DSP56307

24-Bit Digital Signal Processor User's Manual

Motorola, Incorporated
Semiconductor Products Sector
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.motorola-dsp.com.

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.

OnCE™ is a trademark of Motorola, Inc.
Intel® is a registered trademark of the Intel Corporation.
All other trademarks are those of their respective owners.

®Reg. U.S. Pat. & Tm. Off.

Order this document by **DSP56307UM/D**.
Rev. 0, 08/10/98

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function, or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not authorized for use as components in life support devices or systems intended for surgical implant into the body or intended to support or sustain life. Buyer agrees to notify Motorola of any such intended end use whereupon Motorola shall determine availability and suitability of its product or products for the use intended. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity /Affirmative Action Employer.

TABLE OF CONTENTS

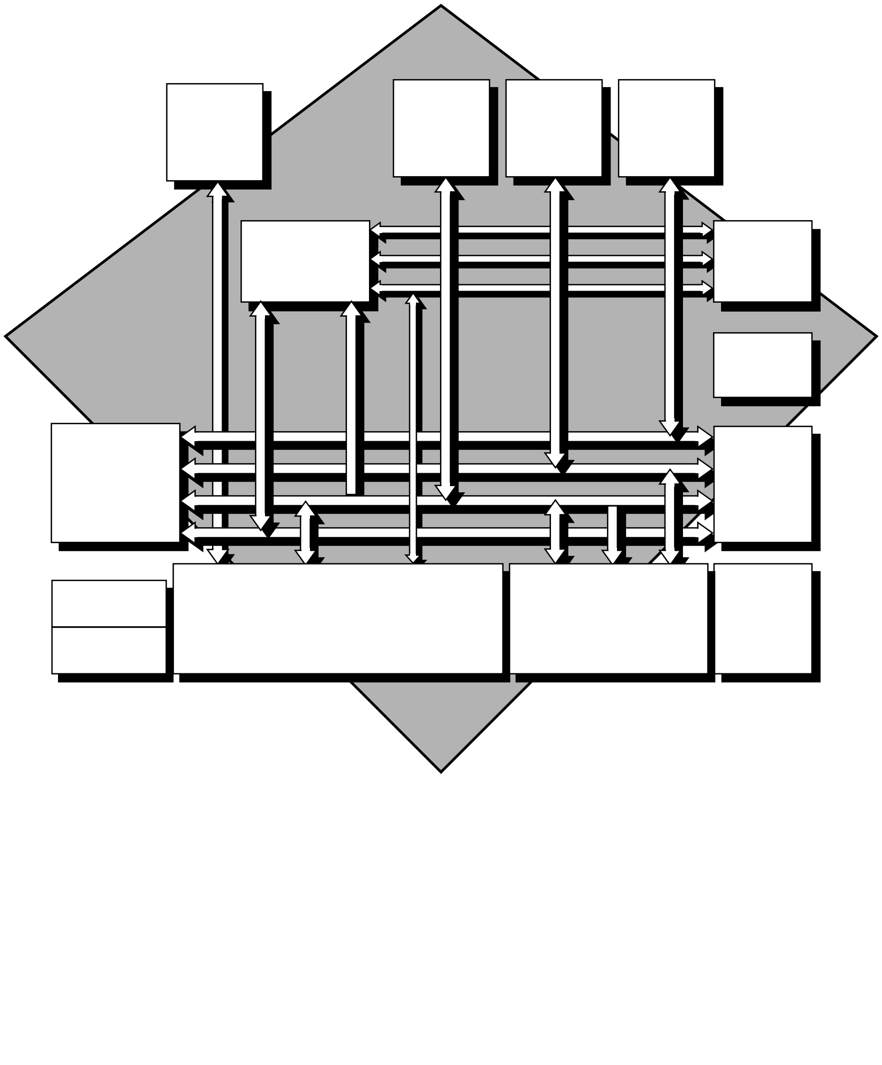
APPENDICES:

LIST OF FIGURES

LIST OF TABLES

SECTION 1

OVERVIEW



1.1 INTRODUCTION

This manual describes the DSP56307 24-bit digital signal processor (DSP), its memory, operating modes, and peripheral modules. The DSP56307 is an implementation of the DSP56300 core with a unique configuration of on-chip memory, cache, and peripherals.

This manual is to be used with the *DSP56300 Family Manual (DSP56300FM/AD)*, which describes the CPU, core programming models, and instruction set details. *DSP56307 Technical Data (DSP56307/D)*—referred to as the data sheet—provides electrical specifications, timing, pinout, and packaging descriptions of the DSP56307.

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 on this DSP, access the Motorola DSP home page at the address given on the back cover of this document.

1.2 MANUAL ORGANIZATION

This manual contains the following sections and appendices:

- **Section 1 Overview** provides a brief description of the DSP56307, including a features list and block diagram, lists related documentation needed to use this chip, and describes the organization of this manual.
- **Section 2 Signal/Connection Descriptions** describes the DSP56307 signals and their functional groupings.
- **Section 3 Memory Configuration** describes the DSP56307 memory spaces, RAM configuration, memory configuration bit settings, memory configurations, and memory maps.
- **Section 4 Core Configuration** describes the registers for configuring the DSP56300 core when programming the DSP56307, in particular the interrupt vector locations and the operation of the interrupt priority registers, and explains the operating modes and how they affect the processor's program and data memories.
- **Section 5 GeneraL-Purpose Input/Output** describes the DSP56307 general-purpose input/output (GPIO) capability and the programming model for the GPIO signals (operation, registers, and control).
- **Section 6 Host Interface (HI08)** describes the 8-bit host interface (HI08), including a quick reference to the HI08 programming model.

- **Section 7 Enhanced Synchronous Serial Interface** describes the 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) protocol.
- **Section 8 Serial Communication Interface** describes the 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** describes the three identical devices that may be used as internals or event counters.
- **Section 10 Enhanced Filter Coprocessor** describes the echo cancellation and filtering coprocessor.
- **Section 11 On-Chip Emulation Module** describes the On-Chip Emulation (OnCE™) module, which is accessed through the joint test action group (JTAG) port.
- **Section 12 Joint Test Action Group Port** describes the specifics of the JTAG port on the DSP56307.
- **Appendix Overview** lists the bootstrap code used for the DSP56307.
- **Appendix B Equates** lists the equates (I/O, HI08, SCI, ESSI, exception processing, timer, direct memory access (DMA), phase-locked loop (PLL), BIU, and interrupts for the DSP56307.
- **Appendix C DSP56307 BSDL Listing** provides the BSDL listing for the DSP56307.
- **Appendix D EFCOP Programming** describes the two types of digital filters supported by the enhanced filter coprocessor (EFCOP) and provides examples of how to use them.
- **Appendix E Programming Reference** lists peripheral addresses, interrupt addresses, and interrupt priorities for the DSP56307 and contains programming sheets listing the contents of the major DSP56307 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) are indicated like this in text:
  - In text, they have an overbar: for example, $\overline{\text{RESET}}$ is asserted low.
  - In code examples, they 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.
- "Input / Output" indicates a bidirectional signal. "Input or Output" indicates a signal that is exclusively one or the other.

- 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, as follows: $FFFFFF is the X memory address for the core interrupt priority register.
- 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 FEATURES

Motorola developed the DSP56307, a member of the DSP56300 core family of programmable DSPs, to support wireless infrastructure applications with general filtering operations. The on-chip EFCOP processes filter algorithms in parallel with core operation, thus increasing overall DSP performance and efficiency. Like the other family members, the DSP56307 uses a high-performance, single-clock-cycle-per-instruction engine (code compatible with Motorola's popular DSP56000 core family), a barrel shifter, 24-bit addressing, instruction cache, and DMA controller. The DSP56307 offers 100 million instructions per second (MIPS) performance using an internal 100 MHz clock with 2.5 V core and independent 3.3 V input/output (I/O) power.

All DSP56300 core family members contain the DSP56300 core and additional modules. The modules are chosen from a library of standard predesigned elements, such as memories and peripherals. New modules may 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. In particular, the DSP56307 includes Motorola’s JTAG port as well as Motorola’s OnCE module.

The DSP56307, with its large on-chip memory array of 64K words and its EFCOP, is well suited for high-end multichannel telecommunication applications, such as wireless infrastructure, multi-line voice/data/fax processing, video conferencing, and general digital signal processing.

1.5 CORE DESCRIPTION

Core features are described fully in the *DSP56300 Family Manual*. (This manual, in contrast, documents pinout, memory, and peripheral features.)

- 100 MIPS with a 100 MHz clock at 3.3 V
- Object code compatible with the DSP56000 core
- Highly parallel instruction set
- Large on-chip RAM memory of 64K words
- EFCOP running concurrently with the core, capable of executing 100 million filter taps per second at peak performance
- Hardware debugging support
  - JTAG test access port (TAP)
  - OnCE module
  - Address trace mode reflects internal accesses at the external port
- 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 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 arithmetic logic unit (ALU)
- Address generation unit

- Program control unit
- PLL and clock oscillator
- JTAG TAP and OnCE module
- Memory

In addition, the DSP56307 provides a set of on-chip peripherals, shown in **Section 1.8**.

1.6.1 Data ALU

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

- Fully pipelined 24 × 24-bit parallel multiplier-accumulator
- Bit field unit, comprising a 56-bit parallel barrel shifter (fast shift and normalization; bit stream generation and parsing)
- Conditional ALU instructions
- Software-controllable 24-bit or 16-bit arithmetic support
- 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 and the Y data bus 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 cycle, 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 immediate 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. In the case of 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 two's-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 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 work in parallel and share common inputs. The only difference between them is that the 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 own 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
- Program address generator
- Program interrupt controller

The program decode controller decodes the 24-bit instruction loaded into the instruction latch and generates all signals necessary for pipeline control. The program address generator contains all the hardware needed for program address generation, system stack, and loop control. The program interrupt controller 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 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:

- Program counter register
- Status register
- Loop address register
- Loop counter register
- Vector base address register
- Size register
- Stack pointer
- Operating mode register
- Stack counter register

The PCU also includes a hardware system stack.

1.6.4 PLL and Clock Oscillator

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

These features allow you to do the following:

- Change the low-power divide factor without losing the lock.
- Output a 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 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 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 about the JTAG port is provided in **Section 12 Joint Test Action Group Port** on page 12-1**.**

The 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. This facilitates hardware and software development on the DSP56300 core processor. OnCE module functions are provided through the JTAG TAP signals. More information on the OnCE module is provided in **Section 11 On-Chip Emulation Module** on page 11-1**.**

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** on page 3-1. Program RAM, instruction cache, X data RAM, and Y data RAM size are programmable, as shown in **Table 1-2**:

Table 1-2 Available Memory Configurations

Program RAM Size	Instruction Cache Size	X Data RAM Size*	Y Data RAM Size*	Instruction Cache	Switch Mode	M1	M0
16K × 24-bit	0	24K × 24-bit	24K × 24-bit	disabled	disabled	0/1	0/1
15K × 24-bit	1024 × 24-bit	24K × 24-bit	24K × 24-bit	enabled	disabled	0/1	0/1
48K × 24-bit	0	8K × 24-bit	8K × 24-bit	disabled	enabled	0	0
47K × 24-bit	1024 × 24-bit	8K × 24-bit	8K × 24-bit	enabled	enabled	0	0
40K × 24-bit	0	12K × 24-bit	12K × 24-bit	disabled	enabled	0	1
39K × 24-bit	1024 × 24-bit	12K × 24-bit	12K × 24-bit	enabled	enabled	0	1
32K × 24-bit	0	16K × 24-bit	16K × 24-bit	disabled	enabled	1	0
31K × 24-bit	1024 × 24-bit	16K × 24-bit	16K × 24-bit	enabled	enabled	1	0
24K × 24-bit	0	20K × 24-bit	20K × 24-bit	disabled	enabled	1	1
23K × 24-bit	1024 × 24-bit	20K × 24-bit	20K × 24-bit	enabled	enabled	1	1

*Includes 4K × 24-bit shared memory (i.e., memory shared by the core and the EFCOP and not accessible by the DMA controller)

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 the following capacities:

- Data memory expansion to two 16 M × 24-bit word memory spaces in 24-bit address mode (64K in 16-bit address mode)
- Program memory expansion to one 16 M × 24-bit word memory space in 24-bit address mode (64K in 16-bit address mode)

Further features of off-chip memory include the following:

- External memory expansion port

- Simultaneous glueless interface to static RAM (SRAM) and dynamic RAM (DRAM)

1.7 INTERNAL BUSES

To provide data exchange between these blocks, the DSP56307 implements the following buses:

- Peripheral I/O expansion bus to peripherals
- Program memory expansion bus to program ROM
- X memory expansion bus to X memory
- Y memory expansion bus to Y memory
- Global data bus between PCU and other core structures
- Program data bus for carrying program data throughout the core
- X memory data bus for carrying X data throughout the core
- Y memory data bus for carrying Y data throughout the core
- Program address bus for carrying program memory addresses throughout the core
- X memory address bus for carrying X memory addresses throughout the core
- Y memory address bus for carrying Y memory addresses throughout the core.

The block diagram in **Figure 1-1** illustrates those buses among other components.

1.8 BLOCK DIAGRAM

With the exception of the program data bus, all internal buses on the DSP56300 family members are 24-bit buses. The program data bus is also a 24-bit bus. **Figure 1-1** shows a block diagram of the DSP56307.

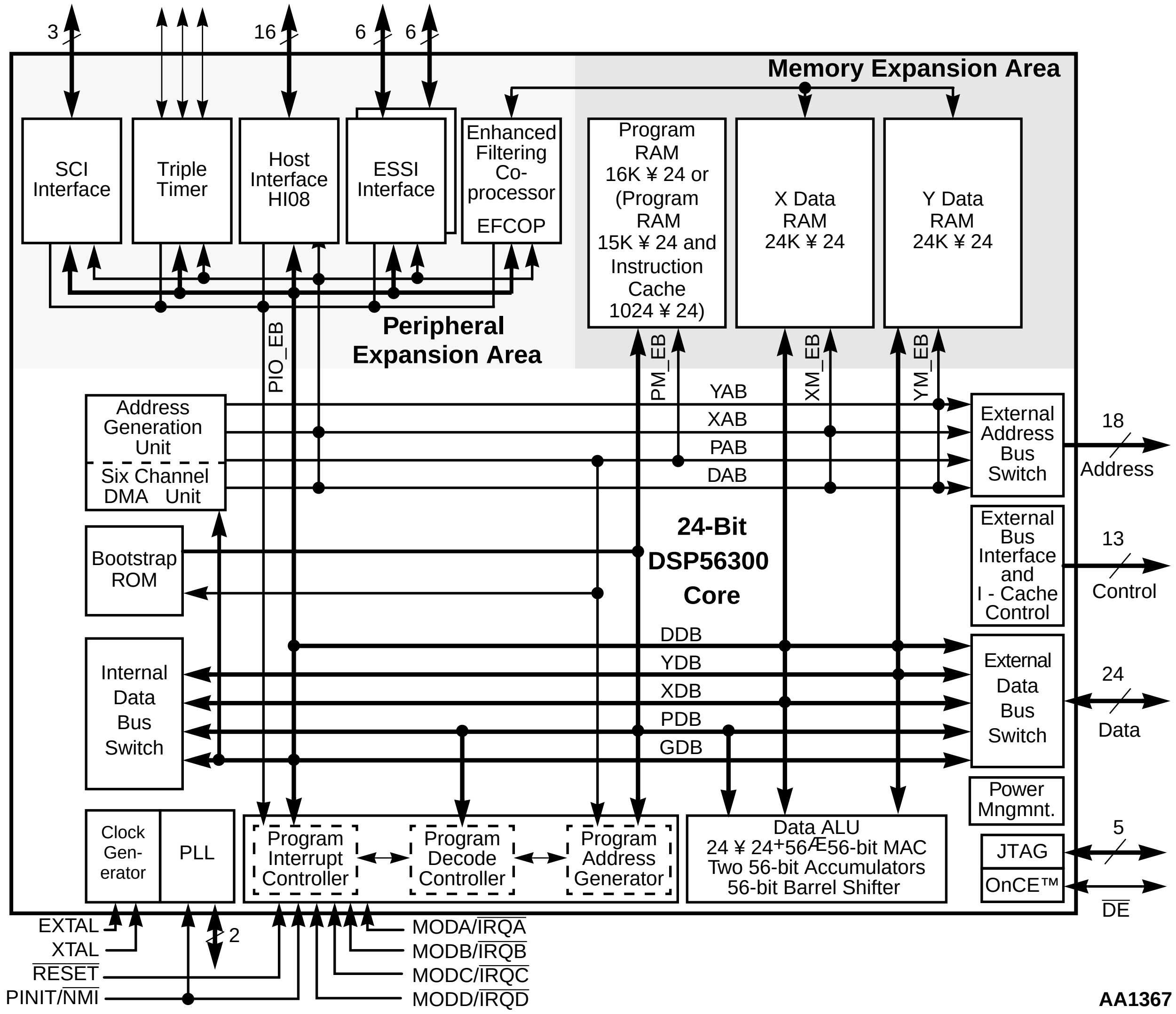


Figure 1-1 DSP56307 Block Diagram

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

1.9 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 and all peripherals

1.10 ARCHITECTURE

Motorola designed the DSP56307 to perform a wide variety of fixed-point digital signal processing functions. In addition to the core features previously discussed, the DSP56307 provides the following peripherals:

- As many as 34 user-configurable GPIO signals
- HI08 to external hosts
- Dual ESSI
- SCI
- Triple timer module
- Memory switch mode
- Four external interrupt/mode control lines

1.10.1 GPIO Functionality

The GPIO port consists of as many as 34 programmable signals, all of which are also used by the peripherals (HI08, ESSI, SCI, and timer). There are no dedicated GPIO signals. After a reset, the signals are automatically configured as GPIO. Three memory-mapped registers per peripheral control GPIO functionality. The techniques to use when programming these registers to control GPIO functionality are detailed in **Section 5 GeneraL-Purpose Input/Output** .

1.10.2 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 provides connection with a number of industry-standard DSPs, microcomputers, and microprocessors without requiring any 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 you to program DSP core communication with the HI08 registers by using standard instructions and addressing modes.

1.10.3 ESSI

The DSP56307 provides two independent and identical ESSI. Each ESSI provides 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

1.10.4 SCI

The DSP56307'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, receive data, and SCI serial clock. It supports industry-standard asynchronous bit rates and protocols, as well as high-speed synchronous data transmission (up to 12.5 Mbps for a 100 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 DSP56307 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 one having 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 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.

1.10.6 EFCOP

The EFCOP is a peripheral block interfacing with the DSP core via the peripheral module bus. The EFCOP is a general-purpose, fully programmable coprocessor designed to perform filtering tasks concurrently with the DSP core with minimum core overhead. The DSP core and the EFCOP can share data by means of an 8K-word shared data memory. DMA channels shuttle input and output data between the DSP core and the EFCOP.

The EFCOP supports a variety of filter modes, some of which are optimized for cellular base station applications:

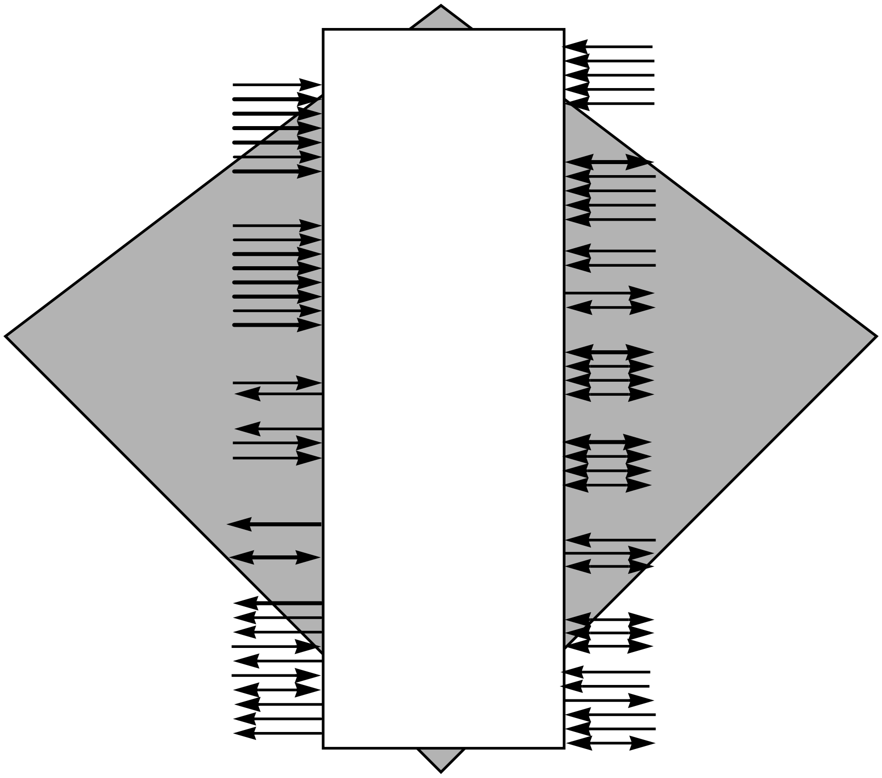
- Real finite impulse response (FIR) with real taps
- Complex FIR with complex taps
- Complex FIR generating pure real or pure imaginary outputs alternately
- A 4-bit decimation factor in FIR filters, thus providing a decimation ratio up to 16
- Direct form 1 (DFI) infinite impulse response (IIR) filter
- Direct form 2 (DFII) IIR filter
- Four scaling factors (1, 4, 8, 16) for IIR output
- Adaptive FIR filter with true least mean square (LMS) coefficient updates
- Adaptive FIR filter with delayed LMS coefficient updates

The EFCOP supports up to 4K taps and 4K coefficients in any combination of number and length of filters (e.g., eight filters of length 512, or 16 filters of length 256). It can perform either 24-bit or 16-bit precision arithmetic with full support for saturation arithmetic.

The EFCOP is a cost-effective and power-efficient coprocessor that accelerates filtering tasks, such as echo cancellation or correlation, concurrently with software running on the DSP core.

SECTION 2

SIGNAL/CONNECTION DESCRIPTIONS



2.1 SIGNAL GROUPINGS

The DSP56307 input and output signals are organized into functional groups, as shown in **Table 2-1** and as illustrated in **Figure 2-1**.

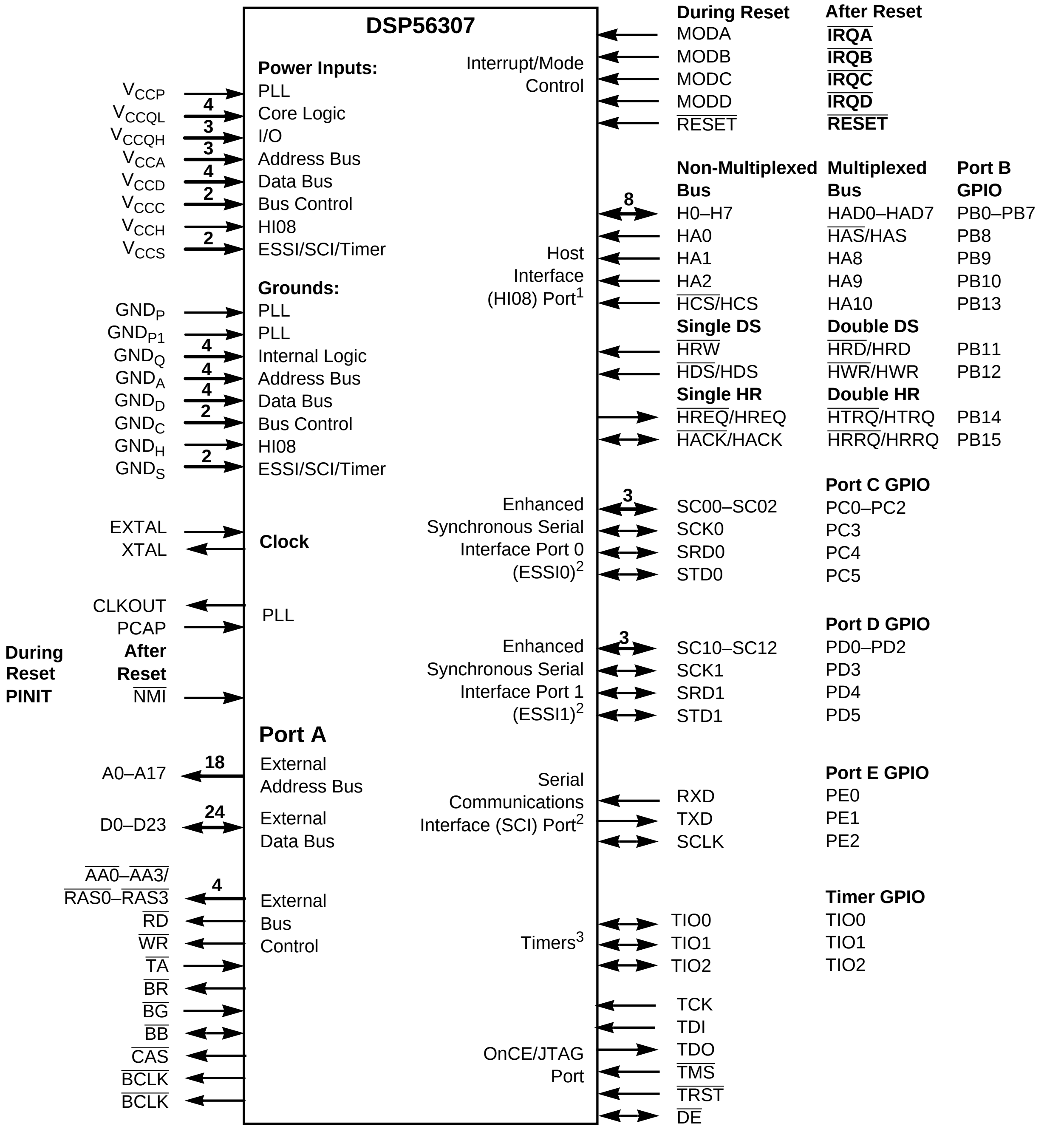
Table 2-1 DSP56307 Functional Signal Groupings

Functional Group		Number of Signals	Detailed Description
Power (V_{CC})		20	**Figure 2-2**
Ground (GND)		19	**Figure 2-3**
Clock		2	**Figure 2-4**
PLL		3	**Figure 2-5**
Address bus	Port A[1]	18	**Figure 2-6**
Data bus		24	**Figure 2-7**
Bus control		13	**Figure 2-8**
Interrupt and mode control		5	**Figure 2-9**
HI08	Port B[2]	16	**Figure 2-10**
ESSI	Ports C and D[3]	12	**Figure 2-11** and **Figure 2-12**
SCI	Port E[4]	3	**Figure 2-13**
Timer[5]		3	**Figure 2-14**
OnCE/JTAG Port		6	**Figure 2-15**

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

Figure 2-1 is a diagram of DSP56307 signals by functional group.

Signal Groupings



Note:

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., $\overline{\text{HAS}}$/HAS) have configurable polarity.
2. The ESSI0, ESSI1, and SCI signals are multiplexed: ESSI0 with the Port C GPIO signals (PC0–PC5), ESSI1 with Port D GPIO signals (PD0–PD5), and SCI with Port E GPIO signals (PE0–PE2).
3. TIO0–TIO2 can be configured as GPIO signals.

AA1502

Figure 2-1 Signals Identified by Functional Group

2.2 POWER

Table 2-2 Power Inputs

Power Name	Description
V_{CCP}	**PLL Power**—V_{CCP} is V_{CC} dedicated for 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_{CCQL}	**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.
V_{CCQH}	**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 decoupling capacitors.
V_{CCA}	**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.
V_{CCD}	**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, ***except*** V_{CCQL}. The user must provide adequate external decoupling capacitors.
V_{CCC}	**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.
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.
V_{CCS}	**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.

2.3 GROUND

Table 2-3 Grounds

Ground Name	Description
GND_P	**PLL Ground**—GND_P is 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.
GND_{P1}	**PLL Ground 1**—GND_{P1} is ground-dedicated for PLL use. The connection should be provided with an extremely low-impedance path to ground.
GND_Q	**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.
GND_A	**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.
GND_D	**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.
GND_C	**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.
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.
GND_S	**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.

2.4 CLOCK

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.
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 PLL

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 may be tied to V_{CC}, 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	Input	Input	**PLL Initial**—During assertion of $\overline{\text{RESET}}$, the value of PINIT is written into the PLL enable (PEN) bit of the PLL control (PCTL) register, determining whether the PLL is enabled or disabled.
$\overline{\text{NMI}}$	Input		**Nonmaskable Interrupt**—After $\overline{\text{RESET}}$ deassertion and during normal instruction processing, this Schmitt-trigger input is the negative-edge-triggered NMI request internally synchronized to CLKOUT.

2.6 EXTERNAL MEMORY EXPANSION PORT (PORT A)

Note: When the DSP56307 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

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

Table 2-7 External Data Bus Signals

Signal Name	Type	State During Reset	Signal Description
D0–D23	Input/ Output	Tri-stated	**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 tri-stated. These lines have weak keepers to maintain the last state even if all drivers are tri-stated.

2.6.3 External Bus Control

Table 2-8 External Bus Control Signals

Signal Name	Type	State During Reset	Signal Description
AA0–AA3	Output	Tri-stated	**Address Attribute**—When defined as AA, these signals can be used as chip selects or additional address lines. The default use defines a priority scheme under which only one AA signal can be asserted at a time. Setting the AA priority disable (APD) bit (Bit 14) of the OMR, the priority mechanism is disabled and the lines can be used together as four external lines that can be decoded externally into 16 chip select signals.
$\overline{\text{RAS0}}$–$\overline{\text{RAS3}}$	Output		**Row Address Strobe**—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
$\overline{\text{TA}}$	Input	Ignored Input	**Transfer Acknowledge**—If the DSP56307 is the bus master and there is no external bus activity, or the DSP56307 is not the bus master, the $\overline{\text{TA}}$ input is ignored. The $\overline{\text{TA}}$ input is a data transfer acknowledge (DTACK) function that can extend an external bus cycle indefinitely. Any number of wait states (1, 2. . .infinity) may be added to the wait states inserted by the bus control register (BCR) by keeping $\overline{\text{TA}}$ deasserted. In typical operation, $\overline{\text{TA}}$ 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 $\overline{\text{TA}}$ is asserted synchronous to CLKOUT. The number of wait states is determined by the $\overline{\text{TA}}$ 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 $\overline{\text{TA}}$ functionality, the BCR must be programmed to at least one wait state. A zero wait state access cannot be extended by $\overline{\text{TA}}$ deassertion; otherwise, improper operation may result. $\overline{\text{TA}}$ can operate synchronously or asynchronously depending on the setting of the TAS bit in the OMR. $\overline{\text{TA}}$ functionality may not be used while performing DRAM type accesses; otherwise, improper operation may result.

Table 2-8 External Bus Control Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{\text{BR}}$	Output	Output (deasserted)	**Bus Request**—$\overline{\text{BR}}$ is an active-low output, never tri-stated. $\overline{\text{BR}}$ is asserted when the DSP requests bus mastership. $\overline{\text{BR}}$ is deasserted when the DSP no longer needs the bus. $\overline{\text{BR}}$ may be asserted or deasserted independently of whether the DSP56307 is a bus master or a bus slave. Bus “parking” allows $\overline{\text{BR}}$ to be deasserted even though the DSP56307 is the bus master. (See the description of bus “parking” in the $\overline{\text{BB}}$ signal description.) The bus request hole (BRH) bit in the BCR allows $\overline{\text{BR}}$ to be asserted under software control even though the DSP does not need the bus. $\overline{\text{BR}}$ is typically sent to an external bus arbitrator that controls the priority, parking, and tenure of each master on the same external bus. $\overline{\text{BR}}$ is only affected by DSP requests for the external bus, never for the internal bus. During hardware reset, $\overline{\text{BR}}$ is deasserted and the arbitration is reset to the bus slave state.
$\overline{\text{BG}}$	Input	Ignored Input	**Bus Grant**—$\overline{\text{BG}}$ is an active-low input. $\overline{\text{BG}}$ must be asserted/deasserted synchronous to CLKOUT for proper operation. $\overline{\text{BG}}$ is asserted by an external bus arbitration circuit when the DSP56307 becomes the next bus master. When $\overline{\text{BG}}$ is asserted, the DSP56307 must wait until $\overline{\text{BB}}$ is deasserted before taking bus mastership. When $\overline{\text{BG}}$ is deasserted, bus mastership is typically given up at the end of the current bus cycle. This may occur in the middle of an instruction that requires more than one external bus cycle for execution. The default operation of this bit requires a setup and hold time as specified in *DSP56307 Technical Data* (the data sheet). An alternate mode can be invoked: set the asynchronous bus arbitration enable (ABE) bit (Bit 13) in the OMR. When this bit is set, $\overline{\text{BG}}$ and $\overline{\text{BB}}$ are synchronized internally. This eliminates the respective setup and hold time requirements but adds a required delay between the deassertion of an initial $\overline{\text{BG}}$ input and the assertion of a subsequent $\overline{\text{BG}}$ input.

Table 2-8 External Bus Control Signals (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{\text{BB}}$	Input/ Output	Input	**Bus Busy**—$\overline{\text{BB}}$ is a bidirectional active-low input/output and must be asserted and deasserted synchronous to CLKOUT. $\overline{\text{BB}}$ indicates that the bus is active. Only after $\overline{\text{BB}}$ is deasserted can the pending bus master become the bus master (and then assert the signal again). The bus master may keep $\overline{\text{BB}}$ asserted after ceasing bus activity regardless of whether $\overline{\text{BR}}$ is asserted or deasserted. Called "bus parking," this allows the current bus master to reuse the bus without rearbitration until another device requires the bus. The deassertion of $\overline{\text{BB}}$ is done by an "active pull-up" method (i.e., $\overline{\text{BB}}$ is driven high and then released and held high by an external pull-up resistor). The default operation of this bit requires a setup and hold time as specified in the *DSP56307 Technical Data sheet*. An alternate mode can be invoked: set the ABE bit (Bit 13) in the OMR. When this bit is set, $\overline{\text{BG}}$ and $\overline{\text{BB}}$ are synchronized internally. See $\overline{\text{BG}}$ for additional information. $\overline{\text{BB}}$ requires an external pull-up resistor.
$\overline{\text{CAS}}$	Output	Tri-stated	**Column Address Strobe**—When the DSP is the bus master, $\overline{\text{CAS}}$ 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 (i.e., the ATE bit in the OMR is set). When BCLK is active and synchronized to CLKOUT by the internal PLL, BCLK precedes CLKOUT by one-fourth of a clock cycle. The BCLK rising edge may be used to sample the internal program memory access on the A0–A23 address lines.
$\overline{\text{BCLK}}$	Output	Tri-stated	**Bus Clock Not**—When the DSP is the bus master, $\overline{\text{BCLK}}$ 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 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	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.
$\overline{\text{IRQA}}$	Input		**External Interrupt Request A**—After reset, this input 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 will exit 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 the OMR when the $\overline{\text{RESET}}$ signal is deasserted.
$\overline{\text{IRQB}}$	Input		**External Interrupt Request B**—After reset, this input 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. If the processor is in the stop standby state and $\overline{\text{IRQB}}$ is asserted, the processor will exit the stop 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 the OMR when the $\overline{\text{RESET}}$ signal is deasserted.
$\overline{\text{IRQC}}$	Input		**External Interrupt Request C**—After reset, this input 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. If the processor is in the stop standby state and $\overline{\text{IRQC}}$ is asserted, the processor will exit the stop 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 the OMR when the $\overline{\text{RESET}}$ signal is deasserted.
$\overline{\text{IRQD}}$	Input		**External Interrupt Request D**—After reset, this input 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. If the processor is in the stop standby state and $\overline{\text{IRQD}}$ is asserted, the processor will exit the stop state.

2.8 HI08

The HI08 provides a fast parallel-data-to-8-bit port that may be connected 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.

Table 2-10 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 nonmultiplexed 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 host port control register (HPCR), these signals are individually programmed as inputs or outputs through the HI08 data direction register (HDDR). Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-10 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
HA0	Input	Input	**Host Address Input 0**—When the HI08 is programmed to interface a nonmultiplexed 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
HA1	Input	Input	**Host Address Input 1**—When the HI08 is programmed to interface a nonmultiplexed 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-10 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
HA2	Input	Input	**Host Address Input 2**—When the HI08 is programmed to interface a nonmultiplexed 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
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 HRD strobe 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-10 Host Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{\text{HDS}}$/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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
HCS	Input	Input	**Host Chip Select**—When HI08 is programmed to interface a nonmultiplexed 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 ($\overline{\text{HCS}}$) 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-10 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 may 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 may 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-10 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 may 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

2.9 ENHANCED SYNCHRONOUS SERIAL INTERFACE 0

There are two synchronous serial interfaces (ESSI0 and ESSI1) that 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 serial peripheral interface (SPI).

Table 2-11 Enhanced Synchronous Serial Interface 0

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 will be 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.
PC0			**Port C 0**—The default configuration following reset is GPIO input PC0. When configured as PC0, signal direction is controlled through the port directions register (PRR0). The signal can be configured as ESSI signal SC00 through the port control register (PCR0). Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
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.
PC1	Input or Output		**Port C 1**—The default configuration following reset is GPIO input PC1. When configured as PC1, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SC01 through PCR0. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-11 Enhanced Synchronous Serial Interface 0 (Continued)

Signal Name	Type	State During Reset	Signal Description
SC02 PC2	Input/ Output Input or 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). **Port C 2**—The default configuration following reset is GPIO input PC2. When configured as PC2, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SC02 through PCR0. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
SCK0 PC3	Input/ Output Input or Output	Input	**Serial Clock**—SCK0 is a bidirectional Schmitt-trigger input signal providing the serial bit rate clock for the ESSI. 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 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. **Port C 3**—The default configuration following reset is GPIO input PC3. When configured as PC3, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SCK0 through PCR0. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-11 Enhanced Synchronous Serial Interface 0 (Continued)

Signal Name	Type	State During Reset	Signal Description
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.
PC4	Input or Output		**Port C 4**—The default configuration following reset is GPIO input PC4. When configured as PC4, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal SRD0 through PCR0. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
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.
PC5	Input or Output		**Port C 5**—The default configuration following reset is GPIO input PC5. When configured as PC5, signal direction is controlled through PRR0. The signal can be configured as an ESSI signal STD0 through PCR0. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

2.10 ENHANCED SYNCHRONOUS SERIAL INTERFACE 1

Table 2-12 Enhanced Serial Synchronous Interface 1

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 will be 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.
PD0	Input or Output		**Port D 0**—The default configuration following reset is GPIO input PD0. When configured as PD0, signal direction is controlled through the port directions register (PRR1). The signal can be configured as an ESSI signal SC10 through the port control register (PCR1). Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
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.
PD1	Input or Output		**Port D 1**—The default configuration following reset is GPIO input PD1. When configured as PD1, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SC11 through PCR1. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-12 Enhanced Serial Synchronous Interface 1 (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 (and the receiver in synchronous operation).
PD2	Input or Output		**Port D 2**—The default configuration following reset is GPIO input PD2. When configured as PD2, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SC12 through PCR1. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
SCK1	Input/ Output	Input	**Serial Clock**—SCK1 is a bidirectional Schmitt-trigger input signal providing the serial bit rate clock for the ESSI. 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.
PD3	Input or Output		**Port D 3**—The default configuration following reset is GPIO input PD3. When configured as PD3, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SCK1 through PCR1. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-12 Enhanced Serial Synchronous Interface 1 (Continued)

Signal Name	Type	State During Reset	Signal Description
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.
PD4	Input or Output		**Port D 4**—The default configuration following reset is GPIO input PD4. When configured as PD4, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal SRD1 through PCR1. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
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.
PD5	Input or Output		**Port D 5**—The default configuration following reset is GPIO input PD5. When configured as PD5, signal direction is controlled through PRR1. The signal can be configured as an ESSI signal STD1 through PCR1. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

2.11 SCI

The SCI provides a full duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems.

Table 2-13 Serial Communication Interface

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.
PE0	Input or Output		**Port E 0—**The default configuration following reset is GPIO input PE0. When configured as PE0, signal direction is controlled through the SCI port directions register (PRR). The signal can be configured as an SCI signal RXD through the SCI port control register (PCR). Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
TXD	Output	Input	**Serial Transmit Data—**This signal transmits data from SCI transmit data register.
PE1	Input or Output		**Port E 1—**The default configuration following reset is GPIO input PE1. When 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-13 Serial Communication Interface (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.
PE2	Input or Output		**Port E 2**—The default configuration following reset is GPIO input PE2. When 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. Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

2.12 TIMERS

Three identical and independent timers are implemented in the DSP56307. Each timer can use internal or external clocking and can either interrupt the DSP56307 after a specified number of events (clocks) or signal an external device after counting a specific number of internal events.

Table 2-14 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. The default mode after reset is GPIO input. This can be changed to output or configured as a timer I/O through the timer 0 control/status register (TCSR0). Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.
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. The default mode after reset is GPIO input. This can be changed to output or configured as a timer I/O through the timer 1 control/status register (TCSR1). Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

Table 2-14 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. The default mode after reset is GPIO input. This can be changed to output or configured as a timer I/O through the timer 2 control/status register (TCSR2). Note: This signal has a weak keeper to maintain the last state even if all drivers are tri-stated.

2.13 JTAG AND OnCE INTERFACE

The DSP56300 family and in particular the DSP56307 support circuit-board test strategies based on the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*, the industry standard developed under the sponsorship of the Test Technology Committee of IEEE and the JTAG.

The OnCE module interfaces nonintrusively with the DSP56300 core and its peripherals so that you can examine registers, memory, or on-chip peripherals. Functions of the OnCE module are provided through the JTAG TAP signals.

For programming models, see **Section 12 Joint Test Action Group Port** and **Section 11 On-Chip Emulation Module** .

Table 2-15 OnCE/JTAG Interface

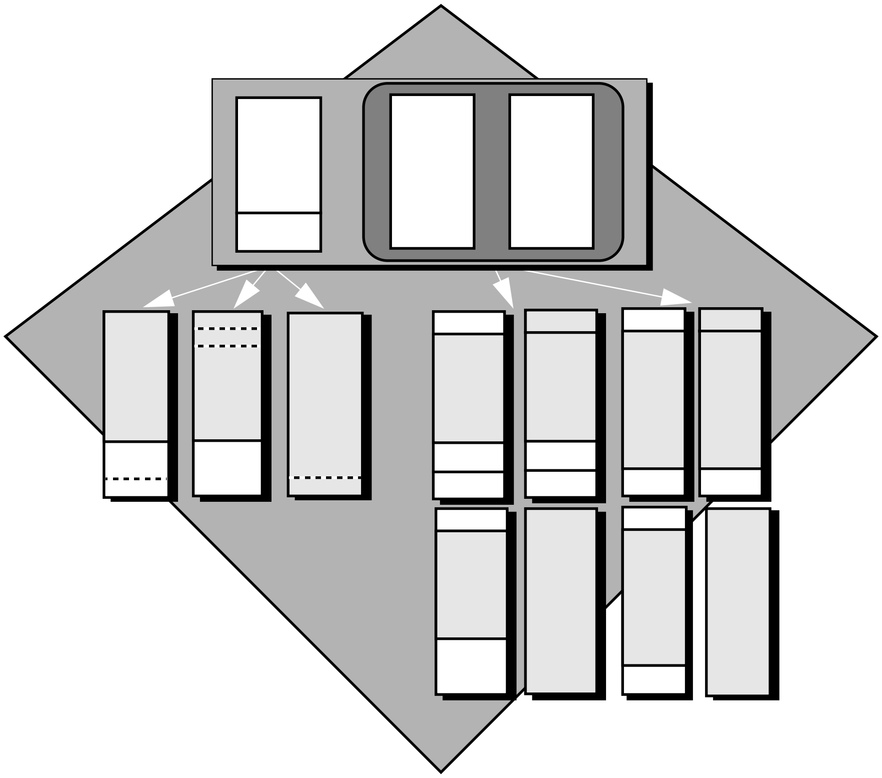
Signal Name	Type	State During Reset	Signal Description
TCK	Input	Input	**Test Clock**—TCK is a test clock input signal used to synchronize the JTAG test logic.
TDI	Input	Input	**Test Data Input**—TDI is a test data serial input signal used for test instructions and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
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 2-15 OnCE/JTAG Interface (Continued)

Signal Name	Type	State During Reset	Signal Description
$\overline{\mathrm{DE}}$	Input/ Output	Input	**Debug Event**—$\overline{\mathrm{DE}}$ is an open-drain, bidirectional, active-low signal that provides, as an input, a means of entering the debug mode of operation from an external command controller, and, as an output, a means of acknowledging that the chip has entered the 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 the 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 the debug mode as a result of a debug request or as a result of meeting a breakpoint condition. The $\overline{\mathrm{DE}}$ 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 the debug mode. All other interface with the OnCE module must occur through the JTAG port.

SECTION 3

MEMORY CONFIGURATION



3.1 INTRODUCTION

Like all members of the DSP56300 core family, the DSP56307 can address three sets of 16 M × 24-bit memory internally: program, X data, and Y data. Each of these memory spaces can include both on-chip and external memory (accessed through the external memory interface). The DSP56307 is extremely flexible because it has several modes to allocate on-chip memory between the program memory and the two data memory spaces. You can also configure it to operate in a special sixteen-bit compatibility mode that allows the chip to use DSP56000 object code without any change; this can result in higher performance of existing code for applications that do not require a larger address space. This section provides detailed information about each of these memory spaces.

See the *DSP56300 Family Manual* for general information about memory.

3.2 PROGRAM MEMORY SPACE

Program memory space consists of the following:

- Internal program memory (program RAM, 16K by default, up to 40K)
- (Optional) instruction cache (1K) formed from program RAM
- Bootstrap program ROM (192 x 24-bit)
- (Optional) Off-chip memory expansion (as much as 64K in 16-bit mode or 256K in 24-bit mode using the 18 external address lines or 4 M using the external address lines and the four address attribute lines)

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

3.2.1 Internal Program Memory

The default on-chip program memory consists of a 24-bit-wide, high-speed, SRAM occupying the lowest 16K locations ($0–$3FFF) in program memory space. The on-chip program RAM is organized in 64 banks with 256 locations each. You can make additional program memory available using the memory switch mode described in **Section 3.2.2Memory Switch Modes—Program Memory** .

3.2.2 Memory Switch Modes—Program Memory

Memory switch mode allows reallocation of portions of X and Y data RAM as program RAM. Bit 7 in the OMR is the memory switch (MS) bit that controls this function, as follows:

- When the MS bit is cleared, program memory consists of the default 16K × 24-bit memory space described in the previous section. In this default mode, the lowest external program memory location is $4000.
- When the MS bit is set, a portion of the higher locations of the internal X and Y data memory are switched to internal program memory. The memory switch configuration (MSW[1:0]) bits (also called M1 and M0) in the OMR select one of the following options:
  - **MSW[1:0] = 00**—The 16K higher locations ($2000–$5FFF) of the internal X data memory and the 16K higher locations ($2000–$5FFF) of the internal Y data memory are switched to internal program memory. In such a case, the on-chip program memory occupies the lowest 48K locations ($0–$BFFF) in the program memory space. The instruction cache, if enabled, occupies the lowest 1K program words (locations $0–$3FF). The lowest external program memory location in this mode is $C000.
  - **MSW[1:0] = 01**—The 12K higher locations ($3000–$5FFF) of the internal X data memory and the 12K higher locations ($3000–$5FFF) of the internal Y data memory are switched to internal program memory. In such a case, the on-chip program memory occupies the lowest 40K locations ($0–$9FFF) in the program memory space. The instruction cache, if enabled, occupies the lowest 1K program words (locations $0–$3FF). The lowest external program memory location in this mode is $C000, while program memory locations $A000–$BFFF are considered reserved and should not be accessed.
  - **MSW[1:0] = 10**—The 8K higher locations ($4000–$5FFF) of the internal X data memory and the 8K higher locations ($4000–$5FFF) of the internal Y data memory are switched to internal program memory. In such a case, the on-chip program memory occupies the lowest 32K locations ($0–$7FFF) in the program memory space. The instruction cache, if enabled, occupies the lowest 1K program words (locations $0–$3FF). The lowest external program memory location in this mode is $C000, while program memory locations $8000–$BFFF are considered reserved and should not be accessed.
  - **MSW[1:0] = 11**—The 4K higher locations ($5000–$5FFF) of the internal X memory and the 4K higher locations ($5000–$5FFF) of the internal Y memory are switched to internal program memory. In such a case, the on-chip program memory occupies the lowest 24K locations ($0–$5FFF) in the

program memory space. The instruction cache, if enabled, occupies the lowest 1 K program words (locations $0–$3FF). The lowest external program memory location in this mode is $C000, while program memory locations $6000-$BFFF are considered reserved and should not be accessed.

3.2.3 Instruction Cache

In program memory space, the lowest 1024 (1K) program words (located at locations $0–$3FF) function as an internal instruction cache. When the instruction cache is enabled (i.e., the CE bit in the SR is set), the lowest 1K program words are reserved for instruction cache and should not be accessed for other purposes.

Note: When using an enabled instruction cache, you must assign a valid value for the vector address bus so that interrupts can be handled properly.

3.2.4 Program Bootstrap ROM

The program memory space occupying locations $FF0000–$FF00BF includes the internal bootstrap ROM. This ROM contains 192 words combining the bootstrap program for the DSP56307.

3.2.5 Accessing External Program Memory

Refer to the *DSP56300 Family Manual,* especially *Section 2 Expansion Port,* for detailed information on using the external memory interface to access external program memory.

3.3 X DATA MEMORY SPACE

The X data memory space consists of the following:

- Internal X data memory (24K by default down to 8K)
- Internal X I/O space (upper 128 locations)
- (Optionally) Off-chip memory expansion (as much as 64K in 16-bit mode, or 256K in 24-bit mode using the 18 external address lines, or 4 M using the external address lines and the four address attribute lines).

Note: The X memory space, located at locations $FF0000–$FFEFFF, is reserved and should not be accessed.

3.3.1 Internal X Data Memory

The default on-chip X data RAM is a 24-bit-wide, internal, static memory occupying the lowest 24 K locations ($0–$5FFF) in X memory space. The on-chip X data RAM is organized in 96 banks, 256 locations each. Available X data memory space is reduced and reallocated to program memory using the memory switch mode described in the next section.

3.3.2 Memory Switch Modes—X Data Memory

Memory switch mode reallocates of portions of X and Y data RAM as program RAM. Bit 7 in the OMR is the MS bit that controls this function, as follows:

- When the MS bit is cleared, the X data memory consists of the default 24K × 24-bit memory space described in the previous section. In this default mode, the lowest external X data memory location is $6000.
- When the MS bit is set, a portion of the higher locations of the internal X memory is switched to internal program memory. The memory switch (MSW[1:0]) configuration bits in the OMR select one of the following options:
  - **MSW[1:0] = 00**—The 16K higher locations ($2000–$5FFF) of the internal X memory are switched to internal program memory, and therefore the highest internal X memory location is $1FFF. The X memory space at the switched locations ($2000–$5FFF) becomes reserved and should not be accessed. The lowest external X memory location is $6000.
  - **MSW[1:0] = 01**—The 12K higher locations ($3000–$5FFF) of the internal X memory are switched to internal program memory, and therefore the highest internal X memory location will be $2FFF. The X memory space at the switched locations ($3000–$5FFF) becomes reserved and should not be accessed. The lowest external X memory location is $6000.
  - **MSW[1:0] = 10**—The 8K higher locations ($4000–$5FFF) of the internal X memory are switched to internal program memory, and therefore the highest internal X memory location will be $3FFF. The X memory space at the switched locations ($4000–$5FFF) becomes reserved and should not be accessed. The lowest external X memory location is $6000.

– **MSW[1:0] = 11**—The 4K higher locations ($5000–$5FFF) of the internal X memory are switched to internal program memory, and therefore the highest internal X memory location will be $4FFF. The X memory space at the switched locations ($5000–$5FFF) becomes reserved and should not be accessed. The lowest external X memory location is $6000.

Note: The 4K lowest locations ($0–$FFF) of the internal X memory are considered shared memory. The shared memory is accessible by the core and by the on-chip EFCOP. The EFCOP connects to the shared memory in place of the DMA bus. Therefore, there is no DMA accessibility to the shared memory, and simultaneous accesses are not permitted by core and EFCOP to the same memory bank (of 256 locations) of the shared memory. It is your responsibility to prevent such simultaneous accesses.

3.3.3 Internal X I/O Space

One part of the on-chip peripheral registers and some of the DSP56307 core registers occupy the top 128 locations of the X data memory ($FFFF80–$FFFFFF). This area is referred to as the internal X I/O space and it can be accessed by MOVE, MOVEP instructions and by bit-oriented instructions (BCHG, BCLR, BSET, BTST, BRCLR, BRSET, BSCLR, BSSET, JCLR, JSET, JSCLR and JSSET). The contents of the internal X I/O memory space are listed in **Appendix E Programming Reference**, **Table E-2** on page E-5.

3.3.4 Accessing External X Data Memory

Refer to the *DSP56300 Family Manual*, especially *Section 2 Expansion Port,* for detailed information on using the external memory interface to access external X data memory.

3.4 Y DATA MEMORY SPACE

The Y data memory space consists of the following:

- Internal Y data memory (24K by default down to 8K)
- Internal Y I/O space (16 locations—$FFFF80–$FFFF8F)
- External Y I/O space (upper 112 locations)

- (Optionally) Off-chip memory expansion (as much as 64K in 16-bit mode or 256K in 24-bit mode using the 18 external address lines or 4 M using the external address lines and the four address attribute lines)

Note: The Y memory space at locations $FF0000–$FFEFFF is reserved and should not be accessed.

3.4.1 Internal Y Data Memory

The default on-chip Y data RAM is a 24-bit-wide, internal, static memory occupying the lowest 24K locations ($0–$5FFF) in Y memory space. The on-chip Y data RAM is organized in 96 banks, 256 locations each. Available Y data memory space is reduced and reallocated to program memory by using the memory switch mode described in the following paragraphs.

3.4.2 Memory Switch Modes—Y Data Memory

Memory switch mode reallocates of portions of X and Y data RAM as program RAM. Bit 7 in the OMR is the MS bit that controls this function, as follows:

- When the MS bit is cleared, the Y data memory consists of the default 24K × 24-bit memory space described in the previous section. In this default mode, the lowest external Y data memory location is $6000.
- When MS mode bit in the OMR is set, a portion of the higher locations of the internal Y memory are switched to internal program memory. The memory switch configuration (MSW[1:0]) bits in the OMR select one of the following options:
  - **MSW[1:0] = 00**—The 16K higher locations ($2000–$5FFF) of the internal Y memory are switched to internal program memory, and therefore the highest internal Y memory location is $1FFF. The Y memory space at the switched locations ($2000–$5FFF) becomes reserved and should not be accessed by the user. The lowest external Y memory location is $6000.
  - **MSW[1:0] = 01**—The 12K higher locations ($3000–$5FFF) of the internal Y memory are switched to internal program memory, and therefore the highest internal Y memory location is $2FFF. The Y memory space at the switched locations ($3000–$5FFF) becomes reserved and should not be accessed. The lowest external Y memory location is $6000.

- **MSW[1:0] = 10**—The 8K higher locations ($4000–$5FFF) of the internal Y memory are switched to internal program memory, and therefore the highest internal Y memory location is $3FFF. The Y memory space at the switched locations ($4000–$5FFF) becomes reserved and should not be accessed. The lowest external Y memory location is $6000.
- **MSW[1:0] = 11**—The 4K higher locations ($5000–$5FFF) of the internal Y memory are switched to internal program memory, and therefore the highest internal Y memory location is $4FFF. The Y memory space at the switched locations ($5000-$5FFF) becomes reserved and should not be accessed. The lowest external Y memory location is $6000.

Note: The 4K lowest locations ($0-$FFF) of the internal Y memory are ***shared memory***, which is accessible to the core and the on-chip EFCOP. The EFCOP connects to the shared memory in place of the DMA bus. Therefore, DMA cannot access the shared memory, and simultaneous accesses are not permitted by core and EFCOP to the same memory bank (of 256 locations) of the shared memory. It is your responsibility to prevent such simultaneous accesses.

3.4.3 Internal Y I/O Space

The second part of the on-chip peripheral registers occupies 64 locations ($FFFF80–$FFFFBF) of the Y data memory. This area is referred to as the internal Y I/O space, and it can be accessed by MOVE, MOVEP instructions and by bit-oriented instructions (BCHG, BCLR, BSET, BTST, BRCLR, BRSET, BSCLR, BSSET, JCLR, JSET, JSCLR and JSSET). The contents of the internal Y I/O memory space are listed in **Appendix E Programming Reference, Table E-3 Internal Y I/O Memory Map** on page E-12.

3.4.4 External Y I/O Space

Off-chip peripheral registers should be mapped into the top 64 locations ($FFFFC0–$FFFFFF) 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). This area is referred to as the external Y I/O space.

3.4.5 Accessing External Y Data Memory

Refer to the *DSP56300 Family Manual* for detailed information about using the external memory interface to access external Y data memory.

3.5 DYNAMIC MEMORY CONFIGURATION SWITCHING

As described in the previous sections, the internal memory configuration is altered by remapping RAM modules from X and Y data memories into program memory space and vice versa. Data contents of the switched RAM modules are preserved. The memory can be dynamically switched from one configuration to another by changing the MS and MSW[1:0] bits in the OMR. Address ranges directly affected by the switch operation are described in the previous sections and summarized by the memory maps presented at the end of this section. The memory switch can be accomplished provided that the affected address ranges are not being accessed during the instruction cycle in which the switch operation takes place.

CAUTION

To ensure that dynamic switching is trouble-free, do not allow any accesses (including instruction fetches) to or from the affected address ranges in program and data memories during the switch cycle.

Note: The switch cycle actually occurs 3 instruction cycles after the instruction that modifies MS/MSW bits.

Any sequence that complies with the switch condition is valid. For example, if the program flow executes in the address range that is not affected by the switch, the switch condition can be met very easily. In this case, a switch can be accomplished by just changing MS/MSW bits in OMR in the regular program flow, assuming no accesses to the affected address ranges of the data memory occur up to three instructions after the instruction that changes the OMR bits. Because an interrupt could cause the DSP to fetch instructions out of sequence and might violate the switch condition, special care should be taken in relation to the interrupt vector routines.

CAUTION

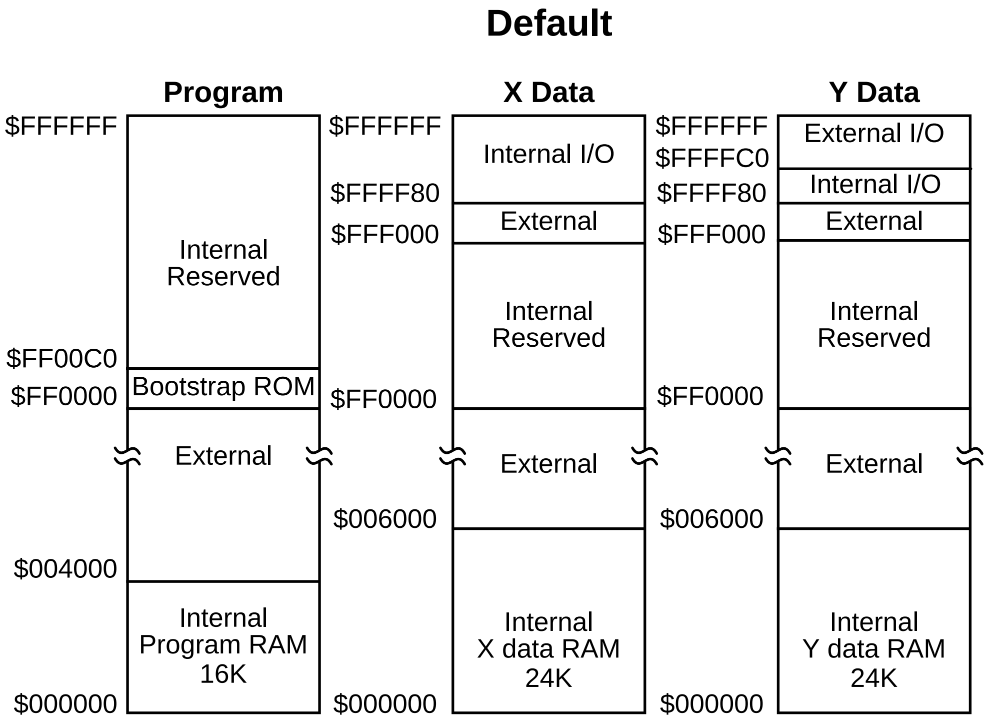
Special attention must be paid when executing a memory switch routine using the OnCE port. Running the switch routine in trace mode, for example, can cause the switch to complete after the MS/MSW bits change while the DSP is in debug mode. As a result, subsequent instructions might be fetched according to the new memory configuration (after the switch), and thus might execute improperly.

3.6 SIXTEEN-BIT COMPATIBILITY MODE CONFIGURATION

The sixteen-bit compatibility (SC) mode allows the DSP56307 to use DSP56000 object code without change. The SC bit (Bit 13 in the SR) is used to switch from the default 24-bit mode to this special 16-bit mode. SC is cleared by reset. You must set this bit to select the SC mode. The address ranges described in the previous sections apply in the SC mode with regard to the reallocation of X and Y data memory to program memory in MS mode, but the maximum addressing ranges are limited to $FFFF, and all data and program code are 16 bits wide.

3.7 MEMORY MAPS

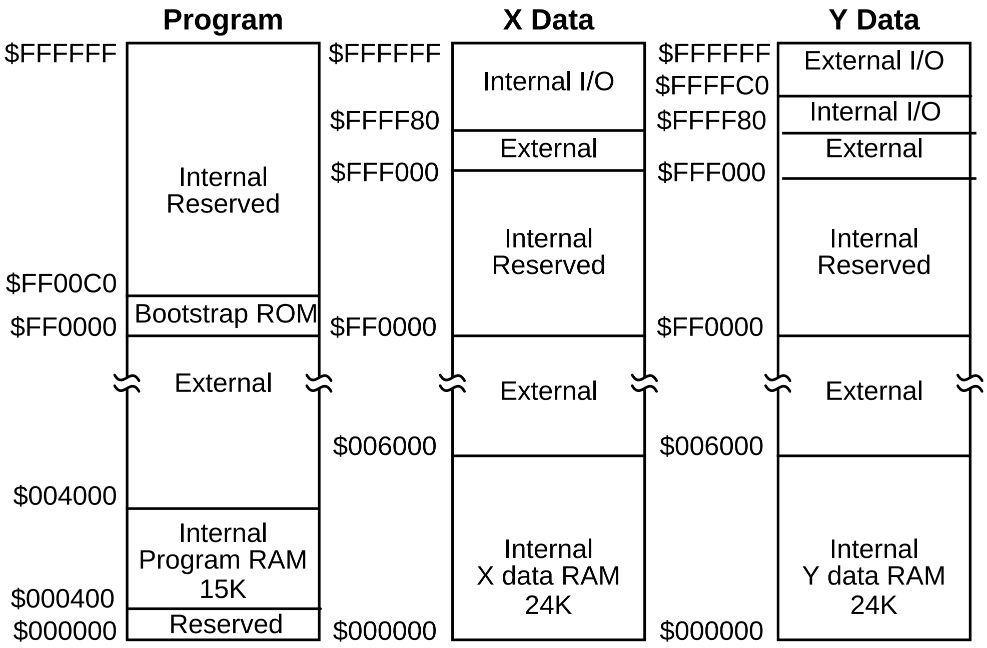
The following figures illustrate each of the memory space and RAM configurations defined by the settings of the MS (and MSW[1:0]), CE, and SC bits. The figures show the configuration and describe the bit settings, memory sizes, and memory locations.



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
0	any value	0	0	16K $0000–$3FFF	24K $0000–$5FFF	24K $0000–$5FFF	None	16 M

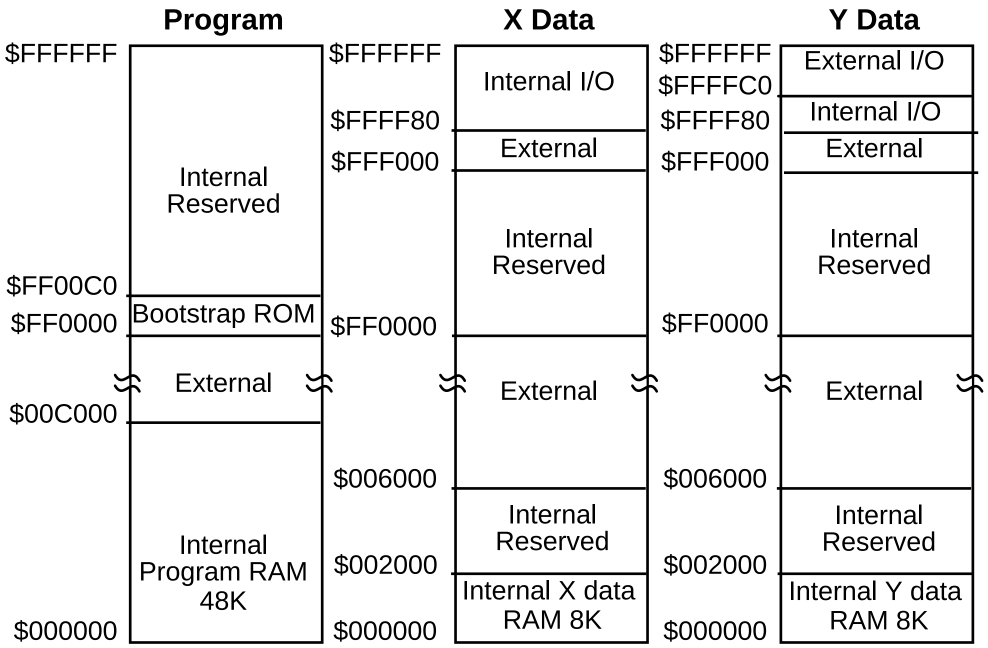
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

Figure 3-1 Memory Switch Off, Cache Off, 24-Bit Mode (default)



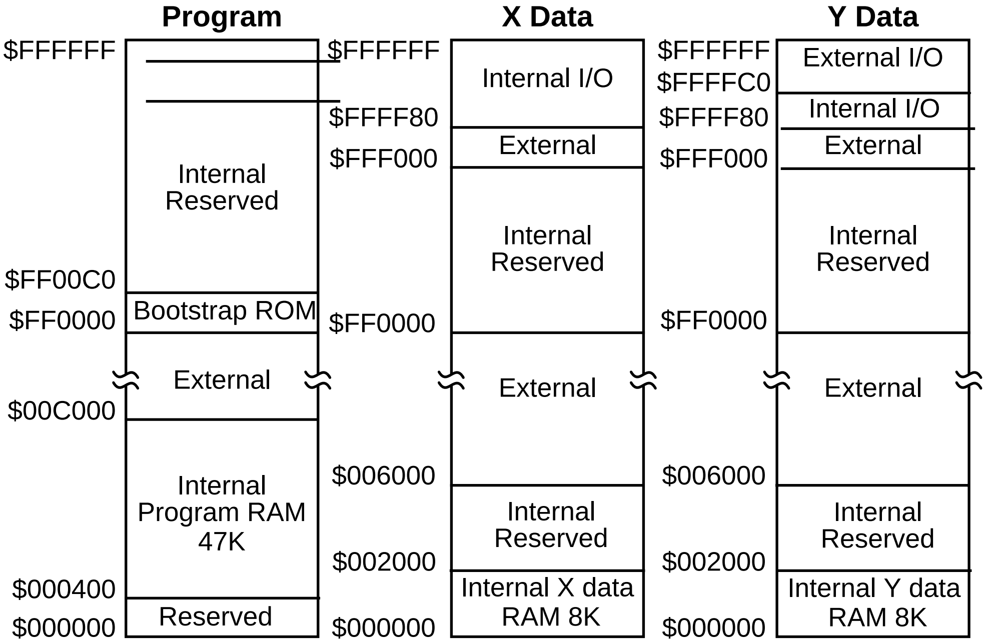
Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
0	any value	1	0	15K $0400–$3FFF	24K $0000–$5FFF	24K $0000–$5FFF	Enabled	16 M
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

Figure 3-2 Memory Switch Off, Cache On, 24-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	00	0	0	48K $0000–$BFFF	8K $0000–$1FFF	8K $0000–$1FFF	None	16 M
*	Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.							

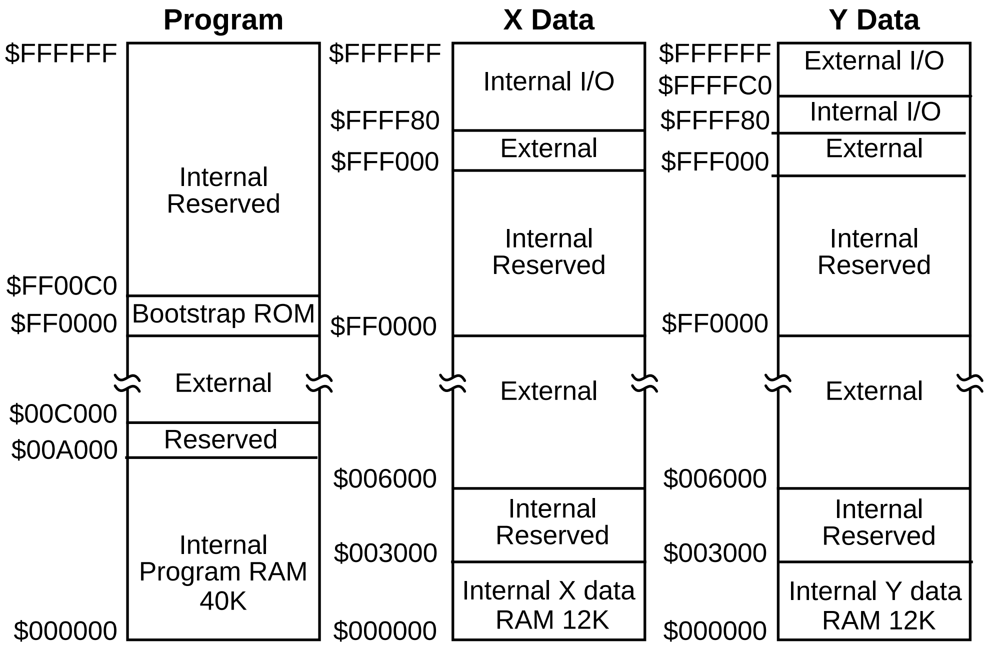
Figure 3-3 Memory Switch On (MSW = 00), Cache Off, 24-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	00	1	0	47K $0400–$BFFF	8K $0000–$1FFF	8K $0000–$1FFF	Enabled	16 M

* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

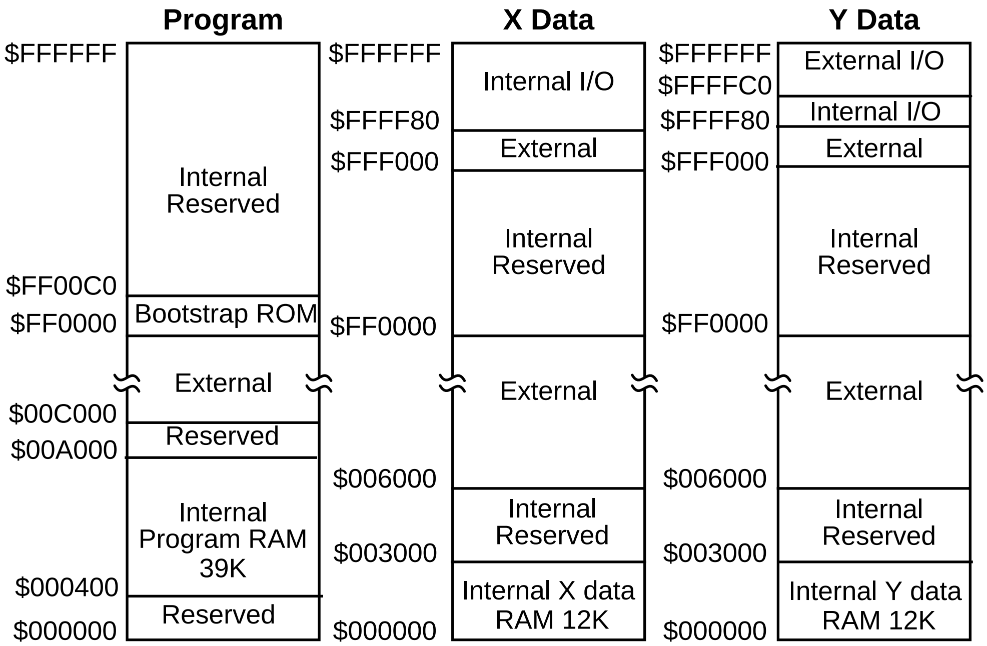
Figure 3-4 Memory Switch On (MSW = 00), Cache On, 24-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	01	0	0	40K $0000–$9FFF	12K $0000–$2FFF	12K $0000–$2FFF	None	16 M

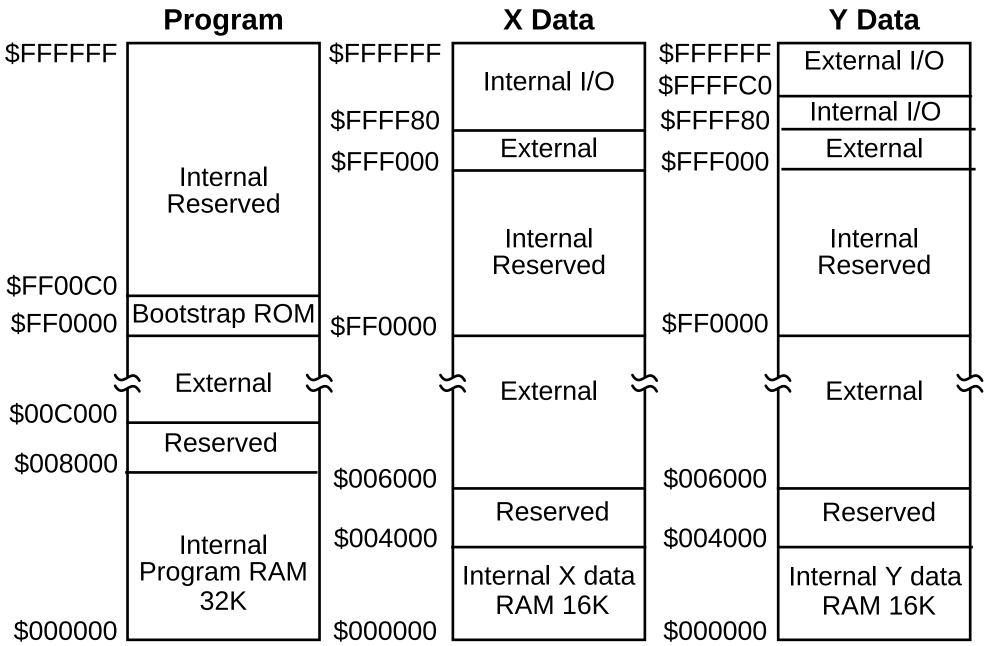
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

Figure 3-5 Memory Switch On (MSW = 01), Cache Off, 24-Bit Mode



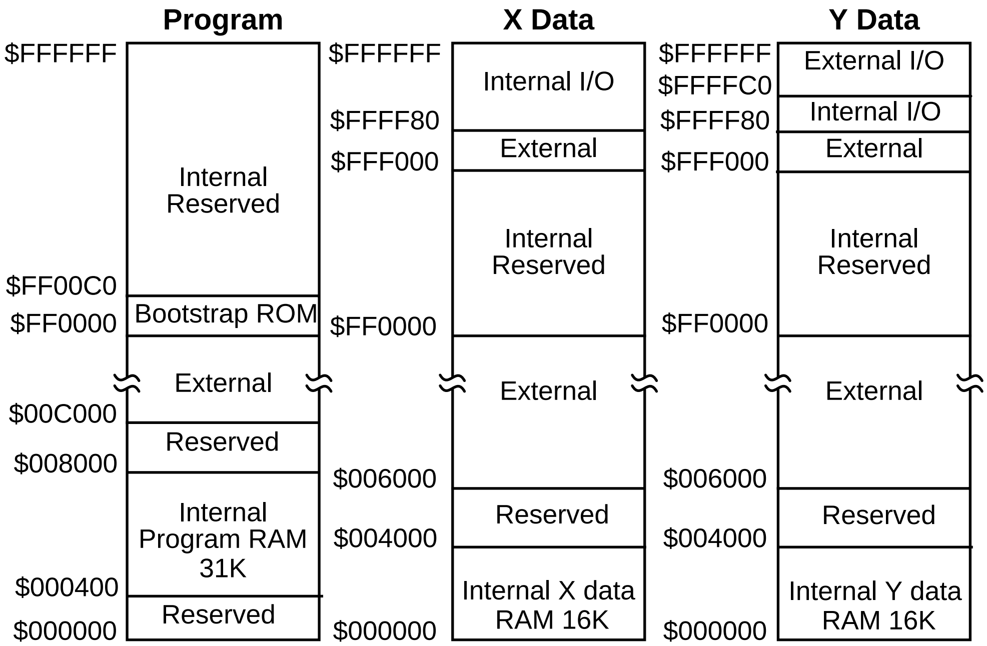
Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	01	1	0	39K $0400–$9FFF	12K $0000–$2FFF	12K $0000–$2FFF	Enabled	16 M
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

Figure 3-6 Memory Switch On (MSW = 01), Cache On, 24-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	10	0	0	32K $0000–$7FFF	16K $0000–$3FFF	16K $0000–$3FFF	None	16 M
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

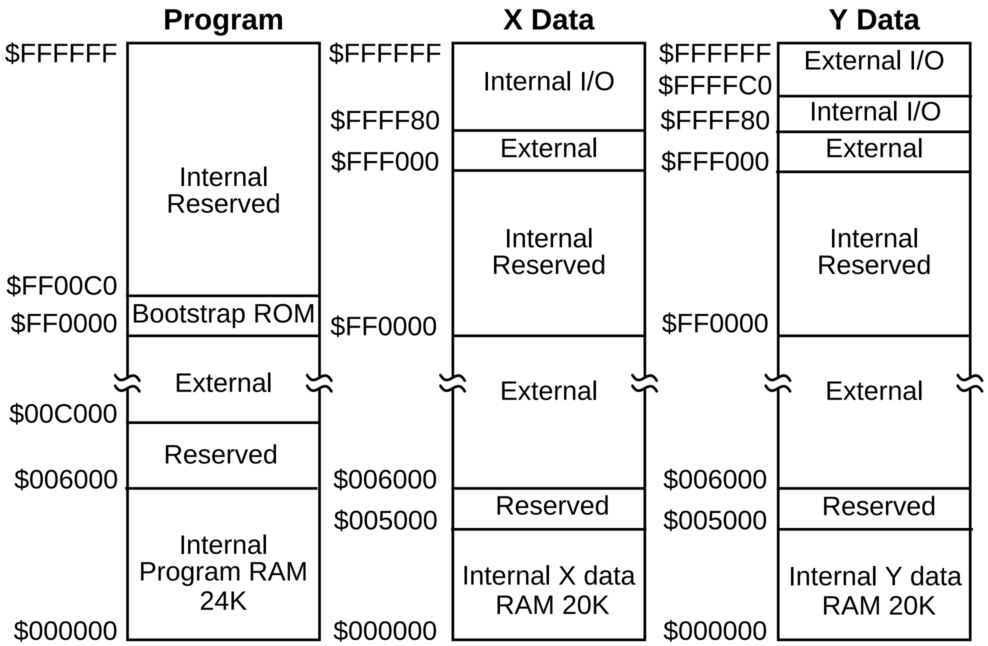
Figure 3-7 Memory Switch On (MSW = 10), Cache Off, 24-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	10	1	0	31K $0400–$7FFF	16K $0000–$3FFF	16K $0000–$3FFF	Enabled	16 M

* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

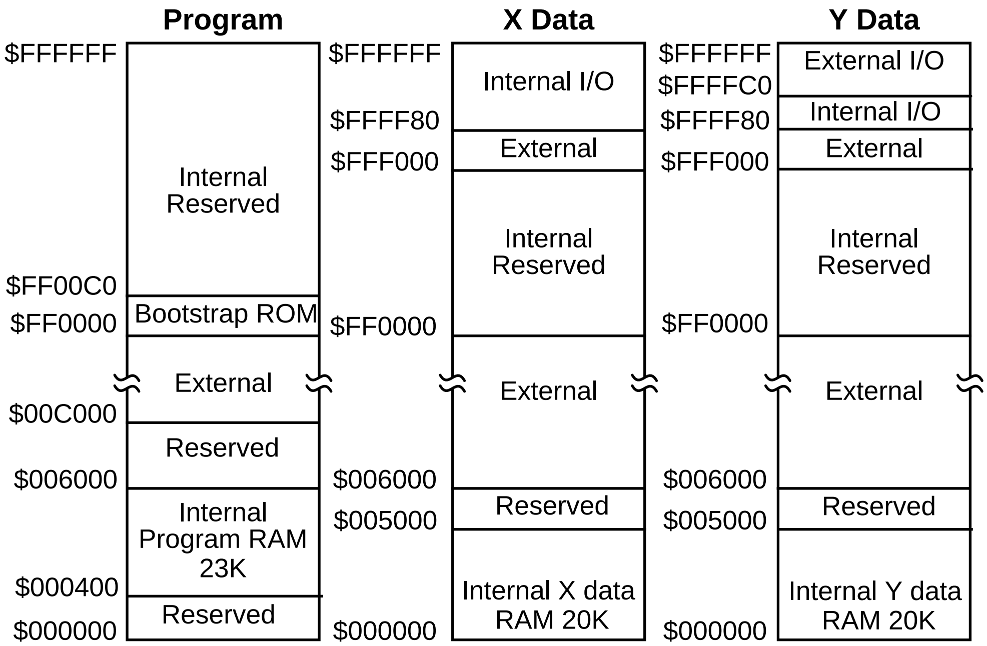
Figure 3-8 Memory Switch On (MSW = 10), Cache On, 24-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	11	0	0	24K $0000–$5FFF	20K $0000–$4FFF	20K $0000–$4FFF	None	16 M

* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

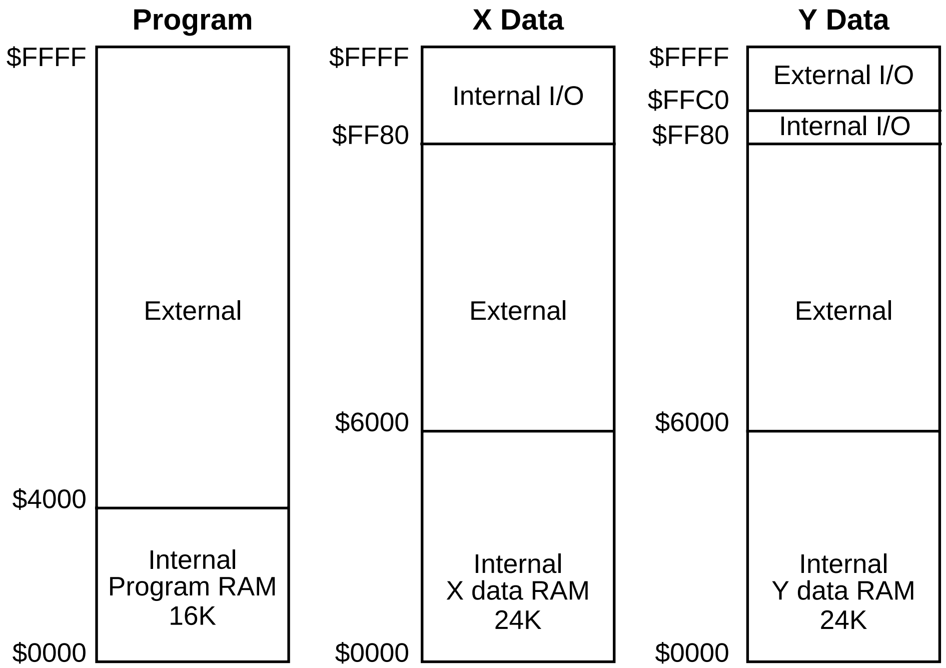
Figure 3-9 Memory Switch On (MSW = 11), Cache Off, 24-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	11	1	0	23K $0400–$5FFF	20K $0000–$4FFF	20K $0000–$4FFF	Enabled	16 M

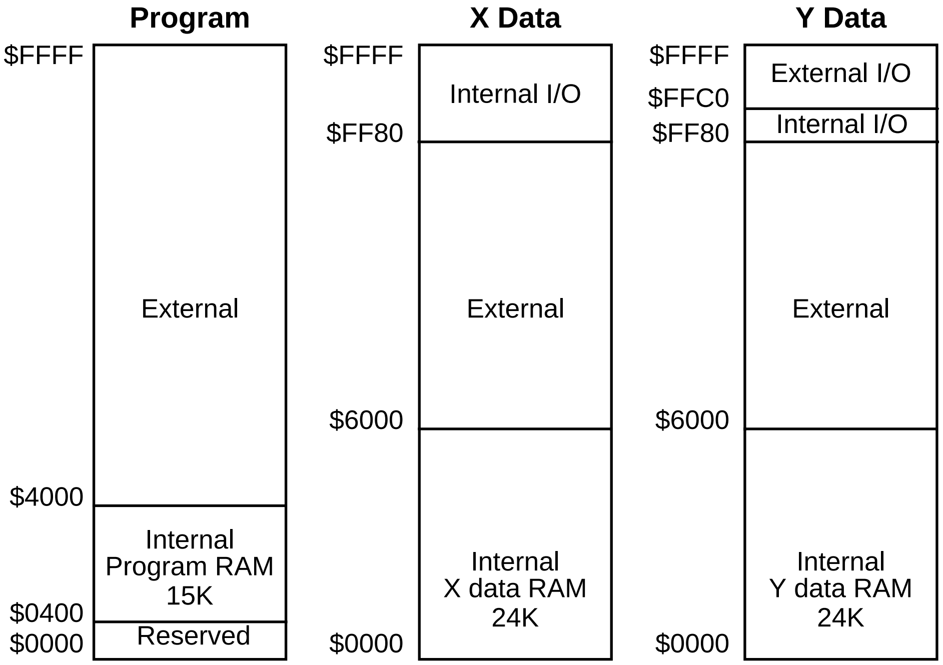
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

Figure 3-10 Memory Switch On (MSW = 11), Cache On, 24-Bit Mode



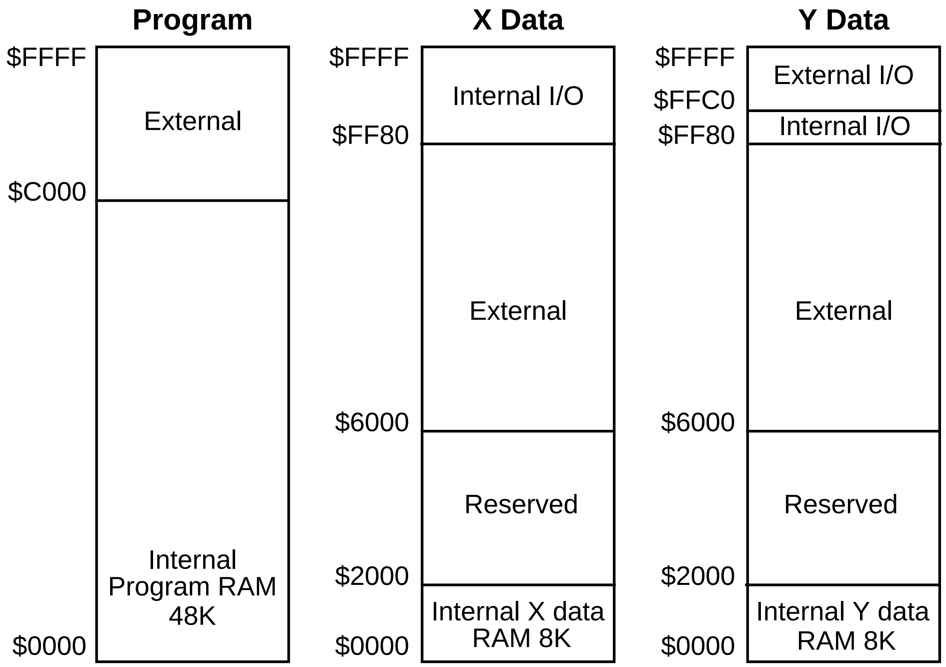
Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
0	any value	0	1	16K $0000–$3FFF	24K $0000–$5FFF	24K $0000–$5FFF	None	64K
*Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

Figure 3-11 Memory Switch Off, Cache Off, 16-Bit Mode



Bit Settings				Memory Configuration				
MS	**MSW [1:0]**	**CE**	**SC**	**Program RAM**	**X Data RAM***	**Y Data RAM***	**Cache**	**Addressable Memory Size**
0	any value	1	1	15K $0400–$3FFF	24K $0000–$5FFF	24K $0000–$5FFF	Enabled	64K
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

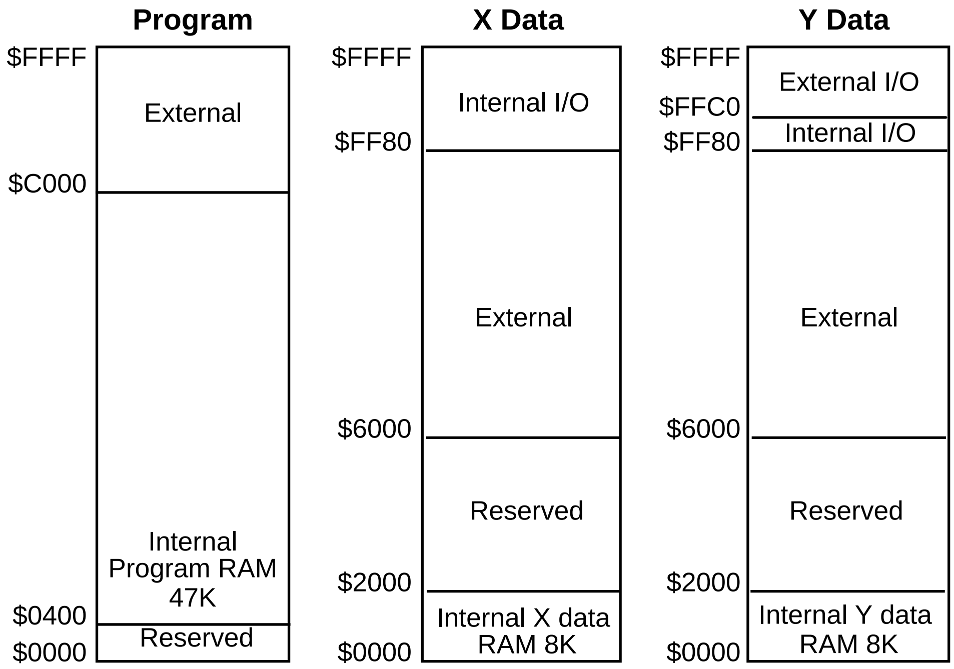
Figure 3-12 Memory Switch Off, Cache On, 16-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	00	0	1	48K $0000–$BFFF	8K $0000–$1FFF	8K $0000–$1FFF	None	64K

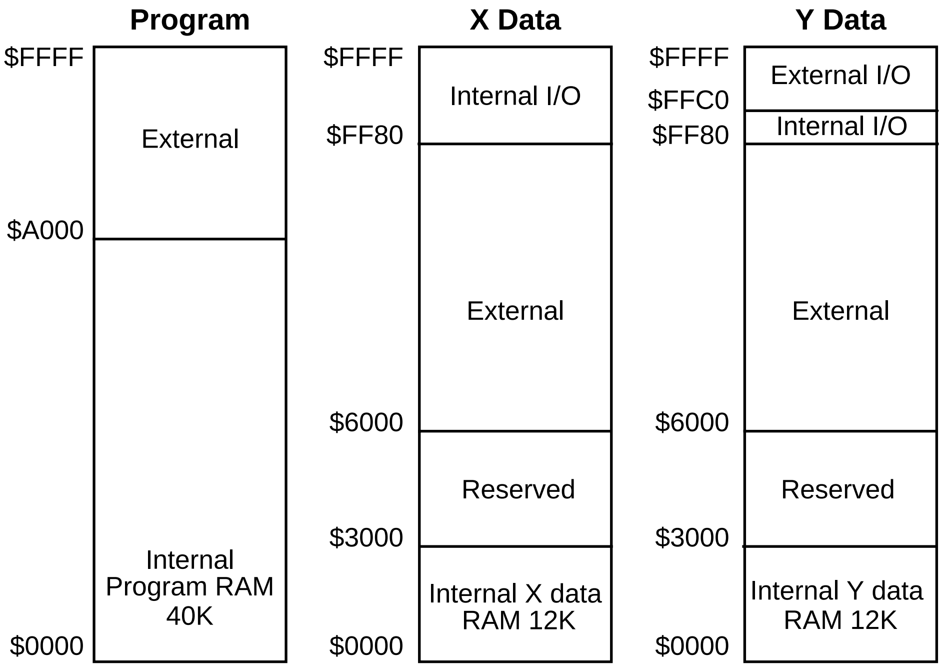
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

Figure 3-13 Memory Switch On (MSW = 00), Cache Off, 16-Bit Mode



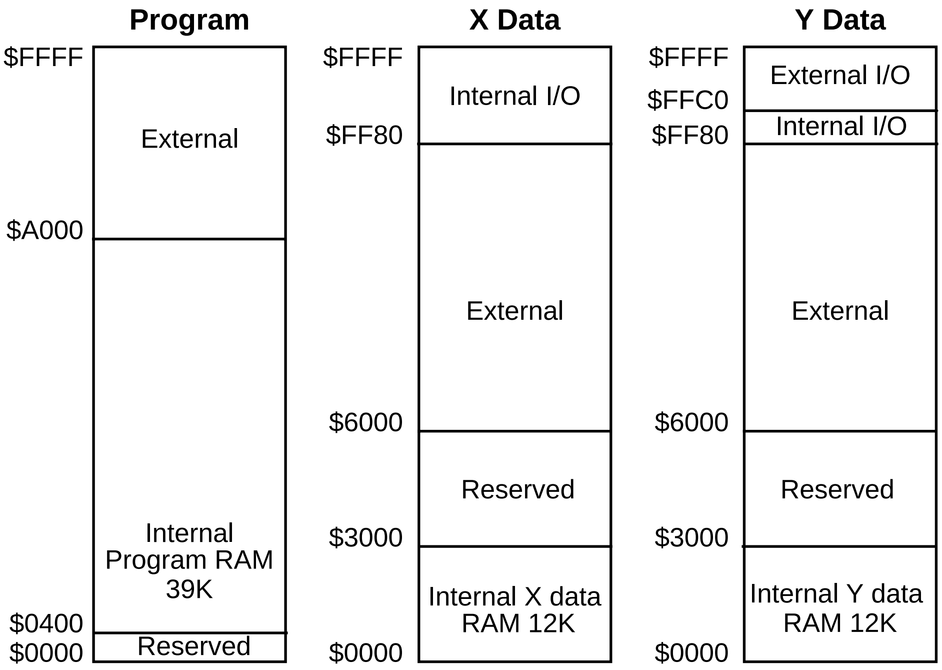
Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	00	1	1	47K $0400–$BFFF	8K $0000–$1FFF	8K $0000–$1FFF	Enabled	64K
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

Figure 3-14 Memory Switch On (MSW = 00), Cache On, 16-Bit Mode



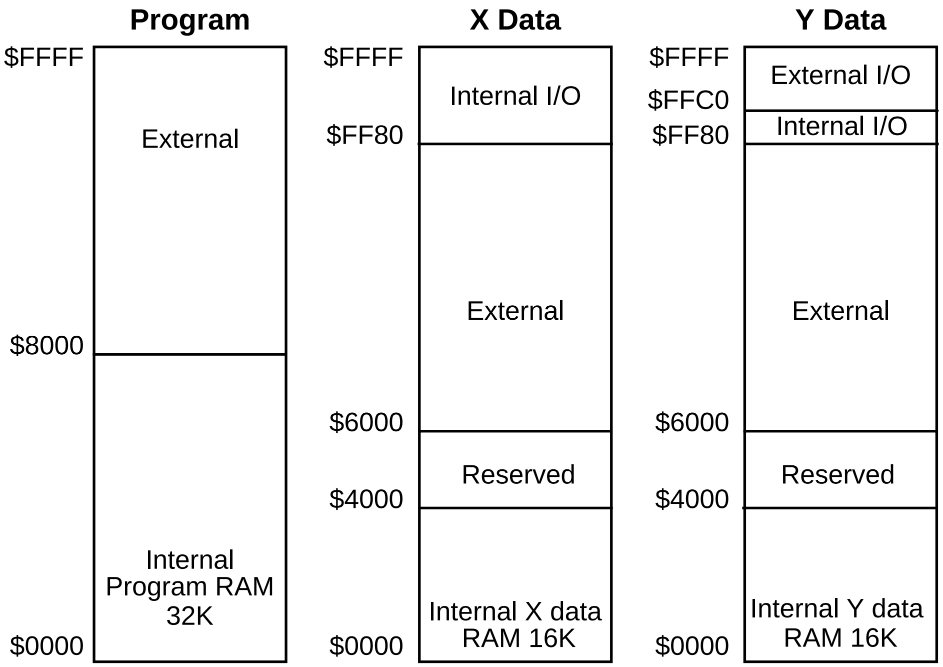
Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	01	0	1	40K $0000–$9FFF	12K $0000–$2FFF	12K $0000–$2FFF	None	64K
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

Figure 3-15 Memory Switch On (MSW = 01), Cache Off, 16-Bit Mode



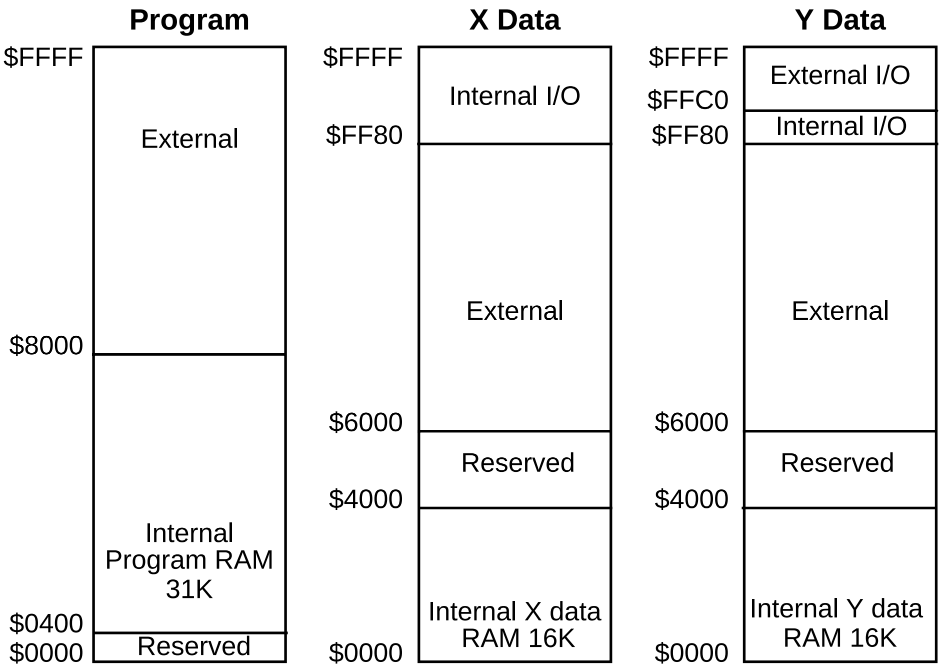
Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	01	1	1	39K $0400–$9FFF	12K $0000–$2FFF	12K $0000–$2FFF	Enabled	64K
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

Figure 3-16 Memory Switch On (MSW = 01), Cache On, 16-Bit Mode



Bit Settings				Memory Configuration				
MS	**MSW [1:0]**	**CE**	**SC**	**Program RAM**	**X Data RAM***	**Y Data RAM***	**Cache**	**Addressable Memory Size**
1	10	0	1	32K $0000–$7FFF	16K $0000–$3FFF	16K $0000–$3FFF	None	64K
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

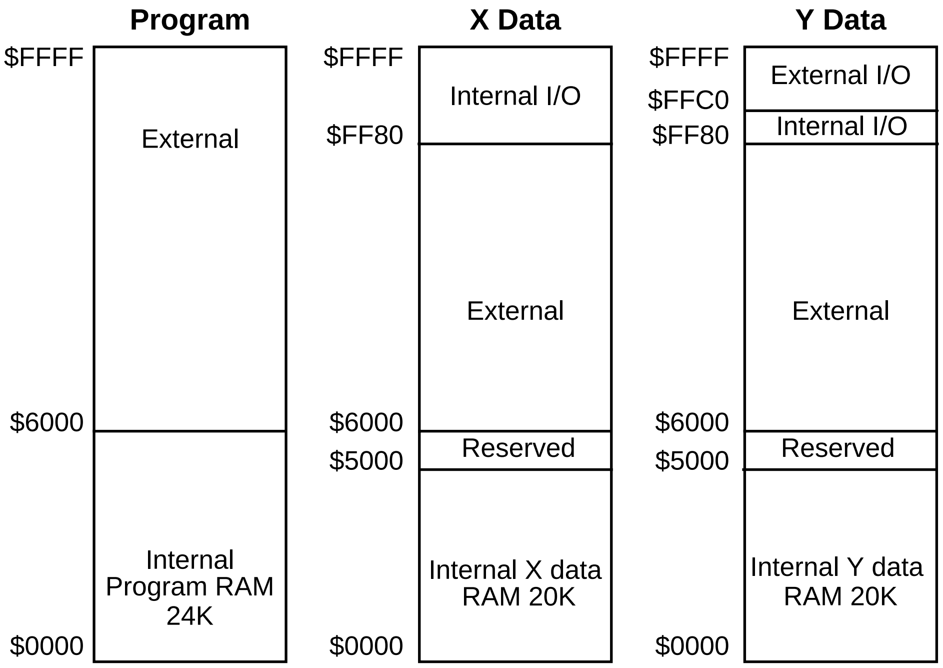
Figure 3-17 Memory Switch On (MSW = 10), Cache Off, 16-Bit Mode



Bit Settings				**Memory Configuration**				
MS	**MSW [1:0]**	**CE**	**SC**	**Program RAM**	**X Data RAM***	**Y Data RAM***	**Cache**	**Addressable Memory Size**
1	10	1	1	31K $0400–$7FFF	16K $0000–$3FFF	16K $0000–$3FFF	Enabled	64K

* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

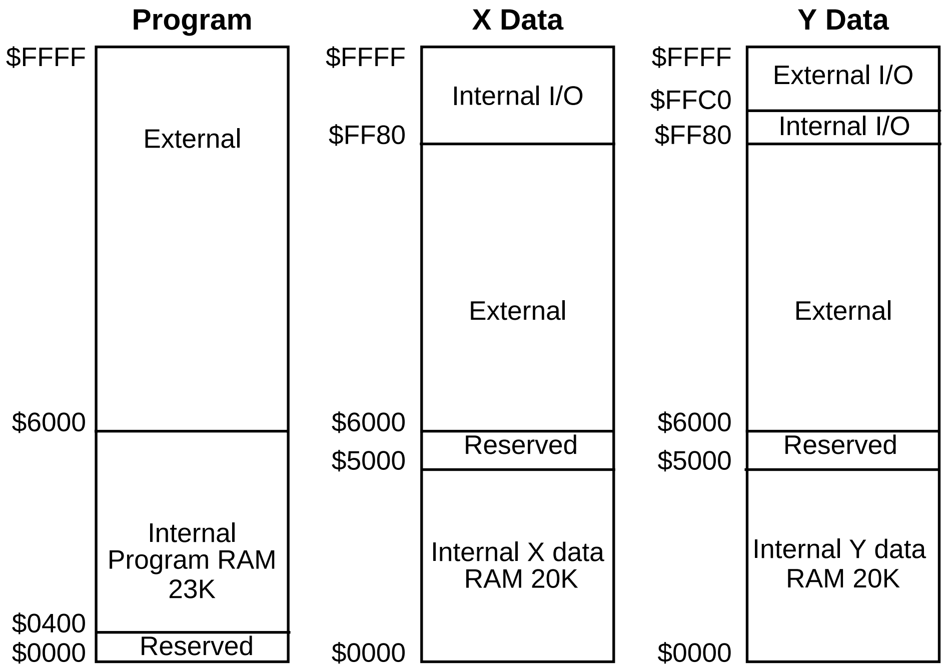
Figure 3-18 Memory Switch On (MSW = 10), Cache On, 16-Bit Mode



Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	11	0	1	24K $0000–$5FFF	20K $0000–$4FFF	20K $0000–$4FFF	None	64K

* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.

Figure 3-19 Memory Switch On (MSW = 11), Cache Off, 16-Bit Mode

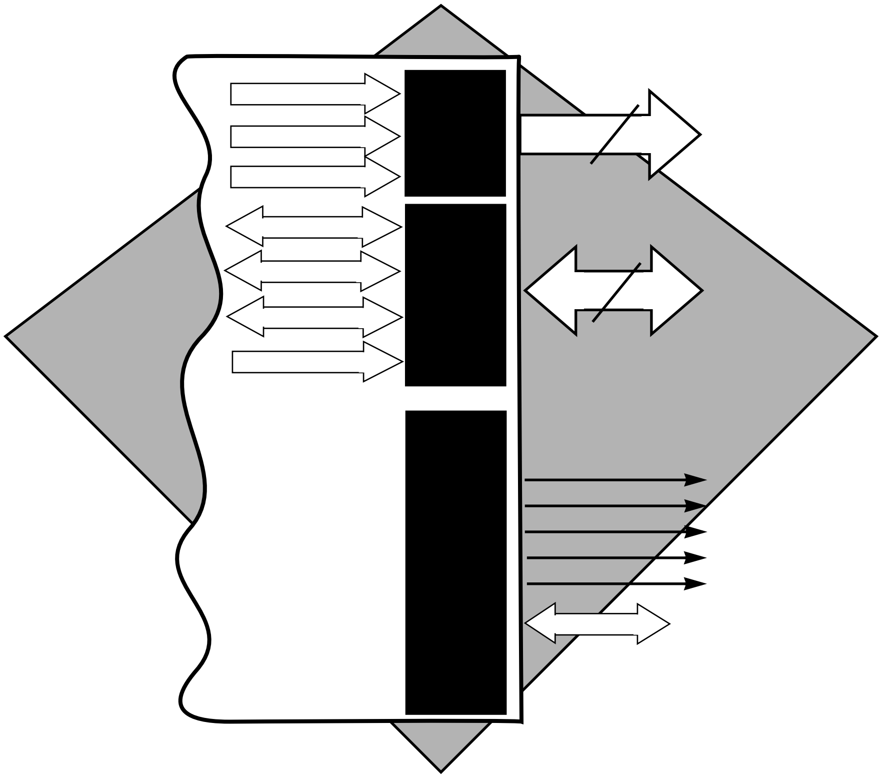


Bit Settings				Memory Configuration				
MS	MSW [1:0]	CE	SC	Program RAM	X Data RAM*	Y Data RAM*	Cache	Addressable Memory Size
1	11	1	1	23K $0400–$5FFF	20K $0000–$4FFF	20K $0000–$4FFF	Enabled	64K
* Lowest 4K of X data RAM and 4K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but is not accessible by the DMA controller.								

Figure 3-20 Memory Switch On (MSW = 11), Cache On, 16-Bit Mode

SECTION 4

CORE CONFIGURATION



4.1 INTRODUCTION

This chapter presents DSP56300 core configuration details specific to the DSP56307. These configuration details include the following:

- Operating modes
- Bootstrap program
- Interrupt sources and priorities
- DMA request sources
- OMR
- 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*.

4.2 OPERATING MODES

The DSP56307 begins operation by leaving the Reset state and going into one of eight operating modes. As the DSP56307 exits the Reset state, it loads the values of MODA, MODB, MODC, and MODD into bits MA, MB, MC, and MD of the OMR. These bit settings determine the chip’s operating mode, which determines the bootstrap program option the chip 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 DSP56307 to execute the specified bootstrap program option (except modes 0 and 8).

Table 4-1 shows the DSP56307 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 DSP56307 jumps once it leaves the Reset state.

Table 4-1 DSP56307 Operating Modes

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
0	0	0	0	0	$C00000	Expanded mode [1]
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	Bootstrap from byte-wide memory
A	1	0	1	0	$FF0000	Bootstrap through SCI
B	1	0	1	1	$FF0000	Reserved
C	1	1	0	0	$FF0000	HI08 bootstrap in ISA/DSP5630X mode
D	1	1	0	1	$FF0000	HI08 bootstrap in HC11 nonmultiplexed mode
E	1	1	1	0	$FF0000	HI08 bootstrap in 8051 multiplexed bus mode
F	1	1	1	1	$FF0000	HI08 bootstrap in MC68302 bus mode
Note: 1. Address $C00000 is reflected as address $00000 on Port A signals A0–A17.						

4.2.1 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 DSP56307 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 (default).

4.2.2 Modes 1 to 7: Reserved

These modes are reserved for future use.

4.2.3 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 DSP56307 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.

4.2.4 Mode 9—Boot from Byte-Wide External Memory

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
9	1	0	0	1	$FF0000	Bootstrap 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 A0–A17. The boot program concatenates every 3 bytes read from the external memory into a 24-bit wide DSP56307 word.

4.2.5 Mode A—Boot from SCI

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

Instructions are loaded through the SCI. The bootstrap program sets the SCI to operate in 10-bit asynchronous mode, with 1 start bit, 8 data bits, 1 stop bit, and no parity. Data is received in this order: start bit, 8 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. The boot program concatenates every 3 bytes read from the SCI into a 24-bit wide DSP56307 word.

4.2.6 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.

4.2.7 Mode C—Boot from HI08 in ISA Mode (8-Bit Bus)

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

In this mode, the HI08 is configured to interface with an ISA bus or with the memory expansion port of a master DSP5630***n*** processor through the HI08. The HI08 pin configuration is optimized for connection to the ISA bus or memory expansion port of a master DSP based on DSP56300 core.

4.2.8 Mode D—Boot from HI08 in HC11 Nonmultiplexed Mode

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

The bootstrap program sets the host interface to interface with the Motorola HC11 microcontroller through the HI08. The HI08 pin configuration is optimized for connection to Motorola HC11 nonmultiplexed bus.

.

4.2.9 Mode E—Boot from HI08 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

The bootstrap program sets the host interface to interface with the Intel 8051 bus through the HI08. The program stored in this location, after testing MODA, MODB, MODC, and MODD, bootstraps through HI08. The HI08 pin configuration is optimized for connection to the Intel 8051 multiplexed bus.

4.2.10 Mode F—Boot from HI08: MC68302/68360 Bus Mode

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
7	F	1	1	1	$FF0000	HI08 bootstrap in MC68302 bus

The bootstrap program sets the host interface to interface with the Motorola MC68302 or MC68360 bus through the HI08. The HI08 pin configuration is optimized for connection to a Motorola MC68302 or MC68360 bus.

4.3 BOOTSTRAP PROGRAM

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**.

Upon exiting the Reset state, the DSP56307 performs these steps:

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

As explained in Section 4.2, the mode input signals (MODA–MODD) and the resulting MA, MB, MC, and MD bits determine which bootstrap mode the DSP56307 enters:

- If MA, MB, MC, and MD are all cleared (bootstrap mode 0), the program bypasses the bootstrap ROM, and the DSP56307 starts loading instructions from external program memory location $C00000.
- If MA, MB, and MC are cleared and MD is set (bootstrap mode 8), the program bypasses the bootstrap ROM and the DSP56307 starts loading in instruction values from external program memory location $008000.
- Otherwise, the DSP56307 jumps to the bootstrap program entry point at $FF0000.

Note: The bootstrap in any HI08 bootstrap mode may be stopped by setting the Host Flag 0 (HF0). This starts execution of the loaded user program from the specified starting address.

You can invoke the bootstrap program options (except modes 0 and 8) 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 DSP56307 functioning modes. Bootstrap modes 9, A, and C–F select different specific bootstrap loading source devices. In these 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 DSP56307 program memory
3. The user program (three bytes for each 24-bit program word)

Note: The three bytes for each data sequence must be loaded 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.

4.4 INTERRUPT SOURCES AND PRIORITIES

DSP56307 interrupt handling, like that of all DSP56300 family members, is 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 by the vector base address (VBA) register in the PCU points.

4.4.1 Interrupt Sources

Because each interrupt is allocated two instructions in the table, there are 128 table entries for interrupt handling. **Table 4-2** shows the table entry address for each interrupt source. The DSP56307 initialization program loads the table entry for each interrupt serviced with two interrupt servicing instructions. In the DSP56307, only some of the 128 vector addresses are used for specific interrupt sources. The remaining interrupt vectors are reserved and can be used for host NMI (IPL = 3) or for host command interrupt (IPL = 2). If certain interrupts are not to 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:$0A	3	Nonmaskable interrupt ($\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
VBA:$38	0–2	ESSI0 transmit data with exception status
VBA:$3A	0–2	ESSI0 transmit last slot

Table 4-2 Interrupt Sources (Continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
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:$68	0 - 2	EFCOP data input buffer empty
VBA:$6A	0 - 2	EFCOP data output buffer full
VBA:$6C	0 - 2	Reserved
VBA:$6E	0 - 2	Reserved
:	:	:
VBA:$FE	0–2	Reserved

4.4.2 Interrupt Priority Levels

There are two interrupt priority registers in the DSP56307. The IPR–C is dedicated to DSP56300 core interrupt sources, and IPR–P is dedicated to DSP56307 peripheral interrupt sources. IPR–C is shown on **Figure 4-1** and IPR–P is shown in **Figure 4-2**.

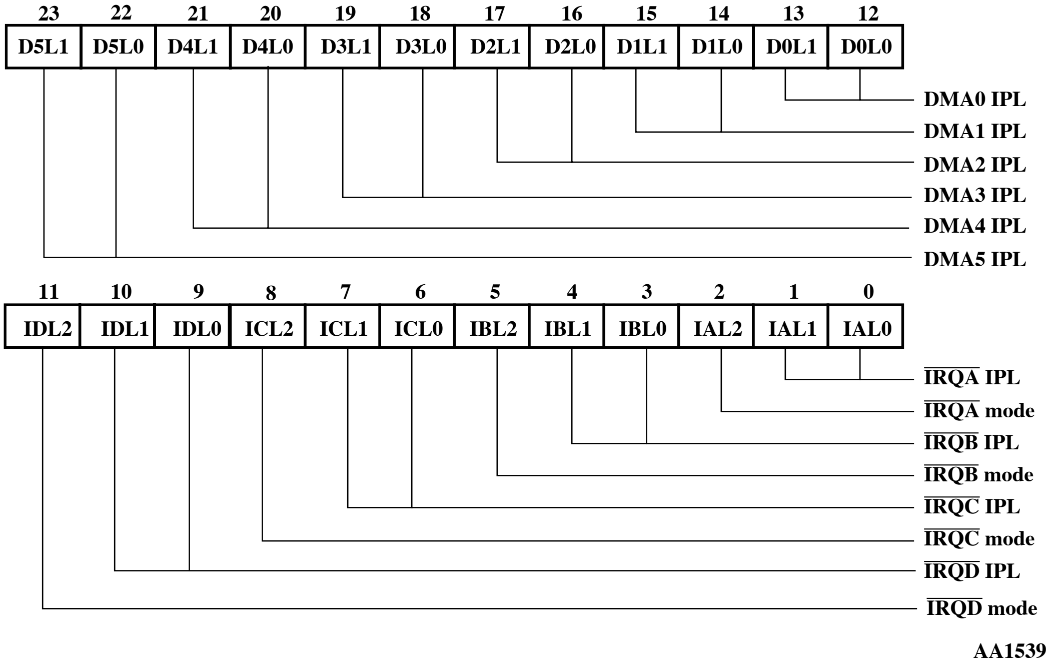

23 22 21 20 19 18 17 16 15 14 13 12
D5L1 D5L0 D4L1 D4L0 D3L1 D3L0 D2L1 D2L0 D1L1 D1L0 D0L1 D0L0
DMA0 IPL
DMA1 IPL
DMA2 IPL
DMA3 IPL
DMA4 IPL
DMA5 IPL
11 10 9 8 7 6 5 4 3 2 1 0
IDL2 IDL1 IDL0 ICL2 ICL1 ICL0 IBL2 IBL1 IBL0 IAL2 IAL1 IAL0
$\overline{\text{IRQA}}$ IPL
$\overline{\text{IRQA}}$ mode
$\overline{\text{IRQB}}$ IPL
$\overline{\text{IRQB}}$ mode
$\overline{\text{IRQC}}$ IPL
$\overline{\text{IRQC}}$ mode
$\overline{\text{IRQD}}$ IPL
$\overline{\text{IRQD}}$ mode
AA1539

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

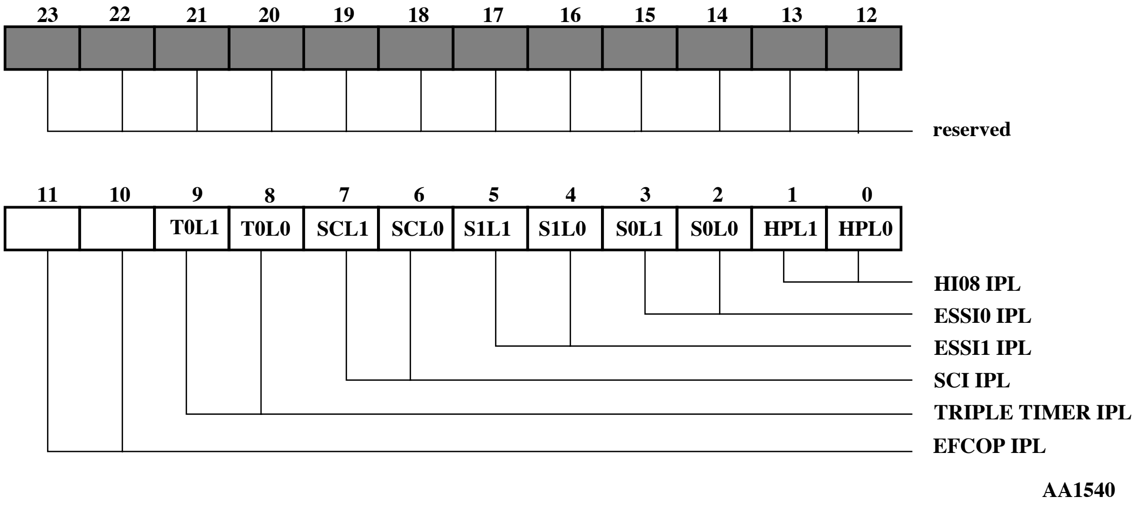


Figure 4-2 Interrupt Priority Register P (IPR-P) (X:$FFFFFE)

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

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 with 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	Nonmaskable 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

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

Priority	Interrupt Source
	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
	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
	TIMER2 compare interrupt
	EFCOP data input buffer empty
Lowest	EFCOP data output buffer full

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 may 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** shows the values of the DRS bits.

Table 4-5 DMA Request Sources

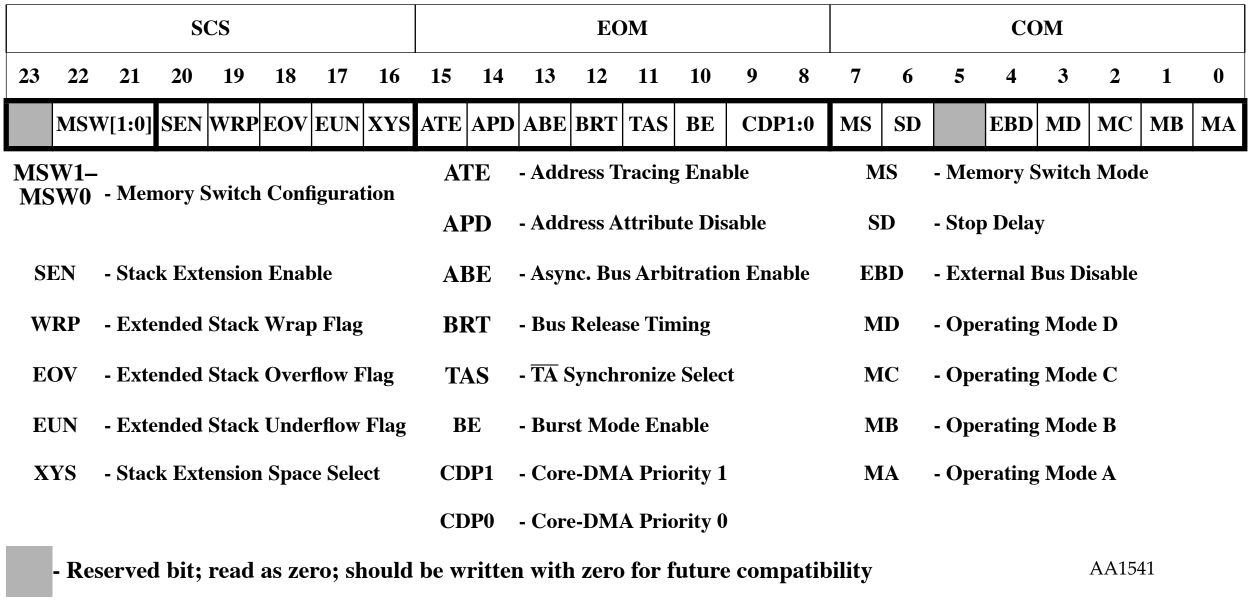
DMA Request Source Bits DRS4 . . . DRS0	Requesting Device
00000	External ($\overline{\text{IRQA}}$ signal)
00001	External ($\overline{\text{IRQB}}$ signal)
00010	External ($\overline{\text{IRQC}}$ signal)
00011	External ($\overline{\text{IRQD}}$ signal)
00100	Transfer done from DMA channel 0
00101	Transfer 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)
01101	ESSI1 transmit data (TDE1 = 1)
01110	SCI receive data (RDRF = 1)
01111	SCI transmit data (TDRE = 1)

Table 4-5 DMA Request Sources (Continued)

DMA Request Source Bits DRS4 . . . DRS0	Requesting Device
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	EFCOP input buffer empty (FDIBE=1)
10110	EFCOP output buffer full (FDOBF=1)
10111–11111	Reserved

4.6 OMR

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



SCS								EOM								COM							
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	MSW[1:0]		SEN	WRP	EOV	EUN	XYS	ATE	APD	ABE	BRT	TAS	BE	CDP1:0		MS	SD	▨	EBD	MD	MC	MB	MA

MSW1– MSW0	- Memory Switch Configuration	ATE	- Address Tracing Enable	MS	- Memory Switch Mode
		APD	- Address Attribute Disable	SD	- Stop Delay
SEN	- Stack Extension Enable	ABE	- Async. Bus Arbitration Enable	EBD	- External Bus Disable
WRP	- Extended Stack Wrap Flag	BRT	- Bus Release Timing	MD	- Operating Mode D
EOV	- Extended Stack Overflow Flag	TAS	- $\overline{TA}$ Synchronize Select	MC	- Operating Mode C
EUN	- Extended Stack Underflow Flag	BE	- Burst Mode Enable	MB	- Operating Mode B
XYS	- Stack Extension Space Select	CDP1	- Core-DMA Priority 1	MA	- Operating Mode A
		CDP0	- Core-DMA Priority 0		

▨ - Reserved bit; read as zero; should be written with zero for future compatibility

AA1541

Figure 4-3 DSP56307 Operating Mode Register (OMR) Format

4.7 PLL CONTROL REGISTER

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

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

AA0852

Figure 4-4 PLL Control Register (PCTL)

4.7.1 Predivider Factor Bits (PD[3:0])—PCTL Bits 23–20

The predivider factor bits (PD[3:0]) define the predivision factor (PDF) to be applied to the PLL input frequency. The PD[3:0] bits are cleared during DSP56307 hardware reset, which corresponds to a PDF of one.

4.7.2 Clock Output Disable (COD) Bit—PCTL Bit 19

The COD bit controls the output buffer of the clock at the CLKOUT pin. When COD is set, the CLKOUT output is pulled high. When COD is cleared, the CLKOUT pin provides a 50% duty cycle clock synchronized to the internal core clock.

4.7.3 PLL Enable (PEN) Bit—PCTL Bit 18

The PEN bit enables PLL operation.

4.7.4 PLL Stop State (PSTP) Bit—Bit 17

The PSTP bit controls PLL and on-chip crystal oscillator behavior during the stop processing state.

4.7.5 XTAL Disable (XTLD) Bit—PCTL Bit 16

The XTLD bit controls the on-chip crystal oscillator XTAL output. The XTLD bit is cleared during DSP56307 hardware reset; consequently, the XTAL output signal is active, permitting normal operation of the crystal oscillator.

4.7.6 Crystal Range (XTLR) Bit—PCTL Bit 15

The XTLR bit controls the on-chip crystal oscillator transconductance. The XTLR bit is set to a predetermined value during hardware reset.

4.7.7 PCTL Bits 14–12

The division factor bits DF[2:0] define the DF of the low-power divider. These bits specify the DF as a power of two in the range from 2^0 to 2^7.

4.7.8 PLL Multiplication Factor—PCTL Bits 11–0

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 DSP56307 hardware reset, and thus it corresponds to an MF of one.

4.8 DEVICE IDENTIFICATION REGISTER (IDR)

The 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 is used in testing or by software. **Figure 4-5** shows the contents of the IDR.

Revision numbers are assigned as follows: $0 is revision 0, $1 is revision A, and so on.

23 16	15 12	11 0
Reserved	**Revision Number**	**Derivative Number**
$00	**$0**	**$307**

AA1503

Figure 4-5 Identification Register Configuration (Revision 0)

4.9 ADDRESS ATTRIBUTE REGISTERS (AAR1–AAR4)

An AAR is shown in Figure 4-6. There are four of these registers in the DSP56307 (AAR0–AAR3), one for each address attribute signal. For a full description of the address attribute registers see the *DSP56300 Family Manual*. Address multiplexing is not supported by the DSP56307. Bit 6 (BAM) of the AAR is reserved and should be written with 0 only.

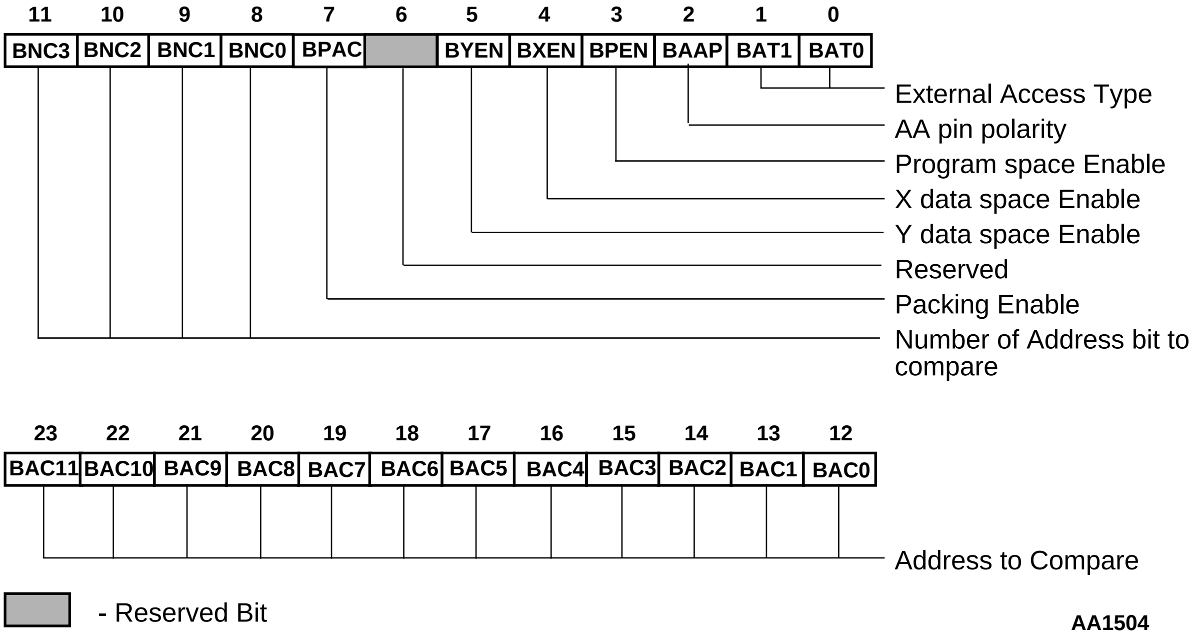


Figure 4-6 Address Attribute Registers (AAR0–AAR3) (X:$FFFFF9–$FFFFF6)

4.10 JTAG IDENTIFICATION (ID) REGISTER

The JTAG ID register is a 32-bit read-only factory-programmed register that distinguishes the component on a board according to the IEEE 1149.1 standard. **Figure 4-7** shows the JTAG ID register configuration. Version information corresponds to the revision number ($0 for revision 0, $1 for revision A, etc.).

31 28	27 22	21 12	11 1	0
Version Information	Customer Part Number	Sequence Number	Manufacturer Identity	1
0000	000110	0000000111	00000001110	1 AA1505

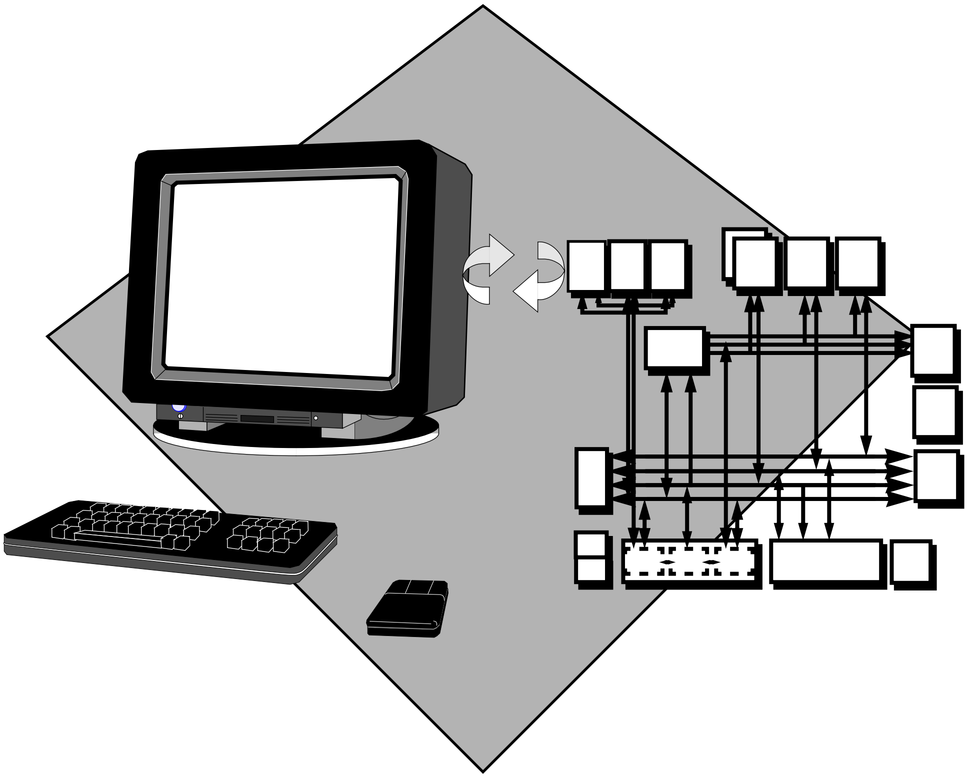
Figure 4-7 JTAG Identification Register Configuration (Revision 0)

4.11 JTAG BOUNDARY SCAN REGISTER (BSR)

The BSR in the DSP56307 JTAG implementation contains bits for all device signals, clock pins, and their associated control signals. All DSP56307 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 12 Joint Test Action Group Port** of this manual, and the JTAG code listing is in **Appendix C DSP56307 BSDL Listing**.

SECTION 5

GENERAL-PURPOSE INPUT/OUTPUT



5.1 INTRODUCTION

The DSP56307 provides 34 bidirectional signals that can be configured as GPIO signals or as peripheral dedicated signals. No dedicated GPIO signals are provided. All of these signals are GPIO by default after reset. The control register settings of the DSP56307 peripherals determine whether these signals function as GPIO or as peripheral dedicated signals. This section tells how signals can be used as GPIO.

5.2 PROGRAMMING MODEL

Section 2 Signal/Connection Descriptions of this manual documents in detail the special uses of the 34 bidirection signals. These signals fall into five groups and are controlled separately or as the following groups.

- Port B: 16 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 an 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** of this manual.

5.2.3 Port D Signals and Registers

Each of the six Port D signals not used as an 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 documented in **Section 7 Enhanced Synchronous Serial Interface** of this manual.

5.2.4 Port E Signals and Registers

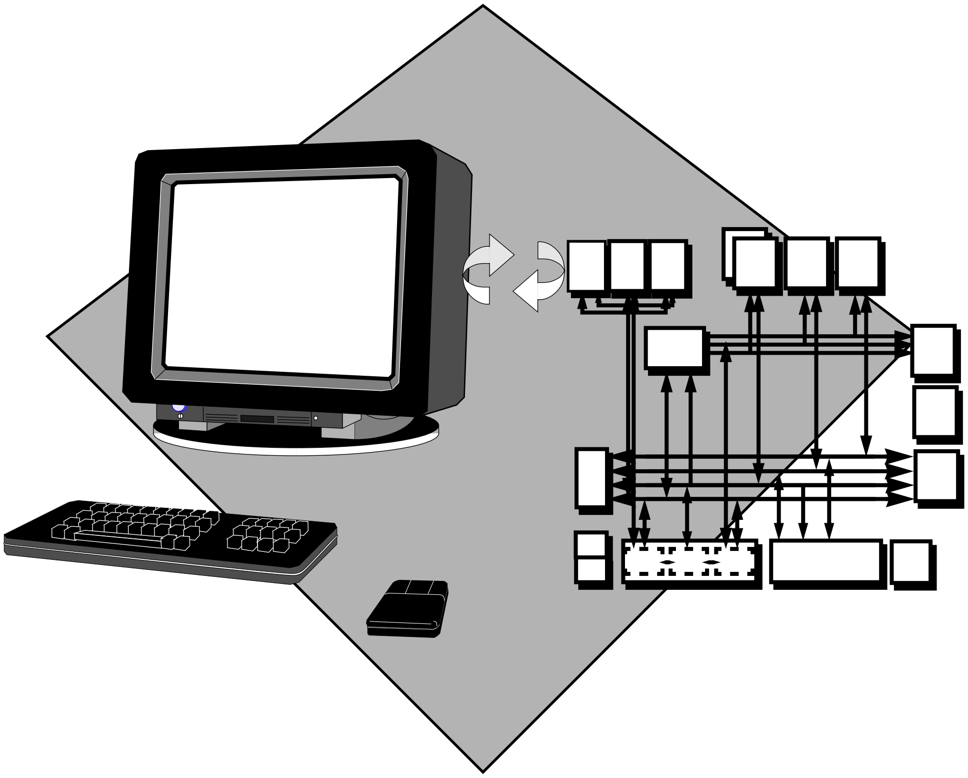
Each of the three Port E signals not used as an SCI signal can be configured as a GPIO signal. The GPIO functionality of Port E is controlled by three registers: Port E control register (PCRE), Port E direction register (PRRE) and Port E data register (PDRE). These registers are documented in **Section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

Because the host bus can operate asynchronously to the DSP core clock, 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 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. **Sections 6.5** and **6.6** contain further details.

6.2.1 Host-to-DSP Core Interface

- Mapping:
  - Registers are directly mapped into eight internal X data memory locations.
- Data word:
  - DSP56307 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 transfer data between the DSP56307 and external hosts.
  - A 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 nonmultiplexed 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/HA10 host chip select (HCS) or host address line (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 bits

- 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
        - Real-time production diagnostics
        - Creation of a debugging window for program development
        - Host control protocols
- Interface capabilities:
    - Glueless interface (no external logic required) to
        - Motorola HC11
        - Hitachi H8
        - 8051 family
        - Thomson P6 family
    - Minimal glue-logic (pull-ups, pull-downs) required to interface to
        - ISA bus
        - Motorola 68K family
        - Intel X86 family

6.3 HI08 HOST PORT SIGNALS

The host port signals are documented in **Section 2 Signal/Connection Descriptions**. Each host port signal may be programmed as a host port signal or as a GPIO signal, PB0–PB15. See **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	Nonmultiplexed 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/HA10	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

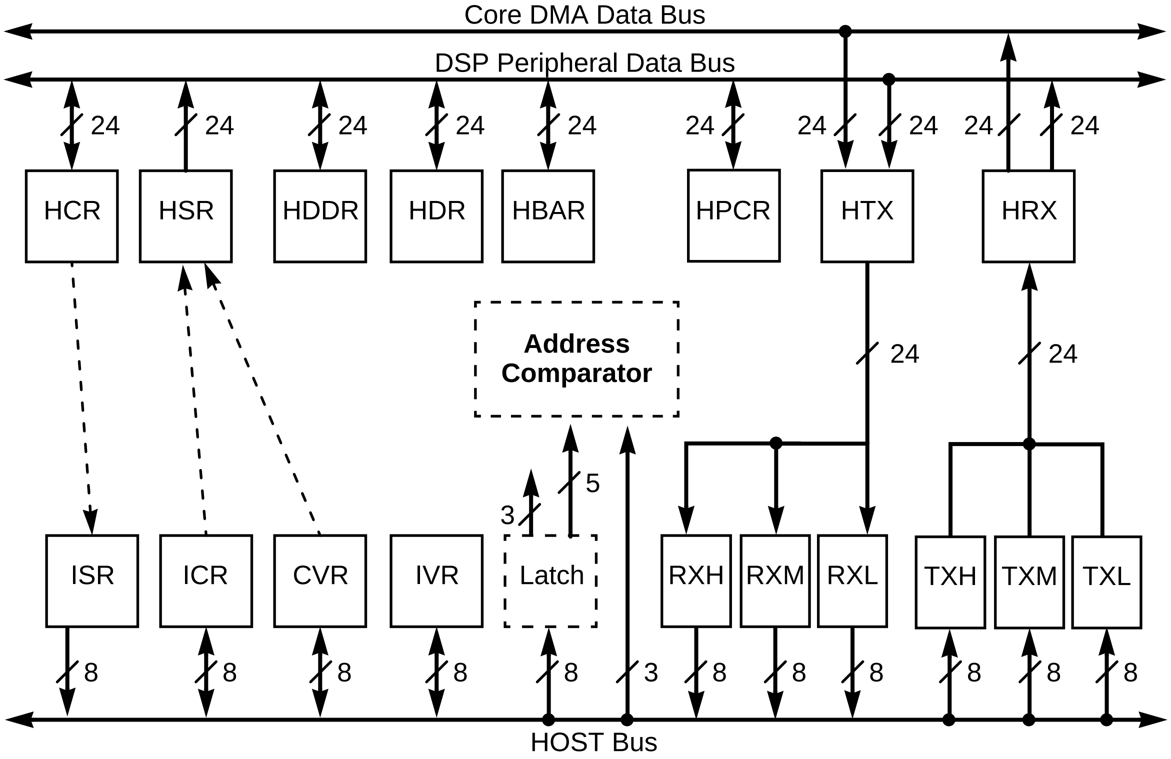
HI08 Port Signal	Vector Required	No Vector Required	GPIO Mode
HREQ/ HTRQ	$\overline{\text{HREQ}}$/HREQ	$\overline{\text{HTRQ}}$/HTRQ	PB14
HACK/ HRRQ	$\overline{\text{HACK}}$/HACK	$\overline{\text{HRRQ}}$/HRRQ	PB15

6.4 HI08 BLOCK DIAGRAM

Figure 6-1 shows the HI08 registers. The DSP core can access the top row of registers (HCR, HSR, HDDR, HDR, HBAR, HPCR, HTX, HRX) can be accessed by the DSP core. The host processor can access the bottom row of registers (ISR, ICR, CVR, IVR, RXH:RXM:RXL, and TXH:TXM:TXL).

HCR = host control register
HSR = host status register
HPCR = host port control register
HBAR = host base address register

HTX = host transmit register
HRX = host receive register
HDDR = host data direction register
HDR = host data register



ICR = interface control register
CVR = command vector register
ISR = interface status register
IVR = interrupt vector register
RXH = receive register high

RXM = receive register middle
RXL = receive register low
TXH = transmit register high
TXM = transmit register middle
TXL = transmit register low

AA0657

Figure 6-1 HI08 Block Diagram

6.5 HI08—DSP-SIDE PROGRAMMER'S MODEL

The DSP56307 core treats the HI08 as a memory-mapped peripheral occupying eight 24-bit words in X data memory space. The DSP can use 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 DSP56307 core to communicate with the HI08 registers using standard instructions and addressing modes. In addition, the MOVEP instruction allows direct data transfers between DSP56307 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.

The following are data registers, 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 DSP56307 needs the eight MSBs in the DSP-side control registers as 0. These registers are the following:

- 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 and software resets disable the HI08. After a reset, the HI08 signals are configured as GPIO and disconnected from the DSP56307 core (i.e., the signals are left floating).

6.5.1 Host Receive Data Register (HRX)

The HRX register performs host-to-DSP data transfers.The DSP56307 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 transmit data register empty (TXDE (ISR, Bit 1), on the host side) and host receive data full (HRDF (HSR, Bit 0) on the DSP side) 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 DSP56307 may set the HRIE bit (HCR, Bit 0) to cause a host receive data interrupt when HRDF is set. When the DSP56307 reads the HRX register, the HRDF bit is cleared.

6.5.2 Host Transmit Data Register (HTX)

The HTX register performs DSP-to-host data transfers. The DSP56307 views it as a 24-bit write-only register. Its address is X:$FFFFC7. Writing to the HTX register clears the host transfer data empty bit (HTDE (HSR Bit 1), on the DSP side). 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 receive data full (RXDF (ISR, Bit 0), on the host side) bits are cleared. This transfer operation sets the RXDF and HTDE bits. The DSP56307 can set 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: 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 the user reads any of those status bits within the next two cycles, the bit will 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. The HCR bits are described in the following paragraphs. Initialization values for HCR bits are documented in **Section 6.5.9 DSP-Side Registers after Reset** on page 6-19. 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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

▢—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

When set, 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 transferred to the HRX from the TXH, TXM, or TXL registers. If HRIE is cleared, HRDF interrupts are disabled.

6.5.3.2 HCR Host Transmit Interrupt Enable (HTIE) Bit 1

When set, 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 transferred from the HTX to the RXH, RXM, or RXL registers. If HTIE is cleared, HTDE interrupts are disabled.

6.5.3.3 HCR Host Command Interrupt Enable (HCIE) Bit 2

When set, 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 function as general-purpose flags for DSP-to-host communication. HF[3:2] can be set or cleared by the DSP core. 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 general-purpose 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 HIO8 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 documented in **Section 6.5.9 DSP-Side Registers after Reset** on page 6-19. The HSR bits are documented in the following paragraphs.

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 HRX is read by the DSP core. HRDF can also be cleared by the host processor using 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 can also be set by the host processor using 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] may 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.

These two general-purpose flags 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 to select 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. An internal chip select is generated if a match is found. **Figure 6-5** shows how the chip-select logic uses this register.

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

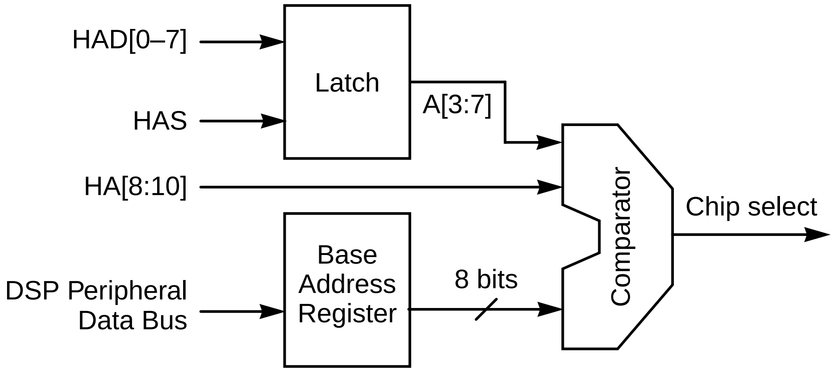
Figure 6-4 Host Base Address Register (HBAR) (X:$FFFFC5)

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.



AA0666

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 given in **Section 6.5.9 DSP-Side Registers after Reset** on page 6-19. The HPCR bits are documented in the following paragraphs.

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 DSP56307, the HPCR bits HAP, HRP, HCSP, HDDS, HMUX, HASP, HDSP, HROD, HAEN, and HREN should be changed only if HEN is cleared.

Likewise, 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 nor simultaneously with HEN being set.

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, 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 used in multiplexed bus mode, then HA8/A1 is used as host address line 8 (HA8). If this bit is cleared and the HI08 is used in multiplexed bus mode, then HA8/HA1 is used 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 (i.e., when HMUX is cleared).

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

If HA9EN is set and the HI08 is used in multiplexed bus mode, then HA9/HA2 is used as host address line 9 (HA9). If this bit is cleared, and the HI08 is used 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 (i.e., when 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 nonmultiplexed bus mode (i.e., when HMUX is cleared), and as host address line 10 (HA10) in the multiplexed bus mode (i.e., when 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 (i.e., if HDRQ is cleared in the host interface control register (ICR)), then 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 (i.e., if HDRQ is set in the ICR), then 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 (i.e., when HDRQ is cleared in ICR), if HROD is cleared and host requests are enabled (i.e., if HREN is set and HEN is set in the host port control register (HPCR)), then 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 (i.e., when HDRQ is set in the ICR), if HROD is cleared and host requests are enabled (i.e., if HREN is set and HEN is set in the HPCR), then 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

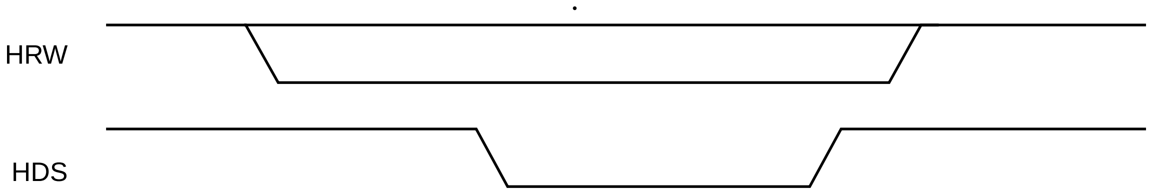
If HASP is cleared, the host address strobe (HAS) signal is an active low input, and the address on the host address/data bus is sampled when the HAS signal is low. If HASP is set, HAS is an active-high address strobe input, and the address on the host address or data bus is sampled when the HAS signal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

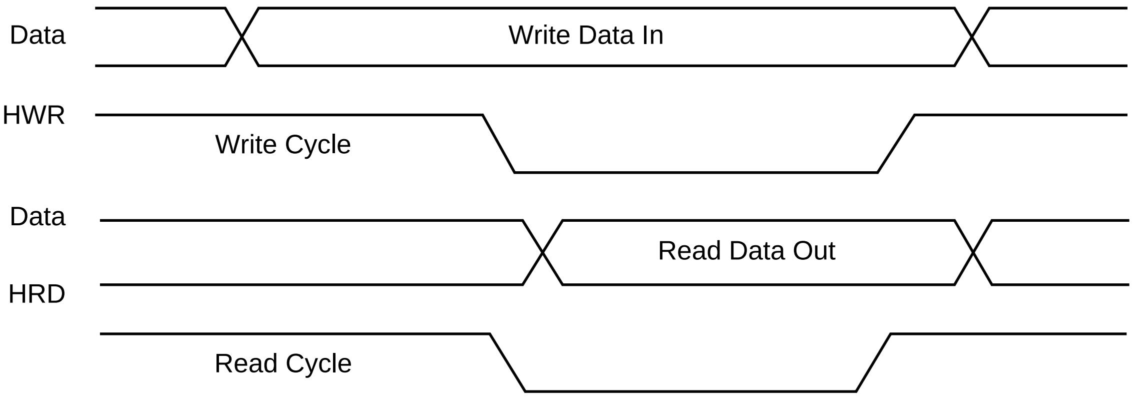
If the HDDS bit is cleared, the HI08 operates in single strobe bus mode. In this mode, the bus has a single data strobe signal for both reads and writes. If the HDDS bit is set, the HI08 operates in dual strobe bus mode. In this mode, the bus has two separate data strobes: one for data reads, the other for data writes. See **Figure 6-7** and **Figure 6-8** for more information about 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 single host request mode (i.e., when HDRQ is cleared in the ICR), if HRP is cleared and host requests are enabled (i.e., if HREN is set and HEN is set), then 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 (i.e., when HDRQ is set in the ICR), if HRP is cleared and host requests are enabled (i.e., if HREN is set and HEN is set), then 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. Even when the HI08 functions as the host interface, its unused signals can be configured as GPIO signals. For information about the HI08 GPIO configuration options, see **Section 6.6.8 GPIO** on page 6-30. If Bit DR*xx* is set, the corresponding HI08 signal is configured as an output signal. If Bit DR*xx* 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. The functionality of Bit D*xx* depends on the corresponding HDDR bit (i.e., DR*xx*). 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	D*xx*	
	GPIO Signal[a]	Non-GPIO Signal[a]
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.

a. defined by the selected configuration

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 to the DSP56307. The hardware reset (HW) is caused by the $\overline{\text{RESET}}$ signal. The software reset (SW) is caused by execution of the RESET instruction. The individual reset (IR) is caused by the HEN bit (HPCR Bit 6) being cleared. The stop reset (ST) is caused by execution of 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	—[a]	—
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

Note: a. The bit value is indeterminate after reset.

6.5.10 Host Interface DSP Core Interrupts

The HI08 can request interrupt service from either the DSP56307 or the host processor. The DSP56307 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 DSP56307. The DSP56307 acknowledges interrupts caused by the host processor by jumping to the appropriate interrupt service routine. The following interrupts are possible.

- 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. In the case of host command interrupts, the interrupt acknowledge from the DSP56307 program controller clears the pending interrupt condition.

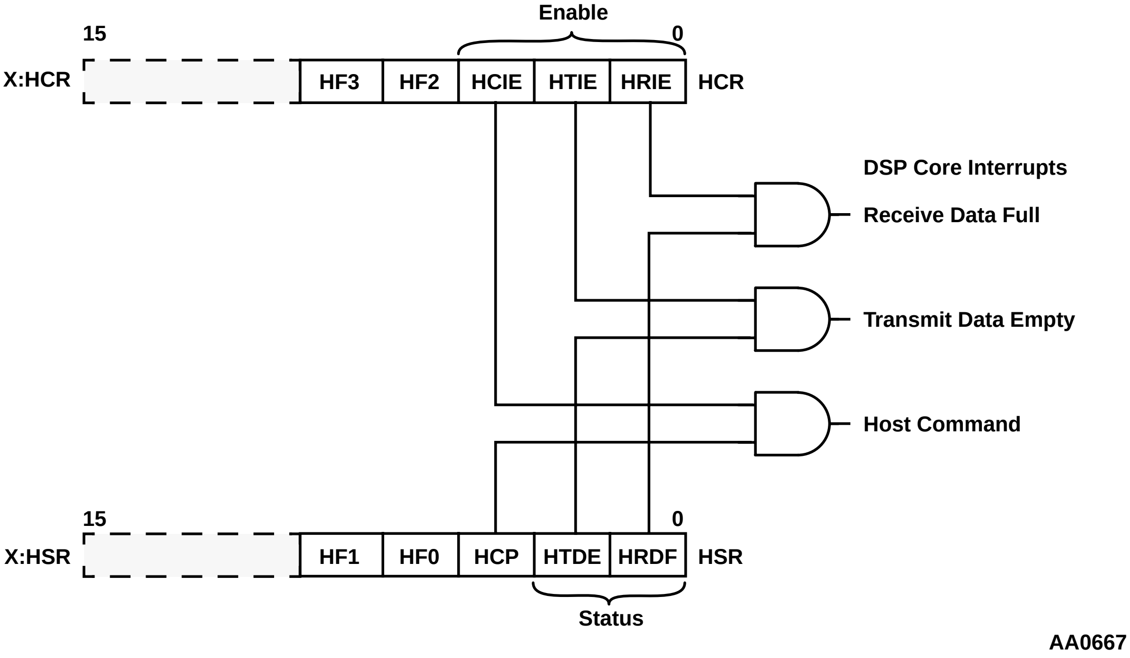


Figure 6-11 HSR-HCR Operation

6.6 HI08—EXTERNAL HOST PROGRAMMER'S MODEL

The HI08 provides 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. (See **Table 6-7**.)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 DSP56307. Only by 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-, or 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 DSP56307 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 DSP56307 as a fast device, and data can be transferred between the host processor and the DSP56307 at the fastest data rate of the host processor.

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 DSP56307. 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 DSP56307. For example, the DSP56307 host interrupts 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 or 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.

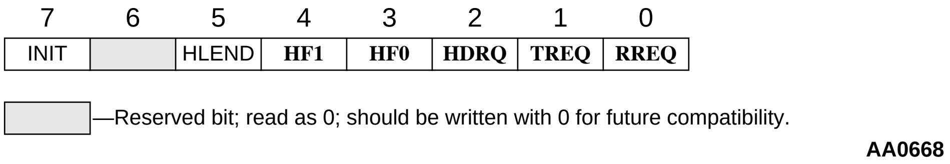
Do not issue a STOP command to the DSP via the HI08 unless you provide some other mechanism to exit stop mode.

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. 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.



7	6	5	4	3	2	1	0
INIT	▨	HLEND	**HF1**	**HF0**	**HDRQ**	**TREQ**	**RREQ**

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

AA0668

Figure 6-12 Interface Control Register (ICR)

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 is 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 Signal	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 is configured as HTRQ, and HACK/HRRQ is configured as HRRQ.

6.6.1.4 ICR Host Flag 0 (HF0) Bit 3

The HF0 bit is a general-purpose flag for host-to-DSP communication. HF0 can be set or cleared by the host processor and cannot be changed by the DSP56307. 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. HF1 can be set or cleared by the host processor and cannot be changed by the DSP56307. 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.

It may or may not be necessary to use the INIT bit to initialize the HI08 hardware, depending on the software design of the interface.

The type of initialization 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 conveniently configures the HI08 into the desired data transfer mode. **Table 6-10** shows the effect of the INIT command. When the host sets the INIT bit, the HI08 hardware executes the INIT command. The interface hardware clears the INIT bit after the command executes.

Table 6-10 INIT Command Effects

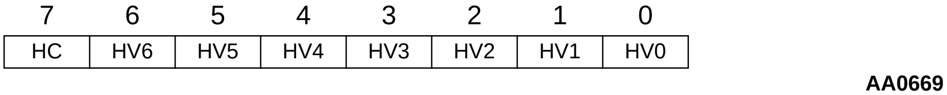
TREQ	RREQ	After INIT Execution	Transfer Direction Initialized
0	0	INIT = 0	None
0	1	INIT = 0; RXDF = 0; HTDE = 1	DSP to host
1	0	INIT = 0; TXDE = 1; HRDF = 0	Host to DSP

Table 6-10 INIT Command Effects

TREQ	RREQ	After INIT Execution	Transfer Direction Initialized
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 DSP56307 to execute an interrupt. The host command feature is independent of any of the data transfer mechanisms in the HI08. It can cause execution of any of the 128 possible interrupt routines in the DSP core.



7	6	5	4	3	2	1	0
HC	HV6	HV5	HV4	HV3	HV2	HV1	HV0

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 2 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 hardware, software, individual, and stop resets.

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 DSP56307. When the host command interrupt is acknowledged by the DSP56307, HIO8 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 HIO8 hardware clears the 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 host processor uses the ISR, an 8-bit read-only status register, 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 ISR cannot be accessed by the DSP core. The following paragraphs document the ISR bits.

7	6	5	4	3	2	1	0
HREQ	▨	▨	HF3	HF2	TRDY	TXDE	RXDF

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

AA0670

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 DSP56307 and may 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 register (RXL or RXH according to HLEND bit) is read by the host processor. The host processor can clear RXDF 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 register (TXL or TXH according to HLEND bit) is written by the host processor. The host processor can set TXDE using the initialize function. TXDE may be used to 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 polling techniques may be used by the host processor).

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.

CAUTION

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 that causes the DSP56307 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 DSP56307.

6.6.3.4 ISR Host Flag 2 (HF2) Bit 3

The HF2 bit in the ISR indicates the state of HF2 in the HCR on the DSP side. HF2 can be changed only by the DSP56307. See **Section 6.5.3.4 HCR Host Flags 2, 3 (HF[3:2]) Bits 3, 4** on page 6-10 for related information.

6.6.3.5 ISR Host Flag 3 (HF3) Bit 4

The HF3 bit in the ISR indicates the state of HF3 in the HCR on the DSP side. HF3 can be changed only by the DSP56307. See **Section 6.5.3.4 HCR Host Flags 2, 3 (HF[3:2]) Bits 3, 4** on page 6-10 for related information.

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

If HDRQ is set, the HREQ bit indicates the status of the external transmit and receive request output signals (HTRQ and HRRQ). If HDRQ is cleared, it indicates the status of the external host request output signal (HREQ).

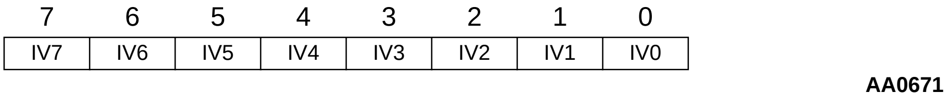
Table 6-11 HREQ and HDRQ Settings

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 bit can be 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 status bits: ISR RXDF indicates that the receive byte registers are full, and ISR TXDE indicates that the transmit byte registers are empty. If the interrupt source is 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 that 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 or software reset. This value corresponds to the uninitialized interrupt vector in the MC68000 family.



7	6	5	4	3	2	1	0
IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0

Figure 6-15 Interrupt Vector Register (IVR)

6.6.5 Receive Byte Registers (RXH: RXM: RXL)

The host processor views the receive byte registers 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 are 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 transferred from the HTX register to the receive byte register at host address $7, the receive data register full (RXDF) bit is set. The host processor may 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 may program the RREQ bit to assert the external HREQ signal when RXDF is set. Assertion of 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 can be written into the transmit byte registers when the transmit data register empty (TXDE) bit is set. The host processor can program 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 will 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. To cause a hardware reset, assert the $\overline{\text{RESET}}$ signal. To cause a software reset, execute the RESET instruction. To reset the HEN bit individually, clear the HEN bit in the HPCR. To cause a stop reset, execute 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 GPIO

When configured as GPIO, the HI08 is viewed by the DSP56307 as memory-mapped registers (as in **Section 6.5**) that control up to 16 I/O signals. Software and hardware resets 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. You can select either GPIO or HI08: to select

GPIO functions, clear HPCR bits 6 through 1; to select other HI08 functions, set those same bits. 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. For more details, see **Sections 6.5.7 Host Data Direction Register (HDDR)** on page 6-17 and **6.5.8 Host Data Register (HDR)** on page 6-18.

6.7 SERVICING THE HOST INTERFACE

The HI08 can be serviced by 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 appears as a contiguous block of SRAM. To transfer data between itself and the HI08, the host processor performs the following steps:

1. The host processor asserts the HI08 address to select the register to be read or written. Chip select is in non-mux mode. The address strobe is in mux mode.
2. The host processor selects the direction of the data transfer. If it is writing, the host processor sources the data on the bus.
3. The host processor 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; see **Section 6.5.6 Host Port Control Register (HPCR)** on page 6-13 for more about that topic.

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

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

- If RXDF is set, the receive data register is full, and a data read can be performed by the host processor.
- If TXDE is set, the transmit data register is empty. A data write can be performed by the host processor.
- 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 condition may indicate that an application-specific state within the DSP56307 has been reached. Intervention by the host processor may be required.
- If HREQ is set, the HREQ/TRQ signal has been asserted, and the DSP56307 is requesting the attention of the host processor. One of the other four conditions exists.

If the host processor issues a command to the DSP56307 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 is accepted for execution, the HC bit is cleared by the interrupt controller in the DSP core.

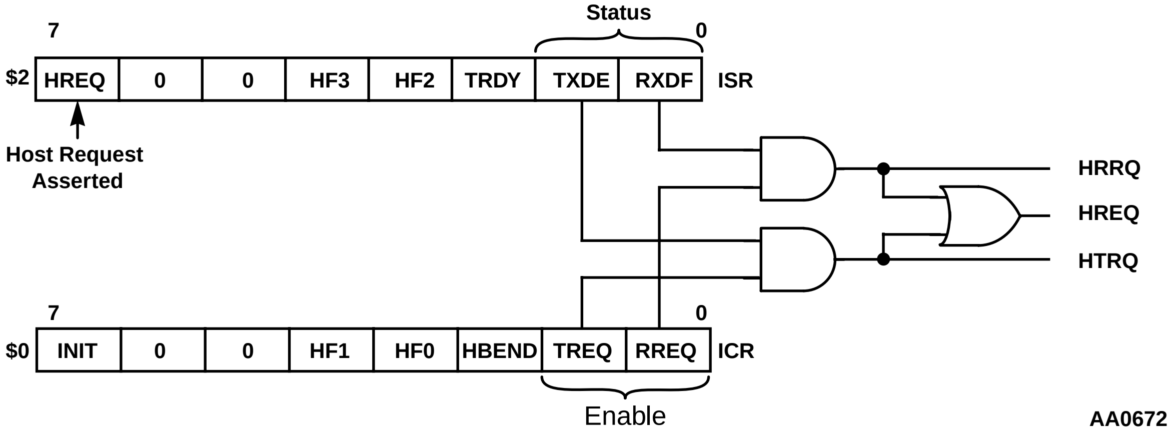


Figure 6-16 HI08 Host Request Structure

6.7.3 Servicing Interrupts

If either the HREQ/HTRQ or HRRQ signal or both are connected to the interrupt input of a host processor, the HI08 can request service from that host processor by asserting one of these signals. **Figure 6-16** illustrates several possibilities:

- When TXDE is set and the corresponding enabling bit (TREQ) is set, then HTRQ is asserted.
- When RXDF is set and the corresponding enabling bit (RREQ) is set, then HRRQ is asserted.
- When both TXDE and RXDF as well as their corresponding enabling bits (TREQ and RREQ) are all set, then HREQ is asserted.

HREQ normally connects to the maskable interrupt input of the host processor. The host processor acknowledges host interrupts by executing an interrupt service routine. The two LSBs (RXDF and TXDE) of the ISR register may be tested by the host processor to determine the interrupt source, as indicated 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 DSP56307. Instead, the DSP56307 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 DSP56307.

6.8 HI08 PROGRAMMING MODEL—QUICK REFERENCE

The HI08 programming model consists of a DSP side and a host side.

Table 6-13 shows the DSP-side registers.

Table 6-13 Host Control Register (HCR)

Bit #	Bit Name	Bit Values	Function	Value After Reset Type:		
				H/W or S/W	Ind.	STOP
0	Host Receive Interrupt Enable (HRIE)	0 1	HRRQ Interrupt Disabled HRRQ Interrupt Enabled	0	—	—
1	Host Transmit Interrupt Enable (HTIE)	0 1	HTRQ Interrupt Disabled HTRQ Interrupt Enabled	0	—	—
2	Host Command Interrupt Enable (HCIE)	0 1	HCP Interrupt Disabled HCP Interrupt Enabled	0	—	—
3	Host Flag 2	General purpose flags defined by user software.		0		
4	Host Flag 3			0	—	—

Table 6-14 shows the DSP-side programming model. **Table 6-15** beginning on page 6-38 shows the host-side programming model.

Table 6-14 HI08 Programming Model: DSP Side

Reg	Bit					Comments	Reset Type		
	#		Name	Value	Function		HW/ SW	IR	ST
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 6-14 HI08 Programming Model: DSP Side

Reg	Bit					Comments	Reset Type		
	#		Name	Value	Function		HW/ SW	IR	ST
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 6-14 HI08 Programming Model: DSP Side

Reg	Bit					Comments	Reset Type		
	#		Name	Value	Function		HW/ SW	IR	ST
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 6-15 HI08 Programming Model: Host Side

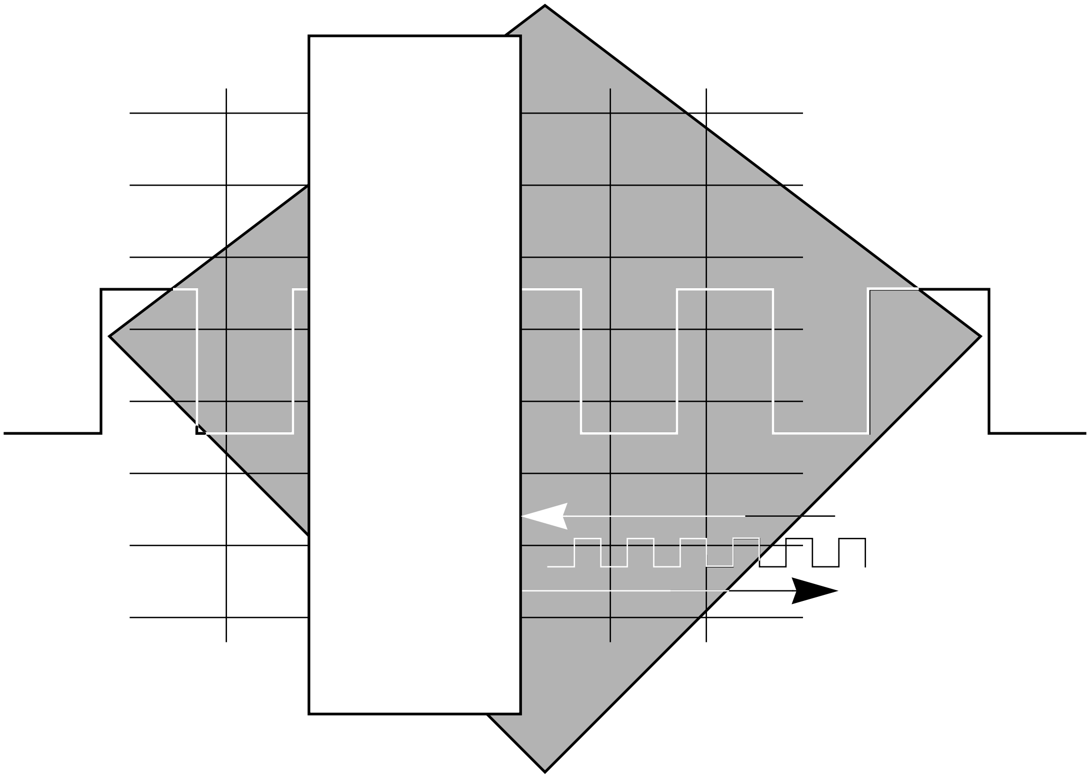
Reg	Bit					Comments	Reset Type		
	#		Name	Value	Function		HW/ SW	IR	ST
ICR	0	RREQ	Receive Request Enable	0 1	HRRQ interrupt disabled HRRQ interrupt enabled		0	—	—
	1	TREQ	Transmit Request Enable	0 1	HTRQ interrupt disabled HTRQ interrupt enabled		0	—	—
	2	HDRQ	Double Host Request	0 1	HREQ/HTRQ = HREQ, 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	—	—
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

Table 6-15 HI08 Programming Model: Host Side

Reg	Bit					Comments	Reset Type		
	#		Name	Value	Function		HW/ SW	IR	ST
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



7.1 INTRODUCTION

The 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 DSP56307: ESSI0 and ESSI1. For the sake of simplicity, a single generic ESSI is described here.

The ESSI block diagram is shown in **Figure 7-1**. This interface is synchronous because all serial transfers are synchronized to one clock.

Note: This synchronous interface should not be confused with the asynchronous channels mode of the ESSI, in which separate clocks are used for the receiver and transmitter. In that 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. The normal mode of operation is used to transfer data at a periodic rate, one word per period. The network mode is similar in that it is also intended for periodic transfers; however, it supports up to 32 words (time slots) per period. The network mode can be used to build time division multiplexed (TDM) networks. In contrast, the on-demand mode is intended for nonperiodic 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 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)
  - End-of-frame interrupt
  - Drive enable signal (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
  - Control register A prescaler range (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 Signal (STD)

The STD signal transmits data from the 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 is transmitted if another data word does not follow immediately. If sequential data words are 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 is use.

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 in use.

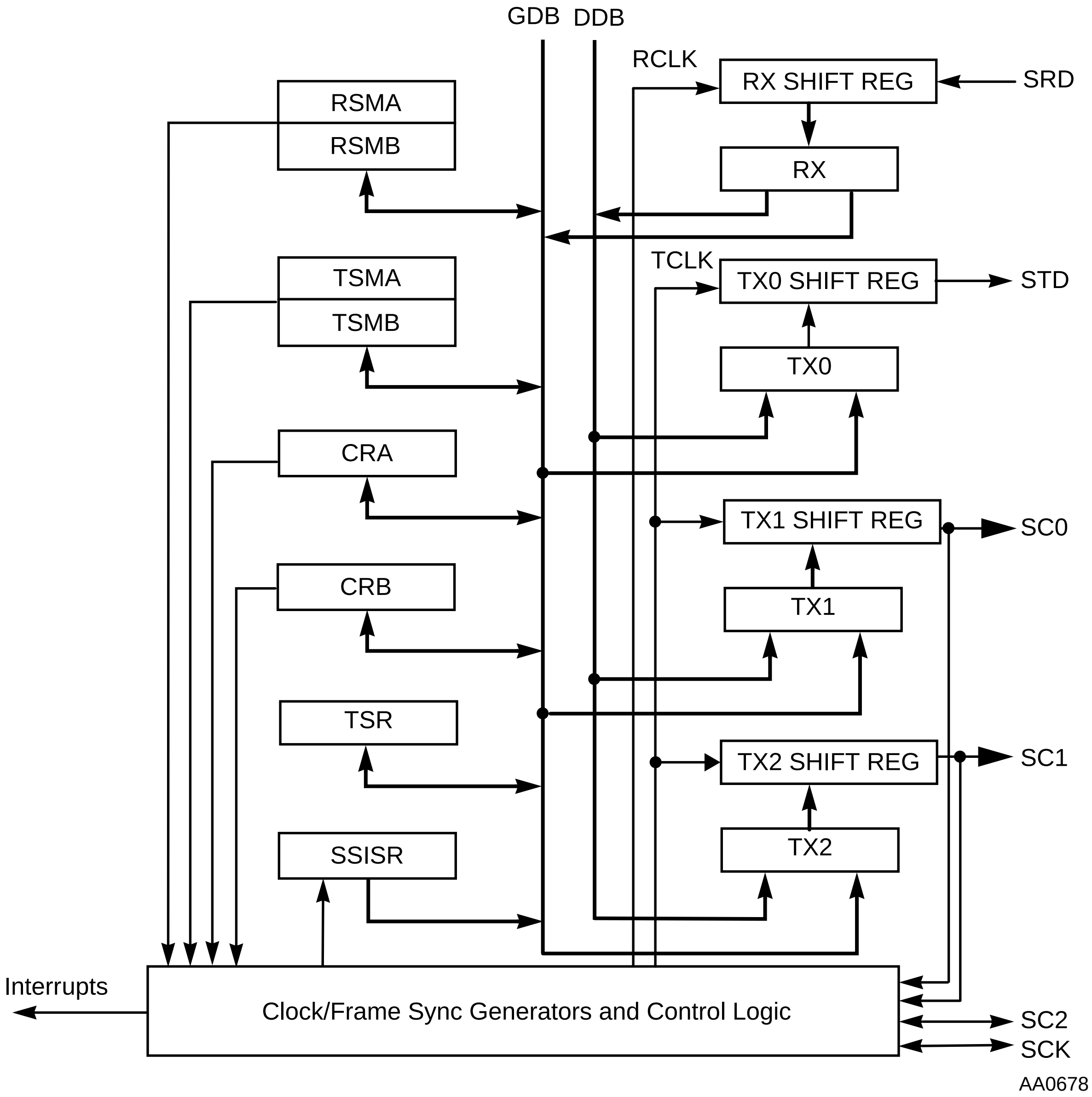


Figure 7-1 ESSI Block Diagram

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 for details. SCK can be programmed as a GPIO signal (P3) when the ESSI SCK function is not in use.

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)

ESSI0: SC00; ESSI1: SC10

The programmer determines the function of this signal by selecting either synchronous or asynchronous mode, according to **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 TX1 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 or receive clock signal, its direction is determined by the serial control direction 0 (SCD0) bit in ESSI control register B (CRB). When configured as an output, SC0 functions as the serial output flag 0 (OF0) or as a receive shift register clock output. If SC0 is used as the serial output flag 0, its value is determined by the value of the serial output flag 0 (OF0) bit in the CRB.

If SC0 is an input, it functions as either serial Input Flag 0 or a receive shift register clock input. As serial input flag 0, SC0 controls the state of the 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).

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

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

7.3.5 Serial Control Signal (SC1)

ESSI0:SC01; ESSI1: SCI11

The programmer determines the function of this signal by selecting either synchronous or asynchronous mode, according to**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 transmitter 0 drive-enabled signal, or for serial flag I/O.

When used as serial flag I/O, it operates like SC0. SC0 and SC1 are independent flags, but can be used together for multiple serial device selection. SC0 and SC1 can be unencoded to select up to two codecs or 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.

If SC1 is configured as a serial flag or receive frame sync signal, its direction is determined by the serial control direction 1 (SCD1) bit in the CRB.

When configured as an output, the SC1 signal functions as a serial output flag, as the transmitter 0 drive-enabled signal, or as the receive frame sync signal output. If SC1 is used as serial output flag 1, its value is determined by the value of the serial output flag 1 (OF1) bit in the CRB.

When configured as an input, this signal can receive frame sync signals from an external source, or it acts as a serial input flag. As a serial input flag, SC1controls status bit IF1 in the SSISR.

When this signal 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).

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

Table 7-1 ESSI Clock Sources

SYN	SCKD	SCD0	RX Clock Source	RX Clock Out	TX 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)

ESSI0:SC02; ESSI1:SC02

This signal is used for frame sync I/O. The frame sync is SC2 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 both the transmitter and receiver when in synchronous mode.

SC2 can be programmed as a GPIO signal (P2) when the ESSI SC2 function is not in use.

7.4 ESSI PROGRAMMING MODEL

The ESSI is composed of the following registers:

- Two control registers (CRA, CRB)
- One status register (SSISR)
- Three transmit data registers (TX0, TX1, TX2)
- One receive data register (RX)
- Two transmit slot mask registers (TSMA, TSMB)
- Two receive slot mask registers (RSMA, RSMB)
- One special-purpose time slot register (TSR)

The following paragraphs document each of the bits in the ESSI registers. **Section 7.6** documents the GPIO of the ESSI.

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)

11	10	9	8	7	6	5	4	3	2	1	0
TS11	TS10	TS9	TS8	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	TS15	TS14	TS13	TS12

AA0860

Figure 7-5 ESSI Transmit Slot Mask Register A (TSMA)
(ESSI0 X:$FFFFB4, ESSI1 X:$FFFFA4)

11	10	9	8	7	6	5	4	3	2	1	0
TS27	TS26	TS25	TS24	TS23	TS22	TS21	TS20	TS19	TS18	TS17	TS16

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	TS31	TS30	TS29	TS28

AA0861

Figure 7-6 ESSI Transmit Slot Mask Register B (TSMB)
(ESSI0 X:$FFFFB3, ESSI1 X:$FFFFA3)

11	10	9	8	7	6	5	4	3	2	1	0
RS11	RS10	RS9	RS8	RS7	RS6	RS5	RS4	RS3	RS2	RS1	RS0

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	RS15	RS14	RS13	RS12

AA0862

Figure 7-7 ESSI Receive Slot Mask Register A (RSMA)
(ESSI0 X:$FFFFB2, ESSI1 X:$FFFFA2)

11	10	9	8	7	6	5	4	3	2	1	0
RS27	RS26	RS25	RS24	RS23	RS22	RS21	RS20	RS19	RS18	RS17	RS16

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	RS31	RS30	RS29	RS28

▨ – Reserved bit; read as zero; should be written with zero for future compatibility

AA0863

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. CRA controls the ESSI clock generator bit and frame sync rates, word length, and number of words per frame for serial data. The CRA control bits are documented in the following paragraphs and in **Figure 7-2**.

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 DSP56307 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 generates 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 needed. When PSR is set, the fixed prescaler is bypassed. When PSR is cleared, the fixed divide-by-eight prescaler is operational, as in **Figure 7-9**.

This definition is reversed from that of the SSI in other members of the DSP56000 family.

The maximum allowed internally generated bit clock frequency is the internal DSP56307 clock frequency divided by 4; the minimum possible internally generated bit clock frequency is the DSP56307 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 thus should not be used.

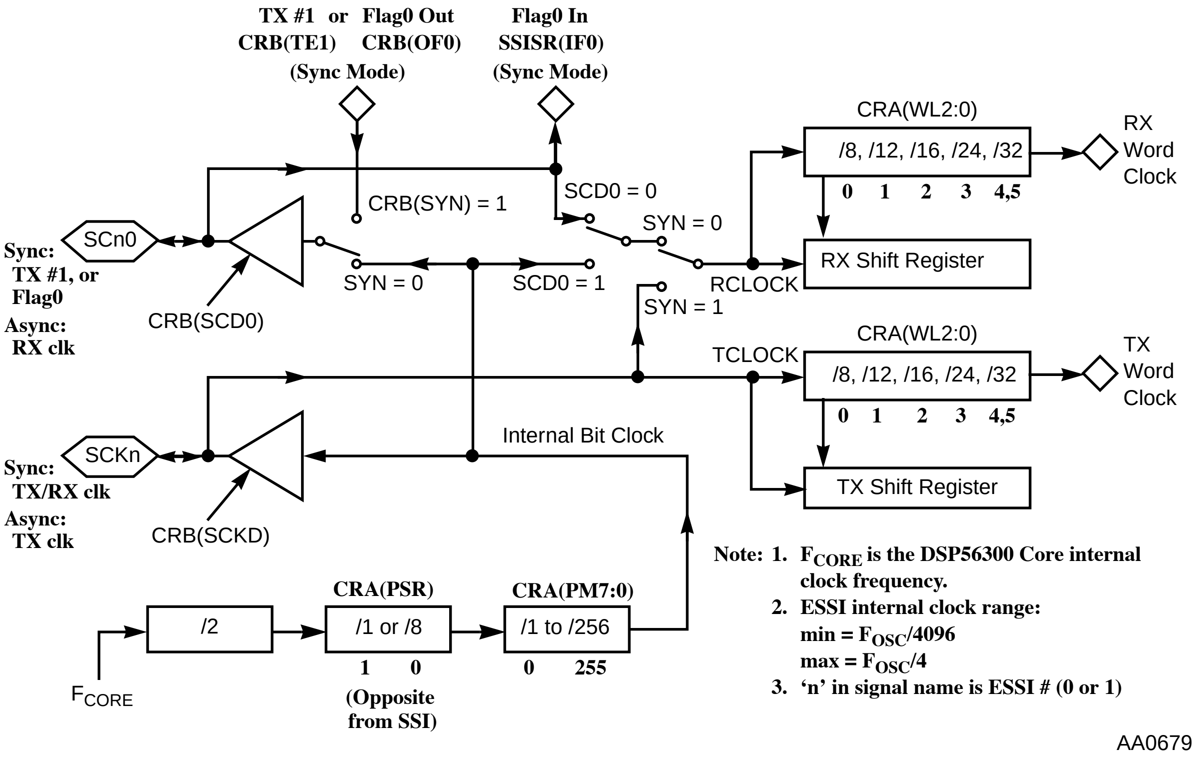


Figure 7-9 ESSI Clock Generator Functional Block Diagram

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; you select it 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**.

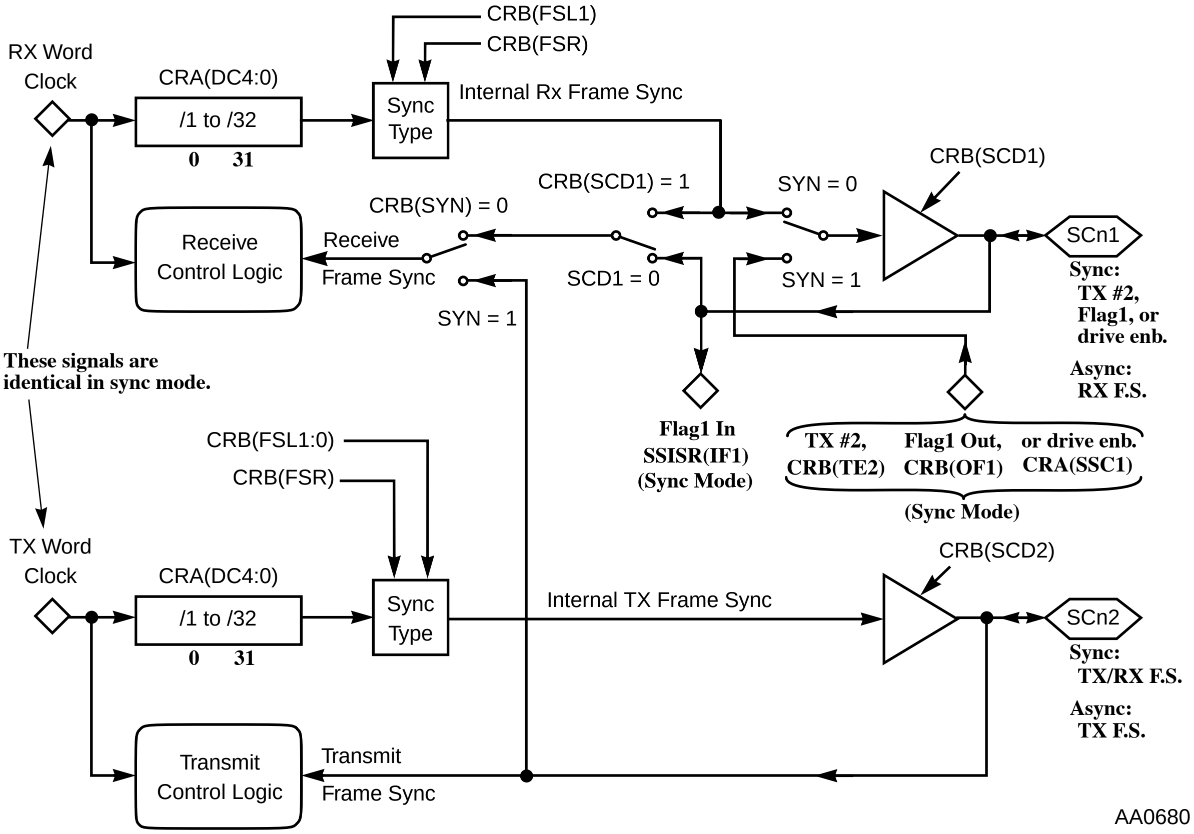


Figure 7-10 ESSI Frame Sync Generator Functional Block Diagram

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 handles 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 select the length of the data words 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** and **Figure 7-17** shows additional information about how to select different length data words. The ESSI data registers are 24 bits long. The ESSI transmits 32-bit words either by duplicating the last bit 8 times when WL[2:0] = 100, or by duplicating the first bit 8 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. If SSC1 is set, the ESSI is configured in Synchronous mode (the CRB synchronous/asynchronous bit (SYN) is set), and transmitter 2 is disabled (transmit enable (TE2) = 0)), then the SC1 signal acts as the transmitter 0 driver-enabled signal while the SC1 signal is configured as output (SCD1 = 1). This configuration enables an external buffer for the transmitter 0 output.

If SSC1 is cleared, the ESSI is configured in synchronous mode (SYN = 1), and transmitter 2 is disabled (TE2 = 0), then the SC1 acts as the serial I/O flag while the SC1 signal is configured as output (SCD1 = 1).

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)

Control Register B (CRB) is one of two 24-bit read/write control registers that direct the operation of the ESSI, as shown in **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 clear 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 follows for setting output flags when transmitting data (by transmitter 0 through the STD signal only).

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 bit 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 that clocks the transmit shift register in asynchronous mode and both the transmit and receive shift registers in synchronous mode. 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 may 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 SHFD bit determines the shift direction of the transmit or receive shift register. If SHFD is set, data is shifted in and out with the LSB first. If SHFD is cleared, data is shifted in and out with the MSB first, as in **Figure 7-16** on page 7-30 and **Figure 7-17** on page 7-31.

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, as in **Figure 7-11** on page 7-19, **Figure 7-14** on page 7-22, and **Figure 7-15** on page 7-23. **Table 7-3** shows the values of FSL[1:0].

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

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; that is, the frame start is indicated by the frame sync signal going high. When FSP is set, the frame sync signal polarity is negative; that is, 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 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.

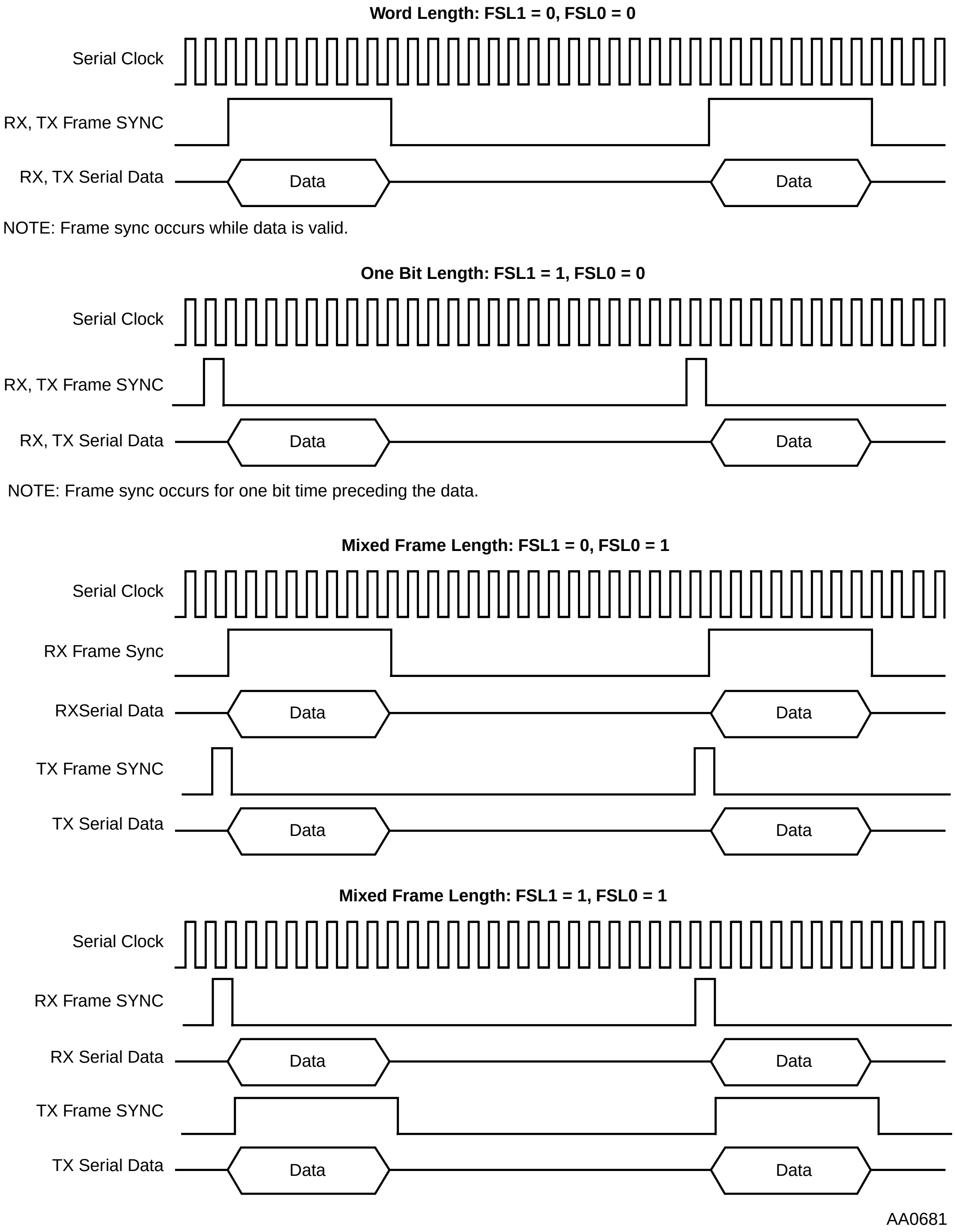


Figure 7-11 CRB FSL0 and FSL1 Bit Operation (FSR = 0)

7.4.2.12 CRB ESSI Mode Select (MOD) Bit 13

MOD selects the operational mode of the ESSI, as in **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, the normal mode is selected; when MOD is set, the 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 details, see **Section 7.5**. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears MOD.

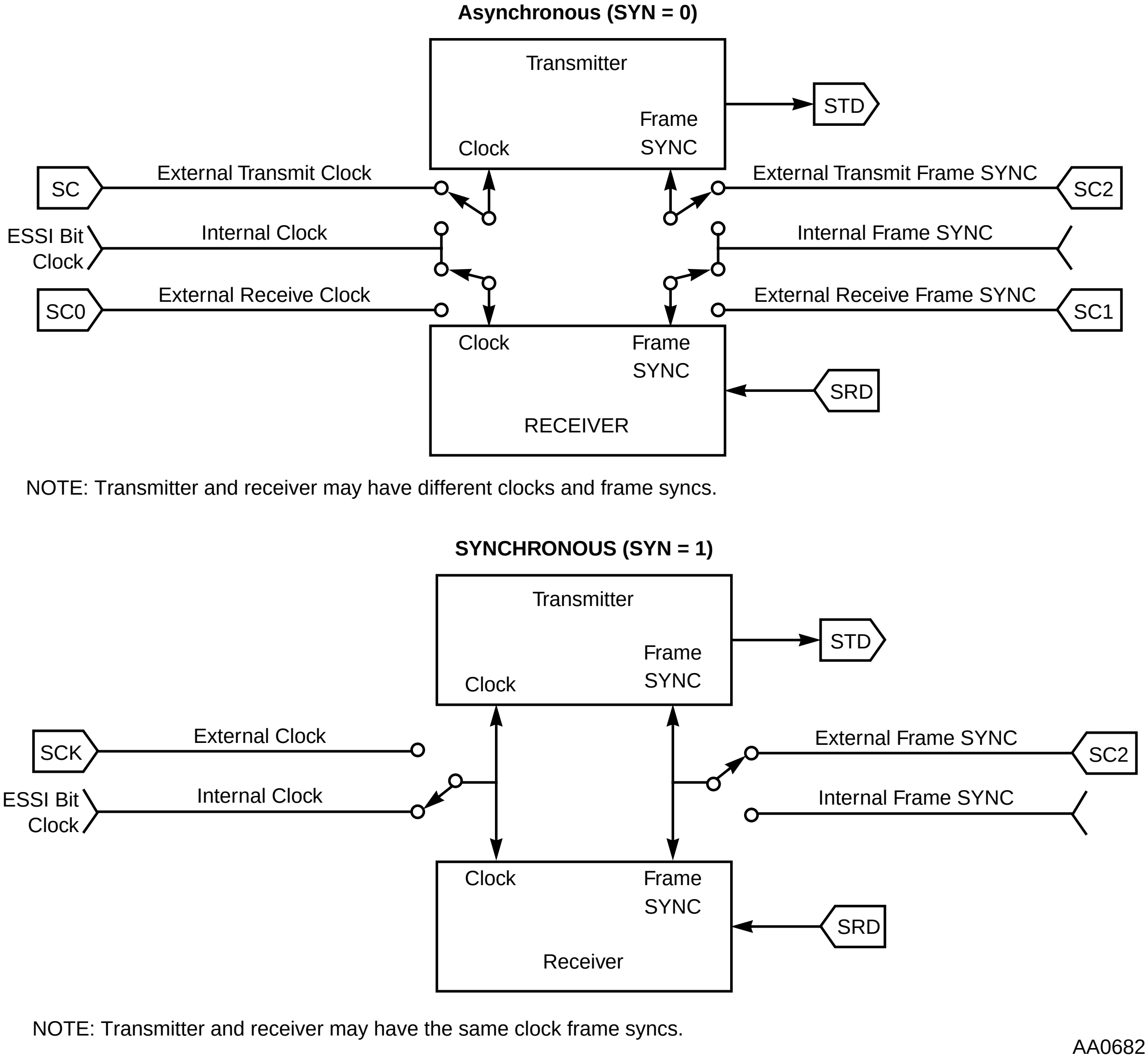


Figure 7-12 CRB SYN Bit Operation

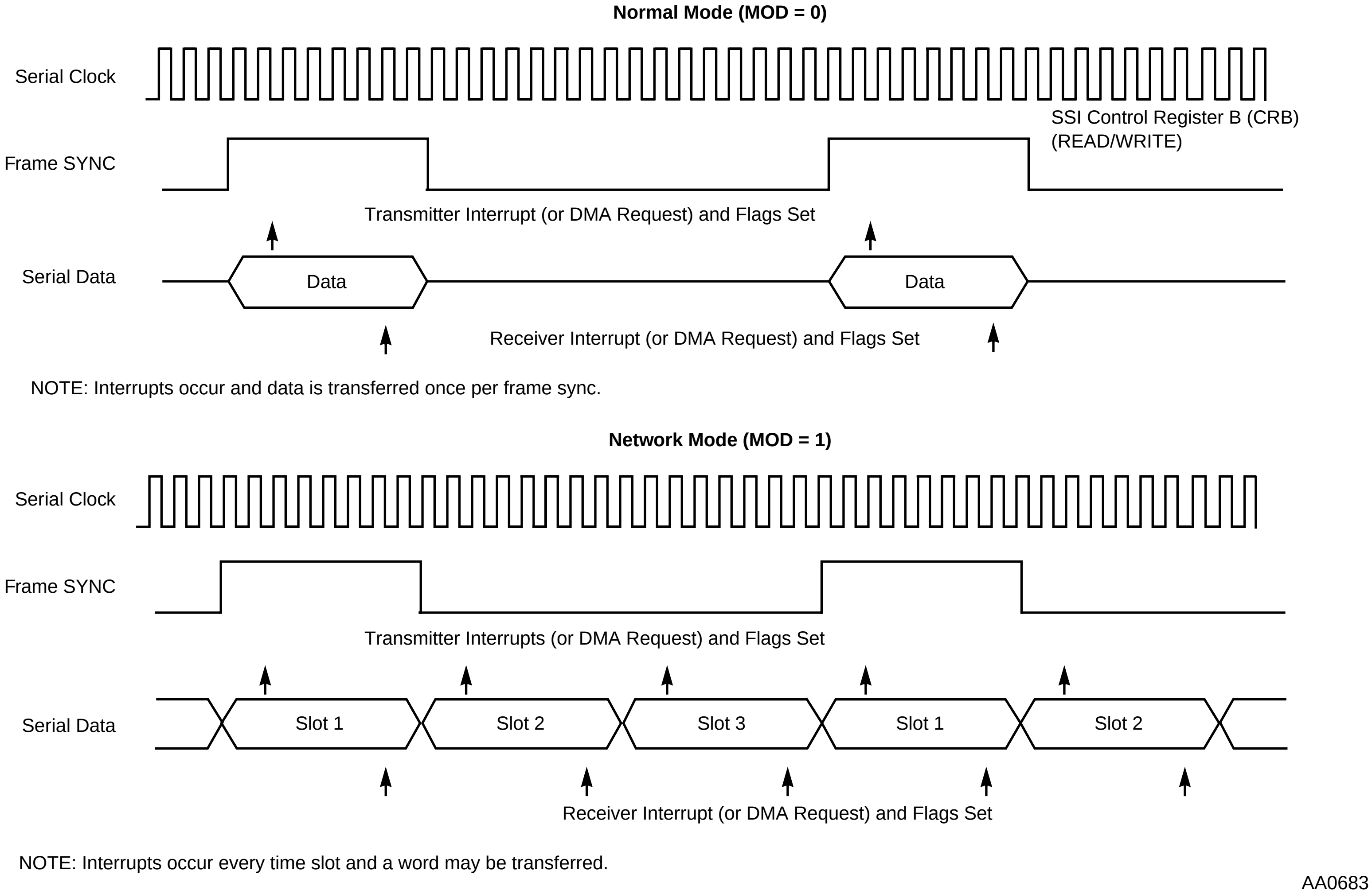

Normal Mode (MOD = 0)
Serial Clock
SSI Control Register B (CRB)
(READ/WRITE)
Frame SYNC
Transmitter Interrupt (or DMA Request) and Flags Set
Serial Data
Data
Data
Receiver Interrupt (or DMA Request) and Flags Set
NOTE: Interrupts occur and data is transferred once per frame sync.
Network Mode (MOD = 1)
Serial Clock
Frame SYNC
Transmitter Interrupts (or DMA Request) and Flags Set
Serial Data
Slot 1
Slot 2
Slot 3
Slot 1
Slot 2
Receiver Interrupt (or DMA Request) and Flags Set
NOTE: Interrupts occur every time slot and a word may be transferred.
AA0683

Figure 7-13 CRB MOD Bit Operation

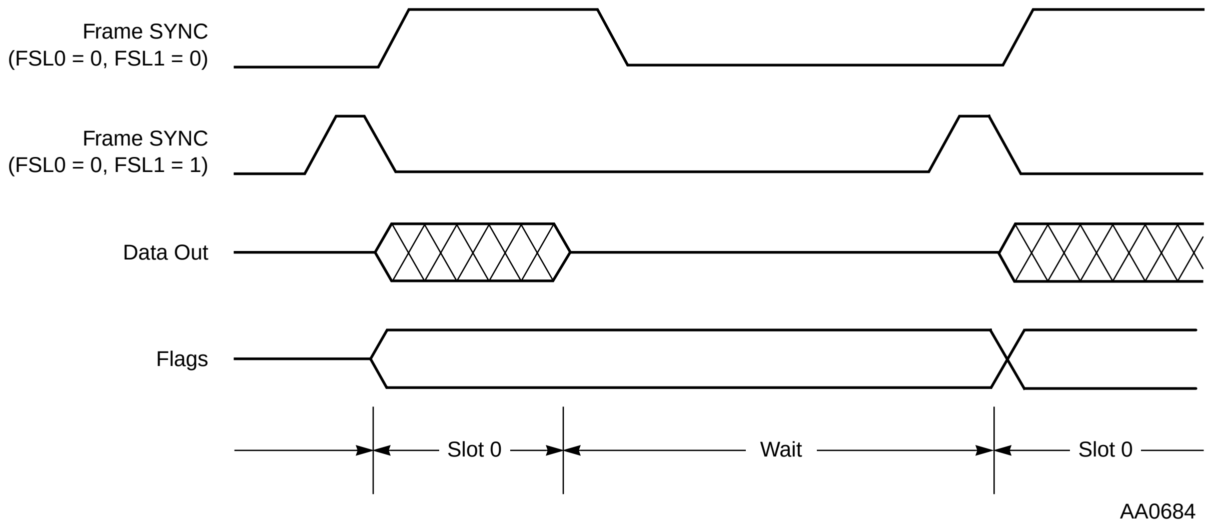


Figure 7-14 Normal Mode, External Frame Sync (8 Bit, 1 Word in Frame)

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 differs from those two bits in that it 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 you set the TE bit. The normal transmit disable sequence is to set the transmit data empty (TDE) bit and then to clear the TE, transmit interrupt enable (TIE), and transmit exception interrupt enable (TEIE) bits. In network mode, if you clear the appropriate TE bit and set it again, then you disable 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 a 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.

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

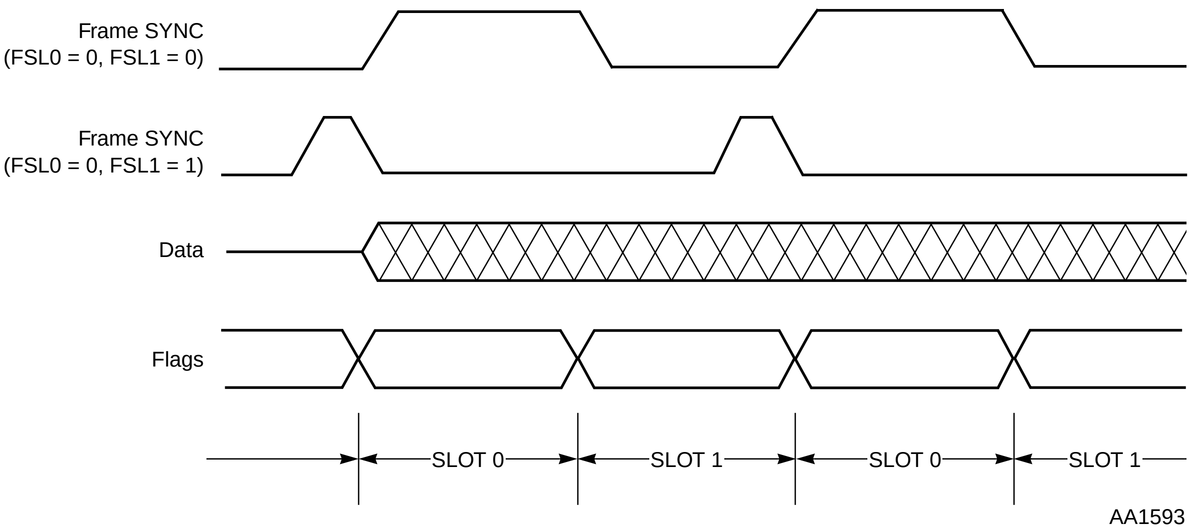


Figure 7-15 Network Mode, External Frame Sync (8 Bit, 2 Words in Frame)

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, 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.

If the TE1 bit is kept cleared until the start of the next frame, it 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 in on-demand mode can be the same as in 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 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 TX1 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 the 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. In other words, 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 in on-demand mode can be the same as in 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 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
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

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	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

TXC	=	Transmitter clock
RXC	=	Receiver clock
XC	=	Transmitter/receiver clock (synchronous operation)
FST	=	Transmitter frame sync
FSR	=	Receiver frame sync
FS	=	Transmitter/receiver frame sync (synchronous operation)
TD0	=	Transmit data signal 0
TD1	=	Transmit data signal 1
TD2	=	Transmit data signal 2
T0D	=	Transmitter 0 drive enable if SSC1 = 1 & SCD1 = 1
RD	=	Receive data
F0	=	Flag 0
F1	=	Flag 1 if SSC1 = 0
U	=	Unused (can be used as GPIO signal)
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: data transfer into RX is inhibited. 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 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

When the TIE bit is set, it enables a DSP transmit interrupt; the interrupt 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 transmit interrupt is documented in **Section 7.5.3**. When data is written to the data registers of the enabled transmitters or to the TSR, it 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

When the RIE bit is set, it enables a DSP receive data interrupt; the interrupt 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 receive interrupt is documented in **Section 7.5.3**. When the receive data register is read, it 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

When the TLIE bit is set, it 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 transmit last slot interrupt is documented in **Section 7.5.3**.

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

When the RLIE bit is set, it 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 documented in **Section 7.5.3**.

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 documented in **Section 7.5.3**. A read of the status register, followed by a write 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 receive interrupt is documented in **Section 7.5.3**. A read of the status register followed by a read of 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 the synchronous mode is selected; that is, when SC0 is programmed as ESSI in the port control register (PCR), the SYN bit is set, and the TE1 and SCD0 bits are cleared.

The ESSI latches any 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 reset all clear 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; that is, when SC1 is programmed as ESSI in the port control register (PCR), the SYN bit is set, and the TE2 and SCD1 bits are cleared.

The ESSI latches any 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 reset all clear 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 (i.e., when TE0, TE1, or TE2 is set).

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

Note: In normal mode, TFS is always read as 1 when data is being transmitted 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. In other words, 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., if the RE bit is set).

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

Note: In normal mode, RFS is always read as 1 when data is read 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 (i.e., there is 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 not written) is retransmitted. In normal mode, there is only one transmit time slot per frame. In network mode, there can be up to 32 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 reset all clear TUE. The programmer can also clear TUE 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 reset all clear ROE. The programmer can also clear ROE by reading the SSISR with the ROE bit set and then reading the RX.

7.4.3.7 SSISR ESSI Transmit Data Register Empty (TDE) Bit 6

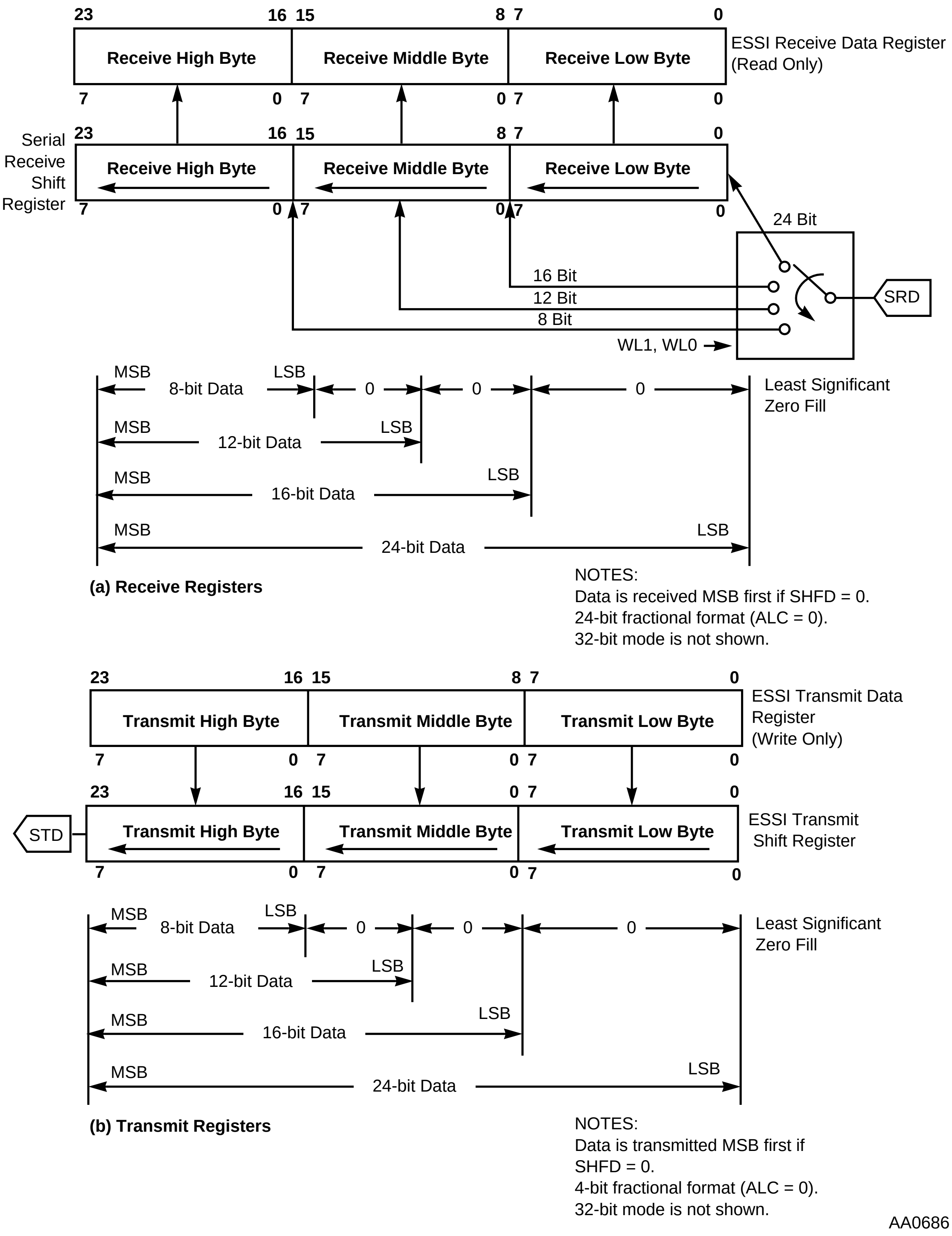
The TDE bit is set when the contents of the transmit data register of every enabled transmitter are transferred to the transmit shift register. It is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR has been written). When set, the TDE bit 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 DSP56307 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 reset all clear the TDE bit.

7.4.3.8 SSISR 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.

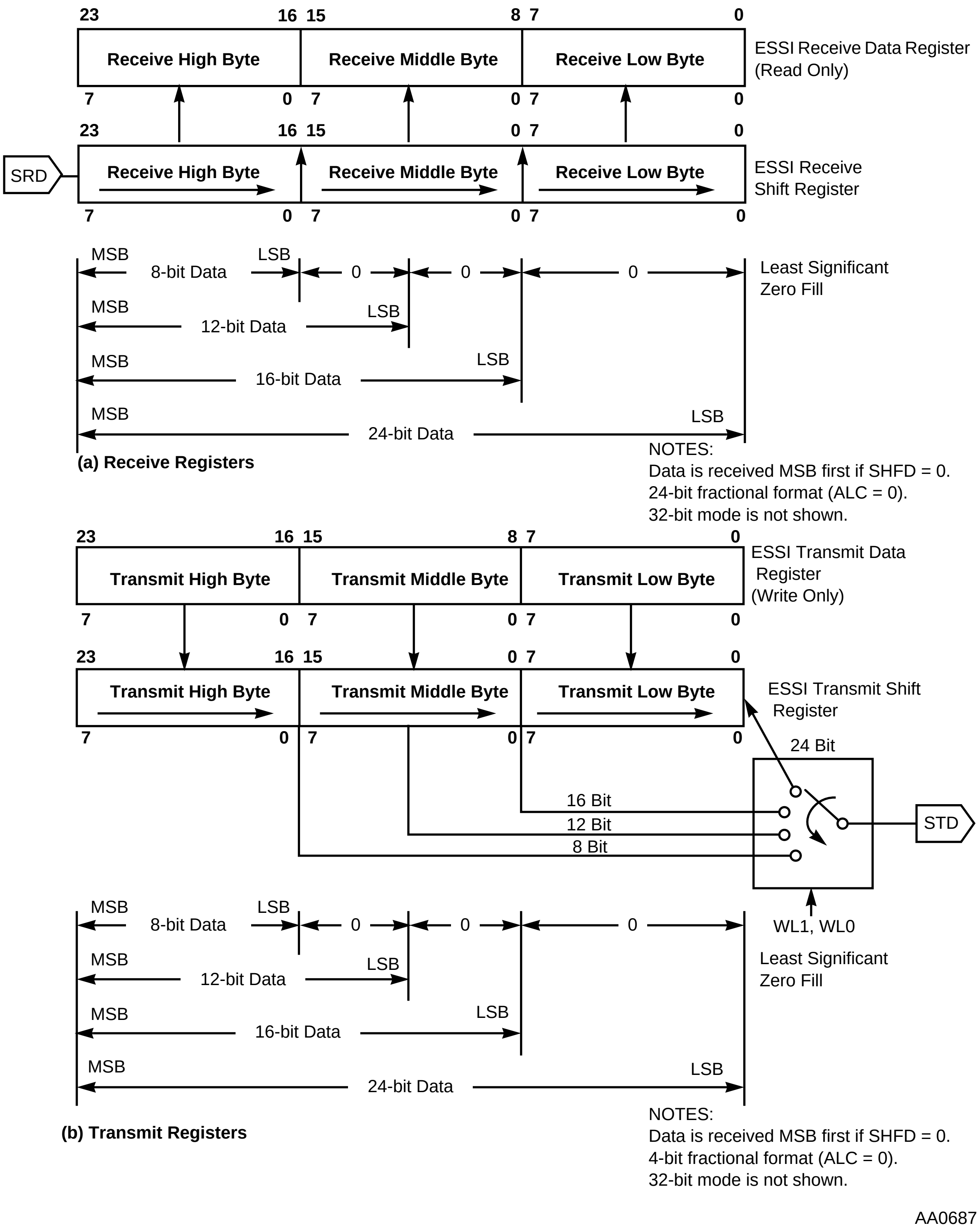
A hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction reset all clear the RDF bit.



NOTES:
Data is received MSB first if SHFD = 0.
24-bit fractional format (ALC = 0).
32-bit mode is not shown.

NOTES:
Data is transmitted MSB first if
SHFD = 0.
4-bit fractional format (ALC = 0).
32-bit mode is not shown.

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



NOTES:
Data is received MSB first if SHFD = 0.
24-bit fractional format (ALC = 0).
32-bit mode is not shown.

NOTES:
Data is received MSB first if SHFD = 0.
4-bit fractional format (ALC = 0).
32-bit mode is not shown.

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

7.4.4 ESSI Receive Shift Register

The 24-bit receive shift register (see **Figure 7-16** and **Figure 7-17**) receives 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 have been 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, according to **Figure 7-16** and **Figure 7-17**. 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 0. 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, as in **Figure 7-16** and **Figure 7-17**. Data is shifted out to the serial transmit data signals by the selected (whether internal or external) bit clock when the associated frame sync I/O is asserted. The word-length control bits in CRA determine the number of bits that must be shifted out before the shift registers are considered empty and can be written again. Depending on the setting of the CRA, the number of bits to be shifted out can be 8, 12, 16, 24, or 32.

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 0. 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)

ESSI0:TX20, TX10, TX00; ESSI1:TX21, TX11, TX01

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** and **Figure 7-17**.) 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. For details, see the *DSP56300 Family Manual*, *Appendix B*, *Polling a Peripheral Device for Write*.

7.4.8 ESSI Time Slot Register (TSR)

TSR is effectively a write-only null data register that prevents data transmission in the current transmit time slot. For timing purposes, 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 filled by 0. In network mode the transmitter(s) use these registers to determine which 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 (as in **Figure 7-16** and **Figure 7-17**) 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. Consequently, 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 the transmitter empty interrupt request is being serviced 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.

The TSM slot mask does not conflict with the TSR. Even if a slot is enabled in the TSM, you can chose to 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 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; that value enables all 32 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, the receiver(s) use these registers to determine which 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 (as in **Figure 7-16** and **Figure 7-17**) 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), but 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.

When the bits in the RSM are set, their setting 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 RSMA or RSMB is 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 filled by 0.

After a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction, the RSM register is reset to $FFFFFFFF; that value enables all 32 time slots for data transmission.

7.5 OPERATING MODES

The ESSI operating modes are selected via the ESSI control registers (CRA and CRB). The operating modes are documented 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, thus configuring all the ESSI signals as GPIO. The ESSI is in the reset state while all ESSI signals are programmed as GPIO; it 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: hardware $\overline{\text{RESET}}$ signal, software RESET instruction, ESSI individual reset, or STOP instruction reset.
2. Program the ESSI control and time slot registers.
3. Write data to all the enabled transmitters.
4. Configure at least one signal as ESSI signal.
5. If an external frame sync is used, from the moment the ESSI is activated, at least five (5) serial clocks are needed before the first external frame sync is supplied. Otherwise, improper operation may result.

When the PC[5:0] bits in the GPIO port control register (PCR) are cleared during program execution, the ESSI stops serial activity and enters 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 there are undefined.

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

Here is an example of how to initialize 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 in use during operation. This step is needed even if DMA services 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 when you first read the SSISR and then read RX.
2. ESSI receive data:
   Occurs when the receive interrupt is enabled, the receive data register is full, and no receive error conditions exist. A read of 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 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 is 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 when you first read the SSISR and then write to all the transmit data registers of the enabled transmitters, or when you write to 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 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 is 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. Write to all the enabled TX registers or to the TSR to clear this interrupt. This error-free interrupt uses 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. Write data to all enabled transmit registers. `TX00`
   c. Enable a peripheral interrupt-generating function. `CRB (TE0)`
   d. Enable a specific peripheral interrupt. `CRB0 (TIE)`
   e. Enable peripheral and associated signals. `PCRC (PC5:0)`
   f. Unmask interrupts at the 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 is complete. Since **step c.** may cause an immediate transmit without generating an interrupt, perform the transmit data preload in **step b.** before **step c.** 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 c.** and **d.** be performed in a single instruction.

5. If an interrupt trigger event occurs at a time before all interrupt trigger configuration steps are performed, the event is ignored forever; in other words, the event is not queued.

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 and operation formats. These modes can be programmed using the ESSI control registers. The data and operation formats available to the ESSI are selected when you set or clear 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

You select either normal mode or network mode by clearing or setting 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, 2 to 32 time slots per frame may be selected. During each frame, 0 to 32 data words may be received or transmitted (from each enabled receiver or transmitter). In either case, the transfers are periodic.

The normal mode typically transfers data to or from a single device. Network mode is typically used in time division multiplexed networks of codecs or DSPs with multiple words per frame.

Network mode has a submode called on-demand mode. You set the MOD bit in the CRB for network mode, and you set the frame rate divider to 0 (DC = $00000) to select on-demand mode.This submode 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. 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. You can enable data transmission that is data-driven 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 to interface to codecs requiring a continuous clock.

7.5.4.2 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of the ESSI interface are 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 are generated internally by the DSP or 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 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 pulse code modulation serial I/O.

7.5.4.3.2 Frame Sync Length for Multiple Devices

The ability to mix frame sync lengths is useful to configure 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; that is, 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; that is, 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 the previous frame immediately. 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. You select either MSB or LSB 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 and 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 the 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 the transmitter 0 drive-enabled signal (Bit SSC1 = 0). The direction of SC1 is determined 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 is latched at the same time as the first bit of the received 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 SSISR IF[1:0] bits respectively and can be read by software.

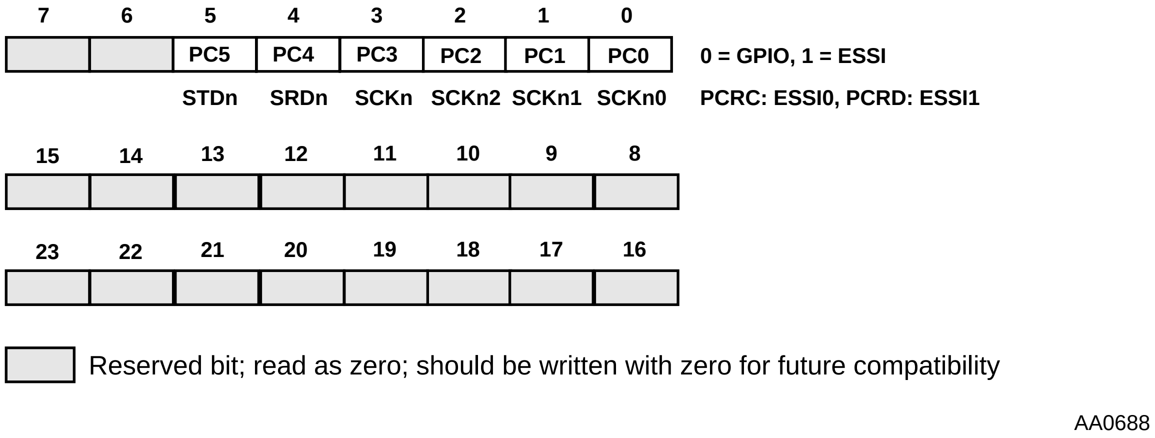
When programmed as output flags, the value of the SC[1:0] bits is taken from the value of the OF[1:0] bits. The value of the OF[1:0] bits is latched when the contents of TX are transferred to the transmit shift register. The value on SC[1:0] is stable from the time the first bit of the transmit data 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 DSP56307 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 (whether C or 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 an 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.



7	6	5	4	3	2	1	0	
▨	▨	PC5	PC4	PC3	PC2	PC1	PC0	0 = GPIO, 1 = ESSI
		STDn	SRDn	SCKn	SCKn2	SCKn1	SCKn0	PCRC: ESSI0, PCRD: ESSI1

15	14	13	12	11	10	9	8
▨	▨	▨	▨	▨	▨	▨	▨

23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨

▨ Reserved bit; read as zero; should be written with zero for future compatibility

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.

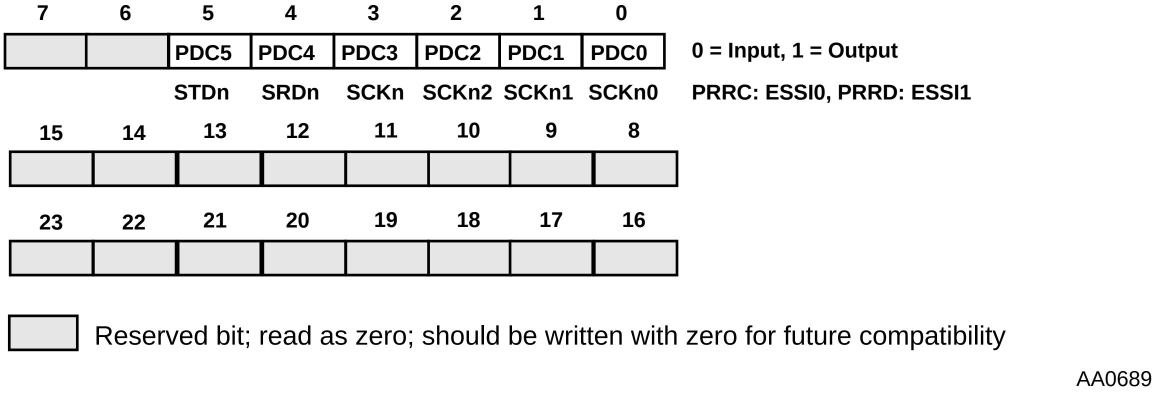

7 6 5 4 3 2 1 0
PDC5 PDC4 PDC3 PDC2 PDC1 PDC0
0 = Input, 1 = Output
STDn SRDn SCKn SCKn2 SCKn1 SCKn0
PRRC: ESSI0, PRRD: ESSI1
15 14 13 12 11 10 9 8
23 22 21 20 19 18 17 16
Reserved bit; read as zero; should be written with zero for future compatibility
AA0689

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.

The following table 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 this signal.

7	6	5	4	3	2	1	0	
▨	▨	PD5	PD4	PD3	PD2	PD1	PD0	
		STDn	SRDn	SCKn	SCKn2	SCKn1	SCKn0	PDRD: ESSI0, PDRD: ESSI1

15	14	13	12	11	10	9	8
▨	▨	▨	▨	▨	▨	▨	▨

23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨

▨ Reserved bit; read as zero; should be written with zero for future compatibility

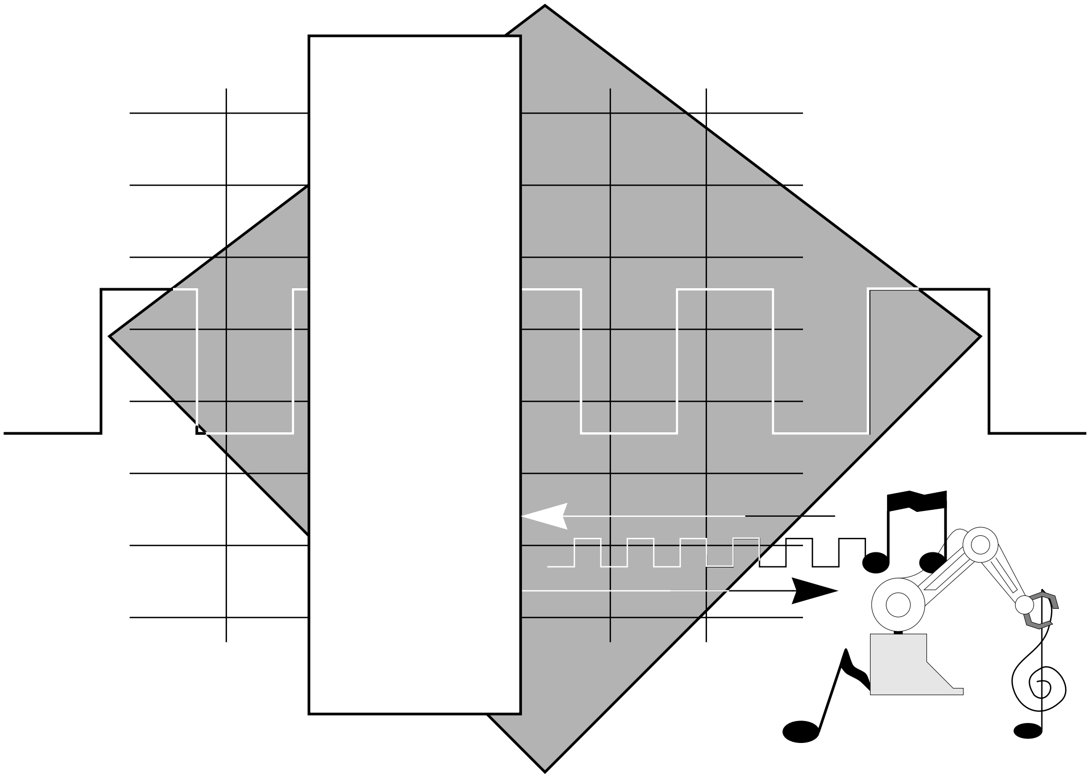
AA0690

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



8.1 INTRODUCTION TO THE SCI

The DSP56307 serial communication interface (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 RS232C, RS422, etc.

This interface uses three dedicated signals: transmit data, receive data, and SCI serial clock. 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 DSP56302 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 are 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**
  - SCI Transmit Data Registers (STX) in **Figure 8-7**
  - SCI Transmit Data Address Register (STXA) in **Figure 8-7**

The SCI includes the GPIO functions documented in **Section 8.5 GPIO Signals and Registers** on page 8-27. The following paragraphs describe each bit in the programming model.

7	6	5	4	3	2	1	0
WOMS	RWU	WAKE	SBK	SSFTD	WDS2	WDS1	WDS0

15	14	13	12	11	10	9	8
SCKP	STIR	TMIE	TIE	RIE	ILIE	TE	RE

23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	REIE

AA0854

Figure 8-1 SCI Control Register (SCR)

7	6	5	4	3	2	1	0
R8	FE	PE	OR	IDLE	RDRF	TDRE	TRNE

15	14	13	12	11	10	9	8
▨	▨	▨	▨	▨	▨	▨	▨

23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨

AA0855

Figure 8-2 SCI Status Register (SSR)

7	6	5	4	3	2	1	0
CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0

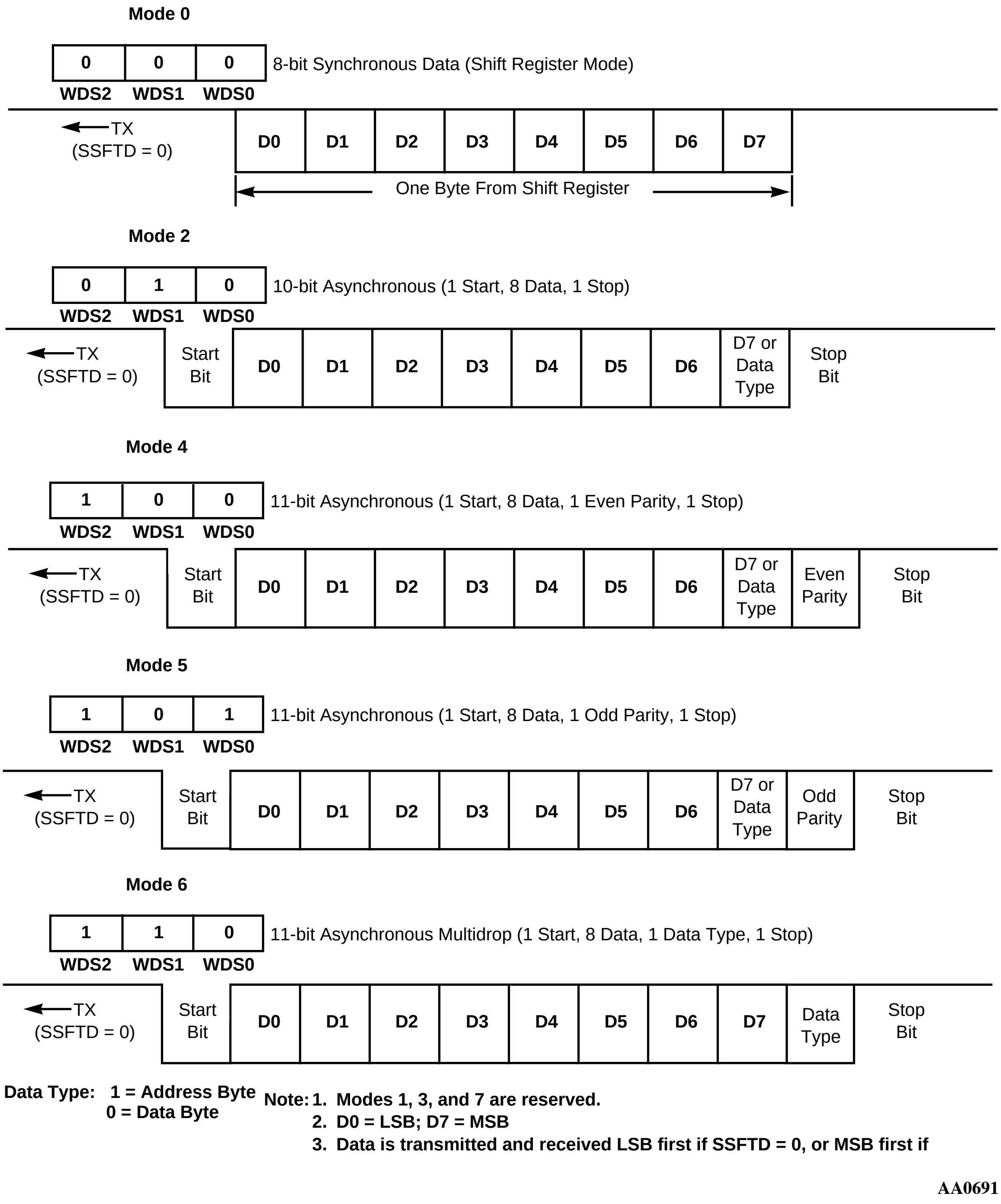
15	14	13	12	11	10	9	8
TCM	RCM	SCP	COD	CD11	CD10	CD9	CD8

23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨

▨ Reserved bit; read as 0; should be written with 0 for future compatibility

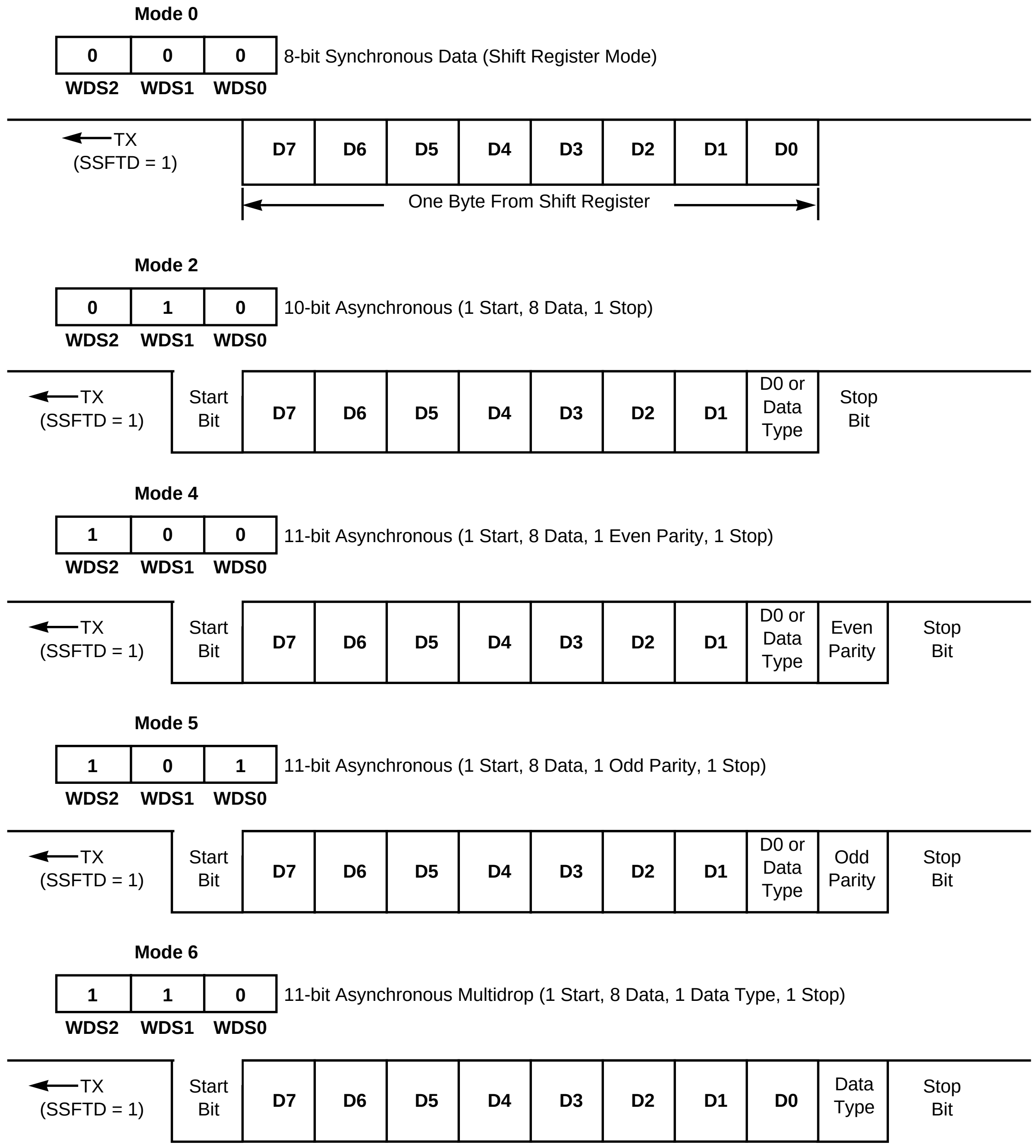
AA0856

Figure 8-3 SCI Clock Control Register (SCCR)



Data Type: 1 = Address Byte
0 = Data Byte

Note:
1. **Modes 1, 3, and 7 are reserved.**
2. **D0 = LSB; D7 = MSB**
3. **Data is transmitted and received LSB first if SSFTD = 0, or MSB first if**



Data Type: 1 = Address Byte
0 = Data Byte

Note:
1. **Modes 1, 3, and 7 are reserved.**
2. **D0 = LSB; D7 = MSB**
3. **Data is transmitted and received LSB first if SSFTD = 0, or MSB first if**

AA0691 (cont.)

Figure 8-4 SCI Data Word Formats

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 24 bits are currently defined. Each bit is documented 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** and shown in **Figure 8-4**.

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. You can synchronize data by using a gated transmit and receive clock compatible with the Intel 8051 serial interface mode 0.

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.

Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears the word select bits.

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, and remain adjacent to the stop bit. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears SSFTD.

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 10 or 11 0s (depending on WDS mode), and reverts to idle or sending data. If SBK remains set, the transmitter continually sends whole frames of 0s (10 or 11 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; the frame error 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 re-enabled by an idle string of at least 10 or 11 (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 there 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 re-enabled 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 asynchronous mode, the wakeup function is enabled; i. e., the SCI is asleep and can be awakened by the event defined by the WAKE bit. In

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 (i. e., WAKE is cleared) or wakeup on address bit (i. e., WAKE is set) must be chosen.

- 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.
- When WAKE is set and RWU is set, the receiver does not respond to data on the data line until a data frame with Bit 9 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 whether the message is intended for this DSP.

- If the message is for this DSP, the message is received, and RWU is set to wait for the next message.
- 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, and 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 ensure that the line goes idle before starting a new message. To force this separation of messages by the minimum idle line time, we recommend the following sequence:

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 to queue 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 remains 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:

- The IDLE bit shows the real status of the receive line at all times.
- 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 whether 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 whether 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 data being written 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). Either a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction sets TRNE.

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. Either a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction sets TDRE.

In synchronous mode, when the internal SCI clock is in use, 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 delay of 2 to 4 serial clock cycles 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 2 cycles (of the 16 × clock) after the start bit; that is, 2 (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. Also a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction clears RDRF.

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). A hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction clears IDLE.

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, overrun error has higher priority than FE or PE. A hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction clears OR.

8.3.2.6 SSR Parity Error (PE) Bit 5

In 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. A hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction also clears PE. 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 mode 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 the SRX being read. A hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction 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 reads of the SRX or SCI status register. A hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, an SCI individual reset, or a STOP instruction clears R8.

8.3.3 SCI Clock Control Register (SCCR)

The SCCR is a 24-bit read/write register that controls the selection of clock modes and baud rates for the transmit and receive sections of the SCI interface. The control bits are documented 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** and **Figure 8-6**, follow:

- The SCI logic always uses a 16 × internal clock in asynchronous mode 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. These maximum rates are the same for internally or externally supplied clocks.
- 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 if the user wishes 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.

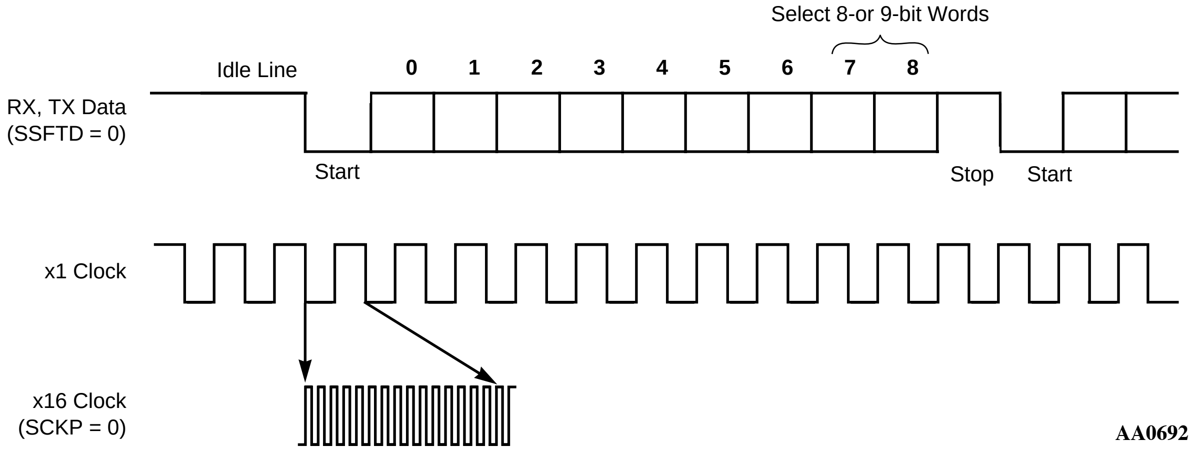


Figure 8-5 16 x Serial Clock

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. 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 the SCI mode. If the SCI mode is synchronous, the output divider is fixed at divide by 2.

If the SCI mode is asynchronous, either:

- If COD is cleared and SCLK is an output (i.e., TCM and RCM are both cleared), then 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.

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. 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. 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

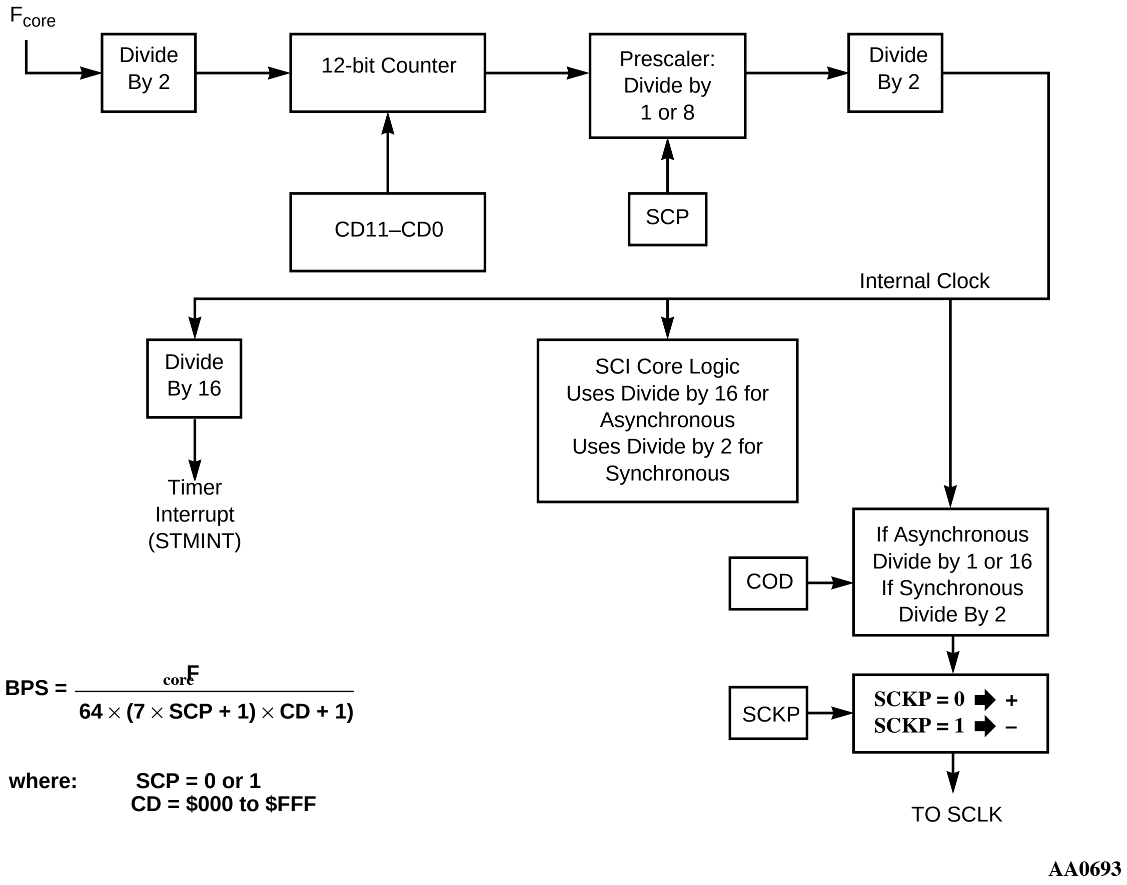

F core
Divide By 2
12-bit Counter
Prescaler: Divide by 1 or 8
Divide By 2
CD11–CD0
SCP
Internal Clock
Divide By 16
Timer Interrupt (STMINT)
SCI Core Logic Uses Divide by 16 for Asynchronous Uses Divide by 2 for Synchronous
If Asynchronous Divide by 1 or 16 If Synchronous Divide By 2
COD
SCKP
SCKP = 0 ➡ +
SCKP = 1 ➡ –
TO SCLK
BPS = F core / 64 × (7 × SCP + 1) × CD + 1)
where: SCP = 0 or 1
CD = $000 to $FFF
AA0693

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. 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.

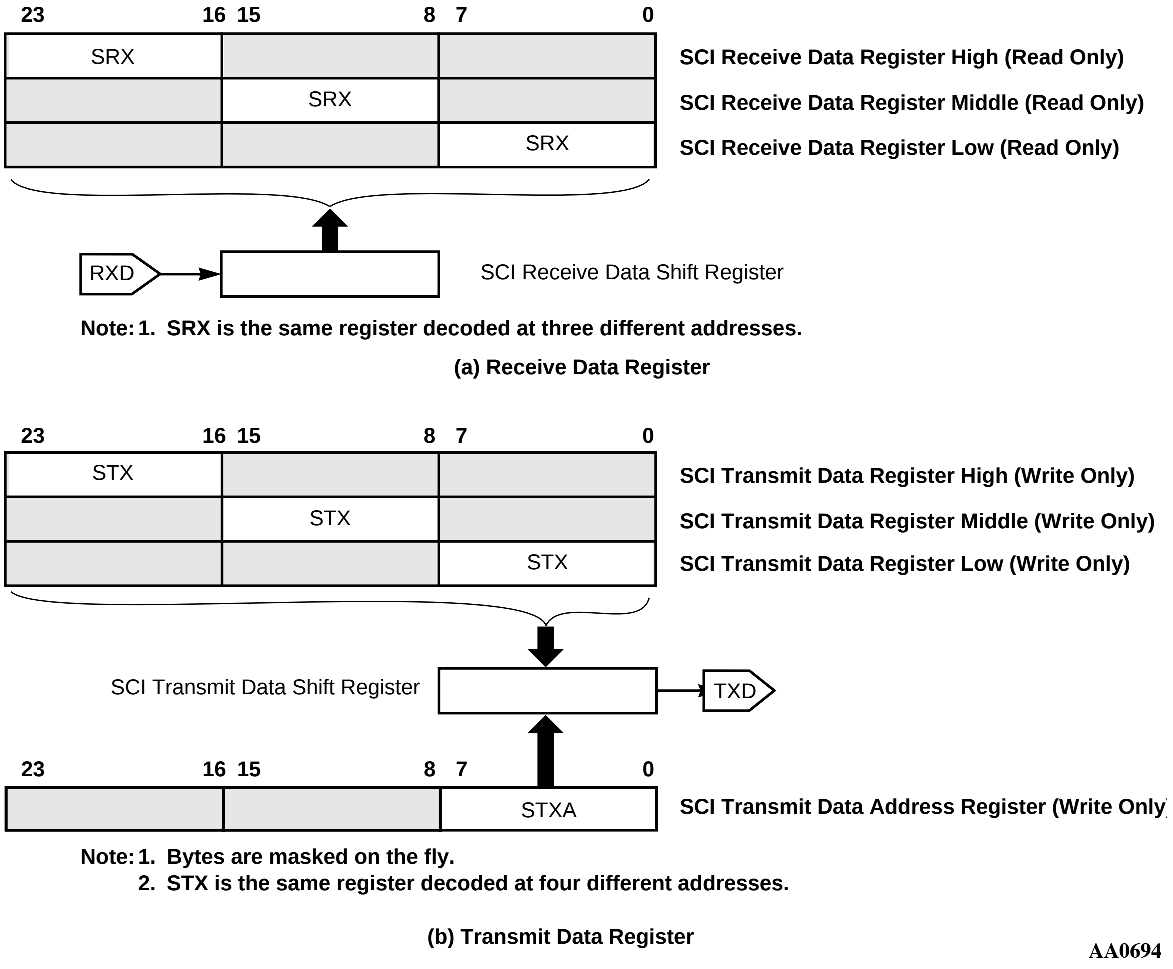

23
16 15
8 7
0
SRX
SCI Receive Data Register High (Read Only)
SRX
SCI Receive Data Register Middle (Read Only)
SRX
SCI Receive Data Register Low (Read Only)
RXD
SCI Receive Data Shift Register
Note: 1. SRX is the same register decoded at three different addresses.
(a) Receive Data Register
23
16 15
8 7
0
STX
SCI Transmit Data Register High (Write Only)
STX
SCI Transmit Data Register Middle (Write Only)
STX
SCI Transmit Data Register Low (Write Only)
SCI Transmit Data Shift Register
TXD
23
16 15
8 7
0
STXA
SCI Transmit Data Address Register (Write Only)
Note: 1. Bytes are masked on the fly.
2. STX is the same register decoded at four different addresses.
(b) Transmit Data Register
AA0694

Figure 8-7 SCI Programming Model - Data Registers

8.3.4.1 SCI Receive Register (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 are read as 0s. This way of mapping SRX efficiently packs three bytes into one 24-bit word by OR-ing 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 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, a gated clock provides synchronization.

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 Register (STX)

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 11-bit asynchronous multidrop mode when the data is an address and the programmer wants to set the ninth bit (the address bit). 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 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, after the last bit from the previous word is shifted out; that is, the transmit shift register is empty. Like the receiver, the transmitter is double-buffered. However, a delay of two to four serial clock cycles 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 there is no dedicated flag for this register. Because of this fact and the two- to four-cycle delay, two bytes cannot be written consecutively to STX or STXA without polling, as the second byte might overwrite the first byte. Consequently, you should always poll the TDRE flag 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; that clock 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 mode, 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, SCI synchronous mode is equivalent to the SSI operation in 8-bit data on-demand mode.

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. For details see the *DSP56300 Family Manual*, *Appendix B*, *Polling a Peripheral Device for Write*.

8.4 OPERATING MODES

The operating modes for the DSP56307 SCI are the following:

- 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 DSP56307 to share a single serial line efficiently with other peripherals.

These modes are selected by the WD[0:2] bits in the SCR.

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 synchronizes data.

Asynchronous modes are compatible with most UART-type serial devices. Standard RS232C communication links are supported by these modes.

Multidrop asynchronous mode is compatible with the MC68681 DUART, the M68HC11 SCI interface, and the Intel 8051 serial interface.

8.4.1 SCI after Reset

There are several different ways to reset the SCI:

- Hardware $\overline{\text{RESET}}$ signal
- 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.
- Individual reset:
  During program execution, the PC2, PC1, and PC0 bits can be cleared (i.e., individually reset), causing 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, to allow 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.
- Stop processing state reset (i.e., the STOP instruction)
  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 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	—	—	—	–

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. Use a hardware $\overline{\text{RESET}}$ signal or software RESET instruction.
2. Program SCI control registers.
3. Configure at least one SCI signal as not 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. Consequently, the crystal frequency must 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. Ensure that the SCI is in its individual reset state (PCR = $0).
2. Configure the control registers (SCR, SCCR) according to the operating mode, but do not enable transmitter (TE = 0) or receiver (RE = 0).

   It is now possible to set the interrupts enable bits that used during the operation. No interrupt occurs yet.
3. Enable the SCI by setting the PCR bits according to which signals are used 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 use.

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 (i.e., a 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 the initialization sequence has been transmitted (depending on the operating mode).

8.4.4 Preamble, Break, and Data Transmission Priority

Two or three transmission commands can be set simultaneously:

- A preamble (TE is set.)
- A break (SBK is set or is cleared.)
- An indication that 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). To clear the pending interrupt, read the SCI status register; then read 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. Read SRX to clear 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. Write STX to clear 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 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 an SCI signal. When a PC[i] bit is cleared, the corresponding port signal is configured as a GPIO signal.

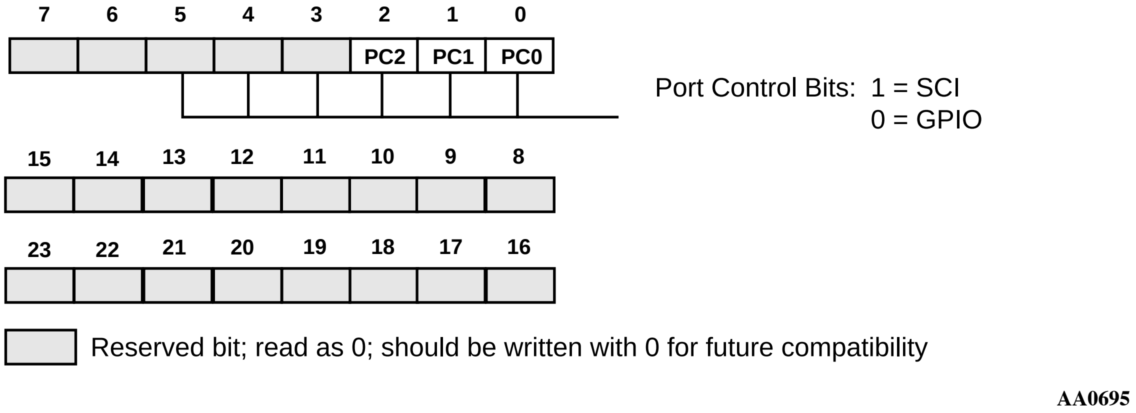

7
6
5
4
3
2
1
0
PC2
PC1
PC0
Port Control Bits: 1 = SCI
0 = GPIO
15
14
13
12
11
10
9
8
23
22
21
20
19
18
17
16
Reserved bit; read as 0; should be written with 0 for future compatibility
AA0695

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.

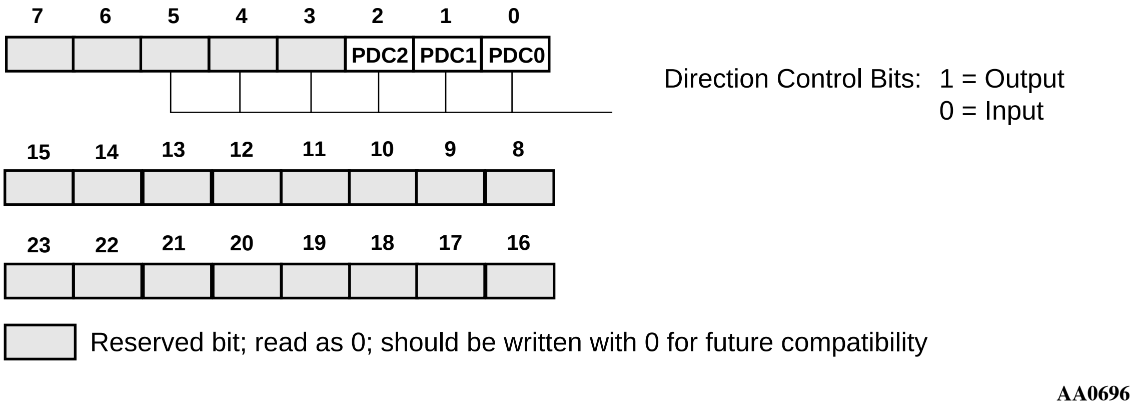

7
6
5
4
3
2
1
0
PDC2
PDC1
PDC0
Direction Control Bits: 1 = Output
0 = Input
15
14
13
12
11
10
9
8
23
22
21
20
19
18
17
16
Reserved bit; read as 0; should be written with 0 for future compatibility
AA0696

Figure 8-9 Port E Direction Register (PRRE)

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

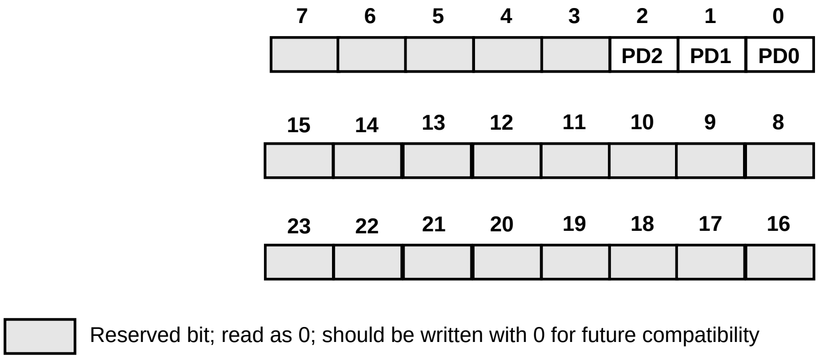
The following table shows the port signal configurations.

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 reads or writes data to or from SCI GPIO signals. Bits PD[2:0] read or write data to or from 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.


7
6
5
4
3
2
1
0
PD2
PD1
PD0
15
14
13
12
11
10
9
8
23
22
21
20
19
18
17
16
Reserved bit; read as 0; should be written with 0 for future compatibility

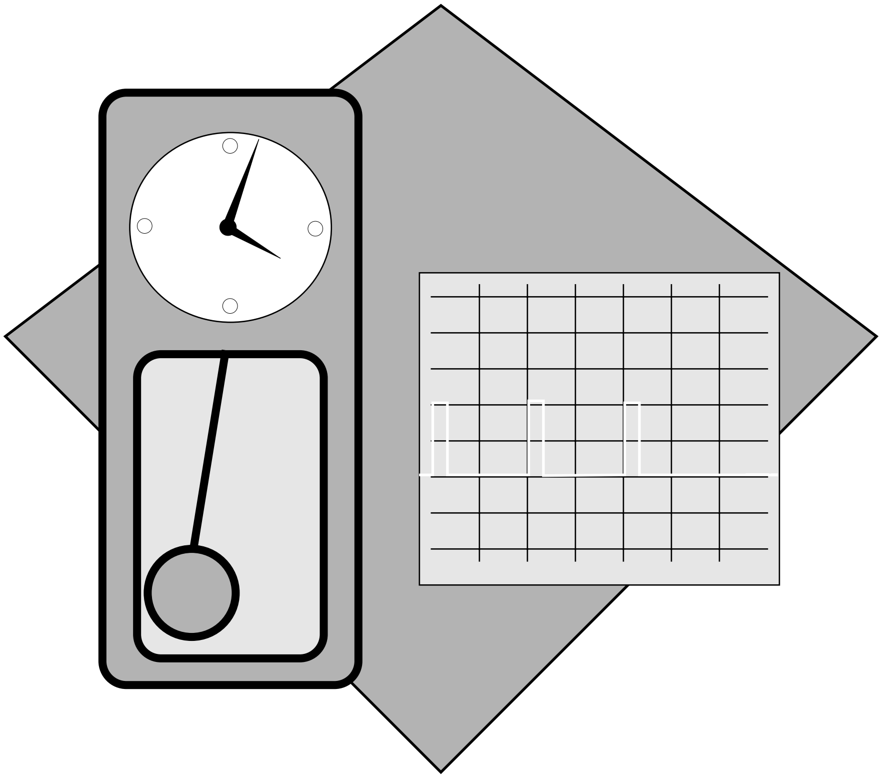
AA0697

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

The timers in the DSP56307 internal triple timer module act as timed pulse generators or as pulse-width modulators. Each timer has a single signal that can function as a GPIO signal or as a timer signal. Each timer can also function 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 contains a common 21-bit prescaler and three independent and identical general-purpose 24-bit timer/event counters, each with its own register set. Each timer can use internal or external clocking and can interrupt the DSP56307 after a specified number of events (clocks) or signal an external device after counting internal events. Each timer can also trigger DMA transfers after a specified number of events (clocks) occurs. Each timer connects to the external world through one bidirectional signal, designated TIO0–TIO2 for timers 0–2.

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. Note the 24-bit timer prescaler load register (TPLR) and the 24-bit timer prescaler count register (TPCR). Each of the three timers can use the prescaler clock as its clock source.

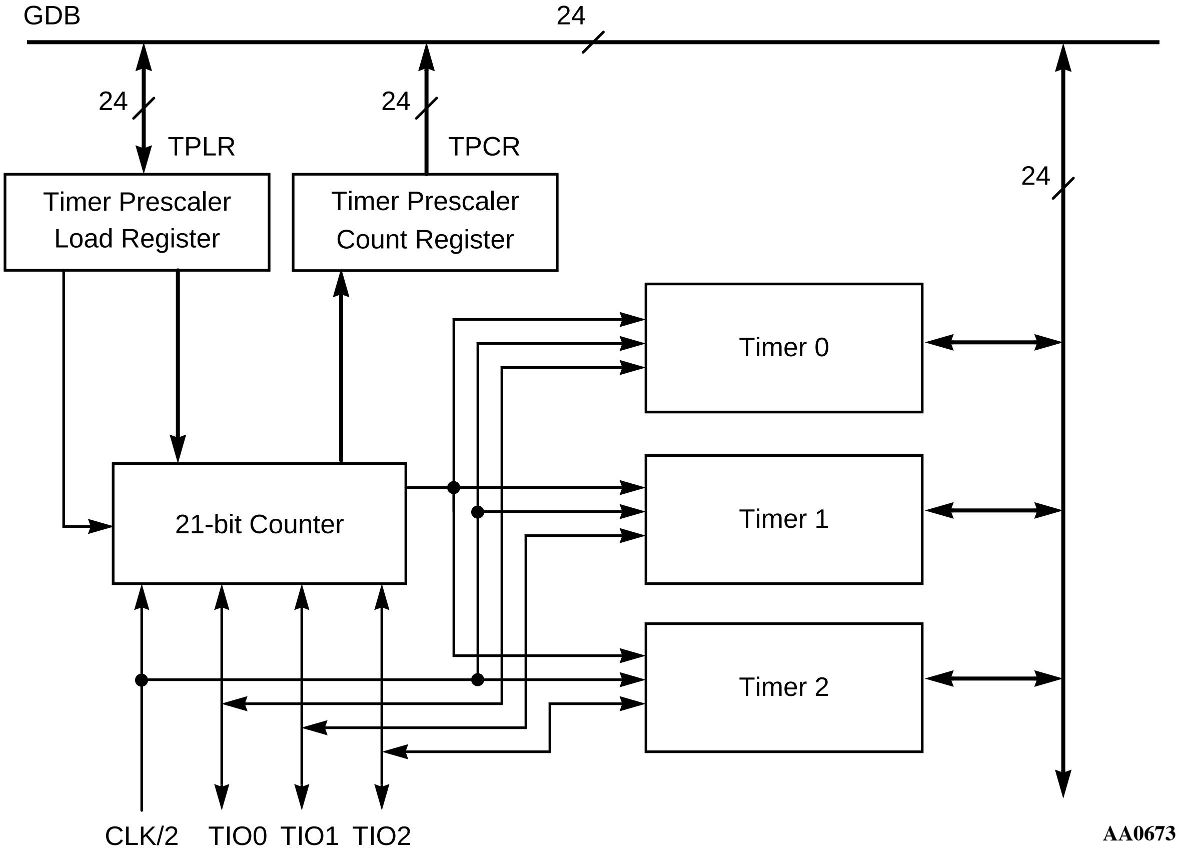


Figure 9-1 Triple Timer Module Block Diagram

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** 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 9.4**. For a description of their operation, see **Section 9.4.1**.

The DSP56307 treats 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 on page 9-6.

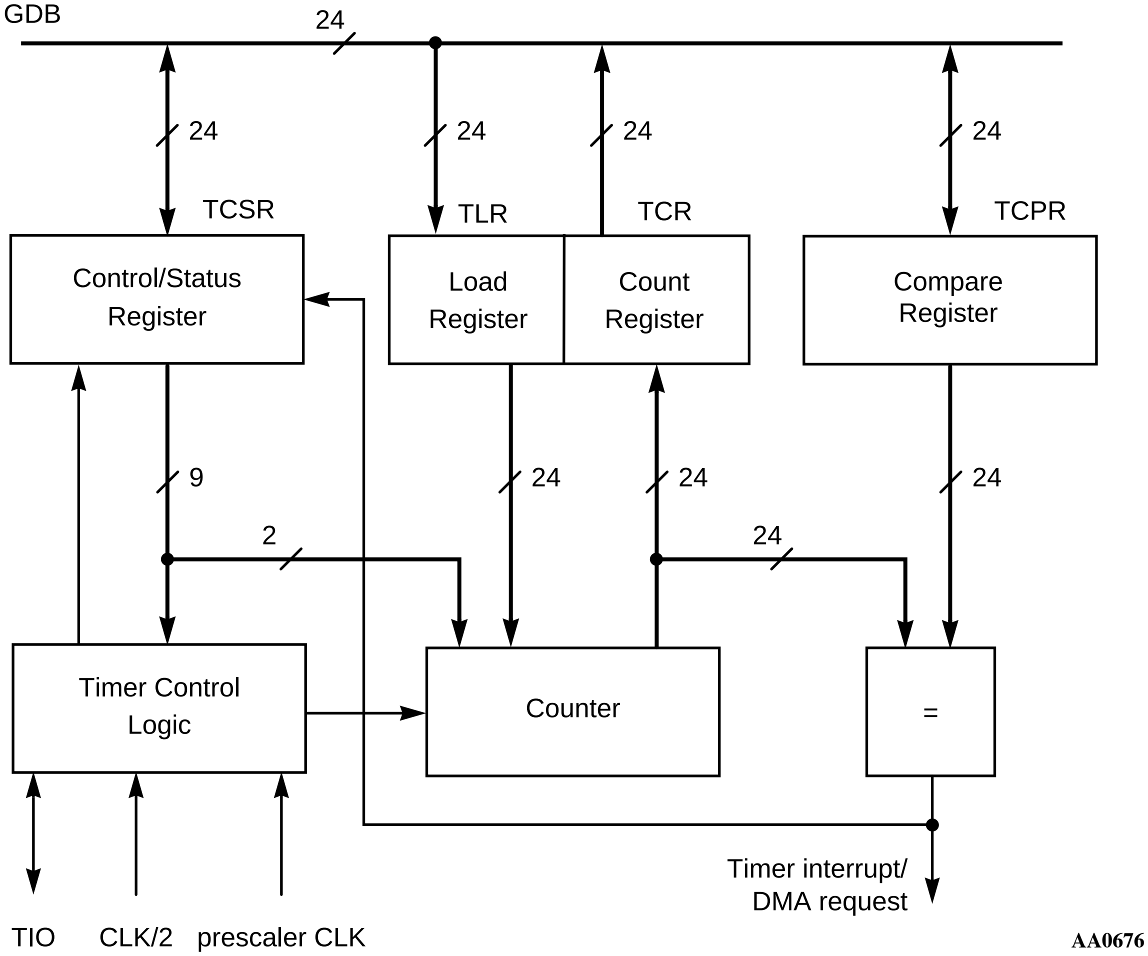
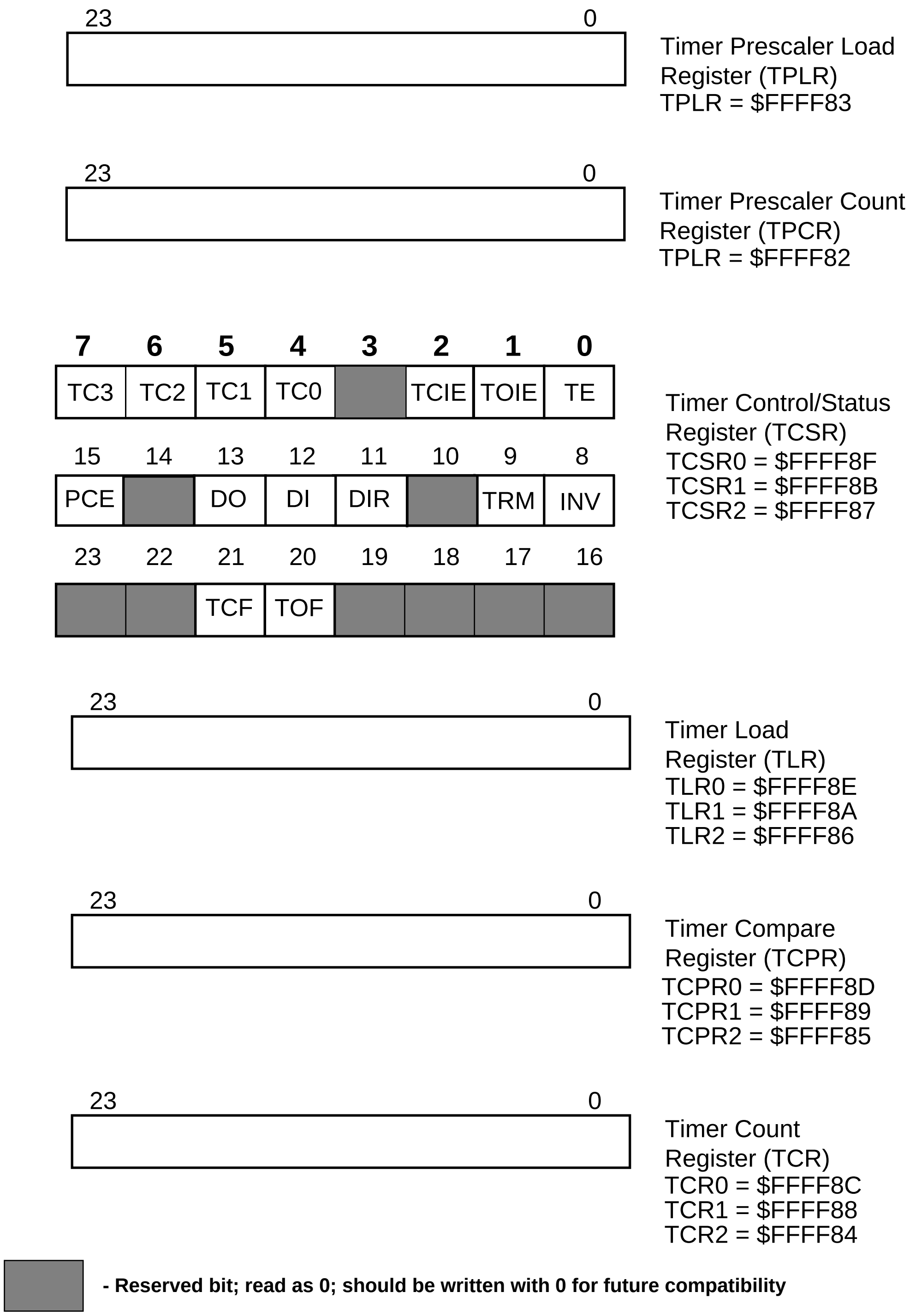


Figure 9-2 Timer Module Block Diagram

9.3 TRIPLE TIMER MODULE PROGRAMMING MODEL

The programming model for the triple timer module is shown in **Figure 9-3**.

Triple Timer Module Programming Model



Timer Prescaler Load
Register (TPLR)
TPLR = $FFFF83

Timer Prescaler Count
Register (TPCR)
TPLR = $FFFF82

7	6	5	4	3	2	1	0
TC3	TC2	TC1	TC0	▨	TCIE	TOIE	TE
15	14	13	12	11	10	9	8
PCE	▨	DO	DI	DIR	▨	TRM	INV
23	22	21	20	19	18	17	16
▨	▨	TCF	TOF	▨	▨	▨	▨

Timer Control/Status
Register (TCSR)
TCSR0 = $FFFF8F
TCSR1 = $FFFF8B
TCSR2 = $FFFF87

Timer Load
Register (TLR)
TLR0 = $FFFF8E
TLR1 = $FFFF8A
TLR2 = $FFFF86

Timer Compare
Register (TCPR)
TCPR0 = $FFFF8D
TCPR1 = $FFFF89
TCPR2 = $FFFF85

Timer Count
Register (TCR)
TCR0 = $FFFF8C
TCR1 = $FFFF88
TCR2 = $FFFF84

▨ **- Reserved bit; read as 0; should be written with 0 for future compatibility**

Figure 9-3 Timer Module Programmer's Model

9.3.1 Prescaler Counter

The prescaler counter is a 21-bit counter that decrements 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 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 timer prescaler load register (TPLR) is a 24-bit read/write register that controls the prescaler divide factor (i. e., the number that the prescaler counter will load and begin counting from) and the source for the prescaler input clock. The control bits are documented in the following paragraphs and 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 bit; 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, which 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 of the corresponding TIO signal.

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 DSP56307 internal operating frequency divided by 4 (i.e., 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 documented in the following paragraphs and 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 bit; 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 control and status bits are documented in the following paragraphs and in **Table 9-2**.

9.3.4.1 Timer Enable (TE) Bit 0

The TE bit enables or disables the timer. When set, 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.

When clear, 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 the 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 you switch from tri-state into active state, these signals should be tied to the high or low signal state by pull-up or pull-down resistors.

9.3.4.2 Timer Overflow Interrupt Enable (TOIE) Bit 1

The TOIE bit enables timer overflow interrupts. When set, 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.

When cleared, 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 enables or disables the timer compare interrupts. When set, 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.

When cleared, 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 the Timer mode of operation. **Table 9-2** summarizes the TC bit functionality. A detailed description of the timer operating modes appears in **Section 9.4** on page 9-16.

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 (i.e., CLK/4).

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
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	Output	Internal
1	0	0	0	8	Reserved	—	—
1	0	0	1	9	Watchdog pulse	Output	Internal

Table 9-2 Timer Control Bits (Continued)

Bit Settings				Mode Characteristics			
TC3	**TC2**	**TC1**	**TC0**	**Mode Number**	**Mode Function**	**TIO**	**Clock**
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: 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 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.

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.

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**
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.	—	—
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.

Note: The INV bit affects both the timer and GPIO modes of operation. To ensure correct operation, change this bit only when one or both of the following conditions is true: the timer is disabled (the TE bit in the TCSR is cleared). The timer is in GPIO mode.

The INV bit does not affect the polarity of the prescaler source when the TIO is 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 functions 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 functions as a GPIO signal only when 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 the 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 selects 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 the timer source clock for the counter, regardless of the timer operating mode. To insure proper operation, the PCE bit is changed only when the timer is disabled (i.e., when the TE bit is cleared). The PS[1:0] bits of the TPLR determine which source clock is used for the prescaler. 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 a counter overflow has occurred. This bit is cleared by a 1 written to the TOF bit. A 0 written to the TOF bit itself 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 timer, PWM, and watchdog modes, the TCF bit is set after (N – M + 1) events are 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 has been completed.

Writing a 1 to the TCF bit clears it. A 0 written to 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 a 1 written to the specific bit. To insure that only the target bit is cleared, do not use the BSET command. The proper way to clear these bits is to write 1, using a MOVEP instruction, to the flag to be cleared and 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 TRM bit in the TCSR is set, the counter is reloaded each time after reaches the value contained by the timer compare register 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 overflows 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 reaches the value contained by the timer compare register 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 bit is set and an interrupt is generated if interrupts are enabled (i.e., if the timer compare interrupt enable bit in the TCSR is set). The TCPR is ignored in measurement modes.

9.3.7 Timer Count Register (TCR)

The TCR is a 24-bit read-only register. In timer and watchdog modes, the contents of the counter can be read at any time from 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 mode, the TIO signal is used for the input signal.

9.4 TIMER MODES OF OPERATION

Each timer has operational modes that meet a variety of system requirements, as follows:

- 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

To select timer modes, select the TC[3:0] bits in the TCSR. **Table 9-2** on page 9-10 shows you how to select different timer modes by setting the bits in the TCSR. The table also shows the TIO signal direction and the clock source for each timer mode. The following paragraphs document these modes in detail.

Note: To insure proper operation, the TC[3:0] bits are changed only when the timer is disabled (i.e., when the TE bit in the TCSR is cleared).

9.4.1 Timer Modes

The following timer 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 DSP56307 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 repeats 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. You can read the counter contents at any time from 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 DSP56307 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 repeats until the TE bit is cleared (disabling the timer). You can read the counter contents at any time from the TCR.

The value of the TLR sets the delay between starting the timer and generating the output pulse. To generate successive output pulses with a delay of X clocks between signals, set the TLR value to X/2 and set the TRM bit. This process repeats 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. You can read the counter contents at any time from 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 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 DSP56307 clock divided by two (i.e., 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 resumes. If the TRM bit is cleared, the counter continues to increment on each timer clock.

This process is repeats until the TE bit is cleared, disabling the timer. You can read the counter contents at any time from 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, set the TLR value to X/2, and set the TRM bit. This process repeats 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.

You can read the counter contents at any time from 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 (if interrupt enable bits are set) 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 DSP56307 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 increment on each timer clock. This process repeats 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.

You can read the counter contents at any time from 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 that increments 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 count value of the timer 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 DSP56307 clock divided by two (i.e., 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 (i.e., 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 repeats 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. You can read the counter contents at any time from 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 DSP56307 clock divided by two (i.e., 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. This process repeats 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. You can read the counter contents at any time from 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 DSP56307 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. 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.

You can read the counter contents at any time from 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 DSP56307 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 increment on each timer clock.

If counter overflow occurs, 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 resumes. If the TRM bit is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is disabled by the TE bit being cleared.

TIO signal polarity is determined by the value of the INV bit. When the counter is started by the TE bit being set, the TIO signal assumes the value of the INV bit. On each subsequent toggle 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.

You can read the counter contents at any time from 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 increments 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 percent 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 DSP56307 internal clock divided by two (i.e., 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 resumes. If the TRM bit is cleared, the counter continues to be incremented on each subsequent timer clock. This process repeats 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).

You can read the counter contents at any time from 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 the DSP56307 hardware $\overline{\text{RESET}}$ signal is asserted. This convention insures that a valid RESET signal is generated when the TIO signal is used to reset the DSP56307.

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 a 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 DSP56307 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 resumes. If the TRM bit is cleared, the counter continues to increment on each subsequent timer clock.

When a counter overflow occurs, 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 when the TE bit is cleared. The counter contents can be read at any time from the TCR.

Note: **I**n this mode, internal logic preserves the TIO value and direction for an additional 2.5 internal clock cycles after the DSP56307 hardware $\overline{\text{RESET}}$ signal is asserted. This convention insures that a valid reset signal is generated when the TIO signal resets the DSP56307.

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 DSP56307 leaves the wait state and services the interrupt.

9.4.6.2 Timer Behavior during Stop

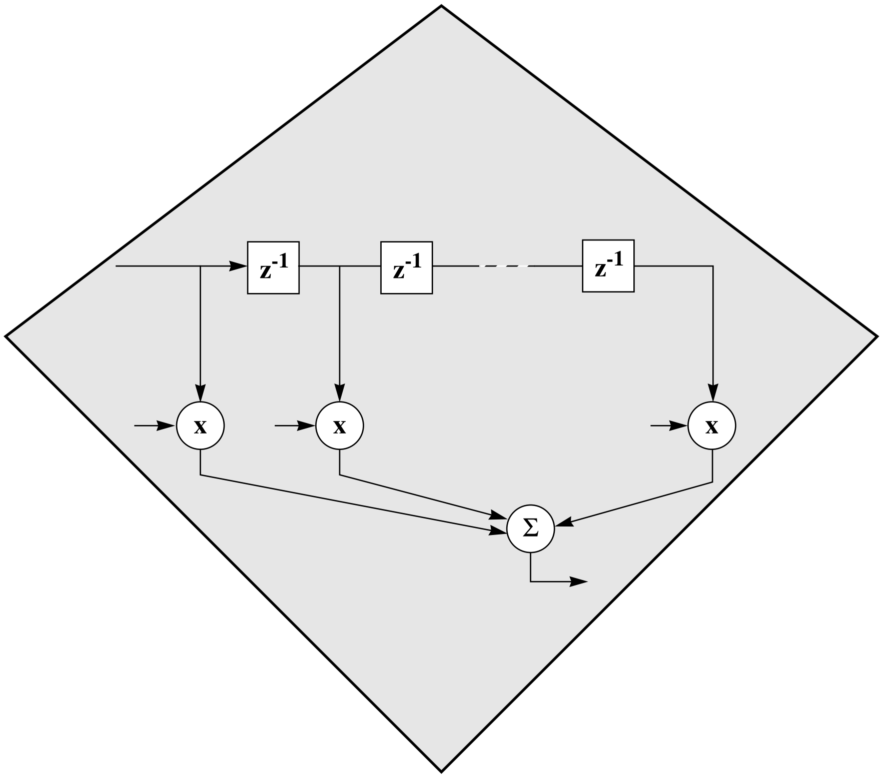
During the execution of the STOP instruction, the timer clocks are disabled, timer activity is stopped, and the TIO signals are disconnected. Any external changes that happen to the TIO signals are ignored when the DSP56307 is in stop state. To insure correct operation, disable the timers before the DSP56307 is placed in stop state.

9.4.7 DMA Trigger

Each timer can also trigger DMA transfers if a DMA channel is 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 save multiple DMA triggers generated by the timer. To ensure that all DMA triggers are serviced, provide for the preceding DMA trigger to be serviced before the DMA channel receives the next trigger.

SECTION 10

ENHANCED FILTER COPROCESSOR



10.1 INTRODUCTION TO EFCOP

The EFCOP is a peripheral module that functions as a general-purpose, fully programmable filter. It has optimized modes of operation to perform real and complex finite impulse response (FIR) filtering, infinite impulse response (IIR) filtering, adaptive FIR filtering, and multichannel FIR filtering.

The EFCOP allows filter operations to be completed concurrently with DSP56300 core operations with minimal CPU intervention. For optimal performance, the EFCOP has one dedicated Filter Multiplier Accumulator (FMAC) unit. Thus, for filtering, the combination Core/EFCOP offers dual MAC capabilities.

Its dedicated modes make the EFCOP a very flexible filtering co-processor with operations optimized for cellular base station applications. The EFCOP architecture also allows adaptive FIR filtering in which the filter coefficient update is performed using any fixed-point standard or non-standard adaptive algorithms—for example, the well-known Least Mean Square (LMS) algorithm, the Normalized LMS and customized update algorithms. In a transceiver base station, the EFCOP can perform complex matched filtering to maximize the signal-to-noise ratio (SNR) within an equalizer. In a transcoder base station or a mobile switching center, the EFCOP can perform all types of FIR and IIR filtering within a vocoder, as well as LMS-type echo cancellation.

10.2 KEY FEATURES

- Fully programmable real/complex filter machine with 24-bit resolution
- FIR filter options
  - Four modes of operation with optimized performance:
    - Mode 0—FIR machine with real taps
    - Mode 1—FIR machine with complex taps
    - Mode 2—Complex FIR machine generating pure real/imaginary outputs alternately
    - Mode 3—Magnitude (calculate the square of each input sample)
  - 4-bit decimation factor in FIR filters providing up to 1:16 decimation ratio
  - Easy to use adaptive mode supporting true or delayed LMS-type algorithms
  - K-constant input register for coefficient updates (in adaptive mode)
- IIR filter options:
  - Direct form 1 (DFI) and direct form 2 (DFII) configurations[1]
  - Three optional output scaling factors (1, 8, or 16)

[1] For details on DFI and DFII modes, refer to the Motorola application note entitled *Implementing IIR/FIR Filters with Motorola's DSP56000/DSP56001* (APR7/D).

- Multichannel mode to process multiple, equal-length filter channels (up to 64) simultaneously with minimal core intervention
- Optional input scaling for both FIR and IIR filters
- Two filter initialization modes
  - No initialization
  - Data initialization
- Sixteen-bit arithmetic mode support
- Three rounding options available:
  - No rounding
  - Convergent rounding
  - Two's complement rounding
- Arithmetic saturation mode support for bit-exact applications
- Sticky saturation status bit indication
- Sticky data/coefficient transfer contention status bit
- 4-word deep input data buffer for maximum performance
- EFCOP-shared and core-shared 4K-word filter data memory bank and 4K-word filter coefficient memory bank
- Two memory bank base address pointers, one for data memory (shared with X memory) and one for coefficient memory (shared with Y memory)
- I/O data transfers via core or DMA with minimal core intervention
- Core-concurrent operation with minimal core intervention

10.3 GENERAL DESCRIPTION

As shown in **Figure 10-1**, the EFCOP comprises these main functional blocks:

- Peripheral module bus (PMB) interface, including:
  - Data input buffer
  - Constant input buffer
  - Output buffer
  - Filter counter
- Filter data memory (FDM) bank
- Filter coefficient memory (FCM) bank
- Filter multiplier accumulator (FMAC) machine
- Address generator
- Control logic

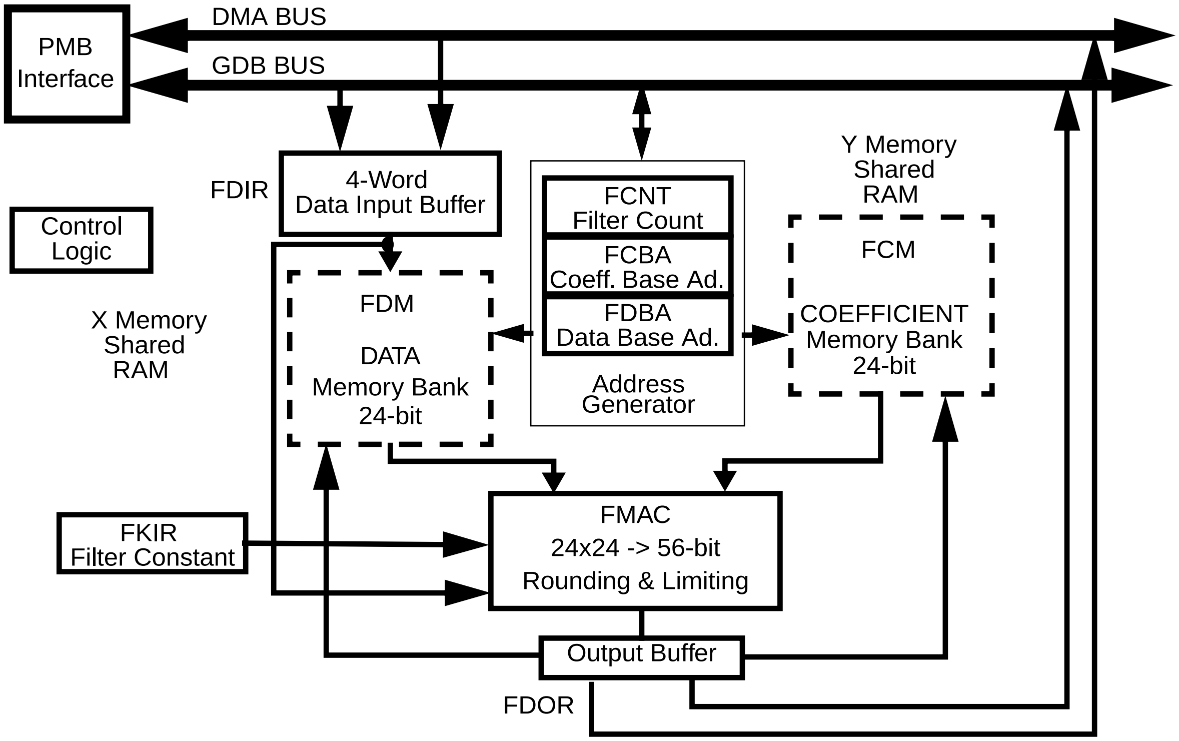


Figure 10-1 EFCOP Block Diagram

10.3.1 PMB Interface

The PMB interface block provides control and status registers, buffering of the internal bus from the PMB, address decoding and generation, and control of the handshake signals required for DMA and interrupt operations. Various control and status registers are provided by the interface logic. The block generates interrupt and DMA trigger signals when data transfer is required. The interface registers accessible to the DSP56300 core through the PMB are shown in **Figure 10-2** and summarized in **Table 10-1**.

23 0

EFCOP data input register (FDIR)
EFCOP data output register (FDOR)
EFCOP K-constant input register (FKIR)
EFCOP filter count register (FCNT)
EFCOP control status register (FCSR)
EFCOP ALU control register (FACR)
EFCOP data buffer base address (FDBA)
EFCOP coeff. buffer base address (FCBA)
EFCOP decimation/channel register(FDCH)

Figure 10-2 EFCOP Register Layout

Table 10-1 EFCOP Register Descriptions

Register Name	Description
Filter data input register (FDIR)	The FDIR is a 4-word-deep 24-bit-wide FIFO used for DSP-to-EFCOP data transfers. Data from the FDIR is transferred to the FDM for filter processing.
Filter data output register (FDOR)	The FDOR is a 24-bit-wide register used for EFCOP-to-DSP data transfers. Data is transferred to FDOR after processing of all filter taps is completed for a specific set of input samples.
Filter K-constant input register (FKIR)	The FKIR is a 24-bit register for DSP-to-EFCOP constant transfers.
Filter count (FCNT) register	The FCNT is a 24-bit register that specifies the number of filter taps. The count stored in the FCNT register is used by the EFCOP address generation logic to generate correct addressing to the FDM and FCM.
EFCOP control status register (FCSR)	The FCSR is a 24-bit read/write register used by the DSP56300 core to program the EFCOP and to examine the status of the EFCOP module.
EFCOP ALU control register (FACR)	The FACR is a 24-bit read/write register used by the DSP56300 core to program the EFCOP data ALU operating modes.
EFCOP data buffer base address (FDBA)	The FDBA is a 16-bit read/write register used by the DSP56300 core to indicate the EFCOP the data buffer base start address pointer in FDM RAM.
EFCOP coefficient buffer base address (FCBA)	The FCBA is a 16-bit read/write register by which the DSP56300 core indicates the EFCOP coefficient buffer base start address pointer in FCM RAM.
Decimation/ channel count register (FDCH)	The FDCH is a 24-bit register that sets the number of channels in multichannel mode and the filter decimation ratio. The EFCOP address generation logic uses this information to supply the correct addressing to the FDM and FCM.

10.3.2 EFCOP Memory Banks

The EFCOP contains two memory banks:

- **Filter Data Memory (FDM)**—This 24-bit-wide memory bank is mapped as X memory and stores input data samples for EFCOP filter processing. The FDM is written via a 4-word FIFO (FDIR), and its addressing is generated by the EFCOP address generation logic. The input data samples are read sequentially from the FDM into the MAC. The FDM is accessible for writes by the core, and the DMA controller and is shared with the 4K lowest locations ($0–$FFF) of the on-chip internal X memory.
- **Filter Coefficient Memory (FCM)**—This 24-bit-wide memory bank is mapped as Y memory and stores filter coefficients for EFCOP filter processing. The FCM is written via the DSP56300 core, and the EFCOP address generation logic generates its addressing. The filter coefficients are read sequentially from the FCM into the MAC. The FCM is accessible for writes only by the core. The FCM is shared with the 4K lowest locations ($0–$FFF) of the on-chip internal Y memory.

Note: The filter coefficients, H(n), are stored in “reverse order,” where H(N-1) is stored at the lowest address of the FCM register as shown in **Figure 10-4**.

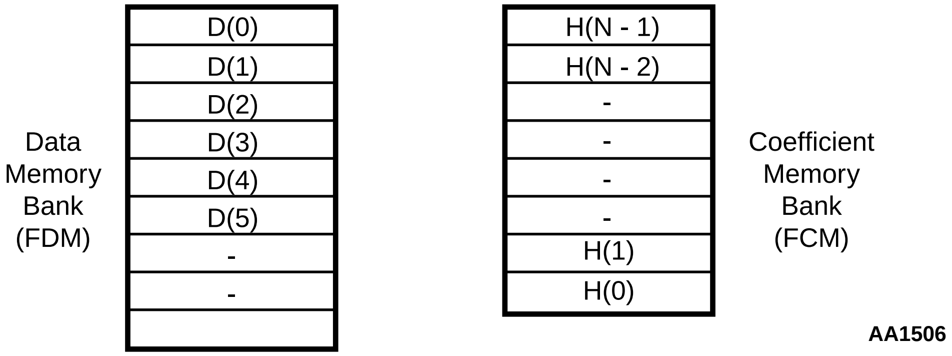


Figure 10-3 Storage of Filter Coefficients

The EFCOP connects to the shared memory in place of the DMA bus. Simultaneous accesses by core and EFCOP to the same memory module block (256 locations) of the shared memory are not permitted. It is the user’s responsibility to prevent such simultaneous accesses. **Figure 10-4** illustrates the shared memory between core and EFCOP.

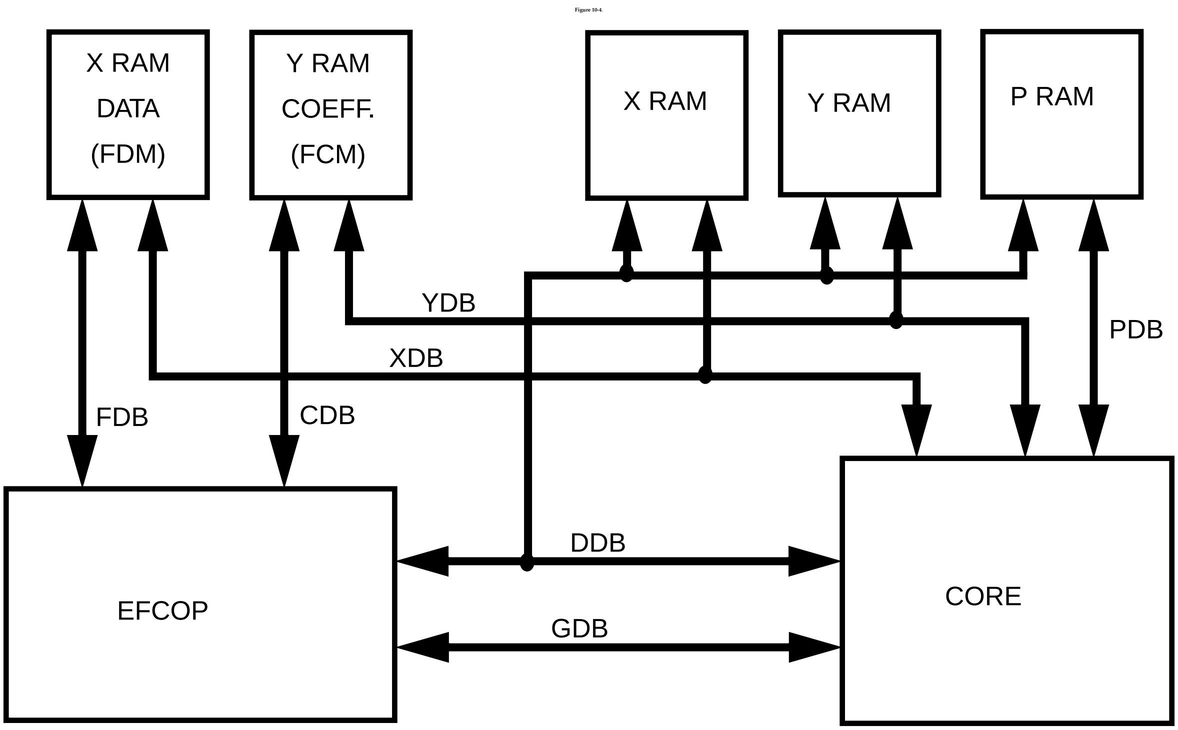


Figure 10-4 EFCOP Memory Organization

10.3.3 Filter Multiplier and Accumulator (FMAC)

The FMAC machine can perform a 24-bit × 24-bit multiplication with accumulation in a 56-bit accumulator. The FMAC operates a pipeline: the multiplication is performed in one clock cycle, and the accumulation occurs in the following clock cycle. Throughput is one MAC result per clock cycle. The two MAC operands are read from the FDM and from the FCM. The full 56-bit width of the accumulator is used for intermediate results during the filter calculations.

For operations in which saturation mode is disabled, the final result is rounded according to the selected rounding mode and limited to the most positive number ($7FFFFF, if overflow occurred) or most negative number ($800000, if underflow occurred) after processing of all filter taps is completed. In saturation mode, the result is limited to the most positive number ($7FFFFF, if overflow occurred), or the most negative number ($800000, if underflow occurred) after each MAC operation. The 24-bit result from the FMAC is stored in the EFCOP output buffer, FDOR.

When operating in sixteen-bit arithmetic mode, the FMAC performs a 16-bit × 16-bit multiplication with accumulation into a 40-bit accumulator. As with 24-bit operations, if saturation mode is disabled, the result is rounded according to the selected rounding mode and limited to the most positive number ($7FFF, if overflow occurred) or the most negative number ($8000, if underflow occurred) after processing of all filter taps is completed. In saturation mode, the result is limited to the most positive number ($7FFF, if overflow occurred) or the most negative number ($8000, if underflow occurred) after every MAC operation. The 16-bit result from the FMAC is stored in the EFCOP output buffer, FDOR.

10.4 EFCOP PROGRAMMING MODEL

This section documents the registers for configuring and operating the EFCOP. **Appendix D EFCOP Programming** on page D-1 discusses EFCOP programming. The EFCOP interrupt vector table is shown in **Table 10-7** on page 10-19. The EFCOP registers available to the DSP programmer are listed in **Table 10-2**. The following paragraphs describe these registers in detail.

Table 10-2 EFCOP Registers and Base Addresses

Address	EFCOP Register Name
Y:$FFFFB0	Filter data input register (FDIR)
Y:$FFFFB1	Filter data output register (FDOR)
Y:$FFFFB2	Filter K-constant register (FKIR)
Y:$FFFFB3	Filter count register (FCNT)
Y:$FFFFB4	Filter control status register (FCSR)
Y:$FFFFB5	Filter ALU control register (FACR)
Y:$FFFFB6	Filter data buffer base address (FDBA)
Y:$FFFFB7	Filter coefficient base address (FCBA)
Y:$FFFFB8	Filter decimation/channel register (FDCH)
NOTE: The EFCOP registers are mapped onto Y data memory space.	

10.4.1 Filter Data Input Register (FDIR)

The FDIR is a 4-word deep, 24-bit wide FIFO for DSP-to-EFCOP data transfers. Up to four data samples can be written into the FDIR at the same address. Data from the FDIR is transferred to the FDM for filter processing. For proper operation, write data to the FDIR only if the FDIBE status bit is set, indicating that the FIFO is empty. A write to the FDIR clears the FDIBE bit. Data transfers can be triggered by an interrupt request (for core transfers) or a DMA request (for DMA transfers). The FDIR is accessible for writes by the DSP56300 core and the DMA controller.

10.4.2 Filter Data Output Register (FDOR)

The FDOR is a 24-bit read-only register for EFCOP-to-DSP data transfers. The result of the filter processing is transferred from the FMAC to the FDOR. For proper operation, read data from the FDOR only if the FDOBF status bit is set, indicating that the FDOR contains data. A read from the FDOR clears the FDOBF bit. Data transfers can be triggered by an interrupt request (for core transfers) or a DMA request (DMA transfers). The FDOR is accessible for reads by the DSP56300 core and the DMA controller.

10.4.3 Filter K-Constant Input Register (FKIR)

The Filter K-Constant Input Register (FKIR) is a 24-bit write-only register for DSP-to-EFCOP data transfers in adaptive mode where the value stored in FKIR represents the weight update multiplier. FKIR is accessible only to the DSP core for reads or writes. When adaptive mode is enabled, the EFCOP immediately starts the coefficient update if a K-Constant value is written to FKIR. If no value is written to FKIR for the current data sample, the EFCOP halts processing until the K-Constant is written to FKIR. After the weight update multiplier is written to FKIR, the EFCOP transfers it to the FMAC unit and starts updating the filter coefficients according to the following equation:

New_coefficients = Old_coefficients + FKIR * Input_buffer

10.4.4 Filter Count (FCNT) Register

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

11	10	9	8	7	6	5	4	3	2	1	0
FCNT11	FCNT10	FCNT9	FCNT8	FCNT7	FCNT6	FCNT5	FCNT4	FCNT3	FCNT2	FCNT1	FCNT0

▨ = Reserved bit; read as 0; should be written with 0 for future compatibility

The FCNT register is a 24-bit read/write register used to select the filter length (number of filter taps). Always write the initial count into the FCNT register before you enable the EFCOP (i.e., before you set FEN).

Note: To ensure correct operation, never change the contents of the FCNT register unless the EFCOP is in the individual reset state (i.e., FEN = 0).

The number stored in FCNT is used by the EFCOP address generation logic to generate the correct addressing for the FDM and for the FCM.

Note: In the individual reset state (i.e., FEN = 0), the EFCOP module is inactive, but the contents of the FCNT register are preserved.

Table 10-3 FCNT Register Bit Descriptions

Bit #	Abbr.	Description
23–12	—	These bits are reserved and unused. They are read as 0 and should be written with 0 for future compatibility.
11–0	FCNT	**Filter Count**—The actual value written to the FCNT register must be the number of coefficient values minus one. The number of coefficient values is the number of locations used in the FCM. For a real FIR filter, the number of coefficient values is identical to the number of filter taps. For a complex FIR filter, the number of coefficient values is twice the number of filter taps.

10.4.5 EFCOP Control Status Register (FCSR)

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	FDOBF	FDIBE	FCONT	FSAT

11	10	9	8	7	6	5	4	3	2	1	0
FDOIE	FDIIE	▨	FSCO	FPRC	FMLC	FOM1	FOM0	FUPD	FADP	FLT	FEN

▨ = Reserved bit; read as 0; should be written with 0 for future compatibility

The FCSR is a 24-bit read/write register by which the DSP56300 core controls the main operation modes of the EFCOP and monitors the EFCOP status. The FCSR bits are described in the following paragraphs. All FCSR bits are cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Table 10-4 FCSR Bit Descriptions

Bit #	Abbr.	Description
23–16	—	These bits are reserved and unused. They are read as 0 and should be written with 0 for future compatibility.
15	FDOBF	**Filter Data Output Buffer Full**—This read-only status bit indicates, when set, that the FDOR is full and the DSP can read data from the FDOR. The FDOBF bit is set when a result from FMAC is transferred to the FDOR. For proper operation, read data from the FDOR only if the FDOBF status bit is set, indicating that the FDOR is full. A read from the FDOR clears the FDOBF bit. FDOBF is also cleared by a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, or an individual reset. When the FDOBF bit is set, the EFCOP generates an FDOBF interrupt request to the DSP56300 core if that interrupt is enabled (i.e., FDOIE = 1). A DMA request is always generated when the FDOBF bit is set, but a DMA transfer takes place only if a DMA channel is activated and triggered by this event.
14	FDIBE	**Filter Data Input Buffer Empty**—This read-only status bit indicates, when set, that the FDIR is empty and the DSP can write data to the FDIR. The FDIBE bit is set when all four FDIR locations are empty. For proper operation, write data to the FDIR only if the FDIBE status bit is set, indicating that the FDIR is empty. A write to the FDIR clears the FDIBE bit. The FDIBE bit is also cleared by a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, or an individual reset. After EFCOP is enabled by setting FEN, FDIBE is set, indicating that the FDIR is empty. When FDIBE is set, the EFCOP generates a FDIR empty interrupt request to the DSP56300 core if enabled (i.e., FDIIE = 1). A DMA request is always generated when the FDIBE bit is set, but a DMA transfer takes place only if a DMA channel is activated and triggered by this event.
13	FCONT	**Filter Contention**—This read-only status bit indicates, when set, that both the DSP56300 core and the EFCOP tried to access the same 256-word bank in either the shared FDM or FCM. A dual access could result in faulty data output in the FDOR. Once set, the FCONT bit is a sticky bit that can only be cleared by a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, or an individual reset.
12	FSAT	**Filter Saturation**—This read-only status bit indicates, when set, that an overflow or underflow occurred in the MAC result. When an overflow occurs, the FSAT bit is set, and the result is saturated to the most positive number (i.e., \$7FFFFF). When an underflow occurs, the FSAT bit is set, and the result is saturated to the most negative number (i.e., \$800000). FSAT is a sticky status bit that is set by hardware and can be cleared only by a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, or an individual reset.

Table 10-4 FCSR Bit Descriptions (Continued)

Bit #	Abbr.	Description
11	FDOIE	**Filter Data Output Interrupt Enable**—This read/write control bit is used to enable the filter data output interrupt. If FDOIE is cleared, the filter data output interrupt is disabled, and the FDOBF status bit should be polled to determine whether the FDOR is full. If both FDOIE and FDOBF are set, the EFCOP requests a data output buffer full interrupt service from the DSP56300 core. A DMA transfer is enabled if a DMA channel is activated and triggered by this event. Note: For proper operation, enable the interrupt service routine and the corresponding interrupt for core processing *or* enable the DMA transfer and configure the proper trigger for the selected channel. *Never* enable both simultaneously.
10	FDIIE	**Filter Data Input Interrupt Enable**—This read/write control bit is used to enable the data input buffer empty interrupt. If FDIIE is cleared, the data input buffer empty interrupt is disabled, and the FDIBE status bit should be polled to determine whether the FDIR is empty. If both FDIIE and FDIBE are set, the EFCOP requests a data input buffer empty interrupt service from the DSP56300 core. DMA transfer is enabled if a DMA channel is activated and triggered by this event. Note: For proper operation, enable the interrupt service routine and the corresponding interrupt for core processing *or* enable the DMA transfer and configure the proper trigger for the selected channel. *Never* enable both simultaneously.
9	—	This bit is reserved. It is read as 0 and should be written with 0 for future compatibility.
8	FSCO	**Filter Shared Coefficients Mode**—This read/write control bit is valid only when the EFCOP is operating in multichannel mode (i.e., FMLC is set). When the FSCO bit is set, the EFCOP uses the coefficients in the same memory area (i.e., the same coefficients) to implement the filter for each channel. This mode is used when several channels are being filtered through the same filter. When the FSCO bit is cleared, the EFCOP filter coefficients are stored sequentially in memory for each channel. Note: To ensure proper operation, never change the FSCO bit unless the EFCOP is in individual reset state (i.e., FEN = 0). FSCO is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Table 10-4 FCSR Bit Descriptions (Continued)

Bit #	Abbr.	Description
7	FPRC	**Filter Processing (FPRC) State Initialization Mode**—This read/write control bit defines the EFCOP processing initialization mode. When this bit is cleared, the EFCOP starts processing after a state initialization. (The EFCOP machine starts computing once the FDM bank contains N input samples for an N tap filter). When this bit is set, the EFCOP starts processing with no state initialization. (The EFCOP machine starts computing as soon as the first data sample is available in the input buffer.) Note: To ensure proper operation, never change the FPRC bit unless the EFCOP is in individual reset state (i.e., FEN = 0). FPRC is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
6	FMLC	**Filter Multichannel (FMLC) Mode**—This read/write control bit, when set, enables multichannel mode, allowing the EFCOP to process several filters (defined by FCHL[5:0] bits in FDCH register) concurrently by sequentially entering a different sample to each filter. If FMLC is cleared, multichannel mode is disabled, and the EFCOP operates in single filter mode. Note: To ensure proper operation, never change the FMLC bit unless the EFCOP is in individual reset state (i.e., FEN = 0). FMLC is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
5–4	FOM	**Filter Operation Mode**—This pair of read/write control bits defines one of four operation modes if the FIR filter has been selected (i.e., FLT is cleared): • FOM = 00—Mode 0: Real FIR filter • FOM = 01—Mode 1: Full complex FIR filter • FOM = 10—Mode 2: Complex FIR filter with alternate real and imaginary outputs • FOM = 11—Mode 3: Magnitude Note: To ensure proper operation, never change the FOM bits unless the EFCOP is in the individual reset state (i.e., FEN = 0). The FOM bits are cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
3	FUPD	**Filter Update**—When set, this read/write control/status bit enables the EFCOP to start a single coefficient update session. Upon completion of the session, the FUPD bit is automatically cleared. FUPD is set automatically when the EFCOP is in adaptive mode (i.e., FADP = 1). FUPD is cleared by a hardware $\overline{\text{RESET}}$ signal, a software RESET instruction, or an individual reset.

Table 10-4 FCSR Bit Descriptions (Continued)

Bit #	Abbr.	Description
2	FADP	**Filter Adaptive (FADP) Mode**—When set, this read/write control bit enables adaptive mode. adaptive mode provides an efficient way to implement a LMS-type filter, and therefore it is used when the EFCOP operates in FIR filter mode (FLT = 0). In adaptive mode, processing of every input data sample consists of FIR processing followed by a coefficient update. When this bit is set, the EFCOP completes the FIR processing on the current data sample and immediately starts the coefficient update assuming that a K-constant value is written to the FKIR. If no value has been written to the FKIR for the current data sample, the EFCOP halts processing until the K-constant is written to the FKIR. During the coefficient update, the FUPD bit is automatically set to indicate an update session. After completion of the update, the EFCOP starts processing the next data sample. FADP is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
1	FLT	**Filter (FLT) Type**—This read/write control bit selects one of two available filter types: • FLT = 0—FIR filter • FLT = 1—IIR filter Note: To ensure proper operation, never change the FLT bit unless the EFCOP is in the individual reset state (i.e., FEN = 0). The FLT bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
0	FEN	**Filter Enable**—This read/write control bit, when set, enables the operation of the EFCOP. When FEN is cleared, operation is disabled and the EFCOP is in the individual reset state. Note: While in the individual reset state, the EFCOP is inactive, internal logic and status bits assume the same state as that produced by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction, the contents of the FCNT, FDBA, and FCBA registers are preserved, and the control bits in FCSR and FACR remain unchanged.

10.4.6 EFCOP ALU Control Register (FACR)

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

11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	FISL	FSA	FSM	FRM1	FRM0	FSCL1	FSCL0

▨ = Reserved bit; read as 0; should be written with 0 for future compatibility

▣ = Reserved for internal use; read as 0; should be written with 0 for proper use.

The FACR is a 24-bit read/write register by which the DSP56300 core controls the main operation modes of the EFCOP ALU. The FACR bits are described in the following paragraphs. All FACR bits are cleared after hardware and software reset.

Table 10-5 FACR Bit Description

Bit #	Abbr.	Description
23–16	—	These bits are reserved. They are read as 0 and should be written with 0 for future compatibility.
15–12	—	These bits are reserved for internal use and should always be written as 0 for proper operation.
11–7	—	These bits are reserved and unused. They read as 0 and should be written with 0 for future compatibility.
6	FISL	**Filter Input Scale**—When set, this read/write control bit directs the EFCOP ALU to scale the IIR feedback terms but not the IIR input. When cleared, the EFCOP ALU scales both the IIR feedback terms and the IIR input. The scaling value in both cases is determined by the FSCL[1:0] bits. The FISL bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
5	FSA	**Filter Sixteen-bit Arithmetic (FSA) Mode**—When set, this read/write control bit enables FSA mode. In this mode, the rounding of the arithmetic operation is performed on Bit 31 of the 56-accumulator instead of the usual bit 23 of the 56-bit accumulator. The scaling of the EFCOP data ALU is affected accordingly. The FSA bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
4	FSM	**Filter Saturation Mode—**When set, this read/write control bit selects automatic saturation on 48 bits for the results going to the accumulator. A special circuit inside the EFCOP MAC unit then saturates those results. The purpose of this bit is to provide arithmetic saturation mode for algorithms that do not recognize or cannot take advantage of the extension accumulator. The FSM bit is cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

Table 10-5 FACR Bit Description

Bit #	Abbr.	Description
3–2	FRM	**Filter Rounding Mode**—These read/write control bits select the type of rounding performed by the EFCOP data ALU during arithmetic operation: • FRM = 00—Convergent rounding • FRM = 01—Two's complement rounding • FRM = 10—Truncation (no rounding) • FRM = 11—Reserved for future expansion These bits affect operation of the EFCOP data ALU. The FRM bits are cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.
1–0	FSCL	**Filter Scaling (FSCL)**—These read/write control bits select the scaling factor of the FMAC result: • FSCL = 00—Scaling factor = 1 (no shift) • FSCL = 01—Scaling factor = 8 (3-bit arithmetic left shift) • FSCL = 10—Scaling factor = 16 (4-bit arithmetic left shift) • FSCL = 11—Reserved for future expansion Note: To ensure proper operation, never change the FSCL bits unless the EFCOP is in the individual reset state (i.e., FEN = 0). The FSCL bits are cleared by a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction.

10.4.7 EFCOP Data Base Address (FDBA)

The FDBA is a 16-bit read/write counter register used as an address pointer to the EFCOP FDM bank. FDBA points to the location to write the next data sample. The FDBA points to a modulo delay buffer of size M, defined by the filter length (M = FCNT[11:0] + 1). The address range of this modulo delay buffer is defined by lower and upper address boundaries. The lower address boundary is the FDBA value with 0 in the k-LSBs, where $2^k \geq M \geq 2^{k-1}$; it therefore must be a multiple of 2^k. The upper boundary is equal to the lower boundary plus (M – 1). Since $M \leq 2^k$, once M has been chosen (i.e., FCNT has been assigned), a sequential series of data memory blocks (each of length 2^k) will be created where multiple circular buffers for multichannel filtering can be located. If $M < 2^k$, there will be a space between sequential circular buffers of $2^k - M$. The address pointer is not required to start at the lower address boundary or to end on the upper address boundary. It can point anywhere within the defined modulo address range. If the data address pointer (FDBA) increments and reaches the upper boundary of the modulo buffer, it will wrap around to the lower boundary.

10.4.8 EFCOP Coefficient Base Address (FCBA)

The FCBA is a 16-bit read/write counter register used as an address pointer to the EFCOP FCM bank. FCBA points to the first location of the coefficient table. The FCBA points to a modulo buffer of size M, defined by the filter length (M = FCNT[11:0] + 1). The address range of this modulo buffer is defined by lower and upper address boundaries. The lower address boundary is the FCBA value with 0 in the k-LSBs, where $2^k \geq M \geq 2^{k-1}$; it therefore must be a multiple of 2^k. The upper boundary is equal to the lower boundary plus (M – 1). Since $M \leq 2^k$, once M has been chosen (i.e., FCNT has been assigned), a sequential series of coefficient memory blocks (each of length 2^k) is created where multiple circular buffers for multichannel filtering can be located. If $M < 2^k$, there will be a space between sequential circular buffers of $2^k - M$. The FCBA address pointer must be assigned to the lower address boundary (i.e., it must have 0 in its k-LSBs). In a compute session, the coefficient address pointer always starts at the lower boundary and ends at the upper address boundary. Therefore, a FCBA read always gives the value of the lower address boundary.

10.4.9 Decimation/Channel Count Register (FDCH)

The FDCH is a 24-bit read/write register used to set the number of channels used in multichannel mode (FCHL) and to set the decimation ratio in FIR filter mode. FDCH should be written before the EFCOP is enabled by FEN. FDCH should be changed only when EFCOP is in individual reset state (FEN = 0); otherwise, improper operation may result. The number stored in FCHL is used by the EFCOP address generation logic to generate the correct address for the FDM bank and for the FCM bank in multichannel mode. When the EFCOP enable bit (FEN) is cleared, the EFCOP is in individual reset state. In this state, the EFCOP is inactive, and the contents of FDCH register are preserved.

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

11	10	9	8	7	6	5	4	3	2	1	0
FDCM3	FDCM2	FDCM1	FDCM0	▨	▨	FCHL5	FCHL4	FCHL3	FCHL2	FCHL1	FCHL0

▨ = Reserved bit; read as 0; should be written with 0 for future compatibility

Table 10-6 FDCH Register Bit Descriptions

Bit #	Abbr.	Description
23–12	—	These bits are reserved and unused. They are read as 0 and should be written with 0 for future compatibility.
11–8	FDCM	**Filter Decimation**—These read/write control bits select the decimation function. There are 16 decimation factor options (from 1 to 16). Note: To ensure proper operation, never change the FDCM bits unless the EFCOP is in the individual reset state (FEN = 0). The bits are cleared by a hardware $\overline{\text{RESET}}$ or a software RESET instruction.
7–6	—	These bits are reserved and unused. They are read as 0 and should be written with 0 for future compatibility.
5–0	FCHL	**Filter Channels**—These read/write control bits determine the number of filter channels to process simultaneously (from 1 to 64) in multichannel mode. The number represented by the FCHL bits is one less than the number of channels to be processed; that is, if FCHL = 0, then 1 channel is processed; if FCHL =1, then 2 channels are processed; and so on. Note: To ensure proper operation, never change the FCHL bits unless the EFCOP is in the individual reset state (FEN = 0). The bits are cleared by a hardware $\overline{\text{RESET}}$ or a software RESET instruction.

10.4.10 EFCOP Interrupt Vectors

Table 10-7 shows the EFCOP interrupt vectors, and **Table 10-8** shows the DMA request sources.

Table 10-7 EFCOP Interrupt Vectors

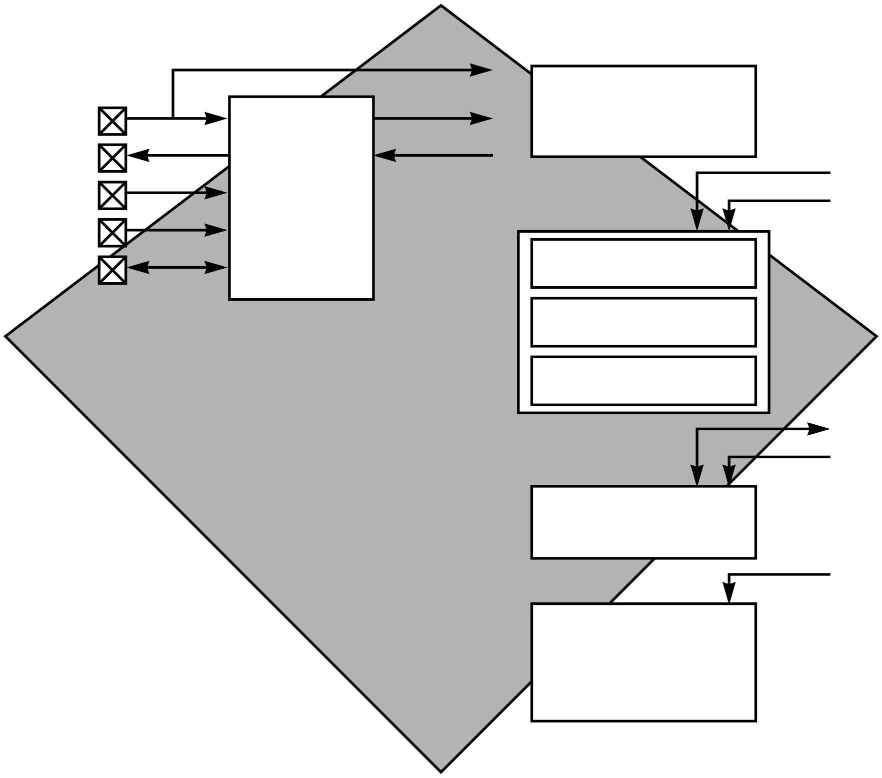
Interrupt Address	Interrupt Vector	Priority	Interrupt Enable	Interrupt Conditions
VBA + $68	Data input buffer empty	Highest	FDIIE	FDIBE = 1
VBA + $6A	Data output buffer full	Lowest	FDOIE	FDOBF = 1

Table 10-8 EFCOP DMA Request Sources

Requesting Device Number	Request Conditions	Peripheral Request MDRQ
EFCOP input buffer empty	FDIBE = 1	MDRQ11
EFCOP output buffer full	FDOBF = 1	MDRQ12

SECTION 11

ON-CHIP EMULATION MODULE



11.1 INTRODUCTION

The DSP56300 core OnCE module interacts with the DSP56300 core and its peripherals non-intrusively so that you can examine registers, memory, or on-chip peripherals, thus facilitating hardware and software development on the DSP56300 core processor. Special circuits and dedicated signals on the DSP56300 core are defined to avoid sacrificing any user-accessible on-chip resource. OnCE module resources are accessible only after execution of the JTAG instruction ENABLE_ONCE; these resources are accessible even when the chip operates in normal mode. See **Section 12 Joint Test Action Group Port** for a description of JTAG and its relation to OnCE. **Figure 11-1** shows the block diagram of the OnCE module.

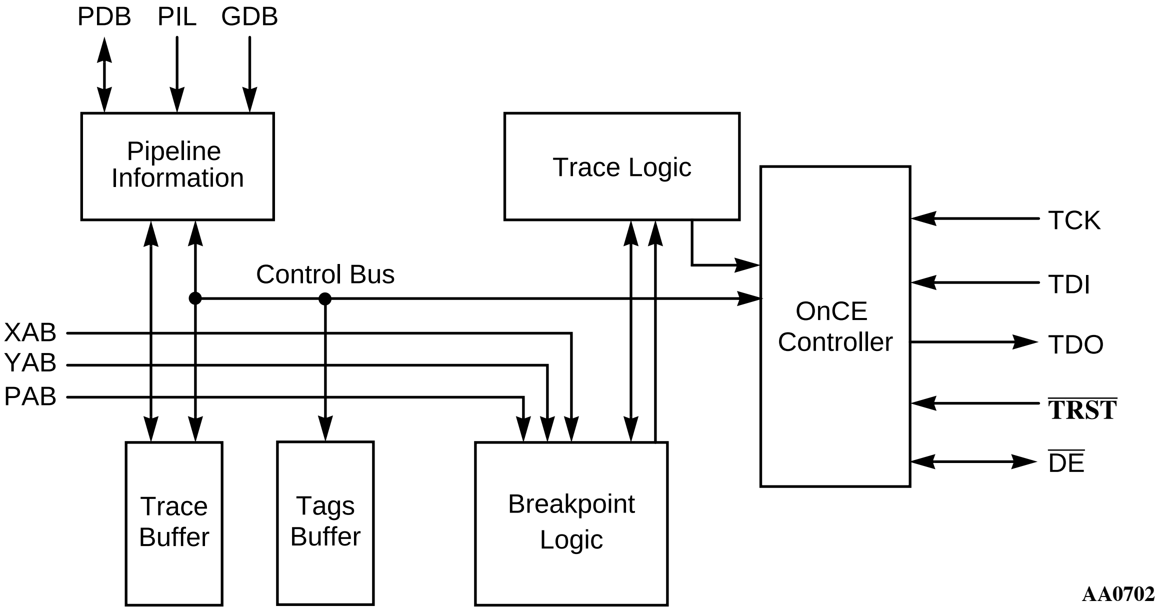


Figure 11-1 OnCE Module Block Diagram

11.2 OnCE MODULE SIGNALS

The OnCE module controller is accessed through the JTAG port. There are no dedicated OnCE module signals for the clock, data in, or data out. The JTAG TCK, TDI, and TDO signals shift data and instructions in and out. See **Section 12.2 JTAG Signals** on page 12-4 for documentation of the JTAG signals. One additional signal, called $\overline{\text{DE}}$ facilitates emulation-specific functions.

11.3 DEBUG EVENT

The debug event signal ($\overline{\text{DE}}$) is a bidirectional open drain. As input, it provides a fast means of entering debug mode from an external command controller; as output, it provides a fast means of acknowledging to an external command controller that the chip has entered debug mode. When a command controller asserts this signal, the DSP56300 core finishes the current instruction being executed, saves the instruction pipeline information, enters debug mode, and waits for commands to be entered from the TDI line. If the $\overline{\text{DE}}$ signal is used to enter debug mode, then it must be deasserted after the OnCE port responds with an acknowledge and before the first OnCE command is sent. 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 $\overline{\text{DE}}$ signal also facilitates multiple processor connections, as shown in **Figure 11-2**.

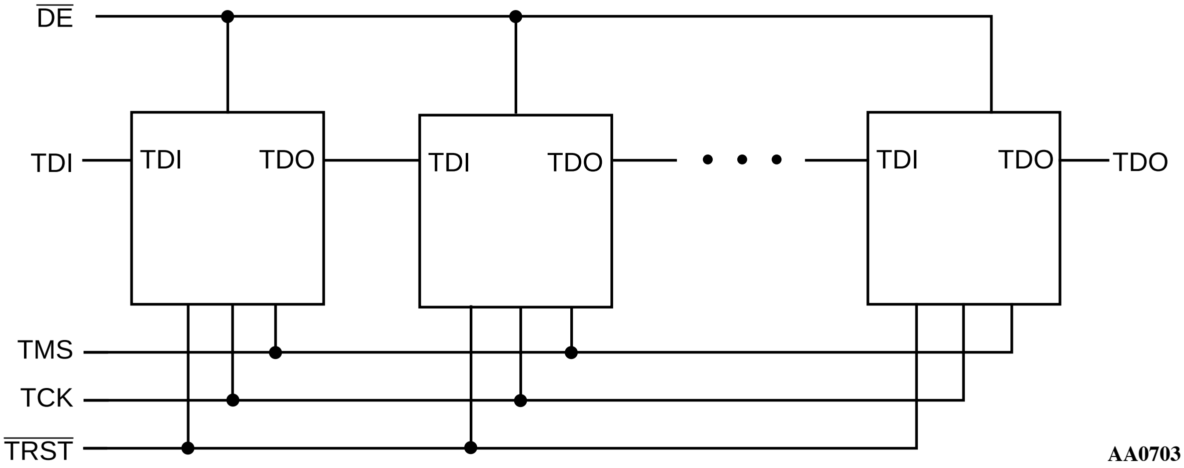


Figure 11-2 OnCE Module Multiprocessor Configuration

Thus you can stop all the devices in the system when one of the devices enters debug mode. You can also stop all devices synchronously by asserting the $\overline{\text{DE}}$ line.

11.4 OnCE CONTROLLER

The OnCE controller contains the following blocks: OnCE command register (OCR), OnCE decoder, and the status/control register. **Figure 11-3** shows a block diagram of the OnCE controller.

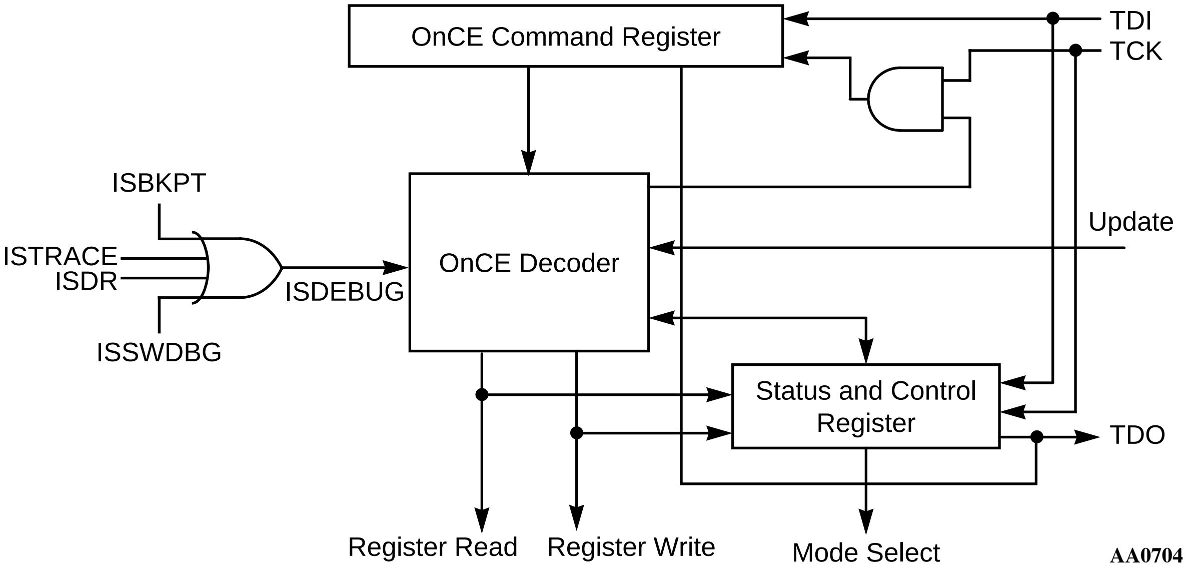


Figure 11-3 OnCE Controller Block Diagram

11.4.1 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 11-4**.

OCR
OnCE Command
Register
Reset = $00
Write Only

7	6	5	4	3	2	1	0
R/W	GO	EX	RS4	RS3	RS2	RS1	RS0

AA0106

Figure 11-4 OnCE Command Register

11.4.1.1 Register Select (RS4–RS0) Bits 0–4

The register select bits define which register is the source or destination for read or write operations. See **Table 11-1** for the OnCE register select encoding.

Table 11-1 OnCE Register Select Encoding

RS[4:0]	Register Selected
00000	OnCE status and control register (OSCR)
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

11.4.1.2 Exit Command (EX) Bit 5

If the EX bit is set, the core leaves debug mode and resumes normal operation. The EXIT command is executed only if the GO command is issued and the operation is a write to OPDBR or a read/write to "no register selected". Otherwise, the EX bit is ignored. **Table 11-2** shows the values of the EX bit.

Table 11-2 EX Bit Definition

EX	Action
0	Remain in debug mode
1	Leave debug mode

11.4.1.3 GO Command (GO) Bit 6

If the GO bit is set, the core executes the instruction that resides in the PIL register. To execute the instruction, the core leaves debug mode. If the EX bit is cleared, the core returns to debug mode immediately after executing the instruction. If the EX bit is set, the core goes on to normal operation. The GO command is executed only if the operation is a write to OPDBR or a read/write to "no register selected". Otherwise, the GO bit is ignored. **Table 11-3** shows the values of the GO bit.

Table 11-3 GO Bit Definition

GO	Action
0	Inactive—no action taken
1	Execute instruction in PIL

11.4.1.4 Read/Write Command (R/$\overline{W}$) Bit 7

The R/$\overline{W}$ bit specifies the direction of a data transfer.

Table 11-4 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.

11.4.2 OnCE Decoder (ODEC)

The ODEC supervises the all OnCE module activity. It receives as input an 8-bit command from the OCR, a signal from JTAG controller (indicating that 8 or 24 bits have been received and that an update of the selected data register must be performed), and a signal indicating that the core has halted. The ODEC generates all the strobes required to read and write the selected OnCE registers.

11.4.3 OnCE Status and Control Register (OSCR)

The OSCR is a 24-bit register that enables trace mode and indicates the reason to enter debug mode. The control bits can be read or written, whereas the status bits are read-only. The OSCR bits are cleared by a hardware $\overline{\text{RESET}}$ signal. The OSCR is shown in **Figure 11-5**.

	23			• • •			9	8	7	6	5	4	3	2	1	0
OnCE Status and Control Register Read/Write									OS1	OS0		TO	MBO	SWO	IME	TME

▨ Reserved bits; read as 0; should be written as 0 for future compatibility AA0705

Figure 11-5 OnCE Status and Control Register (OSCR)

11.4.3.1 Trace Mode Enable (TME) Bit 0

When set, the TME control bit enables trace mode.

11.4.3.2 Interrupt Mode Enable (IME) Bit 1

When the IME control bit is set, the chip executes a vectored interrupt at the address VBA:$06 instead of entering debug mode.

11.4.3.3 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 the core leaves debug mode.

11.4.3.4 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 the core leaves debug mode.

11.4.3.5 Trace Occurrence (TO) Bit 4

The TO bit is a read-only status bit that is set when debug mode is entered because the trace counter is 0 while trace mode is enabled. This bit is cleared when the core leaves debug mode.

11.4.3.6 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.

11.4.3.7 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, you can determine whether the chip has entered debug mode. SWO, MBO, and TO identify the reason for entering debug mode. You can also examine these bits to determine why the chip has not entered debug mode after debug event assertion ($\overline{\text{DE}}$) or after execution of the JTAG debug request instruction (e.g., core waiting for the bus, STOP or WAIT instruction, etc.). These bits are also reflected in the JTAG instruction shift register, so you can poll core status information at the JTAG level. This convention is useful to allow a read of OSCR when the DSP56300 core executes the STOP instruction (and therefore there are no clocks). See **Table 11-5** for the values of the OS0–OS1 bits.

Table 11-5 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 the bus
1	1	DSP56300 core is in debug mode

11.4.3.8 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.

11.5 OnCE MEMORY BREAKPOINT LOGIC

Memory breakpoints can be set on locations in program memory or data memory. In addition, the breakpoint does not have to be in a specific memory address, but within an approximate address range of where the program may be executing. This flexibility significantly increases the programmer's ability to monitor what the program is doing in real-time.

The breakpoint logic, illustrated in **Figure 11-6**, contains a latch for addresses of registers that store the upper and lower address limits, address comparators, and a breakpoint counter.

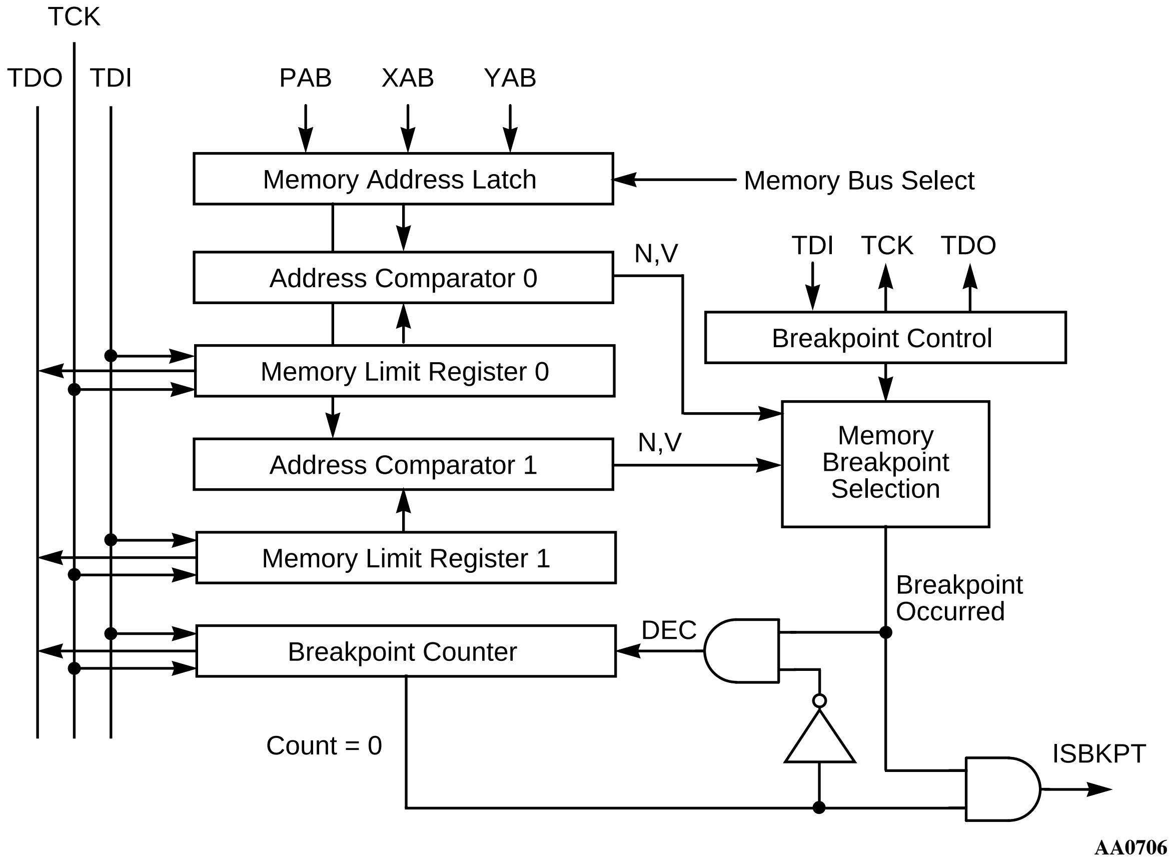


Figure 11-6 OnCE Memory Breakpoint Logic 0

Address comparators are useful to determine 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 or change registers or memory. Using address comparators to set breakpoints enables the user to set breakpoints in RAM or ROM in any operating mode. Memory accesses are monitored according to the contents of the OBCR as specified in **Section 11.5.6 OnCE Breakpoint Control Register (OBCR)** on page 11-11.

11.5.1 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.

11.5.2 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 breakpoints are enabled, OMLR0 must be loaded by the external command controller.

11.5.3 OnCE Memory Address Comparator 0 (OMAC0)

The OMAC0 compares the current memory address (stored in OMAL0) with the OMLR0 contents.

11.5.4 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 breakpoints are enabled, OMLR1 must be loaded by the external command controller.

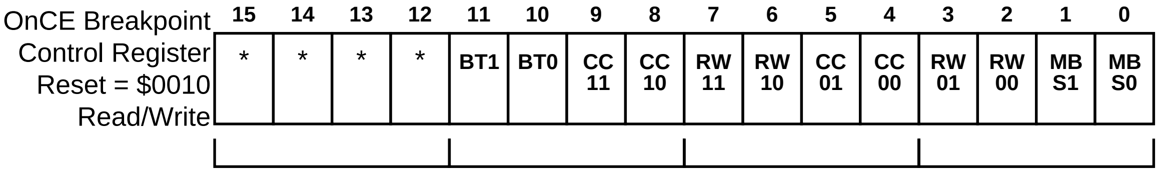
11.5.5 OnCE Memory Address Comparator 1 (OMAC1)

The OMAC1 compares the current memory address (stored in OMAL0) with the OMLR1 contents.

11.5.6 OnCE Breakpoint Control Register (OBCR)

The OBCR is a 16-bit register that defines memory breakpoint events. OBCR can be read or written through the JTAG port. OBCR bits are cleared by a hardware $\overline{\text{RESET}}$ signal.

The OBCR is illustrated in **Figure 11-7**.



* Reserved bits; read as 0; should be written as 0 for future compatibility

Figure 11-7 OnCE Breakpoint Control Register (OBCR)

11.5.6.1 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 in P, X, or Y space. **Table 11-6** lists the values of the MBS0–MBS1 bits.

Table 11-6 Memory Breakpoint 0 and 1 Select

MBS1	MBS0	Description
0	0	Reserved
0	1	Breakpoint on P access
1	0	Breakpoint on X access
1	1	Breakpoint on Y access

11.5.6.2 Breakpoint 0 Read/Write Select (RW00–RW01) Bits 2–3

The RW00–RW01 bits define the memory breakpoints 0 to occur when a memory address access is performed to read, write, or both. **Table 11-7** lists the values of the RW00–RW01 bits.

Table 11-7 Breakpoint 0 Read/Write Select

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

11.5.6.3 Breakpoint 0 Condition Code Select (CC00–CC01) Bits 4–5

The CC00–CC01 bits define what to do after a comparison of a current memory address (OMAL0) and the memory limit register 0 (OMLR0). **Table 11-8** lists the values of the CC00–CC01 bits.

Table 11-8 Breakpoint 0 Condition Select

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

11.5.6.4 Breakpoint 1 Read/Write Select (RW10–RW11) Bits 6–7

The RW10–RW11 bits make a memory breakpoint 1 occur when a memory address access is performed to read, write, or both. **Table 11-9** lists the values of the RW10–RW11 bits.

Table 11-9 Breakpoint 1 Read/Write Select

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

11.5.6.5 Breakpoint 1 Condition Code Select (CC10–CC11) Bits 8–9

The CC10–CC11 bits define what to do after a comparison of a current memory address (OMAL0) and the OnCE memory limit register 1 (OMLR1). **Table 11-10** lists values of the CC10–CC11 bits.

Table 11-10 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

11.5.6.6 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 breakpoint counter (OMBC) is decremented. **Table 11-11** lists values of the BT0–BT1 bits.

Table 11-11 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

11.5.6.7 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 decrements. 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 the limit register changes or a different breakpoint event is selected in the OBCR, the breakpoint counter must be written afterwards. This convention insures that the

OnCE breakpoint logic is reset and that no previous events can affect the newly selected breakpoint event. The breakpoint counter is cleared by a hardware $\overline{\text{RESET}}$ signal.

11.5.6.8 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.

11.6 OnCE TRACE LOGIC

The OnCE trace logic makes it possible to execute instructions in single or multiple steps. The OnCE trace logic causes the chip to enter debug mode after the execution of one or more instructions and to wait for OnCE commands from the debug serial port. The OnCE trace logic block diagram appears in **Figure 11-8**.

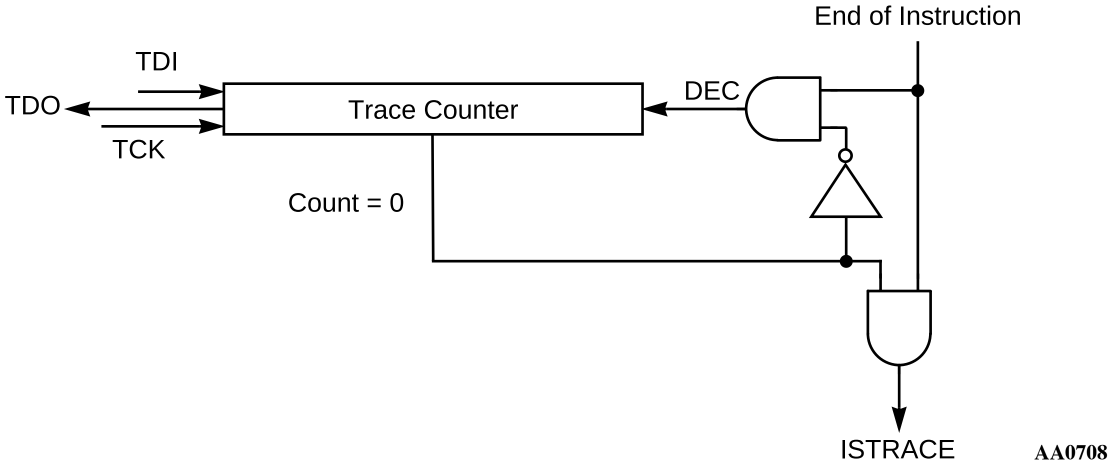


Figure 11-8 OnCE Trace Logic Block Diagram

Trace mode has an associated counter so that more than one instruction can be executed before the core returns to debug mode. This counter allows you to take multiple instruction steps in real time before entering debug mode. This feature helps you debug sections of code that do not have a normal flow or are looping. The trace counter also enables you 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 in 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.

When the core exits debug mode, the counter decrements after each execution of an instruction. Interrupts are serviceable, and all executed instructions (including fast interrupt services and repeated instructions) cause the trace counter to be decremented. When the counter decrements 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 $\overline{\text{DE}}$ signal asserts 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 execute before debug mode, the trace counter should be loaded with N – 1. The trace counter is cleared by a hardware $\overline{\text{RESET}}$ signal.

11.7 METHODS OF ENTERING DEBUG MODE

The DSP56307 acknowledges that it has entered debug mode by setting the core status bits OS1 and OS0 and asserting the $\overline{\text{DE}}$ line. This acknowledgment 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 so that the OnCE module remains inactive until the core executes a write operation to the OGDBRDSP56300.

11.7.1 External Debug Request during RESET Assertion

If the $\overline{\text{DE}}$ line is held asserted during the assertion of $\overline{\text{RESET}}$, then the chip enters debug mode. After receiving the acknowledgment, the external command controller must negate the $\overline{\text{DE}}$ line before sending the first command.

Note: In this case, the chip does not execute any instructions before entering debug mode.

11.7.2 External Debug Request during Normal Activity

If the $\overline{\text{DE}}$ line is held asserted during normal chip activity, the chip finishes executing the current instruction and enters debug mode. After receiving the acknowledge, the external command controller must negate the $\overline{\text{DE}}$ line before sending the first command. This process is the same for any newly-fetched instruction, including instructions fetched by interrupt processing or instructions to be aborted by interrupt processing.

Note: In this case, the chip finishes executing the current instruction and stops after the newly fetched instruction enters the instruction latch.

11.7.3 Executing the JTAG DEBUG_REQUEST Instruction

Execution of the JTAG instruction DEBUG_REQUEST asserts an internal debug request signal. Consequently, the chip finishes execution of the current instruction and stops after the newly-fetched instruction enters the instruction latch. After the core enters debug mode, the core status bits OS1 and OS0 are set, and the $\overline{\mathrm{DE}}$ line is asserted, thus acknowledging to the external command controller that debug mode has been entered.

11.7.4 External Debug Request during Stop

Execution of the JTAG instruction DEBUG_REQUEST (or assertion of $\overline{\mathrm{DE}}$) while the chip is in stop state (i. e., it has executed a STOP instruction) causes the chip to exit stop state and enter debug mode. After receiving the acknowledgment, the external command controller must negate $\overline{\mathrm{DE}}$ before sending its first command.

Note: In this case, the chip finishes executing of the STOP instruction and halts after the next instruction enters the instruction latch.

11.7.5 External Debug Request during Wait

Execution of the JTAG instruction DEBUG_REQUEST (or assertion of $\overline{\mathrm{DE}}$) while the chip is in wait state (i. e., it has executed a WAIT instruction) causes the chip to exit wait state and enter debug mode. After receiving the acknowledgment, the external command controller must negate $\overline{\mathrm{DE}}$ before sending its first command.

Note: In this case, the chip completes execution of the WAIT instruction and halts after the next instruction enters the instruction latch.

11.7.6 Software Request during Normal Activity

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 enters the instruction latch.

11.7.7 Enabling Trace Mode

When trace mode is enabled and the trace counter is greater than zero, the trace counter decrements after the execution of each instruction. Execution of an instruction when the value in the trace counter is 0 causes the chip to enter debug mode after it completes the execution of the instruction. Only instructions actually executed cause the trace counter to decrement. An aborted instruction does not decrement the trace counter, nor does it cause the chip to enter debug mode.

11.7.8 Enabling Memory Breakpoints

When the memory breakpoint mechanism is enabled with a breakpoint counter value of 0, the chip enters debug mode after it finishes executing 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 fetched instruction executes. 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.

11.8 PIPELINE INFORMATION AND OGDB REGISTER

To restore the pipeline and to resume normal chip activity upon a return from debug mode, a number of on-chip registers store the chip pipeline status. **Figure 11-9** shows the block diagram of the pipeline information registers (with the exception of the PAB registers shown in **Figure 11-10** on page 11-22).

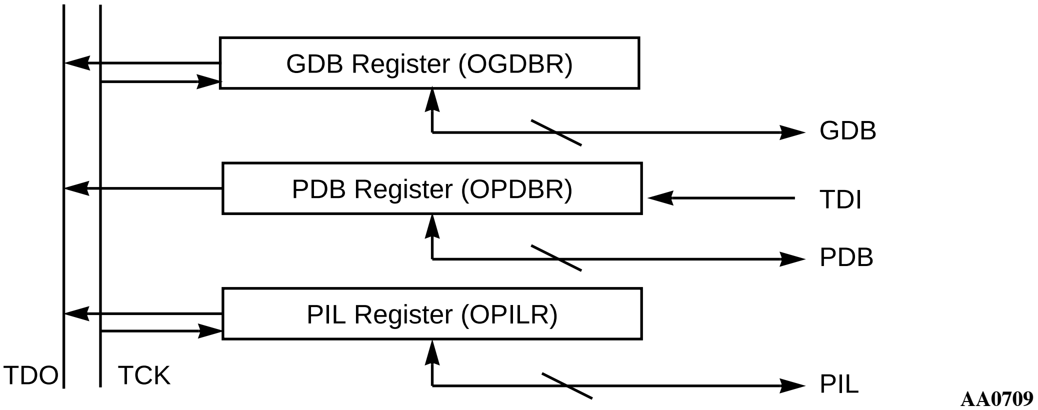


Figure 11-9 OnCE Pipeline Information and GDB Registers

11.8.1 OnCE PDB Register (OPDBR)

The OPDBR is a 24-bit latch that stores the value of the PDB 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 at a return to normal mode.

11.8.2 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 be read only through the JTAG port.

Note: Since the instruction latch is affected by operations performed during debug mode, it must be restored by the external command controller at a return to normal mode. Since there is no direct write access to the instruction latch, it is restored by a write to OPDBR with no-GO and no-EX. In this case, the data written on PDB is transferred into the instruction latch.

11.8.3 OnCE GDB Register (OGDBR)

The OGDBR is a 16-bit latch that can be read only through the JTAG port. The OGDBR is not actually required for restoring pipeline status, but it is required as a means of passing information between the chip and the external command controller. The OGDBR is mapped to 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`.

11.9 TRACE BUFFER

To make debugging easier and to keep track of program flow, the DSP56300 core provides a number of on-chip dedicated resources. Three read-only PAB registers give pipeline information when debug mode is entered, and a trace buffer stores the address of the last instruction that was executed as well as the addresses of the last 12 change-of-flow instructions.

11.9.1 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 be read only through the JTAG port. This register is not affected by the operations performed during debug mode.

11.9.2 PAB Register for Decode (OPABDR)

The OPABDR is a 16-bit register that stores the address of the instruction currently in the PDB. The fetch for this instruction was completed before the chip entered debug mode. The OPABDR can be read only through the JTAG port. This register is not affected by the operations performed during debug mode.

11.9.3 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 instruction would have been decoded and executed if the chip had not entered debug mode. The OPABEX register can be read only through the JTAG port. This register is not affected by operations performed during debug mode.

11.9.4 Trace Buffer

The trace buffer stores the addresses of the last 12 change-of-flow instructions executed, as well as the address of the last executed instruction. The trace buffer is a circular buffer containing 17-bit registers and one 4-bit counter. All the registers have the same address, but any read access to the trace buffer address increments the counter, 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 11-10** on page 11-22 shows the block diagram of the trace buffer. The trace buffer is not affected by operations performed during debug mode except for the trace buffer pointer increment when the trace buffer is being read. When the chip enters debug mode, the trace buffer counter points 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 ensure trace buffer coherence, a complete set of 12 reads of the trace buffer must be performed. Twelve reads are necessary because each read increments the trace buffer pointer, thus pointing to the next location. After 12 reads, the pointer indicates the same location as it did before the read procedure started.

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 changes-of-flow, the address of the change-of-flow instruction is always stored (regardless of whether 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 includes “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 you read the 12 trace buffer registers. Since data is read LSB first, the invalid bit is the first bit to be read.

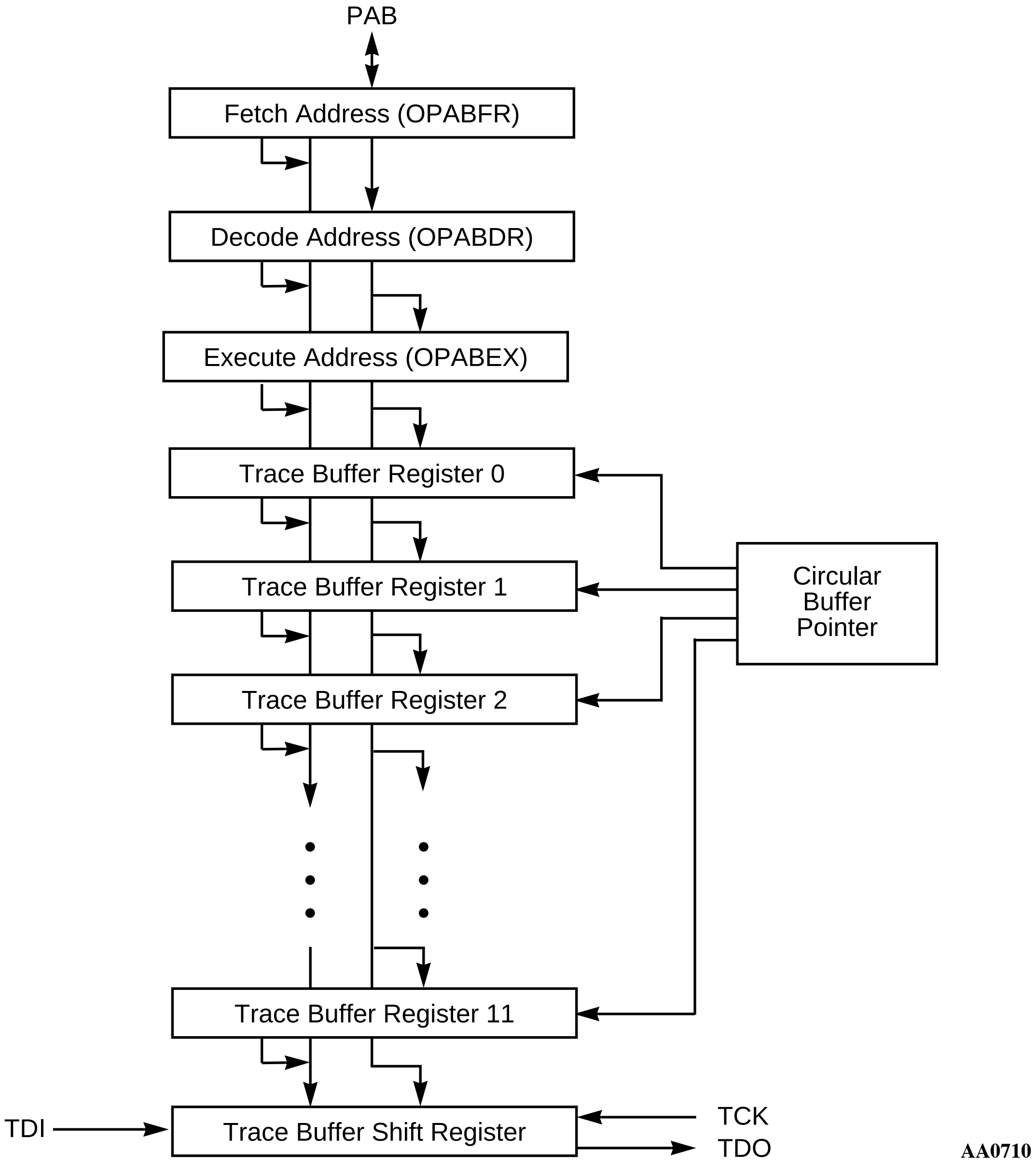


Figure 11-10 OnCE Trace Buffer

11.10 SERIAL PROTOCOL DESCRIPTION

The following protocol promotes efficient communication between the external command controller and the DSP56300 core. Before the chip starts any debugging activity, the external command controller must wait for an acknowledgment on the $\overline{\text{DE}}$

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 for the same information. 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 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.

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 and have the format shown in **Figure 11-4** on page 11-5.

11.11 TARGET SITE DEBUG SYSTEM REQUIREMENTS

A typical debugging 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 used only 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 an interpreter for host computer commands. The controller issues commands based on the host computer inputs from a user interface program that communicates with the user.

11.12 EXAMPLES OF USING THE OnCE

This section presents examples of debugging procedures. All examples assume that the DSP is the only device in the JTAG chain. If there is more than one device in the chain (such as additional DSPs or other devices), the other devices can be forced to execute the JTAG BYPASS instruction so that their effect in the serial stream is one bit per additional device. The events (such as select-DR, select-IR, update-DR, and shift-DR) mean to bring the JTAG TAP into the corresponding state. **Section 12 Joint Test Action Group Port** on page 12-1 contains a detailed description of the JTAG protocol.

11.12.1 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 $\overline{\text{DE}}$ 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 $\overline{\text{DE}}$ line to an interrupt signal in order to sense the acknowledgment.
- 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, ACK denotes the operation performed by the command controller to check whether debug mode has been entered (either by sensing $\overline{\text{DE}}$ or by polling JTAG instruction shift register).

11.12.2 Polling the JTAG Instruction Shift Register

To poll the core status bits in the JTAG instruction shift register, perform the following steps:

1. Select shift-IR. When you pass through capture-IR, it loads the core status bits into the instruction shift register.
2. Shift in ENABLE_ONCE. While the new instruction is being shifted in, 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.

11.12.3 Saving Pipeline Information

Debugging 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 before debugging starts, and of course the state must be restored before a return to normal mode. Here is how to save the current pipeline state; these instructions assume that ENABLE_ONCE has been executed and debug mode has been entered and verified, as explained in **Checking Whether the Chip Has Entered Debug Mode** on page 11-24:

1. Select shift-DR. Shift in “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 “Read PIL”. Pass through update-DR.
4. Select shift-DR. Shift out the 24-bit OPILR register. Pass through update-DR.

There is no need to verify an acknowledgment between steps 1 and 2, nor between 3 and 4, because their completion is guaranteed.

11.12.4 Reading the Trace Buffer

An optional step during debugging is to read the information associated with the trace buffer in order to enable an external program to reconstruct the full trace of the executed program. Here is how to read the trace buffer; these instructions assume that all actions explained in **Saving Pipeline Information** have already been executed:

1. Select shift-DR. Shift in “Read PABFR”. Pass through update-DR.
2. Select shift-DR. Shift out the 16-bit OPABFR. Pass through update-DR.
3. Select shift-DR. Shift in “Read PABDR”. Pass through update-DR.
4. Select shift-DR. Shift out the 16-bit OPABDR. Pass through update-DR.
5. Select shift-DR. Shift in “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 “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 (i.e., 12 times).

Note: You must read the entire FIFO because each read increments the FIFO pointer, thus pointing to the next FIFO location. At the end of this repetition, the FIFO pointer points back to the beginning of the FIFO.

The information 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 changes 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.

11.12.5 Displaying a Specified Register

The DSP56300 must be in debug mode, and all actions described in **Saving Pipeline Information** on page 11-24 must have been executed. The sequence of actions follows:

1. Select shift-DR. Shift in "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 write OPDBR and thus begin execution of the MOVE instruction.
3. Wait for DSP to reenter debug mode (i.e., wait for $\overline{\text{DE}}$ 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 to read.
5. Select shift-DR. Shift out the OGDBR contents. Pass through update-DR. Wait for the next command.

11.12.6 Displaying X Memory Area Starting at Address $xxxx

The DSP56300 must be in Debug mode, and all actions explained in **Saving Pipeline Information** on page 11-24 must have been executed. Since R0 is used as a pointer for the memory, R0 is saved first. The sequence of actions to display X memory, starting at a specific address, is the following:

1. Select shift-DR. Shift in "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 write OPDBR and thus begin execution of the MOVE instruction.
3. Wait for DSP to reenter debug mode (i.e., wait for $\overline{\text{DE}}$ 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 to read.
5. Select shift-DR. Shift out the OGDBR contents. Pass through update-DR. R0 has now been saved.

6. Select shift-DR. Shift in “Write PDB with no-GO no-EX”. Pass through update-DR.
7. Select shift-DR. Shift in the 24-bit opcode: “MOVE #$xxxx,R0”. Pass through update-DR to actually write OPDBR.
8. Select shift-DR. Shift in “Write PDB with GO no-EX”. Pass through update-DR.
9. Select shift-DR. Shift in the second word of the 24-bit opcode: “MOVE #$xxxx,R0” (the $xxxx field). Pass through update-DR to 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 (i.e., wait for $\overline{\text{DE}}$ or poll core status).
11. Select shift-DR. Shift in “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 write OPDBR and thus begin execution of the MOVE instruction.
13. Wait for DSP to reenter debug mode (i.e., wait for $\overline{\text{DE}}$ or poll core status).
14. Select shift-DR, and shift in “READ GDB REGISTER”. Pass through update-DR; this step selects OGDBR as the data register to read.
15. Select shift-DR. Shift out the OGDBR contents. Pass through update-DR. The memory contents of address $xxxx have now been read.
16. Select shift-DR. Shift in “NO SELECT with GO no-EX”. Pass through update-DR; this step re-executes the same “MOVE X:(R0)+, X:OGDB” instruction.
17. Repeat from **Step 14** to finish reading the entire block. When you finish, restore the original value of R0.

11.12.7 Returning from Debug to Normal Mode (Same Program)

To return to normal mode from debug mode, you must have finished examining the current state of the machine, changed some of the registers, and be ready to continue executing a program from the point where it stopped. Therefore, you must restore the pipeline of the machine and enable normal instruction execution, as follows:

1. Select shift-DR. Shift in “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 write the instruction latch.
3. Select shift-DR. Shift in “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 as the internally saved value of the PAB is driven back from the PABFR onto the PAB, the ODEC releases the chip from debug mode, and the normal flow of execution continues.

11.12.8 Returning from Debug to Normal Mode (New Program)

To return to normal mode from debug mode, you must have finished examining the current state of the machine, changed some of the registers, and be ready to start executing a new program (the GOTO command). Therefore, you must force a change-of-flow to the starting address of the new program ($xxxx), as follows:

1. Select shift-DR. Shift in “Write PDB with no-GO no-EX”. Pass through update-DR.
2. Select shift-DR. Shift in the 24-bit $0AF080 (i.e., the opcode of the JUMP instruction). Pass through update-DR to write the Instruction Latch.
3. Select shift-DR. Shift in “Write PDB-GO-TO with GO and EX”. Pass through update-DR.
4. Select shift-DR. Shift in the 16-bit $xxxx. Pass through update-DR to write the PDB. At this time, the ODEC releases the chip from Debug mode, and execution starts from the address $xxxx.

Note: If Debug mode has been entered during a DO LOOP, REP instruction, or other special case (such as interrupt processing, STOP, WAIT, or conditional branching), it is mandatory that the user first resets the DSP56300 and only afterwards proceeds with execution of the new program.

11.13 EXAMPLES OF JTAG AND OnCE INTERACTION

This subsection details interaction between the JTAG port and the OnCE module as well as the TMS sequencing needed to achieve the communication described in **Examples of Using the OnCE** on page 11-23.

The external command controller forces the DSP56300 into debug mode by executing the JTAG instruction DEBUG_REQUEST. To check whether 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 to do so appears in **Table 11-12**.

The sequence to enable the OnCE module appears in **Table 11-13** on page 11-30.

After executing the JTAG instructions DEBUG_REQUEST and ENABLE_ONCE, and after the core status has been polled to verify that the chip is in debug mode, you must save the pipeline. The TMS sequencing for this procedure appears in **Table 11-14** on page 11-30.

Table 11-12 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.
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 11-13 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.
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 11-14 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	

Table 11-14 TMS Sequencing for Reading Pipeline Registers (Continued)

Step	TMS	JTAG Port	OnCE Module	Note
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	
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	

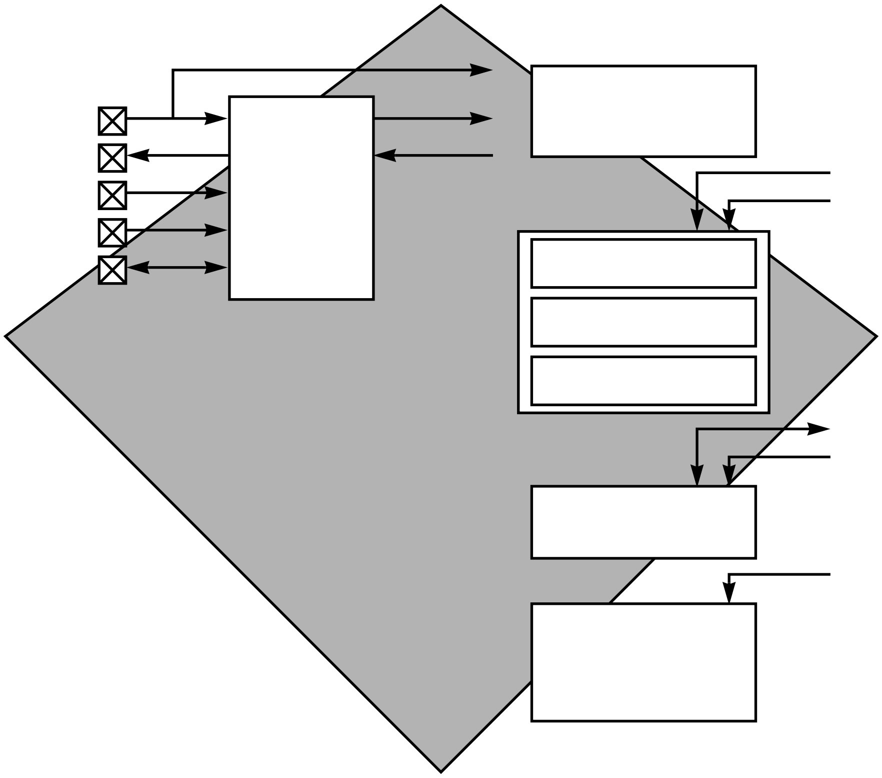
Table 11-14 TMS Sequencing for Reading Pipeline Registers (Continued)

Step	TMS	JTAG Port	OnCE Module	Note
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, and afterwards it can proceed with the debug activities as requested by the user.

SECTION 12

JOINT TEST ACTION GROUP PORT



12.1 INTRODUCTION TO THE JTAG PORT

The DSP56300 core provides a dedicated user-accessible TAP based on the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*. Problems associated with testing high density circuit boards led to the development of this 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. (**Figure 12-1** on page 12-4 shows a block diagram of the TAP port.) A boundary scan register (BSR) links all device signals into a single shift register. The test logic, implemented by static logic design, is independent of the device system logic. The DSP56300 core implementation provides the following capabilities:

- Performing boundary scan operations to test circuit-board electrical continuity (EXTEST).
- Bypassing the DSP56300 core for a given circuit-board test by effectively reducing the BSR to a single cell (BYPASS).
- Sampling the DSP56300 core-based device system signals during operation and transparently shifting out the result in the BSR.
- Preloading values to output signals prior to invoking the EXTEST instruction (SAMPLE/PRELOAD).
- Disabling the output drive to signals during circuit-board testing (HI-Z).
- Providing a means of accessing the OnCE controller and circuits to control a target system (ENABLE_ONCE).
- Providing a means of entering debug mode (DEBUG_REQUEST).
- Querying identification information (manufacturer, part number and version) from a DSP56300 core-based device (IDCODE).
- Forcing 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 for use with the supporting IEEE 1149.1 standard. 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 standard.

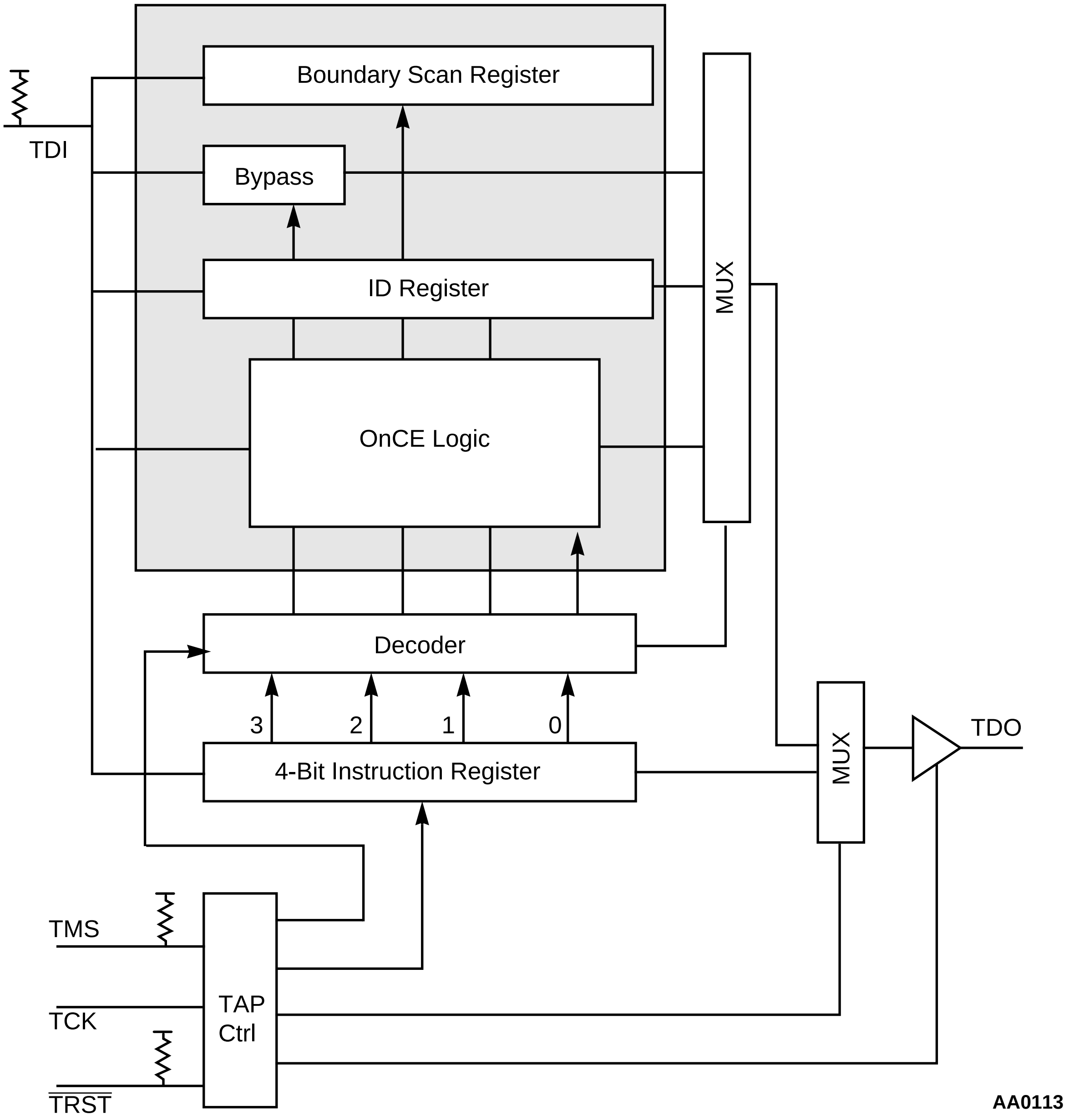


Figure 12-1 TAP Block Diagram

12.2 JTAG SIGNALS

As described in the IEEE 1149.1 standard, 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 DSP56307, the debug event ($\overline{\text{DE}}$) signal is provided for use by the OnCE module; it is documented in **Section 11.3 Debug Event** on page 11-3. The other signals are documented in the following paragraphs.

12.2.1 Test Clock (TCK)

The TCK input signal is used to synchronize the test logic.

12.2.2 Test Mode Select (TMS)

The TMS input signal is used to sequence the state machine of the test controller. The TMS is sampled on the rising edge of TCK, and it has an internal pull-up resistor.

12.2.3 Test Data Input (TDI)

Serial test instruction and data are received through the TDI signal. TDI is sampled on the rising edge of TCK, and it has an internal pull-up resistor.

12.2.4 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.

12.2.5 Test Reset ($\overline{\text{TRST}}$)

The $\overline{\text{TRST}}$ input signal is used asynchronously to initialize the test controller. The $\overline{\text{TRST}}$ signal has an internal pull-up resistor.

12.3 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 JTAG logic. The state machine appears in **Figure 12-2**. The TAP controller responds to changes at the TMS and TCK signals. Transitions from one state to another occur on the rising edge of the TCK signal. The value shown adjacent to each state transition represents the value of the TMS signal sampled on the rising edge of the TCK signal. For a description of the TAP controller states, refer to the IEEE 1149.1 standard.

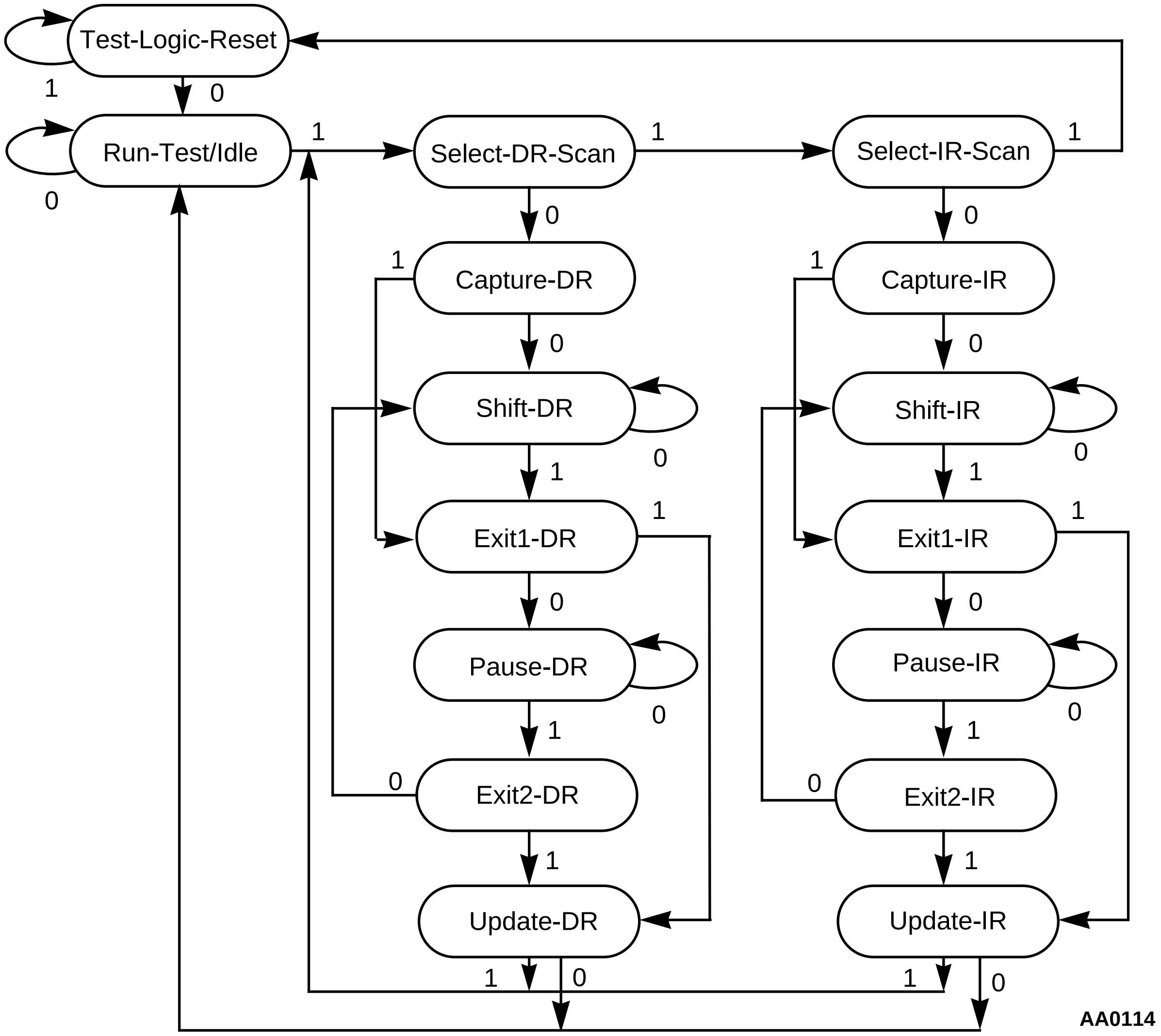


Figure 12-2 TAP Controller State Machine

12.3.1 Boundary Scan Register

The BSR in the DSP56307 JTAG implementation contains bits for all device signals, clock signals, and associated control signals. All DSP56307 bidirectional signals have a single register bit in the BSR for signal data and are controlled by an associated control bit in the BSR. The DSP56307 BSR bit definitions are listed in **Table 12-2** on page 12-13.

12.3.2 Instruction Register

The DSP56307 JTAG implementation includes the three mandatory public instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS); it also supports the optional CLAMP instruction defined by IEEE 1149.1. Using the HI-Z public instruction, it possible to disable 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 12-3** shows the JTAG instruction register.

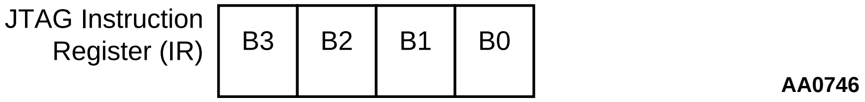


Figure 12-3 JTAG Instruction Register

The four bits are used to decode the eight unique instructions shown in **Table 12-1**. All other encodings are reserved for future enhancements and are decoded as BYPASS.

Table 12-1 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, equivalent to the IDCODE instruction.

In 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 11.4 OnCE Controller** on page 11-4 for a description of the status bits.

12.3.2.1 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 it performs 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 about the function and use of the EXTEST instruction, refer to the IEEE 1149.1 standard.

12.3.2.2 SAMPLE/PRELOAD (B[3:0] = 0001)

The SAMPLE/PRELOAD instruction provides two separate functions.

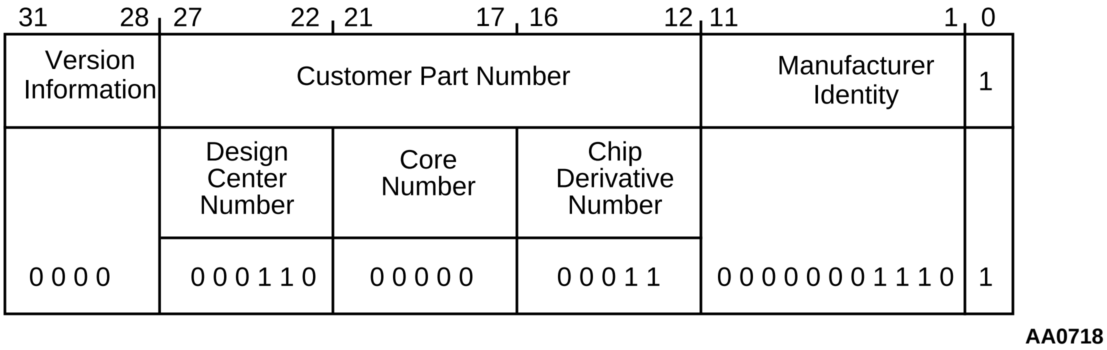
First, it can make a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. You can then scrutinize the data by shifting it transparently through the BSR.

Note: Since there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), you 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 insures that known data appear on the outputs when the EXTEST instruction is entered.

12.3.2.3 IDCODE (B[3:0] = 0010)

The IDCODE instruction selects the ID register. It is provided as a public instruction to allow the manufacturer, part number, and version of a component to be determined through the TAP. **Figure 12-4** shows the ID register configuration.



31 28	27 22	21 17	16 12	11 1	0
Version Information	Customer Part Number			Manufacturer Identity	1
	Design Center Number	Core Number	Chip Derivative Number		
0 0 0 0	0 0 0 1 1 0	0 0 0 0 0	0 0 0 1 1	0 0 0 0 0 0 0 1 1 1 0	1

Figure 12-4 JTAG ID Register

One application of the ID register is to distinguish the manufacturer(s) of components on a multiply sourced board. As more components emerge which conform to the IEEE 1149.1 standard, a system diagnostic controller unit can thus interrogate a board design blindly 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: the 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 the DSP56300 core number is 00001.

Once the IDCODE instruction is decoded, it selects the ID register (a 32-bit data register). 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. Therefore, examination of the first bit of data shifted out of a component during a test data scan sequence and 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.

12.3.2.4 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. While testing integrated circuits on printed circuit boards, 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 since it selects the boundary scan register, 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 boundary scan register of the appropriate integrated circuits when their bypass registers are selected, 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 reset. The CLAMP instruction also asserts internal reset for the DSP56300 core system logic to force a predictable internal state while external boundary scan operations are performed.

12.3.2.5 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

12.3.2.6 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 the user 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 currently being executed. All communication with the OnCE controller is done through the Select-DR-Scan path of the JTAG TAP controller. See **Section 11 On-Chip Emulation Module** for more information about OnCE.

12.3.2.7 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 the user 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.

12.3.2.8 BYPASS (B[3:0] = 1111)

The BYPASS instruction selects the single-bit BYPASS register, as shown in **Figure 12-5**. This 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 state 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.

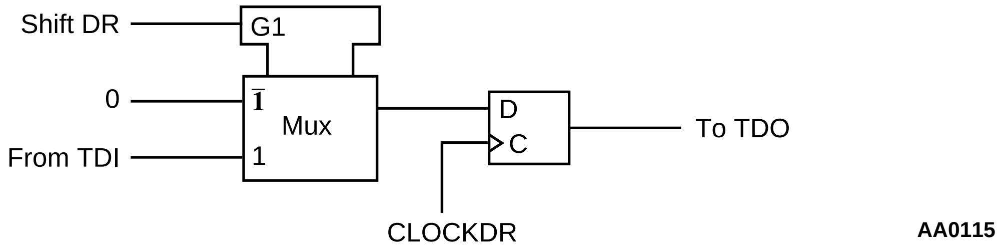


Figure 12-5 BYPASS Register

12.4 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. The user 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 a power-up or regular hardware $\overline{\text{RESET}}$ signal while EXTAL is 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 pull-up resistor and should not be left unconnected. Secondly, the JTAG test logic must be 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; that mode is invoked by the STOP instruction. The JTAG interface interacts with low-power stop mode like this:

- The TAP controller must be in the test-logic-reset state to enter or to remain in low-power stop mode. To leave the TAP controller test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
- 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.
- The TMS and TDI signals include on-chip pull-up resistors. In low-power stop mode, these two signals should remain either unconnected or connected to V_{CC} to achieve minimal power consumption.

All DSP56307 core clocks are disabled during stop mode; consequently, the JTAG interface provides a means of polling the device status (sampled in the capture-IR state).

12.5 DSP56307 BOUNDARY SCAN REGISTER

Table 12-2 provides a listing of the contents of the BSR for the DSP56307.

Table 12-2 DSP56306 BSR Bit Definitions

Bit #	Pin Name	Pin Type	BSR Cell Type	Bit #	Pin Name	Pin Type	BSR Cell Type
	IRQA	Input	Data		RES	Input	Data
	IRQB	Input	Data		HAD0	—	Control
	IRQC	Input	Data		HAD0	Input/Output	Data
	IRQD	Input	Data		HAD1	—	Control
	D23	Input/Output	Data		HAD1	Input/Output	Data
	D22	Input/Output	Data		HAD2	—	Control
	D21	Input/Output	Data		HAD2	Input/Output	Data
	D20	Input/Output	Data		HAD3	—	Control
	D19	Input/Output	Data		HAD3	Input/Output	Data
	D18	Input/Output	Data		HAD4	—	Control
	D17	Input/Output	Data		HAD4	Input/Output	Data
	D16	Input/Output	Data		HAD5	—	Control
	D15	Input/Output	Data		HAD5	Input/Output	Data
	D[23:13]	—	Control		HAD6	—	Control
	D14	Input/Output	Data		HAD6	Input/Output	Data
	D13	Input/Output	Data		HAD7	—	Control
	D12	Input/Output	Data		HAD7	Input/Output	Data
	D11	Input/Output	Data		HAS/A0	—	Control
	D10	Input/Output	Data		HAS/A0	Input/Output	Data
	D9	Input/Output	Data		HA8/A1	—	Control
	D8	Input/Output	Data		HA8/A1	Input/Output	Data
	D7	Input/Output	Data		HA9/A2	—	Control
	D6	Input/Output	Data		HA9/A2	Input/Output	Data
	D5	Input/Output	Data		HCS/A10	—	Control

Table 12-2 DSP56306 BSR Bit Definitions (Continued)

Bit #	Pin Name	Pin Type	BSR Cell Type	Bit #	Pin Name	Pin Type	BSR Cell Type
	D4	Input/Output	Data		HCS/A10	Input/Output	Data
	D3	Input/Output	Data		TIO0	—	Control
	D[12:0]	—	Control		TIO0	Input/Output	Data
	D2	Input/Output	Data		TIO1	—	Control
	D1	Input/Output	Data		TIO1	Input/Output	Data
	D0	Input/Output	Data		TIO2	—	Control
	A17	Tri-State	Data		TIO2	Input/Output	Data
	A16	Tri-State	Data		HREQ/TRQ	—	Control
	A15	Tri-State	Data		HREQ/TRQ	Input/Output	Data
	A[17:9]	—	Control		HACK/RRQ	—	Control
	A14	Tri-State	Data		HACK/RRQ	Input/Output	Data
	A13	Tri-State	Data		HRW/RD	—	Control
	A12	Tri-State	Data		HRW/RD	Input/Output	Data
	A11	Tri-State	Data		HDS/WR	—	Control
	A10	Tri-State	Data		HDS/WR	Input/Output	Data
	A9	Tri-State	Data		SCK0	—	Control
	A8	Tri-State	Data		SCK0	Input/Output	Data
	A7	Tri-State	Data		SCK1	—	Control
	A6	Tri-State	Data		SCK1	Input/Output	Data
	A[8:0]	—	Control		SCLK	—	Control
	A5	Tri-State	Data		SCLK	Input/Output	Data
	A4	Tri-State	Data		TXD	—	Control
	A3	Tri-State	Data		TXD	Input/Output	Data
	A2	Tri-State	Data		RXD	—	Control
	A1	Tri-State	Data		RXD	Input/Output	Data
	A0	Tri-State	Data		SC00	—	Control
	BG	Input	Data		SC00	Input/Output	Data

Table 12-2 DSP56306 BSR Bit Definitions (Continued)

Bit #	Pin Name	Pin Type	BSR Cell Type	Bit #	Pin Name	Pin Type	BSR Cell Type
	AA0	Tri-State	Data		SC10	—	Control
	AA1	Tri-State	Data		SC10	Input/Output	Data
	RD	Tri-State	Data		STD0	—	Control
	WR	Tri-State	Data		STD0	Input/Output	Data
	AA0	—	Control		SRD0	—	Control
	AA1	—	Control		SRD0	Input/Output	Data
	BB	—	Control		PINIT	Input	Data
	BB	Input/Output	Data		DE	—	Control
	BR	Output	Data		DE	Input/Output	Data
	TA	Input	Data		SC01	—	Control
	BCLK	Tri-State	Data		SC01	Input/Output	Data
	BCLK	Tri-State	Data		SC02	—	Control
	CLKOUT	Output	Data		SC02	Input/Output	Data
	$\overline{RD}$,$\overline{WR}$,BCLK, $\overline{BCLK}$,$\overline{BS}$	—	Control		STD1	—	Control
	CAS	—	Control		STD1	Input/Output	Data
	AA2	—	Control		SRD1	—	Control
	AA3	—	Control		SRD1	Input/Output	Data
	EXTAL	Input	Data		SC11	—	Control
	CAS	Tri-State	Data		SC11	Input/Output	Data
	AA2	Tri-State	Data		SC12	—	Control
	AA3	Tri-State	Data		SC12	Input/Output	Data

APPENDIX A

BOOTSTRAP PROGRAMS

```
; BOOTSTRAP CODE FOR DSP56307- (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 DSP56307 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.
```

```
; BOOTSTRAP CODE FOR DSP56307 - (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 DSP56307 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 DSP56302A
; 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       ;; 7k 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_OGDB  EQU     $FFFFFC         ;; OnCE GDB Register
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 Port 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 #$0,r5            ; clear a and init r5=0
        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.
;========================================================================
```

APPENDIX B

EQUATES

```
;*****************************************************************************
;
;     EQUATES for DSP56307 I/O registers and ports
;
;     Last update: June 11 1998
;
;*****************************************************************************

           page       132,55,0,0,0
           opt        mex

ioequ   ident   1,0

;------------------------------------------------------------------------
```

B.1 I/O EQUATES

```
;*************************************************************************
;
;     EQUATES for 56307 I/O registers and ports
;     Reference: Specifications Revision 3.00
;                     + 56307 spec rev 1.2
;*************************************************************************
;------------------------------------------------------------------------
;
;       EQUATES for I/O Port Programming
;
;-------------------------------------------------------------------------

;       Register Addresses

M_DATH   EQU     $FFFFCF         ; Host port GPIO data Register
M_DIRH   EQU     $FFFFCE         ; 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 HI08 EQUATES

```
;-------------------------------------------------------------------------
;
;       EQUATES for Host Interface
;
;-------------------------------------------------------------------------

;       Register Addresses

M_DTXS   EQU     $FFFFCD         ; DSP SLAVE TRANSMIT DATA FIFO (DTXS)
M_DTXM   EQU     $FFFFCC         ; DSP MASTER TRANSMIT DATA FIFO (DTXM)
M_DRXR   EQU     $FFFFCB         ; DSP RECEIVE DATA FIFO (DRXR)
M_DPSR   EQU     $FFFFCA         ; DSP PCI STATUS REGISTER (DPSR)
M_DSR    EQU     $FFFFC9         ; DSP STATUS REGISTER (DSR)
M_DPAR   EQU     $FFFFC8         ; DSP PCI ADDRESS REGISTER (DPAR)
M_DPMC   EQU     $FFFFC7         ; DSP PCI MASTER CONTROL REGSTER (DPMC)
M_DPCR   EQU     $FFFFC6         ; DSP PCI CONTROL REGISTER (DPCR)
M_DCTR   EQU     $FFFFC5         ; DSP CONTROL REGISTER (DCTR)
```

```
;       Host Control Register Bit Flags

M_HCIE   EQU     0               ; Host Command Interrupt Enable
M_STIE   EQU     1               ; Slave Transmit Interrupt Enable
M_SRIE   EQU     2               ; Slave Receive Interrupt Enable
M_HF35   EQU     $38             ; Host Flags 5-3 Mask
M_HF3    EQU     3               ; Host Flag 3
M_HF4    EQU     4               ; Host Flag 4
M_HF5    EQU     5               ; Host Flag 5
M_HINT   EQU     6               ; Host Interrupt A
M_HDSM   EQU     13              ; Host Data Strobe Mode
M_HRWP   EQU     14              ; Host RD/WR Polarity
M_HTAP   EQU     15              ; Host Transfer Acknowledge Polarity
M_HDRP   EQU     16              ; Host Dma Request Polarity
M_HRSP   EQU     17              ; Host Reset Polarity
M_HIRP   EQU     18              ; Host Interrupt  Request Polarity
M_HIRC   EQU     19              ; Host Interupt Request Control
M_HM0    EQU     20              ; Host Interface Mode
M_HM1    EQU     21              ; Host Interface Mode
M_HM2    EQU     22              ; Host Interface Mode
M_HM     EQU     $700000         ; Host Interface Mode Mask

;       Host PCI Control Register Bit Flags

M_PMTIE  EQU     1               ; PCI Master Transmit  Interrupt Enable
M_PMRIE  EQU     2               ; PCI Master Receive   Interrupt Enable
M_PMAIE  EQU     4               ; PCI Master Address   Interrupt Enable
M_PPEIE  EQU     5               ; PCI Parity Error     Interrupt Enable
M_PTAIE  EQU     7               ; PCI Transacton Abort Interrupt Enable
M_PTTIE  EQU     9               ; PCI Transaction Term Interrupt Enable
M_PTCIE  EQU     12              ; PCI Transfer Complet Interrupt Enable
M_CLRT   EQU     14              ; Clear Transmitter
M_MTT    EQU     15              ; Master Transfer Terminate
M_SERF   EQU     16              ; HSERR~ Force
M_MACE   EQU     18              ; Master Access Counter Enable
M_MWSD   EQU     19              ; Master Wait States Disable
M_RBLE   EQU     20              ; Receive Buffer Lock Enable
M_IAE    EQU     21              ; Insert Address Enable

;       Host PCI Master Control Register Bit Flags

M_ARH    EQU     $00ffff         ; DSP PCI Transaction Address (High)
M_BL     EQU     $3f0000         ; PCI Data Burst Length
M_FC     EQU     $c00000         ; Data Transfer Format Control

;       Host PCI Address Register Bit Flags

M_ARL    EQU     $00ffff         ; DSP PCI Transaction Address (Low)
M_C      EQU     $0f0000         ; PCI Bus Command
M_BE     EQU     $f00000         ; PCI Byte Enables

;       DSP Status Register Bit Flags
```

```
M_HCP    EQU     0               ; Host Command pending
M_STRQ   EQU     1               ; Slave Transmit Data Request
M_SRRQ   EQU     2               ; Slave Receive Data Request
M_HF02   EQU     $38             ; Host Flag 0-2 Mask
M_HF0    EQU     3               ; Host Flag 0
M_HF1    EQU     4               ; Host Flag 1
M_HF2    EQU     5               ; Host Flag 2

;       DSP PCI Status Register Bit Flags

M_MWS    EQU    0               ; PCI Master Wait States
M_MTRQ   EQU    1               ; PCI Master Transmit Data Request
M_MRRQ   EQU    2               ; PCI Master Receive Data Request
M_MARQ   EQU    4               ; PCI Master Address Request
M_APER   EQU    5               ; PCI Address Parity Error
M_DPER   EQU    6               ; PCI Data Parity Error
M_MAB    EQU    7               ; PCI Master Abort
M_TAB    EQU    8               ; PCI Target Abort
M_TDIS   EQU    9               ; PCI Target Disconnect
M_TRTY   EQU    10              ; PCI Target Retry
M_TO     EQU    11              ; PCI Time Out Termination
M_RDC    EQU    $3F0000         ; Remaining Data Count Mask (RDC5-RDC0)
M_RDC0   EQU    16              ; Remaining Data Count  0
M_RDC1   EQU    17              ; Remaining Data Count  1
M_RDC2   EQU    18              ; Remaining Data Count  2
M_RDC3   EQU    19              ; Remaining Data Count  3
M_RDC4   EQU    20              ; Remaining Data Count  4
M_RDC5   EQU    21              ; Remaining Data Count  5
M_HACT   EQU    23              ; Hi32 Active
```

B.3 SCI EQUATES

```
;----------------------------------------------------------------------
;
;       EQUATES for Serial Communications Interface (SCI)
;
;----------------------------------------------------------------------

;       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

```
;------------------------------------------------------------------------
;
;       EQUATES for Synchronous Serial Interface (SSI)
;
;------------------------------------------------------------------------

;
;       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 Slct Mask (PM0-PM7)
M_PSR    EQU    11             ; Prescaler Range
M_DC     EQU    $1F000         ; Frm Rate Divider Contl 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 Intrupt Enable
M_SRLIE  EQU     21              ; SSI Receive Last Slot Interupt 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 Xmit Slot Bits Mask A (TS0-TS15)

;       SSI Transmit Slot Mask Register B

M_SSTSB  EQU     $FFFF           ; SSI Xmit Slot Bits Mask B (TS16-TS31)

;       SSI Receive Slot Mask Register A

M_SSRSA  EQU     $FFFF           ; SSI Rcv Slot Bits Mask A (RS0-RS15)

;       SSI Receive Slot Mask Register B

M_SSRSB  EQU     $FFFF           ; SSI Rcv Slot Bits Mask B (RS16-RS31)
```

B.5 EXCEPTION PROCESSING EQUATES

```
;----------------------------------------------------------------------
;
;       EQUATES for Exception Processing
;
;----------------------------------------------------------------------


;       Register Addresses

M_IPRC   EQU     $FFFFFF         ; Interrupt Priority Register Core
M_IPRP   EQU     $FFFFFE         ; Interupt Priority Register Peripheral

;       Interrupt Priority Register Core (IPRC)

M_IAL    EQU     $7              ; IRQA Mode Mask
M_IAL0   EQU     0               ; IRQA Mode Interupt 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 Interupt 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 Interupt 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 Interupt 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)
M_KPL    EQU     $C00           ; FKOP Interrupt Priority Level Mask
M_KPL0   EQU     10             ; FKOP Interrupt Priority Level (low)
M_KPL1   EQU     11             ; FKOP Interrupt Priority Level (high)
M_VPL    EQU     $3000          ; VCOP Interrupt Priority Level Mask
M_VPL0   EQU     12             ; VCOP Interrupt Priority Level (low)
M_VPL1   EQU     13             ; VCOP Interrupt Priority Level (high)
M_CPL    EQU     $C000          ; CCOP Interrupt Priority Level Mask
M_CPL0   EQU     14             ; CCOP Interrupt Priority Level (low)
M_CPL1   EQU     15             ; CCOP Interrupt Priority Level (high)
```

B.6 TIMER MODULE EQUATES

```
;----------------------------------------------------------------------
;
;       EQUATES for TIMER
;
;----------------------------------------------------------------------

;       Register Addresses Of TIMER0

M_TCSR0  EQU     $FFFF8F        ; TIMER0 Control/Status Register
M_TLR0   EQU     $FFFF8E        ; TIMER0 Load Reg
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 Reg
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 Reg
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 Prescalar 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 (TC0-TC3)
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 DMA EQUATES

```
;----------------------------------------------------------------------
;
;       EQUATES for Direct Memory Access (DMA)
;
;----------------------------------------------------------------------

;       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 (DTD0-DTD5)
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 (DCH0-DCH2)
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 PLL EQUATES

```
;------------------------------------------------------------------------
;
;       EQUATES for Phase Locked Loop (PLL)
;
;-----------------------------------------------------------------------

;       Register Addresses Of PLL

M_PCTL   EQU     $FFFFFD         ; PLL Control Register

;       PLL Control Register

M_MF     EQU     $FFF            ; Multiplication Factor Bits Mask (MF0-MF11)
M_DF     EQU     $7000           ; Division Factor Bits 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 Bits Mask (PD0-PD3)
```

B.9 BIU EQUATES

```
;-----------------------------------------------------------------------
;
;       EQUATES for BIU

;-----------------------------------------------------------------------

;       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 Bits Mask (BCW0-BCW1)
M_BRW    EQU     $C              ; Out Of Page Wait States Bits Mask (BRW0-BRW1)
M_BPS    EQU     $300            ; DRAM Page Size Bits 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 (BAT0-BAT1)
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            ; No of Addr Bits to Compare Mask (BNC0-BNC3)
M_BAC    EQU     $FFF000         ; Address to Compare Bits Mask (BAC0-BAC11)

;       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                ; Interupt Mask Bit 0
M_I1    EQU     9                ; Interupt 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_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_XYS   EQU     16               ; Stack Extension space select bit in OMR.
M_EUN   EQU     17               ; Extensed 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 56307 interrupts
;     Reference: Specifications Revision 3.00
;                + 56307 spec 1.2
;     Last update: July   1 1997
;************************************************************************
;------------------------------------------------------------------------
;       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_HPTT   EQU     I_VEC+$60   ; Host PCI Transaction Termination
I_HPTA   EQU     I_VEC+$62   ; Host PCI Transaction Abort
I_HPPE   EQU     I_VEC+$64   ; Host PCI Parity Error
I_HPTC   EQU     I_VEC+$66   ; Host PCI Transfer Complete
I_HPMR   EQU     I_VEC+$68   ; Host PCI Master Receive
I_HSR    EQU     I_VEC+$6A   ; Host Slave Receive
```

```
I_HPMT   EQU     I_VEC+$6C   ; Host PCI Master Transmit
I_HST    EQU     I_VEC+$6E   ; Host Slave Transmit
I_HPMA   EQU     I_VEC+$70   ; Host PCI Master Address
I_HCNMI  EQU     I_VEC+$72   ; Host Command/Host NMI (Default)

;------------------------------------------------------------------------
; EFCOP Filter Interrupts
;------------------------------------------------------------------------

I_FDIIE  EQU     I_VEC+$68    ; EFilter input buffer empty
I_FDOIE  EQU     I_VEC+$6A    ; EFilter output buffer full

;------------------------------------------------------------------------
; INTERRUPT ENDING ADDRESS
;------------------------------------------------------------------------

I_INTEND EQU     I_VEC+$FF    ; last address of interrupt vector space
```

APPENDIX C

DSP56307 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 DSP56307 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: Wed May 20 09:52:21 1998
--
-- Revision History:
--
-- 1) Date : Tue Oct 14 20:31:41 1997
--    Changes : Created for dsp56307 rev0, PBGA, mask 0H83G
--
-- 2) Date : Thu Nov 20 21:26:28 1997
--    Changes : Correction in the COMPONENT_CONFORMANCE statement
--
-- 3) Date : Mon Dec 15 15:28:06 1997
--    Changes : Indentifiers ending with an underscore (_)
--              were replaced by indentifiers ending with (_N)
--
-- 4) Date : Wed May 20 09:52:21 1998
--    Changes : Fix in definition of DE_N, it is Pull1 when disabled
--    Updated by Roman Sajman
--

entity DSP56307 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 DSP56307 : entity is “STD_1149_1_1993”;

        attribute PIN_MAP of DSP56307 : 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 DSP56307 : entity is 4;

attribute INSTRUCTION_OPCODE of DSP56307 : entity is
   “EXTEST              (0000),” &
   “SAMPLE              (0001),” &
   “IDCODE              (0010),” &
   “CLAMP               (0101),” &
   “HIGHZ               (0100),” &
   “ENABLE_ONCE         (0110),” &
   “DEBUG_REQUEST       (0111),” &
   “BYPASS              (1111)”;

attribute INSTRUCTION_CAPTURE of DSP56307 : entity is “0001”;
attribute IDCODE_REGISTER   of DSP56307 : entity is
   “0000”          & -- version
   “000110”        & -- manufacturer’s use
   “0000000111”    & -- sequence number
   “00000001110”   & -- manufacturer identity
   “1”;              -- 1149.1 requirement

attribute REGISTER_ACCESS of DSP56307 : entity is
   “ONCE[8]   (ENABLE_ONCE,DEBUG_REQUEST)” ;

attribute BOUNDARY_LENGTH of DSP56307 : entity is 144;

attribute BOUNDARY_REGISTER of DSP56307 : 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),” &
```

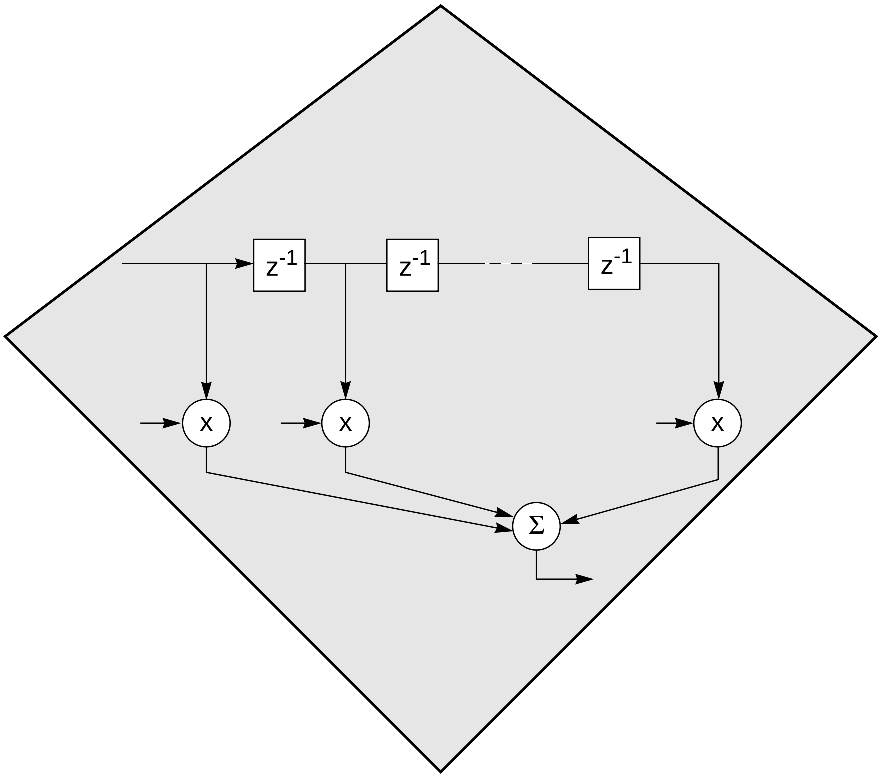
```
     “53   (BC_1, RD_N,      output3,       X,     64,   1,   Z),” &
     “54   (BC_1, WR_N,      output3,       X,     64,   1,   Z),” &
     “55   (BC_1, *,         control,       1),” &
     “56   (BC_1, *,         control,       1),” &
     “57   (BC_1, *,         control,       1),” &
     “58   (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 DSP56307;
```

APPENDIX D

EFCOP PROGRAMMING



D.1 EFCOP PROGRAMMING

The DSP56307 Enhanced Filter Coprocessor (EFCOP) supports both Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters.[1] This appendix discusses the different ways data can be transferred in and out of the EFCOP and presents some programming examples in which such transfers are performed for FIR and IIR filters.

EFCOP operation is determined by the control bits in the EFCOP Control/Status Register (FCSR), described in **Section 10.4.5**. Further filtering operations are enabled via the appropriate bits in the FACR and FDCH registers. After the FCSR is configured to the mode of choice, enable the EFCOP by setting Bit 0 of the FCSR. To ensure proper EFCOP operation, most FCSR bits must not be changed while the EFCOP is enabled. **Table D-1** summarizes the EFCOP operating modes.

Table D-1 EFCOP Operating Modes

Mode Description[3]	FCSR Bits					
	6 FMLC	5–4 FOM	3 FUPD[2]	2 FADP[2]	1 FLT	0 FEN
EFCOP Disabled[1]	x	x	x	x	x	0
FIR, Real, single channel	0	00	0	0	0	1
FIR, Real, adaptive, single channel	0	00	0	1	0	1
FIR, Real, coeff. update, single channel	0	00	1	0	0	1
FIR, Real, adaptive + coeff. update, single channel	0	00	1	1	0	1
FIR, Real, multichannel	1	00	0	0	0	1
FIR, Real, adaptive, multichannel	1	00	0	1	0	1
FIR, Real, coeff. update, multichannel	1	00	1	0	0	1
FIR, Real, adaptive + coeff. update, multichannel	1	00	1	1	0	1
FIR, Full Complex, single channel	0	01	0	0	0	1
FIR, Complex Alternating, single channel	0	10	0	0	0	1
FIR, Magnitude, single channel	0	11	0	0	0	1
IIR, Real, single channel	0	00	0	0	1	1
IIR, Real, multichannel	1	00	0	0	1	1

[1] For details on FIR and IIR filters, refer to the Motorola application note entitled *Implementing IIR/FIR Filters with Motorola's DSP56000/DSP56001* (APR7/D).

Table D-1 EFCOP Operating Modes (Continued)

Mode Description[3]	FCSR Bits					
	6 FMLC	5–4 FOM	3 FUPD[2]	2 FADP[2]	1 FLT	0 FEN

Note:
1. An x indicates that the specified value can be 1 or 0.
2. If the user sets the FUPD bit, the EFCOP updates the coefficients and clears the FUPD bit. The adaptive mode (i.e., FADP = 1) sets the FUPD bit, which causes the EFCOP to update the coefficients and then automatically clear the FUPD bit. Therefore, the value assigned to the FUPD bit in this table refers only to its initial setting and not its dynamic state during operation.
3. All bit combinations not defined by this table are reserved for future development.

D.2 OVERVIEW

This section describes how to transfer data to and from the EFCOP using an FIR filter configuration. Here, we provide background information to help you understand the examples in **D.3 Examples** on page D-6.

The following notations are used throughout the examples:

- D(n): Data sample at time n
- H(n): Filter coefficient at time n
- F(n): Output result at time n
- #filter_count: Number of coefficient values in the coefficient memory bank FCM; it is equal to the initial value written to the FCNT register plus 1.
- Compute: Perform all calculations to determine one filter output F(n) for a specific set of input data samples

D.2.1 Transferring Data to and From the EFCOP

To transfer data to/from the EFCOP input/output registers, the Filter Data Input Register (FDIR) and the Filter Data Output Register (FDOR) are triggered by three different methods:

- Direct Memory Access (DMA)
- Interrupts
- Polling

Two FCSR bits (FDIBE and FDOBF) indicate the status of the FDIR and the FDOR, respectively. All three data transfer methods use these two FCSR bits as their control mechanism. If FDIBE is set, the input buffer is empty; if FDOBF is set, the output buffer is full. Because these bits come into full operation only when the EFCOP is enabled (FCSR:FEN is set), the polling, DMA, or interrupt methods can be initialized either before or after the EFCOP is enabled. No service request is issued until the EFCOP is enabled, since FDIBE and FDOBF are cleared while the EFCOP is in the Individual Reset state.

The most straightforward EFCOP transfer implementation is to use the core processor to poll the status flags, monitoring for input/output service requests. The disadvantage of this approach is that it demands large amounts of (if not all of) the core's processing time. The interrupt and DMA methods are more efficient in their use of the core processor. Interrupts intervene on the core processor infrequently in order to service input/output data.

DMA can operate concurrently with the processor core and demands only minimal core resource for setup. DMA transfers are recommended when the EFCOP is in FIR/IIR filtering mode since the core can operate independently of the EFCOP while DMA transfers data to the FDIR and from the FDOR. Since the EFCOP's input buffer (FDIR) is four words deep, the DMA can input in blocks of up to four words. A combination of DMA transfer for input and an interrupt request for processing the output is recommended for adaptive FIR mode. This combination gives the following benefits:

- Input data transfers to the FDIR can occur independently of the core.
- There is minimal intervention of the core while the weight update multiplier is updated.

If the initialization mode is enabled (i.e., if the FCSR:FPRC bit is cleared), the core can initialize the coefficient bank while the DMA controller concurrently transfers initial data values to the data bank. The EFCOP state machine starts computation as soon as #filter_count data samples are input.

If no initialization mode is used (the FPRC bit is set), the EFCOP machine starts computation as soon as the first data sample becomes available in the input buffer. The filter coefficient bank must therefore be initialized prior an input data transfer starts. The DMA input channel can continue transferring data whenever the input FIFO becomes empty, while the EFCOP state machine takes data words from the FIFO whenever required.

D.3 EXAMPLES

The following sections provide examples of how to use the EFCOP in different modes.

D.3.1 Real FIR Filter: Mode 0

In this example, an N tap FIR filter is represented as follows:

$$F(n) = \sum_{i=0}^{N-1} H(i) \cdot D(n-i)$$

The filter is implemented with three different data transfers using the EFCOP in data initialization mode:

1. DMA input/DMA output
2. DMA input/Polling output
3. DMA input/Interrupt output

This transfer combination is only one of many possible combinations.[2]

D.3.1.1 DMA Input/DMA Output

A 20-tap FIR filter using a 28-input sample signal is implemented in the following stages:

1. Setup:
   - Set the filter count register (FCNT) to the length of the filter coefficients –1 (i.e N-1).
   - Set the Data and Coefficient Base Address pointers (FDBA, FCBA).
   - Set the operation mode (FCSR[5:4] = FOM[00]).
   - Set Initialization mode (FCSR[7] = FPRC = 0).
   - Set DMA registers:
     - DMA input: A two-dimensional (2D) DMA transfer fills up the FDM bank via channel 0. DCR0 (DMA Control Register 0) is initialized as follows:
       - DIE = 0 Disables end-of-transfer interrupt.
       - DTM = 2 Chooses line transfer triggered by request; DE auto clear on end of transfer.
       - DPR = 2 Priority 2.

[2] For information on DMA transfers, refer to the Motorola application note entitled *Using the DSP56300 Direct Memory Access Controller* (APR23/D).

- DCON = 0 Disables continuous mode.
- DRS = $15 Chooses DMA to trigger on EFCOP input buffer empty.
- D3D = 0 Chooses non-3D mode.
- DAM = $20 Sets the following DMA Address Mode: source address - 2D, counter mode B, offset DOR0; destination address - no update, no offset.
- DDS = 1 Destination in Y memory space (because EFCOP is in Y memory).
- DSS = 0 Source in X memory space.
- DOR0=1 Offset register
- DCO0= $006003. Gives transfer of 7 * 4 = 28 items (input sequence length).
- DSR0 = address of source data
- DDR0 = $FFFFB0

- DMA output: Channel 1 is used, with a configuration similar to that of the DMA input channel, except for a 1D transfer. DCR1 (DMA Control Register 1) is initialized as follows:

  - DIE = 0 Disables end-of-transfer interrupt.
  - DTM = 1 Chooses word transfer triggered by request, DE auto clear on end of transfer.
  - DPR = 3 Priority 3.
  - DCON = 0 Disables continuous mode.
  - DRS = $16 Chooses DMA to trigger on EFCOP output buffer full.
  - D3D = 0 Chooses non-3D mode.
  - DAM = $2C Sets the following DMA Address Mode: source address - no update, no offset; destination address - 1D, post-increment by 1, no offset.
  - DDS = 0 Destination in X memory space.
  - DSS = 1 Source in Y memory space (because EFCOP is in Y memory).
  - DCO1 = $12 Gives transfer of 9 items.
  - DSR1 = address of FDOR = $FFFFFB1
  - DDR1 = address of destination memory space

Note: Setting the DCO0 and DCO1 must be considered carefully. These registers must be loaded with one less than the number of items to be transferred. Also, the following equality must hold: DCO1=input length - filter length.

2. Initialization:

   - Enable DMA channel 1 (output) DCR1[23] DE=1

- – Enable EFCOP FCSR[0] FEN=1
- – Enable DMA channel 0. (input) DCR0[23] DE=1

3. Processing:

- – Whenever the Input Data Buffer (FDIR) is empty (i.e., FDIBE = 1), the EFCOP triggers DMA input to transfer up to four new data words to FDIR.
- – Compute F(n); The result is stored in FDOR, and this triggers the DMA for an output data transfer.

Note: The filter coefficients are stored in “reverse order,” as shown in **Figure D-1**.

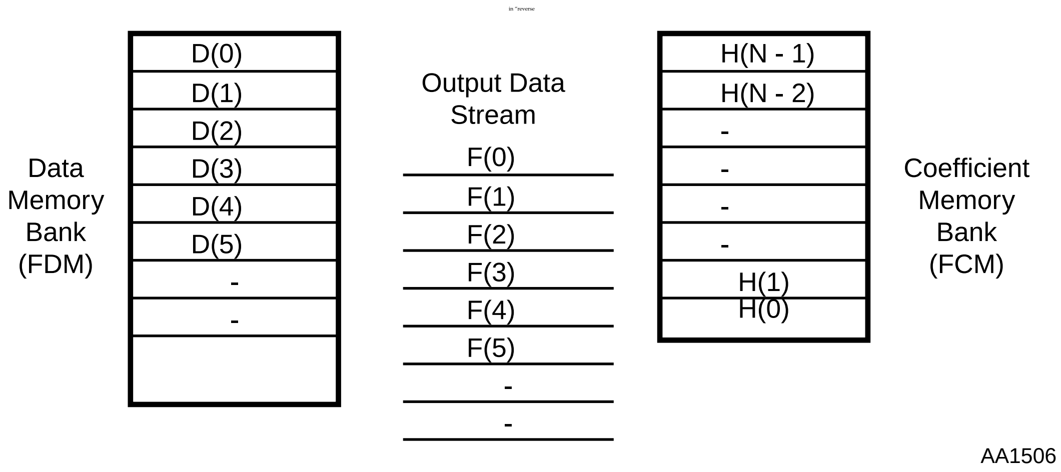


Figure D-1 Real FIR Filter Data stream

Example D-1 Real FIR Filter Using DMA Input/DMA Output

```
           INCLUDE 'ioequ.asm'
;;******************************************************************
; equates
;;******************************************************************
Start                 equ        $00100                ; main program starting address
FCON                  equ        $001                  ; EFCOP FSCR register contents:
                                                       ; enable the EFCOP
FIR_LEN               equ        20                    ; EFCOP FIR length
SRC_ADDRS             equ     $10A0         ; DMA source address point to DATA bank
DST_ADDRS             equ     $1000         ; address at which to begin output
SRC_COUNT             equ     $006003       ; DMA0 count (7*4 word transfers)
DST_COUNT             equ     8             ; number of outputs generated.
FDBA_ADDRS            equ     0             ; Input samples Start Address x:$0
FCBA_ADDRS            equ     0             ; Coeff. Start Address y:$0
;;******************************************************************
; main program
;;******************************************************************

           ORG        p:Start
           move      #FDBA_ADDRS,r0        ; FDM memory area
           move      #0,x0
           rep       #DST_COUNT
           move      x0,x:(r0)+            ; clear FDM memory area
; ** DMA channel 1 initialisation  - output from EFCOP **
           movep     #M_FDOR,x:M_DSR1      ; DMA source address points to the EFCOP FDIR
           movep     #DST_ADDRS,x:M_DDR1   ; Init DMA destination address.
           movep     #DST_COUNT,x:M_DCO1   ; Init DMA count.
           movep     #$8EB2C1,x:M_DCR1     ; Start DMA 1 with FDOBF request.
; ** EFCOP initialisation **
           movep     #FIR_LEN-1,y:M_FCNT   ; FIR length
           movep     #FDBA_ADDRS,y:M_FDBA  ; FIR input samples Start Address
           movep     #FCBA_ADDRS,y:M_FCBA  ; FIR Coeff. Start Address
           movep     #FCON,y:M_FCSR        ; Enable EFCOP

; ** DMA channel 0 initialisation - input to EFCOP **
           movep     #SRC_ADDRS,x:M_DSR0   ; DMA source address points to the DATA bank.
           movep     #M_FDIR,x:M_DDR0      ; Init DMA destination address.
           movep     #SRC_COUNT,x:M_DCO0   ; Init DMA count to line mode.
           movep     #$1,x:M_DOR0          ; DMA offset reg. is 1.
           movep     #$94AA04,x:M_DCR0     ; Init DMA control reg to line mode FDIBE request.
           nop
           nop
;;****************************************************************
           jclr #0,x:M_DSTR,*
           jclr #1,x:M_DSTR,*
           nop
           nop
stop_label
           nop
           jmp stop_label
           org x:SRC_ADDRS
           INCLUDE 'input.asm'

           org y:FCBA_ADDRS
           INCLUDE 'coefs.asm'
```

D.3.1.2 DMA Input/Polling Output

The different stages of Input/Polling are as follows:

1. Setup:
   - Set the filter count register (FCNT) to the length of the filter coefficients –1 (i.e N-1).
   - Set the data and coefficient base address pointers (FDBA, FCBA).
   - Set the operation mode (FCSR[5:4] = FOM[00],), = 1).
   - Set the initialization mode (FCSR[7] = FPRC = 0).
   - Set DMA registers: DMA input: as per channel 0 in **Section D.3.1.1**

2. Initialization:
   - Enable EFCOP FCSR[0] FEN=1.
   - Enable DMA input channel, DCR0[23] DE=1.
3. Processing:
   - Whenever the Input Data Buffer (FDIR) is empty (i.e., FDIBE = 1), the EFCOP triggers DMA input to transfer up to four new data words to FDM via FDIR.
   - Compute F(n); the result is stored in FDOR.
   - The core keeps polling the FCSR:FDOBF bit and stores the data in memory.

Example D-2 Real FIR Filtering using DMA input/Polling output

```
INCLUDE 'ioequ.asm'
;;******************************************************************
; equates
;;******************************************************************
Start      equ        $00100     ; main program starting address
FCON            equ     $001     ; EFCOP FSCR register contents:
                                 ; enable the EFCOP
FIR_LEN         equ     20       ; EFCOP FIR length
SRC_ADDRS       equ     $10A0    ; DMA source address point to DATA bank
DST_ADDRS       equ     $1000    ; address at which to begin output
SRC_COUNT       equ     $006003  ; DMA0 count (7*4 word transfers)
DST_COUNT       equ     8        ; number of outputs generated.
FDBA_ADDRS      equ     0        ; Input samples Start Address x:$0
FCBA_ADDRS      equ     0        ; Coeff. Start Address y:$0
;;******************************************************************
; main program
;;*****************************************************************


           ORG        p:Start
           move       #0,b
           move       #0,a
           move       #DST_COUNT,b0         ; counter for output interrupt
           move       #FDBA_ADDRS,r0        ; FDM memory area
           move       #0,x0
           rep        #DST_COUNT
           move       x0,x:(r0)+            ; clear FDM memory area
           move       #DST_ADDRS,r0         ; Destination address
; ** EFCOP initialisation **
           movep      #FIR_LEN-1,y:M_FCNT   ; FIR length
           movep      #FDBA_ADDRS,y:M_FDBA  ; FIR input samples Start Address
           movep      #FCBA_ADDRS,y:M_FCBA  ; FIR Coeff. Start Address
           movep      #FCON,y:M_FCSR        ; Enable EFCOP

; ** DMA channel 0 initialisation - input to EFCOP **
           movep       #SRC_ADDRS,x:M_DSR0  ; DMA source address points to the DATA bank.
           movep      #M_FDIR,x:M_DDR0      ; Init DMA destination address.
           movep      #SRC_COUNT,x:M_DCO0   ; Init DMA count to line mode.
           movep      #$1,x:M_DOR0          ; DMA offset reg. is 1.
           movep      #$94AA04,x:M_DCR0     ; Init DMA control reg to line mode FDIBE request.
           nop
           nop
;;****************************************************************
           do      #DST_COUNT,endd
           nop
           jclr #15,y:M_FCSR,*
           movep y:M_FDOR,x:(r0)+
endd
           nop
           nop
stop_label
           nop
           jmp stop_label
           org x:SRC_ADDRS
           INCLUDE 'input.asm'

           org y:FCBA_ADDRS
INCLUDE 'coefs.asm'
```

D.3.1.3 DMA Input/Interrupt Output

The different stages of DMA input and interrupt output are as follows:

1. Setup:
   - Set the filter count register (FCNT) to the length of the filter coefficients –1 (i.e N-1).
   - Set the Data and Coefficient Base Address pointers (FDBA, FCBA).
   - Set the operation mode (FCSR[5:4] = FOM[00],).
   - Set Initialization mode (FCSR[7] = FPRC = 0).
   - Set Filter Data Output Interrupt Enable FSCR[11]=FDOIE=1.
   - Set DMA register with DMA input as per channel 0 in **Section D.3.1.1**.
2. Initialization:
   - Enable interrupts in the Interrupt Priority Register IPRP[10:11]=E0L=11.
   - Enable interrupts in the Status Register SR[8:9]=00.
   - Enable EFCOP FCSR[0]=FEN=1.
   - Enable the DMA input channel, DCR0[23]=DE=1.
3. Processing:
   - Whenever the Input Data Buffer (FDIR) is empty (i.e., FDIBE = 1), the EFCOP triggers DMA, which loads the next input into the FDIR.

- Compute F(n); the result is stored in FDOR; The core is interrupted when FDOBF is set and stores the data in memory.

Example D-3 Real FIR Filter DMA Input/Interrupt Output

```
INCLUDE 'ioequ.asm'
;;*****************************************************************
; equates
;;*****************************************************************
Start      equ        $00100     ; main program starting address
FCON       equ        $801       ; EFCOP FSCR register contents:
                                 ; enable output interrupt
                                 ; enable the EFCOP
FIR_LEN    equ        20         ; EFCOP FIR length
SRC_ADDRS  equ        $10A0      ; DMA source address point to DATA bank
DST_ADDRS  equ        $1000      ; address at which to begin output
SRC_COUNT  equ        $006003    ; DMA0 count (7*4 word transfers)
DST_COUNT  equ        8          ; number of outputs generated.
FDBA_ADDRS equ        0          ; Input samples Start Address x:$0
FCBA_ADDRS equ        0          ; Coeff. Start Address y:$0
;;*****************************************************************
           org P:$0
           jmp        Start

           ORG        p:$6a
           jsr        >kdo
           nop
           nop
;;*****************************************************************
; main program
;;*****************************************************************

           ORG        p:Start
; ** interrupt initialisation **
           bset       #10,x:M_IPRP          ;
           bset       #11,x:M_IPRP          ; enable EFCOP interrupts in IPRP
           bclr       #8,SR                 ;
           bclr       #9,SR                 ; enable interrupts in SR
           move       #0,b
           move       #0,a
           move       #DST_COUNT,b0         ; counter for output interrupt
           move       #FDBA_ADDRS,r0        ; FDM memory area
           move       #0,x0
           rep        #DST_COUNT
           move       x0,x:(r0)+            ; clear FDM memory area
           move       #DST_ADDRS,r0         ; Destination address
; ** EFCOP initialisation **
           movep      #FIR_LEN-1,y:M_FCNT   ; FIR length
           movep      #FDBA_ADDRS,y:M_FDBA  ; FIR input samples Start Address
           movep      #FCBA_ADDRS,y:M_FCBA  ; FIR Coeff. Start Address
           movep      #FCON,y:M_FCSR        ; Enable EFCOP
```

```
; ** DMA channel 0 initialisation - input to EFCOP **
           movep      #SRC_ADDRS,x:M_DSR0    ; DMA source address points to the DATA bank.
           movep      #M_FDIR,x:M_DDR0       ; Init DMA destination address.
           movep      #SRC_COUNT,x:M_DCO0    ; Init DMA count to line mode.
           movep      #$1,x:M_DOR0           ; DMA offset reg. is 1.
           movep      #$94AA04,x:M_DCR0      ; Init DMA control reg to line mode FDIBE request.
           nop
           nop
;;******************************************************************
wait1
           jset       #11,y:M_FCSR,*         ; Wait until FDOIE is cleared.
           do         #40,endd
           nop
endd
           nop
           nop
stop_label
           nop
           jmp stop_label
;;******************************************************************
kdo                                          ; Interrupt handler for EFCOP output
           movep   y:M_FDOR,x:(r0)+ ; Get y(k) from FDOR
                                             ; Store in destination memory space.
           nop
           dec        b
           jne        cont
           nop
           bclr       #11,y:M_FCSR      ; Disable output interrupt
cont
           rti
           nop
           nop
           nop
           org        x:SRC_ADDRS
           INCLUDE    'input.asm'

           org        y:FCBA_ADDRS
           INCLUDE    'coefs.asm'
```

D.3.2 Real FIR Filter With Decimation by M

An N tap real FIR filter with decimation by M of a sequence of real numbers is represented by,

$$F(n|_{M}) = \sum_{i=0}^{N-1} H(i) \cdot D(n-i)$$

A DMA data transfer occurs in the following stages for both input and output.The stages are the similar to the ones described in **Section D.3.1.1**. The difference is: set FDCH[11:8] = FDCM =M.

1. Processing:
   - Whenever the Input Data Buffer (FDIR) is empty (i.e., FDIBE = 1), the EFCOP triggers DMA input to transfer up to four new data words to FDM via FDIR.
   - Compute F(n); the result is stored in FDOR; the EFCOP triggers DMA output for an output data transfer.
   - Repeat M times:

```
{
     Get new data word; EFCOP increments data memory pointer.
}
```

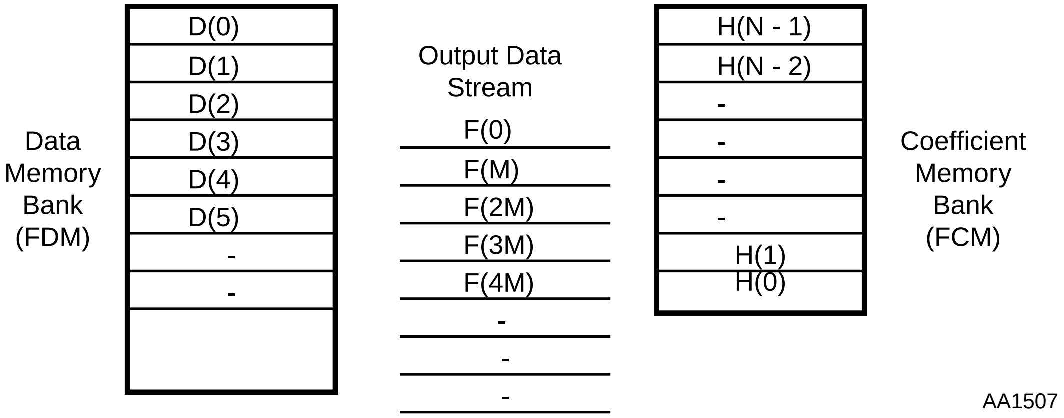


Figure D-2 Real FIR Filter Data stream with Decimation by M

D.3.3 Adaptive FIR Filter

An adaptive FIR filter is represented in **Figure D-3**. The goal of the FIR filter is to adjust the filter coefficients so that the output, F(n), becomes as close as possible to the desired signal, R(n)—that is, E(n) -> 0.

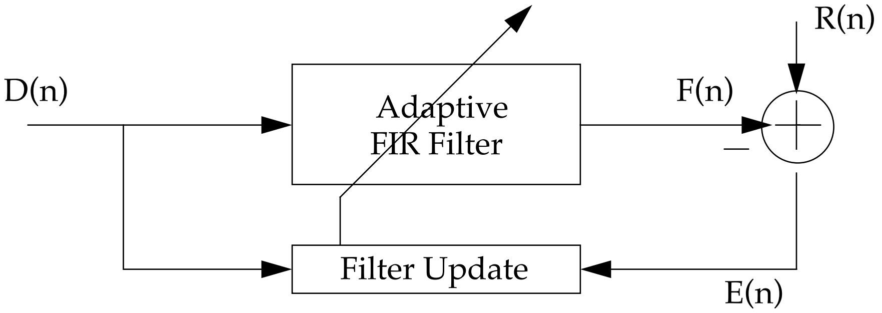


Figure D-3 Adaptive FIR Filter

The adaptive FIR filter typically comprises four stages, which are performed for each input sample at time n:

- **Stage 1.** The FIR filter output value is calculated for the EFCOP FIR session according to this equation:

$$F(n) = \sum_{i=0}^{N-1} H_n(i)D(n-i)$$

  where $H_n(i)$ are the filter coefficients at time n, D(n) is the input signal and F(n) is the filter output at time n,. This stage requires N MAC operations, calculated by the EFCOP FMAC unit.

- **Stage 2.** The core calculates the error signal, E(n), in software according to the following equation:

$$E(n) = R(n) - F(n)$$

  where R(n) is the desired signal at time n. This stage requires a single arithmetic operation.

- **Stage 3.** The core calculates the weight multiplier, Ke(n), in software according to the following equation:

$$K_e(n) = K * E(n)$$

where K is the convergence factor of the algorithm. After calculating the weight multiplier, K_e, the core must write it into the FKIR.

- **Stage 4.** The coefficients are updated by the EFCOP update session:

$$H_{n+1}(i) = H_n(i) + K_e D(n-i)$$

where $H_{n+1}(i)$ are the adaptive filter coefficients at time n+1, $K_e(n)$ is the weight multiplier at time n, and D(n) is the input signal. This stage starts immediately after $K_e(n)$ is written in the FKIR (if adaptive mode is enabled).

D.3.3.1 Implementation Using Polling

Figure D-4 shows a flowchart for an adaptive FIR filter that uses polling to transfer data.

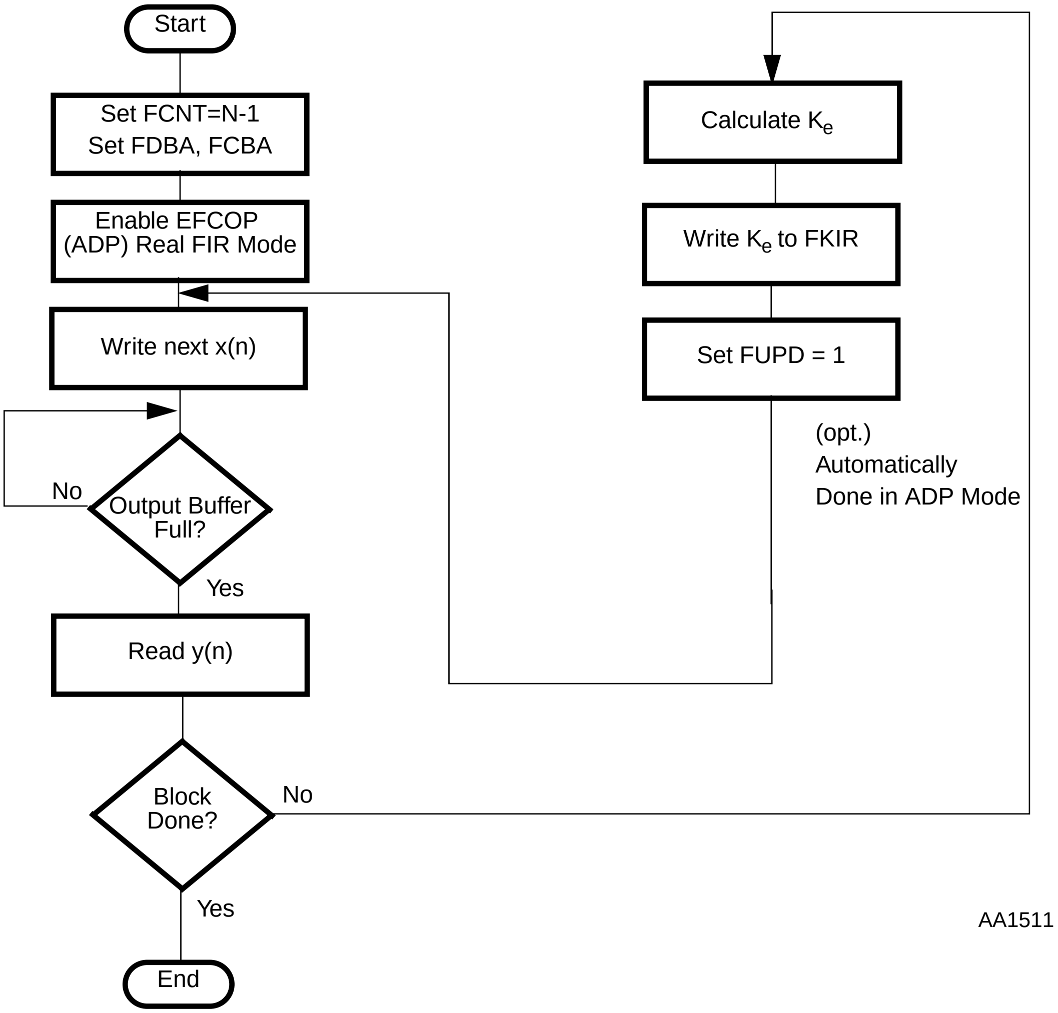


Figure D-4 Adaptive FIR Filter Using Polling

D.3.3.2 Implementation Using DMA Input and Interrupt Output

Figure D-5 shows a flowchart for an adaptive FIR filter that uses DMA and an interrupt to transfer data.

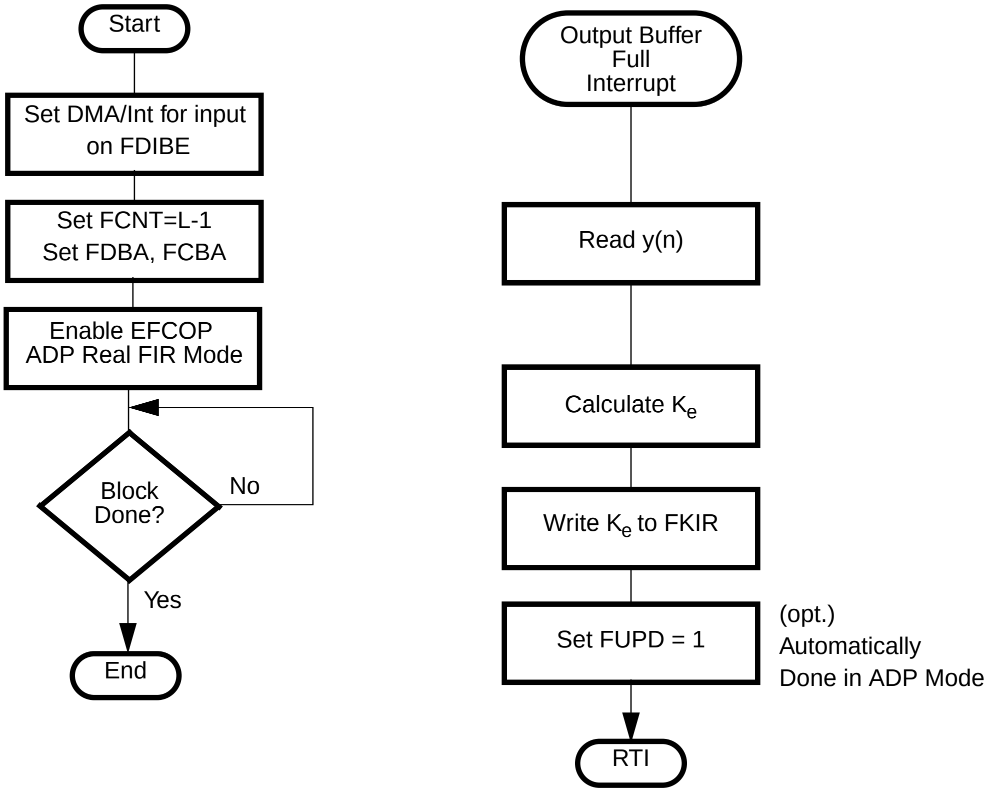


Figure D-5 Adaptive FIR Filter Using DMA Input and Interrupt Output

D.3.3.3 Updating an FIR Filter

The following example shows an FIR adaptive filter that is updated using the LMS algorithm.

Example D-4 FIR Adaptive Filter Update Using the LMS Algorithm

```
           TITLE 'ADAPTIVE'
           INCLUDE 'ioequ.asm'

;;*************************************************************************************
; equates
;;*************************************************************************************
Start           equ     $00100          ; main program starting address
FCON            equ     $805            ; EFCOP FSCR register contents:
           ; enable output interrupt
           ; Choose adaptive real FIR mode
           ; enable the EFCOP
```

```
FIR_LEN         equ     20              ; EFCOP FIR length
DES_ADDRS       equ     $1200           ; Desired signal R(n)
SRC_ADDRS       equ     $1100           ; Reference signal D(n)
DST_ADDRS       equ     $1000           ; address at which to begin output (signal F(n))
SRC_COUNT       equ     $06003          ; DMA0 count (7*4 word transfers)
DST_COUNT       equ     8               ; number of outputs generated.
MU2             equ     $100000         ; stepsize mu = 0.0625 (i.e. 2mu = 0.125)
FDBA_ADDRS      equ     0               ; Input samples Start Address x:$0
FCBA_ADDRS      equ     0               ; Coeff. Start Address y:$0

;;**************************************************************************************
           org p:$0
           jmp     Start

           ORG p:$6a

           jsr     >kdo
           nop
           nop

;;**************************************************************************************
; main program
;;**************************************************************************************
           ORG p:Start

; ** interrupt initialisation **

           bset    #10,x:M_IPRP            ;
           bset    #11,x:M_IPRP            ; enable EFCOP interrupts in IPRP

           bclr    #8,SR                   ;
           bclr    #9,SR                   ; enable interrupts in SR

           move    #0,b
           move    #0,a
           move    #DST_COUNT,b0           ; counter for output interrupt

; ** FDM memory initialisation **

           move    #FDBA_ADDRS,r0          ; FDM memory area
           move    #0,x0
           rep     #FIR_LEN
           move    x0,x:(r0)+              ; clear FDM memory area

; ** address register initialisation **

           move    #DST_ADDRS,r0           ; Destination address
           move    #DES_ADDRS,r1           ; Desired signal address

           rep     #FIR_LEN-1
           move    (r1)+                     ; Set reference pointer correctly
```

```
; ** EFCOP initialisation **

          movep   #FIR_LEN-1,y:M_FCNT        ; FIR length
          movep   #FDBA_ADDRS,y:M_FDBA       ; FIR input samples Start Address
          movep   #FCBA_ADDRS,y:M_FCBA       ; FIR Coeff. Start Address
          movep   #FCON,y:M_FCSR             ; Enable EFCOP

; ** DMA channel 0 initialisation - input to EFCOP **

          movep   #SRC_ADDRS,x:M_DSR0        ; DMA source address points to the DATA bank.
          movep   #M_FDIR,x:M_DDR0           ; Init DMA destination address.
          movep   #SRC_COUNT,x:M_DCO0        ; Init DMA count to line mode.
          movep   #$1,x:M_DOR0               ; DMA offset reg. is 1.
          movep   #$94AA04,x:M_DCR0          ; Init DMA control reg to line mode FDIBE request.

          nop
          nop

;;******************************************************************
wait1
          jset    #11,y:M_FCSR,*             ; Wait until FDOIE is cleared.
          do      #40,endd
          nop
endd
          nop
          nop
stop_label
          nop
          jmp stop_label

;;******************************************************************
kdo                                          ; Interrupt handler for EFCOP output

          movep   y:M_FDOR,x:(r0)            ; Get F(n) from FDOR

                                             ; Store in destination memory space.
;******* Calculate Ke *********

          move    x:(r1)+,a                  ; Retrieve desired value R(n)

          move    x:(r0)+,y0                 ;
          sub     y0,a                       ; calculate E(n) = R(n) - F(n)

          move    #MU2,y0                    ;
          move    a,y1                       ;
          mpy     y0,y1,a                    ; calculate Ke = mu * 2 * E(n)

;******************************
          movep      a1,y:M_FKIR             ; store Ke in FKIR

          dec     b
          jne     cont
          nop
          bclr    #11,y:M_FCSR               ; Disable output interrupt
```

```
cont
          rti
          nop
          nop
          nop

;;*********************************************************************
;;*********************************************************************
          ORG y:FCBA_ADDRS

          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000
          dc      $000000

          ORG x:SRC_ADDRS

          INCLUDE 'input.asm'; Reference signal D(n)

          ORG x:DES_ADDRS

          INCLUDE 'desired.asm'; Desired signal R(n)
```

D.4 VERIFICATION For All Exercises

D.4.1 Input Sequence (input.asm)

```
dc    $000000
dc    $37cc8a
dc    $343fae
dc    $0b63b1
dc    $0595b4
dc    $38f46e
dc    $6a4ea2
dc    $5e8562
dc    $2beda5
dc    $1b3cd0
dc    $42f452
dc    $684ca0
dc    $5093b0
dc    $128ab8
dc    $f74ee1
dc    $15c13a
dc    $336e48
dc    $15e98e
dc    $d428d2
dc    $b76af5
dc    $d69eb3
dc    $f749b6
dc    $dee460
dc    $a43601
dc    $903d59
dc    $b9999a
dc    $e5744e
```

D.4.2 Filter Coefficients (coefs.asm)

```
dc  $F8125C
dc  $F77839
dc  $F4E9EE
dc  $F29373
dc  $F2DC9A
dc  $F51D6E
dc  $F688CE
dc  $F6087E
dc  $F5B5D3
dc  $F7E65E
dc  $FBE0F8
dc  $FEC8B7
dc  $FF79F5
dc  $000342
dc  $02B24F
dc  $06C977
dc  $096ADD
dc  $097556
dc  $08FD54
dc  $0A59A5
```

D.4.3 Output Sequence for Examples D-1, D-2, and D-3

```
$d69ea9
$ccae36
$c48f2a
$be8b28
$bad8c5
$b9998c
$bad8cb
$be8b2d
$c48f27
```

D.4.4 Desired Signal for Example D-4

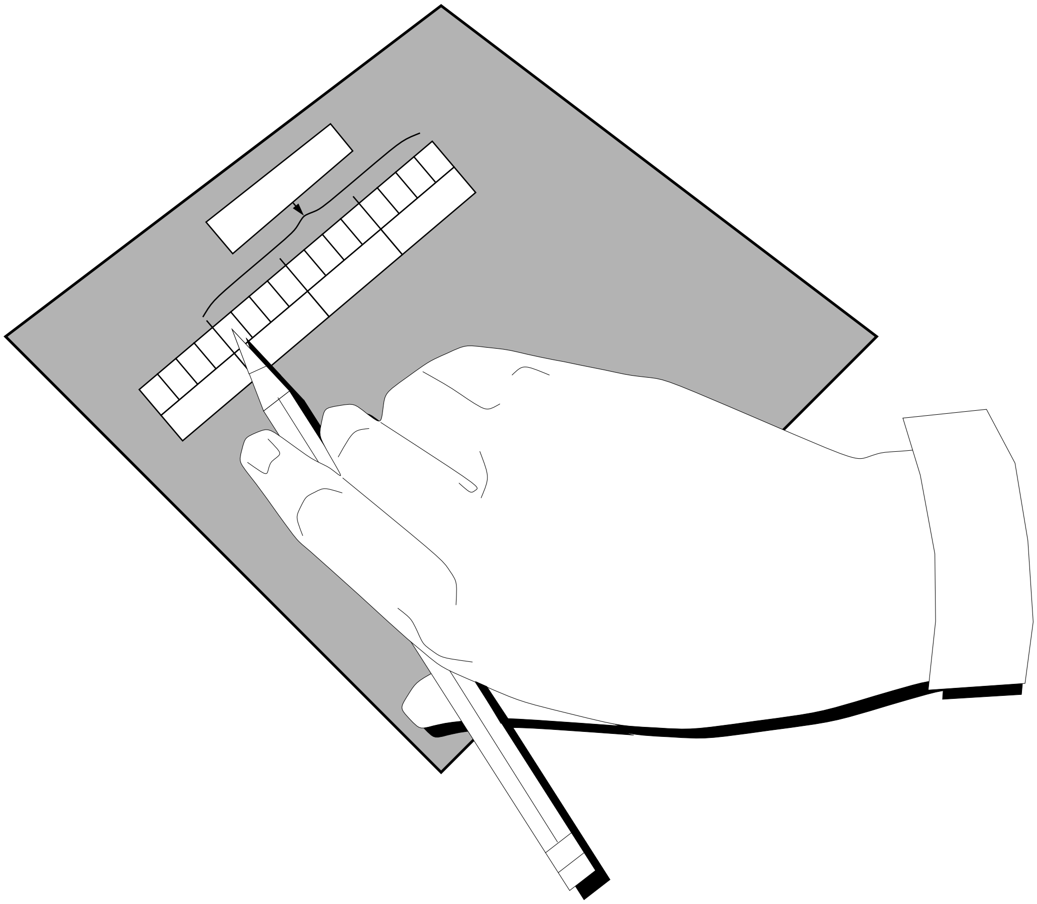
```
dc    $000000
dc    $0D310F
dc    $19EA7C
dc    $25B8E2
dc    $30312E
dc    $38F46D
dc    $3FB327
dc    $443031
dc    $4642D6
dc    $45D849
dc    $42F452
dc    $3DB126
dc    $363E7F
dc    $2CDFE8
dc    $21EA5B
dc    $15C13A
dc    $08D2CE
dc    $FB945D
dc    $EE7E02
dc    $E2066F
dc    $D69EB2
dc    $CCAE3C
dc    $C48F2F
dc    $BE8B32
dc    $BAD8D3
dc    $B99999
dc    $BAD8D3
dc    $BE8B32
```

D.4.5 Output Sequence for Example D-4

```
$000000
$f44c4c
$ee54b3
$e7cd6b
$daed26
$cc1071
$c7db1c
$cdfe45
```

APPENDIX E

PROGRAMMING REFERENCE



E.1 INTRODUCTION

This reference for programmers includes a table showing the addresses of all DSP memory-mapped peripherals, an exception priority table, and programming sheets for the major programmable DSP registers. The programming sheets are grouped in the following order: central processor, Phase Lock Loop, (PLL), Host Interface (HI08), Enhanced Synchronous Serial Interface (ESSI), Serial Communication Interface (SCI), Timer, GPIO, and EFCOP. Each sheet provides room to write in the value of each bit and the hexadecimal value for each register. You can photocopy these sheets and reuse them for each application development project. For details on the instruction set of the DSP56300 family of DSPs, see the *DSP56300 Family Manual*.

- **Table E-2 Internal X I/O Memory Map** on page E-5 and **Table E-3 Internal Y I/O Memory Map** on page E-12 list the memory addresses of all on-chip peripherals.
- **Table E-4 Interrupt Sources** on page E-13 lists the interrupt starting addresses and sources.
- **Table E-5 Interrupt Source Priorities within an IPL** on page E-15 lists the priorities of specific interrupts within interrupt priority levels.
- The programming sheets appear in this manual as figures (listed in **Table E-1**); they show the major programmable registers on the DSP56307.

Table E-1 Programming Sheets

Central Processor

PLL

HI08

Table E-1 Programming Sheets

ESSI

SCI

Timers

GPIO

EFCOP

E.2 INTERNAL I/O MEMORY MAP

Table E-2 Internal X 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 E-2 Internal X 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 E-2 Internal X 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 Port Control Register (HPCR)
	$FFC3	$FFFFC3	Host Status Register (HSR)
	$FFC2	$FFFFC2	Host Control Register (HCR)
	$FFC1	$FFFFC1	Reserved
	$FFC0	$FFFFC0	Reserved

Table E-2 Internal X 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 E-2 Internal X 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 E-2 Internal X 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 E-2 Internal X 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

Table E-3 Internal Y I/O Memory Map

Peripheral	16-Bit Address	24-Bit Address	Register Name
Enhanced Filter Coprocessor (EFCOP)	$FFBF	$FFFFBF	RESERVED
	$FFBE	$FFFFBE	RESERVED
	$FFBD	$FFFFBD	RESERVED
	$FFBC	$FFFFBC	RESERVED
	$FFBB	$FFFFBB	RESERVED
	$FFBA	$FFFFBA	RESERVED
	$FFB9	$FFFFB9	RESERVED
	$FFB8	$FFFFB8	EFCOP Decimation/Channel (FDCH) Register
	$FFB7	$FFFFB7	EFCOP Coefficient Base Address (FCBA)
	$FFB6	$FFFFB6	EFCOP Data Base Address (FDBA)
	$FFB5	$FFFFB5	EFCOP ALU Control Register (FACR)
	$FFB4	$FFFFB4	EFCOP Control Status Register (FCSR)
	$FFB3	$FFFFB3	EFCOP Filter Count (FCNT) Register
	$FFB2	$FFFFB2	EFCOP K-Constant Register (FKIR)
	$FFB1	$FFFFB1	EFCOP Data Output Register (FDOR)
	$FFB0	$FFFFB0	EFCOP Data Input Register (FDIR)
	$FFAF– $FF80	$FFFFAF– $FFFF80	RESERVED

E.3 INTERRUPT ADDRESSES AND SOURCES

Table E-4 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 E-4 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:$68	0–2	EFCOP Data Input Buffer Empty
VBA:$6A	0–2	EFCOP Data Output Buffer Full
VBA:$6C	0–2	Reserved
VBA:$6E	0–2	Reserved
:	:	:
VBA:$FE	0–2	Reserved

E.4 INTERRUPT PRIORITIES

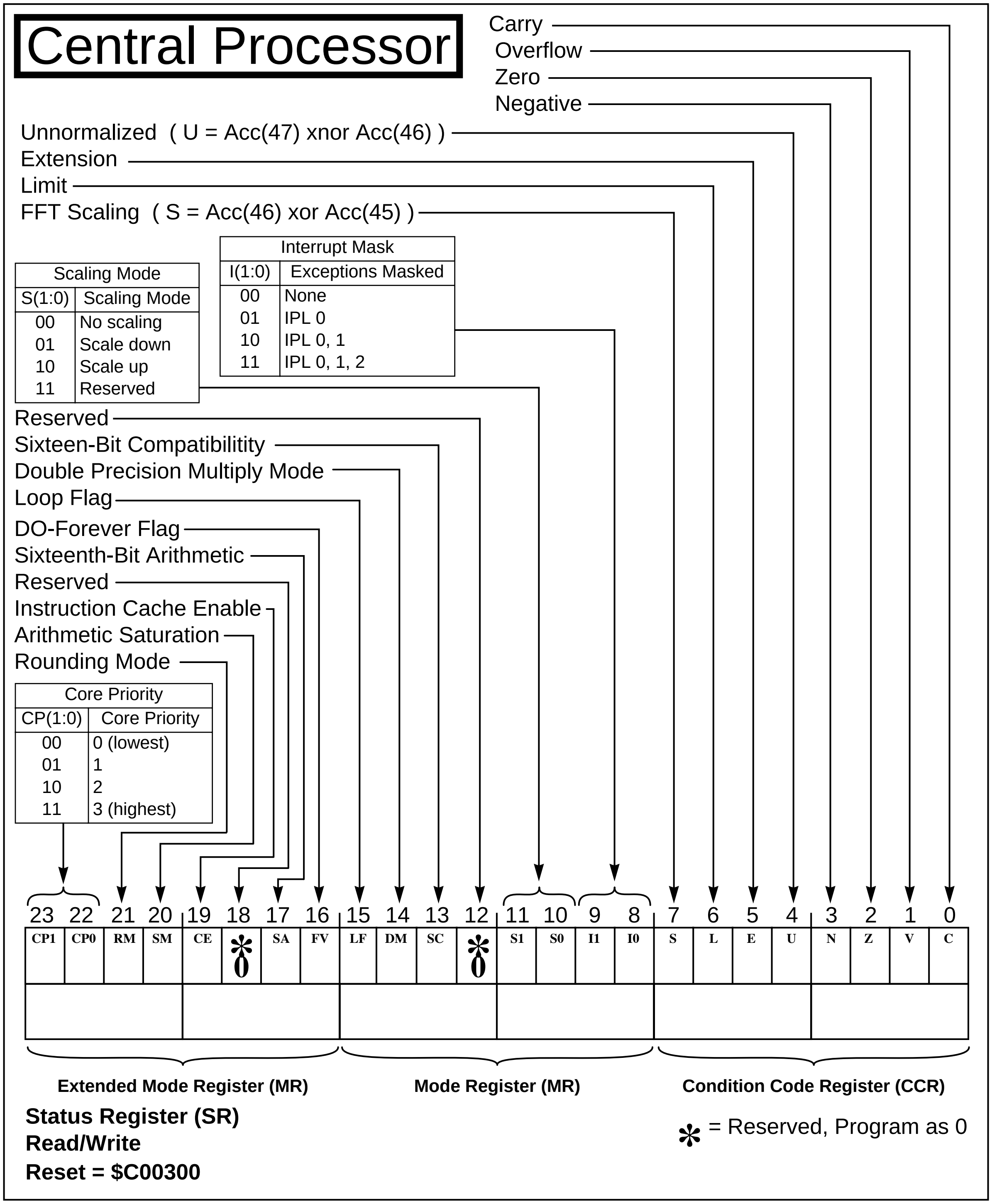
Table E-5 Interrupt Source Priorities within an IPL

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 E-5 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
	TIMER2 Compare Interrupt
	EFCOP Data Input Buffer Empty
Lowest	EFCOP Data Output Buffer Full

E.5 PROGRAMMING SHEETS


Central Processor
Carry
Overflow
Zero
Negative
Unnormalized (U = Acc(47) xnor Acc(46))
Extension
Limit
FFT Scaling (S = Acc(46) xor Acc(45))
Reserved
Sixteen-Bit Compatibilitity
Double Precision Multiply Mode
Loop Flag
DO-Forever Flag
Sixteenth-Bit Arithmetic
Reserved
Instruction Cache Enable
Arithmetic Saturation
Rounding Mode
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
CP1 CP0 RM SM CE * 0 SA FV LF DM SC * 0 S1 S0 I1 I0 S L E U N Z V C
Extended Mode Register (MR)
Mode Register (MR)
Condition Code Register (CCR)

Scaling Mode	
S(1:0)	Scaling Mode
00	No scaling
01	Scale down
10	Scale up
11	Reserved

Interrupt Mask	
I(1:0)	Exceptions Masked
00	None
01	IPL 0
10	IPL 0, 1
11	IPL 0, 1, 2

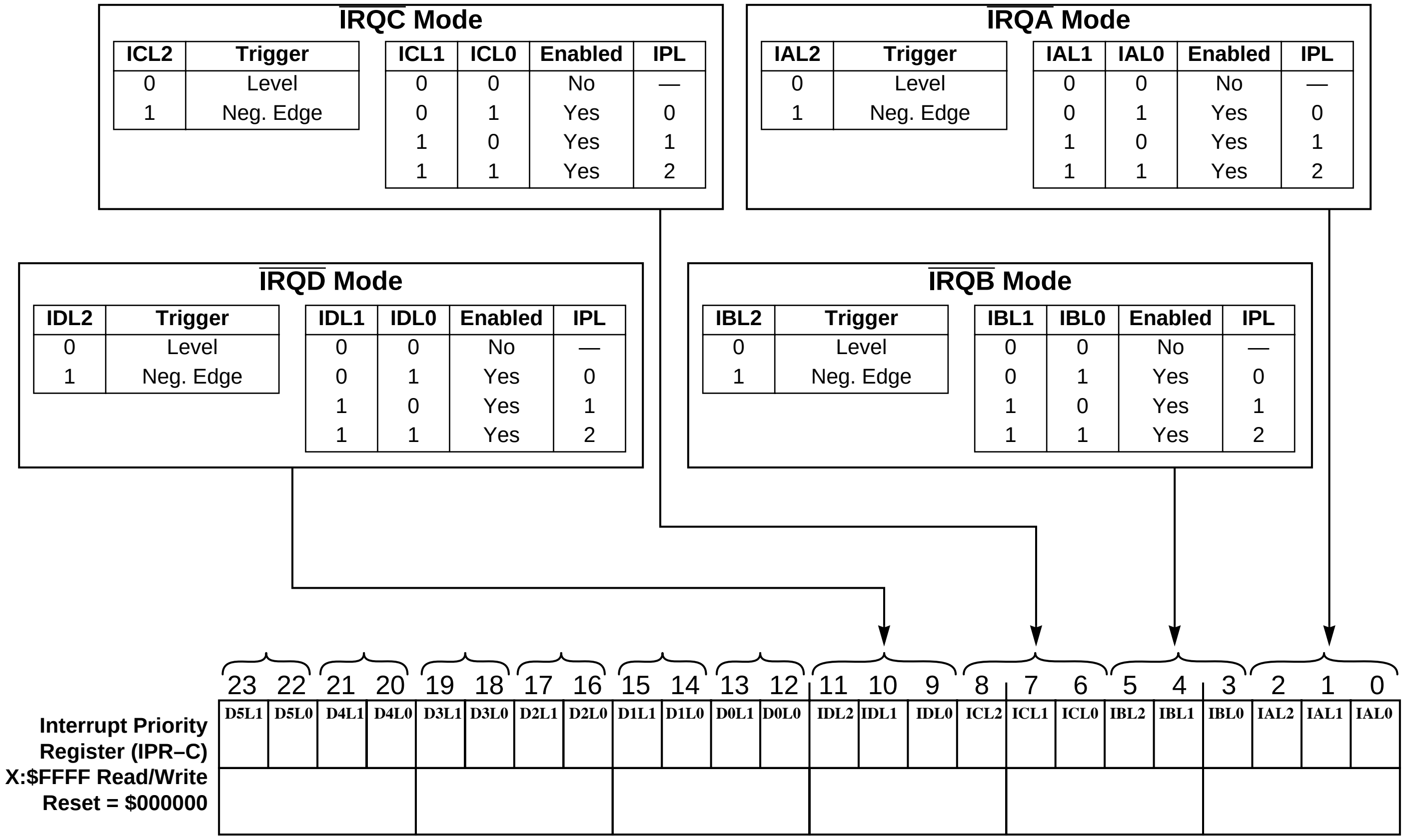
Core Priority	
CP(1:0)	Core Priority
00	0 (lowest)
01	1
10	2
11	3 (highest)

Status Register (SR)
Read/Write
Reset = $C00300

* = Reserved, Program as 0

Figure E-1 Status Register (SR)

CENTRAL PROCESSOR



$\overline{\text{IRQC}}$ Mode

ICL2	Trigger
0	Level
1	Neg. Edge

ICL1	ICL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

$\overline{\text{IRQA}}$ Mode

IAL2	Trigger
0	Level
1	Neg. Edge

IAL1	IAL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

$\overline{\text{IRQD}}$ Mode

IDL2	Trigger
0	Level
1	Neg. Edge

IDL1	IDL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

$\overline{\text{IRQB}}$ Mode

IBL2	Trigger
0	Level
1	Neg. Edge

IBL1	IBL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

Interrupt Priority Register (IPR–C)
X:$FFFF 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
D5L1	D5L0	D4L1	D4L0	D3L1	D3L0	D2L1	D2L0	D1L1	D1L0	D0L1	D0L0	IDL2	IDL1	IDL0	ICL2	ICL1	ICL0	IBL2	IBL1	IBL0	IAL2	IAL1	IAL0

Figure E-2 Interrupt Priority Register–Core (IPR–C)

Figure E-3 Interrupt Priority Register – Peripherals (IPR–P)

CENTRAL PROCESSOR

ESSI1 IPL

S1L1	S1L0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

Host IPL

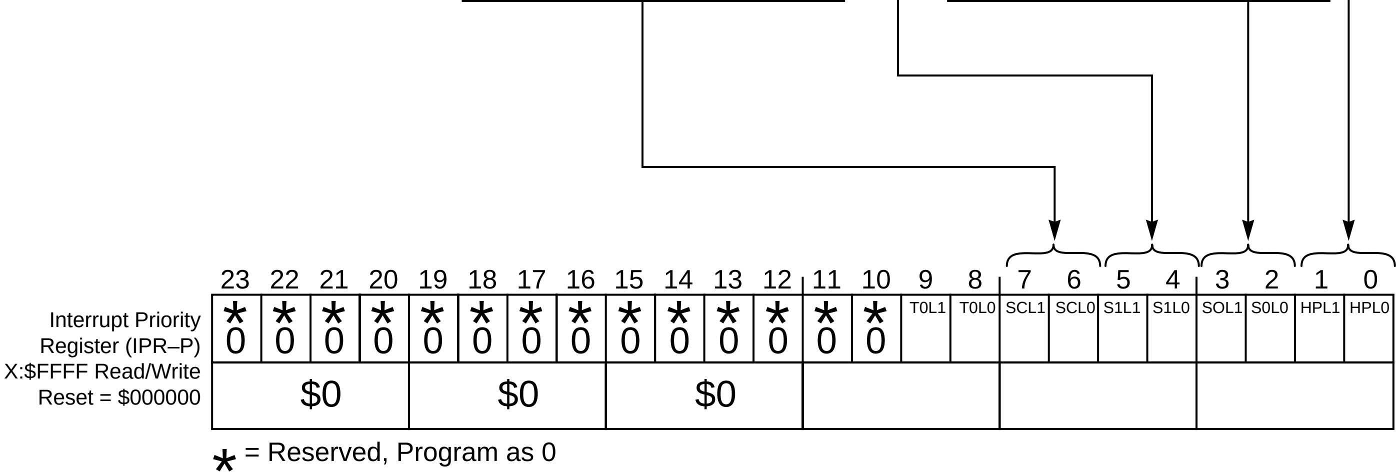
HPL1	HPL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

SCI IPL

SCL1	SCL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

ESSI0 IPL

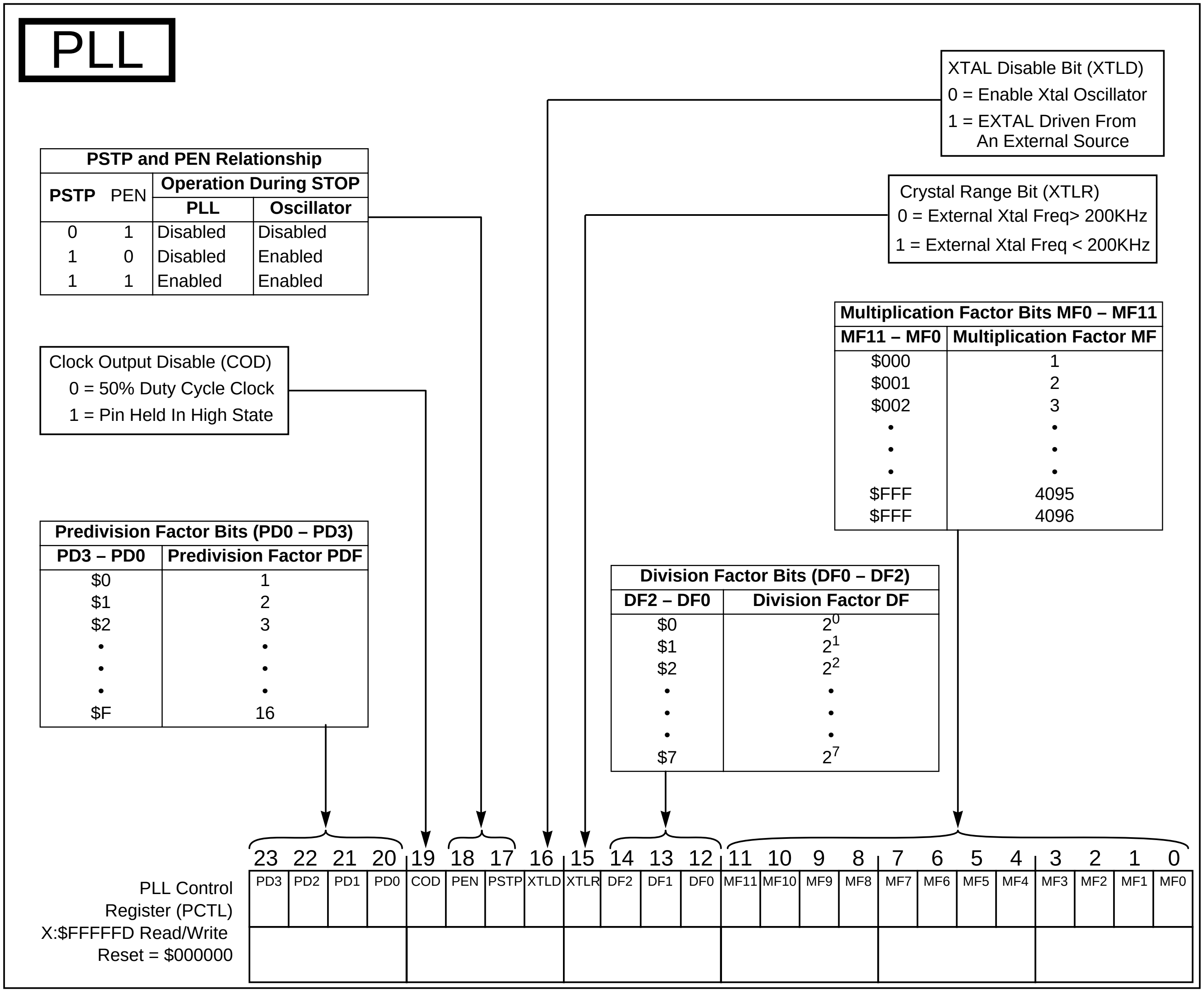
S0L1	S0L0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2



Interrupt Priority Register (IPR–P)
X:$FFFF 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	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	T0L1	T0L0	SCL1	SCL0	S1L1	S1L0	SOL1	S0L0	HPL1	HPL0
$0				$0				$0															

* = Reserved, Program as 0



PSTP and PEN Relationship			
		Operation During STOP	
PSTP	PEN	PLL	Oscillator
0	1	Disabled	Disabled
1	0	Disabled	Enabled
1	1	Enabled	Enabled

Clock Output Disable (COD)
0 = 50% Duty Cycle Clock
1 = Pin Held In High State

Predivision Factor Bits (PD0 – PD3)	
PD3 – PD0	Predivision Factor PDF
$0	1
$1	2
$2	3
•	•
•	•
•	•
$F	16

XTAL Disable Bit (XTLD)
0 = Enable Xtal Oscillator
1 = EXTAL Driven From An External Source

Crystal Range Bit (XTLR)
0 = External Xtal Freq> 200KHz
1 = External Xtal Freq < 200KHz

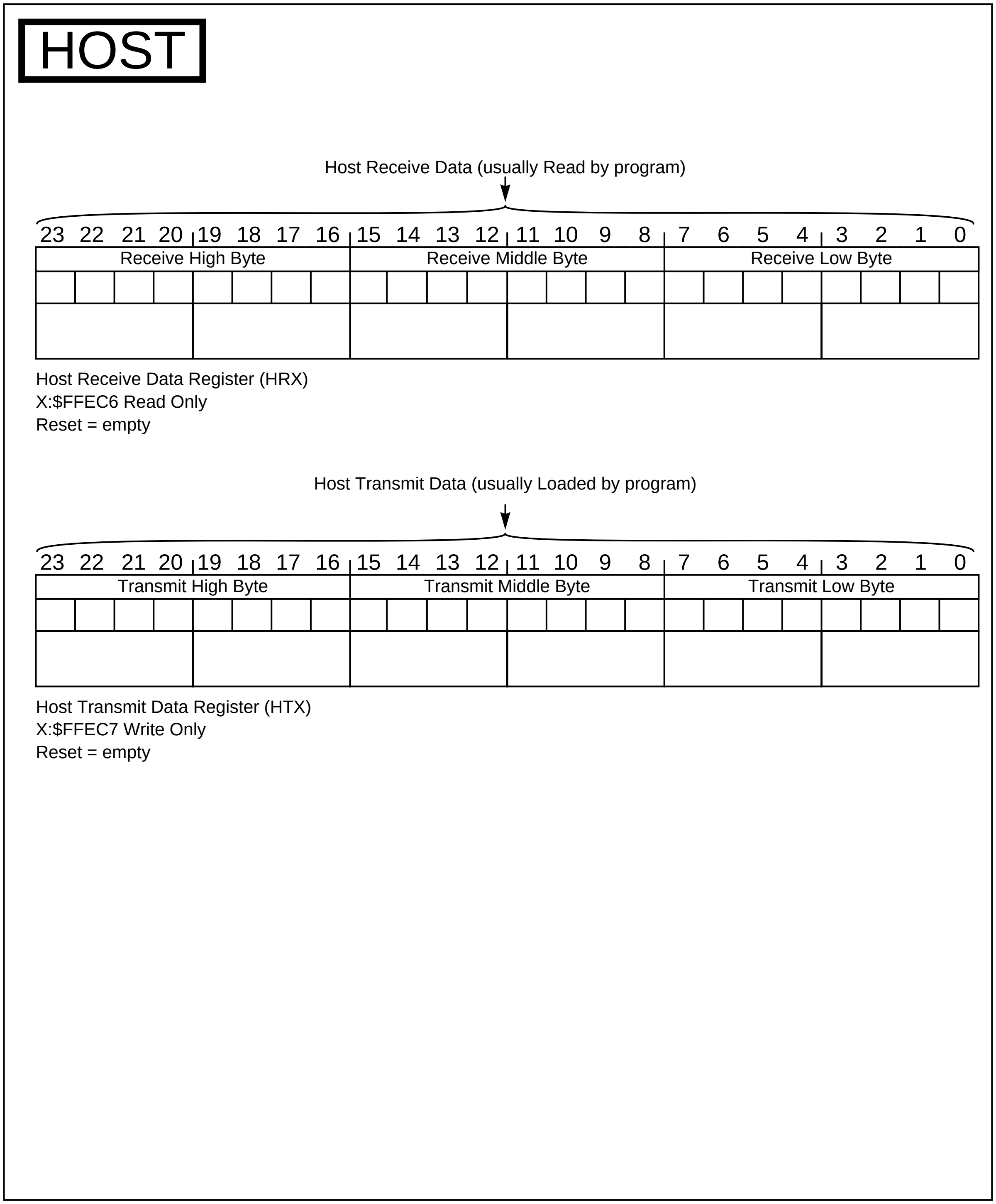
Multiplication Factor Bits MF0 – MF11	
MF11 – MF0	Multiplication Factor MF
$000	1
$001	2
$002	3
•	•
•	•
•	•
$FFF	4095
$FFF	4096

Division Factor Bits (DF0 – DF2)	
DF2 – DF0	Division Factor DF
$0	2^0
$1	2^1
$2	2^2
•	•
•	•
•	•
$7	2^7

PLL Control
Register (PCTL)
X:$FFFFFD 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
PD3	PD2	PD1	PD0	COD	PEN	PSTP	XTLD	XTLR	DF2	DF1	DF0	MF11	MF10	MF9	MF8	MF7	MF6	MF5	MF4	MF3	MF2	MF1	MF0

Figure E-4 Phase Lock Loop Control Register (PCTL)



Host Receive Data Register (HRX)
X:$FFEC6 Read Only
Reset = empty

Host Transmit Data Register (HTX)
X:$FFEC7 Write Only
Reset = empty

Figure E-5 Host Receive and Host Transmit Data Registers

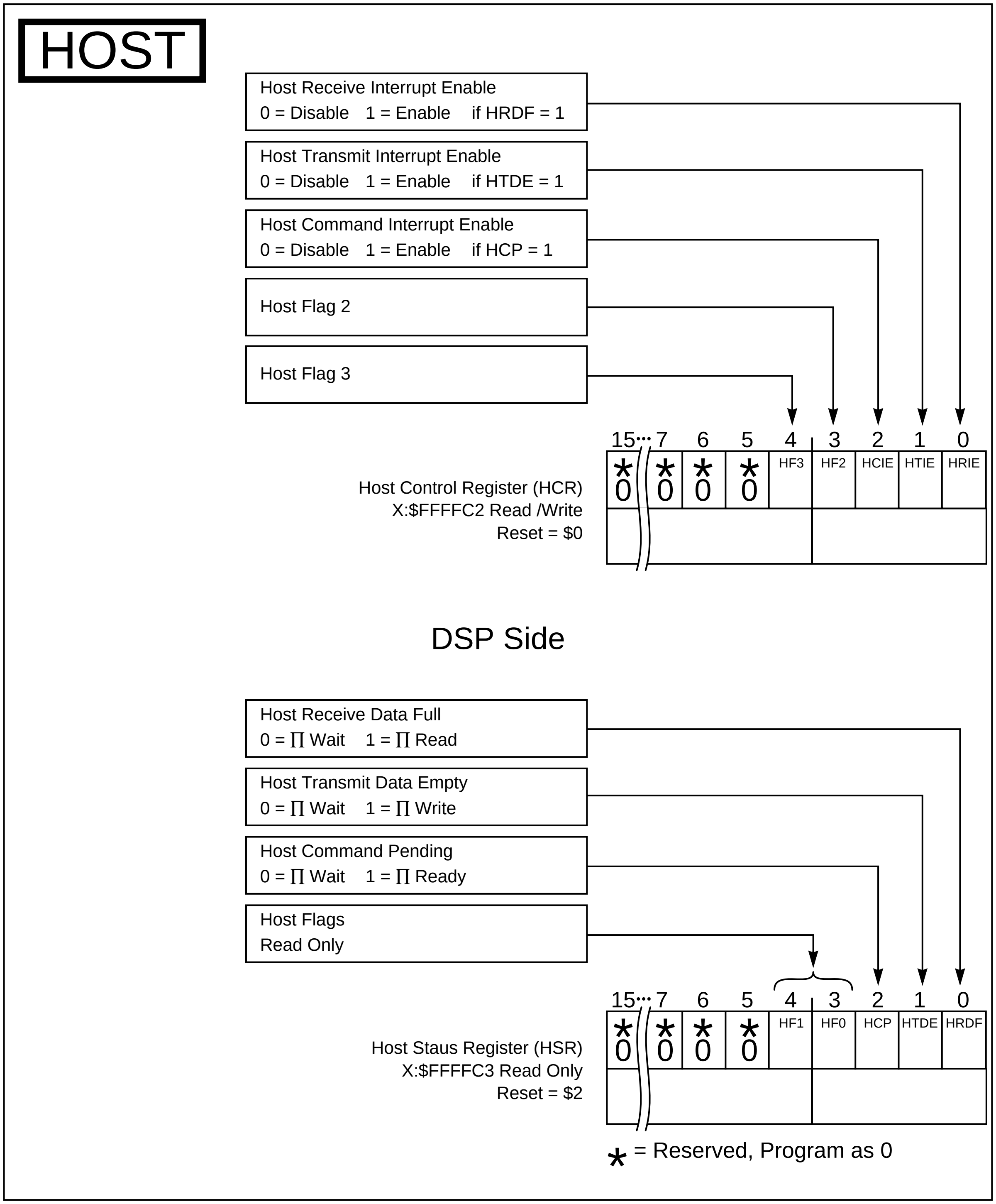
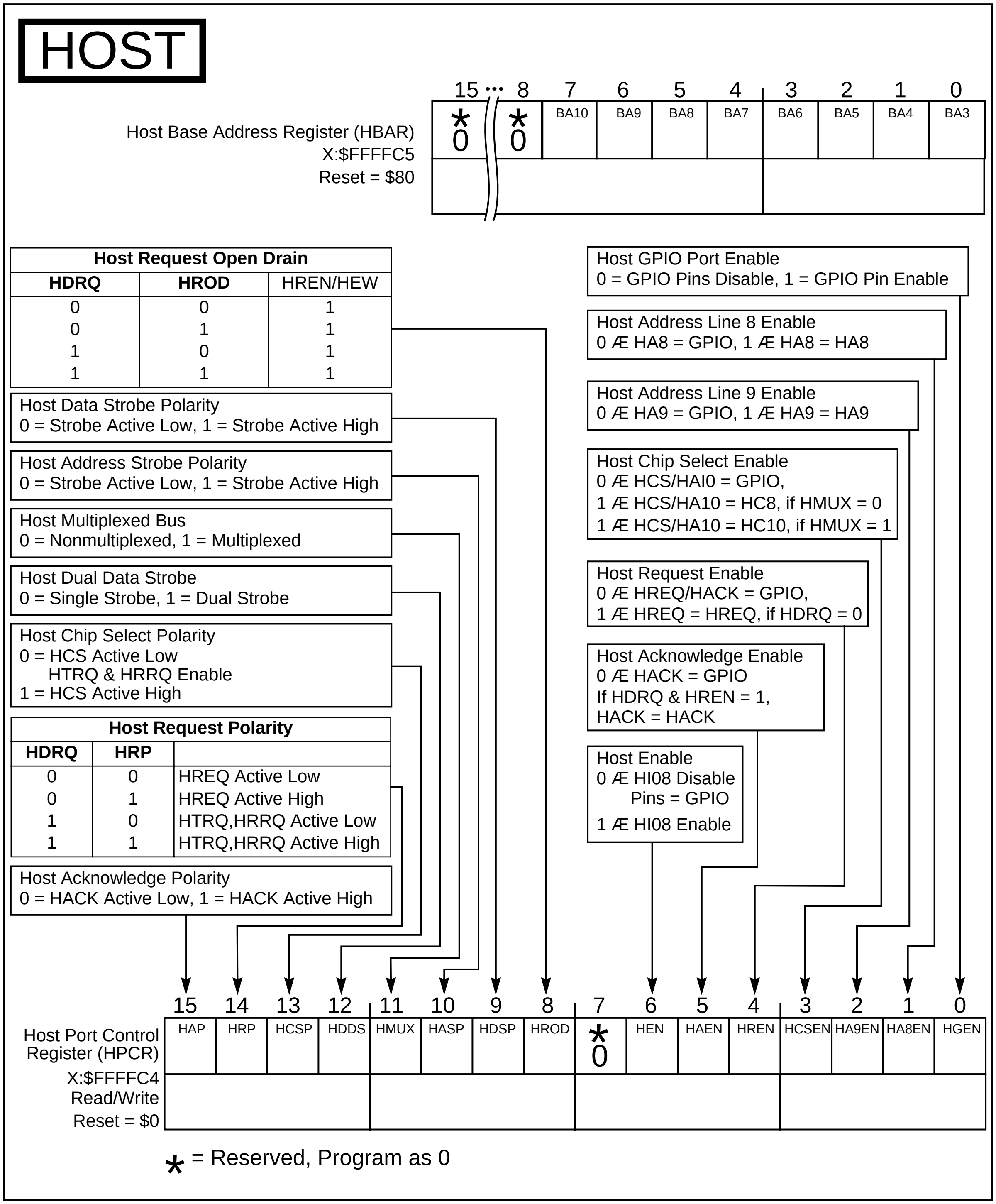

HOST
Host Receive Interrupt Enable
0 = Disable 1 = Enable if HRDF = 1
Host Transmit Interrupt Enable
0 = Disable 1 = Enable if HTDE = 1
Host Command Interrupt Enable
0 = Disable 1 = Enable if HCP = 1
Host Flag 2
Host Flag 3
15··· 7 6 5 4 3 2 1 0
* 0 * 0 * 0 * 0 HF3 HF2 HCIE HTIE HRIE
Host Control Register (HCR)
X:$FFFFC2 Read /Write
Reset = $0
DSP Side
Host Receive Data Full
0 = ∏ Wait 1 = ∏ Read
Host Transmit Data Empty
0 = ∏ Wait 1 = ∏ Write
Host Command Pending
0 = ∏ Wait 1 = ∏ Ready
Host Flags
Read Only
15··· 7 6 5 4 3 2 1 0
* 0 * 0 * 0 * 0 HF1 HF0 HCP HTDE HRDF
Host Staus Register (HSR)
X:$FFFFC3 Read Only
Reset = $2
* = Reserved, Program as 0

Figure E-6 Host Control and Host Status Registers


HOST
Host Base Address Register (HBAR)
X:$FFFFC5
Reset = $80
Host GPIO Port Enable
0 = GPIO Pins Disable, 1 = GPIO Pin Enable
Host Address Line 8 Enable
0 Æ HA8 = GPIO, 1 Æ HA8 = HA8
Host Address Line 9 Enable
0 Æ HA9 = GPIO, 1 Æ HA9 = HA9
Host Chip Select Enable
0 Æ HCS/HAI0 = GPIO,
1 Æ HCS/HA10 = HC8, if HMUX = 0
1 Æ HCS/HA10 = HC10, if HMUX = 1
Host Request Enable
0 Æ HREQ/HACK = GPIO,
1 Æ HREQ = HREQ, if HDRQ = 0
Host Acknowledge Enable
0 Æ HACK = GPIO
If HDRQ & HREN = 1,
HACK = HACK
Host Enable
0 Æ HI08 Disable
Pins = GPIO
1 Æ HI08 Enable
Host Data Strobe Polarity
0 = Strobe Active Low, 1 = Strobe Active High
Host Address Strobe Polarity
0 = Strobe Active Low, 1 = Strobe Active High
Host Multiplexed Bus
0 = Nonmultiplexed, 1 = Multiplexed
Host Dual Data Strobe
0 = Single Strobe, 1 = Dual Strobe
Host Chip Select Polarity
0 = HCS Active Low
HTRQ & HRRQ Enable
1 = HCS Active High
Host Acknowledge Polarity
0 = HACK Active Low, 1 = HACK Active High
Host Port Control
Register (HPCR)
X:$FFFFC4
Read/Write
Reset = $0
* = Reserved, Program as 0

15 ···	8	7	6	5	4	3	2	1	0
* 0	* 0	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3

Host Request Open Drain		
HDRQ	**HROD**	HREN/HEW
0	0	1
0	1	1
1	0	1
1	1	1

Host Request Polarity		
HDRQ	**HRP**	
0	0	HREQ Active Low
0	1	HREQ Active High
1	0	HTRQ,HRRQ Active Low
1	1	HTRQ,HRRQ Active High

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

Figure E-7 Host Base Address and Host Port Control Registers

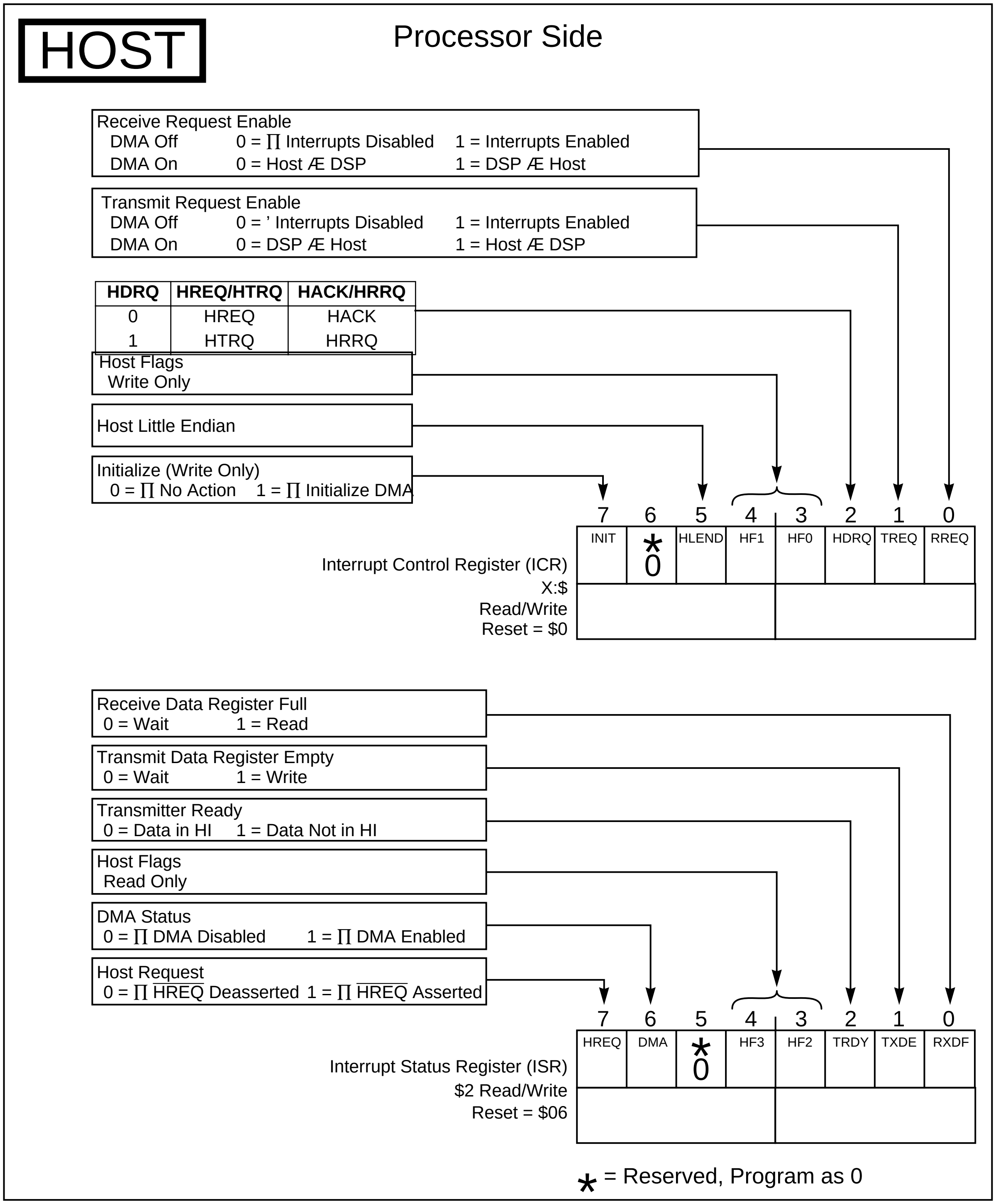

HOST
Processor Side
Receive Request Enable
DMA Off 0 = ∏ Interrupts Disabled 1 = Interrupts Enabled
DMA On 0 = Host Æ DSP 1 = DSP Æ Host
Transmit Request Enable
DMA Off 0 = ’ Interrupts Disabled 1 = Interrupts Enabled
DMA On 0 = DSP Æ Host 1 = Host Æ DSP
HDRQ HREQ/HTRQ HACK/HRRQ
0 HREQ HACK
1 HTRQ HRRQ
Host Flags
Write Only
Host Little Endian
Initialize (Write Only)
0 = ∏ No Action 1 = ∏ Initialize DMA
7 6 5 4 3 2 1 0
INIT * 0 HLEND HF1 HF0 HDRQ TREQ RREQ
Interrupt Control Register (ICR)
X:$
Read/Write
Reset = $0
Receive Data Register Full
0 = Wait 1 = Read
Transmit Data Register Empty
0 = Wait 1 = Write
Transmitter Ready
0 = Data in HI 1 = Data Not in HI
Host Flags
Read Only
DMA Status
0 = ∏ DMA Disabled 1 = ∏ DMA Enabled
Host Request
0 = ∏ HREQ Deasserted 1 = ∏ HREQ Asserted
7 6 5 4 3 2 1 0
HREQ DMA * 0 HF3 HF2 TRDY TXDE RXDF
Interrupt Status Register (ISR)
$2 Read/Write
Reset = $06
* = Reserved, Program as 0

Figure E-8 Interrupt Control and Interrupt Status Registers

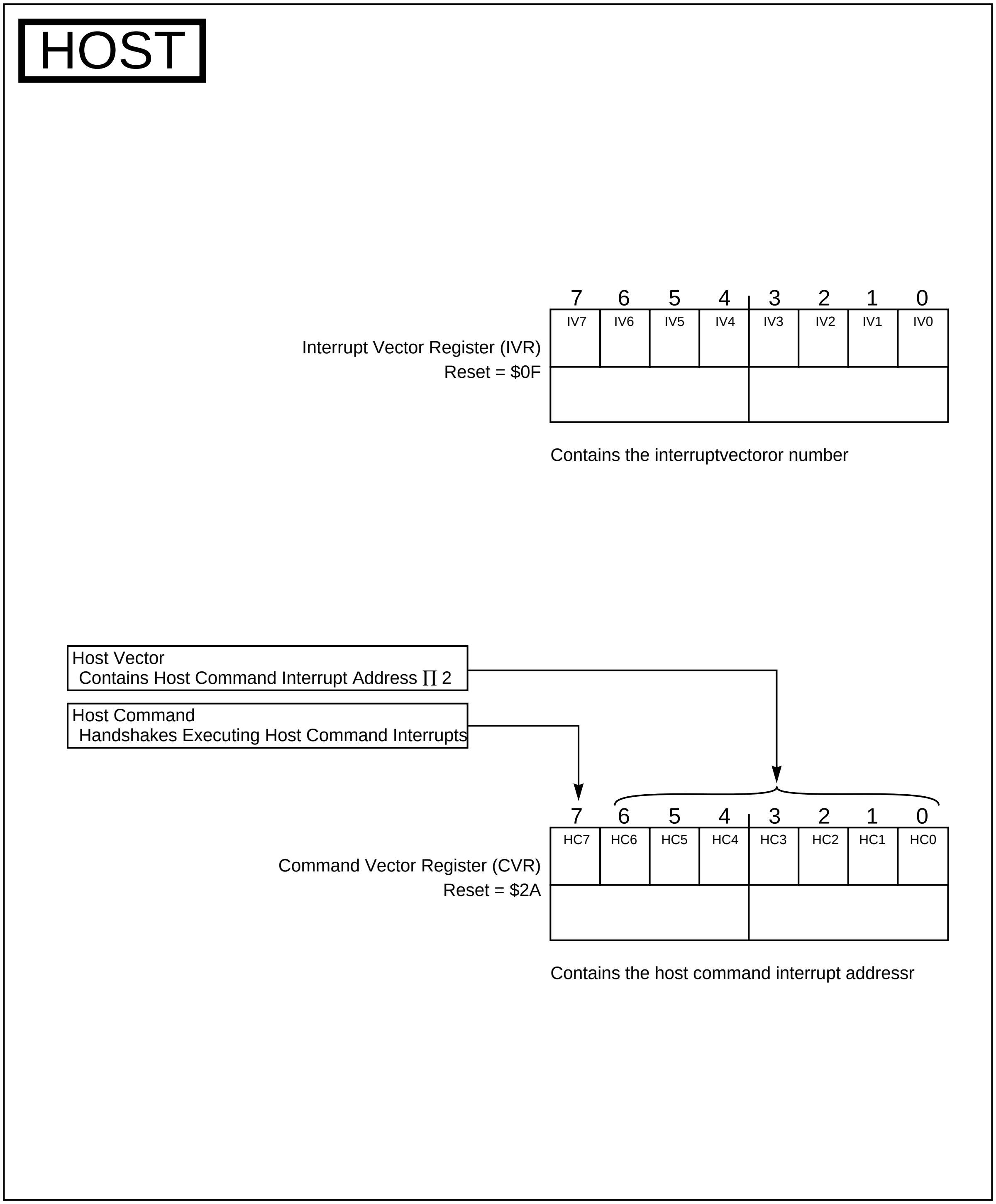

HOST
7 6 5 4 3 2 1 0
IV7 IV6 IV5 IV4 IV3 IV2 IV1 IV0
Interrupt Vector Register (IVR)
Reset = $0F
Contains the interruptvectoror number
Host Vector
Contains Host Command Interrupt Address ∏ 2
Host Command
Handshakes Executing Host Command Interrupts
7 6 5 4 3 2 1 0
HC7 HC6 HC5 HC4 HC3 HC2 HC1 HC0
Command Vector Register (CVR)
Reset = $2A
Contains the host command interrupt addressr

Figure E-9 Interrupt Vector and Command Vector Registers

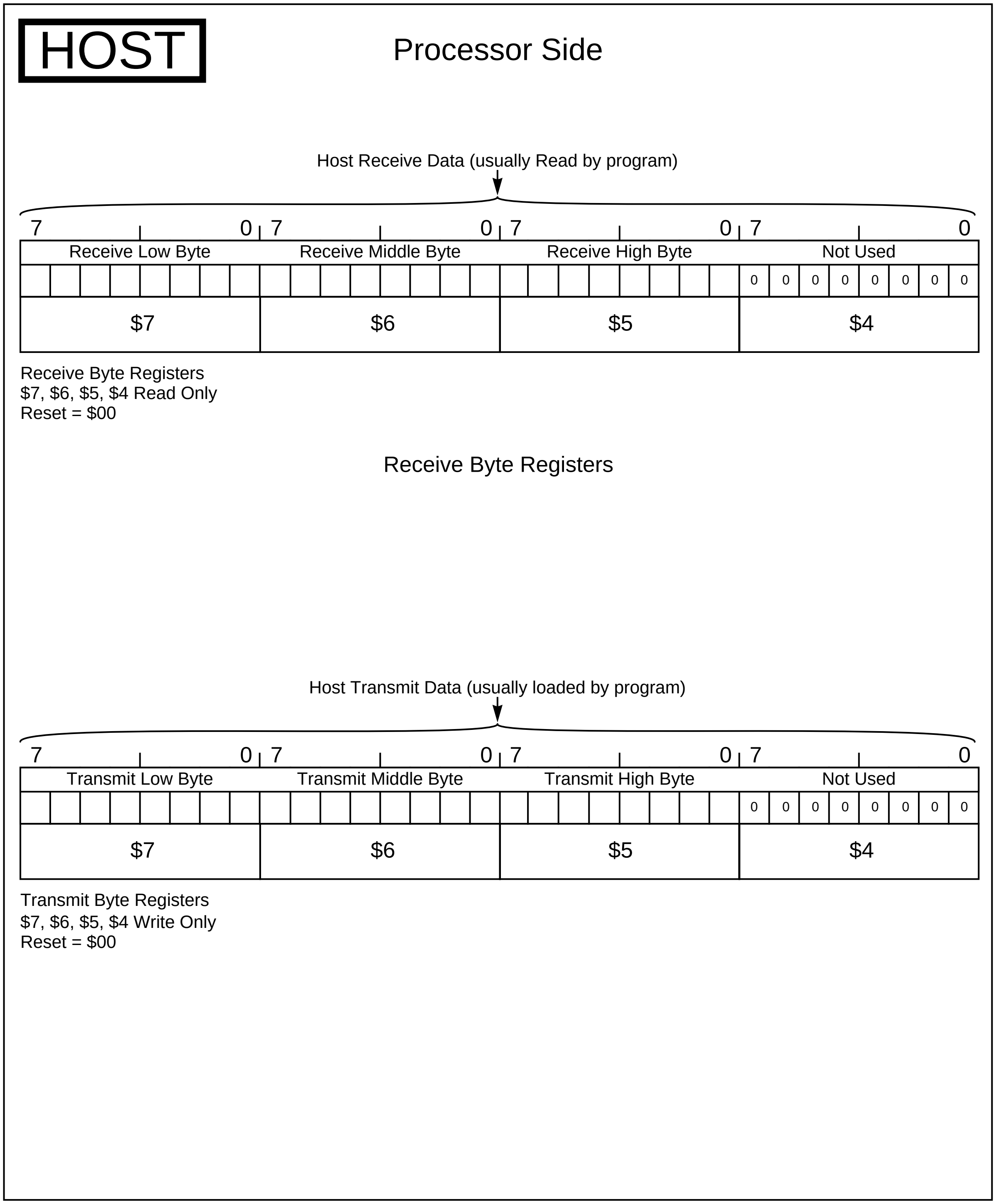
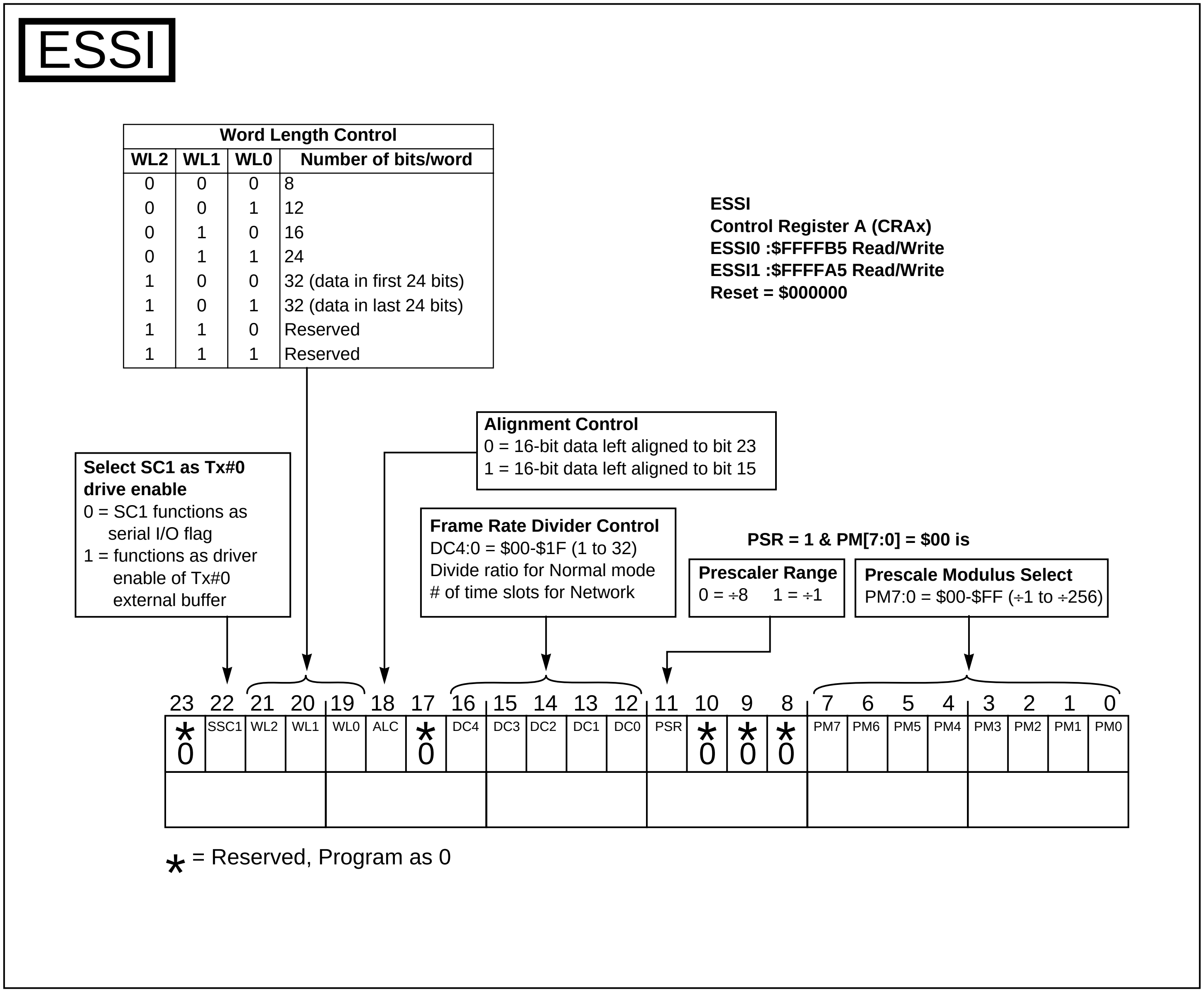


Figure E-10 Host Receive and Host Transmit Data Registers



ESSI

Word Length Control

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 (data in first 24 bits)
1	0	1	32 (data in last 24 bits)
1	1	0	Reserved
1	1	1	Reserved

ESSI
Control Register A (CRAx)
ESSI0 :$FFFFB5 Read/Write
ESSI1 :$FFFFA5 Read/Write
Reset = $000000

Alignment Control
0 = 16-bit data left aligned to bit 23
1 = 16-bit data left aligned to bit 15

Select SC1 as Tx#0 drive enable
0 = SC1 functions as serial I/O flag
1 = functions as driver enable of Tx#0 external buffer

Frame Rate Divider Control
DC4:0 = $00-$1F (1 to 32)
Divide ratio for Normal mode
# of time slots for Network

PSR = 1 & PM[7:0] = $00 is

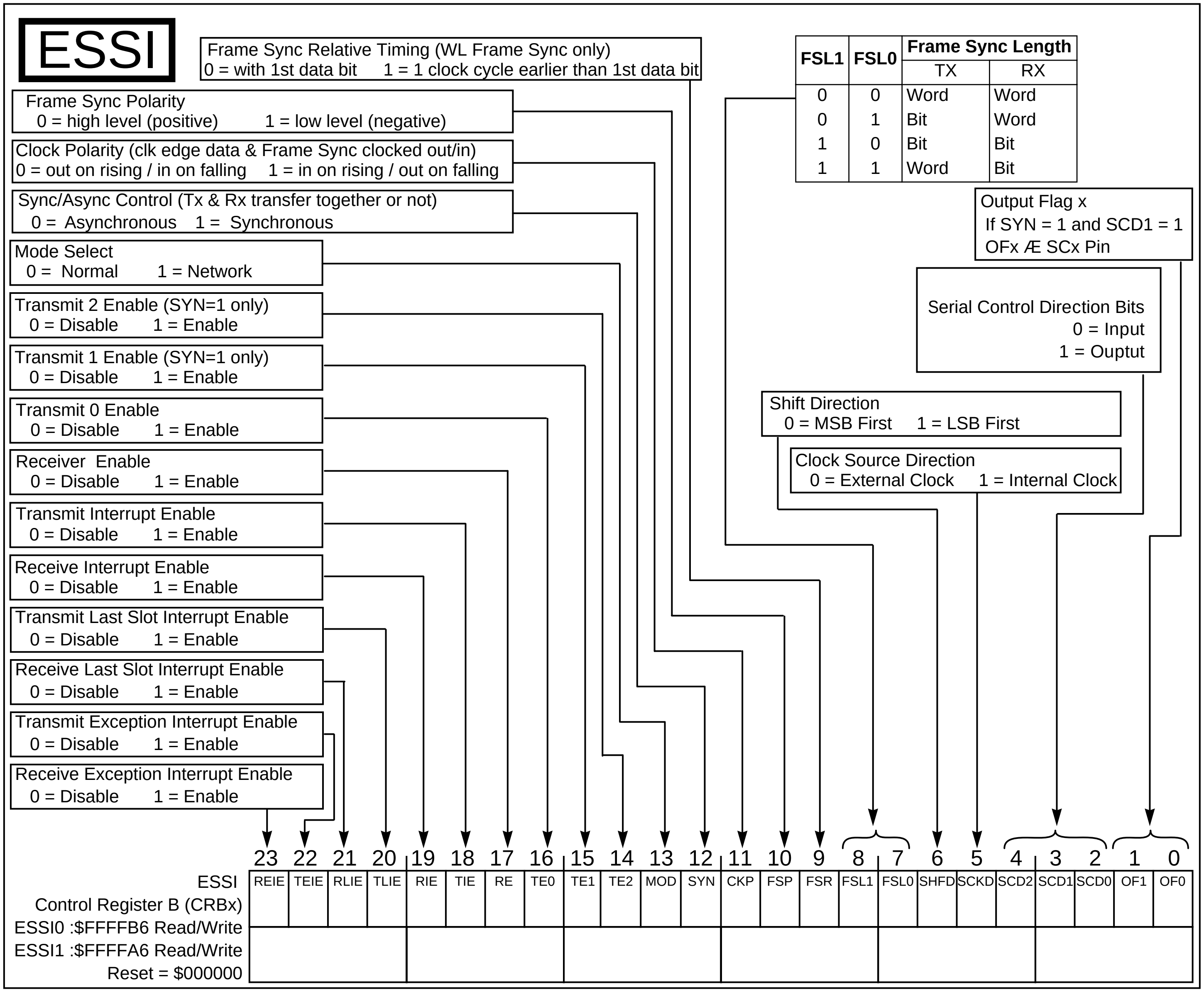
Prescaler Range
0 = ÷8 1 = ÷1

Prescale Modulus Select
PM7:0 = $00-$FF (÷1 to ÷256)

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	SSC1	WL2	WL1	WL0	ALC	* 0	DC4	DC3	DC2	DC1	DC0	PSR	* 0	* 0	* 0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

* = Reserved, Program as 0

Figure E-11 ESSI Control Register A (CRA)



ESSI

Frame Sync Relative Timing (WL Frame Sync only)
0 = with 1st data bit 1 = 1 clock cycle earlier than 1st data bit

Frame Sync Polarity
0 = high level (positive) 1 = low level (negative)

Clock Polarity (clk edge data & Frame Sync clocked out/in)
0 = out on rising / in on falling 1 = in on rising / out on falling

Sync/Async Control (Tx & Rx transfer together or not)
0 = Asynchronous 1 = Synchronous

Mode Select
0 = Normal 1 = Network

Transmit 2 Enable (SYN=1 only)
0 = Disable 1 = Enable

Transmit 1 Enable (SYN=1 only)
0 = Disable 1 = Enable

Transmit 0 Enable
0 = Disable 1 = Enable

Receiver Enable
0 = Disable 1 = Enable

Transmit Interrupt Enable
0 = Disable 1 = Enable

Receive Interrupt Enable
0 = Disable 1 = Enable

Transmit Last Slot Interrupt Enable
0 = Disable 1 = Enable

Receive Last Slot Interrupt Enable
0 = Disable 1 = Enable

Transmit Exception Interrupt Enable
0 = Disable 1 = Enable

Receive Exception Interrupt Enable
0 = Disable 1 = Enable

FSL1	FSL0	Frame Sync Length	
		TX	RX
0	0	Word	Word
0	1	Bit	Word
1	0	Bit	Bit
1	1	Word	Bit

Output Flag x
If SYN = 1 and SCD1 = 1
OFx Æ SCx Pin

Serial Control Direction Bits
0 = Input
1 = Ouptut

Shift Direction
0 = MSB First 1 = LSB First

Clock Source Direction
0 = External Clock 1 = Internal Clock

ESSI
Control Register B (CRBx)
ESSI0 :$FFFFB6 Read/Write
ESSI1 :$FFFFA6 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
REIE	TEIE	RLIE	TLIE	RIE	TIE	RE	TE0	TE1	TE2	MOD	SYN	CKP	FSP	FSR	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0

Figure E-12 ESSI Control Register B (CRB)

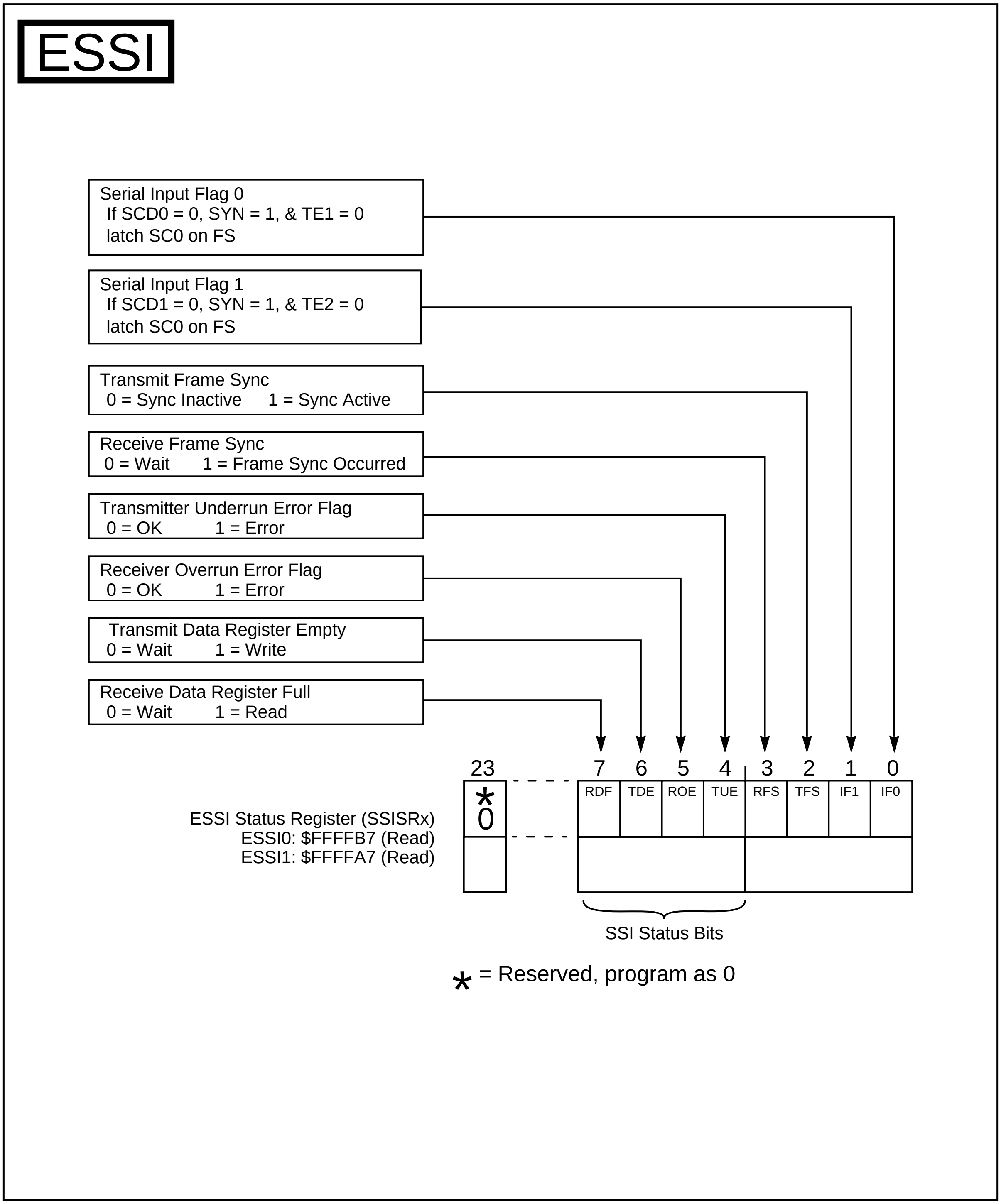

ESSI
Serial Input Flag 0
If SCD0 = 0, SYN = 1, & TE1 = 0
latch SC0 on FS
Serial Input Flag 1
If SCD1 = 0, SYN = 1, & TE2 = 0
latch SC0 on FS
Transmit Frame Sync
0 = Sync Inactive 1 = Sync Active
Receive Frame Sync
0 = Wait 1 = Frame Sync Occurred
Transmitter Underrun Error Flag
0 = OK 1 = Error
Receiver Overrun Error Flag
0 = OK 1 = Error
Transmit Data Register Empty
0 = Wait 1 = Write
Receive Data Register Full
0 = Wait 1 = Read
23
*
0
7
6
5
4
3
2
1
0
RDF
TDE
ROE
TUE
RFS
TFS
IF1
IF0
ESSI Status Register (SSISRx)
ESSI0: $FFFFB7 (Read)
ESSI1: $FFFFA7 (Read)
SSI Status Bits
* = Reserved, program as 0

Figure E-13 ESSI Status Register (SSISR)

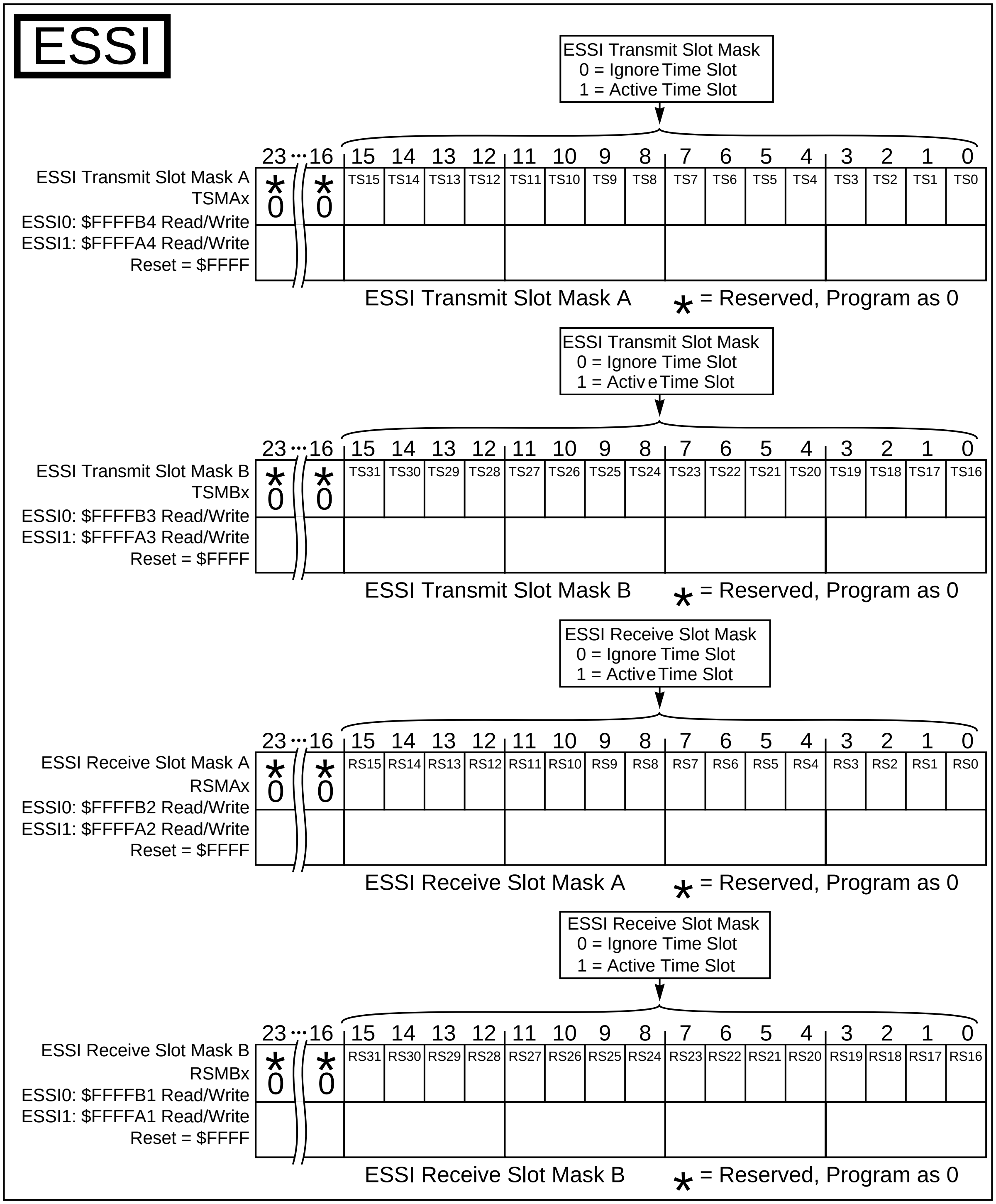
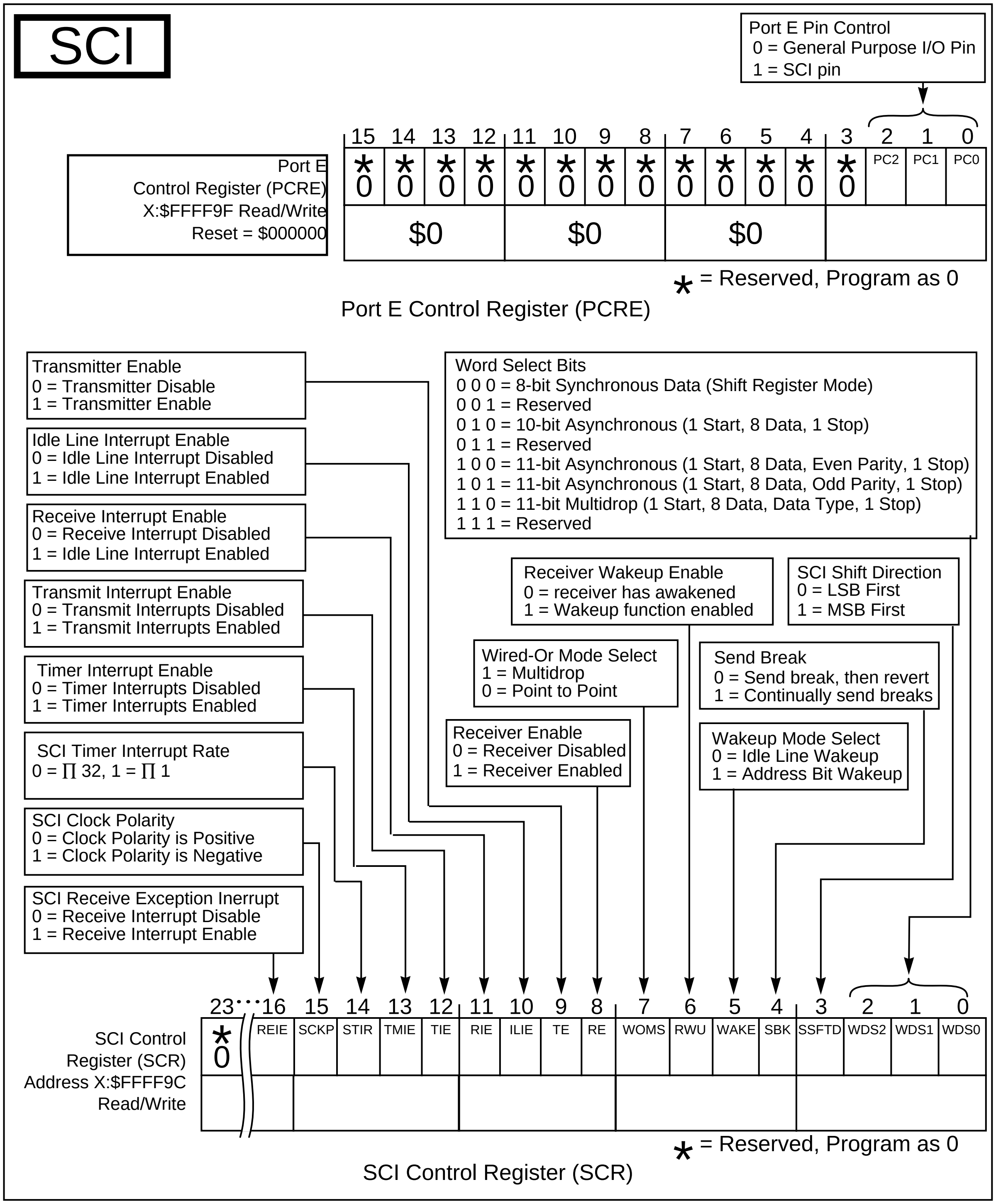

ESSI
ESSI Transmit Slot Mask
0 = Ignore Time Slot
1 = Active Time Slot
23 ··· 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ESSI Transmit Slot Mask A
TSMAx
ESSI0: $FFFFB4 Read/Write
ESSI1: $FFFFA4 Read/Write
Reset = $FFFF
* 0 * 0 TS15 TS14 TS13 TS12 TS11 TS10 TS9 TS8 TS7 TS6 TS5 TS4 TS3 TS2 TS1 TS0
ESSI Transmit Slot Mask A
* = Reserved, Program as 0
ESSI Transmit Slot Mask
0 = Ignore Time Slot
1 = Active Time Slot
23 ··· 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ESSI Transmit Slot Mask B
TSMBx
ESSI0: $FFFFB3 Read/Write
ESSI1: $FFFFA3 Read/Write
Reset = $FFFF
* 0 * 0 TS31 TS30 TS29 TS28 TS27 TS26 TS25 TS24 TS23 TS22 TS21 TS20 TS19 TS18 TS17 TS16
ESSI Transmit Slot Mask B
* = Reserved, Program as 0
ESSI Receive Slot Mask
0 = Ignore Time Slot
1 = Active Time Slot
23 ··· 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ESSI Receive Slot Mask A
RSMAx
ESSI0: $FFFFB2 Read/Write
ESSI1: $FFFFA2 Read/Write
Reset = $FFFF
* 0 * 0 RS15 RS14 RS13 RS12 RS11 RS10 RS9 RS8 RS7 RS6 RS5 RS4 RS3 RS2 RS1 RS0
ESSI Receive Slot Mask A
* = Reserved, Program as 0
ESSI Receive Slot Mask
0 = Ignore Time Slot
1 = Active Time Slot
23 ··· 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ESSI Receive Slot Mask B
RSMBx
ESSI0: $FFFFB1 Read/Write
ESSI1: $FFFFA1 Read/Write
Reset = $FFFF
* 0 * 0 RS31 RS30 RS29 RS28 RS27 RS26 RS25 RS24 RS23 RS22 RS21 RS20 RS19 RS18 RS17 RS16
ESSI Receive Slot Mask B
* = Reserved, Program as 0

Figure E-14 ESSR Transmit and Receive Slot Mask Registers (TSM, RSM)



Port E Pin Control
0 = General Purpose I/O Pin
1 = SCI pin

Port E
Control Register (PCRE)
X:$FFFF9F Read/Write
Reset = $000000

* = Reserved, Program as 0

Port E Control Register (PCRE)

Transmitter Enable
0 = Transmitter Disable
1 = Transmitter Enable

Idle Line Interrupt Enable
0 = Idle Line Interrupt Disabled
1 = Idle Line Interrupt Enabled

Receive Interrupt Enable
0 = Receive Interrupt Disabled
1 = Idle Line Interrupt Enabled

Transmit Interrupt Enable
0 = Transmit Interrupts Disabled
1 = Transmit Interrupts Enabled

Timer Interrupt Enable
0 = Timer Interrupts Disabled
1 = Timer Interrupts Enabled

SCI Timer Interrupt Rate
0 = ∏ 32, 1 = ∏ 1

SCI Clock Polarity
0 = Clock Polarity is Positive
1 = Clock Polarity is Negative

SCI Receive Exception Inerrupt
0 = Receive Interrupt Disable
1 = Receive Interrupt Enable

Word Select Bits
0 0 0 = 8-bit Synchronous Data (Shift Register Mode)
0 0 1 = Reserved
0 1 0 = 10-bit Asynchronous (1 Start, 8 Data, 1 Stop)
0 1 1 = Reserved
1 0 0 = 11-bit Asynchronous (1 Start, 8 Data, Even Parity, 1 Stop)
1 0 1 = 11-bit Asynchronous (1 Start, 8 Data, Odd Parity, 1 Stop)
1 1 0 = 11-bit Multidrop (1 Start, 8 Data, Data Type, 1 Stop)
1 1 1 = Reserved

Receiver Wakeup Enable
0 = receiver has awakened
1 = Wakeup function enabled

SCI Shift Direction
0 = LSB First
1 = MSB First

Wired-Or Mode Select
1 = Multidrop
0 = Point to Point

Send Break
0 = Send break, then revert
1 = Continually send breaks

Receiver Enable
0 = Receiver Disabled
1 = Receiver Enabled

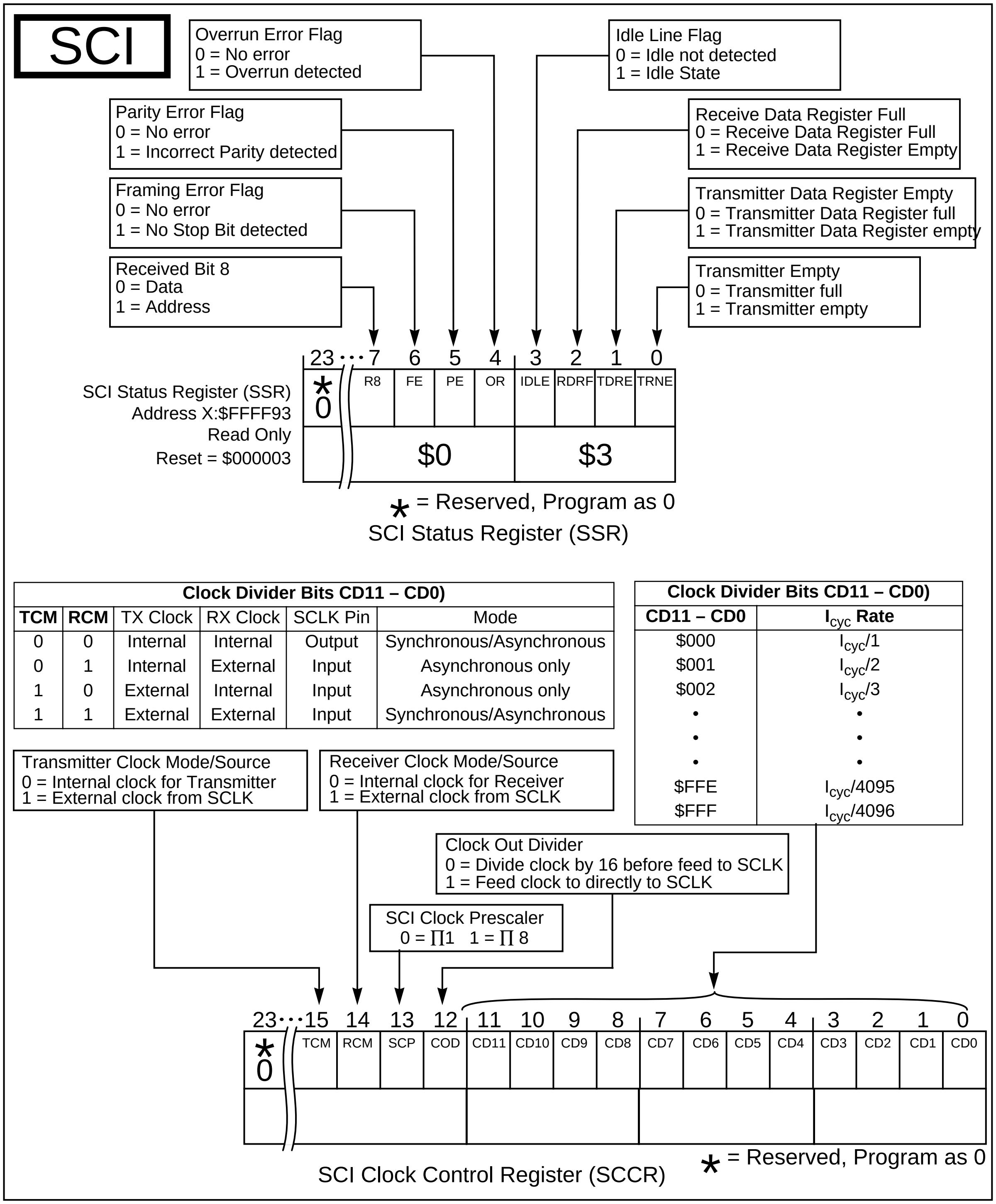
Wakeup Mode Select
0 = Idle Line Wakeup
1 = Address Bit Wakeup

SCI Control
Register (SCR)
Address X:$FFFF9C
Read/Write

* = Reserved, Program as 0

SCI Control Register (SCR)

Figure E-15 SCI Control Register (SCR)


SCI
Overrun Error Flag
0 = No error
1 = Overrun detected
Idle Line Flag
0 = Idle not detected
1 = Idle State
Parity Error Flag
0 = No error
1 = Incorrect Parity detected
Receive Data Register Full
0 = Receive Data Register Full
1 = Receive Data Register Empty
Framing Error Flag
0 = No error
1 = No Stop Bit detected
Transmitter Data Register Empty
0 = Transmitter Data Register full
1 = Transmitter Data Register empty
Received Bit 8
0 = Data
1 = Address
Transmitter Empty
0 = Transmitter full
1 = Transmitter empty
23 · · · 7 6 5 4 3 2 1 0
SCI Status Register (SSR)
Address X:$FFFF93
Read Only
Reset = $000003
* 0
R8 FE PE OR IDLE RDRF TDRE TRNE
$0
$3
* = Reserved, Program as 0
SCI Status Register (SSR)
Transmitter Clock Mode/Source
0 = Internal clock for Transmitter
1 = External clock from SCLK
Receiver Clock Mode/Source
0 = Internal clock for Receiver
1 = External clock from SCLK
Clock Out Divider
0 = Divide clock by 16 before feed to SCLK
1 = Feed clock to directly to SCLK
SCI Clock Prescaler
0 = ∏1 1 = ∏ 8
23 · · · 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
* 0
TCM RCM SCP COD CD11 CD10 CD9 CD8 CD7 CD6 CD5 CD4 CD3 CD2 CD1 CD0
* = Reserved, Program as 0
SCI Clock Control Register (SCCR)

Clock Divider Bits CD11 – CD0)					
TCM	**RCM**	TX Clock	RX Clock	SCLK Pin	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

Clock Divider Bits CD11 – CD0)	
CD11 – CD0	**I_{cyc} Rate**
$000	$I_{cyc}/1$
$001	$I_{cyc}/2$
$002	$I_{cyc}/3$
•	•
•	•
•	•
$FFE	$I_{cyc}/4095$
$FFF	$I_{cyc}/4096$

Figure E-16 SCI Status and Clock Control Registers (SSR, SCCR)

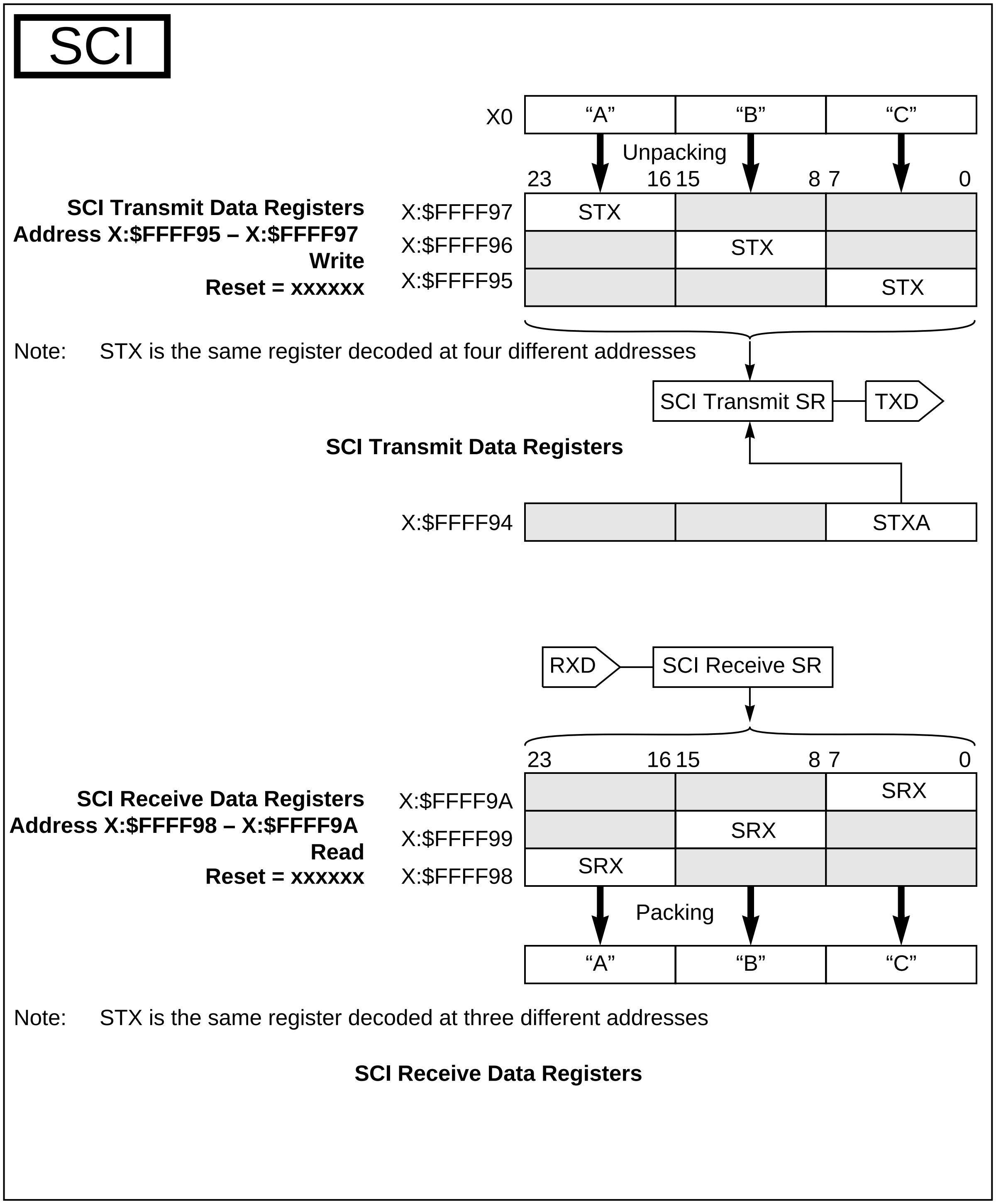
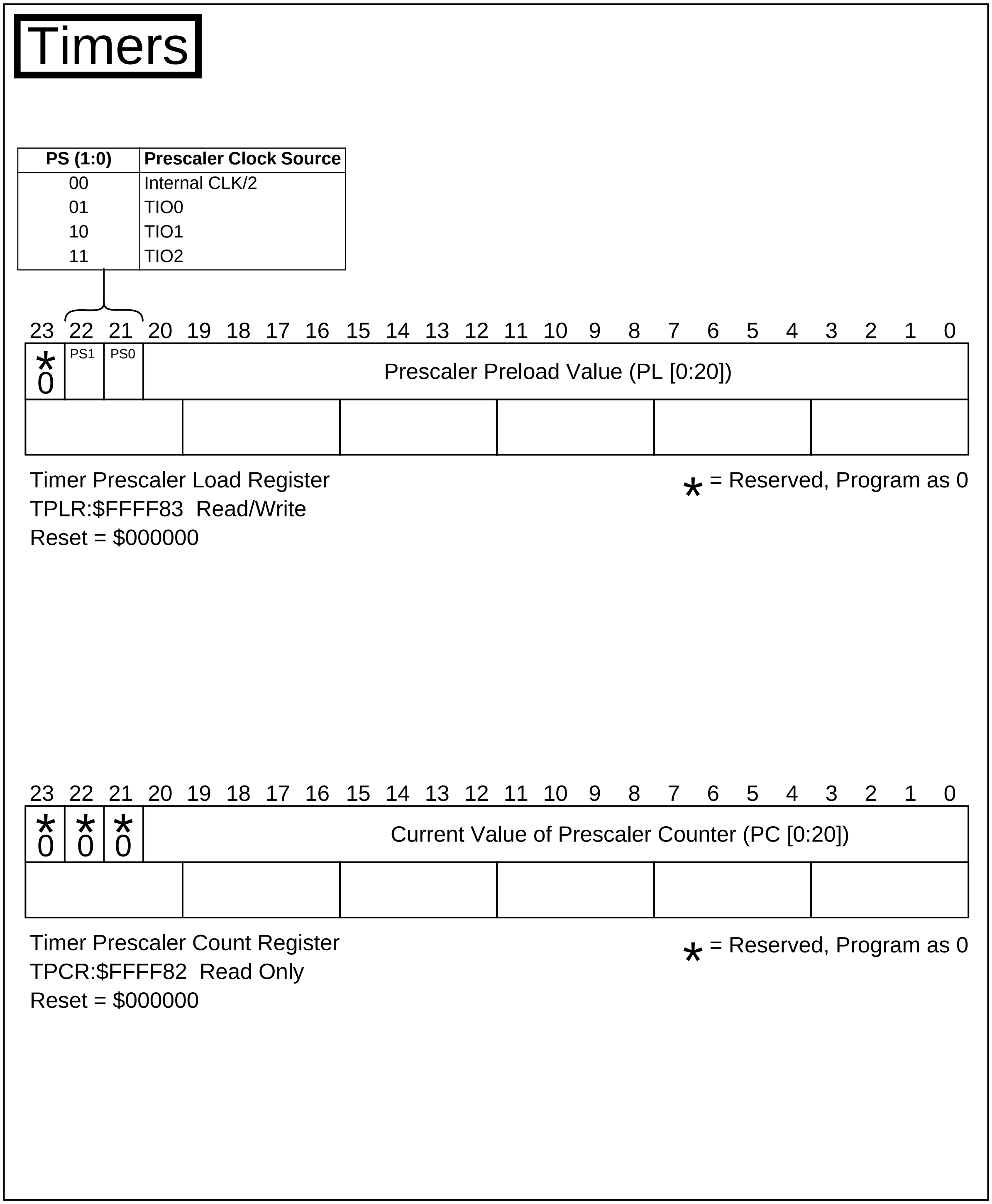

SCI
X0
“A”
“B”
“C”
Unpacking
23
16 15
8 7
0
SCI Transmit Data Registers
Address X:$FFFF95 – X:$FFFF97
Write
Reset = xxxxxx
X:$FFFF97
X:$FFFF96
X:$FFFF95
STX
STX
STX
Note: STX is the same register decoded at four different addresses
SCI Transmit SR
TXD
SCI Transmit Data Registers
X:$FFFF94
STXA
RXD
SCI Receive SR
23
16 15
8 7
0
SCI Receive Data Registers
Address X:$FFFF98 – X:$FFFF9A
Read
Reset = xxxxxx
X:$FFFF9A
X:$FFFF99
X:$FFFF98
SRX
SRX
SRX
Packing
“A”
“B”
“C”
Note: STX is the same register decoded at three different addresses
SCI Receive Data Registers

Figure E-17 SCI Receive and Transmit Data Registers (SRX, TRX)



PS (1:0)	Prescaler Clock Source
00	Internal CLK/2
01	TIO0
10	TIO1
11	TIO2

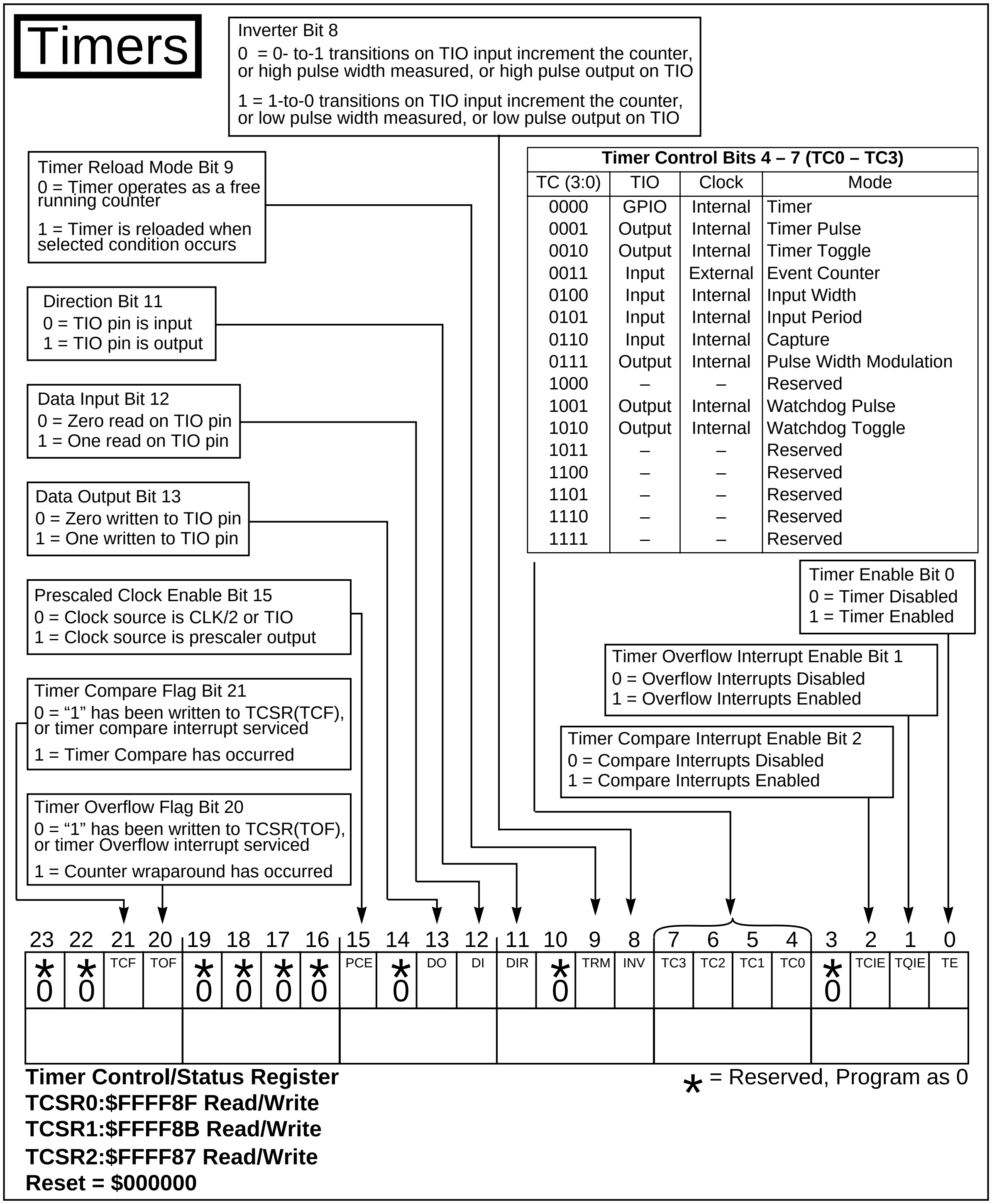
Timer Prescaler Load Register
TPLR:$FFFF83 Read/Write
Reset = $000000

* = Reserved, Program as 0

Timer Prescaler Count Register
TPCR:$FFFF82 Read Only
Reset = $000000

* = Reserved, Program as 0

Figure E-18 Timer Prescaler Load/Count Register (TPLR, TPCR)



Timers

Inverter Bit 8
0 = 0- to-1 transitions on TIO input increment the counter, or high pulse width measured, or high pulse output on TIO
1 = 1-to-0 transitions on TIO input increment the counter, or low pulse width measured, or low pulse output on TIO

Timer Reload Mode Bit 9
0 = Timer operates as a free running counter
1 = Timer is reloaded when selected condition occurs

Direction Bit 11
0 = TIO pin is input
1 = TIO pin is output

Data Input Bit 12
0 = Zero read on TIO pin
1 = One read on TIO pin

Data Output Bit 13
0 = Zero written to TIO pin
1 = One written to TIO pin

Prescaled Clock Enable Bit 15
0 = Clock source is CLK/2 or TIO
1 = Clock source is prescaler output

Timer Compare Flag Bit 21
0 = “1” has been written to TCSR(TCF), or timer compare interrupt serviced
1 = Timer Compare has occurred

Timer Overflow Flag Bit 20
0 = “1” has been written to TCSR(TOF), or timer Overflow interrupt serviced
1 = Counter wraparound has occurred

Timer Control Bits 4 – 7 (TC0 – TC3)

TC (3:0)	TIO	Clock	Mode
0000	GPIO	Internal	Timer
0001	Output	Internal	Timer Pulse
0010	Output	Internal	Timer Toggle
0011	Input	External	Event Counter
0100	Input	Internal	Input Width
0101	Input	Internal	Input Period
0110	Input	Internal	Capture
0111	Output	Internal	Pulse Width Modulation
1000	–	–	Reserved
1001	Output	Internal	Watchdog Pulse
1010	Output	Internal	Watchdog Toggle
1011	–	–	Reserved
1100	–	–	Reserved
1101	–	–	Reserved
1110	–	–	Reserved
1111	–	–	Reserved

Timer Enable Bit 0
0 = Timer Disabled
1 = Timer Enabled

Timer Overflow Interrupt Enable Bit 1
0 = Overflow Interrupts Disabled
1 = Overflow Interrupts Enabled

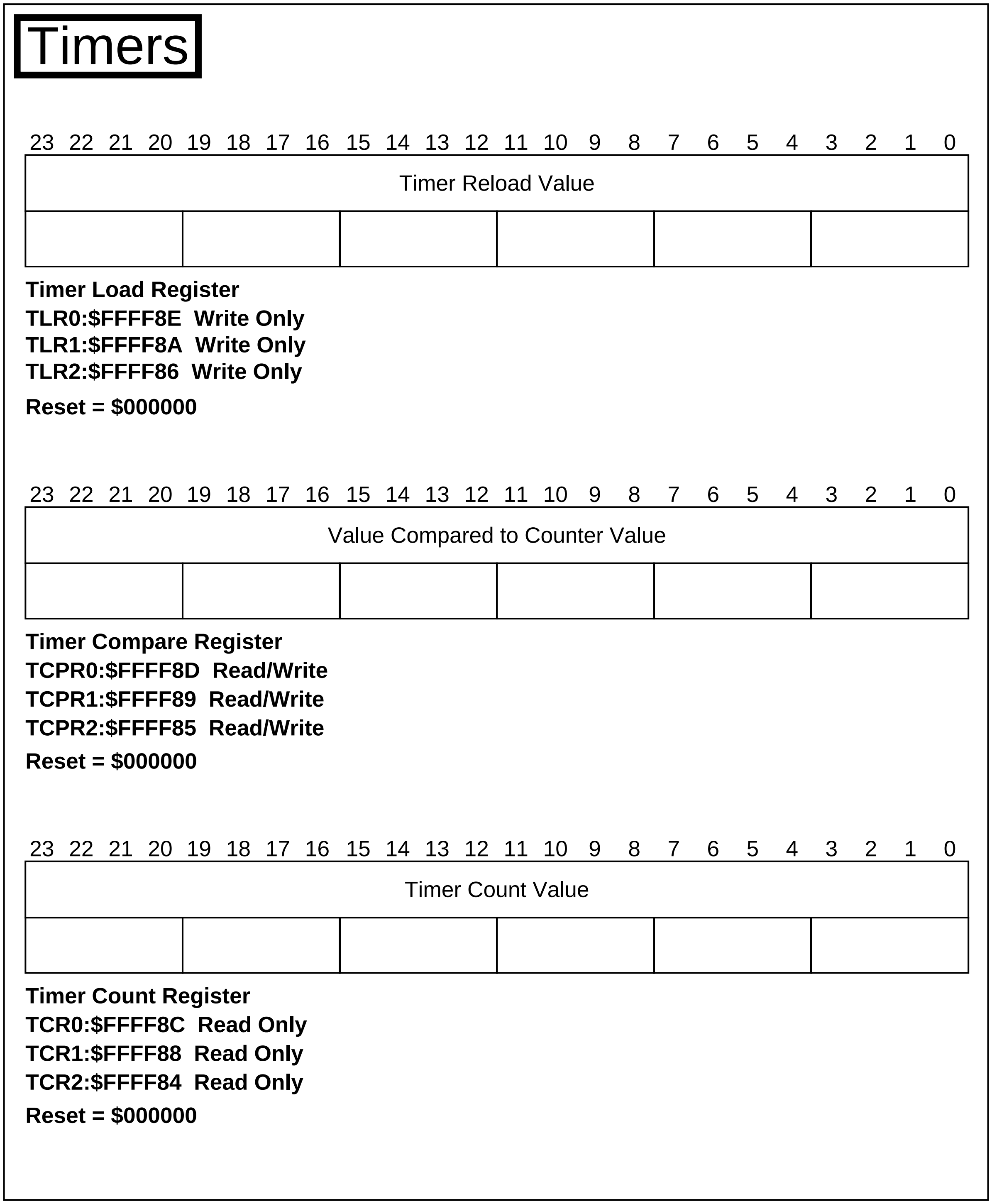
Timer Compare Interrupt Enable Bit 2
0 = Compare Interrupts Disabled
1 = Compare Interrupts Enabled

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	TCF	TOF	* 0	* 0	* 0	* 0	PCE	* 0	DO	DI	DIR	* 0	TRM	INV	TC3	TC2	TC1	TC0	* 0	TCIE	TQIE	TE

Timer Control/Status Register
TCSR0:$FFFF8F Read/Write
TCSR1:$FFFF8B Read/Write
TCSR2:$FFFF87 Read/Write
Reset = $000000

* = Reserved, Program as 0

Figure E-19 Timer Control/Status Register (TCSR)



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 E-20 Timer Load, Compare, Count Registers (TLR, TCPR, TCR)

GPIO

Port B (HI08)

Host Data Direction Register (HDDR)
X:$FFFFC8
Write
Reset = $0

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

DRx = 1 Æ HIx is Output   DRx = 0 Æ HIx is Input

Host Data Register (HDR)
X:$FFFFC9
Write
Reset = Undefined

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

DRx holds value of corresponding HI08 GPIO pin.
Function depends on HDDR.

Figure E-21 Host Data Direction and Host Data Registers (HDDR, HDR)

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 Æ Port Pin configured as ESSI
PCn = 0 Æ 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 Æ Port Pin is Output
PDCn = 0 Æ Port Pin is Input

Port C GPIO Data Register
(PDRC)
X:$FFFFBD
ReadWrite
Reset = $0

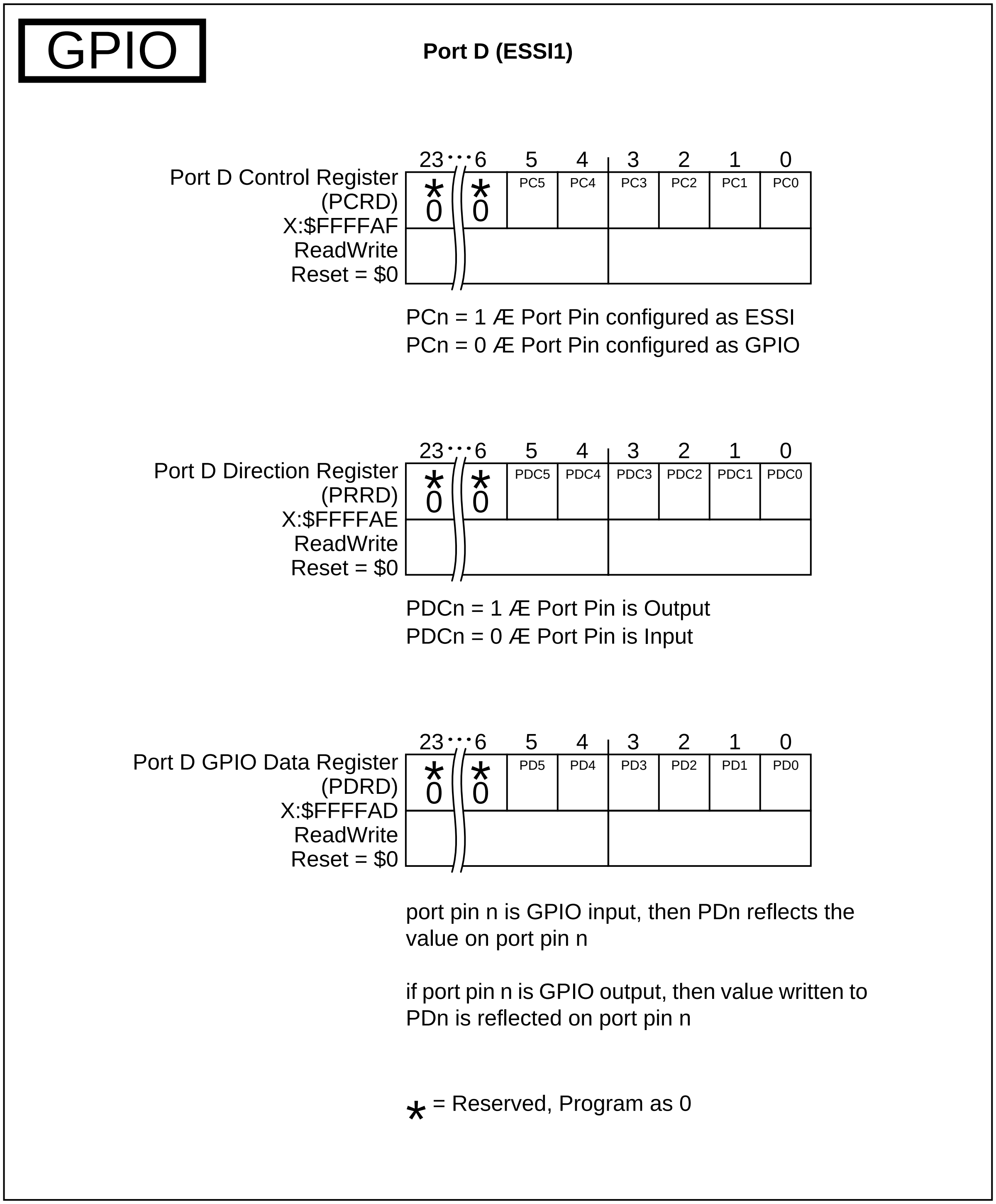
23···	···6	5	4	3	2	1	0
* 0	* 0	PD5	PD4	PD3	PD2	PD1	PD0

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 E-22 Port C Registers (PCRC, PRRC, PDRC)



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 Æ Port Pin configured as ESSI
PCn = 0 Æ 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 Æ Port Pin is Output
PDCn = 0 Æ 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 E-23 Port D Registers (PCRD, PRRD, PDRD)

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 Æ Port Pin configured as SCI
PCn = 0 Æ 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 Æ Port Pin is Output
PDCn = 0 Æ 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 E-24 Port E Registers (PCRE, PRRE, PDRE)

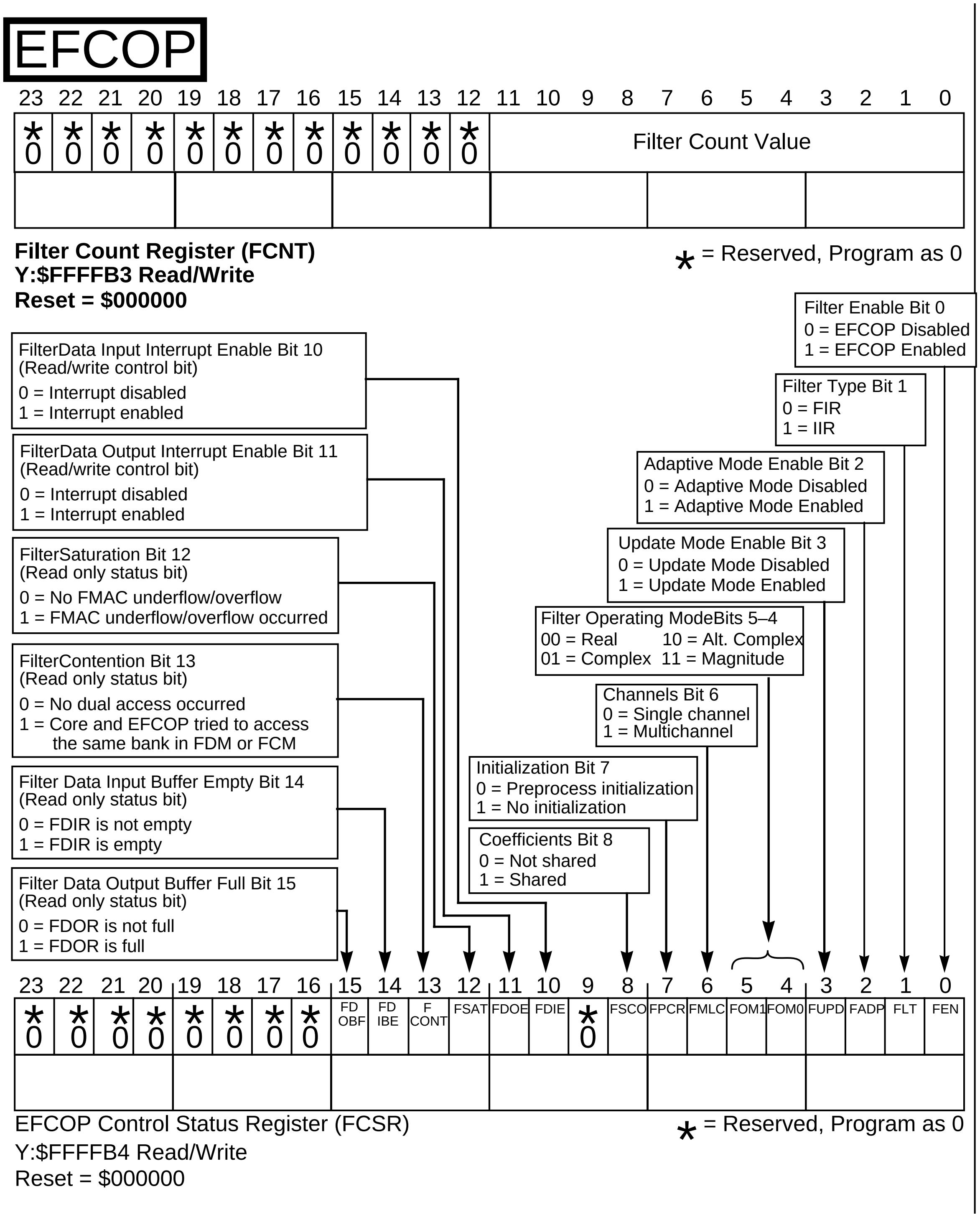
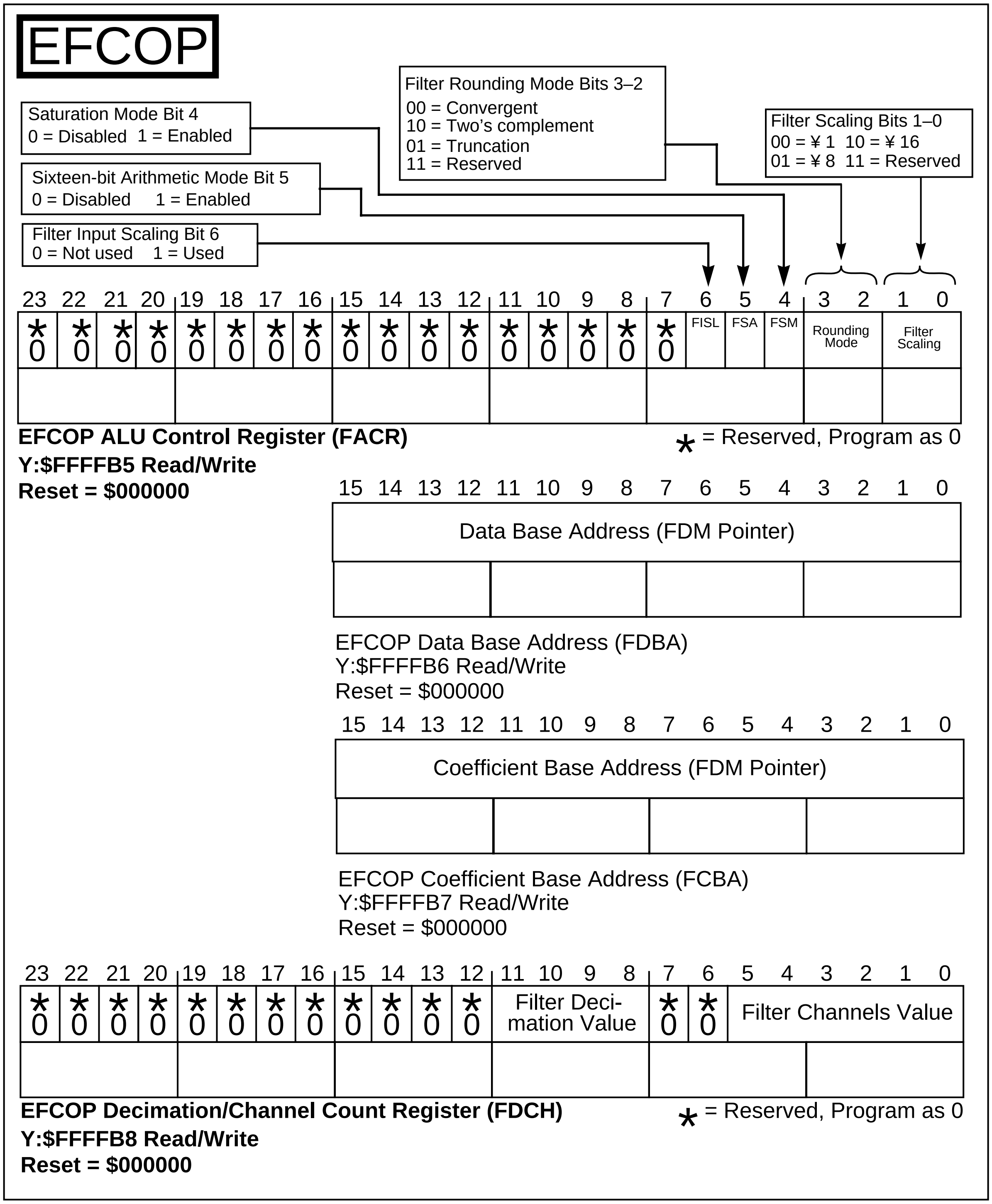


Figure E-25 EFCOP Counter and Control Status Registers (FCNT and FCSR)



EFCOP ALU Control Register (FACR)
Y:$FFFFB5 Read/Write
Reset = $000000

EFCOP Data Base Address (FDBA)
Y:$FFFFB6 Read/Write
Reset = $000000

EFCOP Coefficient Base Address (FCBA)
Y:$FFFFB7 Read/Write
Reset = $000000

EFCOP Decimation/Channel Count Register (FDCH)
Y:$FFFFB8 Read/Write
Reset = $000000

Figure E-26 EFCOP FACR, FDBA, FCBA, and FDCH Registers

INDEX