds value of corresponding HI08 GPIO pin.
Function depends on HDDR.

Figure D-22 Host Data Direction and Host Data Registers (HDDR, HDR)

Application:_____ Date:_____

_____ Programmer:_____

GPIO

Port C (ESSI0)

Port C Control Register (PCRC)
X:$FFFFBF
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0 * 0	PC5	PC4	PC3	PC2	PC1	PC0

PCn = 1 → Port Pin configured as ESSI
PCn = 0 → Port Pin configured as GPIO

Port C Direction Register (PRRC)
X:$FFFFBE
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0 * 0	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0

PDCn = 1 → Port Pin is Output
PDCn = 0 → Port Pin is Input

Port C GPIO Data Register (PDRC)
X:$FFFFBD
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0 * 0	PD5	PD4	PD3	PD2	PD1	PD0

port pin n is GPIO input, then PDn reflects the value on port pin n

if port pin n is GPIO output, then value written to PDn is reflected on port pin n

*= Reserved, Program as 0

Figure D-23 Port C Registers (PCRC, PRRC, PDRC)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 3 of 4

GPIO

Port D (ESSI1)

Port D Control Register (PCRD)
X:$FFFFAF
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0 * 0	PC5	PC4	PC3	PC2	PC1	PC0

PCn = 1 → Port Pin configured as ESSI
PCn = 0 → Port Pin configured as GPIO

Port D Direction Register (PRRD)
X:$FFFFAE
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0 * 0	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0

PDCn = 1 → Port Pin is Output
PDCn = 0 → Port Pin is Input

Port D GPIO Data Register (PDRD)
X:$FFFFAD
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0 * 0	PD5	PD4	PD3	PD2	PD1	PD0

port pin n is GPIO input, then PDn reflects the value on port pin n

if port pin n is GPIO output, then value written to PDn is reflected on port pin n

*= Reserved, Program as 0

Figure D-24 Port D Registers (PCRD, PRRD, PDRD)

Application:_____ Date:_____

_____ Programmer:_____

Sheet 4 of 4

GPIO

Port E (SCI)

Port E Control Register (PCRE)
X:$FFFF9F
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	PC2	PC1	PC0

PCn = 1 → Port Pin configured as SCI
PCn = 0 → Port Pin configured as GPIO

Port E Direction Register (PRRE)
X:$FFFF9E
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	PDC2	PDC1	PDC0

PDCn = 1 → Port Pin is Output
PDCn = 0 → Port Pin is Input

Port E GPIO Data Register (PDRE)
X:$FFFF9D
ReadWrite
Reset = $0

23 ··· 6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	PD2	PD1	PD0

port pin n is GPIO input, then PDn reflects the value on port pin n

if port pin n is GPIO output, then value written to PDn is reflected on port pin n

*= Reserved, Program as 0

Figure D-25 Port E Registers (PCRE, PRRE, PDRE)

INDEX

A

B

C

D

E

F

G

H

K

L

M

N

O

P

R

S

T