DSP56321 Reference Manual

24-Bit Digital Signal Processor

DSP56321RM
Rev. 1, March 2005

semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Document Order Number: DSP56321RM
Rev. 1
3/2005

Contents

1 Overview

2 Signals/Connections

3 Memory Configuration

4 Core Configuration

5 Clock Configuration

6 Programming the Peripherals

7 Host Interface (HI08)

8 Enhanced Synchronous Serial Interface

9 Serial Communication Interface

10 Triple Timer Module

11 Enhanced Filter Coprocessor

A Bootstrap Program

B Programming Reference

Index

Overview 1

The Freescale DSP56321, a member of the DSP56300 DSP family, supports networking, security encryption, and home entertainment using a high-performance, single-clock-cycle-per-instruction engine (DSP56000 code-compatible), a barrel shifter, 24-bit addressing, an instruction cache, and a direct memory access (DMA) controller. The DSP56321 offers 275 million multiply- accumulates per second (MMACS) performance, attaining 550 MMACS when the EFCOP is in use. It operates with an internal 275 MHz clock with a 1.6 volt core and independent 3.3 volt input/output (I/O) power. By operating in parallel with the core, the EFCOP provides overall enhanced performance and signal quality with no impact on channel throughput or total channel support. This device is pin-compatible with the Freescale DSP56303, DSP56L307, DSP56309, and DSP56311. 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. 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 DSP56321 includes the Freescale JTAG port and OnCE module.

The DSP56321 is intended for applications requiring a large amount of internal memory, such as networking and wireless infrastructure applications. The on-board EFCOP can accelerate general filtering applications, such as echo-cancellation applications, correlation, and general-purpose convolution-based algorithms.

This manual describes the DSP56321 24-bit DSP, its memory, operating modes, and peripheral modules. The DSP56321 is an implementation of the DSP56300 core with a unique configuration of on-chip memory, cache, and peripherals. Use this manual in conjunction with the *DSP56300 Family Manual (DSP56300FM/AD),* which describes the CPU, core programming models, and instruction set.

Note: The DSP56321 digital phase-lock loop (DPLL) and clock modules differ from other members of the DSP56300 family. For details on these modules, including the programming model, see **Chapter 4** of this manual.

The *DSP56321 Technical Data (DSP56321)*—referred to as the data sheet—provides DSP56321 electrical specifications, timing, pinout, and packaging descriptions.

You can obtain these documents and the DSP development tools through a local Freescale semiconductor sales office or authorized distributor. To receive the latest information on this DSP, access the web site listed on the back cover of this document.

1.1 Manual Organization

This manual contains the following sections and appendices:

- **Chapter 1,** *Overview*. Features list and block diagram, related documentation, organization of this manual, and the notational conventions used.
- **Chapter 2,** *Signals/Connections*. DSP56321 signals and their functional groupings.
- **Chapter 3,** *Memory Configuration***.** DSP56321 memory spaces, RAM configuration, memory configuration bit settings, memory configurations, and memory maps.
- **Chapter 4,** *Core Configuration*. Registers for configuring the DSP56300 core when programming the DSP56321, in particular the interrupt vector locations and the operation of the interrupt priority registers; operating modes and how they affect the processor's program and data memories. This chapter also provides detailed information about the Digital Phase-Lock Loop (DPLL) and clock modules that are unique to the DSP56321.
- **Chapter 5,** *Programming the Peripherals*. Guidelines on initializing the DSP56321 peripherals, including mapping control registers, specifying a method of transferring data, and configuring for general-purpose input/output (GPIO).
- **Chapter 6,** *Host Interface (HI08)*. Features, signals, architecture, programming model, reset, interrupts, external host programming model, initialization, and a quick reference to the HI08 programming model.
- **Chapter 7,** *Enhanced Synchronous Serial Interface (ESSI)***.** Enhancements, data and control signals, programming model, operating modes, initialization, exceptions, and GPIO.
- **Chapter 8,** *Serial Communication Interface (SCI)*. Signals, programming model, operating modes, reset, initialization, and GPIO.
- **Chapter 9,** *Triple Timer Module*. Architecture, programming model, and operating modes of three identical timer devices available for use as internals or event counters.
- **Chapter 10,** *Enhanced Filter Coprocessor (EFCOP).* Structure and function of the EFCOP, including features, architecture, and programming model; programming topics such as data transfer to and from the EFCOP, its use in different modes, and examples of usage.
- **Appendix A,** *Bootstrap Code*. Bootstrap code for the DSP56321.
- **Appendix B,** *Programming Reference*. Peripheral addresses, interrupt addresses, and interrupt priorities for the DSP56321; programming sheets listing the contents of the major DSP56321 registers for programmer's reference.

1.2 Manual Conventions

This manual uses the following conventions:

- Register bits are 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 described. 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 bit locations within a register.
- A "set," bit has a value of 1. A "cleared" bit has a value of 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
PIN1	True	Asserted	Ground[2]
PIN	False	Deasserted	V_{CC}[3]
PIN	True	Asserted	V_{CC}
PIN	False	Deasserted	Ground

Notes:
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 as follows:
  - — 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 HA[0–3].
- "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
```

- Hexadecimal values are indicated with a *$* 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
  — the reset function, written as reset

1.3 Manual Revision History for Revision 1

Table 1-2 lists the changes made in this manual from Revision 0 to Revision 1.

Table 1-2. Change History, Revision 3 to Revision 4

Change	Section Number	Revision 3 Page Number	Revision 4 Page Number
Modified signal definitions for some external bus control, HI08, ESSI, SCI, and timer signals.	**Figure 2-1** **Table 2-8** **Table 2-10** **Table 2-11** **Table 2-12** **Table 2-13** **Table 2-14**	Page 2-2 Page 2-5 Page 2-8 Page 2-11 Page 2-13 Page 2-15 Page 2-16	Page 2-2 Page 2-5 Page 2-8 Page 2-11 Page 2-12 Page 2-14 Page 2-15
New Operating Mode Register (OMR) layout and bit definitions.	**Section 4.3.2**	Page 4-10	Page 4-9
New Bus Control Register (BCR) layout and bit definitions.	**Section 4.5.1**	Page 4-20	Page 4-18
Updated GPIO signal names.	**Section 6.5.1, Figure 6-2** **Section 6.5.4, Figure 6-5**	Page 6-7 Page 6-8	Page 6-6 Page 6-7
Updated HDR address	**Section 7.6.3, Figure 7-9**	Page 7-16	Page 7-14
Updated SCI Receive Register (SRX) description	**Section 9.6.4.1**	Page 9-24	Page 9-21
Updated programming sheets for the OMR, BCR, Address Attribute Registers (AAR[3–0]), and timer registers (TLR, TCPR, TCR).	**Figure B-2** **Figure B-7** **Figure B-8** **Figure B-22**	Page B-14 Page B-19 Page B-20 Page B-34	Page B-12 Page B-15 Page B-16 Page B-30

1.4 Features

Table 2 lists the features of the DSP56321 device.

Table 1-3. DSP56321 Features

Feature	Description
High-Performance DSP56300 Core	• 275 million multiply-accumulates per second (MMACS) (550 MMACS using the EFCOP in filtering applications) with a 275 MHz clock at 1.6 V core and 3.3 V I/O • Object code compatible with the DSP56000 core with highly parallel instruction set • Data arithmetic logic unit (Data ALU) with fully pipelined 24 × 24-bit parallel Multiplier-Accumulator (MAC), 56-bit parallel barrel shifter (fast shift and normalization; bit stream generation and parsing), conditional ALU instructions, and 24-bit or 16-bit arithmetic support under software control • Program control unit (PCU) with position independent code (PIC) support, addressing modes optimized for DSP applications (including immediate offsets), internal instruction cache controller, internal memory-expandable hardware stack, nested hardware DO loops, and fast auto-return interrupts • Direct memory access (DMA) with six DMA channels supporting internal and external accesses; one-, two-, and three-dimensional transfers (including circular buffering); end-of-block-transfer interrupts; and triggering from interrupt lines and all peripherals • Phase-lock loop (PLL) allows change of low-power divide factor (DF) without loss of lock and output clock with skew elimination • Hardware debugging support including on-chip emulation (OnCE) module, Joint Test Action Group (JTAG) test access port (TAP)
Enhanced Filter Coprocessor (EFCOP)	• Internal 24 × 24-bit filtering and echo-cancellation coprocessor that runs in parallel to the DSP core • Operation at the same frequency as the core (up to 275 MHz) • Support for 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
Internal Peripherals	• Enhanced 8-bit parallel host interface (HI08) supports a variety of buses (for example, ISA) and provides glueless connection to a number of industry-standard microcomputers, microprocessors, and DSPs • Two enhanced synchronous serial interfaces (ESSI), each with one receiver and three transmitters (allows six-channel home theater) • Serial communications interface (SCI) with baud rate generator • Triple timer module • Up to 34 programmable general-purpose input/output (GPIO) pins, depending on which peripherals are enabled

Table 1-3. DSP56321 Features (Continued)

<table>
<tr><th>Feature</th><th>Description</th></tr>
<tr><td><b>Internal Memories</b></td><td>• 192 × 24-bit bootstrap ROM
• 192 K × 24-bit RAM total
• Program RAM, instruction cache, X data RAM, and Y data RAM sizes are programmable:
<table>
<tr><th>Program RAM Size</th><th>Instruction Cache Size</th><th>X Data RAM Size*</th><th>Y Data RAM Size*</th><th>Instruction Cache</th><th>MSW2</th><th>MSW1</th><th>MSW0</th></tr>
<tr><td>32 K × 24-bit</td><td>0</td><td>80 K × 24-bit</td><td>80 K × 24-bit</td><td>disabled</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>31 K × 24-bit</td><td>1024 × 24-bit</td><td>80 K × 24-bit</td><td>80 K × 24-bit</td><td>enabled</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>40 K × 24-bit</td><td>0</td><td>76 K × 24-bit</td><td>76 K × 24-bit</td><td>disabled</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>39 K × 24-bit</td><td>1024 × 24-bit</td><td>76 K × 24-bit</td><td>76 K × 24-bit</td><td>enabled</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>48 K × 24-bit</td><td>0</td><td>72 K × 24-bit</td><td>72 K × 24-bit</td><td>disabled</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>47 K × 24-bit</td><td>1024 × 24-bit</td><td>72 K × 24-bit</td><td>72 K × 24-bit</td><td>enabled</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>64 K × 24-bit</td><td>0</td><td>64 K × 24-bit</td><td>64 K × 24-bit</td><td>disabled</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>63 K × 24-bit</td><td>1024 × 24-bit</td><td>64 K × 24-bit</td><td>64 K × 24-bit</td><td>enabled</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>72 K × 24-bit</td><td>0</td><td>60 K × 24-bit</td><td>60 K × 24-bit</td><td>disabled</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>71 K × 24-bit</td><td>1024 × 24-bit</td><td>60 K × 24-bit</td><td>60 K × 24-bit</td><td>enabled</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>80 K × 24-bit</td><td>0</td><td>56 K × 24-bit</td><td>56 K × 24-bit</td><td>disabled</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>79 K × 24-bit</td><td>1024 × 24-bit</td><td>56 K × 24-bit</td><td>56 K × 24-bit</td><td>enabled</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>96 K × 24-bit</td><td>0</td><td>48 K × 24-bit</td><td>48 K × 24-bit</td><td>disabled</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>95 K × 24-bit</td><td>1024 × 24-bit</td><td>48 K × 24-bit</td><td>48 K × 24-bit</td><td>enabled</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>112 K × 24-bit</td><td>0</td><td>40 K × 24-bit</td><td>40 K × 24-bit</td><td>disabled</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>111 K × 24-bit</td><td>1024 × 24-bit</td><td>40 K × 24-bit</td><td>40 K × 24-bit</td><td>enabled</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td colspan="8">*Includes 12 K × 24-bit shared memory (that is, 24 K total memory shared by the core and the EFCOP)</td></tr>
</table>
</td></tr>
<tr><td><b>External Memory Expansion</b></td><td>• Data memory expansion to two 256 K × 24-bit word memory spaces using the standard external address lines
• Program memory expansion to one 256 K × 24-bit words memory space using the standard external address lines
• External memory expansion port
• Chip select logic for glueless interface to static random access memory (SRAMs)</td></tr>
<tr><td><b>Power Dissipation</b></td><td>• Very low-power CMOS design
• Wait and Stop low-power standby modes
• Fully static design specified to operate down to 0 Hz (dc)
• Optimized power management circuitry (instruction-dependent, peripheral-dependent, and mode-dependent)</td></tr>
<tr><td><b>Packaging</b></td><td>• Molded array plastic-ball grid array (MAP-BGA) package in lead-free or lead-bearing versions.</td></tr>
</table>

1.5 DSP56300 Core Functional Blocks

The functional blocks of the DSP56300 core are:

- Data arithmetic logic unit (ALU)
- Address generation unit (AGU)
- Program control unit
- DPLL and clock oscillator
- JTAG TAP and OnCE module
- Memory

In addition, the DSP56321 provides a set of on-chip peripherals, discussed in **Section 1.9**, *Peripherals*, on page 1-13.

1.5.1 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, 48-bit, or 56-bit arithmetic support
- Four 24-bit or 48-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

1.5.1.1 Data ALU Registers

The data ALU registers are 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 can be 16, 32, or 40 bits and always originate from data ALU registers. The results of all data ALU operations are stored in an accumulator. Data ALU operations are performed in two clock cycles in a pipeline 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 a source operand for the immediately following operation without penalty.

1.5.1.2 Multiplier-Accumulator (MAC)

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

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

1.5.2 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 halves, each with its own identical address ALU. Each address ALU has four sets of register triplets, and each register triplet includes an address register, offset register, and modifier register. Each contains a 24-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 specify the type of arithmetic used in the address register update calculation. The modifier value is decoded in the address ALU.

1.5.3 Program Control Unit (PCU)

The PCU fetches and decodes instructions, controls hardware DO loops, and processes exceptions. Its seven-stage pipeline controls the different processing states of the DSP56300 core. The PCU consists of three hardware blocks:

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

The PCU uses 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

1.5.4 Clock Generator Circuit

The DSP56321 contain a Clock Generator (CLKGEN) that is different from the clock generator and Phase-Lock Loop (PLL) used by other members of the DSP56300 family. The DSP56321 uses a CLKGEN module with an integrated Digital Phase-Lock Loop (DPLL). As with the previous clock generator, the DSP56321 allows you to change the low-power Division Factor (DF) without losing lock.

The CLKGEN module uses two internal X-I/O-mapped registers that are different from the standard DSP56300 design described in the *DSP56300 Family Manual*. **Chapter 5** describes the new clock registers in detail.

The DPLL 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.

1.5.5 JTAG TAP and OnCE Module

In the DSP56300 core is a dedicated user-accessible TAP that is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*. Problems 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 with 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. For details on the JTAG port, consult the *DSP56300 Family Manual*.

The OnCE module interacts with the DSP56300 core and its peripherals nonintrusively so that you 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. For details on the OnCE module, consult the *DSP56300 Family Manual*.

1.5.6 On-Chip Memory

The memory space of the DSP56300 core is partitioned into program, X data, and Y data memory space. The data memory space is divided into X 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. There is an on-chip 192 × 24-bit bootstrap ROM. For details on internal memory, see **Chapter 3,** *Memory Configuration*. Program RAM, instruction cache, X data RAM, and Y data RAM size are programmable, as **Table 1-3**, *DSP56321 Features,* on page 1-5 shows.

1.5.7 Off-Chip Memory Expansion

Memory can be expanded off chip to the following capacities:

- Data memory expansion to two 256 K × 24-bit word memory spaces using standard address lines
- Program memory expansion to one 256 K × 24-bit word memory space using standard address lines

Further features of off-chip memory include the following:

- External memory expansion port
- Simultaneous glueless interface to Static RAM (SRAM)

1.6 Internal Buses

To provide data exchange between blocks, the DSP56321 implements the following buses:

- 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 these buses among other components.

1.7 Block Diagram

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 DSP56321.

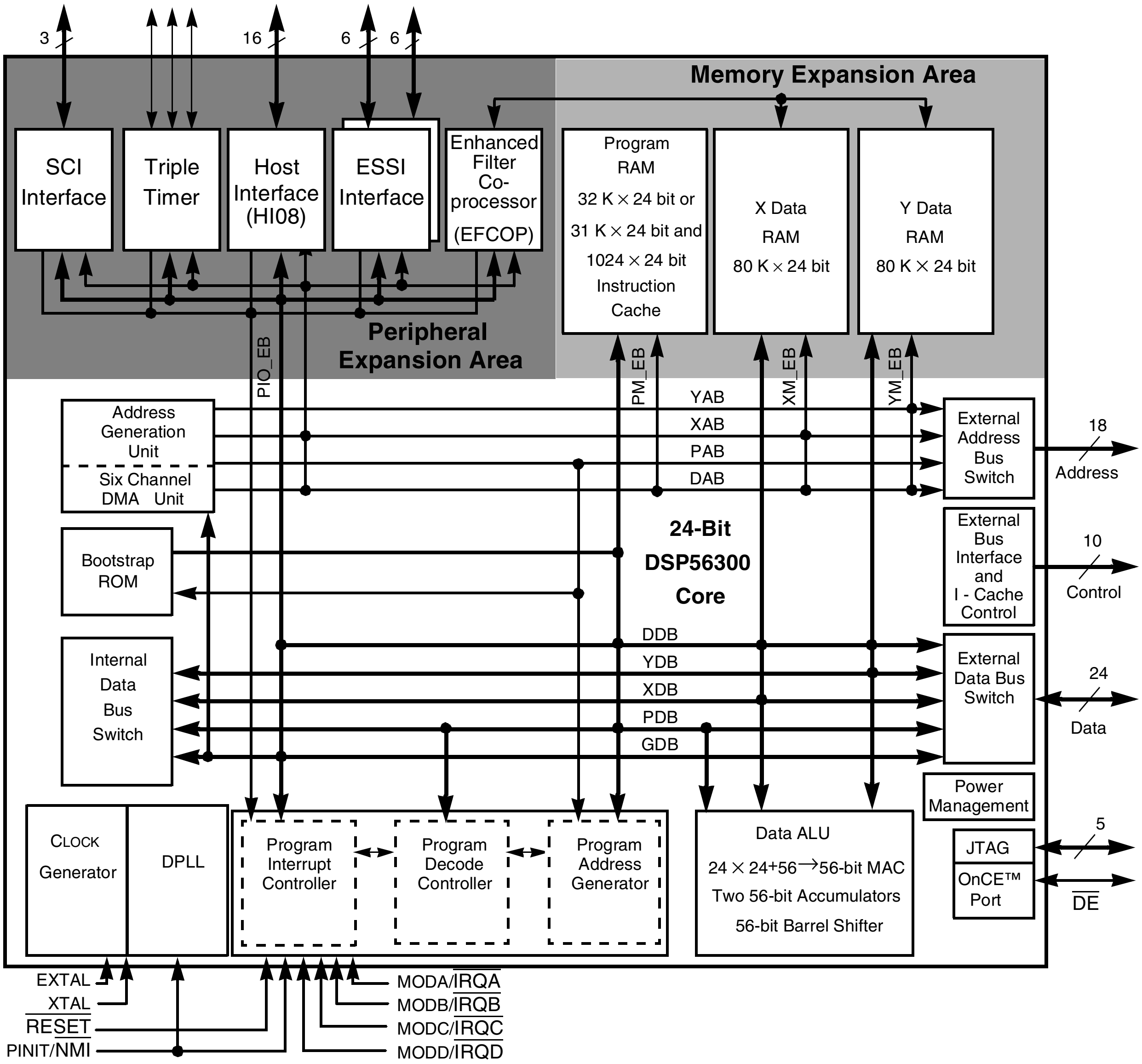


Figure 1-1. DSP56321 Block Diagram

Note: See **Section 1.5.6**, *On-Chip Memory*, on page 1-10 for details on memory size.

1.8 DMA Controller

The features of the DMA controller are as follows:

- 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

1.9 Peripherals

In addition to the core features, the DSP56321 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

1.9.1 GPIO Functionality

The GPIO port consists of up to 34 programmable signals, 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. Programming techniques for these registers to control GPIO functionality are detailed in **Chapter 6**, *Programming the Peripherals*.

1.9.2 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 double-buffered transmit and receive data registers 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 using standard instructions and addressing modes.

1.9.3 ESSI

The DSP56321 provides two independent and identical ESSIs. Each ESSI has a full-duplex serial port for communication with a variety of serial devices, including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals that implement the Freescale SPI. The ESSI consists of independent transmitter and receiver sections and a common ESSI clock generator. ESSI capabilities 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

1.9.4 SCI

The 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). SCI asynchronous protocols include a multidrop mode for master/slave operation with wake-up on idle line and wake-up on address bit capability. This mode allows the DSP56321 to share a single serial line efficiently with other peripherals.

Separate SCI transmit and receive sections 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 allow the baud-rate generator to function as a general-purpose timer when the SCI is not using it or when the interrupt timing is the same as that of the SCI.

1.9.5 Timer Module

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

- A single signal that can function as a GPIO signal or as a timer signal
- Uses 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
- Connection to the external world through one bidirectional signal. When this signal is configured as an input, the timer functions as an external event counter or measures external pulse width/signal period. When the signal is used as an output, the timer functions as either a timer, a watchdog, or a pulse width modulator.

1.9.6 EFCOP

The EFCOP interfaces with the DSP core via the peripheral module bus. It is a general-purpose, fully programmable coprocessor that performs filtering tasks concurrently with the DSP core, with minimum core overhead. The DSP core and the EFCOP can share data via 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 10K taps and 10K coefficients in any combination of number and length of filters (for example, eight filters of length 512, or 16 filters of length 256). It performs either 24-bit or 16-bit precision arithmetic with full support for saturation arithmetic. A cost-effective and power-efficient coprocessor, the EFCOP accelerates filtering tasks, such as echo cancellation or correlation, concurrently with software running on the DSP core.

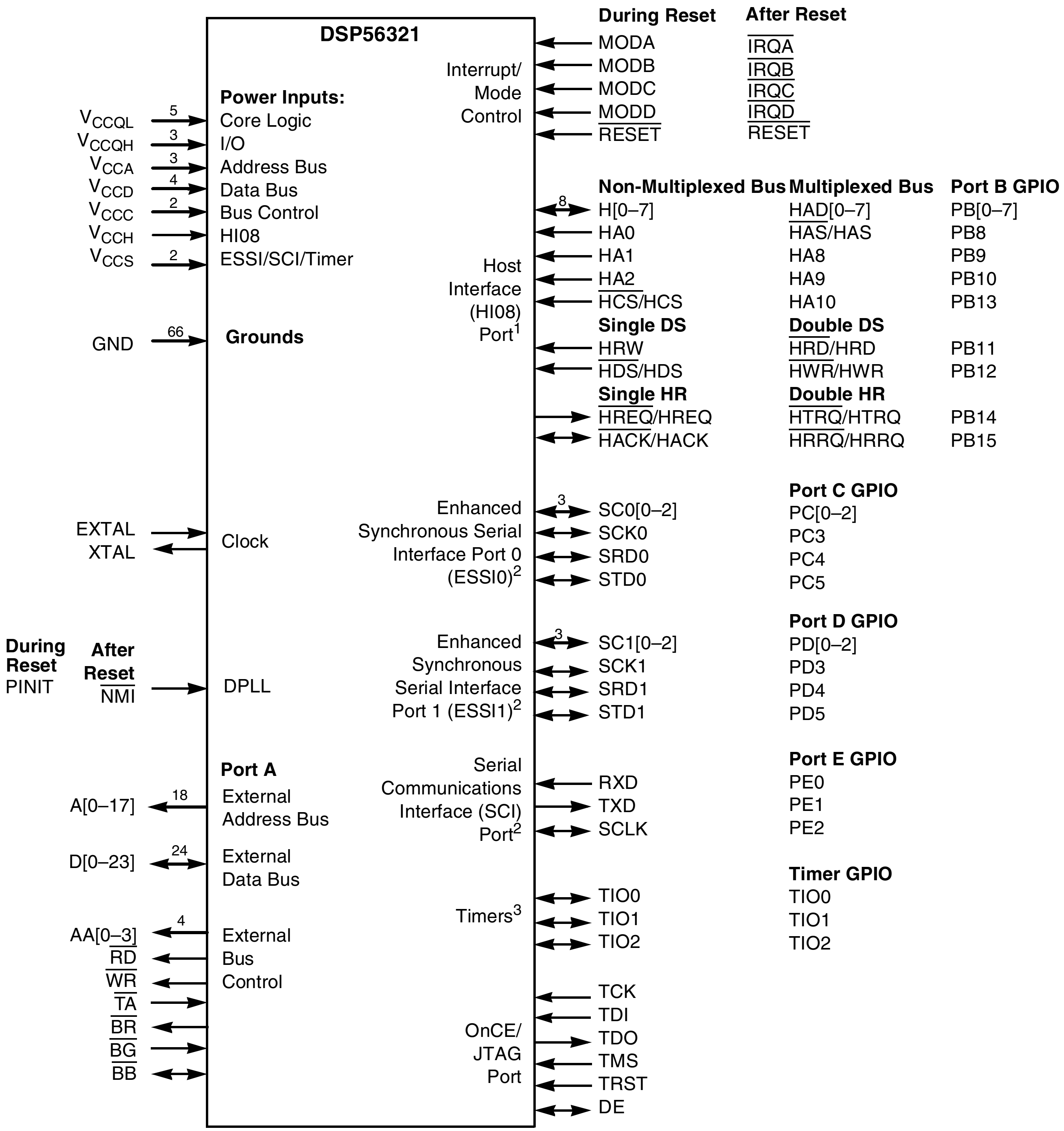
Signals/Connections 2

The DSP56321 input and output signals are organized into functional groups as shown in **Table 2-1**. **Figure 2-1** diagrams the DSP56321 signals by functional group. The remainder of this chapter describes the signal pins in each functional group

Table 2-1. DSP56321 Functional Signal Groupings

Functional Group		Number of Signals
Power (V_{CC})		20
Ground (GND)		66
Clock		2
PLL		1
Address bus	Port A[1]	18
Data bus		24
Bus control		10
Interrupt and mode control		5
Host interface (HI08)	Port B[2]	16
Enhanced synchronous serial interface (ESSI)	Ports C and D[3]	12
Serial communication interface (SCI)	Port E[4]	3
Timer		3
JTAG/OnCE Port		6

Notes:
1. Port A signals define the external memory interface port, including the external address bus, data bus, and control signals.
2. Port B signals are the HI08 port signals multiplexed with the GPIO signals.
3. Port C and D signals are the two ESSI port signals multiplexed with the GPIO signals.
4. Port E signals are the SCI port signals multiplexed with the GPIO signals.
5. There are two reserved and eight NC (not connected) signal pins.



Notes:

1. The HI08 port supports a non-multiplexed or a multiplexed bus, single or double Data Strobe (DS), and single or double Host Request (HR) configurations. Since each of these modes is configured independently, any combination of these modes is possible. These HI08 signals can also be configured alternately as GPIO signals (PB[0–15]). Signals with dual designations (for example, HAS/HAS) have configurable polarity.
2. The ESSI0, ESSI1, and SCI signals are multiplexed with the Port C GPIO signals (PC[0–5]), Port D GPIO signals (PD[0–5]), and Port E GPIO signals (PE[0–2]), respectively.
3. TIO[0–2] can be configured as GPIO signals.

Figure 2-1. Signals Identified by Functional Group

2.1 Power

Table 2-2. Power Inputs

Power Name	Description
V_{CCQL}	**Quiet Core (Low) Power** An isolated power for the DSP56300 core processing logic. This input must be isolated externally from all other chip power inputs.
V_{CCQH}	**Quiet External (High) Power** A quiet power source for I/O lines. This input must be tied externally to all other chip power inputs, *except* V_{CCQL}.
V_{CCA}	**Address Bus Power** 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}.
V_{CCD}	**Data Bus Power** 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}.
V_{CCC}	**Bus Control Power** 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}.
V_{CCH}	**Host Power** An isolated power for the HI08 I/O drivers. This input must be tied externally to all other chip power inputs**,** *except* V_{CCQL}.
V_{CCS}	**ESSI, SCI, and Timer Power** 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}.
Note: The user must provide adequate external decoupling capacitors for all power connections.	

2.2 Ground

Table 2-3. Grounds

Ground Name	Description
GND	**Ground** Connected to an internal device ground plane.
Note: The user must provide adequate external decoupling capacitors for all GND connections	

2.3 Clock

Table 2-4. Clock Signals

Signal Name	Type	State During Reset	Signal Description
EXTAL	Input	Input	**External Clock/Crystal Input** Interfaces the internal crystal oscillator input to an external crystal or an external clock.
XTAL	Output	Chip-driven	**Crystal Output** Connects the internal crystal oscillator output to an external crystal. If an external clock is used, leave XTAL unconnected.

2.4 PLL

Table 2-5. Digital Phase-Lock Loop Signals

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.

2.5 External Memory Expansion Port (Port A)

When the DSP56321 enters a low-power standby mode (stop or wait), it releases bus mastership and tri-states the relevant port A signals: A[0–17], D[0–23], AA[0–3], $\overline{\text{RD}}$, $\overline{\text{WR}}$, and $\overline{\text{BB}}$.

2.5.1 External Address Bus

Table 2-6. External Address Bus Signals

Signal Name	Type	State During Reset	Signal Description
A[0–17]	Output	Tri-stated	**Address Bus** When the DSP is the bus master, A[0–17] 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, A[0–17] do not change state when external memory spaces are not being accessed.

2.5.2 External Data Bus

Table 2-7. External Data Bus Signals

Signal Name	Type	State During Reset	Signal Description
D[0–23]	Input/ Output	Tri-stated	**Data Bus** When the DSP is the bus master, D[0–23] are active-high, bidirectional input/outputs that provide the bidirectional data bus for external program and data memory accesses. Otherwise, D[0–23] are tri-stated. These lines have weak keepers to maintain the last state even if all drivers are tri-stated.

2.5.3 External Bus Control

Table 2-8. External Bus Control Signals

Signal Name	Type	State During Reset, Stop, or Wait	Signal Description
AA[0–3]	Output	Tri-stated	**Address Attribute** These signals are used as chip selects or additional address lines. The default use defines a priority scheme under which only one AA signal is asserted at a time. Setting the AA priority disable (APD) bit (Bit 14) of the OMR disables the priority mechanism and the lines are used together as four external lines decoded externally into 16 chip select signals.
$\overline{\text{RD}}$	Output	Tri-stated	**Read Enable** When the DSP is the bus master, $\overline{\text{RD}}$ is asserted to read external memory on the data bus (D[0–23]). 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 asserted to write external memory on the data bus (D[0–23]). Otherwise, the signals are tri-stated.
$\overline{\text{TA}}$	Input	Ignored Input	**Transfer Acknowledge** If the DSP56321 is the bus master and there is no external bus activity, or the DSP56321 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) can 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. For correct operation, the $\overline{\text{TAS}}$ bit must be set in the Operating Mode Register (OMR) to synchronize the $\overline{\text{TA}}$ signal with the internal clock. The current bus cycle completes one clock period after $\overline{\text{TA}}$ is deasserted. The number of wait states is determined by the $\overline{\text{TA}}$ input or by the BCR, whichever is longer. The BCR sets the minimum number of wait states in external bus cycles. 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.

Table 2-8. External Bus Control Signals (Continued)

Signal Name	Type	State During Reset, Stop, or Wait	Signal Description
$\overline{\text{BR}}$	Output	Reset: Output (deasserted) State during Stop/Wait depends on BRH bit setting:	**Bus Request** Asserted when the DSP requests bus mastership. $\overline{\text{BR}}$ is deasserted when the DSP no longer needs the bus. $\overline{\text{BR}}$ is asserted or deasserted independently of whether the DSP56321 is a bus master or a bus slave. Bus “parking” allows $\overline{\text{BR}}$ to be deasserted even though the DSP56321 is the bus master. (See the description of bus “parking” in the $\overline{\text{BB}}$ signal description.) The Bus Request Hold (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 affected only 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. The state during Stop/Wait depends on the BRH bit setting: • BRH = 0: Output, deasserted • BRH = 1: Maintains last state (that is, if asserted, remains asserted)
$\overline{\text{BG}}$	Input	Ignored Input	**Bus Grant** Asserted by an external bus arbitration circuit when the DSP56321 becomes the next bus master. When $\overline{\text{BG}}$ is asserted, the DSP56321 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 *DSP56321 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.
$\overline{\text{BB}}$	Input/ Output	Ignored input	**Bus Busy** 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 can 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. $\overline{\text{BB}}$ is deasserted by an “active pull-up” method (that is, $\overline{\text{BB}}$ 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 *DSP56321 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.

2.6 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
MODA $\overline{\text{IRQA}}$	Input Input	Input, Schmitt-trigger	**Mode Select A** MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into the OMR when the $\overline{\text{RESET}}$ signal is deasserted. **External Interrupt Request A** After reset, this input becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If the processor is in the STOP or WAIT standby state and $\overline{\text{IRQA}}$ is asserted, the processor exits the STOP or WAIT state.
MODB $\overline{\text{IRQB}}$	Input Input	Input, Schmitt-trigger	**Mode Select B** MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into the OMR when the $\overline{\text{RESET}}$ signal is deasserted. **External Interrupt Request B** After reset, this input becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If the processor is in the WAIT standby state and $\overline{\text{IRQB}}$ is asserted, the processor exits the WAIT state.
MODC $\overline{\text{IRQC}}$	Input Input	Input, Schmitt-trigger	**Mode Select C** MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into the OMR when the $\overline{\text{RESET}}$ signal is deasserted. **External Interrupt Request C** After reset, this input becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If the processor is in the WAIT standby state and $\overline{\text{IRQC}}$ is asserted, the processor exits the WAIT state.
MODD $\overline{\text{IRQD}}$	Input Input	Input, Schmitt-trigger	**Mode Select D** MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into the OMR when the $\overline{\text{RESET}}$ signal is deasserted. **External Interrupt Request D** After reset, this input becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. If the processor is in the WAIT standby state and $\overline{\text{IRQD}}$ is asserted, the processor exits the WAIT state.
$\overline{\text{RESET}}$	Input	Input, Schmitt-trigger	**Reset** When asserted, places the chip in the Reset state and resets the internal phase generator. The Schmitt-trigger input allows a slowly rising input (such as a capacitor charging) to reset the chip reliably. 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.

2.7 HI08

The HI08 provides a fast, 8-bit, parallel data port that connects directly to the host bus. The HI08 supports a variety of standard buses and can directly connect to a number of industry-standard microcomputers, microprocessors, DSPs, and DMA hardware.

Table 2-10. Host Interface

Signal Name	Type	State During Reset[1]	Signal Description
H[0–7]	Input/Output	Disconnected internally	**Host Data** When the HI08 is programmed to interface with a non-multiplexed host bus and the HI function is selected, these signals are lines 0–7 of the Data bus.
HAD[0–7]	Input/Output		**Host Address** When the HI08 is programmed to interface with a multiplexed host bus and the HI function is selected, these signals are lines 0–7 of the Address/Data bus.
PB[0–7]	Input or Output		**Port B 0–7** When the HI08 is configured as GPIO through the HPCR, these signals are individually programmed through the HI08 Data Direction Register (HDDR).
HA0	Input	Disconnected internally	**Host Address Input 0** When the HI08 is programmed to interface with a non-multiplexed host bus and the HI function is selected, this signal is line 0 of the Host Address bus.
$\overline{\text{HAS}}$/HAS	Input		**Host Address Strobe** When the HI08 is programmed to interface with 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 through the HDDR.
HA1	Input	Disconnected internally	**Host Address Input 1** When the HI08 is programmed to interface with a non-multiplexed host bus and the HI function is selected, this signal is line 1 of the Host Address bus.
HA8	Input		**Host Address 8** When the HI08 is programmed to interface with a multiplexed host bus and the HI function is selected, this signal is line 8 of the Host Address bus.
PB9	Input or Output		**Port B 9** When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed through the HDDR.

Table 2-10. Host Interface (Continued)

Signal Name	Type	State During Reset[1]	Signal Description
HA2	Input	Disconnected internally	**Host Address Input 2** When the HI08 is programmed to interface with a non-multiplexed host bus and the HI function is selected, this signal is line 2 of the Host Address bus.
HA9	Input		**Host Address 9** When the HI08 is programmed to interface with a multiplexed host bus and the HI function is selected, this signal is line 9 of the Host Address bus.
PB10	Input or Output		**Port B 10** When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed through the HDDR.
HRW	Input	Disconnected internally	**Host Read/Write** When the HI08 is programmed to interface with a single-data-strobe host bus and the HI function is selected, this signal is the Host Read/$\overline{\text{Write}}$ input.
$\overline{\text{HRD}}$/HRD	Input		**Host Read Data** When the HI08 is programmed to interface with a double-data-strobe host bus and the HI function is selected, this signal is the Host Read Data strobe (HRD) Schmitt-trigger input. The polarity of the data strobe is programmable, but is configured as active-low ($\overline{\text{HRD}}$) after reset.
PB11	Input or Output		**Port B 11** When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed through the HDDR.
$\overline{\text{HDS}}$/HDS	Input	Disconnected internally	**Host Data Strobe** When the HI08 is programmed to interface with 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 the HI08 is programmed to interface with 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 through the HDDR.

Table 2-10. Host Interface (Continued)

Signal Name	Type	State During Reset[1]	Signal Description
HCS	Input	Disconnected internally	**Host Chip Select** When the HI08 is programmed to interface with a non-multiplexed host bus and the HI function is selected, this signal is the Host Chip Select (HCS) input. The polarity of the chip select is programmable, but is configured active-low ($\overline{HCS}$) after reset.
HA10	Input		**Host Address 10** When the HI08 is programmed to interface with a multiplexed host bus and the HI function is selected, this signal is line 10 of the Host Address bus.
PB13	Input or Output		**Port B 13** When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed through the HDDR.
$\overline{HREQ}$/HREQ	Output	Disconnected internally	**Host Request** When the HI08 is programmed to interface with 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 ($\overline{HREQ}$) following reset. The host request can be programmed as a driven or open-drain output.
$\overline{HTRQ}$/HTRQ	Output		**Transmit Host Request** When the HI08 is programmed to interface with 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 ($\overline{HTRQ}$) following reset. The host request may be programmed as a driven or open-drain output.
PB14	Input or Output		**Port B14** When the HI08 is configured as GPIO through the HPCR, this signal is individually programmed through the HDDR.
$\overline{HACK}$/HACK	Input	Disconnected internally	**Host Acknowledge** When the HI08 is programmed to interface with 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 ($\overline{HACK}$) after reset.
$\overline{HRRQ}$/HRRQ	Output		**Receive Host Request** When the HI08 is programmed to interface with 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 ($\overline{HRRQ}$) 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 through the HDDR.

Notes: **1.** In the Stop state, the signal maintains the last state as follows:

- If the last state is input, the signal is an ignored input.
- If the last state is output, the keeper circuit maintains the last output level even if the internal driver is tri-stated.

2. The Wait processing state does not affect the signal state.

2.8 Enhanced Synchronous Serial Interface 0

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

Table 2-11. Enhanced Synchronous Serial Interface 0 (ESSI0)

Signal Name	Type	State During Reset[1, 2]	Signal Description
SC00 PC0	Input or Output	Ignored input	**Serial Control 0** Functions in either Synchronous or Asynchronous mode. For Asynchronous mode, this signal is the receive clock I/O (Schmitt-trigger input). For Synchronous mode, this signal is either for Transmitter 1 output or Serial I/O Flag 0. **Port C 0** The default configuration following reset is GPIO. For PC0, signal direction is controlled through the Port C Direction Register (PRRC). This signal is configured as SC00 or PC0 through the Port C Control Register (PCRC).
SC01 PC1	Input/Output Input or Output	Ignored input	**Serial Control 1** Functions in either Synchronous or Asynchronous mode. For Asynchronous mode, this signal is the receiver frame sync I/O. For Synchronous mode, this signal is either Transmitter 2 output or Serial I/O Flag 1. **Port C 1** The default configuration following reset is GPIO. For PC1, signal direction is controlled through PRRC. This signal is configured as SC01 or PC1 through PCRC.
SC02 PC2	Input/Output Input or Output	Ignored input	**Serial Control Signal 2** 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. For PC2, signal direction is controlled through PRRC. This signal is configured as SC02 or PC2 through PCRC.
SCK0 PC3	Input/Output Input or Output	Ignored input	**Serial Clock** Provides the serial bit rate clock for the ESSI interface for both the transmitter and receiver in Synchronous modes, or the transmitter only in Asynchronous modes. Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 6 T (that is, 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. For PC3, signal direction is controlled through PRRC. This signal is configured as SCK0 or PC3 through PCRC.

Table 2-11. Enhanced Synchronous Serial Interface 0 (ESSI0) (Continued)

Signal Name	Type	State During Reset[1, 2]	Signal Description
SRD0 PC4	Input Input or Output	Ignored input	**Serial Receive Data** Receives serial data and transfers the data to the ESSI receive shift register. SRD0 is an input when data is being received. **Port C 4** The default configuration following reset is GPIO. For PC4, signal direction is controlled through PRRC. This signal is configured as SRD0 or PC4 through PCRC.
STD0 PC5	Output Input or Output	Ignored input	**Serial Transmit Data** Transmits data from the serial transmit shift register. STD0 is an output when data is being transmitted. **Port C 5** The default configuration following reset is GPIO. For PC5, signal direction is controlled through PRRC. This signal is configured as STD0 or PC5 through PCRC.

Notes: **1.** In the Stop state, the signal maintains the last state as follows:

- If the last state is input, the signal is an ignored input.
- If the last state is output, the keeper circuit maintains the last output level even if the internal driver is tri-stated.

2. The Wait processing state does not affect the signal state.

2.9 Enhanced Synchronous Serial Interface 1

Table 2-12. Enhanced Synchronous Serial Interface 1 (ESSI1)

Signal Name	Type	State During[1]		Signal Description
		Reset	Stop	
SC10 PD0	Input or Output	Input	Disconnected internally	**Serial Control 0** Functions in either Synchronous or Asynchronous mode. For Asynchronous mode, this signal is the receive clock I/O (Schmitt-trigger input). For Synchronous mode, this signal is either for Transmitter 1 output or Serial I/O Flag 0. **Port D 0** The default configuration following reset is GPIO. For PD0, signal direction is controlled through the Port D Direction Register (PRRD). This signal is configured as SC10 or PD0 through the Port D Control Register (PCRD).
SC11 PD1	Input/Output Input or Output	Input	Disconnected internally	**Serial Control 1** Functions in either Synchronous or Asynchronous mode. For Asynchronous mode, this signal is the receiver frame sync I/O. For Synchronous mode, this signal is either Transmitter 2 output or Serial I/O Flag 1. **Port D 1** The default configuration following reset is GPIO. For PD1, signal direction is controlled through PRRD. This signal is configured as SC11 or PD1 through PCRD.

Table 2-12. Enhanced Synchronous Serial Interface 1 (ESSI1) (Continued)

Signal Name	Type	State During[1]		Signal Description
		Reset	Stop	
SC12	Input/Output	Input	Disconnected internally	**Serial Control Signal 2** 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. For PD2, signal direction is controlled through PRRD. This signal is configured as SC12 or PD2 through PCRD.
SCK1	Input/Output	Input	Disconnected internally	**Serial Clock** Provides the serial bit rate clock for the ESSI interface for both the transmitter and receiver in Synchronous modes, or the transmitter only in Asynchronous modes. Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 6 T (that is, 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. For PD3, signal direction is controlled through PRRD. This signal is configured as SCK1 or PD3 through PCRD.
SRD1	Input	Input	Disconnected internally	**Serial Receive Data** Receives serial data and transfers the data to the ESSI receive shift register. SRD0 is an input when data is being received.
PD4	Input or Output			**Port D 4** The default configuration following reset is GPIO. For PD4, signal direction is controlled through PRRD. This signal is configured as SRD1 or PD4 through PCRD.
STD1	Output	Input	Disconnected internally	**Serial Transmit Data** Transmits data from the serial transmit shift register. STD1 is an output when data is being transmitted.
PD5	Input or Output			**Port C 5** The default configuration following reset is GPIO. For PD5, signal direction is controlled through PRRD. This signal is configured as STD1 or PD5 through PCRD.

Note: The Wait processing state does not affect the signal state.

2.10 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 (SCI)

Signal Name	Type	State During[1]		Signal Description
		Reset	Stop	
RXD PE0	Input Input or Output	Input	Disconnected internally	**Serial Receive Data** Receives byte-oriented serial data and transfers it to the SCI receive shift register. **Port E 0** The default configuration following reset is GPIO. When configured as PE0, signal direction is controlled through the Port E Directions Register (PRRE). This signal is configured as RXD or PE0 through the Port E Control Register (PCRE).
TXD PE1	Output Input or Output	Input	Disconnected internally	**Serial Transmit Data** Transmits data from SCI transmit data register. **Port E 1** The default configuration following reset is GPIO. When configured as PE1, signal direction is controlled through the SCI PRRE. This signal is configured as TXD or PE1 through PCRE.
SCLK PE2	Input/Output Input or Output	Input	Disconnected internally	**Serial Clock** Provides the input or output clock used by the transmitter and/or the receiver. **Port E 2** The default configuration following reset is GPIO. For PE2, signal direction is controlled through the SCI PRRE. This signal is configured as SCLK or PE2 through PCRE.

Note: The Wait processing state does not affect the signal state.

2.11 Timers

Three identical and independent timers are implemented in the DSP56321. Each timer can use internal or external clocking and can either interrupt the DSP56321 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[1]		
		Reset	Stop	Signal Description
TIO0	Input or Output	Input	Disconnected internally	**Timer 0 Schmitt-Trigger Input/Output** As an external event counter or in Measurement mode, TIO0 is input. In Watchdog, Timer, or Pulse Modulation mode, TIO0 is output. The default mode after reset is GPIO input. This can be changed to output or configured as a Timer Input/Output through the Timer 0 Control/Status Register (TCSR0).
TIO1	Input or Output	Input	Disconnected internally	**Timer 1 Schmitt-Trigger Input/Output** As an external event counter or in Measurement mode, TIO1 is input. In Watchdog, Timer, or Pulse Modulation mode, TIO1 is output. The default mode after reset is GPIO input. This can be changed to output or configured as a Timer Input/Output through the Timer 1 Control/Status Register (TCSR1).
TIO2	Input or Output	Input	Disconnected internally	**Timer 2 Schmitt-Trigger Input/Output** As an external event counter or in Measurement mode, TIO2 is input. In Watchdog, Timer, or Pulse Modulation mode, TIO2 is output. The default mode after reset is GPIO input. This can be changed to output or configured as a Timer Input/Output through the Timer 2 Control/Status Register (TCSR2).

Note: The Wait processing state does not affect the signal state.

2.12 JTAG and OnCE Interface

The DSP56300 family and in particular the DSP56321 support circuit-board test strategies that are 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 provides a means to interface 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 the chapter on debugging support in the *DSP56300 Family Manual*.

Table 2-15. OnCE/JTAG Interface

Signal Name	Type	State During Reset	Signal Description
TCK	Input	Input	**Test Clock** A test clock input signal to synchronize the JTAG test logic.
TDI	Input	Input	**Test Data Input** 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** A test data serial output signal 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** Sequences 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** Initializes the test controller asynchronously. $\overline{\text{TRST}}$ has an internal pull-up resistor. $\overline{\text{TRST}}$ must be asserted after power up.
$\overline{\text{DE}}$	Input/ Output	Input	**Debug Event** As an input, provides a means of entering debug mode from an external command controller. As an output, provides a means of acknowledging that the chip has entered debug mode. Asserted as an input, $\overline{\text{DE}}$ causes the DSP56300 core to finish executing the current instruction, save the instruction pipeline information, enter debug mode, and wait for commands from the debug serial input line. This signal is asserted as an output for three clock cycles when the chip enters debug mode as a result of a debug request or a breakpoint condition. The $\overline{\text{DE}}$ has an internal pull-up resistor. $\overline{\text{DE}}$ is not a standard part of the JTAG TAP controller. The signal connects directly to the OnCE module to initiate debug mode directly or to provide a direct external indication that the chip has entered debug mode. All other interaction with the OnCE module must occur through the JTAG port.

Memory Configuration 3

Like all members of the DSP56300 core family, the DSP56321 can address three sets of 16 M × 24-bit memory internally: program, X data, and Y data. Each of these memory spaces includes both on-chip and external memory (accessed through the external memory interface). The DSP56321 is extremely flexible because it has several modes to allocate on-chip memory between the program memory and the two data memory spaces. **Section 3.1** provides a general description of the program memory space. **Section 3.2** and **Section 3.3** describe the X and Y data memory spaces. **Section 3.4** summarizes the available memory configurations. **Section 3.5** provides guidelines for changing the memory configuration dynamically.

3.1 Program Memory Space

Program memory space consists of the following:

- Internal program memory (program RAM, 32 K by default, up to 112 K)
- (Optional) instruction cache (1 K) taken from internal program RAM
- Bootstrap program ROM (192 × 24-bit)
- Optional off-chip memory expansion (as much as 64 K in 16-bit mode or 256 K in 24-bit mode using the 18 external address lines). Refer to the *DSP56300 Family Manual*, especially the Expansion Port chapter, for detailed information on using the external memory interface to access external program memory.

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

3.1.1 Internal Program Memory

The default on-chip program memory consists of a 24-bit-wide, high-speed, SRAM occupying the lowest 32 K locations ($0–$7FFF) in program memory space. The on-chip program RAM is organized in 32 banks with 1024 locations each. You can make additional program memory available using the memory switch mode described in **Section 3.1.2**, *Memory Switch Modes—Program Memory*.

3.1.2 Memory Switch Modes—Program Memory

Memory switch mode allows reallocation of portions of X and Y data RAM as program RAM. OMR[22, 21, and 7] are the memory switch (MSW[2–0]) bits that control this function. When MSW[2–0] = 000, program memory consists of the default 32 K × 24-bit memory space described in the previous section. In this default mode, the lowest external program memory location is $8000. When MSW[2–0] ≠ 000, portions of the X and Y data memory are reallocated to the program memory (see **Section 3.4** for details). In any of the Memory Switch modes, if the instruction cache is enabled (that is, if the CE bit in the Status Register is set), the lowest 1 K internal program words (locations $0–$3FF) are used as cache memory. With the cache enabled, the lowest 1 K of internal program memory is inaccessible and the address range is redirected to external program memory.

Note: When using an enabled instruction cache, do not use the vector address bus to access any address between P:$0–$3FF. (See the memory maps, starting with **Figure 3-1** on page -7.)

3.1.3 Program Bootstrap ROM

The program memory space occupying locations $FF0000–$FF00BF includes the internal bootstrap ROM. This ROM contains the 192-word DSP56321 bootstrap program.

3.2 X Data Memory Space

The X data memory space consists of the following:

- Internal X data memory (80 K by default down to 40 K)
- Internal X I/O space (upper 128 locations)
- Optional off-chip memory expansion (as much as 64 K in 16-bit mode, or 256 K in 24-bit mode using the 18 external address lines). Refer to the *DSP56300 Family Manual,* especially **Section 2**, *Expansion Port*, for details on using the external memory interface to access external X data memory.

Note: Do not access the reserved X memory space at locations $FF0000–$FFEFFF.

3.2.1 Internal X Data Memory

The default on-chip X data RAM is a 24-bit-wide, internal, static memory occupying the lowest 80 K locations ($0–$13FFF) in X memory space. The on-chip X data RAM is organized into 80 banks with 1024 locations each. Available X data memory space can be reallocated to program memory using the memory switch mode described in the next section.

3.2.2 Memory Switch Modes—X Data Memory

Memory switch mode reallocates of portions of X and Y data RAM as program RAM. OMR[22, 21, and 7] are the memory switch (MSW[2–0]) bits that control this function. When MSW[2–0] = 000, X data memory consists of the default 80 K × 24-bit memory space described in the previous section. In this default mode, the lowest external program memory location is $14000. When MSW[2–0] ≠ 000, portions of the X and Y data memory are reallocated to the program memory (see **Section 3.4** for details).

Note: The 12 K lowest locations ($0–$2FFF) of internal X memory are *shared memory* accessible by both the core and EFCOP. The EFCOP connects to the shared memory instead of the DMA bus, so the DMA cannot access shared memory. Simultaneous core and EFCOP accesses to the same shared memory bank (1024 locations) are not permitted. The programmer must prevent simultaneous accesses.

3.2.3 Internal X I/O Space

One part of the on-chip peripheral registers and some of the DSP56321 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). **Appendix B** lists the internal X I/O memory contents.

3.3 Y Data Memory Space

The Y data memory space consists of the following:

- Internal Y data memory (80 K by default down to 40 K)
- Internal Y I/O space (16 locations at $FFFF80–$FFFF8F)
- External Y I/O space (upper 112 locations)
- Optional off-chip memory expansion (up to 64 K in 16-bit mode or 256 K in 24-bit mode using the 18 external address lines). Refer to the *DSP56300 Family Manual* for details on using the external memory interface to access external Y data memory.

Note: Do not access the reserved Y memory space at locations $FF0000–$FFEFFF.

3.3.1 Internal Y Data Memory

The default on-chip Y data RAM is a 24-bit-wide, internal, static memory occupying the lowest 80 K locations ($0–$13FFF) in Y memory space. The on-chip Y data RAM is organized in 24 banks, 1024 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.

3.3.2 Memory Switch Modes—Y Data Memory

Memory switch mode reallocates of portions of X and Y data RAM as program RAM. OMR[22, 21, and 7] are the memory switch (MSW[2–0]) bits that control this function. When MSW[2–0] = 000, Y data memory consists of the default 80 K × 24-bit memory space described in the previous section. In this default mode, the lowest external program memory location is $14000. When MSW[2–0] ≠ 000, portions of the X and Y data memory are reallocated to the program memory (see **Section 3.4** for details).

Note: The 12 K lowest locations ($0–$2FFF) of internal Y memory are *shared memory* accessible by both the core and EFCOP. The EFCOP connects to the shared memory instead of the DMA bus, so the DMA cannot access shared memory. Simultaneous core and EFCOP accesses to the same shared memory bank (1024 locations) are not permitted. The programmer must prevent simultaneous accesses.

3.3.3 Internal Y I/O Space

The second part of the on-chip peripheral registers occupies 9 locations ($FFFFB0–$FFFFB8) of the Y data memory and contains the registers supporting the EFCOP. This area is 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 B**.

3.3.4 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 the *external Y I/O space*.

3.4 Summary of Memory Switch Mode Configurations

Table 3-1 summarizes the various combinations of memory switch modes and instruction cache

Table 3-1. DSP56321 Switch Memory Configuration

Program RAM Size	Instruction Cache Size	X Data RAM Size*	Y Data RAM Size*	Instruction Cache (CE)	MSW2	MSW1	MSW0
32 K × 24-bit	0	80 K × 24-bit	80 K × 24-bit	disabled	0	0	0
31 K × 24-bit	1024 × 24-bit	80 K × 24-bit	80 K × 24-bit	enabled	0	0	0
40 K × 24-bit	0	76 K × 24-bit	76 K × 24-bit	disabled	0	0	1
39 K × 24-bit	1024 × 24-bit	76 K × 24-bit	76 K × 24-bit	enabled	0	0	1
48 K × 24-bit	0	72 K × 24-bit	72 K × 24-bit	disabled	0	1	0

Table 3-1. DSP56321 Switch Memory Configuration (Continued)

Program RAM Size	Instruction Cache Size	X Data RAM Size*	Y Data RAM Size*	Instruction Cache (CE)	MSW2	MSW1	MSW0
47 K × 24-bit	1024 × 24-bit	72 K × 24-bit	72 K × 24-bit	enabled	0	1	0
64 K × 24-bit	0	64 K × 24-bit	64 K × 24-bit	disabled	0	1	1
63 K × 24-bit	1024 × 24-bit	64 K × 24-bit	64 K × 24-bit	enabled	0	1	1
72 K × 24-bit	0	60 K × 24-bit	60 K × 24-bit	disabled	1	0	0
71 K × 24-bit	1024 × 24-bit	60 K × 24-bit	60 K × 24-bit	enabled	1	0	0
80 K × 24-bit	0	56 K × 24-bit	56 K × 24-bit	disabled	1	0	1
79 K × 24-bit	1024 × 24-bit	56 K × 24-bit	56 K × 24-bit	enabled	1	0	1
96 K × 24-bit	0	48 K × 24-bit	48 K × 24-bit	disabled	1	1	0
95 K × 24-bit	1024 × 24-bit	48 K × 24-bit	48 K × 24-bit	enabled	1	1	0
112 K × 24-bit	0	40 K × 24-bit	40 K × 24-bit	disabled	1	1	1
111 K × 24-bit	1024 × 24-bit	40 K × 24-bit	40 K × 24-bit	enabled	1	1	1
*Includes 12 K × 24-bit shared memory (that is, memory shared by the DSP56300 core and the EFCOP)							

3.5 Dynamic Memory Configuration Switching

When 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. 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. A switch can be accomplished just by changing the OMR[MS/MSW] bits 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.

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.

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

Pay special attention 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 MSW bits change while the DSP is in Debug mode. As a result, subsequent instructions may be fetched according to the new memory configuration (after the switch) and thus may execute improperly.

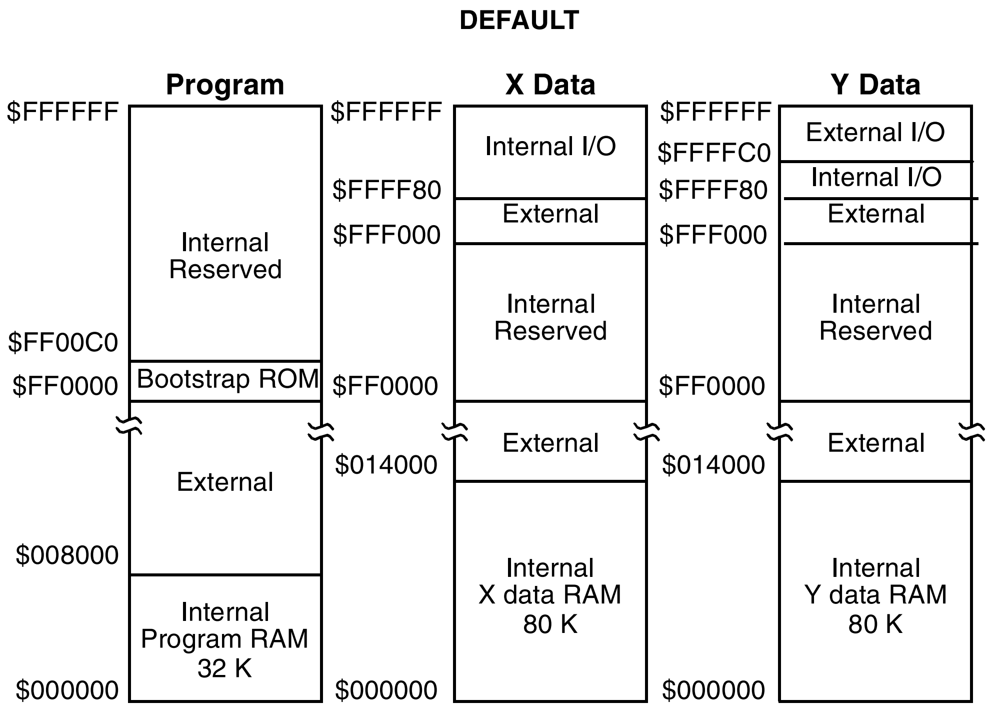
3.6 Sixteen-Bit Compatibility Mode Configuration

The sixteen-bit compatibility (SC) mode allows the DSP56321 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. Refer to the memory maps in the following section for details about internal memory allocation in Sixteen-Bit Compatibility Mode.

3.7 Memory Maps

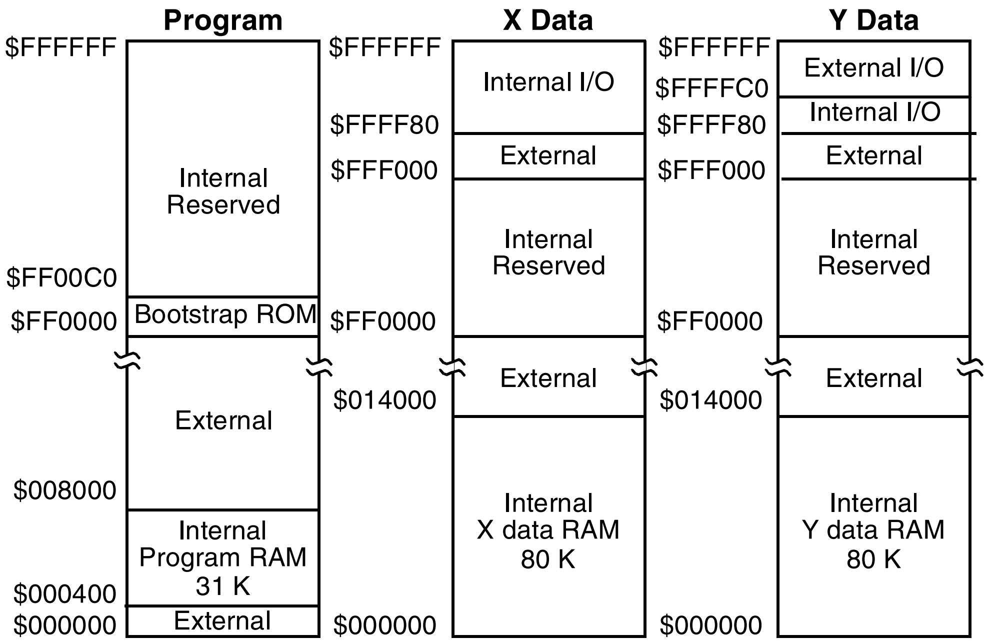
The following figures illustrate each of the memory space and RAM configurations defined by the settings of the MSW[2–0], CE, and SC bits. The figures show the configuration and describe the bit settings, memory sizes, and memory locations.

Note: In 16-bit mode, if MSW[2–0] ≠ 000, very little external program memory is accessible, so enabling the instruction cache is of little benefit. In addition, certain 16-bit modes prevent you from accessing the entire internal DSP56321 memory space.



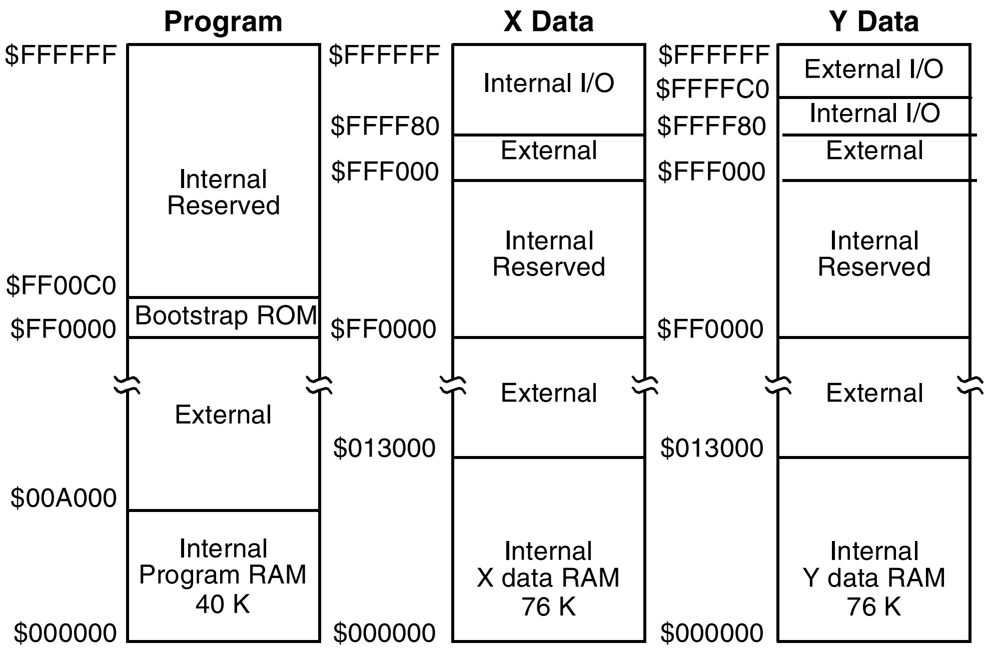
Bit Settings					**Internal Memory Configuration**				
MS2	**MS1**	**MS0**	**CE**	**SC**	**Program RAM**	**X Data RAM**	**Y Data RAM**	**Cache**	**Addressable Memory Size**
0	0	0	0	0	32 K $0000–$7FFF	80 K $0000–$13FFF	80 K $0000–$13FFF	None	16 M
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-1. MSW[2–0] = 000, Cache Off, 24-Bit Mode



Bit Settings					**Internal Memory Configuration**				
MS2	**MS1**	**MS0**	**CE**	**SC**	**Program RAM**	**X Data RAM**	**Y Data RAM**	**Cache**	**Addressable Memory Size**
0	0	0	1	0	31 K $0400–$7FFF	80 K $0000–$13FFF	80 K $0000–$13FFF	1 K internal not accessible	16 M
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

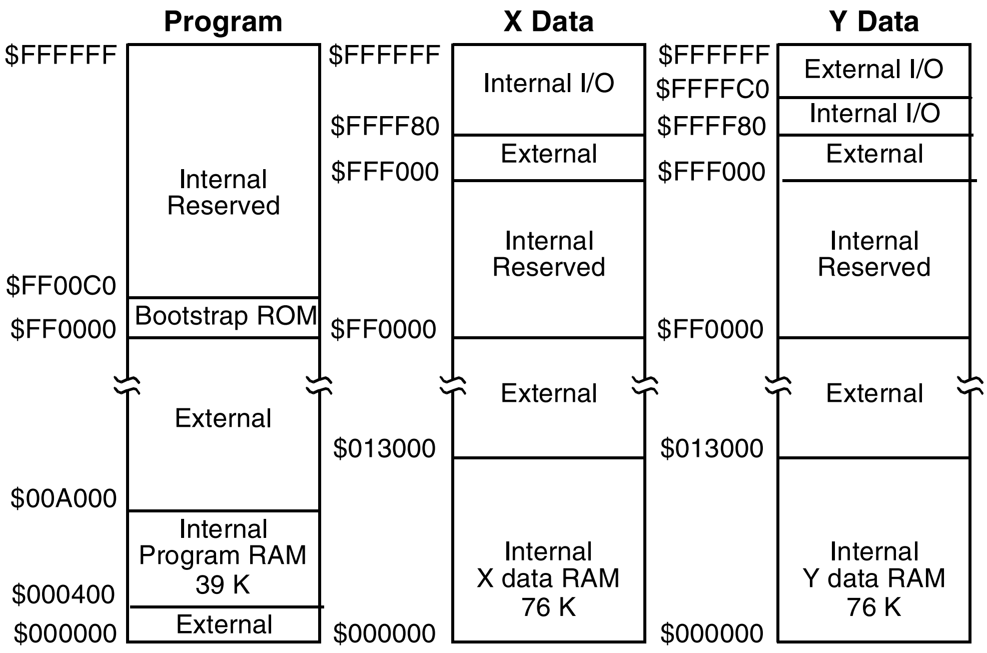
Figure 3-2. MSW[2–0] = 000, Cache On, 24-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	1	0	0	40 K $0000–$9FFF	76 K $0000–$12FFF	76 K $0000–$12FFF	None	16 M

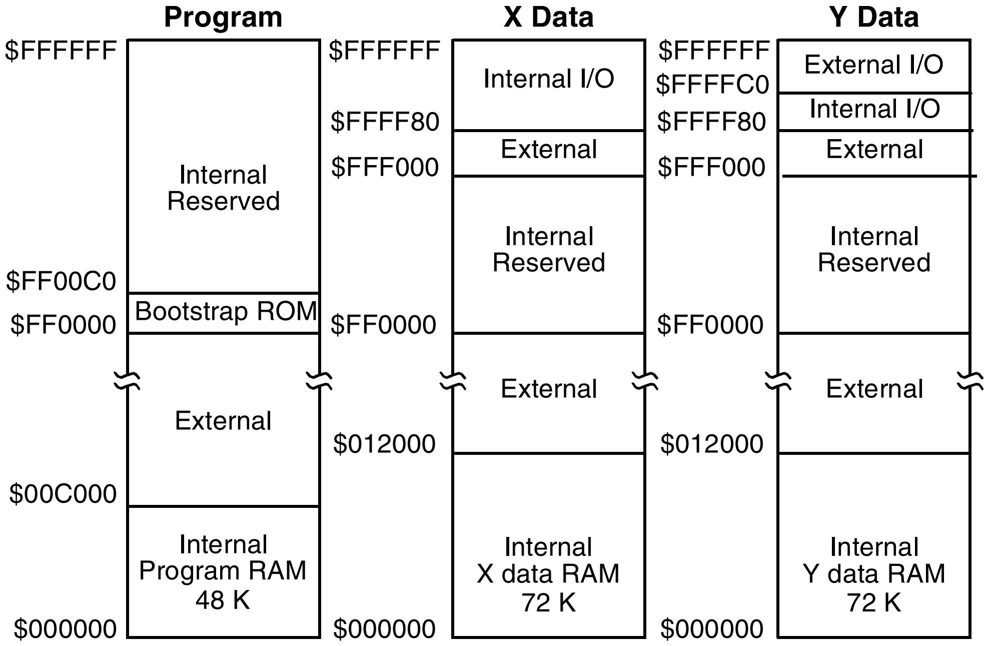
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.

Figure 3-3. MSW[2–0] = 001, Cache Off, 24-Bit Mode



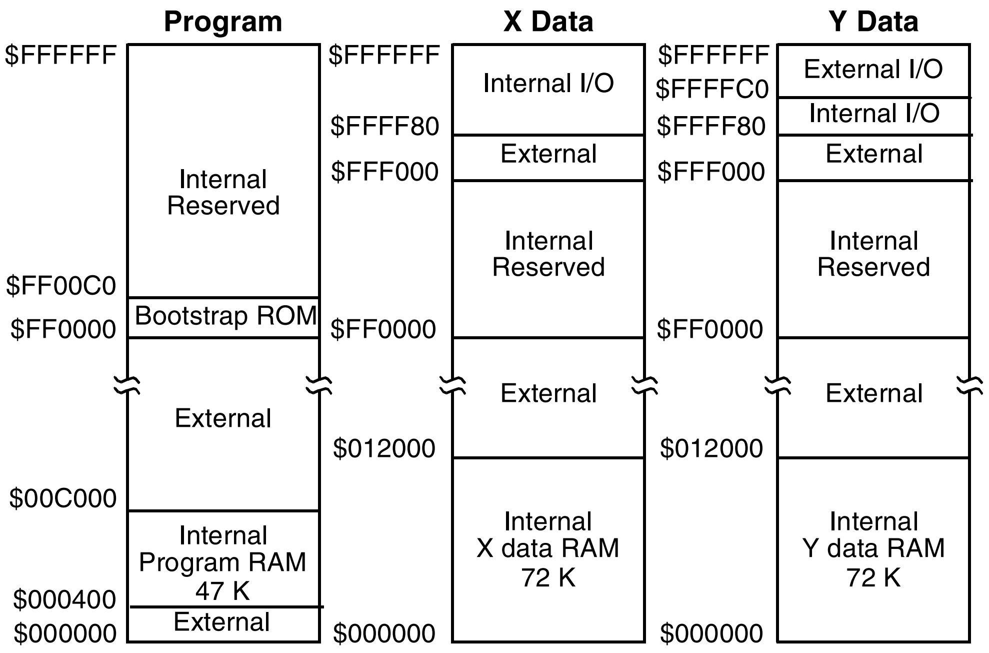
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	1	1	0	39 K $0400–$9FFF	76 K $0000–$12FFF	76 K $0000–$12FFF	1 K internal not accessible	16 M
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-4. MSW[2–0] = 001, Cache On, 24-Bit Mode



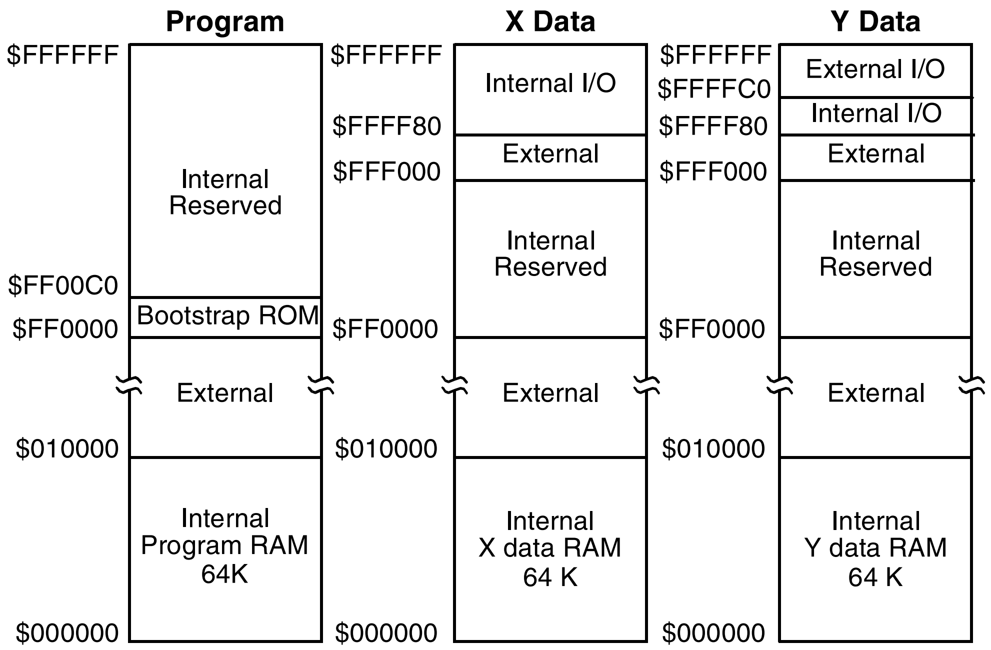
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	0	0	0	48 K $0000–$BFFF	72 K $0000–$11FFF	72 K $0000–$11FFF	None	16 M
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-5. MSW[2–0] = 010, Cache Off, 24-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	0	1	0	47 K $0400–$BFFF	72 K $0000–$11FFF	72 K $0000–$11FFF	1 K internal not accessible	16 M
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

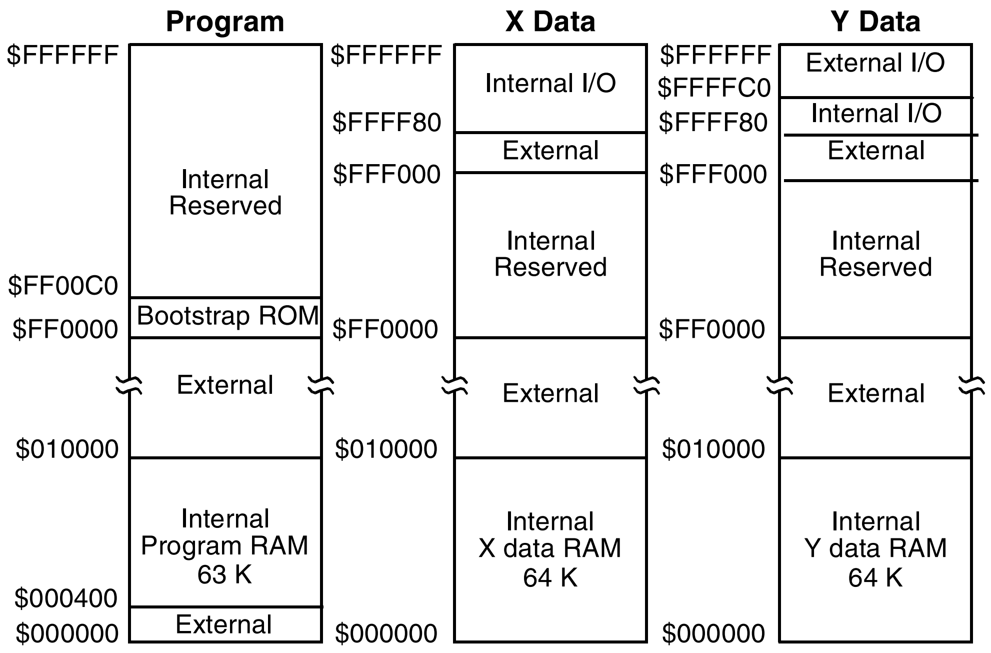
Figure 3-6. MSW[2–0] = 010, Cache On, 24-Bit Mode



Bit Settings					Internal Memory Configuration					
MS2	**MS1**	**MS0**	**CE**	**SC**	**Program RAM**	**X Data RAM**	**Y Data RAM**	**Cache**	**Addressable Memory Size**
0	1	1	0	0	64 K $0000–$FFFF	64 K $0000–$FFFF	64 K $0000–$FFFF	None	16 M

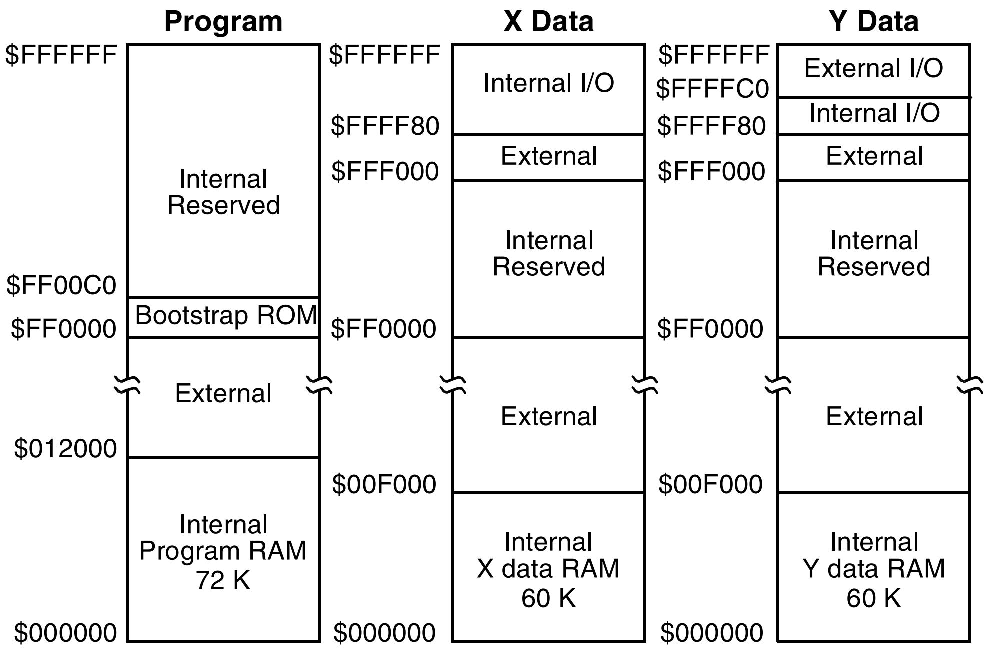
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.

Figure 3-7. MSW[2–0] = 011, Cache Off, 24-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	**MS1**	**MS0**	**CE**	**SC**	**Program RAM**	**X Data RAM**	**Y Data RAM**	**Cache**	**Addressable Memory Size**
0	1	1	1	0	63 K $0400–$FFFF	64 K $0000–$FFFF	64 K $0000–$FFFF	1 K internal not accessible	16 M
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

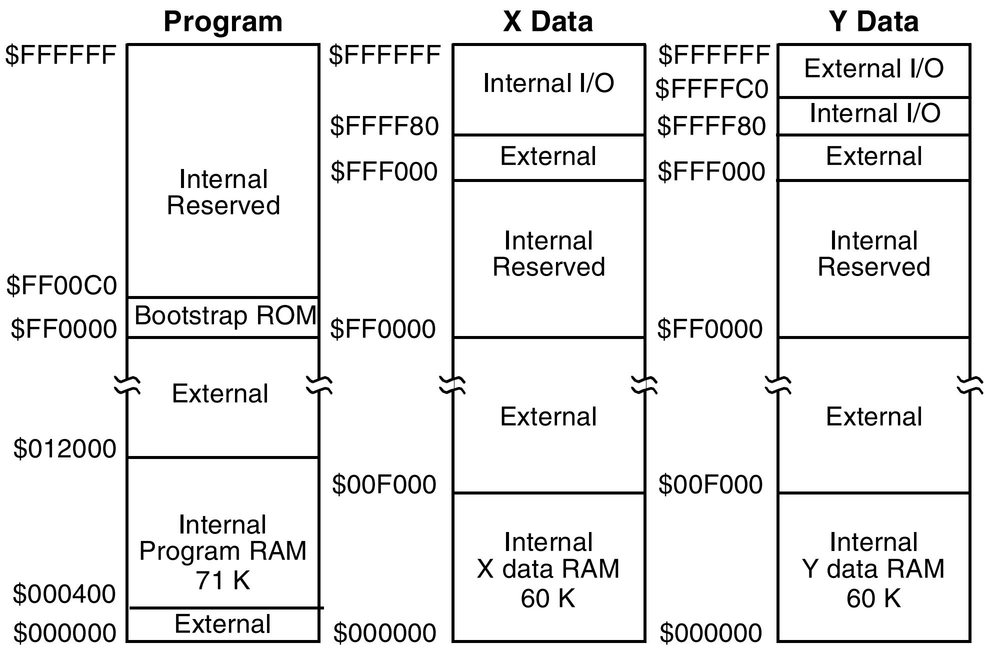
Figure 3-8. MSW[2–0] = 011, Cache On 24-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	0	0	0	72 K $0000–$11FFF	60 K $0000–$EFFF	60 K $0000–$EFFF	None	16 M

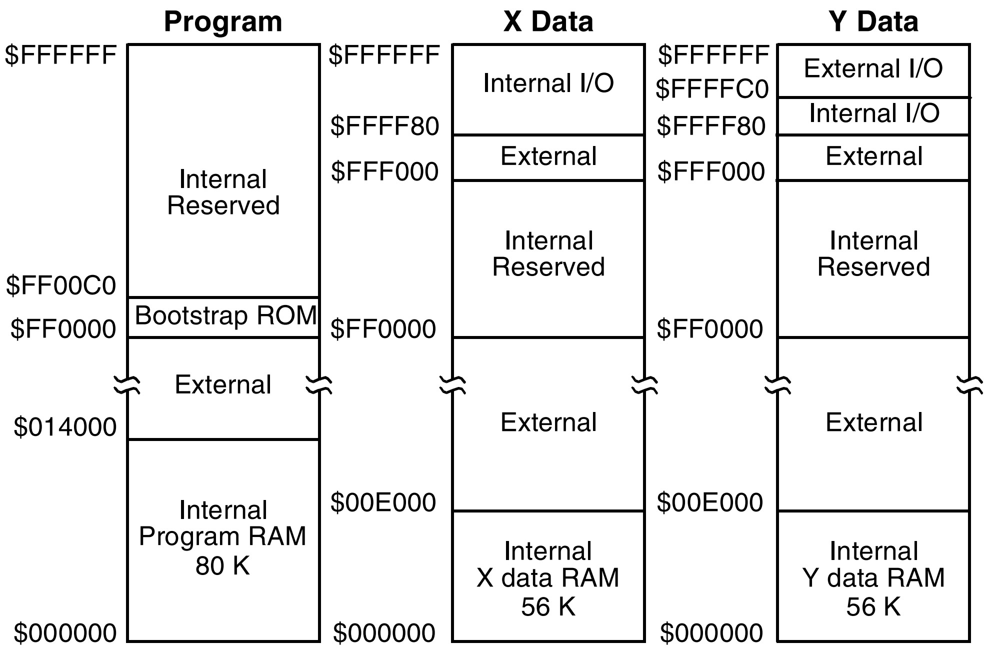
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.

Figure 3-9. MSW[2–0] = 100, Cache Off, 24-Bit Mode



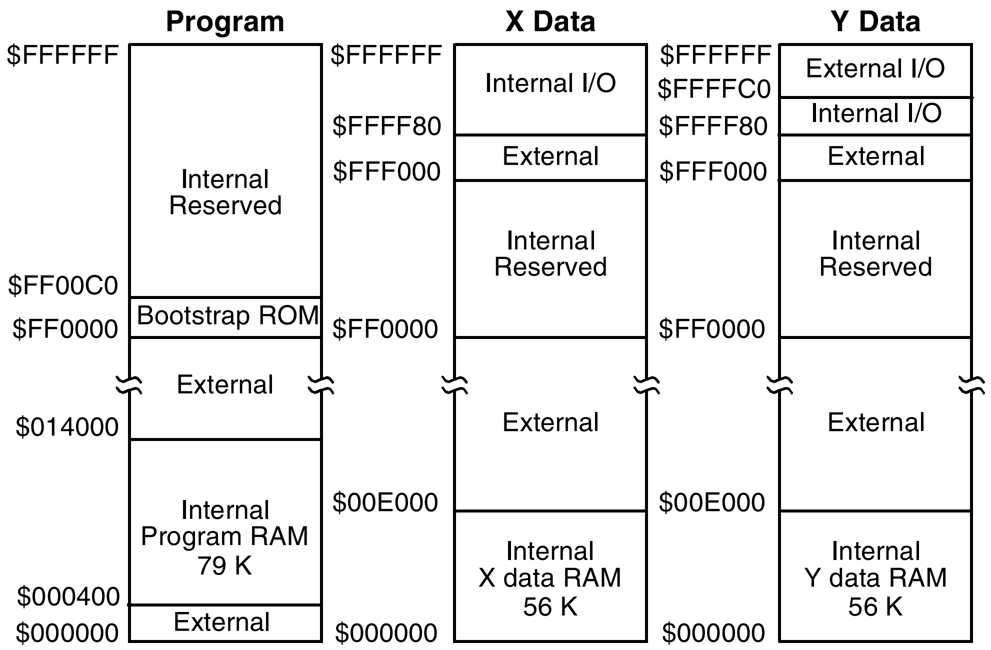
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	0	1	0	71 K $0400–$11FFF	60 K $0000–$EFFF	60 K $0000–$EFFF	1 K internal not accessible	16 M
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-10. MSW[2–0] = 100, Cache On 24-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	1	0	0	80 K $0000–$13FFF	56 K $0000–$DFFF	56 K $0000–$DFFF	None	16 M
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

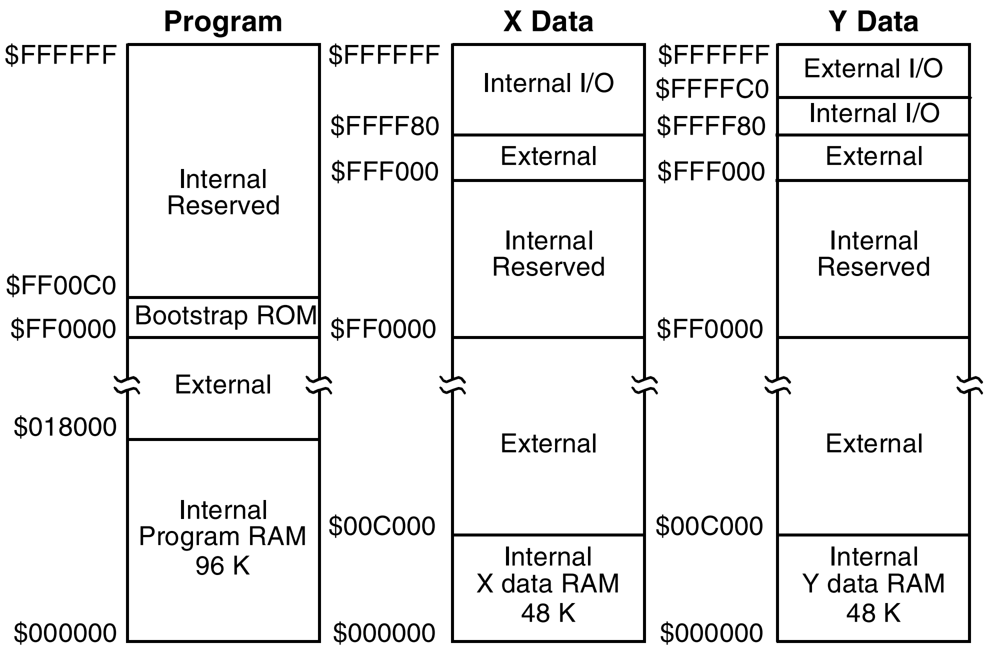
Figure 3-11. MSW[2–0] = 101, Cache Off, 24-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	**MS1**	**MS0**	**CE**	**SC**	**Program RAM**	**X Data RAM**	**Y Data RAM**	**Cache**	**Addressable Memory Size**
1	0	1	1	0	79 K $0400–$13FFF	56 K $0000–$DFFF	56 K $0000–$DFFF	1 K internal not accessible	16 M

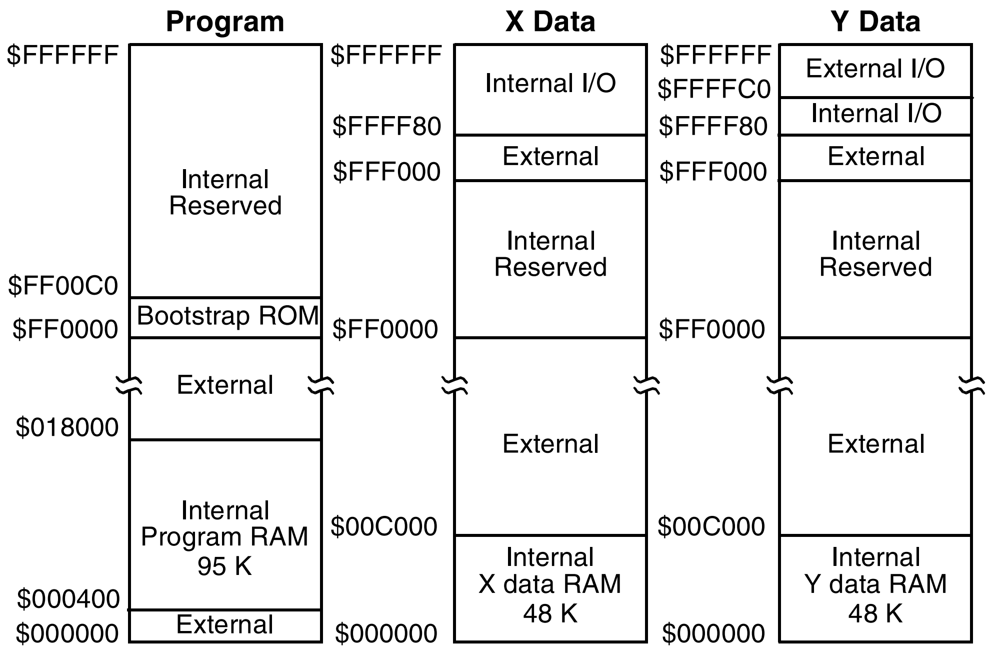
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.

Figure 3-12. MSW[2–0] = 101, Cache On, 24-Bit Mode



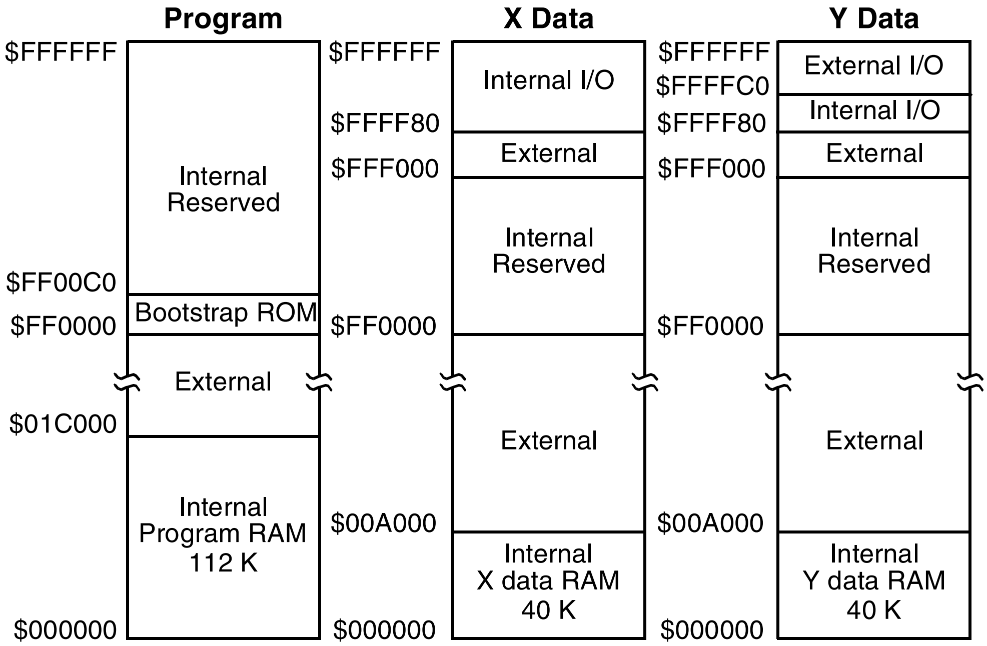
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	0	0	0	96 K $0000–$17FFF	48 K $0000–$BFFF	48 K $0000–$BFFF	None	16 M
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-13. MSW[2–0] = 110, Cache Off, 24-Bit Mode



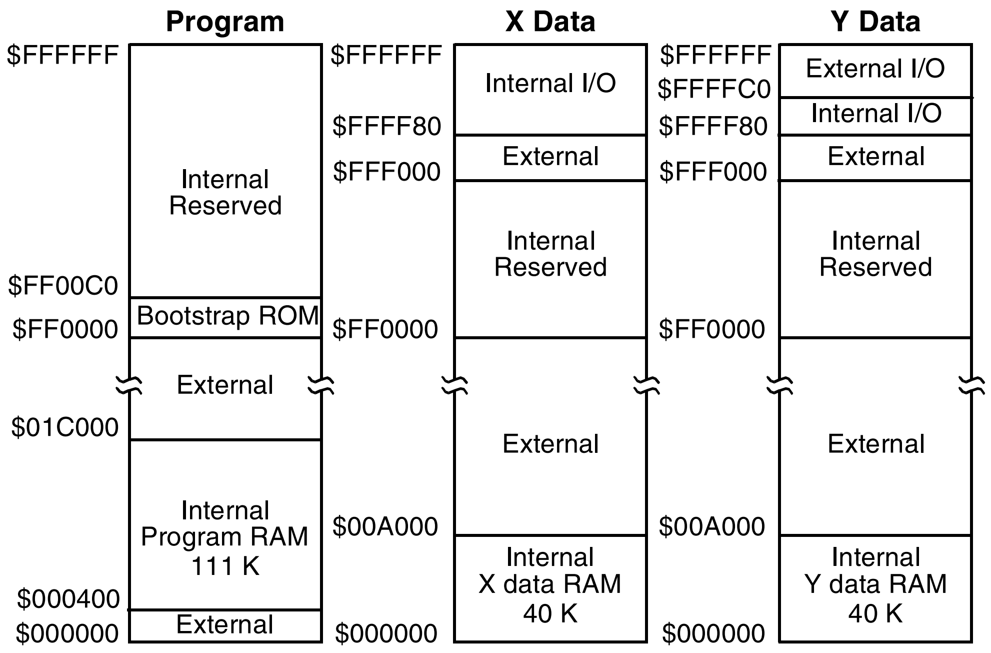
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	0	0	0	95 K $0400–$17FFF	48 K $0000–$BFFF	48 K $0000–$BFFF	1 K internal not accessible	16 M
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-14. MSW[2–0] = 110, Cache On, 24-Bit Mode



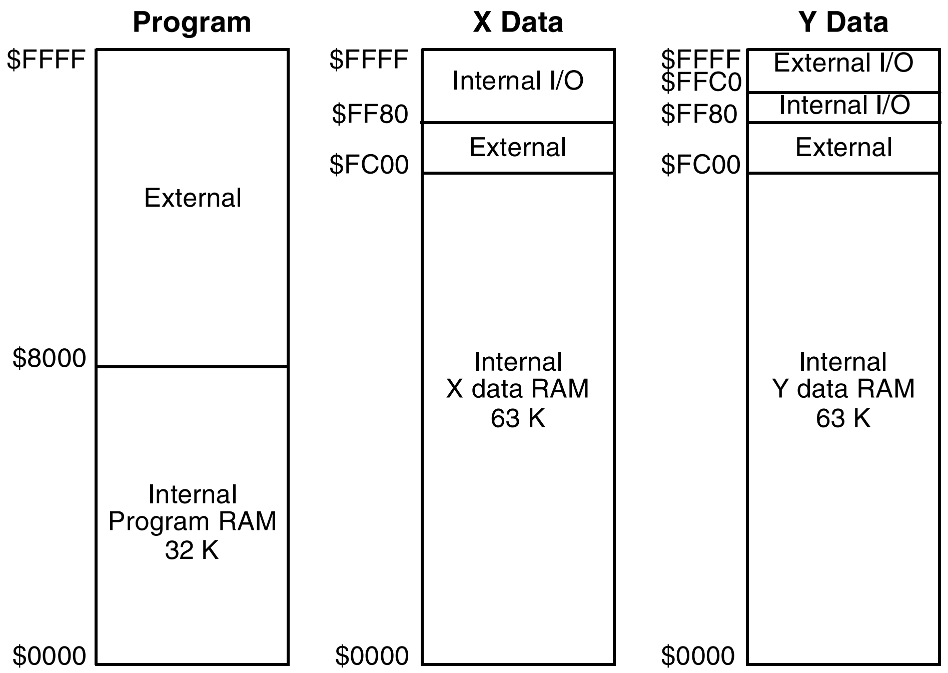
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	1	0	0	112 K $0000–$1BFFF	40 K $0000–$9FFF	40 K $0000–$9FFF	None	16 M
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-15. MSW[2–0] = 111, Cache Off, 24-Bit Mode



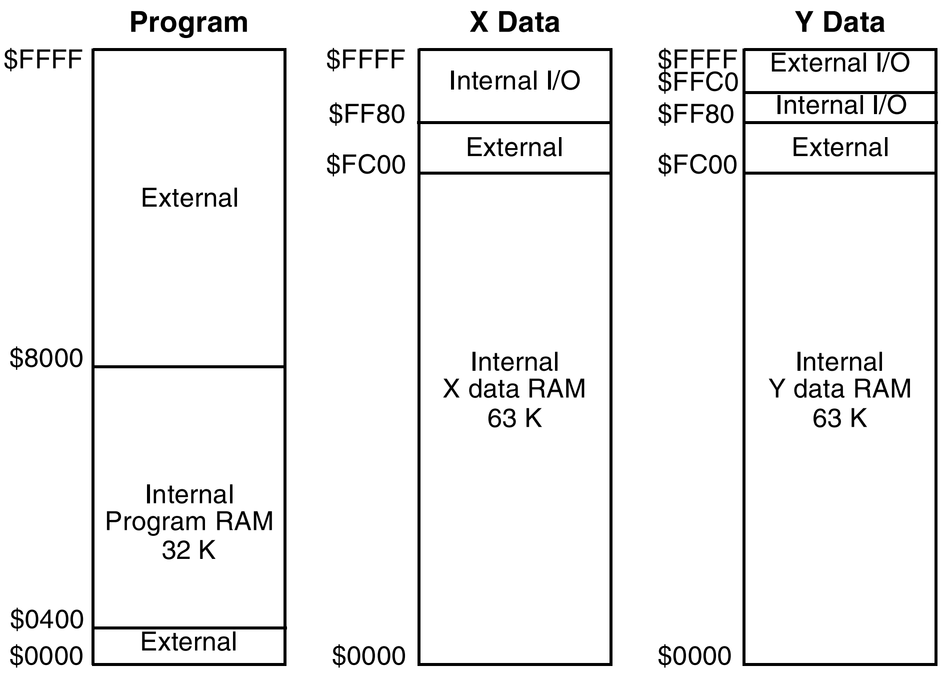
Bit Settings					**Internal Memory Configuration**				
MS2	**MS1**	**MS0**	**CE**	**SC**	**Program RAM**	**X Data RAM**	**Y Data RAM**	**Cache**	**Addressable Memory Size**
1	1	1	1	0	111 K $0400–$1BFFF	40 K $0000–$9FFF	40 K $0000–$9FFF	1 K internal not accessible	16 M
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-16. MSW[2–0] = 111, Cache On, 24-Bit Mode



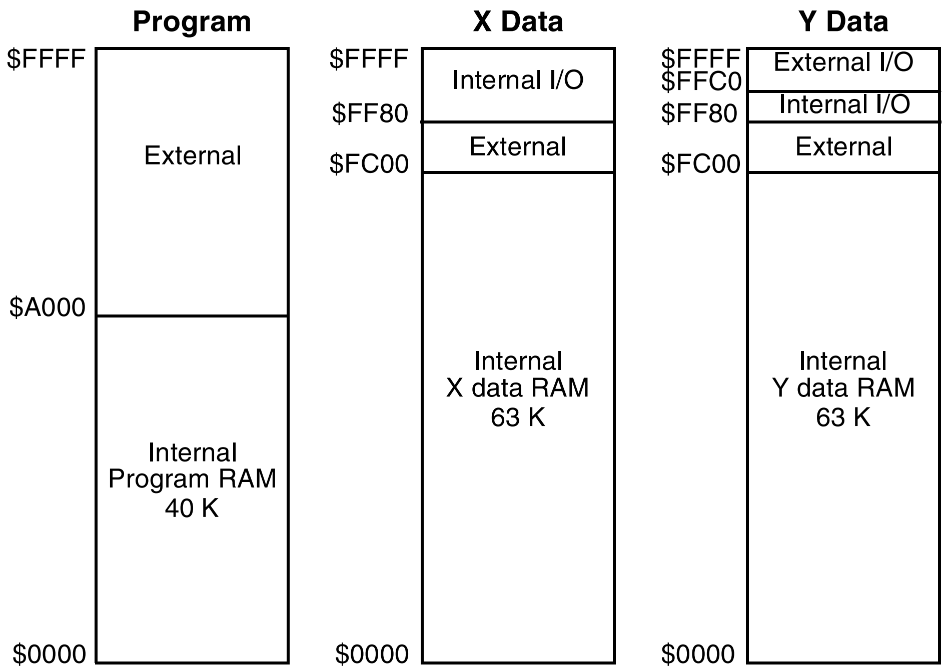
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	0	0	1	32 K $0000–$7FFF	63 K $0000–$FBFF	63 K $0000–$FBFF	None	64 K
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-17. MSW[2–0] = 000, Cache Off, 16-Bit Mode



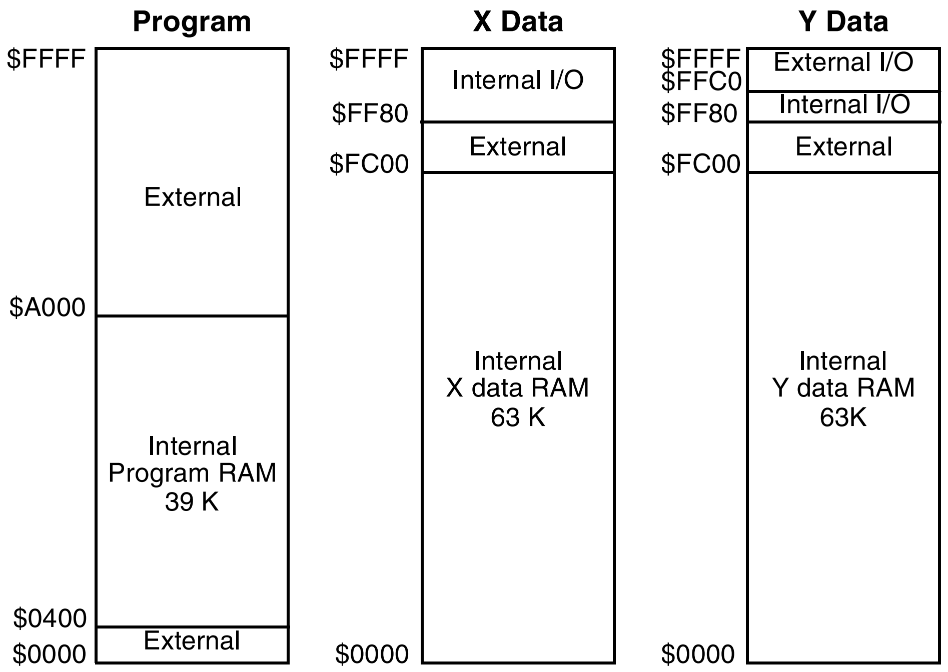
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	0	1	1	31 K $0400–$7FFF	63 K $0000–$FBFF	63 K $0000–$FBFF	1 K internal not accessible	64 K
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-18. MSW[2–0] = 000, Cache On, 16-Bit Mode



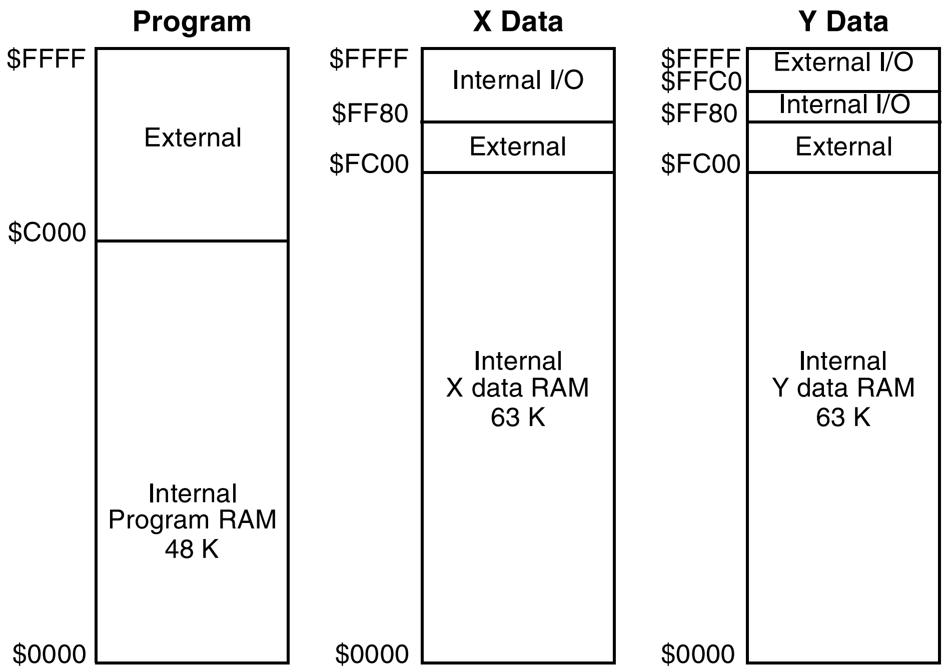
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	1	0	1	40 K $0000–$9FFF	63 K $0000–$FBFF	63 K $0000–$FBFF	None	64 K
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-19. MSW[2–0] = 001, Cache Off, 16-Bit Mode



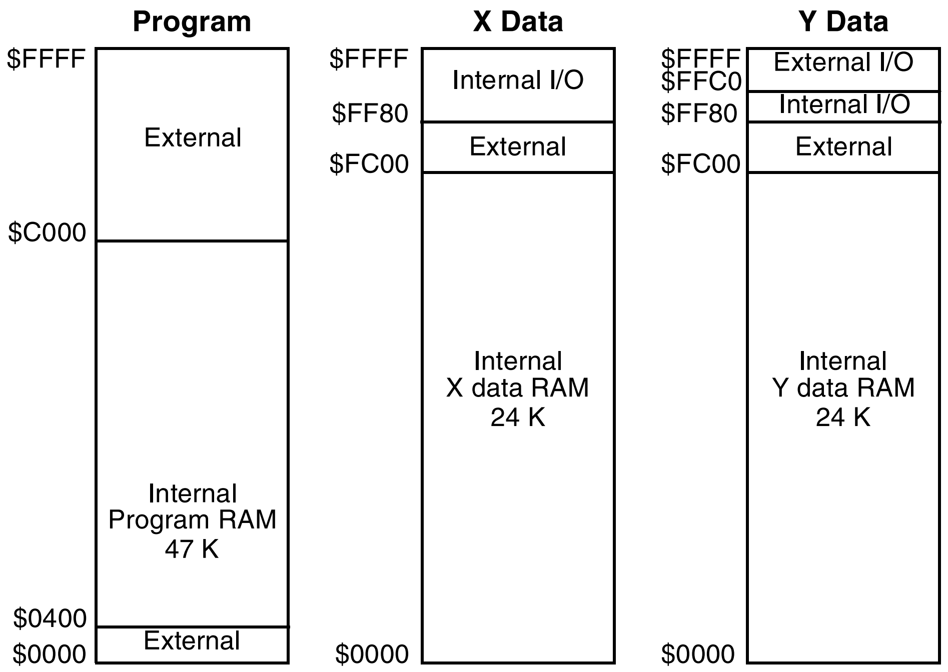
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	0	1	1	1	39 K $0400–$9FFF	63 K $0000–$FBFF	63 K $0000–$FBFF	1 K internal not accessible	64 K
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-20. MSW[2–0] = 001, Cache On, 16-Bit Mode



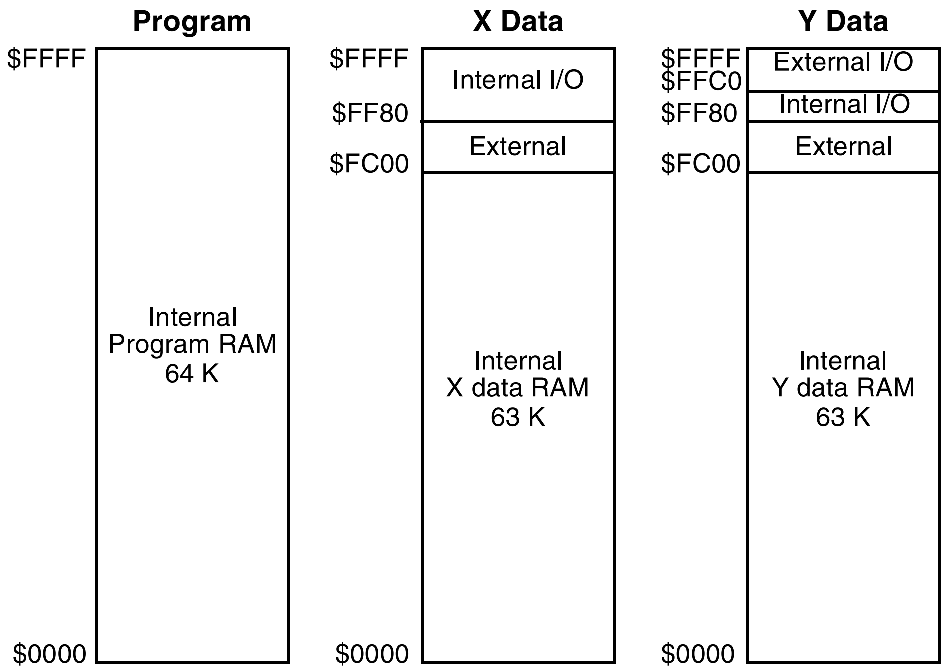
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	0	0	1	48 K $0000–$BFFF	63 K $0000–$FBFF	63 K $0000–$FBFF	None	64 K
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-21. MSW[2–0] = 010, Cache Off, 16-Bit Mode



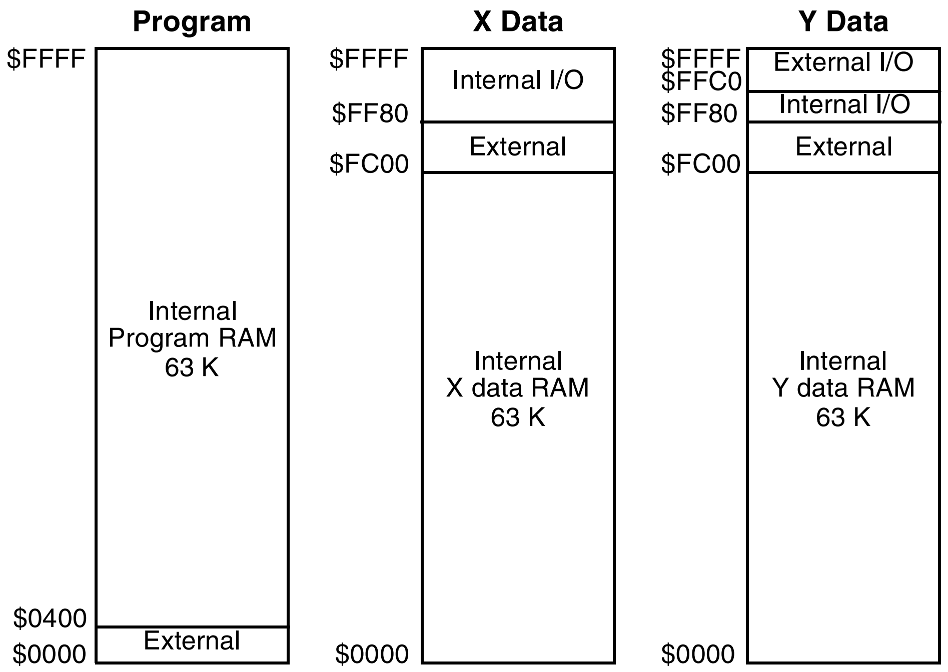
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	0	1	1	47 K $0400–$BFFF	63 K $0000–$FBFF	63 K $0000–$FBFF	1 K internal not accessible	64 K
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-22. MSW[2–0] = 010, Cache On, 16-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	1	0	1	64 K $0000–$FFFF	63 K $0000–$FBFF	63 K $0000–$FBFF	None	64 K
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

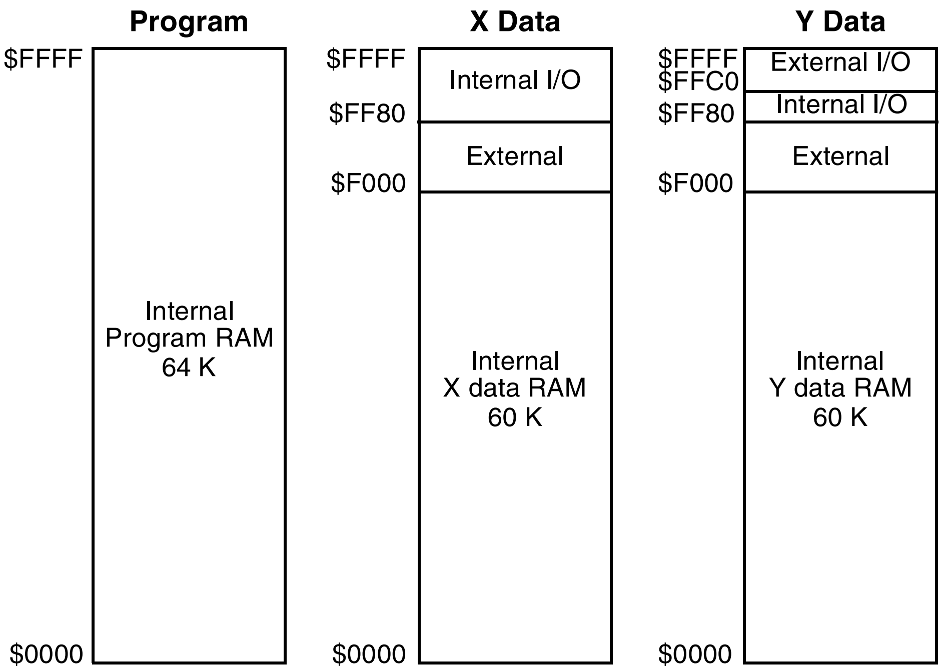
Figure 3-23. MSW[2–0] = 011, Cache Off, 16-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
0	1	1	1	1	63 K $0400–$FFFF	63 K $0000–$FBFF	63 K $0000–$FBFF	1 K internal not accessible	64 K

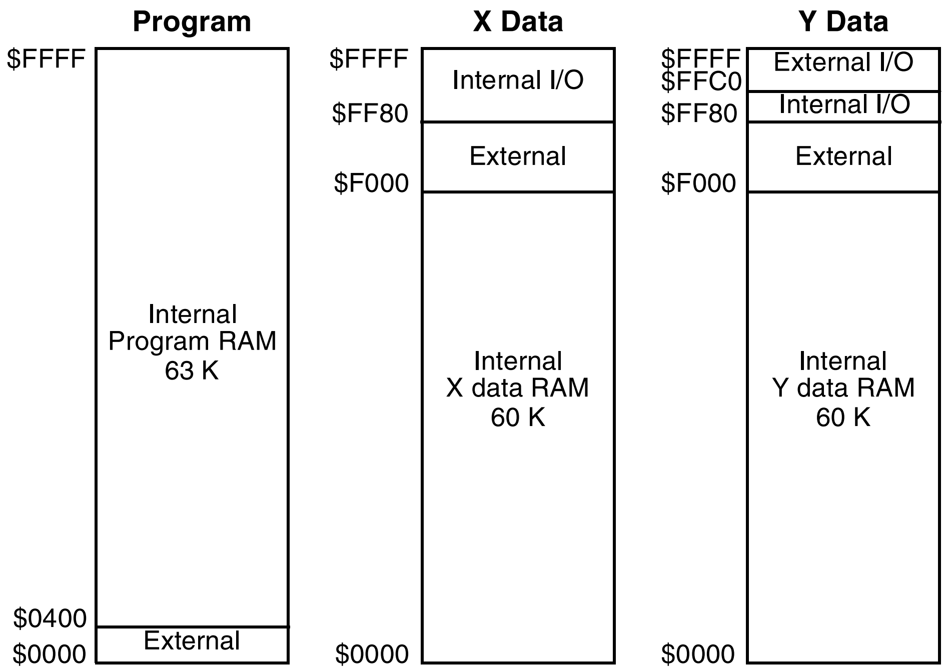
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.

Figure 3-24. MSW[2–0] = 011, Cache On, 16-Bit Mode



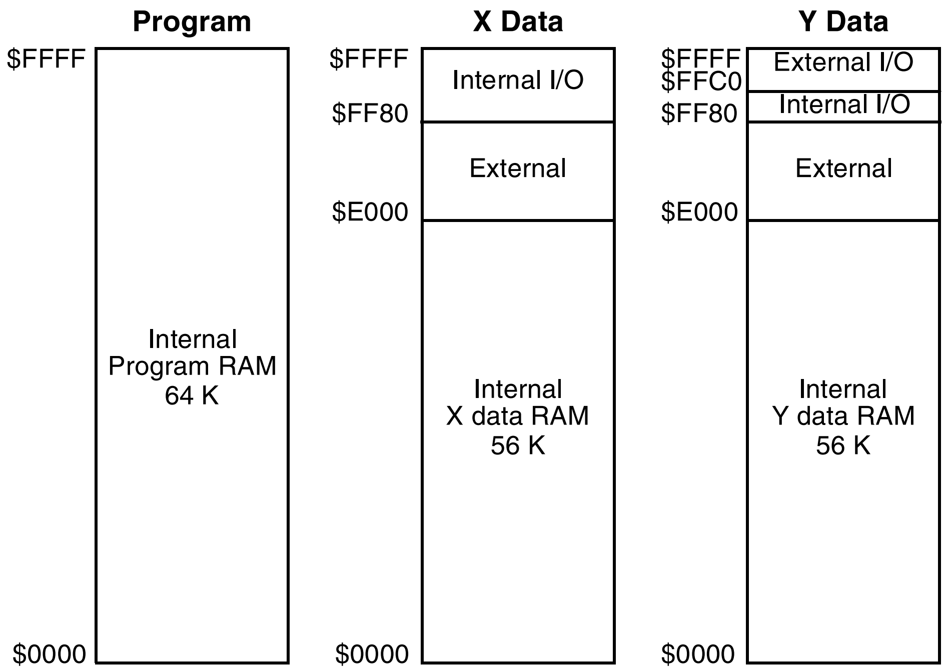
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	0	0	1	64 K $0000–$FFFF	60 K $0000–$EFFF	60 K $0000–$EFFF	None	64 K
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-25. MSW[2–0] = 100, Cache Off, 16-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	0	1	1	63 K $0400–$FFFF	60 K $0000–$EFFF	60 K $0000–$EFFF	1 K internal not accessible	64 K
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

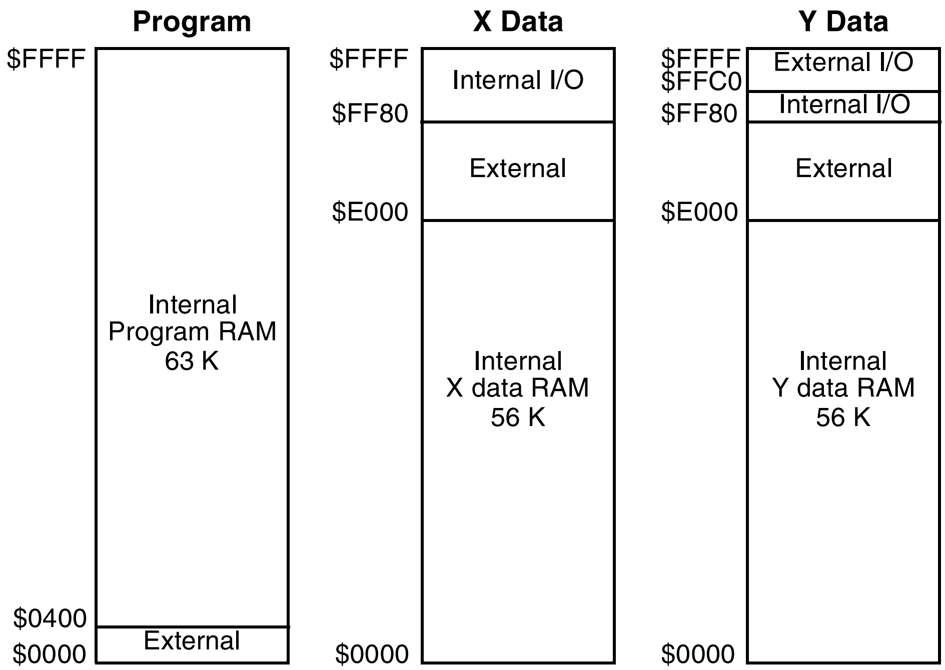
Figure 3-26. MSW[2–0] = 100, Cache On, 16-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	1	0	1	64 K $0000–$FFFF	56 K $0000–$DFFF	56 K $0000–$DFFF	None	64 K

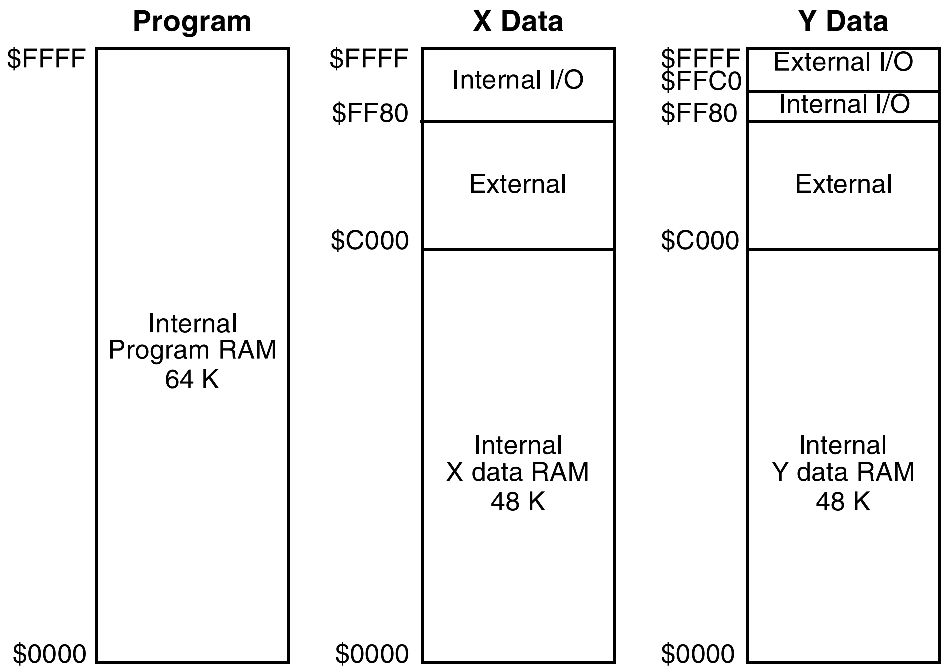
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.

Figure 3-27. MSW[2–0] = 101, Cache Off, 16-Bit Mode



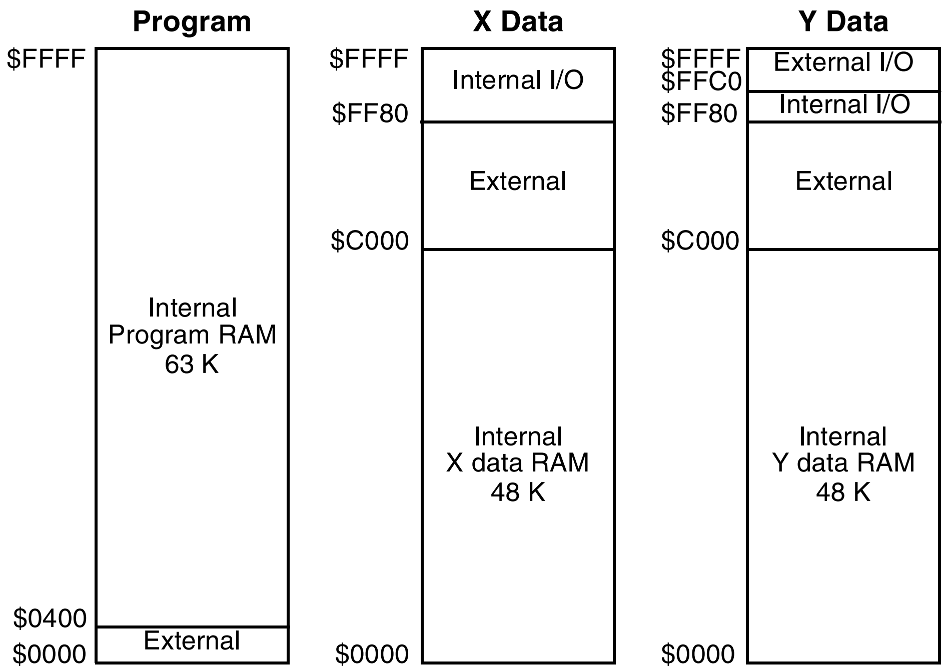
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	0	1	1	1	63 K $0400–$FFFF	56 K $0000–$DFFF	56 K $0000–$DFFF	1 K internal not accessible	64 K
Note: Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.									

Figure 3-28. MSW[2–0] = 101, Cache On, 16-Bit Mode



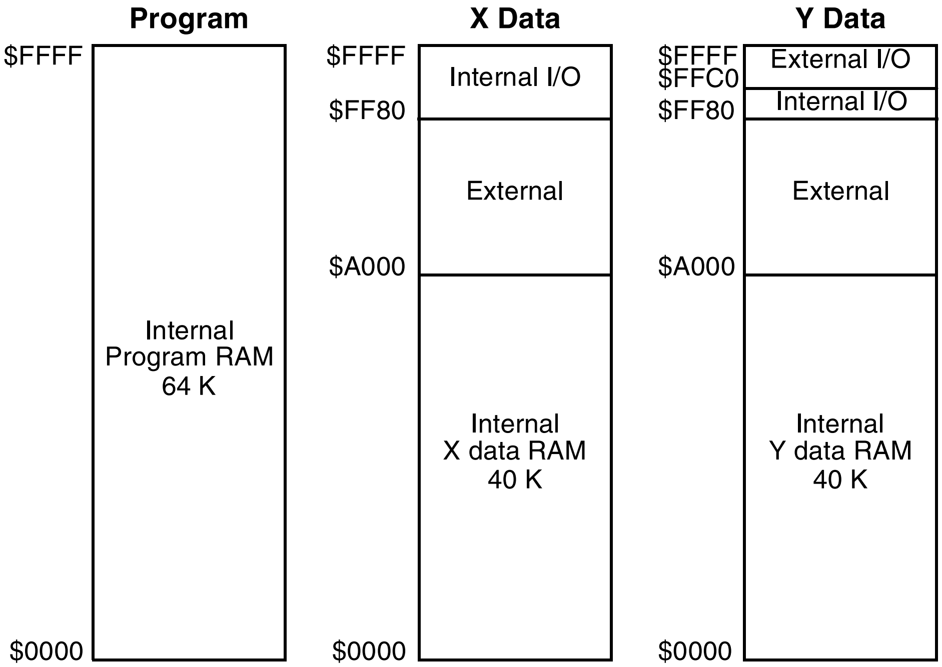
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	0	0	1	64 K $0000–$FFFF	48 K $0000–$BFFF	48 K $0000–$BFFF	None	64 K
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-29. MSW[2–0] = 110, Cache Off, 16-Bit Mode



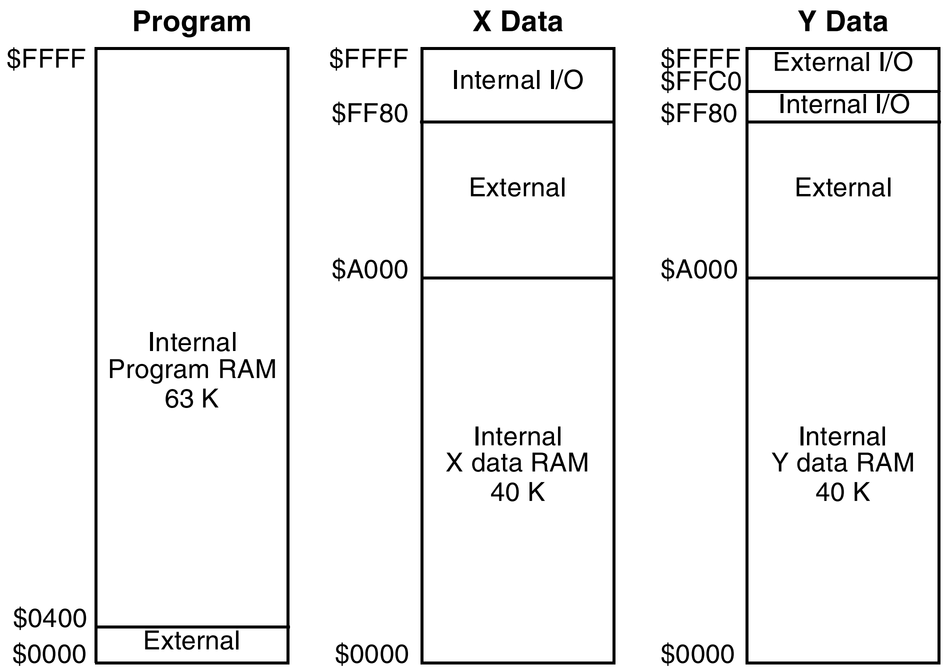
Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	0	1	1	63 K $0400–$FFFF	48 K $0000–$BFFF	48 K $0000–$BFFF	1 K internal not accessible	64 K
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-30. MSW[2–0] = 110, Cache On, 16-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	1	0	1	64 K $0000–$FFFF	40 K $0000–$9FFF	40 K $0000–$9FFF	None	64 K
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-31. MSW[2–0] = 111, Cache Off, 16-Bit Mode



Bit Settings					Internal Memory Configuration				
MS2	MS1	MS0	CE	SC	Program RAM	X Data RAM	Y Data RAM	Cache	Addressable Memory Size
1	1	1	1	1	63 K $0400–$FFFF	40 K $0000–$9FFF	40 K $0000–$9FFF	1 K internal not accessible	64 K
Note:	Lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory accessible by the core and EFCOP but not by the DMA controller.								

Figure 3-32. MSW[2–0] = 111, Cache On, 16-Bit Mode

Core Configuration 4

This chapter presents DSP56300 core configuration details specific to the DSP56321. These configuration details include the following:

- Operating modes
- Bootstrap program
- Central Processor registers
  - Status register (SR)
  - Operating mode register (OMR)
- Interrupt Priority Registers (IPRC and IPRP)
- Bus Interface Unit registers
  - Bus Control Register (BCR)
  - Address Attribute Registers (AAR[3–0])
- DMA Control Registers 5–0 (DCR[5–0])
- Device identification register (IDR)
- JTAG identification register
- JTAG boundary scan register (BSR)

For information on specific registers or modules in the DSP56300 core, refer to the *DSP56300 Family Manual*.

4.1 Operating Modes

The DSP56321 begins operation by leaving the Reset state and going into one of eight operating modes. As the DSP56321 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 in turn determines the bootstrap program option the chip uses to start up.

Software can also directly set the OMR[MA–MD] bits. A jump directly to the bootstrap program entry point ($FF0000) after the OMR bits are set causes the DSP56321 to execute the specified bootstrap program option (except modes 0 and 8). **Table 4-1** shows the DSP56321 bootstrap operation modes, the corresponding settings of the external operational mode signal lines (the OMR[MA–MD] bits), and the reset vector address to which the DSP56321 jumps once it leaves the Reset state.

Table 4-1. DSP56321 Operating Modes

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
0	0	0	0	0	$C00000	**Expanded mode** Bypasses the bootstrap ROM, and the DSP56321 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). Address $C00000 is reflected as address $00000 on Port A signals A[0–17].
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** Bypasses the bootstrap ROM, and the DSP56321 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.
9	1	0	0	1	$FF0000	**Bootstrap from byte-wide memory** The bootstrap program loads instructions through Port A from external byte-wide memory, starting at P:$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 A[0–17]. The boot program concatenates every 3 bytes read from the external memory into a 24-bit wide DSP56321 word.
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 DSP56321 word.
B	1	0	1	1	$FF0000	Reserved

Table 4-1. DSP56321 Operating Modes (Continued)

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
C	1	1	0	0	$FF0000	**HI08 bootstrap in ISA/DSP563xx mode** The HI08 is configured to interface with an ISA bus or with the memory expansion port of a master DSP563xx 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 the DSP56300 core.
D	1	1	0	1	$FF0000	**HI08 bootstrap in HC11 nonmultiplexed mode** The bootstrap program sets the host interface to interface with the Freescale HC11 microcontroller through the HI08. The HI08 pin configuration is optimized for connection to the Freescale HC11 nonmultiplexed bus.
E	1	1	1	0	$FF0000	**HI08 bootstrap in 8051 multiplexed bus mode** 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.
F	1	1	1	1	$FF0000	**HI08 bootstrap in MC68302 bus mode** The bootstrap program sets the host interface to interface with the Freescale MC68302 or MC68360 bus through the HI08. The HI08 pin configuration is optimized for connection to a Freescale MC68302 or MC68360 bus.

4.2 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**. Upon exiting the Reset state, the DSP56321 samples the MOD[A–D] signal lines and loads their values into OMR[MA–MD]. The mode input signals (MOD[A–D]) and the resulting MA, MB, MC, and MD bits determine which bootstrap mode the DSP56321 enters (see **Table 4-1**).

Note: To stop the bootstrap in any HI08 bootstrap mode, set the Host Flag 0 (HF0). The loaded user program begins executing 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 DSP56321 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 that specify the number of (24-bit) program words to be loaded
2. Three bytes that specify the (24-bit) start address where the user program loads in the DSP56321 program memory
3. The user program (three bytes for each 24-bit program word)

The three bytes for each data sequence are loaded least significant byte first. When the bootstrap program finishes loading the specified number of words, it jumps to the specified starting address and executes the loaded program.

4.3 Central Processor Unit (CPU) Registers

There are two CPU registers that must be configured to initialize operation. The Status Register (SR) selects various arithmetic processing protocols and contains several status reporting flag bits. The Operating Mode Register (OMR) configures several system operating modes and characteristics.

4.3.1 Status Register (SR)

The Status Register (SR) (**Figure 4-1**) is a 24-bit register that indicates the current system state of the processor and the results of previous arithmetic computations. The SR is pushed onto the system stack when program looping is initialized or a JSR is performed, including long interrupts. The SR consists of the following three special-purpose 8-bit control registers:

- *Extended Mode Register (EMR) (SR[23–16]) and Mode Register (MR) (SR[15–8])*. These special-purpose registers define the current system state of the processor. The bits in both registers are affected by hardware reset, exception processing, ENDDO (end current DO loop) instructions, RTI (return from interrupt) instructions, and TRAP instructions. In addition, the EMR bits are affected by instructions that specify SR as their destination (for example, DO FOREVER instructions, BRKcc instructions, and MOVEC). During hardware reset, all EMR bits are cleared. The MR register bits are affected by DO instructions, and instructions that directly reference the MR (for example, ANDI, ORI, or instructions, such as MOVEC, that specify SR as the destination). During processor reset, the interrupt mask bits are set and all other bits are cleared.
- *Condition Code Register (CCR) (SR[7–0])*. Defines the results of previous arithmetic computations. The CCR bits are affected by Data Arithmetic Logic Unit (Data ALU) operations, parallel move operations, instructions that directly reference the CCR (for example, ORI and ANDI), and instructions that specify SR as a destination (for example, MOVEC). Parallel move operations affect only the S and L bits of the CCR. During processor reset, all CCR bits are cleared.

The definition of the three 8-bit registers within the SR is primarily for the purpose of compatibility with other Freescale DSPs. Bit definitions in the following paragraphs identify the bits within the SR and not within the subregister.

<table>
<tr><th colspan="8">Extended Mode Register (EMR)</th><th colspan="8">Mode Register (MR)</th><th colspan="8">Condition Code Register (CCR)</th></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">CP[1–0]</td><td>RM</td><td>SM</td><td>CE</td><td>▨</td><td>SA</td><td>FV</td><td>LF</td><td>DM</td><td>SC</td><td>▨</td><td colspan="2">S[1–0]</td><td colspan="2">I[1–0]</td><td>S</td><td>L</td><td>E</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td></tr>
<tr><td colspan="24">Reset:</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

▨ Reserved bit. Read as zero; write to zero for future compatibility

Figure 4-1. Status Register (SR)

Table 4-2. Status Register Bit Definitions

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th>Description</th></tr>
<tr><td>23–22</td><td>CP[1–0]</td><td>11</td><td><b>Core Priority</b>
Under control of the CDP[1–0] bits in the OMR, the CP bits specify the priority of core accesses to external memory. These bits are compared against the priority bits of the active DMA channel. If the core priority is greater than the DMA priority, the DMA waits for a free time slot on the external bus. If the core priority is less than the DMA priority, the core waits for a free time slot on the external bus. If the core priority equals the DMA priority, the core and DMA access the external bus in a round robin pattern (for example, ... P, X, Y, DMA, P, X, Y, ...).
<table>
<tr><th>Priority Mode</th><th>Core Priority</th><th>DMA Priority</th><th>OMR (CDP[1-0])</th><th>SR (CP[1–0])</th></tr>
<tr><td rowspan="4">Dynamic</td><td>0 (Lowest)</td><td rowspan="4">Determined by DCRn (DPR[1–0]) for active DMA channel</td><td>00</td><td>00</td></tr>
<tr><td>1</td><td>00</td><td>01</td></tr>
<tr><td>2</td><td>00</td><td>10</td></tr>
<tr><td>3 (Highest)</td><td>00</td><td>11</td></tr>
<tr><td rowspan="3">Static</td><td colspan="2">core < DMA</td><td>01</td><td>xx</td></tr>
<tr><td colspan="2">core = DMA</td><td>10</td><td>xx</td></tr>
<tr><td colspan="2">core > DMA</td><td>11</td><td>xx</td></tr>
</table></td></tr>
<tr><td>21</td><td>RM</td><td>0</td><td><b>Rounding Mode</b>
Selects the type of rounding performed by the Data ALU during arithmetic operations. If RM is cleared, convergent rounding is selected. If RM is set, two’s-complement rounding is selected.</td></tr>
<tr><td>20</td><td>SM</td><td>0</td><td><b>Arithmetic Saturation Mode</b>
Selects automatic saturation on 48 bits for the results going to the accumulator. This saturation is performed by a special circuit inside the MAC unit. The purpose of this bit is to provide an Arithmetic Saturation mode for algorithms that do not recognize or cannot take advantage of the extension accumulator.</td></tr>
</table>

Table 4-2. Status Register Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
19	CE	0	**Cache Enable** Enables/disables the instruction cache controller. If CE is set, the cache is enabled, and instructions are cached into and fetched from the internal Program RAM. If CE is cleared, the cache is disabled and the DSP56300 core fetches instructions from external or internal program memory, according to the memory space table of the specific DSP56300 core-based device. **Note:** To ensure proper operation, do not clear Cache Enable mode while Burst mode is enabled (OMR[BE] is set).
18		0	Reserved. Write to zero for future compatibility.
17	SA	0	**Sixteen-Bit Arithmetic Mode** Affects data width functionality, enabling the Sixteen-bit Arithmetic mode of operation. When SA is set, the core uses 16-bit operations instead of 24-bit operations. In this mode, 16-bit data is right-aligned in the 24-bit memory locations, registers, and 24-bit register portions. Shifting, limiting, rounding, arithmetic instructions, and moves are performed accordingly. For details on Sixteen-Bit Arithmetic mode, consult the *DSP56300 Family Manual*.
16	FV	0	**DO FOREVER Flag** Set when a DO FOREVER loop executes. The FV flag, like the LF flag, is restored from the stack when a DO FOREVER loop terminates. Stacking and restoring the FV flag when initiating and exiting a DO FOREVER loop, respectively, allow program loops to be nested. When returning from the long interrupt with an RTI instruction, the system stack is pulled and the value of the FV bit is restored.
15	LF	0	**Do Loop Flag** When a program loop is in progress, enables the detection of the end of the loop. The LF is restored from stack when a program loop terminates. Stacking and restoring the LF when initiating and exiting a program loop, respectively, allow program loops to be nested. When returning from the long interrupt with an RTI instruction, the System Stack is pulled and the LF bit value is restored.
14	DM	0	**Double-Precision Multiply Mode** Enables four multiply/MAC operations to implement a double-precision algorithm that multiplies two 48-bit operands with a 96-bit result. Clearing the DM bit disables the mode. **Note:** The Double-Precision Multiply mode is supported to maintain object code compatibility with devices in the DSP56000 family. For a more efficient way of executing double precision multiply, refer to the chapter on the Data Arithmetic Logic Unit in the *DSP56300 Family Manual*. In Double-Precision Multiply mode, the behavior of the four specific operations listed in the double-precision algorithm is modified. Therefore, do not use these operations (with those specific register combinations) in Double-Precision Multiply mode for any purpose other than the double precision multiply algorithm. All other Data ALU operations (or the four listed operations, but with other register combinations) can be used. The double-precision multiply algorithm uses the Y0 Register at all stages. Therefore, do not change Y0 when running the double-precision multiply algorithm. If the Data ALU must be used in an interrupt service routine, Y0 should be saved with other Data ALU registers to be used and restored before the interrupt routine terminates.

Table 4-2. Status Register Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="5">Description</th></tr>
<tr><td>13</td><td>SC</td><td>0</td><td colspan="5"><b>Sixteen-Bit Compatibility Mode</b>
Affects addressing functionality, enabling full compatibility with object code written for the DSP56000 family. When SC is set, MOVE operations to/from any of the following PCU registers clear the eight MSBs of the destination: LA, LC, SP, SSL, SSH, EP, SZ, VBA and SC. If the source is either the SR or OMR, then the eight MSBs of the destination are also cleared. If the destination is either the SR or OMR, then the eight MSBs of the destination are left unchanged. To change the value of one of the eight MSBs of the SR or OMR, clear SC.
SC also affects the contents of the Loop Counter Register. If SC is cleared (normal operation), then a loop count value of zero causes the loop body to be skipped, and a loop count value of $FFFFFF causes the loop to execute the maximum number of 2^{24} – 1 times. If the SC bit is set, a loop count value of zero causes the loop to execute 2^{16} times, and a loop count value of $FFFFFF causes the loop to execute 2^{16} – 1 times.
<b>Note:</b> Due to pipelining, a change in the SC bit takes effect only after three instruction cycles. Insert three NOP instructions after the instruction that changes the value of this bit to ensure proper operation.</td></tr>
<tr><td>12</td><td></td><td>0</td><td colspan="5">Reserved. Write to 0 for future compatibility.</td></tr>
<tr><td rowspan="6">11–10</td><td rowspan="6">S[1–0]</td><td rowspan="6">0</td><td colspan="5"><b>Scaling Mode</b>
Specify the scaling to be performed in the Data ALU shifter/limiter and the rounding position in the Data ALU MAC unit. The Shifter/limiter Scaling mode affects data read from the A or B accumulator registers out to the X-data bus (XDB) and Y-data bus (YDB). Different scaling modes can be used with the same program code to allow dynamic scaling. One application of dynamic scaling is to facilitate block floating-point arithmetic. The scaling mode also affects the MAC rounding position to maintain proper rounding when different portions of the accumulator registers are read out to the XDB and YDB. Scaling mode bits are cleared at the start of a long Interrupt Service Routine and during a hardware reset.</td></tr>
<tr><td><b>S1</b></td><td><b>S0</b></td><td><b>Scaling Mode</b></td><td><b>Rounding Bit</b></td><td><b>SEquation</b></td></tr>
<tr><td>0</td><td>0</td><td>No scaling</td><td>23</td><td>S = (A46 XOR A45)
OR (B46 XOR B45)
OR S (previous)</td></tr>
<tr><td>0</td><td>1</td><td>Scale down</td><td>24</td><td>S = (A47 XOR A46)
OR (B47 XOR B46)
OR S (previous)</td></tr>
<tr><td>1</td><td>0</td><td>Scale up</td><td>22</td><td>S = (A45 XOR A44)
OR (B45 XOR B44)
OR S (previous)</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td><td>—</td><td>S undefined</td></tr>
</table>

Table 4-2. Status Register Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th>Description</th></tr>
<tr><td>9–8</td><td>I[1–0]</td><td>11</td><td><b>Interrupt Mask</b>
Reflect the current Interrupt Priority Level (IPL) of the processor and indicate the IPL needed for an interrupt source to interrupt the processor. The current IPL of the processor can be changed under software control. The interrupt mask bits are set during hardware reset, but not during software reset.
<table>
<tr><th>Priority</th><th>I1</th><th>I0</th><th>Exceptions Permitted</th><th>Exceptions Masked</th></tr>
<tr><td>Lowest</td><td>0</td><td>0</td><td>IPL 0, 1, 2, 3</td><td>None</td></tr>
<tr><td></td><td>0</td><td>1</td><td>IPL 1, 2, 3</td><td>IPL 0</td></tr>
<tr><td></td><td>1</td><td>0</td><td>IPL 2, 3</td><td>IPL 0, 1</td></tr>
<tr><td>Highest</td><td>1</td><td>1</td><td>IPL 3</td><td>IPL 0, 1, 2</td></tr>
</table></td></tr>
<tr><td>7</td><td>S</td><td>0</td><td><b>Scaling</b>
Set when a result moves from accumulator A or B to the XDB or YDB buses (during an accumulator to memory or accumulator to register move) and remains set until explicitly cleared; that is, the S bit is a <i>sticky bit</i>. The logical equations of this bit are dependent on the Scaling mode. The scaling bit is set if the absolute value in the accumulator, before scaling, is ≥ 0.25 or &lt; 0.75.</td></tr>
<tr><td>6</td><td>L</td><td>0</td><td><b>Limit</b>
Set if the overflow bit is set or if the data shifter/limiter circuits perform a limiting operation. In Arithmetic Saturation mode, the L bit is also set when an arithmetic saturation occurs in the Data ALU result; otherwise, it is not affected. The L bit is cleared only by a processor reset or by an instruction that specifically clears it (that is, a <i>sticky bit</i>); this allows the L bit to be used as a latching overflow bit. The L bit is affected by data movement operations that read the A or B accumulator registers.</td></tr>
<tr><td>5</td><td>E</td><td>1</td><td><b>Extension</b>
Cleared if all the bits of the integer portion of the 56-bit result are all ones or all zeros; otherwise, this bit is set. The Scaling mode defines the integer portion. If the E bit is cleared, then the low-order fraction portion contains all the significant bits; the high-order integer portion is sign extension. In this case, the accumulator extension register can be ignored. If the E bit is set, it indicates that the accumulator extension register is in use.
<table>
<tr><th>S1</th><th>S0</th><th>Scaling Mode</th><th>Integer Portion</th></tr>
<tr><td>0</td><td>0</td><td>No scaling</td><td>Bits 55–47</td></tr>
<tr><td>0</td><td>1</td><td>Scale down</td><td>Bits 55–48</td></tr>
<tr><td>1</td><td>0</td><td>Scale up</td><td>Bits 5–46</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td><td>Undefined</td></tr>
</table></td></tr>
<tr><td>4</td><td>U</td><td>0</td><td><b>Unnormalized</b>
Set if the two MSBs of the Most Significant Portion (MSP) of the result are identical; otherwise, this bit is cleared. The MSP portion of the A or B accumulators is defined by the Scaling mode.
<table>
<tr><th>S1</th><th>S0</th><th>Scaling Mode</th><th>Integer Portion</th></tr>
<tr><td>0</td><td>0</td><td>No scaling</td><td>$U = \overline{(\text{Bit 47 XOR Bit 46})}$</td></tr>
<tr><td>0</td><td>1</td><td>Scale down</td><td>$U = \overline{(\text{Bit 48 XOR Bit 47})}$</td></tr>
<tr><td>1</td><td>0</td><td>Scale up</td><td>$U = \overline{(\text{Bit 46 XOR Bit 45})}$</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td><td>U undefined</td></tr>
</table></td></tr>
</table>

Table 4-2. Status Register Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
3	N	0	**Negative** Set if the MSB of the result is set; otherwise, this bit is cleared.
2	Z	0	**Zero** Set if the result equals zero; otherwise, this bit is cleared.
1	V	0	**Overflow** Set if an arithmetic overflow occurs in the 56-bit result; otherwise, this bit is cleared. V indicates that the result cannot be represented in the accumulator register (that is, the register overflowed). In Arithmetic Saturation mode, an arithmetic overflow occurs if the Data ALU result is not representable in the accumulator without the extension part (that is, 48-bit accumulator or the 32-bit accumulator in Arithmetic Sixteen-bit mode).
0	C	0	**Carry** Set if a carry is generated by the MSB resulting from an addition operation. This bit is also set if a borrow is generated in a subtraction operation; otherwise, this bit is cleared. The carry or borrow is generated from Bit 55 of the result. The C bit is also affected by bit manipulation, rotate, and shift instructions.

4.3.2 Operating Mode Register (OMR)

The OMR is a read/write register divided into three byte-sized units. The lowest two bytes (EOM and COM) control the chip's operating mode. The high byte (SCS) controls and monitors the stack extension. The OMR control bits are shown in **Figure 4-2**.

	Stack Control/Status (SCS)								Extended Operating Mode (EOM)								Chip Operating Mode (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[2–1]		SEN	WRP	EOV	EUN	XYS	▨	APD	ABE	BRT	TAS	BE	CDP[1–0]		MSW0	SD	▨	EBD	MD	MC	MB	MA
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	*	*	*	*

* After reset, these bits reflect the corresponding value of the mode input (that is, MODD, MODC, MODB, or MODA, respectively).

▨ Reserved bit. Read as zero; write to zero for future compatibility

The Enhanced Operating Mode (EOM) and Chip Operating Mode (COM) bytes are affected only by processor reset and by instructions directly referencing the OMR (that is, ANDI, ORI, and other instructions, such as MOVEC, that specify OMR as a destination). The Stack Control/Status (SCS) byte is referenced implicitly by some instructions, such as DO, JSR, and RTI, or directly by the MOVEC instruction. During processor reset, the chip operating mode bits (MD, MC, MB, and MA) are loaded from the external mode select pins MODD, MODC, MODB, and MODA respectively. **Table 4-3** defines the DSP56321 OMR bits.

Table 4-3. Operating Mode Register (OMR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23		0	Reserved. Write to 0 for future compatibility.
22–21	MSW[2–1]	0	**Memory Switch Mode Bits 2, 1** Used with bit 7 (MSW0), the three bits configure the internal memory sizes for Program, X-data, and Y-data memory. See **Table 3-1** for details. **Notes:** **1.** To ensure proper operation, place six NOP instructions after the instruction that changes the MSW bits. **2.** To ensure proper operation, do not change the MSW bits while the Instruction Cache is enabled (SR[CE] bit is set).
19	WRP	0	**Stack Extension Wrap Flag** Set when copying from the on-chip hardware stack (System Stack Register file) to the stack extension memory begins. You can use this flag during the debugging phase of the software development to evaluate and increase the speed of software-implemented algorithms. The WRP flag is a *sticky bit* (that is, cleared only by hardware reset or by an explicit MOVEC operation to the OMR).
18	EOV	0	**Stack Extension Overflow Flag** Set when a stack overflow occurs in Stack Extended mode. Extended stack overflow is recognized when a push operation is requested while SP = SZ (Stack Size register), and the Extended mode is enabled by the SEN bit. The EOV flag is a *sticky bit* (that is, cleared only by hardware reset or by an explicit MOVEC operation to the OMR). The transition of the EOV flag from zero to one causes a Priority Level 3 (Non-maskable) stack error exception.
17	EUN	0	**Stack Extension Underflow Flag** Set when a stack underflow occurs in Extended Stack mode. Extended stack underflow is recognized when a pull operation is requested, SP = 0, and the SEN bit enables Extended mode. The EUN flag is a *sticky bit* (that is, cleared only by hardware reset or by an explicit MOVEC operation to the OMR). Transition of the EUN flag from zero to one causes a Priority Level 3 (Non-maskable) stack error exception. **Note:** While the chip is in Extended Stack mode, the UF bit in the SP acts like a normal counter bit.
16	XYS	0	**Stack Extension XY Select** Determines whether the stack extension is mapped onto X or Y memory space. If the bit is clear, then the stack extension is mapped onto the X memory space. If the XYS bit is set, the stack extension is mapped to the Y memory space.
15		0	Reserved. Write to 0 for future compatibility.

Table 4-3. Operating Mode Register (OMR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="2">Description</th></tr>
<tr><td>14</td><td>APD</td><td>0</td><td colspan="2">**Address Attribute Priority Disable**
Disables the priority assigned to the Address Attribute signals (AA[0–3]). When APD = 0 (default setting), the four Address Attribute signals each have a certain priority: AA3 has the highest priority, AA0 has the lowest priority. Therefore, only one AA signal can be active at one time. This allows continuous partitioning of external memory; however, certain functions, such as using the AA signals as additional address lines, require the use of additional interface hardware. When APD is set, the priority mechanism is disabled, allowing more than one AA signal to be active simultaneously. Therefore, the AA signals can be used as additional address lines without the need for additional interface hardware. For details on the Address Attribute Registers, see **Section 4.5.2**, *Address Attribute Registers (AAR[0–3])*, on page 4-20.</td></tr>
<tr><td>13</td><td>ABE</td><td>0</td><td colspan="2">**Asynchronous Bus Arbitration Enable**
Eliminates the setup and hold time requirements for $\overline{BB}$ and $\overline{BG}$, and substitutes a required non-overlap interval between the deassertion of one $\overline{BG}$ input to a DSP56300 family device and the assertion of a second $\overline{BG}$ input to a second DSP56300 family device on the same bus. When the ABE bit is set, the $\overline{BG}$ and $\overline{BB}$ inputs are synchronized. This synchronization causes a delay between a change in $\overline{BG}$ or $\overline{BB}$ until this change is actually accepted by the receiving device.</td></tr>
<tr><td>12</td><td>BRT</td><td>0</td><td colspan="2">**Bus Release Timing**
Selects between fast or slow bus release. If BRT is cleared, a Fast Bus Release mode is selected (that is, no additional cycles are added to the access and $\overline{BB}$ is not guaranteed to be the last Port A pin that is tri-stated at the end of the access). If BRT is set, a Slow Bus Release mode is selected (that is, an additional cycle is added to the access, and $\overline{BB}$ is the last Port A pin that is tri-stated at the end of the access).</td></tr>
<tr><td>11</td><td>TAS</td><td>0</td><td colspan="2">**$\overline{TA}$ Synchronize Select**
Selects the synchronization method for the input Port A pin—$\overline{TA}$ (Transfer Acknowledge). For correct operation if $\overline{TA}$ is used, TAS must be set to synchronize the $\overline{TA}$ signal with the internal clock.</td></tr>
<tr><td>10</td><td>BE</td><td>0</td><td colspan="2">**Cache Burst Mode Enable**
Enables/disables Burst mode in the memory expansion port during an instruction cache miss. If the bit is cleared, Burst mode is disabled and only one program word is fetched from the external memory when an instruction cache miss condition is detected. If the bit is set, Burst mode is enabled, and up to four program words are fetched from the external memory when an instruction cache miss is detected.</td></tr>
<tr><td rowspan="5">9–8</td><td rowspan="5">CDP[1–0]</td><td rowspan="5">11</td><td colspan="2">**Core-DMA Priority**
Specify the priority of core and DMA accesses to the external bus.</td></tr>
<tr><td>00</td><td>Determined by comparing status register CP[1–0] to the active DMA channel priority</td></tr>
<tr><td>01</td><td>DMA accesses have higher priority than core accesses</td></tr>
<tr><td>10</td><td>DMA accesses have the same priority as the core accesses</td></tr>
<tr><td>11</td><td>DMA accesses have lower priority than the core accesses</td></tr>
<tr><td>7</td><td>MS</td><td>0</td><td colspan="2">**Memory Switch Mode Bit 0**
Used with bits 22–21 (MSW[2–1]). See bits 22–21 for details.</td></tr>
</table>

Table 4-3. Operating Mode Register (OMR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
6	SD	0	**Stop Delay Mode** Determines the length of the delay invoked when the core exits the Stop state. The STOP instruction suspends core processing indefinitely until a defined event occurs to restart it. If SD is cleared, a 128K clock cycle delay is invoked before a STOP instruction cycle continues. However, if SD is set, the delay before the instruction cycle continues is 16 clock cycles. The long delay allows a clock stabilization period for the internal clock to begin oscillating and to stabilize. When a stable external clock is used, the shorter delay allows faster start-up of the DSP56300 core.
5		0	Reserved. Write to zero for future compatibility.
4	EBD	0	**External Bus Disable** Disables the external bus controller to reduce power consumption when external memories are not used. When EBD is set, the external bus controller is disabled and external memory cannot be accessed. When EBD is cleared, the external bus controller is enabled and external access can be performed. Hardware reset clears the EBD bit.
3–0	MD–MA	*	**Chip Operating Mode** Indicate the operating mode of the DSP56300 core. On hardware reset, these bits are loaded from the external mode select pins, MODD, MODC, MODB, and MODA, respectively. After the DSP56300 core leaves the Reset state, MD–MA can be changed under program control.

* The MD–MA bits reflect the corresponding value of the mode input (that is, MODD–MODA), respectively.

4.4 Configuring Interrupts

DSP56321 interrupt handling, like that for all DSP56300 family members, is optimized for DSP applications. Refer to the sections describing interrupts in **Chapter 2**, *Core Architecture Overview*, in the *DSP56300 Family Manual*. Two registers are used to configure the interrupt characteristics:

- Interrupt Priority Register-Core (IPRC)—Programmed to configure the priority levels for the core DMA interrupts and the external interrupt lines as well as the interrupt line trigger modes
- Interrupt Priority Register-Peripherals (IPRP)—Programmed to configure the priority levels for the interrupts used with the on-chip peripheral devices

The interrupt table resides in the 256 locations of program memory to which the PCU vector base address (VBA) register points. These locations store the starting instructions of the interrupt handler for each specified interrupt. The memory is programmed by the bootstrap program at startup.

4.4.1 Interrupt Priority Registers (IPRC and IPRP)

There are two interrupt priority registers in the DSP56321. The IPRC (**Figure 4-3**) is dedicated to DSP56300 core interrupt sources, and IPRP (**Figure 4-4**) is dedicated to DSP56321 peripheral interrupt sources.

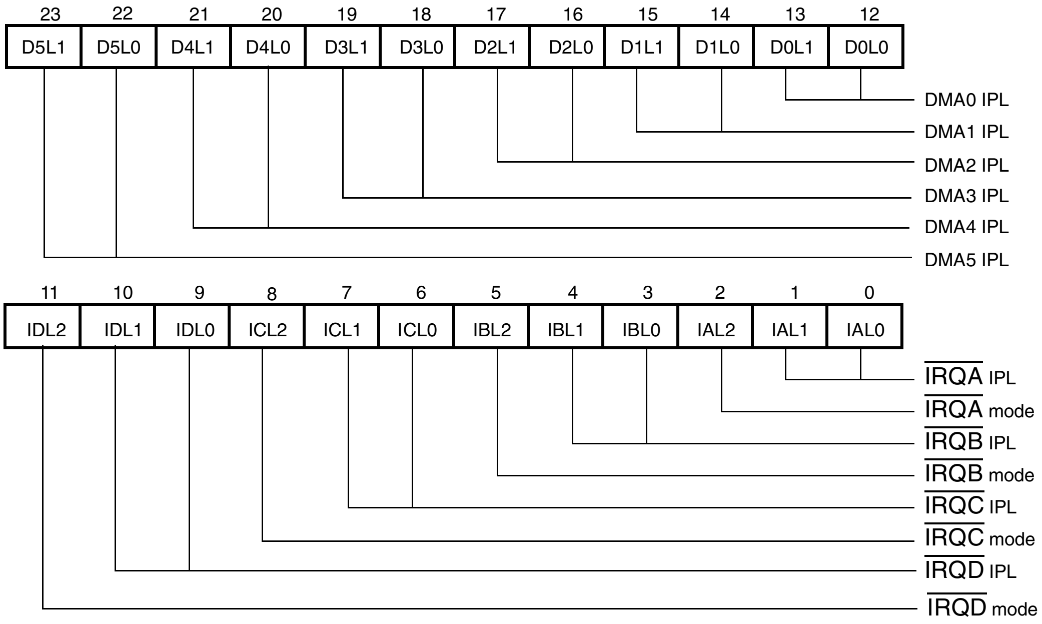


Figure 4-2. Interrupt Priority Register-Core (IPRC) (X:$FFFFFF)

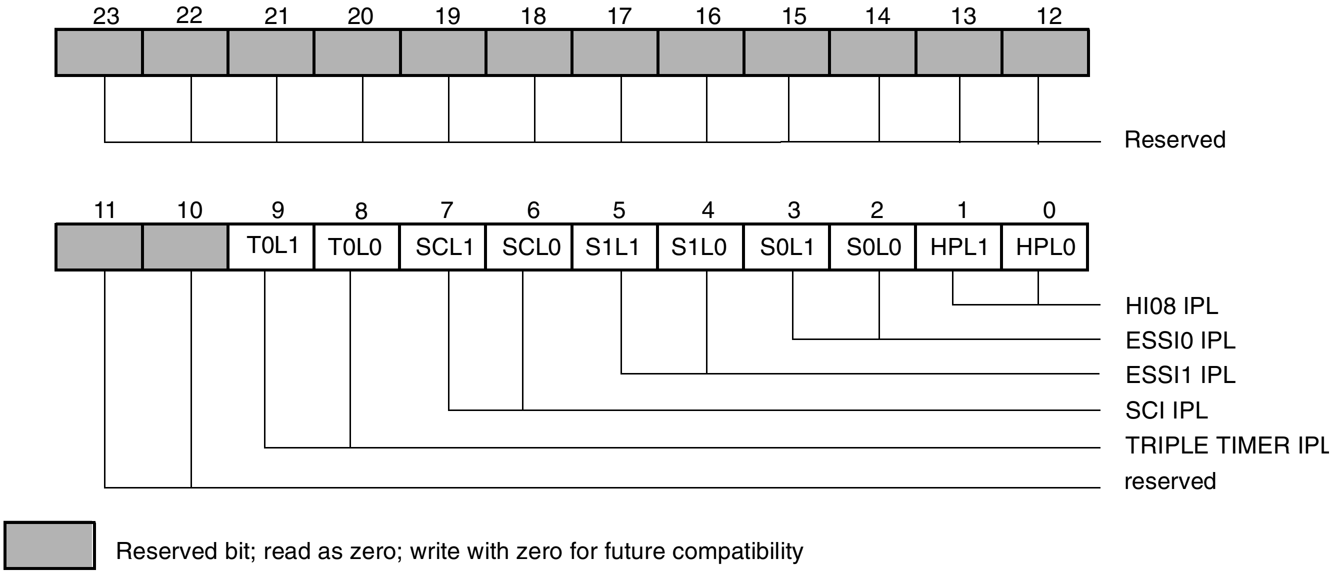


Figure 4-3. Interrupt Priority Register-Peripherals (IPRP) (X:$FFFFFE)

The DSP56321 has a four-level interrupt priority structure. Each interrupt has two interrupt priority level bits (IPL[1–0]) that determine its interrupt priority level. Level 0 is the lowest priority; Level 3 is the highest-level priority and is non-maskable. **Table 4-4** defines the IPL bits.

Table 4-4. 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

The IPRC also selects the trigger mode of the external interrupts ($\overline{\text{IRQA}}$–$\overline{\text{IRQD}}$). If the value of the IxL2 bit is 0, the interrupt mode is level-triggered. If the value is 1, the interrupt mode is negative-edge-triggered.

4.4.2 Interrupt Table Memory Map

Each interrupt is allocated two instructions in the interrupt table, resulting in 128 table entries for interrupt handling. **Table 4-5** shows the table entry address for each interrupt source. The DSP56321 initialization program loads the table entry for each interrupt serviced with two interrupt servicing instructions. In the DSP56321, 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 $\overline{\text{NMI}}$ (IPL = 3) or for host command interrupt (IPL = 2). Unused interrupt vector locations can be used for program or data storage.

Table 4-5. 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	$\overline{\text{IRQB}}$

Table 4-5. Interrupt Sources (Continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
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
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

Table 4-5. Interrupt Sources (Continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
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:$6C	0–2	Reserved
:	:	:
VBA:$FE	0–2	Reserved

4.4.3 Processing 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. **Table 4-6** shows this fixed-priority list of interrupt sources within an IPL, from highest to lowest at each level The interrupt mask bits in the Status Register (I[1–0]) can be programmed to ignore low priority-level interrupt requests.

Table 4-6. 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)

Table 4-6. Interrupt Source Priorities Within an IPL (Continued)

Priority	Interrupt Source
	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
	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

4.5 Bus Interface Unit (BIU) Registers

The three Bus Interface Unit (BIU) registers configure the external memory expansion port (Port A). They include the following:

- Bus Control Register (BCR)
- Address Attribute Registers (AAR[3–0])

To use Port A correctly, configure these registers as part of the bootstrap process. The following subsections describe these registers.

4.5.1 Bus Control Register

The Bus Control Register (BCR), depicted in **Figure 4-5**, is a read/write register that controls the external bus activity and Bus Interface Unit (BIU) operation. All BCR bits except bit 21, BBS, are read/write bits.

23	22	21	20	19	18	17	16	15	14	13	12
BRH	▨	BBS	BDFW4	BDFW3	BDFW2	BDFW1	BDFW0	BA3W2	BA3W1	BA3W0	BA2W2

11	10	9	8	7	6	5	4	3	2	1	0
BA2W1	BA2W0	BA1W4	BA1W3	BA1W2	BA1W1	BA1W0	BA0W4	BA0W3	BA0W2	BA0W1	BA0W0

▨ Reserved bit. Read as zero; write to zero for future compatibility

Table 4-7. Bus Control Register (BCR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23	BRH	0	**Bus Request Hold** Asserts the $\overline{BR}$ signal, even if no external access is needed. When BRH is set, the $\overline{BR}$ signal is always asserted. If BRH is cleared, the $\overline{BR}$ is asserted only if an external access is attempted or pending.
22	▨	0	Reserved. Write to 0 for future compatibility.
21	BBS	0	**Bus State** This read-only bit is set when the DSP is the bus master and is cleared otherwise.
20–16	BDFW[4–0]	11111 (31 wait states)	**Bus Default Area Wait State Control** Defines the number of wait states (one through 31) inserted into each external access to an area that is not defined by any of the AAR registers. The access type for this area is SRAM only. These bits should not be programmed as zero since SRAM memory access requires at least one wait state. When three through seven wait states are selected, one additional wait state is inserted at the end of the access. When selecting eight or more wait states, two additional wait states are inserted at the end of the access. These trailing wait states increase the data hold time and the memory release time and do not increase the memory access time.

Table 4-7. Bus Control Register (BCR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
15–13	BA3W[2–0]	111 (7 wait states)	**Bus Area 3 Wait State Control** Defines the number of wait states (1–7) inserted in each external SRAM access to Area 3. Area 3 is the area defined by AAR3. **Note:** Do not program the value of these bits as zero since SRAM memory access requires at least one wait state. When three through seven wait states are selected, one additional wait state is inserted at the end of the access. This trailing wait state increases the data hold time and the memory release time and does not increase the memory access time.
12–10	BA2W[2–0]	111 (7 wait states)	**Bus Area 2 Wait State Control** Defines the number of wait states (1–7) inserted into each external SRAM access to Area 2. Area 2 is the area defined by AAR2. **Note:** Do not program the value of these bits as zero, since SRAM memory access requires at least one wait state. When three through seven wait states are selected, one additional wait state is inserted at the end of the access. This trailing wait state increases the data hold time and the memory release time and does not increase the memory access time.
9–5	BA1W[4–0]	11111 (31 wait states)	**Bus Area 1 Wait State Control** Defines the number of wait states (1–31) inserted into each external SRAM access to Area 1. Area 1 is the area defined by AAR1. **Note:** Do not program the value of these bits as zero, since SRAM memory access requires at least one wait state. When three through seven wait states are selected, one additional wait state is inserted at the end of the access. When selecting eight or more wait states, two additional wait states are inserted at the end of the access. These trailing wait states increase the data hold time and the memory release time and do not increase the memory access time.
4–0	BA0W[4–0]	11111 (31 wait states)	**Bus Area 0 Wait State Control** Defines the number of wait states (1–31) inserted in each external SRAM access to Area 0. Area 0 is the area defined by AAR0. **Note:** Do not program the value of these bits as zero, since SRAM memory access requires at least one wait state. When selecting three through seven wait states, one additional wait state is inserted at the end of the access. When selecting eight or more wait states, two additional wait states are inserted at the end of the access. These trailing wait states increase the data hold time and the memory release time and do not increase the memory access time.

4.5.2 Address Attribute Registers (AAR[0–3])

The Address Attribute Registers (AAR[0–3]) are read/write registers that control the activity of the AA[0–3] pins. The associated AAn pin is asserted if the address defined by the BAC bits in the associated AAR matches the exact number of external address bits defined by the BNC bits, and the external address space (X data, Y data, or program) is enabled by the AAR. **Figure 4-6** shows an AAR register; **Table 4-8** lists the bit definitions.

Note: The DSP56321 does not support address multiplexing.

23	22	21	20	19	18	17	16	15	14	13	12
BAC11	BAC10	BAC9	BAC8	BAC7	BAC6	BAC5	BAC4	BAC3	BAC2	BAC1	BAC0
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BNC3	BNC2	BNC1	BNC0	BPAC	▨	BYEN	BXEN	BPEN	BAAP	BAT1	BAT0

▨ Reserved bit. Read as zero; write to zero for future compatibility

Figure 4-4. Address Attribute Registers (AAR[0–3]) (X:$FFFFF9–$FFFFF6)

Table 4-8. Address Attribute Registers (AAR[0–3]) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23–12	BAC[11–0]	0	**Bus Address to Compare** Read/write control bits that define the upper 12 bits of the 24-bit address with which to compare the external address to determine whether to assert the corresponding AA signal. This is also true of 16-bit compatibility mode. The BNC[3–0] bits define the number of address bits to compare.
11–8	BNC[3–0]	0	**Bus Number of Address Bits to Compare** Specify the number of bits (from the BAC bits) that are compared to the external address. The BAC bits are always compared with the Most Significant Portion of the external address (for example, if BNC[3–0] = 0011, then the BAC[11–9] bits are compared to the 3 MSBs of the external address). If no bits are specified (that is, BNC[3–0] = 0000), the AA signal is activated for the entire 16 M-word space identified by the space enable bits (BPEN, BXEN, BYEN), but only when the address is external to the internal memory map. The combinations BNC[3–0] = 1111, 1110, 1101 are reserved.
7	BPAC	0	**Bus Packing Enable** Enables/disables the internal packing/unpacking logic. When BPAC is set, packing is enabled. In this mode each DMA external access initiates three external accesses to an 8-bit wide external memory (the addresses for these accesses are DAB, then DAB + 1 and then DAB + 2). Packing to a 24-bit word (or unpacking from a 24-bit word to three 8-bit words) is done automatically by the expansion port control hardware. The external memory should reside in the eight Least Significant Bits (LSBs) of the external data bus, and the packing (or unpacking for external write accesses) occurs in “Little Endian” order (that is, the low byte is stored in the lowest of the three memory locations and is transferred first; the middle byte is stored/transferred next; and the high byte is stored/transferred last). When this bit is cleared, the expansion port control logic assumes a 24-bit wide external memory. **Notes:** **1.** BPAC is used only for DMA accesses and not core accesses. **2.** To ensure sequential external accesses, the DMA address should advance three steps at a time in two-dimensional mode with a row length of one and an offset size of three. For details, refer to Freescale application note, **APR23/D**, *Using the DSP56300 Direct Memory Access Controller*. **3.** To prevent improper operation, DMA address + 1 and DMA address + 2 should not cross the AAR bank borders. **4.** Arbitration is not allowed during the packing access (that is, the three accesses are treated as one access with respect to arbitration, and the bus mastership is not released during these accesses).

Table 4-8. Address Attribute Registers (AAR[0–3]) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
6		0	Reserved. Write to 0 for future compatibility.
5	BYEN	0	**Bus Y Data Memory Enable** A read/write control bit that enables/disables the AA pin and logic during external Y data space accesses. When set, BYEN enables the comparison of the external address to the BAC bits during external Y data space accesses. If BYEN is cleared, no address comparison is performed.
4	BXEN	0	**Bus X Data Memory Enable** A read/write control bit that enables/disables the AA pin and logic during external X data space accesses. When set, BXEN enables the comparison of the external address to the BAC bits during external X data space accesses. If BXEN is cleared, no address comparison is performed.
3	BPEN	0	**Bus Program Memory Enable** A read/write control bit that enables/disables the AA pin and logic during external program space accesses. When set, BPEN enables the comparison of the external address to the BAC bits during external program space accesses. If BPEN is cleared, no address comparison is performed.
2	BAAP	0	**Bus Address Attribute Polarity** A read/write Bus Address Attribute Polarity (BAAP) control bit that defines whether the AA signal is active low or active high. When BAAP is cleared, the AA signal is active low (useful for enabling memory modules). If BAAP is set, the appropriate AA signal is active high (useful as an additional address bit).
1–0	BAT[1–0]	0	**Bus Access Type** Read/write bits that define the type of external memory (SRAM) to access for the area defined by the BAC[11–0],BYEN, BXEN, and BPEN bits. The encoding of BAT[1–0] is: • 00 = Reserved • 01 = SRAM access • 10 = Reserved • 11 = Reserved When the external access type is defined as a SRAM access (BAT[1–0] = 01), AA acts as an Address Attribute signal. External accesses to the default area always execute as if BAT[1–0] = 01 (that is, SRAM access). If Port A is used for external accesses, the BAT bits in the AAR3–0 registers must be initialized to the SRAM access type (that is, BAT = 01). To ensure proper operation of Port A, this initialization must occur even for an AAR register that is not used during any Port A access. At reset, the BAT bits are initialized to 00.

4.6 DMA Control Registers 5–0 (DCR[5–0])

The DMA Control Registers (DCR[5–0]) are read/write registers that control the DMA operation for each of their respective channels. All DCR bits are cleared during processor reset.

23	22	21	20	19	18	17	16	15	14	13	12
DE	DIE	DTM2	DTM1	DTM0	DPR1	DPR0	DCON	DRS4	DRS3	DRS2	DRS1

11	10	9	8	7	6	5	4	3	2	1	0
DRS0	D3D	DAM5	DAM4	DAM3	DAM2	DAM1	DAM0	DDS1	DDS0	DSS1	DSS0

Table 4-9. DMA Control Register (DCR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23	DE	0	**DMA Channel Enable** Enables the channel operation. Setting DE either triggers a single block DMA transfer or enables a single-block, single-line, or single-word DMA transfer in the transfer modes that use a requesting device as a trigger. DE is cleared by the end of DMA transfer in some of the transfer modes defined by the DTM bits. If software clears DE during a DMA operation, the channel operation stops only after the current DMA transfer completes (that is, the current word is stored into the destination).
22	DIE	0	**DMA Interrupt Enable** Generates a DMA interrupt at the end of a DMA block transfer after the counter is loaded with its preloaded value. A DMA interrupt is also generated when software explicitly clears $\overline{\text{DE}}$ during a DMA operation. Once asserted, a DMA interrupt request can be cleared only by the service of a DMA interrupt routine. To ensure that a new interrupt request is not generated, clear DIE while the DMA interrupt is serviced and before a new DMA request is generated at the end of a DMA block transfer—that is, at the beginning of the DMA channel interrupt service routine. When DIE is cleared, the DMA interrupt is disabled.

Table 4-9. DMA Control Register (DCR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description			
21–19	DTM[2–0]	0	**DMA Transfer Mode** Specify the operating modes of the DMA channel, as follows:			
			DTM[2–0]	**Trigger**	**DE Cleared After**	**Transfer Mode**
			000	request	Yes	Block Transfer—DE enabled and DMA request initiated. The transfer is complete when the counter decrements to zero and the DMA controller reloads the counter with the original value.
			001	request	Yes	Word Transfer—A word-by-word block transfer (length set by the counter) that is DE enabled. The transfer is complete when the counter decrements to zero and the DMA controller reloads the counter with the original value.
			010	request	Yes	Line Transfer—A line by line block transfer (length set by the counter) that is DE enabled. The transfer is complete when the counter decrements to zero and the DMA controller reloads the counter with the original value.
			011	DE	Yes	Block Transfer—The DE-initiated transfer is complete when the counter decrements to zero and the DMA controller reloads the counter with the original value.
			100	request	No	Block Transfer—The transfer is enabled by DE and initiated by the first DMA request. The transfer is completed when the counter decrements to zero and reloads itself with the original value. The DE bit is not cleared at the end of the block, so the DMA channel waits for a new request. The DMA End-of-Block Transfer Interrupt cannot be used in this mode.
			101	request	No	Word Transfer—The transfer is enabled by DE and initiated by every DMA request. When the counter decrements to zero, it is reloaded with its original value. The DE bit is not automatically cleared, so the DMA channel waits for a new request. The DMA End-of-Block-Transfer Interrupt cannot be used in this mode.
			110	Reserved		
			111	Reserved		
Note:	When DTM[2–0] = 001 or 101, some peripherals can generate a second DMA request while the DMA controller is still processing the first request (see the description of the DRS bits).					

Table 4-9. DMA Control Register (DCR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="2">Description</th></tr>
<tr><td rowspan="7">18–17</td><td rowspan="7">DPR[1–0]</td><td rowspan="7">0</td><td colspan="2"><b>DMA Channel Priority</b>
Define the DMA channel priority relative to the other DMA channels and to the core priority if an external bus access is required. For pending DMA transfers, the DMA controller compares channel priority levels to determine which channel can activate the next word transfer. This decision is required because all channels use common resources, such as the DMA address generation logic, buses, and so forth.</td></tr>
<tr><th>DPR[1–0]</th><th>Channel Priority</th></tr>
<tr><td>00</td><td>Priority level 0 (lowest)</td></tr>
<tr><td>01</td><td>Priority level 1</td></tr>
<tr><td>10</td><td>Priority level 2</td></tr>
<tr><td>11</td><td>Priority level 3 (highest)</td></tr>
<tr><td colspan="2">• If all or some channels have the same priority, then channels are activated in a round-robin fashion—that is, channel 0 is activated to transfer one word, followed by channel 1, then channel 2, and so on.
• If channels have different priorities, the highest priority channel executes DMA transfers and continues for its pending DMA transfers.
• If a lower-priority channel is executing DMA transfers when a higher priority channel receives a transfer request, the lower-priority channel finishes the current word transfer and arbitration starts again.
• If some channels with the same priority are active in a round-robin fashion and a new higher-priority channel receives a transfer request, the higher-priority channel is granted transfer access after the current word transfer is complete. After the higher-priority channel transfers are complete, the round-robin transfers continue. The order of transfers in the round-robin mode may change, but the algorithm remains the same.
• The DPR bits also determine the DMA priority relative to the core priority for external bus access. Arbitration uses the current active DMA priority, the core priority defined by the SR bits CP[1–0], and the core-DMA priority defined by the OMR bits CDP[1–0]. Priority of core accesses to external memory is as follows:</td></tr>
</table>

Table 4-9. DMA Control Register (DCR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="3">Description</th></tr>
<tr><td rowspan="9">18–17 cont.</td><td rowspan="9">DPR[1–0]</td><td rowspan="9"></td><td><b>OMR - CDP[1–0]</b></td><td><b>CP[1–0]</b></td><td><b>Core Priority</b></td></tr>
<tr><td>00</td><td>00</td><td>0 (lowest)</td></tr>
<tr><td>00</td><td>01</td><td>1</td></tr>
<tr><td>00</td><td>10</td><td>2</td></tr>
<tr><td>00</td><td>11</td><td>3 (highest)</td></tr>
<tr><td>01</td><td>xx</td><td>DMA accesses have higher priority than core accesses</td></tr>
<tr><td>10</td><td>xx</td><td>DMA accesses have the same priority as core accesses</td></tr>
<tr><td>11</td><td>xx</td><td>DMA accesses have lower priority than core accesses</td></tr>
<tr><td colspan="3">• If DMA priority > core priority (for example, if CDP = 01, or CDP = 00 and DPR > CP), the DMA performs the external bus access first and the core waits for the DMA channel to complete the current transfer.
• If DMA priority = core priority (for example, if CDP = 10, or CDP = 00 and DPR = CP), the core performs all its external accesses first and then the DMA channel performs its access.
• If DMA priority < core priority (for example, if CDP=11, or CDP = 00 and DPR < CP), the core performs its external accesses and the DMA waits for a free slot in which the core does not require the external bus.
• In Dynamic Priority mode (CDP = 00), the DMA channel can be halted before executing both the source and destination accesses if the core has higher priority. If another higher-priority DMA channel requests access, the halted channel finishes its previous access with a new higher priority before the new requesting DMA channel is serviced.</td></tr>
<tr><td>16</td><td>DCON</td><td>0</td><td colspan="3"><b>DMA Continuous Mode Enable</b>
Enables/disables DMA Continuous mode. When DCON is set, the channel enters the Continuous Transfer mode and cannot be interrupted during a transfer by any other DMA channel of equal priority. DMA transfers in the continuous mode of operation can be interrupted if a DMA channel of higher priority is enabled after the continuous mode transfer starts. If the priority of the DMA transfer in continuous mode (that is, DCON = 1) is higher than the core priority (CDP = 01, or CDP = 00 and DPR > CP), and if the DMA requires an external access, the DMA gets the external bus and the core is not able to use the external bus in the next cycle after the DMA access even if the DMA does not need the bus in this cycle. When DCON is cleared, the priority algorithm operates as for the DPR bits.</td></tr>
</table>

Table 4-9. DMA Control Register (DCR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="2">Description</th></tr>
<tr><td rowspan="26">15–11</td><td rowspan="26">DRS[4–0]</td><td rowspan="26">0</td><td colspan="2"><b>DMA Request Source</b>
Encodes the source of DMA requests that trigger the DMA transfers. The DMA request sources may be external devices requesting service through the $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$ and $\overline{\text{IRQD}}$ pins, triggering by transfers done from a DMA channel, or transfers from the internal peripherals. All the request sources behave as edge-triggered synchronous inputs.</td></tr>
<tr><td><b>DRS[4–0]</b></td><td><b>Requesting Device</b></td></tr>
<tr><td>00000</td><td>External ($\overline{\text{IRQA}}$ pin)</td></tr>
<tr><td>00001</td><td>External ($\overline{\text{IRQB}}$ pin)</td></tr>
<tr><td>00010</td><td>External ($\overline{\text{IRQC}}$ pin)</td></tr>
<tr><td>00011</td><td>External ($\overline{\text{IRQD}}$ pin)</td></tr>
<tr><td>00100</td><td>Transfer done from channel 0</td></tr>
<tr><td>00101</td><td>Transfer done from channel 1</td></tr>
<tr><td>00110</td><td>Transfer done from channel 2</td></tr>
<tr><td>00111</td><td>Transfer done from channel 3</td></tr>
<tr><td>01000</td><td>Transfer done from channel 4</td></tr>
<tr><td>01001</td><td>Transfer done from channel 5</td></tr>
<tr><td>01010</td><td>ESSI0 receive data (RDF0 = 1)</td></tr>
<tr><td>01011</td><td>ESSI0 transmit data (TDE0 = 1)</td></tr>
<tr><td>01100</td><td>ESSI1 receive data (RDF1 = 1)</td></tr>
<tr><td>01101</td><td>ESSI1 transmit data (TDE1 = 1)</td></tr>
<tr><td>01110</td><td>SCI receive data (RDRF = 1)</td></tr>
<tr><td>01111</td><td>SCI transmit data (TDRE = 1)</td></tr>
<tr><td>10000</td><td>Timer0 (TCF0 = 1)</td></tr>
<tr><td>10001</td><td>Timer1 (TCF1 = 1)</td></tr>
<tr><td>10010</td><td>Timer2 (TCF2 = 1)</td></tr>
<tr><td>10011</td><td>Host receive data full (HRDF = 1)</td></tr>
<tr><td>10100</td><td>Host transmit data empty (HTDE = 1)</td></tr>
<tr><td>10101</td><td>EFCOP input buffer empty (FDIBE=1)</td></tr>
<tr><td>10110</td><td>EFCOP output buffer full (FDOBF=1)</td></tr>
<tr><td>10111–11111</td><td>Reserved</td></tr>
<tr><td></td><td></td><td></td><td colspan="2">Peripheral requests 18–21 (DRS[4–0] = 111xx) can serve as fast request sources. Unlike a regular peripheral request in which the peripheral can not generate a second request until the first one is served, a fast peripheral has a full duplex handshake to the DMA, enabling a maximum throughput of a trigger every two clock cycles. This mode is functional only in the Word Transfer mode (that is, DTM = 001 or 101). In the Fast Request mode, the DMA sets an enable line to the peripheral. If required, the peripheral can send the DMA a one cycle triggering pulse. This pulse resets the enable line. If the DMA decides by the priority algorithm that this trigger will be served in the next cycle, the enable line is set again, even before the corresponding register in the peripheral is accessed.</td></tr>
</table>

Table 4-9. DMA Control Register (DCR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="3">Description</th></tr>
<tr><td>10</td><td>D3D</td><td>0</td><td colspan="3"><b>Three-Dimensional Mode</b>
Indicates whether a DMA channel is currently using three-dimensional (D3D = 1) or non-three-dimensional (D3D = 0) addressing modes. The addressing modes are specified by the DAM bits.</td></tr>
<tr><td>9–4</td><td>DAM[5–0]</td><td>0</td><td colspan="3"><b>DMA Address Mode</b>
Defines the address generation mode for the DMA transfer. These bits are encoded in two different ways according to the D3D bit.</td></tr>
<tr><td rowspan="6">3–2</td><td rowspan="6">DDS[1–0]</td><td rowspan="6">0</td><td colspan="3"><b>DMA Destination Space</b>
Specify the memory space referenced as a destination by the DMA.
<b>Note:</b> In Cache mode, a DMA to Program memory space has some limitations (as described in the instruction cache chapter of the <i>DSP56300 Family Manual</i> and in the chapter on operating modes).</td></tr>
<tr><th>DDS1</th><th>DDS0</th><th>DMA Destination Memory Space</th></tr>
<tr><td>0</td><td>0</td><td>X Memory Space</td></tr>
<tr><td>0</td><td>1</td><td>Y Memory Space</td></tr>
<tr><td>1</td><td>0</td><td>P Memory Space</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
<tr><td rowspan="6">1–0</td><td rowspan="6">DSS[1–0]</td><td rowspan="6">0</td><td colspan="3"><b>DMA Source Space</b>
Specify the memory space referenced as a source by the DMA. In Cache mode, a DMA to Program memory space has some limitations (as described in the instruction cache chapter of the <i>DSP56300 Family Manual</i> in and the chapter on operating modes).</td></tr>
<tr><th>DSS1</th><th>DSS0</th><th>DMA Source Memory Space</th></tr>
<tr><td>0</td><td>0</td><td>X Memory Space</td></tr>
<tr><td>0</td><td>1</td><td>Y Memory Space</td></tr>
<tr><td>1</td><td>0</td><td>P Memory Space</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table>

Note: The lowest 12 K of X data RAM and 12 K of Y data RAM are shared memory that can be accessed by the core and the EFCOP but not by the DMA controller.

4.7 Device Identification Register (IDR)

The IDR is a 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 4-8** 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	$321

Figure 4-5. Identification Register Configuration (Revision A)

4.8 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 4-9** 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	**Design Center Number**	**Sequence Number**	**Manufacturer Identity**	**1**
0000	000110	0000010101	00000001110	1

Figure 4-6. JTAG Identification Register Configuration (Revision 0)

4.9 JTAG Boundary Scan Register (BSR)

The BSR in the DSP56321 JTAG implementation contains bits for all device signals, clock pins, and their associated control signals. All DSP56321 bidirectional pins have a corresponding register bit in the BSR for pin data and are controlled by an associated control bit in the BSR. For details on the BSR, consult the *DSP56300 Family Manual*.

Clock Configuration 5

The DSP56321 features a clock generator (CLKGEN) module that allows the processor to operate at a high internal clock frequency derived from a low-frequency clock input, a feature that offers two immediate benefits. The lower frequency clock input reduces the overall electromagnetic interference generated by a system. Oscillating at different frequencies reduces costs by eliminating the need to add oscillators to a system.

Note: The DSP56321 uses a new CLKGEN module with an integrated Digital Phase-Lock Loop (DPLL) circuit. This CLKGEN module is different from the PLL and clock generator provided by the standard DSP56300 core.

5.1 Overview

The DSP56321 clock generator has a digital PLL (DPLL) that performs

- Clock Input Frequency Division
- Frequency multiplication
- Skew reduction

In addition, the CLKGEN has the following functionality:

- Low power division.
- Internal clock generation
- Source clock switching

This chapter describes the implementation of the CLKGEN module in the DSP56321.

5.2 Clock Generation Scheme

Figure 5-1 shows the principal clock generation scheme in the DSP56321.

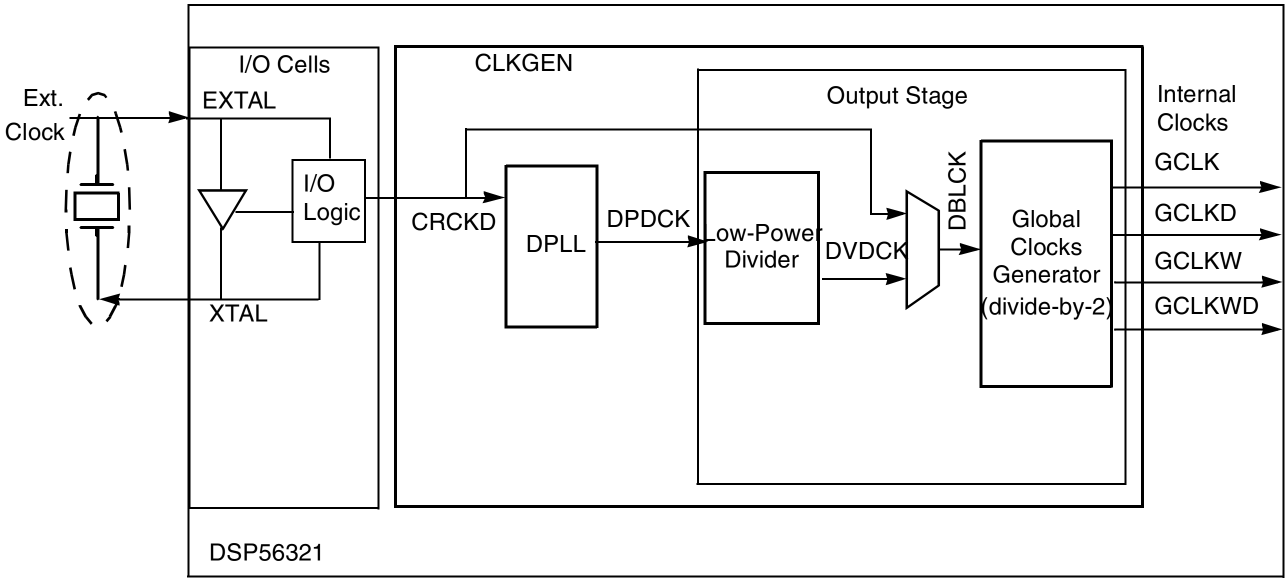


Figure 5-1. Clock Generation Diagram

The clock source can be either an external source applied to EXTAL or a crystal and associated components connected between EXTAL and XTAL. Refer to the *DSP56321 Technical Data* sheet for detailed information about circuit requirements.

5.3 Clock Acronym List

Table 5-1 lists acronyms used in this chapter to describe the CLKGEN module.

Table 5-1. Acronyms

Acronym	Description
BRM	Binary Rate Modulator
CLKGEN	Clock Generator module
CRCKD	CLKGEN Reference Clock to the DPLL
DAC	Digital-to-Analog Converter
DBLCK	Double Clock input to the Global Clock Generator
DF	Post-Division Factor value programmed in the DPLL Control Register (PCTL). See Table 5-2 for details.
DPDCK	DPLL output Clock to the Low Power Divider
DPFD	Digital Phase-Frequency Detector
DPLL	Digital Phase-Lock Loop
DSCR	DPLL Static Control Register. See Table 5-3 for details.
DVDCK	Divided DPLL Clock output
FOL	Frequency-Only Lock mode

Table 5-1. Acronyms (Continued)

Acronym	Description
FPL	Frequency and Phase Lock mode
GCLK, GCLKD, GCLKW, GCLKWD	Global System Clocks used by the DSP56321 internally
MF MFD MFI MFN	DPLL Multiplication Factor is computed from three elements according to the following formula: MF = MFI + (MFN/MFD) MFI, MFN, and MFD are programmed as fields in the DSCR. See Table 5-3 for details. **Note:** MF is limited to the range 5–15. If MFI = 15, then MFN must be programmed as 0.
PCTL	DPLL Control Register. See Table 5-2 for details.
PDF	DPLL Pre-Division Factor programmed in the DSCR. See Table 5-3 for details.
VCO	Voltage-Controlled Oscillator

5.4 CLKGEN External Pins

All external connections link to the CLKGEN module through on-chip I/O cells, as shown in **Figure 5-1**. The following pins are dedicated to the CLKGEN operation:

- EXTAL. External clock/crystal input used to interface an external clock or to support the on-chip oscillator with an external crystal circuit.
- PINIT. During assertion of hardware reset, the value of the PINIT input pin is written into the PCTL DPLL Enable (PEN) bit. After hardware reset is deasserted, the DPLL ignores the PINIT pin, and it can have a different function in the device.
- XTAL. Crystal output connection to support the on-chip oscillator.

5.5 DPLL

This section describes the major DPLL functions. The DPLL functional block diagram is shown in **Figure 5-1**.

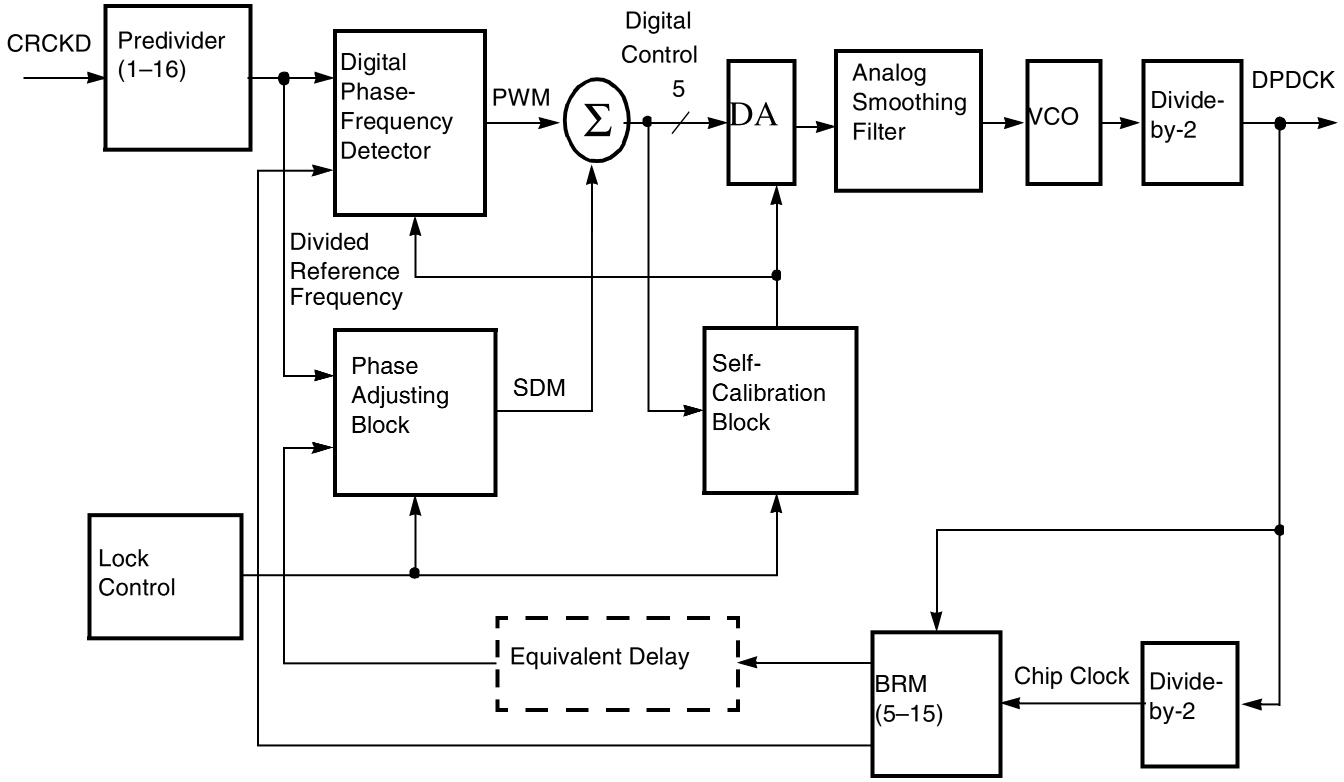


Figure 5-2. DPLL Block Diagram

5.5.1 DPLL Description

The Predivider divides the CLKGEN Reference Frequency (CRCKD) using a division factor with an integer value in the range 1–16. The value is programmed by the PDF bit field in the DSCR.

Note: The Predivider introduces a delay independent of the Pre-Division Factor.

The divided reference clock is fed to the Digital Phase-Frequency Detector (DPFD). The DPFD receives a second input from the Binary Rate Modulator (BRM) output. The BRM receives the Chip Clock, which is the double clock DPDCK divided by 2. The BRM division factor is MF, which is equal to MFI + (MFD/MFN). The values for each of these elements (that is, MFI, MFD, and MFN) are programmed in their respective bit fields in the DSCR.

The DPFD is implemented as a 5-bit asynchronous up/down counter. The counter content is incremented by 1 at a positive edge of the reference clock and decremented by 1 at a positive edge of the divided double clock. Thus, the counter produces a multilevel PWM signal used for control of a frequency of the voltage controlled oscillator (VCO). Digital VCO control is performed by a 5-bit digital-to-analog converter (DAC). A first order lowpass RC filter smooths the multilevel PWM signal from the DAC output. A frequency of the VCO controlled with the LPF output is proportional to a mean level of the PWM signal. The VCO frequency is two times higher than DPDCK. The VCO output is sent through a Divide-by-2 circuit to generate the DPDCK output.

During lock-in time, the divided double clock frequency (that is, the Chip Clock) differs from the divided reference frequency. If the divided reference frequency is higher than the chip clock frequency, the counter content increases and raises the VCO frequency. If the feedback clock (that is, the chip clock) frequency is higher than the divided reference frequency, the counter content decreases and lowers the VCO frequency. The closed feedback loop via the BRM forces the system to go to a steady state, in which both DPFD input signals have equal frequencies (that is, the multilevel PWM signal on the DPFD output has a constant duty cycle). The PWM duty cycle determines the mean DAC output, which in turn tunes the VCO to the proper frequency. The duty cycle may change slowly with variations in a VCO characteristic, such as that caused by changes in temperature, for example.

If the DPLL operates in the FOL mode, the Phase Adjusting Block is disabled and only the DPFD controls the VCO frequency. In this mode, the divided VCO frequency is exactly equal to the divided reference frequency at short time intervals, but there is an unrestricted phase offset between reference and output clocks. This phase offset can drift slowly with temperature and supply voltage changes.

In the FPL mode, the Phase Adjusting Block produced an additional control signal that minimizes the skew between reference and output clocks. Therefore, the skew is constant independent of temperature and supply voltage variations.

The BRM consists of a controlled divider with a division factor from 5 to 16 and a sigma delta modulator. The first or second order of the modulator is selected by BRMO bit. The sigma delta modulator generates pulse series with a mean value of MFN/MFD. If the BRM has the first order, its output may be 0 or 1. The second-order BRM output may be –1, 0, 1, 2. The total mean division factor is MF = MFI + (MFN/MFD). Thus, a current period of the BRM output clock can be variable for a fractional MF but an average clock frequency is equal to DPDCK / (2 × MF).

The DPLL also includes a Self-Calibration Block. This block regulates the DAC gain at start-up in order to ensure that the total loop gain is proportional to the reference frequency and independent of the multiplication factor and process variations.

The Lock Control module contains a set of timers intended to generate all necessary control signals during lock-in time. This module also controls the lock flag.

5.6 Clock Generator Output Stage

The clock generator output stage uses the EXTAL input (PEN = 0 and the DPLL is disabled) or the DPLL output (PEN = 1 and the DPLL is enabled and locked) to generate the core clock.

5.6.1 EXTAL as Clock Source

If the signal comes from EXTAL directly, it feeds directly to the Global Clock Generator. For EXTAL (PEN = 0, PLL disabled) use the following formula:

$$\frac{F_{EXTAL}}{2}$$

5.6.2 DPLL as Clock Source

If the signal comes from the DPLL it passes through the Low Power Divider before going to the Global Clock Generator. The Low-Power Divider (LPD) divides the output frequency of the DPLL by any power of 2 from 2^0 to 2^7. The Division Factor (DF) of the LPD can be modified by changing the value of the PLL Control Register (PCTL) Division Factor bits DF[2–0]. Since the LPD is not in the closed loop of the DPLL, changes in the DF do not cause a loss of lock condition. The result is a significant power savings when the LPD operates in low-power consumption modes as the device is not involved in intensive calculations. When the device is required to exit a low-power mode, it can do so immediately with no time needed for clock recovery or PLL lock.

For the DPLL clock source (PEN = 1, PLL enabled), the operating frequency (F_{CHIP}) can be calculated as:

$$F_{CHIP} = \frac{F_{EXTAL} \times \left(MFI + \frac{MFN}{MFD}\right)}{PDF \times DF}$$

5.6.3 Global Clock Generator

The Global Clock Generator is a divide-by-2 circuit that generates the two-phase chip clock signals to the core and the device peripherals.

5.7 CLKGEN Programming Model

The CLKGEN control registers are two X-I/O mapped 24-bit read/write registers to direct the operation of the on-chip DPLL. The following subsections define these registers.

5.7.1 DPLL Clock Control (PCTL) 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
▨	▨	▨	▨	▨	DF2	DF1	DF0	PEN	XTLD	PSTP	▨

▨ Reserved, write as zeros to maintain future compatibility.

Table 5-2. DPLL Clock Control Register (PCTL)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th>Description</th></tr>
<tr><td>23–7</td><td>reserved</td><td>0</td><td>These bits are reserved for future purposes.</td></tr>
<tr><td>6–4</td><td>DF[2–0]</td><td>011</td><td><b>Division Factor</b>
Defines the low-power divider specified as a power of two in the range from 2^0 to 2^7. Changing the value of the DF[2–0] bits does not cause a loss of lock condition.
<table>
<tr><th>DF[2–0]</th><th>DF Value</th></tr>
<tr><td>000</td><td>2^0</td></tr>
<tr><td>001</td><td>2^1</td></tr>
<tr><td>010</td><td>2^2</td></tr>
<tr><td>011</td><td>2^3</td></tr>
<tr><td>100</td><td>2^4</td></tr>
<tr><td>101</td><td>2^5</td></tr>
<tr><td>110</td><td>2^6</td></tr>
<tr><td>111</td><td>2^7</td></tr>
</table></td></tr>
<tr><td>3</td><td>PEN</td><td>PINIT</td><td><b>DPLL Enable</b>
When PEN is set, the DPLL is enabled; when the DPLL locks, the internal clocks are derived from the DPLL output. When PEN is cleared, the internal clocks are derived directly from the EXTAL signal. Disabling the DPLL minimizes power consumption. The PEN bit may be set or cleared by software any time during the device operation. During hardware reset, this bit is set or cleared based on the value of the DPLL PINIT input.</td></tr>
<tr><td>2</td><td>XTLD</td><td>0</td><td><b>XTAL Disable</b>
Controls the XTAL output from the crystal oscillator on-chip driver. When XTLD is cleared, the XTAL output pin is active, permitting normal operation of the crystal oscillator. When XTLD is set, the XTAL output pin is pulled high, disabling the on-chip oscillator driver. If the on-chip crystal oscillator driver is not used (that is, EXTAL is driven from an external clock source), set XTLD (disabling XTAL) to minimize RFI noise and power dissipation.</td></tr>
<tr><td>1</td><td>PSTP</td><td>0</td><td><b>DPLL Stop State Control</b>
Determines DPLL and on-chip crystal oscillator behavior during the Stop processing state. When PSTP is set, the DPLL and the on-chip crystal oscillator remain operating when the chip is in the Stop state. When PSTP is cleared and the device enters the Stop state, the DPLL and the on-chip crystal oscillator are disabled to reduce power consumption; however, this results in longer recovery time upon exit from the Stop state. To enable rapid recovery when exiting the Stop state (at the cost of higher power consumption during the Stop state), PSTP should be set.
<b>Note:</b> PSTP and PEN are related. When PSTP is set and PEN is cleared, the on-chip crystal oscillator remains operating in the Stop state, but the DPLL is disabled. This power saving feature enables rapid recovery from the Stop state when you operate the device with an on-chip oscillator and with the DPLL disabled.
<table>
<tr><th rowspan="2">PSTP</th><th rowspan="2">PEN</th><th colspan="2">Operation during Stop State</th><th rowspan="2">Recovery Time from Stop State</th><th rowspan="2">Power Consumption during Stop State</th></tr>
<tr><th>DPLL</th><th>Oscillator (XTLD=0)</th></tr>
<tr><td>0</td><td>X</td><td>Disabled</td><td>Disabled</td><td>Long</td><td>Minimal</td></tr>
<tr><td>1</td><td>0</td><td>Disabled</td><td>Enabled</td><td>Short</td><td>Low</td></tr>
<tr><td>1</td><td>1</td><td>Enabled</td><td>Enabled</td><td>Short</td><td>High</td></tr>
</table></td></tr>
</table>

Table 5-2. DPLL Clock Control Register (PCTL) (Continued)

Bit Number	Bit Name	Reset Value	Description
0	reserved	0	This bit is reserved. **Note:** Although this bit is cleared after reset, it enables the output of a reserved signal used for Freescale development purposes. To prevent possible noise generation, Freescale recommends writing a 1 to this bit after reset to disable the reserved signal output.

5.7.2 DPLL Static Control Register (DSCR)

23	22	21	20	19	18	17	16	15	14	13	12
BRMO	PLM	PDF3	PDF2	PDF1	PDF0	MFD6	MFD5	MFD4	MFD3	MFD2	MFD1
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MFD0	MFN6	MFN5	MFN4	MFN3	MFN2	MFN1	MFN0	MFI3	MFI2	MFI1	MFI0

Table 5-3. DPLL Static Control Register (DSCR) Bit Definitions

Bit Number	Bit name	Reset Value	Description	
23	BRMO	1	**Binary Rate Modulation Order** The value of BRMO determines the output order used by the Binary Rate Modulator (BRM). If BRMO is cleared (0), the BRM uses a first order output. If BRMO is set (1), the BRM uses a second order output. If the DPLL Multiplication Factor Denominator (MFD) is < 8, then use a first order output (that is, BRMO = 0). If MFD $\geq$ 8, use a second order output (that is, BRMO = 1). If the DPLL Multiplication Factor Numerator (MFN) is 0, this bit is ignored. **Note:** The DPLL will not operate correctly if this bit is not configured appropriately.	
22	PLM	1	**Phase Lock Mode** When this bit is cleared (0), the DPLL operates in Frequency Only Lock (FOL) mode. When this bit is set (1), the DPLL operates in Frequency and Phase Lock (FPL) mode. FPL mode can be used for both an integer and fractional multiplication factor, but phase skew reduction is accomplished only for the integer MF.	
21–18	PDF[3–0]	0000	**Pre-Division Factor (PDF)** Defines the DPLL PDF value in the range 1–16. **Note:** The value of PDF bits written into DSCR must be the factor value minus 1.	
			PDF[3–0]	**PDF Value**
			0000	1
			0001	2
			0010	3
			. . .	. . .
			1111	16

Table 5-3. DPLL Static Control Register (DSCR) Bit Definitions (Continued)

Bit Number	Bit name	Reset Value	Description	
17–11	MFD[6–0]	0000000	**Multiplication Factor Denominator (MFD)** Defines the denominator of the fractional part of the Multiplication Factor (MF). The MFD can be any integer from 1 to 128. **Note:** The value of MFD bits written into DSCR must be the denominator value minus 1. The MFD value must be higher than the MFN value	
			MFD[6–0]	**MFD Value**
			$00	1
			$01	2
			$02	3
			. . .	
			$7E	127
			$7F	128
10–4	MFN[6–0]	0000000	**Multiplication Factor Numerator (MFN)** Defines the numerator of the fractional part of the MF. The MFN can be any integer from 0 to 127. **Note:** The total MF is limited to a maximum of 15. Therefore, if the MFI is programmed as 15 (that is, MFI[3–0] = 1111), MFN must equal 0.	
			MFN[6–0]	**MFN Value**
			$00	0
			$01	1
			$02	2
			. . .	
			$7E	126
			$7F	127

Table 5-3. DPLL Static Control Register (DSCR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit name</th><th>Reset Value</th><th colspan="2">Description</th></tr>
<tr><td rowspan="13">3–0</td><td rowspan="13">MFI[3–0]</td><td rowspan="13">1000</td><td colspan="2"><b>Multiplication Factor Integer (MFI)</b>—Defines the integer part of the MF. The MFI can be any integer from 5 to 15.
<b>Note:</b> The total MF is limited to a maximum of 15. Therefore, if the MFI is programmed as 15 (that is, MFI[3–0] = 1111), MFN must equal 0.</td></tr>
<tr><td><b>MFI[3–0]</b></td><td><b>MFI Value</b></td></tr>
<tr><td>0000</td><td>5</td></tr>
<tr><td>0001</td><td>5</td></tr>
<tr><td>0010</td><td>5</td></tr>
<tr><td>0011</td><td>5</td></tr>
<tr><td>0100</td><td>5</td></tr>
<tr><td>0101</td><td>5</td></tr>
<tr><td>0110</td><td>6</td></tr>
<tr><td>0111</td><td>7</td></tr>
<tr><td>1000</td><td>8</td></tr>
<tr><td>.
.
.</td><td></td></tr>
<tr><td>1110</td><td>14</td></tr>
<tr><td></td><td></td><td></td><td>1111</td><td>15</td></tr>
</table>

Programming the Peripherals 6

When peripherals are programmed in a given application, a number of possible modes and options are available for use. Chapters 6 through 10 describe in detail the possible modes and configurations for peripheral registers and ports. This chapter presents general guidelines for initializing the peripherals. These guidelines include a description of how the control registers are mapped in the DSP56321, data transfer methods that are available when the various peripherals are used, and information on General-Purpose Input/Output (GPIO) configuration.

6.1 Peripheral Initialization Steps

Each peripheral has its own initialization process. However, all four peripherals share some common steps, which follow:

1. Determine the Register values to be programmed, using the following steps:
   a. Find the peripheral register descriptions in the manual.
   b. Choose the appropriate modes to configure for a given application.
   c. Determine the bit settings for programming those modes.
2. Make sure the peripheral is in individual reset state or disabled. Peripheral registers should not be modified while the peripheral is active.
3. Configure the registers by writing the predetermined values from step 1 into the appropriate register locations.
4. Enable the peripheral. Once the peripheral is enabled, it operates according the programmed modes determined in step 1.

For detailed initialization procedures unique to each peripheral, consult the initialization section within each peripheral chapter.

6.2 Mapping the Control Registers

The I/O peripherals are controlled through registers mapped to the top 128 words of X-data memory ($FFFF80–$FFFFFF). Referred to as the internal I/O space, the control registers are accessed by move (MOVE, MOVEP) instructions and 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 B**, *Programming Reference*

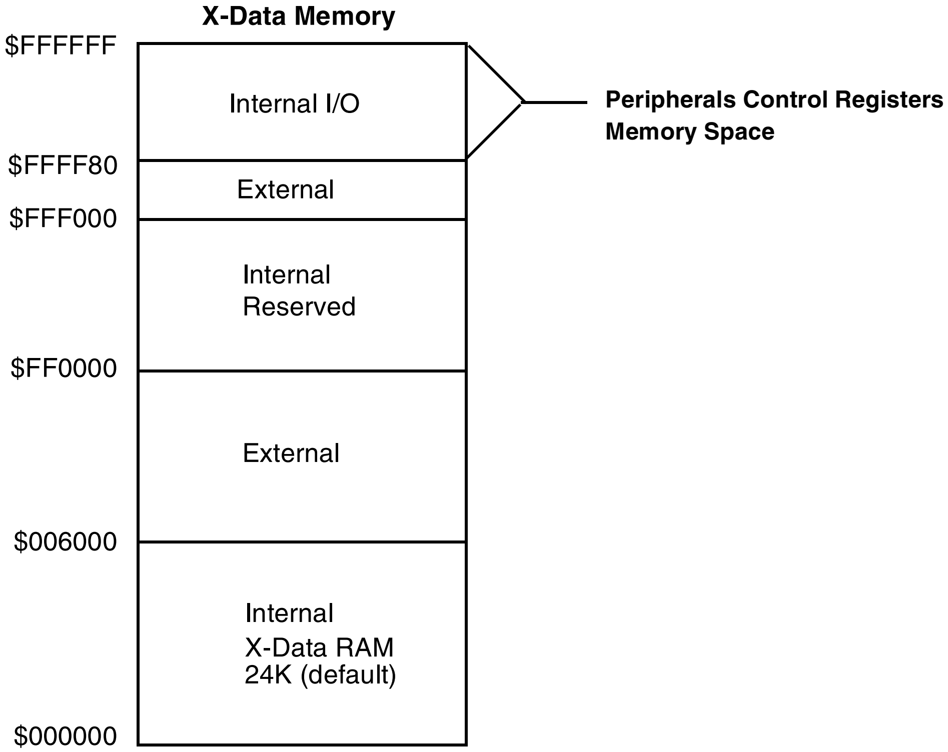


Figure 6-1. Memory Mapping of Peripherals Control Registers

6.3 Reading Status Registers

Each peripheral has a read-only status register that indicate the state of the peripheral at a given time. The HI08, ESSI, and SCI have dedicated status registers. The triple timer has status bits embedded within a control/status register. Changes in the status bits can generate interrupt conditions. For example, the HI08 has a host status register with two host flag bits that can be encoded by the host to generate an interrupt in the DSP.

6.4 Data Transfer Methods

Peripheral I/O on the DSP56321 can be accomplished in three ways:

- Polling
- Interrupts
- DMA

6.4.1 Polling

Polling is the easiest method for data transfers. When polling is chosen, the DSP56321 core continuously checks a specified register flag waiting for an event to happen. One example would be setting an overflow flag in one of the Timers. Once the event occurs, the DSP56321 is free to continue with its next task. However, while it is waiting for the event to occur, the DSP56321 core is not executing any other code. Polling is the easiest transfer method since it does not require register initialization, but it is also the least efficient use of the DSP core.

Each peripheral has its own set of flags which may be polled to determine when data is ready to be transferred. For example, the ESSI control registers provide bits that tell the core when data is ready to be transferred to or from the peripheral. The core polls these bits to determine when to interact with the peripheral. Similar flags exist for each peripheral.

Example 6-1 shows software polling programmed in an application using the HI08.

Example 6-1. Software Polling

```
jclr#1,x:M_HSR,*           ; loop if HSR[1]:HTDE=0
movey:(TBUFF_PTR)+,x1      ; move data to x1
```

In this example, the core waits until the Host Status Register's (HSR) Host Transmit Data Empty (HTDE) flag is set. When the flag is set, the core moves data from Y memory to the X1 register.

6.4.2 Interrupts

Interrupts are more efficient than polling, but interrupts also require additional register initialization. Polling requires the core to remain busy checking a flag in a specified control register and therefore does not allow the core to execute other code at the same time. For interrupts, you can initialize the interrupt so it is triggered off one of the same flags that can also be polled. Then the core does not have to continuously check a flag. Once the interrupt is initialized and the flag is set, the core is notified to execute a data transfer. Until the flag is set, the core can remain busy executing other sections of code.

When an interrupt occurs, the core execution flow jumps to the interrupt start address defined in **Table B-4** in **Appendix B**, *Programming Reference*. It executes code starting at the interrupt address. If it is a short interrupt (i.e., the service routine is two opcodes long), the code automatically returns to the original program flow after executing two opcodes with no impact to the pipeline. Otherwise, if a longer service routine is required the programmer can place a jump-to-subroutine (JSR) instruction at the interrupt service address. In this case, the program executes that service routine and continues until a return-from-interrupt (RTI) instruction executes. The execution flow then resumes from the position the program counter was in before the interrupt was triggered.

Configuring interrupts requires two steps:

1. Setting up the interrupt routine
   - **a.** The interrupt handler is located at the interrupt starting address.
   - **b.** The interrupt routines can be short (only two opcodes long) or long (more than two opcodes and requiring a JSR instruction).
2. Enabling the interrupts

a. Set the corresponding bits in the applicable peripheral control register.

b. Enable peripheral interrupts in the Interrupt Priority Register (IPRP).

c. Enable global interrupts in the Mode Register (MR) portion of the Status Register (SR).

Events that change bits in the peripheral control registers can then trigger the interrupt. Depending on the peripheral, from two to six peripheral interrupt sources are available to the programmer.

Example 6-2 shows a short interrupt programmed for the HI08. The main program enables the Host Receive Interrupt in the Host Control Register (HCR). When the interrupt is triggered during code execution, the core processing jumps to the Host Receive Interrupt routine location at p:$60 and executes the code there. Since this is a short interrupt, the core returns to normal code execution after executing the two move instructions, and an RTI instruction is not necessary.

Example 6-2. Interrupts

```
   bset#M_HRIE,x:M_HCR ; enable host receive interrupt


; Short Interrupt Routine

   orgP:$60

   movepx:M_HRX,x1     ; HI08 Receive Data Full interrupt

   movex1,y:(r0)+
```

6.4.3 DMA

The direct memory access (DMA) controller permits data transfers between internal/external memory and/or internal/external I/O in any combination without the intervention of the DSP56321 core. Dedicated DMA address and data buses and internal memory partitioning ensure that a high level of isolation is achieved so the DMA operation does not interfere with the core operation or slow it down. The DMA controller moves data to/from the peripheral transmit/receive registers. The programmer can use the DMA control registers to configure sources and destinations of data transfers. Depending on the peripheral, there are one to four peripheral request sources. DMA is the most efficient method of data transfer. Core intervention is not required after the DMA channel is initialized.

Table 6-1. DMA-Accessible Registers

Block	Register	DMA	
		Read	Write
ESSI	TX0	No	Yes
	TX1	No	Yes
	TX2	No	Yes
	RX	Yes	No
SCI	SRX	Yes	No
	STX	No	Yes
EFCOP	FDIR	No	Yes
	FDOR	Yes	No
HI08	HTX	No	Yes
	HRX	Yes	No
Timer			

Example 6-3 shows a DMA configuration for transferring data to the Host Transmit register of the HI08.

Example 6-3. DMA Transfers

```
bclr#M_D1L0,x:M_IPRC       ; disable DMA1 interrupts
bclr#M_D1L1,x:M_IPRC
movep#TBUFF_START,x:M_DSR1 ; DMA1 source is transmit buffer
movep#M_HTX,x:M_DDR1       ; DMA1 destination is HTX
movep#TBUFF_SIZE-1,x:M_DCO1; DMA1 count is the full buffer
movep#INIT_DCR1,x:M_DCR1   ; init. DMA1 control register
```

DMA requires more initialization code and consideration of DMA modes. However, it is the most efficient use of core resources. After these registers are programmed, you must enable the DMA by triggering a DMA request off one of the peripheral control flags or enabling it in normal program flow or an interrupt service routine.

6.4.4 Advantages and Disadvantages

Polling is the easiest method to implement, but it requires a large amount of DSP56321 core processing power. The core cannot be involved in other processing activities while it is polling receive and transmit ready bits. Interrupts require more code, but the core can process other routines while waiting for data I/O. An interrupt is generated when data is ready to be transferred to or from the peripheral device. DMA requires even less core intervention, and the setup code is minimal, but the DMA channels must be available.

Note: Do not use interrupt requests and DMA requests simultaneously.

6.5 General-Purpose Input/Output (GPIO)

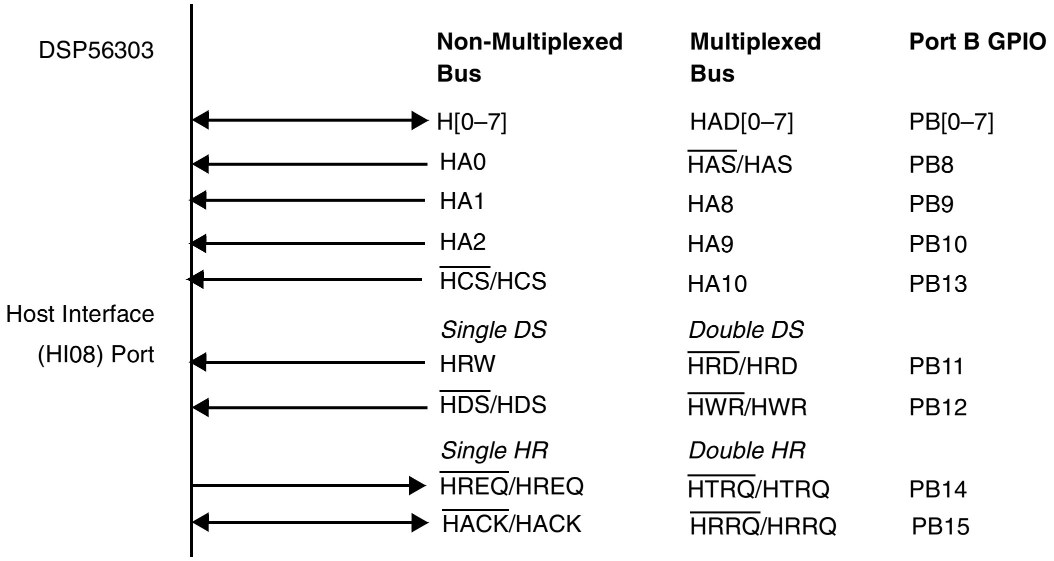
The DSP56321 provides 34 bidirectional signals that can be configured as GPIO signals or as peripheral dedicated signals. No dedicated GPIO signals are provided. All of these signals are GPIO by default after reset. The control register settings of the DSP56321 peripherals determine whether these signals function as GPIO or as peripheral dedicated signals. This section tells how signals can be used as GPIO.

Chapter 2, *Signals/Connections* details the special uses of the 34 bidirectional signals. These signals fall into five groups and are controlled separately or as a group:

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

6.5.1 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. Three registers control the GPIO functionality of Port B: host control register (HCR), host port GPIO data register (HDR), and host port GPIO direction register (HDDR). **Chapter 7**, *Host Interface (HI08)* discusses these registers.



DSP56303
Host Interface
(HI08) Port
Non-Multiplexed Bus
Multiplexed Bus
Port B GPIO
H[0–7]
HAD[0–7]
PB[0–7]
HA0
$\overline{\text{HAS}}$/HAS
PB8
HA1
HA8
PB9
HA2
HA9
PB10
$\overline{\text{HCS}}$/HCS
HA10
PB13
Single DS
Double DS
HRW
$\overline{\text{HRD}}$/HRD
PB11
$\overline{\text{HDS}}$/HDS
$\overline{\text{HWR}}$/HWR
PB12
Single HR
Double HR
$\overline{\text{HREQ}}$/HREQ
$\overline{\text{HTRQ}}$/HTRQ
PB14
$\overline{\text{HACK}}$/HACK
$\overline{\text{HRRQ}}$/HRRQ
PB15

Figure 6-2. Port B Signals

6.5.2 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. Three registers control the GPIO functionality of Port C: Port C control register (PCRC), Port C direction register (PRRC), and Port C data register (PDRC). **Chapter 8**, *Enhanced Synchronous Serial Interface* discusses these registers.

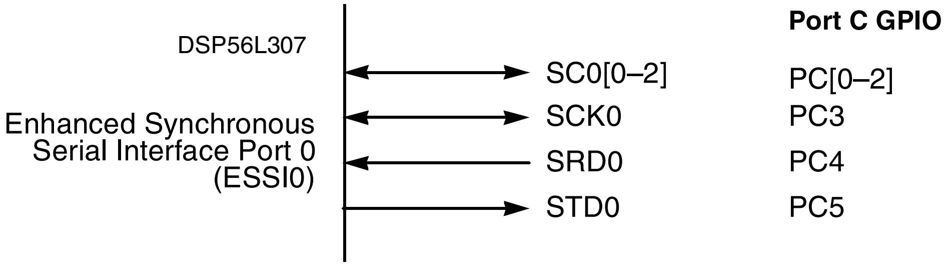


Figure 6-3. Port C Signals

6.5.3 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. Three registers control the GPIO functionality of Port D: Port D control register (PCRD), Port D direction register (PRRD), and Port D data register (PDRD). **Chapter 8**, *Enhanced Synchronous Serial Interface* discusses these registers.

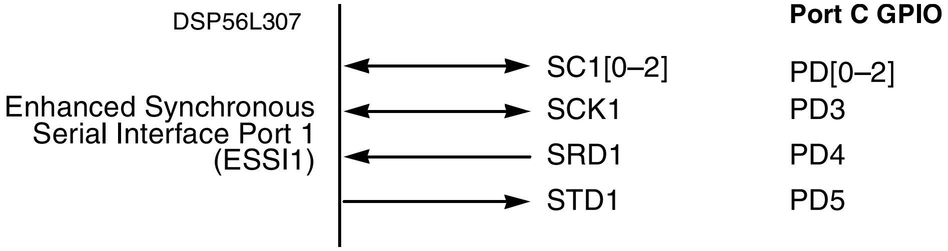


Figure 6-4. Port D Signals

6.5.4 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. Three registers control the GPIO functionality of Port E: Port E control register (PCRE), Port E direction register (PRRE), and Port E data register (PDRE). **Chapter 9**, *Serial Communication Interface* discusses these registers.

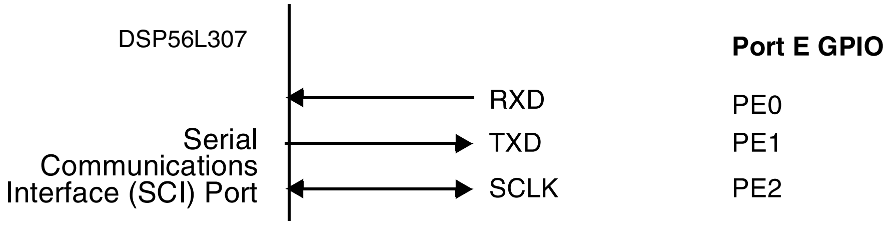


Figure 6-5. Port E Signals

6.5.5 Triple Timer Signals and Registers

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). **Chapter 10**, *Triple Timer Module* discusses these registers.

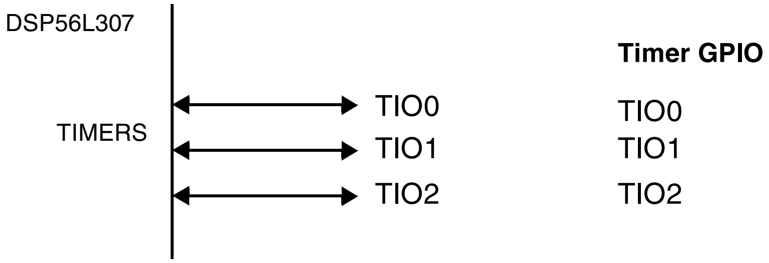


Figure 6-6. Triple Timer Signals

Host Interface (HI08) 7

The host interface (HI08) is a byte-wide, full-duplex, double-buffered parallel port that can connect directly to the data bus of a host processor. The HI08 supports a variety of buses and provides glueless connection with a number of industry-standard microcomputers, microprocessors, and DSPs. The HI08 signals not used to interface to the host can be configured as GPIO signals, up to a total of 16.

7.1 Features

The HI08 host is a slave device that operates asynchronously to the DSP core and host clocks. Thus, the HI08 peripheral has a host processor interface and a DSP core interface. This section lists the features of the host processor and DSP core interfaces.

7.1.1 DSP Core Interface

- Mapping:
  - Registers are directly mapped into eight internal X data memory locations.
- Data word:
  - DSP56321 24-bit (native) data words are supported, as are 8-bit and 16-bit words.
- Handshaking protocols:
  - Software polled
  - Interrupt driven
  - Core DMA accesses
- Instructions:
  - Memory-mapped registers allow the standard MOVE instruction to transfer data between the DSP56321 and external hosts.
  - A special MOVEP instruction for I/O service capability using fast interrupts.
  - Bit addressing instructions (for example, BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, JSSET) simplify I/O service routines.

7.1.2 Host Processor Interface

- Sixteen signals support non-multiplexed or multiplexed buses:
  - H[0–7]/HAD[0–7] host data bus (H[0–7]) or host multiplexed address/data bus (HAD[0–7])
  - 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/$\overline{\mathrm{HRD}}$ read/write select (HRW) or read strobe ($\overline{\mathrm{HRD}}$)
- — $\overline{\mathrm{HDS}}$/$\overline{\mathrm{HWR}}$ data strobe ($\overline{\mathrm{HDS}}$) or write strobe ($\overline{\mathrm{HWR}}$)
- — $\overline{\mathrm{HCS}}$/HA10 host chip select ($\overline{\mathrm{HCS}}$) or host address line (HA10)
- — $\overline{\mathrm{HREQ}}$/$\overline{\mathrm{HTRQ}}$ host request ($\overline{\mathrm{HREQ}}$) or host transmit request ($\overline{\mathrm{HTRQ}}$)
- — $\overline{\mathrm{HACK}}$/$\overline{\mathrm{HRRQ}}$ host acknowledge ($\overline{\mathrm{HACK}}$) or host receive request ($\overline{\mathrm{HRRQ}}$)

Note: The signals in the above list that are shown as asserted low (for example, $\overline{\mathrm{HRD}}$) all have programmable polarity. The default value following reset is shown in the above list.

- ■ Mapping:
  - — HI08 registers are mapped into eight consecutive locations in the host’s external bus address space.
  - — The HI08 acts as a memory or I/O-mapped peripheral for microprocessors, microcontrollers, and so forth.
- ■ 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.)
- ■ Data word: 8 bits
- ■ Dedicated interrupts:
  - — Separate request 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
    - Freescale HC11
    - Hitachi H8
    - 8051 family
    - Thomson P6 family

- Minimal glue logic (pull-ups, pull-downs) required to interface to
  - ISA bus
  - Freescale 68K family
  - Intel X86 family

7.2 Host Port Signals

The host port signals are discussed in **Chapter 2**, *Signals/Connections*. Each host port signal can be programmed as a host port signal or as a GPIO signal, PB[0–15]. See **Table 7-1** through **Table 7-3**.

Table 7-1. HI08 Signal Definitions for Operational Modes

HI08 Port Signal	Multiplexed Address/Data Bus Mode	Non-multiplexed Bus Mode	GPIO Mode
HAD[0–7]	HAD[0–7]	H[0–7]	PB[0–7]
$\overline{\text{HAS}}$/HA0	$\overline{\text{HAS}}$/HAS	HA0	PB8
HA8/HA1	HA8	HA1	PB9
HA9/HA2	HA9	HA2	PB10
$\overline{\text{HCS}}$/HA10	HA10	$\overline{\text{HCS}}$/HCS	PB13

Table 7-2. HI08 Data Strobe Signals

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

Table 7-3. HI08 Host Request Signals

HI08 Port Signal	Single Host Request Mode	Double Host Request Mode	GPIO Mode
$\overline{\text{HREQ}}$/ $\overline{\text{HTRQ}}$	$\overline{\text{HREQ}}$/HREQ	$\overline{\text{HTRQ}}$/HTRQ	PB14
$\overline{\text{HACK}}$/ $\overline{\text{HRRQ}}$	$\overline{\text{HACK}}$/HACK	$\overline{\text{HRRQ}}$/HRRQ	PB15

The HI08 port can operate in multiplexed or non-multiplexed mode. In multiplexed mode (HPCR[11]:HMUX=1), the lower eight address signals multiplex with the eight data lines. In non-multiplexed mode (HPCR[11]:HMUX=0), the HI08 requires a chip select signal and three address lines to select one of the eight registers accessible to the host. Eight lines are used for data. The HI08 port can also be programmed to use a single or dual read/write data strobe and single or double host request.

Software and hardware resets clear all DSP-side control registers and configure the HI08 as GPIO. 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 details, see **Section 7.6.3**, *Host Data Direction Register (HDDR)*, on page 7-14 and **Section 7.6.4**, *Host Data Register (HDR)*, on page 7-15.

7.3 Overview

The HI08 is partitioned into two register banks, as **Figure 7-1** shows. The host-side bank is accessible only to the host, and the DSP-side register bank is accessible only to the DSP core. For the host, the HI08 appears as eight byte-wide locations mapped in its external address space. The DSP-side registers appear to the DSP core as six 24-bit registers mapped into internal I/O X memory space and therefore accessible via standard DSP56300 instructions and addressing modes. In GPIO mode, two additional registers (HDDR and HDR) are related to the HI08 peripheral. The separate receive and transmit data paths are double buffered for efficient, high speed asynchronous transfers. The host-side transmit data path (host writes) is also the DSP-side receive path; the host-side receive data path (host reads) is also the DSP-side transmit path. The Receive (RXH:RXM:RXL) and Transmit Data Registers (TXH:TXM:TXL) use the same host address. During host writes to these addresses, the data is transferred to the Transmit Data Registers while reads are performed from the Receive Data Registers.

7.4 Operation

The HI08 is a slave-only device, so the host is the master of all bus transfers. In host-to-DSP transfers, the host writes data to the Transmit Data Registers (TXH:TXM:TXL). In DSP-to-host transfers the host reads data from the Receive Data Registers (RXH:RXM:RXL). The DSP side has access only to the Host Receive Data Register (HRX) and the Host Transmit Data Register (HTX). Data automatically moves between the host-side data registers and the DSP-side data registers when it is available. This double-buffered mechanism allows for fast data transfers but creates a “pipeline” that can either stall communication (if the pipeline is either full or empty) or cause erroneous data transfers (new data to be overwritten or old data to be read twice). The HI08 port has several handshaking mechanisms to counter these buffering effects.

Suppose the host is writing several pieces of data to the HI08 port. The host first uses one of the handshaking protocols to determine whether any data previously written to the Transmit Data Registers (TXH:TXM:TXL) has successfully transferred to the DSP side. If the host-side Transmit Data Registers (TXH:TXM:TXL) are empty, the host writes the data to these registers. The transfer to the DSP-side Host Receive Data Register (HRX) occurs only if HRX is empty (that is, the DSP has read it). The DSP core then uses an appropriate handshaking protocol to move data from the HRX to the receiving buffer or register. Without handshaking, the host might overwrite data not transferred to the DSP side or the DSP might receive stale data.

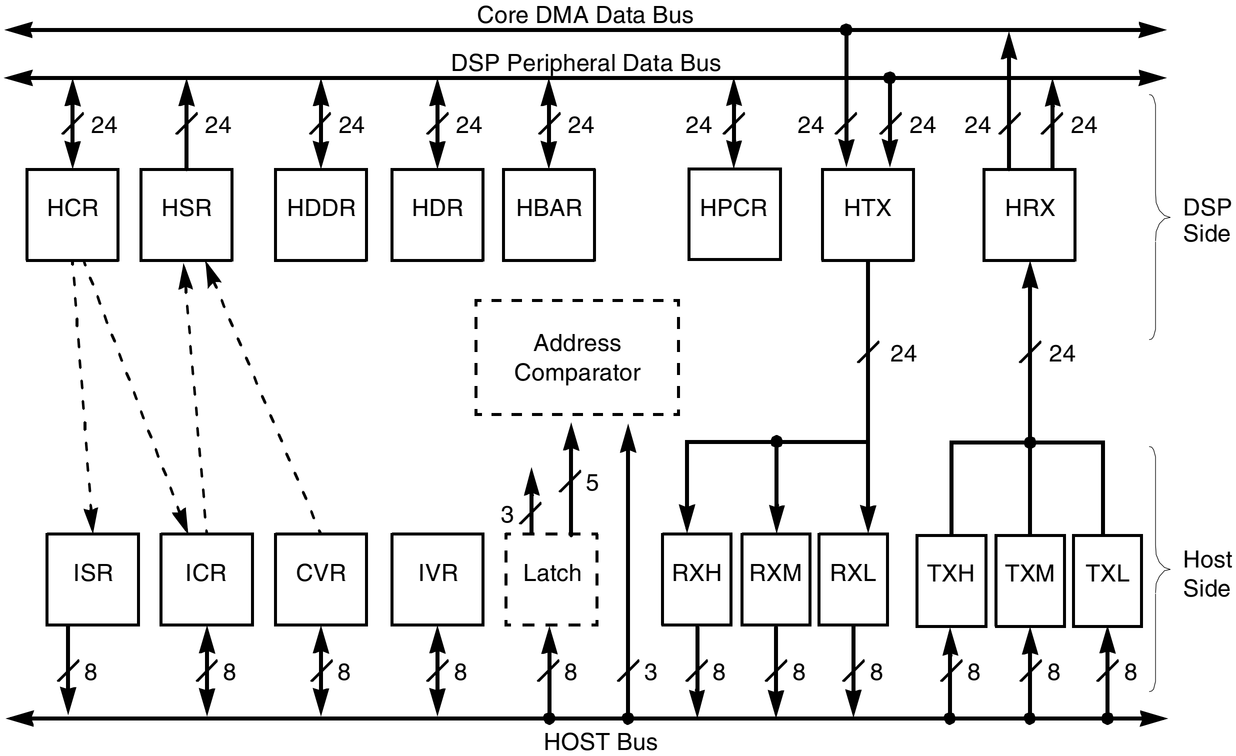
DSP-Side Registers

Control Registers

HCR = Host Control Register
HSR = Host Status Register
HPCR = Host Port Control Register
HBAR = Host Base Address Register

Data Registers

HTX = Host Transmit Register
HRX = Host Receive Register
HDDR = Host Data Direction Register
HDR = Host Data Register



Host-Side Registers

Control Registers

ISR = Interface Status Register
ICR = Interface Control Register
CVR = Command Vector Register
IVR = Interrupt Vector Register

Data Registers

RXH = Receive Register High
RXM = Receive Register Middle
RXL = Receive Register Low
TXH = Transmit Register High
TXM = Transmit Register Middle
TXL = Transmit Register Low

Figure 7-1. HI08 Block Diagram

Similarly, when the host performs multiple reads from the HI08 port Receive Data Registers (RXH:RXM:RXL), the DSP side uses an appropriate handshaking protocol to determine whether any data previously written to the Host Transmit Register (HTX) has successfully transferred to the host-side registers. If HTX is empty, the DSP writes the data to this register. Data transfers to the host-side Receive Data Registers (RXH:RXM:RXL) occur only if they are empty (that is, the host has read them). The host can then use any of the available handshaking protocols to determine whether more data is ready to be read.

The DSP56321 HI08 port offers the following handshaking protocols for data transfers with the host:

- Software polling
- Interrupts
- Core DMA access
- Host requests

The choice of which protocol to use is based on such system constraints as the amount of data to be transferred, the timing requirements for the transfer, and the availability of such resources as processing bandwidth and DMA channels. All of these constraints are discussed in the following sections. The transfers described here occur asynchronously between the host and the DSP; each transferring data at its own pace. However, use of the appropriate handshaking protocol allows data transfers to occur at optimum rates.

7.4.1 Software Polling

Software polling is the simplest data transfer method to use, but it demands the greatest amount of the core's processing power. Status bits are provided for the host or the DSP core to test and determine if the data registers are empty or full. However, the DSP core cannot be involved in other processing activities while it is polling these status bits.

On the DSP side, for transfers from the DSP to the host (host reads), the DSP core must determine the state of Host Transmit Data register (HTX). In transfers from the host to the DSP (host writes), the DSP side should determine the state of the Host Receive Data Register (HRX). Thus, two bits are provided to the core for polling:

- the Host Transmit Data Empty (HTDE) bit in the Host Status register (HSR[1]:HTDE)
- the Host Receive Data Full (HRDF) bit in the Host Status register (HSR[0]:HRDF)

A similar mechanism is available on the host-side to determine the state of the Transmit Registers (TXH:TXM:TXL) and Receive Registers (RXH:RHM:RHL). Two bits are provided to the host for polling:

- the Transmit Data Empty (TXDE) bit in the Interface Status Register (ISR[1]:TXDE)
- the Receive Data Full (RXDF) bit in the Interface Status Register (ISR[0]:RXDF)

The HI08 also offers four general-purpose flags for communication between the host and the DSP. The DSP-side uses the HSR Host Flag bits (HCR[4–3]=HF[3–2]) to pass application-specific information to the host. The status of HF3–HF2 is reflected in the host-side ISR Host Flag bits (ISR[4–3]=HF[3–2]). Similarly, the host side can use the ICR Host Flag bits (ICR[4–3]=HF[1–0]) to pass application-specific information to the DSP. The status of HF[1–0] is reflected in the DSP-side HSR Host Flag bits
(HSR[4–3]=HF[1–0]).

7.4.2 Core Interrupts and Host Commands

The HI08 can request interrupt service from the DSP56321 core. The DSP56321 core 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 interface sets the appropriate bit in the HSR, generating an interrupt request to the DSP56321 interrupt controller (see **Figure 7-2**). The DSP56321 acknowledges interrupts by jumping to the appropriate interrupt service routine. The following DSP core interrupts are possible from the HI08 peripheral:

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

These interrupts are maskable via the Host Receive Interrupt Enable bit (HCR[0]=HRIE), the Host Transmit Interrupt Enable bit (HCR[1]=HTIE), and the Host Command Interrupt Enable bit (HCR[2]=HCIE), respectively. Receive Data Full and Transmit Data Empty interrupts move data to/from the HTX and HRX data registers. The DSP interrupt service routine must read or write the appropriate HI08 data register (HRX or HTX) to clear the interrupt condition.

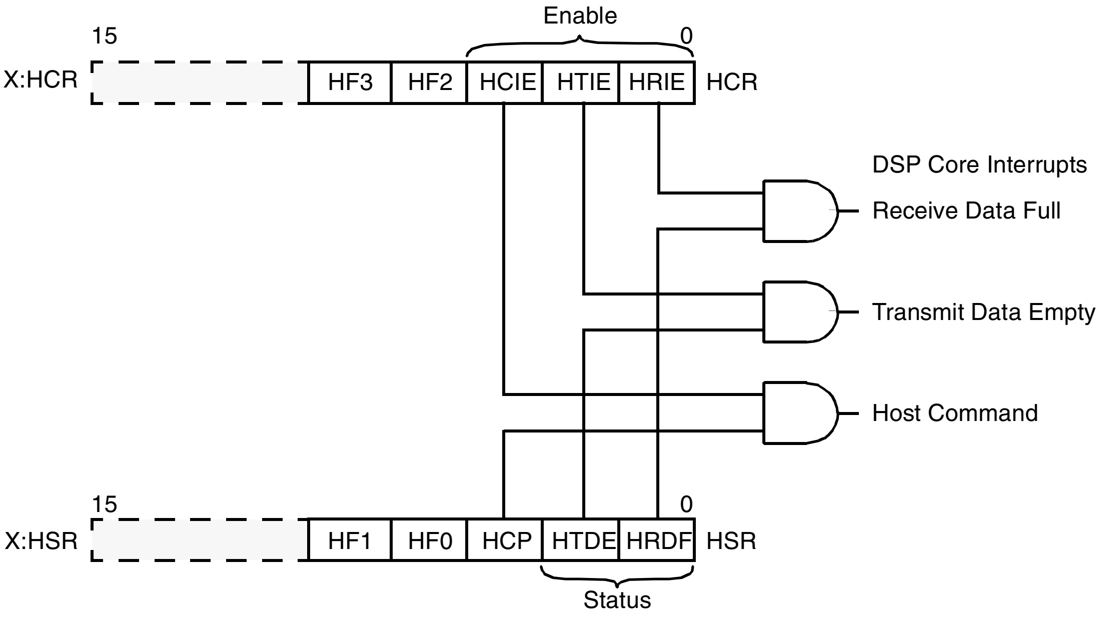


Figure 7-2. HI08 Core Interrupt Operation

Host commands allow the host to issue command requests to the DSP by selecting any of 128 DSP interrupt routines for execution. For example, the host may issue a command via the HI08 that sets up and enables a DMA transfer. The DSP56321 processor has reserved interrupt vector addresses for application-specific service routines. However, this flexibility is independent of the data transfer mechanisms in the HI08 and allows the host to force execution of any interrupt handler (for example, SSI, SCI, IRQx, and so on).

To enable Host Command interrupts, the HCR[2]=HCIE bit is set on the DSP side. The host then uses the Command Vector Register (CVR) to start an interrupt routine. The host sets the Host Command bit (CVR[7]=HC) to request the command interrupt and the seven Host Vector bits CVR[6–0]=HV[6–0] to select the interrupt address to be used. When the DSP core recognizes the host command interrupt, the address of the interrupt taken is 2xHV. For host command interrupts, the interrupt acknowledge from the DSP56321 program controller clears the pending interrupt condition.

Note: When the DSP enters Stop mode, the HI08 pins are electrically disconnected internally, thus disabling the HI08 until the core leaves Stop mode. Do *not* issue a STOP command via the HI08 unless some other mechanism for exiting this mode is provided.

7.4.3 Core DMA Access

The DSP56300 family Direct Memory Access (DMA) controller permits transfers between internal or external memory and I/O without any core intervention. A DMA channel can be set up to transfer data to/from the HTX and HRX data registers, freeing the core to use its processing power on functions other than polling or interrupt routines for the HI08. DMA may well be the best method to use for data transfers, but it requires that one of the six DMA channels be available for use. Two HI08 DMA sources are possible, as **Table 7-4** shows. Refer to the *DSP56300 Family Manual* to learn about DMA accesses.

Table 7-4. DMA Request Sources

Requesting Device	DCRx[15–11]=DRS[4–0]
Host Receive Data Full (HRDF=1)	10011
Host Transmit Data Empty (HTDE=1)	10100

Note: DMA transfers do not access the host bus. The host must determine when data is available in the host-side data registers using an appropriate polling mechanism.

7.4.4 Host Requests

A set of signal lines allow the HI08 to request service from the host. The request signal lines normally connect to the host interrupt request pins (IRQx) and indicate to the host when the DSP HI08 port requires service. The HI08 can be configured to use either a single Host Request (HREQ) line for both receive and transmit requests or two signal lines, a Host Transmit Request (HTRQ) and a Host Receive Request (HRRQ), for each type of transfer. Host requests are enabled on both the DSP-side and host-side. On the DSP side, the HPCR Host Request Enable bit (HPCR[4]=HREN) is set to enable host requests. On the host side, clearing the ICR Double Host Request bit (ICR[2]=HDRQ) configures the HI08 to use a single request line (HREQ). Setting the ICR[2]=HDRQ bit enables both transmit and request lines to be used. Further, the host uses the ICR Receive Request Enable bit (ICR[0]=RREQ) and the ICR Transmit Request Enable bit (ICR[1]=TREQ) to enable receive and transmit requests, respectively. When host requests are enabled, the host request pins operate as shown in **Figure 7-3**.

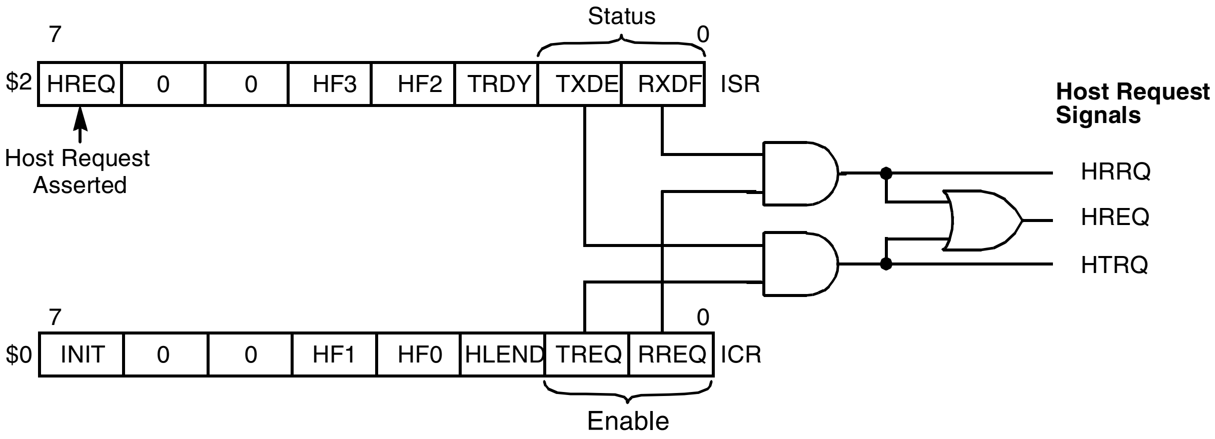


Figure 7-3. HI08 Host Request Structure

Table 7-5 shows the operation of the HREQ pin when a single request line is used. The host can test these ICR bits to determine the interrupt source.

Table 7-5. HREQ Pin Operation In Single Request Mode (ICR[2]=HDRQ=0)

ICR[1]=TREQ	ICR[0]=RREQ	HREQ Pin
0	0	No interrupts
0	1	RXDF request enabled
1	0	TXDE request enabled
1	1	RXDF and TXDE request enabled

Table 7-6 shows the operation of the transmit request (HTRQ) and receive request (HRRQ) lines with dual host requests enabled.

Table 7-6. HTRQ and HRRQ Pin Operation In Double Request Mode (ICR[2]=HDRQ=1)

ICR[1]=TREQ	ICR[0]=RREQ	HTRQ Pin	HRRQ Pin
0	0	No interrupts	No interrupts
0	1	No interrupts	RXDF request enabled
1	0	TXDE Request enabled	No interrupts
1	1	TXDE Request enabled	RXDF request enabled

7.4.5 Endian Modes

The Host Little Endian bit in the host-side Interface Control Register (ICR[5]=HLEND) allows the host to access the HI08 data registers in Big Endian or Little Endian mode. In Little Endian mode (HLEND=1), a host transfer occurs as shown in **Figure 7-4**.

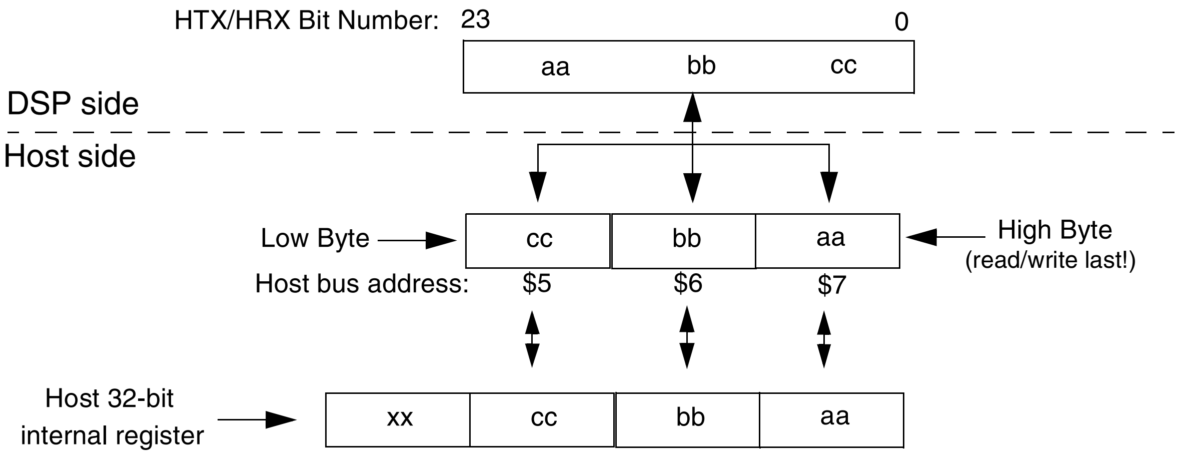


Figure 7-4. HI08 Read and Write Operations in Little Endian Mode

The host can transfer one byte at a time, so a 24-bit datum would be transferred using three store (or load) byte operations, ensuring that the data byte at host bus address $7 is written last since this causes the transfer of the data to the DSP-side HRX. However, the host bus controller may be sophisticated enough that the host can transfer all bytes in a single operation (instruction). For example, in the PowerPC MPC860 processor, the General-Purpose Controller Module (GPCM) in the memory controller can be programmed so that the host can execute a single read (load word, LDW) or write (store word, STW) instruction to the HI08 port and cause four byte transfers to occur on the host bus. The 32-bit datum transfer shown in **Figure 7-4** has byte data xx written to HI08 address $4, byte aa to address $5, byte bb to address $6 and byte cc to address $7 (this assumes the 24-bit datum is contained in the lower 24 bits of the host's 32-bit data register as shown). A similar operation occurs when the HI08 is initialized in Big Endian mode by clearing the Host Little Endian bit (ICR[5]=HLEND). Big Endian mode is depicted in **Figure 7-5**.

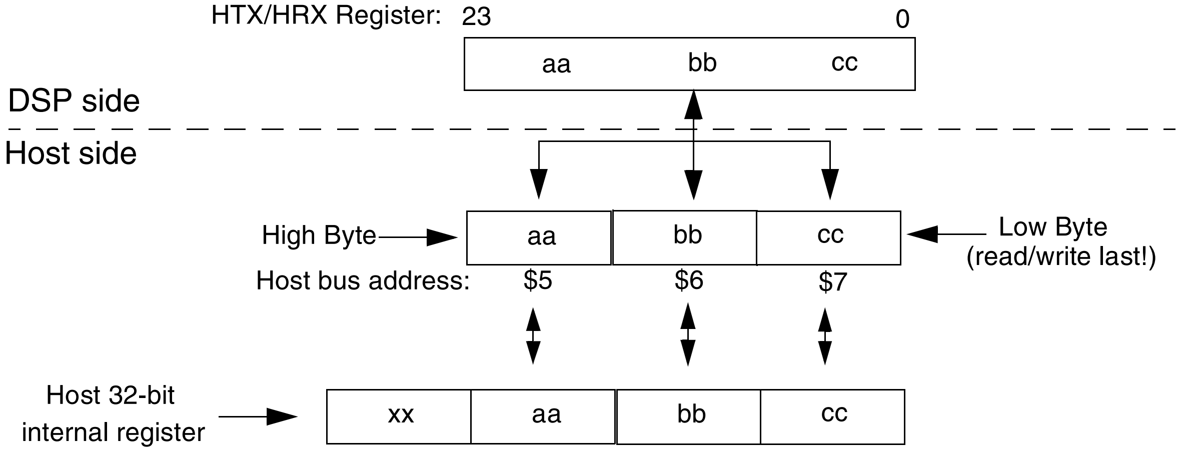


Figure 7-5. HI08 Read and Write Operations in Big Endian Mode

7.5 Boot-up Using the HI08 Host Port

The DSP56300 core has eight bootstrap operating modes to start up after reset. As the processor exits the Reset state the value at the external mode pins MODA/$\overline{\text{IRQA}}$, MODB/$\overline{\text{IRQB}}$, MODC/$\overline{\text{IRQC}}$ and MODD/$\overline{\text{IRQD}}$ are loaded into the Chip Operating Mode bits (MA, MB, MC and MD) of the Operating Mode Register (OMR). These bits determine the bootstrap operating mode. Modes C, D, E and F use the HI08 host port to bootstrap the application code to the DSP. **Table 7-7** describes these modes.

Table 7-7. HI08 Boot Modes

Mode	MODD	MODC	MODB	MODA	HI08 Bootstrap Description
C	1	1	0	0	ISA/DSP5630x mode
D	1	1	0	1	HC11 non-multiplexed bus mode
E	1	1	1	0	8051 multiplexed bus mode
F	1	1	1	1	MC68302 bus mode

The bootstrap program is factory-programmed into an internal 192-word by 24-bit bootstrap ROM at locations $FF0000–$FF00BF of P memory. This program can load program RAM segment from the HI08 host port. When any of the modes in the preceding table are used, the core begins executing the bootstrap program and configures the HI08 based on the OMR mode bits. The bootstrap program then expects the following data sequence when the user program is downloaded from the HI08:

1. Three bytes (least significant byte first) indicating the number of 24-bit program words to be loaded.
2. Three bytes (least significant byte first) indicating the 24-bit starting address in P-memory to load the user's program.
3. The user program (three bytes, least significant byte first, for each program word).

When the bootstrap program finishes loading the specified number of words, it jumps to the specified starting address and executes the loaded program.

7.6 DSP Core Programming Model

The DSP56300 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 DSP56321 core to communicate with the HI08 registers using standard instructions and addressing modes. In addition, the MOVEP instruction allows direct data transfers between DSP56321 internal memory and the HI08 registers or *vice versa*.

There are two types of host processor registers, data and control, with eight registers in all. The DSP core can access all eight registers, but the external host cannot. The following data registers are 24-bit registers used for high-speed data transfers by the DSP core.

- Host data receive register (HRX), on **page 7-19**
- Host data transmit register (HTX), on **page 7-19**

The DSP-side control registers are 16-bit registers that control HI08 functionality:

- Host control register (HCR), on **page 7-12**
- Host status register (HSR), on **page 7-13**
- Host GPIO data direction register (HDDR), on **page 7-14**
- Host GPIO data register (HDR), on **page 7-15**
- Host base address register (HBAR), on **page 7-15**
- Host port control register (HPCR), on **page 7-16**

Both hardware and software resets disable the HI08. After reset, the HI08 signals are configured as GPIO and disconnected from the DSP56300 core (that is, left floating).

7.6.1 Host Control Register (HCR)

This read/write register controls the HI08 interrupt operation. Initialization values for HCR bits are presented in **Section 7.6.9**, *DSP-Side Registers After Reset*, on page 7-20.

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

▨—Reserved bit; read as 0; write to 0 for future compatibility.

Table 7-8. Host Control Register (HCR) Bit Definitions

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="2">Description</th></tr>
<tr><td>15–5</td><td></td><td>0</td><td colspan="2">Reserved. Write to 0 for future compatibility.</td></tr>
<tr><td>4–3</td><td>HF[3 –2]</td><td>0</td><td colspan="2"><b>Host Flags 2, 3</b>
General-purpose flags for DSP-to-host communication. The DSP core can set or clear HF[3–2]. 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. The bit value is indeterminate after an individual reset.</td></tr>
<tr><td rowspan="5">2</td><td rowspan="5">HCIE</td><td rowspan="5">0</td><td colspan="2"><b>Host Command Interrupt Enable</b>
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 (for example, HRDF is set, HCP is set, HRIE is set, and HCIE is set), the HI08 generates interrupt requests according to priorities shown here. The bit value is indeterminate after an individual reset.</td></tr>
<tr><td><b>Priority</b></td><td><b>Interrupt Source</b></td></tr>
<tr><td>Highest</td><td>Host Command (HCP = 1)</td></tr>
<tr><td></td><td>Transmit Data (HTDE = 1)</td></tr>
<tr><td>Lowest</td><td>Receive Data (HRDF = 1)</td></tr>
<tr><td>1</td><td>HTIE</td><td>0</td><td colspan="2"><b>Host Transmit Interrupt Enable</b>
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. The bit value is indeterminate after an individual reset.</td></tr>
<tr><td>0</td><td>HRIE</td><td>0</td><td colspan="2"><b>Host Receive Interrupt Enable</b>
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. The bit value is indeterminate after an individual reset.</td></tr>
</table>

7.6.2 Host Status Register (HSR)

The HSR is a 16-bit read-only status register by which the DSP reads the HI08 status and flags. The host processor cannot access it directly. The initialization values for the HSR bits are discussed in **Section 7.6.9**, *DSP-Side Registers After Reset*, on page 7-20.

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; write to 0 for future compatibility.

Figure 0-1. Host Status Register (HSR) (X:$FFFFC3)

Table 7-9. Host Status Register (HSR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
15–5	▨	0	Reserved. Write to 0 for future compatibility.
4–3	HF[1–0]	0	**Host Flags 0, 1** General-purpose flags for host-to-DSP communication. 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.
2	HCP	0	**Host Command Pending** Reflects the status of the CVR[HC] bit. When set, it indicates that a host command interrupt is pending. HI08 hardware clears HC and HCP when the DSP core services the interrupt request. If the host clears HC, HCP is also cleared.
1	HTDE	0	**Host Transmit Data Empty** 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. The host processor can also set HTDE using the initialize function. HTDE is cleared when the DSP core writes to HTX.
0	HRDF	0	**Host Receive Data Full** 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. The host processor can also clear HRDF using the initialize function.

7.6.3 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 on the HI08 GPIO configuration options, see **Section 7.2**, *Host Port Signals*, on page 7-3. 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. Hardware and software reset clear the HDDR bits.

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

7.6.4 Host Data Register (HDR)

The HDR register (X:$FFFFC9) holds the data value of the corresponding bits of the HI08 signals configured as GPIO signals. The functionality of D*xx* depends on the corresponding HDDR bit (that is, DR*xx*).The host processor can not access the Host Data Register (HDR).

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

Table 7-10. HDR and HDDR Functionality

HDDR	HDR	
DRxx	D*xx*	
	GPIO Signal[1]	Non-GPIO Signal[1]
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.

[1] Defined by the selected configuration.

7.6.5 Host Base Address Register (HBAR)

In multiplexed bus modes, HBAR selects the base address where the host-side registers are mapped into the host 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 7-6** shows how the chip-select logic uses HBAR.

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

▨ —Reserved bit, read as 0, write to 0 for future compatibility.

Table 7-11. Host Base Address Register (HBAR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
15–8	▨	0	Reserved. Write to 0 for future compatibility.
7–0	BA[10–3]	$80	**Base Address** Reflect the base address where the host-side registers are mapped into the bus address space.

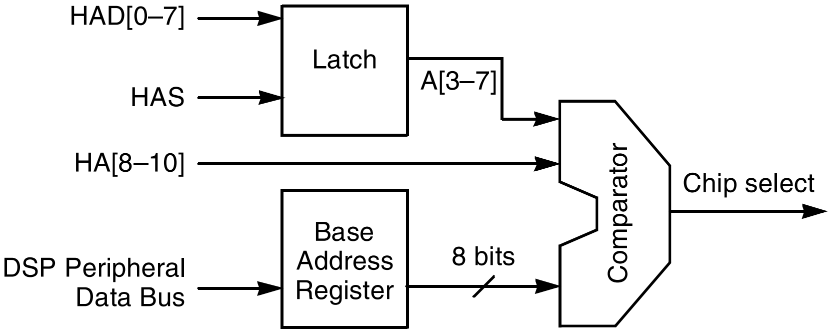


Figure 7-6. Self Chip-Select Logic

7.6.6 Host Port Control Register (HPCR)

The HPCR is a read/write control register that controls the HI08 operating mode. HPCR bit initialization values are discussed in **Section 7.6.9**, *DSP-Side Registers After Reset*, on page 7-20. Hardware and software reset clear the HPCR bits.

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; write to 0 for future compatibility.

To assure proper operation of the DSP56321, the HPCR bits HAP, HRP, HCSP, HDDS, HMUX, HASP, HDSP, HROD, HAEN, and HREN should be changed only if HEN is cleared. Similarly, 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 at the time HEN is set.

Table 7-12. Host Port Control Register (HPCR) Bit Definitions

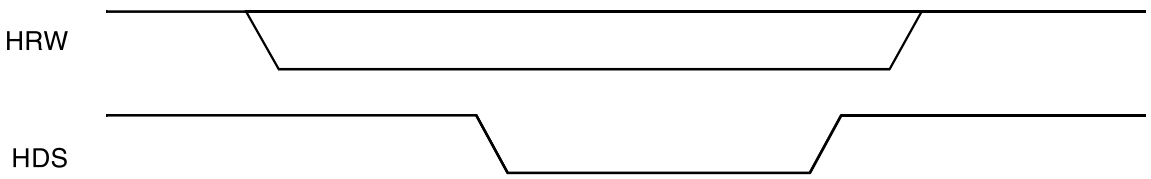
Bit Number	Bit Name	Reset Value	Description
15	HAP	0	**Host Acknowledge Polarity** If HAP is cleared, the host acknowledge (HACK) signal is configured as an active low input. The HI08 drives the contents of the IVR onto the host bus when the HACK signal is low. If the HAP bit is set, the HACK signal is configured as an active high input. The HI08 outputs the contents of the IVR when the HACK signal is high.
14	HRP	0	**Host Request Polarity** Controls the polarity of the host request signals. In single host request mode (that is, when HDRQ is cleared in the ICR), if HRP is cleared and host requests are enabled (that is, 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 (that is, when HDRQ is set in the ICR), if HRP is cleared and host requests are enabled (that is, 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.

Table 7-12. Host Port Control Register (HPCR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
13	HCSP	0	**Host Chip Select Polarity** 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.
12	HDDS	0	**Host Dual Data Strobe** 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 7-7 on page -19 and Figure 7-8 on page -19 for details on dual and single strobe modes.
11	HMUX	0	**Host Multiplexed Bus** If HMUX is set, the HI08 operates in multiplex mode, latching the lower portion of a multiplexed address/data bus. In this mode the internal address line values of the host registers are taken from the internal latch. If HMUX is cleared, it indicates that the HI08 is connected to a non-multiplexed type of bus. The values of the address lines are then taken from the HI08-dedicated address signals.
10	HASP	0	**Host Address Strobe Polarity** 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.
9	HDSP	0	**Host Data Strobe Polarity** 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.
8	HROD	0	**Host Request Open Drain** Controls the output drive of the host request signals. In the single host request mode (that is, when HDRQ is cleared in ICR), if HROD is cleared and host requests are enabled (that is, 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 (that is, when HDRQ is set in the ICR), if HROD is cleared and host requests are enabled (that is, 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.
7		0	Reserved. Write to 0 for future compatibility.
6	HEN	0	**Host Enable** 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.

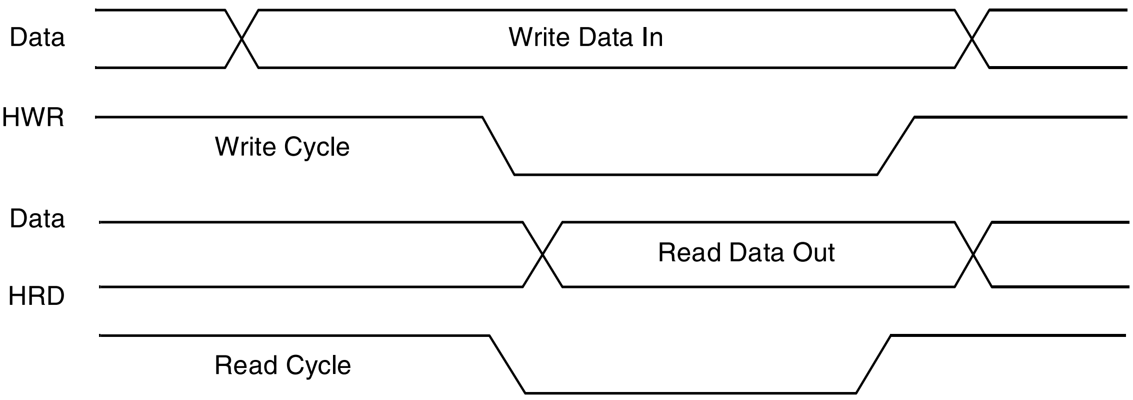
Table 7-12. Host Port Control Register (HPCR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
5	HAEN	0	**Host Acknowledge Enable** 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.
4	HREN	0	**Host Request Enable** Controls the host request signals. If HREN is set and the HI08 is in the single host request mode (that is, 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 (that is, if HDRQ is set in the ICR), HREQ/HTRQ is configured as the host transmit request (HTRQ) output and HACK/HRRQ as the host receive request (HRRQ) output. If HREN is cleared, HREQ/HTRQ and HACK/HRRQ are configured as GPIO signals according to the value of the HDDR and HDR.
3	HCSEN	0	**Host Chip Select Enable** If the HCSEN bit is set, HCS/HA10 is a host chip select (HCS) in the non-multiplexed bus mode (that is, when HMUX is cleared) and host address line 10 (HA10) in the multiplexed bus mode (that is, when HMUX is set). If this bit is cleared, HCS/HA10 is configured as a GPIO signal according to the value of the HDDR and HDR.
2	HA9EN	0	**Host Address Line 9 Enable** If HA9EN is set and the HI08 is in multiplexed bus mode, then HA9/HA2 is host address line 9 (HA9). If this bit is cleared and the HI08 is in multiplexed bus mode, then HA9/HA2 is configured as a GPIO signal according to the value of the HDDR and HDR. NOTE: HA9EN is ignored when the HI08 is not in the multiplexed bus mode (that is, when HMUX is cleared).
1	HA8EN	0	**Host Address Line 8 Enable** If HA8EN is set and the HI08 is in multiplexed bus mode, then HA8/A1 is host address line 8 (HA8). If this bit is cleared and the HI08 is in multiplexed bus mode, then HA8/HA1 is 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 (that is, when HMUX is cleared).
0	HGEN	0	**Host GPIO Port Enable** Enables/disables signals configured as GPIO. 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.



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

Figure 7-7. Single-Strobe Mode



In dual-strobe mode, separate HRD and HWR signals specify the access as a read or write access, respectively.

Figure 7-8. Dual-Strobe Mode

7.6.7 Host Transmit (HTX) Register

The HTX register is used in DSP-to-host data transfers. The DSP56321 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 (HSR[HTDE]) on the DSP side. The contents of the HTX register are transferred as 24-bit data to the Receive Data Registers (RXH:RXM:RXL) when both HSR[HTDE] and receive data full (ISR[RXDF]) on the host-side bits are cleared. This transfer operation sets the ISR[RXDF] and HSR[HTDE] bits. The DSP56321 can set the HCR[HTIE] bit to cause a host transmit data interrupt when HSR[HTDE] is set. To prevent the previous data from being overwritten, the DSP56303 should never write to the HTX when HSR[HTDE] is cleared.

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 the status bits within the next two cycles, the bit does not reflect its current status. For details, see **Section 6.4.1**, *Polling*, on page 6-2.

7.6.8 Host Receive (HRX) Register

The HRX register is used in host-to-DSP data transfers. The DSP56321 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 (ISR[TXDE]) on the host side and host receive data full (HSR[HRDF]) on the DSP side are cleared. The transfer operation sets both ISR[TXDE] and HSR[HRDF]. When the HSR[HRDF] is set, the HRX register contains valid data. The DSP56321 can set the HCR[HRIE] to cause a host receive data interrupt when HSR[HRDF] is set. When the DSP56321 reads the HRX register, the HSR[HRDF] bit is cleared.

Note: The DSP56321 should never try to read the HRX register if the HSR[HRDF] bit is already cleared.

7.6.9 DSP-Side Registers After Reset

Table 7-13 shows the results of the four reset types on the bits in each of the HI08 registers accessible to the DSP56321. 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) occurs when HPCR[HEN] is cleared. The stop reset (ST) occurs when the STOP instruction executes.

Table 7-13. DSP-Side Registers After Reset

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	IR Reset	ST Reset
HCR	All bits	0	0	—	—
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 long dash (—) denotes that the bit value is not affected by the specified reset.

7.7 Host Programmer 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 paths 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 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 7-14**.) The eight HI08 registers include the following:

- A control register (ICR), on **page 7-22**
- A status register (ISR), on **page 7-25**
- Three data registers (RXH/TXH, RXM/TXM, and RXL/TXL), on **page 7-27**
- Two vector registers (CVR and IVR), on **page 7-24** and **page 7-27**

To transfer data between itself and the HI08, the host processor bus performs the following steps:

1. Asserts the HI08 address and strobes to select the register to be read or written. (Chip select in non-multiplexed mode, the address strobe in multiplexed mode.)
2. Selects the direction of the data transfer. If it is writing, the host processor places the data on the bus. Otherwise, the HI08 places the data on the bus.
3. Strobes the data transfer.

Host processors can use standard host processor instructions (for example, 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 DSP56321 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 DSP56321 as a fast device, and data can be transferred between the host processor and the DSP56321 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 DSP56321. 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 DSP56321. For example, the DSP56321 host interrupts allow the host processor to read or write DSP registers (X, Y, or program memory locations), force interrupt handlers (for example, ESSI, 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 7-14. Host-Side Register Map

Host Address	Big Endian HLEND = 0	Little Endian HLEND = 1	Register Name
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 Data
6	RXM/TXM	RXM/TXM	
7	RXL/TXL	RXH/TXH	

7.7.1 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. Hardware and software reset clear the ICR bits.

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

▨ —Reserved bit; read as 0; write to 0 for future compatibility.

Table 7-15. Interface Control Register (ICR) Bit Definitions

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="4">Description</th></tr>
<tr><td rowspan="6">7</td><td rowspan="6">INIT</td><td rowspan="6">0</td><td colspan="4"><b>Initialize</b>
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. Whether it is necessary to use the INIT bit to initialize the HI08 hardware depends 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, configures the HI08 into the desired data transfer mode. 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.</td></tr>
<tr><td><b>TREQ</b></td><td><b>RREQ</b></td><td><b>After INIT Execution</b></td><td><b>Transfer Direction Initialized</b></td></tr>
<tr><td>0</td><td>0</td><td>INIT = 0</td><td>None</td></tr>
<tr><td>0</td><td>1</td><td>INIT = 0; RXDF = 0; HTDE = 1</td><td>DSP to host</td></tr>
<tr><td>1</td><td>0</td><td>INIT = 0; TXDE = 1; HRDF = 0</td><td>Host to DSP</td></tr>
<tr><td>1</td><td>1</td><td>INIT = 0; RXDF = 0; HTDE = 1; TXDE = 1; HRDF = 0</td><td>Host to/from DSP</td></tr>
<tr><td>6</td><td></td><td>0</td><td colspan="4">Reserved. Write to 0 for future compatibility.</td></tr>
<tr><td>5</td><td>HLEND</td><td>0</td><td colspan="4"><b>Host Little Endian</b>
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.</td></tr>
<tr><td>4</td><td>HF1</td><td>0</td><td colspan="4"><b>Host Flag 1</b>
A general-purpose flag for host-to-DSP communication. The host processor can set or clear HF1, and the DSP56321 can not change it. HF1 is reflected in the HSR on the DSP side of the HI08.</td></tr>
<tr><td>3</td><td>HF0</td><td>0</td><td colspan="4"><b>Host Flag 0</b>
A general-purpose flag for host-to-DSP communication. The host processor can set or clear HF0, and the DSP56321 cannot change it. HF0 is reflected in the HSR on the DSP side of the HI08.</td></tr>
<tr><td>2</td><td>HDRQ</td><td>0</td><td colspan="4"><b>Double Host Request</b>
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.</td></tr>
</table>

Table 7-15. Interface Control Register (ICR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="4">Description</th></tr>
<tr><td rowspan="13">1</td><td rowspan="13">TREQ</td><td rowspan="13">0</td><td colspan="4"><b>Transmit Request Enable</b>
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.</td></tr>
<tr><td colspan="4"><b>TREQ and RREQ modes (HDRQ = 0)</b></td></tr>
<tr><td><b>TREQ</b></td><td><b>RREQ</b></td><td colspan="2"><b>HREQ Signal</b></td></tr>
<tr><td>0</td><td>0</td><td colspan="2">No interrupts (polling)</td></tr>
<tr><td>0</td><td>1</td><td colspan="2">RXDF request (interrupt)</td></tr>
<tr><td>1</td><td>0</td><td colspan="2">TXDE request (interrupt)</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">RXDF and TXDE request (interrupts)</td></tr>
<tr><td colspan="4"><b>TREQ and RREQ modes (HDRQ = 1)</b></td></tr>
<tr><td><b>TREQ</b></td><td><b>RREQ</b></td><td><b>HTRQ Signal</b></td><td><b>HRRQ Signal</b></td></tr>
<tr><td>0</td><td>0</td><td>No interrupts (polling)</td><td>No interrupts (polling)</td></tr>
<tr><td>0</td><td>1</td><td>No interrupts (polling)</td><td>RXDF request (interrupt)</td></tr>
<tr><td>1</td><td>0</td><td>TXDE request (interrupt)</td><td>No interrupts (polling)</td></tr>
<tr><td>1</td><td>1</td><td>TXDE request (interrupt)</td><td>RXDF request (interrupt)</td></tr>
<tr><td>0</td><td>RREQ</td><td>0</td><td colspan="4"><b>Receive Request Enable</b>
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.</td></tr>
</table>

7.7.2 Command Vector Register (CVR)

The host processor uses the CVR, an 8-bit read/write register, to cause the DSP56321 to execute an interrupt. The host command feature is independent of any of the data transfer mechanisms in the HI08. It causes execution of any of the 128 possible interrupt routines in the DSP core. Hardware, software, individual, and stop resets clear the CVR bits.

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

Table 7-16. Command Vector Register (CVR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
7	HC	0	**Host Command** 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 DSP56321. When the DSP56321 acknowledges the host command interrupt, HI08 hardware clears the HC bit. The host processor can read the state of HC to determine when the host command has been accepted. After setting HC, the host must not write to the CVR again until the HI08 hardware clears 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.
6–0	HV[6–0]	$32	**Host Vector** Select the host command interrupt address for use by the host command interrupt logic. When the DSP interrupt control logic recognizes the host command interrupt, 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 interrupt handler (ESSI, SCI, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, and so forth) and can use any reserved or otherwise unused addresses (if have been pre-programmed in the DSP). HV is set to $32 (vector location $064) by hardware, software, individual, and stop resets.

7.7.3 Interface Status Register (ISR)

The host processor uses the ISR, an 8-bit read-only status register, to interrogate the HI08 status and flags. The DSP core cannot address the ISR.

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

▨ —Reserved bit; read as 0; write to 0 for future compatibility.

Table 7-17. Interface Status Register (ISR) Bit Definitions

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="3">Description</th></tr>
<tr><td rowspan="6">7</td><td rowspan="6">HREQ</td><td rowspan="6">0 (Hardware and Software reset)
1 (Individual reset and TREQ is set)
1 (Stop reset and TREQ is set)</td><td colspan="3"><b>Host Request</b>
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, HREQ indicates the status of the external host request output signal (HREQ). The HREQ bit is set from either or both of two conditions— 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.</td></tr>
<tr><th>HDRQ</th><th>HREQ</th><th>Effect</th></tr>
<tr><td>0</td><td>0</td><td>HREQ is cleared; no host processor interrupts are requested.</td></tr>
<tr><td>0</td><td>1</td><td>HREQ is set; an interrupt is requested.</td></tr>
<tr><td>1</td><td>0</td><td>HTRQ and HRRQ are cleared, no host processor interrupts are requested.</td></tr>
<tr><td>1</td><td>1</td><td>HTRQ or HRRQ are set; an interrupt is requested.</td></tr>
<tr><td>6–5</td><td></td><td>0</td><td colspan="3">Reserved. Write to 0 for future compatibility.</td></tr>
<tr><td>4</td><td>HF3</td><td>0</td><td colspan="3"><b>Host Flag 3</b>
Indicates the state of HF3 in the HCR on the DSP side. HF3 can be changed only by the DSP56321. Hardware and software reset clear HF3.</td></tr>
<tr><td>3</td><td>HF2</td><td>0</td><td colspan="3"><b>Host Flag 2</b>
Indicates the state of HF2 in the HCR on the DSP side. HF2 can be changed only by the DSP56321. Hardware and software reset clear HF2.</td></tr>
<tr><td>2</td><td>TRDY</td><td>1</td><td colspan="3"><b>Transmitter Ready</b>
Indicates that TXH:TXM:TXL and the HRX registers are empty. 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 DSP56321 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 DSP56321. Hardware, software, individual, and stop resets all set TRDY.</td></tr>
<tr><td colspan="3"></td><td colspan="3"><b>CAUTION:</b>
<b>TRDY = TXDE and $\overline{\text{HRDF}}$</b></td></tr>
<tr><td>1</td><td>TXDE</td><td>1</td><td colspan="3"><b>Transmit Data Register Empty</b>
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 can assert the external HTRQ signal if the TREQ bit is set. Regardless of whether the TXDE interrupt is enabled, TXDE indicates whether the TX registers are full and data can be latched in (so that polling techniques may be used by the host processor). Hardware, software, individual, and stop resets all set TXDE.</td></tr>
</table>

Table 7-17. Interface Status Register (ISR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
0	RXDF	0	**Receive Data Register Full** Indicates that the receive byte registers (RXH:RXM:RXL) contain data from the DSP56321 to be read by the host processor. RXDF is set when the HTX is transferred to the receive byte registers. RXDF is cleared when the host processor reads the receive data register (RXL or RXH according to HLEND bit). 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).

7.7.4 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

7.7.5 Receive Data Registers (RXH:RXM:RXL)

The host processor views the receive byte registers as three 8-bit read-only registers: 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 ICR[HLEND]. If ICR[HLEND] is set, the RXH is located at address $7, RXM at $6, and RXL at $5. If ICR[HLEND] 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 ISR Receive Data Register Full (RXDF) bit is set. The host processor can program the RREQ bit to assert the external HREQ signal when ISR[RXDF] is set. This indicates that the HI08 has a full word (either 8, 16, or 24 bits) for the host processor. The host processor can program the RREQ bit to assert the external HREQ signal when ISR[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 ISR[RXDF] bit is cleared.

Note: The external host should never read the RXH:RXM:RXL registers if the ISR[RXDF] bit is cleared.

7.7.6 Transmit Data 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 ICR[HLEND] 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. The external host should never write to the TXH:TXM:TXL registers if the ISR[TXDE] bit is cleared.

Data can be written into the transmit byte registers when the ISR transmit data register empty (TXDE) bit is set. The host processor can program the ICR[TREQ] bit to assert the external HREQ/HTRQ signal when ISR[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 ISR[TXDE] bit. The contents of the transmit byte registers are transferred as 24-bit data to the HRX register when both ISR[TXDE] and HSR[HRDF] are cleared. This transfer operation sets HSR[TXDE] and HSR[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. For details, see **Section 6.4.1**, *Polling*, on page 6-2.

7.7.7 Host-Side Registers After Reset

Table 7-18 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 HPCR[HEN] bit. To cause a stop reset, execute the STOP instruction.

Table 7-18. Host-Side Registers After Reset

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	Individual Reset	STOP
ICR	All bits	0	0	—	—
CVR	HC	0	0	0	0
	HV[0–6]	$32	$32	—	—

Table 7-18. Host-Side Registers After Reset (Continued)

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	Individual Reset	STOP
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
Note:	A long dash (—) denotes that the bit value is not affected by the specified reset.				

7.8 Programming Model Quick Reference

Table 7-19 summarizes the HI08 programming model.

Table 7-19. HI08 Programming Model, DSP Side

Register	Bit					Reset Type		
	Bit No.	Bit	Name	Value	Function	HW/ SW	Indivi-dual	STOP
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	—	—

Table 7-19. HI08 Programming Model, DSP Side (Continued)

Register	Bit					Reset Type		
	Bit No.	**Bit**	**Name**	**Value**	**Function**	**HW/ SW**	**Indivi-dual**	**STOP**
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	0	—	—
	2	HA9EN	Host Address Line 9 Enable	0 1	HA9/A2 = GPIO HA9/A2 = HA9	0	—	—
	3	HCSEN	Host Chip Select Enable	0 1	HCS/A10 = GPIO HCS/A10 = HCS	0	—	—
	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	0	—	—
	6	HEN	Host Enable	0 1	Host Port = GPIO Host Port Active	0	—	—
	7	—	Reserved	0	Reserved	0	—	—
HPCR	8	HROD	Host Request Open Drain	0 1	HREQ/HTRQ/HRRQ = driven HREQ/HTRQ/HRRQ = open drain	0		
	9	HDSP	Host Data Strobe Polarity	0 1	HDS/HRD/HWR active low HDS/HRD/HWR active high	0	—	—
	10	HASP	Host Address Strobe Polarity	0 1	HAS active low HAS active high	0	—	—
	11	HMUX	Host Multiplexed Bus	0 1	Separate address and data lines Multiplexed address/data	0	—	—
	12	HDDS	Host Dual Data Strobe	0 1	Single Data Strobe (HDS) Double Data Strobe (HWR, HRD)	0	—	—
	13	HCSP	Host Chip Select Polarity	0 1	HCS active low HCS active high	0	—	—
	14	HRP	Host Request Polarity	0 1	HREQ/HTRQ/HRRQ active low HREQ/HTRQ/HRRQ active high	0	—	—
	15	HAP	Host Acknowledge Polarity	0 1	HACK active low HACK active high	0	—	—

Table 7-19. HI08 Programming Model, DSP Side (Continued)

Register	Bit					Reset Type		
	Bit No.	Bit	Name	Value	Function	HW/SW	Individual	STOP
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	BA[10–3]	Host Base Address Register			$80		
HRX	23–0		DSP Receive Data Register			empty		
HTX	23–0		DSP Transmit Data Register			empty		
HDR	16–0	D[16–0]	GPIO signal Data			$0000	—	—
HDRR	16–0	DR[16–0]	GPIO signal Direction	0 1	Input Output	$0000	—	—

Table 7-20. HI08 Programming Model: Host Side

Register	Bit					Reset Type		
	Bit No.	Bit	Name	Value	Function	HW/SW	Individual	STOP
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	0	—	—

Table 7-20. HI08 Programming Model: Host Side (Continued)

Register	Bit					Reset Type		
	Bit No.	Bit	Name	Value	Function	HW/SW	Indivi-dual	STOP
ISR	0	RXDF	Receive Data Register Full	0 1	Host Receive Register is empty Host Receive Register is full	0	0	0
	1	TXDE	Transmit Data Register Empty	1 0	Host Transmit Register is empty Host Transmit Register is full	1	1	1
	2	TRDY	Transmitter Ready	1 0	transmit FIFO (6 deep) is empty transmit FIFO is not empty	1	1	1
	3	HF2	Host Flag 2			0	—	—
	4	HF3	Host Flag 3			0	—	—
	7	HREQ	Host Request	0 1	$\overline{\text{HREQ}}$ signal is deasserted $\overline{\text{HREQ}}$ signal is asserted (if enabled)	0	0	0
CVR	6–0	HV[6–0]	Host Command Vector			$32	—	—
CVR	7	HC	Host Command	0 1	no host command pending host command pending	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	IV[7–0]	Interrupt Register		68000 family vector register	$0F	—	—

Enhanced Synchronous Serial Interface 8

The enhanced synchronous serial interface (ESSI) provides a full-duplex serial port for serial communication with a variety of serial devices, including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals. The ESSI consists of independent transmitter and receiver sections and a common ESSI clock generator. There are two independent and identical ESSIs in the DSP56321: ESSI0 and ESSI1. For simplicity, a single generic ESSI is described here. The ESSI block diagram is shown in **Figure 8-1**. This interface is synchronous because all serial transfers are synchronized to one clock.

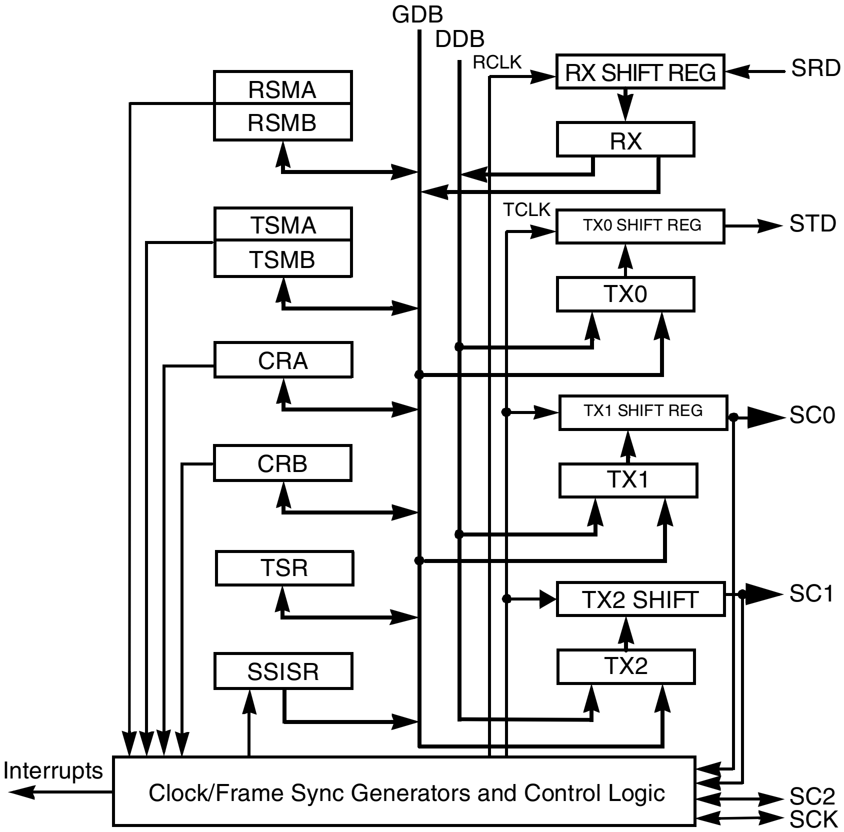


Figure 8-1. ESSI Block Diagram

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. Pin notations for the generic ESSI refer to the analogous pin of ESSI0 (PCx) and ESSI1 (PDx).

Additional synchronization signals delineate the word frames. The Normal mode of operation transfers data at a periodic rate, one word per period. The Network mode is similar in that it is also 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 for nonperiodic transfers of data. This mode, which offers a subset of the Serial Peripheral Interface (SPI) protocol, can transfer data serially at high speed when the data become available. 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).

8.1 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.

8.1.1 Serial Transmit Data Signal (STD)

The STD signal transmits data from the serial transmit shift register. STD is an output when data is 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 in use.

8.1.2 Serial Receive Data Signal (SRD)

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

8.1.3 Serial Clock (SCK)

SCK is a bidirectional signal providing the serial bit rate clock for the ESSI interface. The signal is a clock input or output used by all the enabled transmitters and receivers in Synchronous modes or by all the enabled transmitters in Asynchronous modes. See **Table 8-1** for details. SCK can be programmed as a GPIO signal (P3) when not used as the ESSI clock.

Table 8-1. 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

Note: 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. The internally sourced ESSI clock frequency must not exceed $F_{core}/4$.

8.1.4 Serial Control Signal (SC0)

ESSI0: SC00; ESSI1: SC10

To determine the function of the SC0 signal, select either Synchronous or Asynchronous mode, according to **Table 8-2**. In Asynchronous mode, this signal is used for the receive clock I/O. In Synchronous mode, this signal is 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.

8.1.5 Serial Control Signal (SC1)

ESSI0:SC01; ESSI1: SCI11

To determine the function of SC1, select either Synchronous or Asynchronous mode, according to **Table 8-2**. In Asynchronous mode (as for a single codec with asynchronous transmit and receive), SC1 is the receiver frame sync I/O. In Synchronous mode, SC1 is the transmitter data out signal of transmit shift register TX2, for the transmitter 0 drive-enabled signal, or for serial flag I/O. As serial flag I/O, SC1 operates like SC0. SC0 and SC1are independent flags but can be used together for multiple serial device selection; they 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 or receive frame sync signal, the Serial Control Direction 1 CRB[SCD1] bit determines its direction.

Table 8-2. Mode and Signal Definitions

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
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

Table 8-2. Mode and Signal Definitions (Continued)

Control Bits					ESSI Signals					
SYN	**TE0**	**TE1**	**TE2**	**RE**	**SC0**	**SC1**	**SC2**	**SCK**	**STD**	**SRD**
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

When configured as an output, SC1 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 SC1 is configured as a transmit data signal, it is always an output signal, regardless of the SCD1 bit value. As an output, it is fully synchronized with the other ESSI transmit data signals (STD and SC0). SC1 can be programmed as a GPIO signal (P1) when the ESSI SC1 function is not in use.

8.1.6 Serial Control Signal (SC2)

ESSI0:SC02; ESSI1:SC12

SC2 is a frame sync I/O signal 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.

8.2 Operation

This section discusses ESSI basics: reset state, initialization, and exceptions.

8.2.1 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.

8.2.2 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 ensure 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 is enabled (Step 3), operation starts as follows:

1. For internally generated clock and frame sync, these signals start activity immediately after the ESSI is enabled.
2. The ESSI receives data after a frame sync signal (either internally or externally generated) only when the receive enable (RE) bit is set.
3. Data is transmitted after a frame sync signal (either internally or externally generated) only when the transmitter enable (TE[2–0]) bit is set.

8.2.3 Exceptions

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

- 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 the Receive Data Register (RX).
- 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.
- 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.

  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).
- 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 SSISR[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.
- 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 frame settings and the new frame is serviced with the new frame settings 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).

■ 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 the 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 (PC[5–0])`
   - **f.** Unmask interrupts at the global level. `SR (I1–0)`

The example material to the right of the steps shows register settings for configuring an ESSI0 transmit interrupt using transmitter 0. 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 2c** may cause an immediate transmit without generating an interrupt, perform the transmit data preload in **step 2b** before **step 2c** to ensure that valid data is sent in the first transmission. 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 needs to load the transmit register (N – 1) times. **Steps 2c** and **2d** can be performed in **step**

2a as a single instruction. If an interrupt trigger event occurs before all interrupt trigger configuration steps are performed, the event is ignored and not queued. If interrupts derived from the core or other peripherals need to be enabled at the same time as ESSI interrupts, **step 2f** should be performed last.

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

The ESSI has three basic operating modes and several data and operation formats. These modes are programmed via 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.

8.3.1 Normal/Network/On-Demand Mode Selection

To select either Normal mode or Network mode, clear or set CRB[MOD]. 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 can be selected. During each frame, 0 to 32 data words are 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. Set the CRB[MOD] for Network mode, and 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 for interfacing with codecs requiring a continuous clock.

Note: When the ESSI transmits data in On-Demand mode (that is, MOD = 1 in the CRB and DC[4–0]=$00000 in the CRA) with WL[2–0] = 100, the transmission does not work properly. To ensure correct operation, do not use On-Demand mode with the WL[2–0] = 100 32-bit word length mode.

8.3.2 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 and separate clock and sync signals in Asynchronous mode. The CRB[SYN] bit 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). 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.

8.3.3 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 the CRB FSL[1–0], FSR, and FSP bits.

8.3.4 Frame Sync Signal Format

CRB[FSL1] controls the frame sync signal format:

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

8.3.5 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 (for example, codec) and transmitted to a different type of device. CRB[FSL0] controls whether RX and TX have the same frame sync length.

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

CRB[FSL0] is ignored when CRB[SYN] is set.

8.3.6 Word Length Frame Sync and Data Word Timing

The CRB[FSR] bit controls the relative timing of the word length frame sync relative to the data word timing.

- When CRB[FSR] is cleared, the word length frame sync is generated (or expected) with the first bit of the data word.
- When CRB[FSR] is set, the word length frame sync is generated (or expected) with the last bit of the previous word.

CRB[FSR] is ignored when a bit length frame sync is selected.

8.3.7 Frame Sync Polarity

The CRB[FSP] bit controls the polarity of the frame sync.

- When CRB[FSP] 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 CRB[FSP] 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 CRB[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 CRB[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.

8.3.8 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 CRB[SHFD].

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

8.3.9 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 Synchronous mode only. Each flag can be separately programmed. The SC0 flag 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 is 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 they are 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 OF[1–0] is latched when the contents of TX transfer to the transmit shift register. The value on SC[1–0] is stable from the time the first bit of the transmit data word transmits until the first bit of the next transmit data word transmits. Software can directly set the OF[1–0] values, allowing the DSP56321 to control data transmission by indirectly controlling the value of the SC[1–0] flags.

8.4 ESSI Programming Model

The ESSI is composed of the following registers:

- Two control registers (CRA, CRB), **page 8-13** and **page 8-17**
- One status register (SSISR), **page 8-26**
- One Receive Shift Register, **page 8-27**
- One Receive Data Register (RX), **page 8-28**
- Three Transmit Shift Registers, **page 8-28**
- Three Transmit Data Registers (TX0, TX1, TX2), **page 8-28**
- One special-purpose Time Slot Register (TSR), **page 8-31**
- Two Transmit Slot Mask Registers (TSMA, TSMB), **page 8-31**
- Two Receive Slot Mask Registers (RSMA, RSMB), **page 8-32**

This section discusses the ESSI registers and describes their bits. **Section 8.5**, *GPIO Signals and Registers*, on page 8-33 covers ESSI GPIO.

8.4.1 ESSI Control Register A (CRA)

The ESSI Control Register A (CRA) is one of two 24-bit read/write control registers that 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.

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

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

▨—Reserved bit; read as 0; write to 0 for future compatibility.

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

Table 8-3. ESSI Control Register A (CRA) Bit Definitions

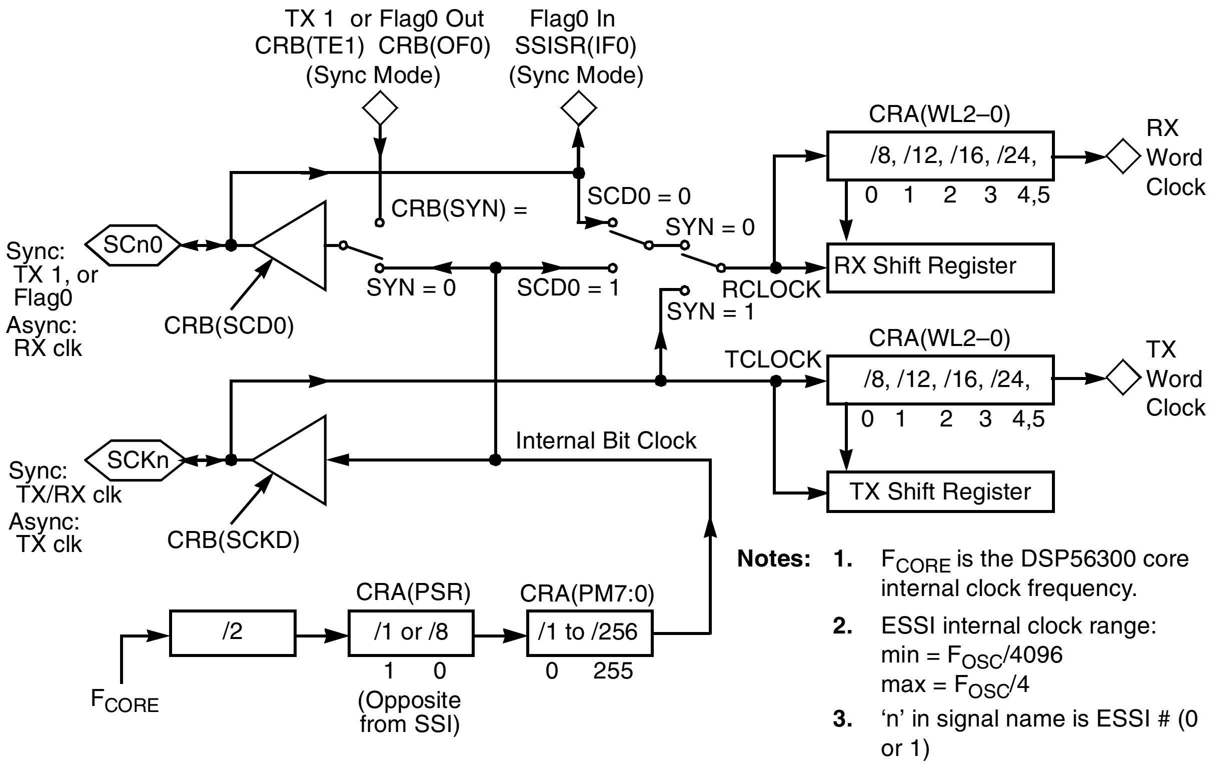
Bit Number	Bit Name	Reset Value	Description
23	▨	0	Reserved. Write to 0 for future compatibility.
22	SSC1	0	**Select SC1** Controls the functionality of the SC1 signal. If SSC1 is set, the ESSI is configured in Synchronous mode (the CRB synchronous/asynchronous bit (SYN) is set), and transmitter 2 is 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).

Table 8-3. ESSI Control Register A (CRA) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="4">Description</th></tr>
<tr><td rowspan="12">21–19</td><td rowspan="12">WL[2–0]</td><td rowspan="12">0</td><td colspan="4"><b>Word Length Control</b>
Select the length of the data words transferred via the ESSI. Word lengths of 8-, 12-, 16-, 24-, or 32-bits can be selected. The ESSI data path programming model in <b>Figure 8-8</b> and <b>Figure 8-9</b> shows additional information on how to select different lengths for data words. The ESSI data registers are 24 bits long. The ESSI transmits 32-bit words in one of two ways:
• by duplicating the last bit 8 times when WL[2–0] = 100
• by duplicating the first bit 8 times when WL[2–0] = 101.
<b>Note:</b> When WL[2–0] = 100, the ESSI is designed to duplicate the last bit of the 24-bit transmission eight times to fill the 32-bit shifter. Instead, after the 24-bit word is shifted correctly, eight zeros (0s) are shifted.</td></tr>
<tr><td colspan="4"><b>ESSI Word Length Selection</b></td></tr>
<tr><td><b>WL2</b></td><td><b>WL1</b></td><td><b>WL0</b></td><td><b>Number of Bits/Word</b></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>8</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>12</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>16</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>24</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>32
(valid data in the first 24 bits)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>32
(valid data in the last 24 bits)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Reserved</td></tr>
<tr><td colspan="4"><b>Note:</b> When the ESSI transmits data in On-Demand mode (that is, MOD = 1 in the CRB and DC[4–0]=00000 in the CRA) with WL[2–0] = 100, the transmission does not work properly. To ensure correct operation, do not use On-Demand mode with the WL[2–0] = 100 32-bit word length mode.</td></tr>
<tr><td>18</td><td>ALC</td><td>0</td><td colspan="4"><b>Alignment Control</b>
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.
<b>Note:</b> If the ALC bit is set, only 8-, 12-, or 16-bit words are used. The use of 24- or 32-bit words leads to unpredictable results.</td></tr>
<tr><td>17</td><td></td><td></td><td colspan="4">Reserved. Write to 0 for future compatibility.</td></tr>
</table>

Table 8-3. ESSI Control Register A (CRA) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
16–12	DC[4–0]	0	**Frame Rate Divider Control** Control the divide ratio for the programmable frame rate dividers that generate the frame clocks. In Network mode, this ratio is the number of words per frame minus one. In Normal mode, this ratio determines the word transfer rate. The divide ratio ranges 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). **Figure 8-2** shows the ESSI frame sync generator functional block diagram.
11	PSR	0	**Prescaler Range** Controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit extends the range of the prescaler when 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 8-1**. This definition is reversed from that of the SSI in other DSP56000 family members. The maximum allowed internally generated bit clock frequency is the internal DSP56321 clock frequency divided by 4; the minimum possible internally generated bit clock frequency is the DSP56321 internal clock frequency divided by 4096. **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.
10–8		0	Reserved. Write to 0 for future compatibility.
7–0	PM[7–0]	0	**Prescale Modulus Select** 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. **Figure 8-1** shows the ESSI clock generator functional block diagram. F_{core} is the DSP56321 core clock frequency (the same frequency as the enabled CLKOUT signal). Careful choice of the crystal oscillator frequency and the prescaler modulus can generate the industry-standard CODEC master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz.



Notes:
1. F_{CORE} is the DSP56300 core internal clock frequency.
2. ESSI internal clock range:
   min = F_{OSC}/4096
   max = F_{OSC}/4
3. ‘n’ in signal name is ESSI # (0 or 1)

Figure 8-2. ESSI Clock Generator Functional Block Diagram

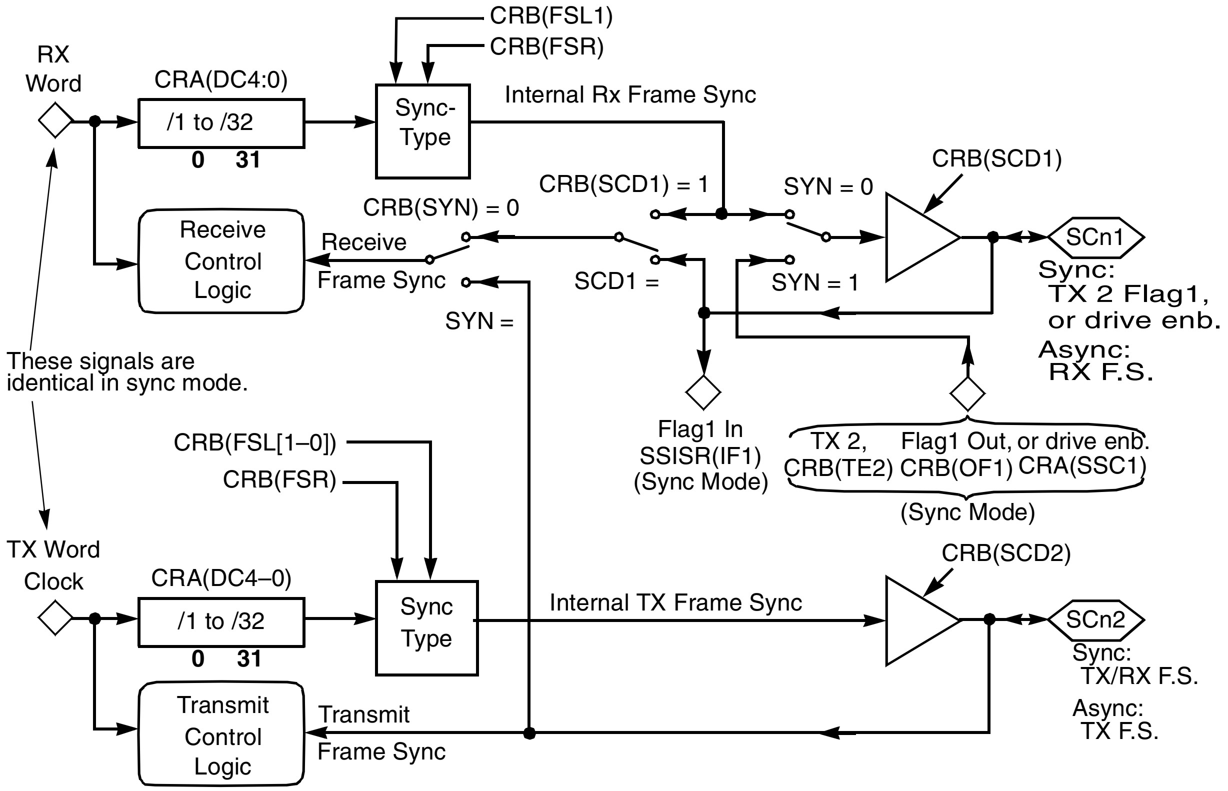


Figure 8-3. ESSI Frame Sync Generator Functional Block Diagram

8.4.2 ESSI Control Register B (CRB)

CRB is one of two read/write control registers that direct the operation of the ESSI. The CRB bit definitions are presented in **Table 8-4**. 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.

23	22	21	20	19	18	17	16	15	14	13	12
REIE	TEIE	RLIE	TLIE	RIE	TIE	RE	TE0	TE1	TE2	MOD	SYN
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CKP	FSP	FSR	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0

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

The CRB contains the serial output flag control bits and the direction control bits for the serial control signals. Also in the CRB are interrupt enable bits for the receiver and the transmitter. Bit settings of the CRB determines how many transmitters are enabled: 0, 1, 2, or 3. The CRB settings also determine the ESSI operating mode. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all the bits in the CRB. **Table 8-2**, *Mode and Signal Definitions,* on page 8-4 summarizes the relationship between the ESSI signals SC[2–0], SCK, and the CRB bits.

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. 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.

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

Table 8-4. ESSI Control Register B (CRB) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23	REIE	0	**Receive Exception Interrupt Enable** 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 8.2.3**, *Exceptions*, on page 8-7. A read of the status register followed by a read of the receive data register clears both ROE and the pending interrupt.
22	TEIE	0	**Transmit Exception Interrupt Enable** 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 8.2.3**, *Exceptions*, on page 8-7. 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.
21	RLIE	0	**Receive Last Slot Interrupt Enable** Enables/disables 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 8.2.3**, *Exceptions*, on page 8-7. RLIE is disabled when the ESSI is in On-Demand mode (DC = $0).
20	TLIE	0	**Transmit Last Slot Interrupt Enable** Enables/disables 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 8.2.3**, *Exceptions*, on page 8-7. TLIE is disabled when the ESSI is in On-Demand mode (DC = $0).
19	RIE	0	**Receive Interrupt Enable** Enables/disables 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 8.2.3**, *Exceptions*, on page 8-7. 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.
18	TIE	0	**Transmit Interrupt Enable** Enables/disables 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 8.2.3**. 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.

Table 8-4. ESSI Control Register B (CRB) Bit Definitions (Continued)

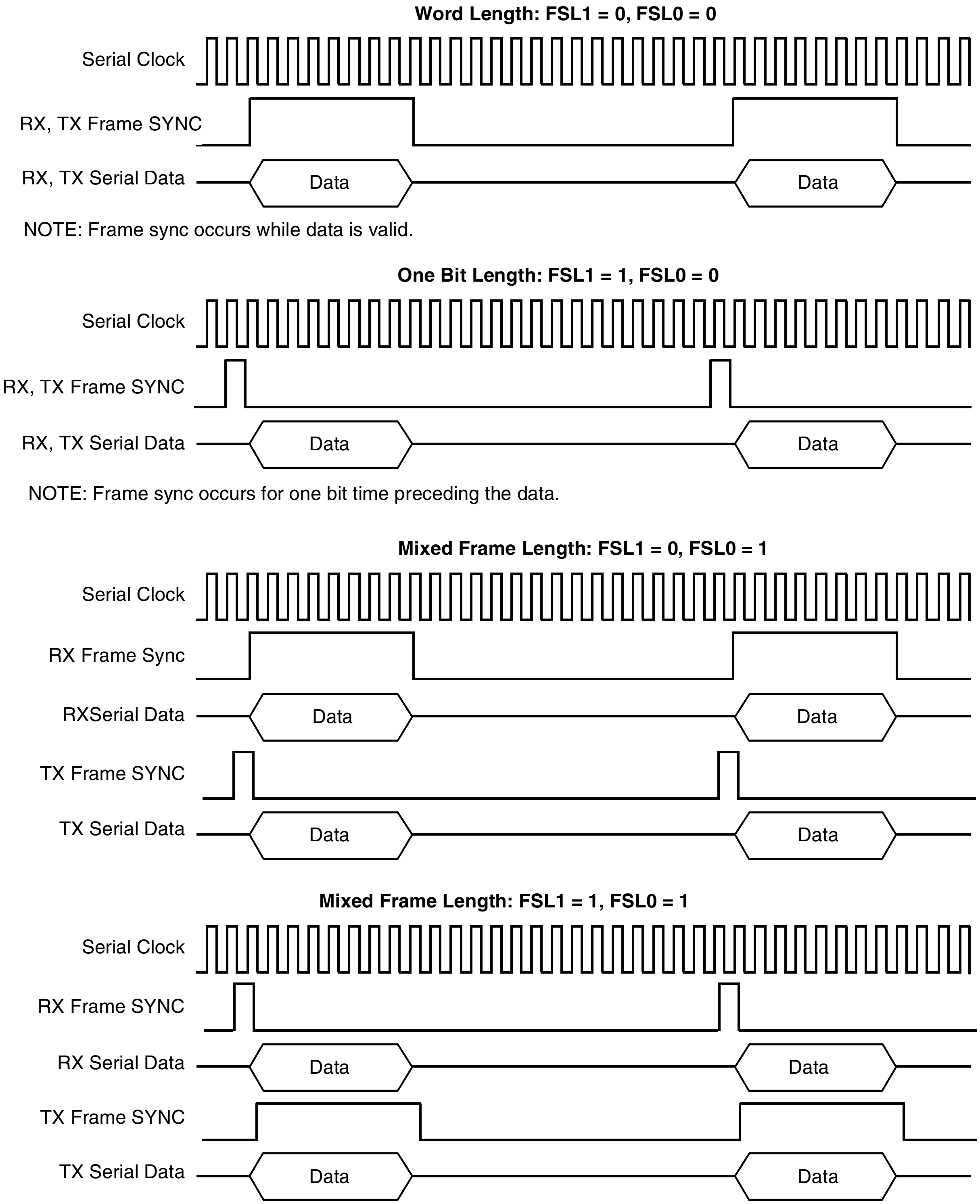
Bit Number	Bit Name	Reset Value	Description
17	RE	0	**Receive Enable** Enables/disables the receive portion of the ESSI. When RE is cleared, the receiver is disabled: data transfer into RX is inhibited. If data is being received while this bit is cleared, the remainder of the word is shifted in and transferred 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. The setting of the RE bit does not affect the generation of a frame sync.
16	TE0	0	**Transmit 0 Enable** Enables the transfer of data from TX0 to Transmit Shift Register 0. TE0 is functional when the ESSI is in either synchronous or Asynchronous mode. When TE0 is set and a frame sync is detected, the transmitter 0 is enabled for that frame. When TE0 is cleared, transmitter 0 is disabled after 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 transmit enable sequence in On-Demand mode can be the same as in Normal mode, or TE0 can be left enabled. Transmitter 0 is the only transmitter that can operate in Asynchronous mode (SYN = 0). The setting of the TE0 bit does not affect the generation of frame sync or output flags.
15	TE1	0	**Transmit 1 Enable** 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 setting of the TE1 bit does not affect the generation of frame sync or output flags.
14	TE2	0	**Transmit 2 Enable** 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 mode 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 setting of the TE2 bit does not affect the generation of frame sync or output flags.

Table 8-4. ESSI Control Register B (CRB) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th>Description</th></tr>
<tr><td>13</td><td>MOD</td><td>0</td><td><b>Mode Select</b>
Selects the operational mode of the ESSI, as in <b>Figure 8-5</b> on page -24, <b>Figure 8-6</b> on page -25, and <b>Figure 8-7</b> on page -25. 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 <b>Section 8.2</b>.</td></tr>
<tr><td>12</td><td>SYN</td><td>0</td><td><b>Synchronous/Asynchronous</b>
Controls whether the receive and transmit functions of the ESSI occur synchronously or asynchronously with respect to each other. (See <b>Figure 8-4</b> on page -23.) 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.</td></tr>
<tr><td>11</td><td>CKP</td><td>0</td><td><b>Clock Polarity</b>
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.</td></tr>
<tr><td>10</td><td>FSP</td><td>0</td><td><b>Frame Sync Polarity</b>
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.</td></tr>
<tr><td>9</td><td>FSR</td><td>0</td><td><b>Frame Sync Relative Timing</b>
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 (that is, simultaneously with the last bit of the previous data word).</td></tr>
<tr><td>8–7</td><td>FSL[1–0]</td><td>0</td><td><b>Frame Sync Length</b>
Selects the length of frame sync to be generated or recognized, as in <b>Figure 8-3</b> on page -22, <b>Figure 8-6</b> on page -25, and <b>Figure 8-7</b> on page -25.
<table>
<tr><th rowspan="2">FSL1</th><th rowspan="2">FSL0</th><th colspan="2">Frame Sync Length</th></tr>
<tr><th>RX</th><th>TX</th></tr>
<tr><td>0</td><td>0</td><td>word</td><td>word</td></tr>
<tr><td>0</td><td>1</td><td>word</td><td>bit</td></tr>
<tr><td>1</td><td>0</td><td>bit</td><td>bit</td></tr>
<tr><td>1</td><td>1</td><td>bit</td><td>word</td></tr>
</table></td></tr>
<tr><td>6</td><td>SHFD</td><td>0</td><td><b>Shift Direction</b>
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 <b>Figure 8-8</b> on page -29 and <b>Figure 8-9</b> on page -30.</td></tr>
</table>

Table 8-4. ESSI Control Register B (CRB) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
5	SCKD	0	**Clock Source Direction** 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.
4	SCD2	0	**Serial Control Direction 2** 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. NOTE: Programming the ESSI to use an internal frame sync (that is, SCD2 = 1 in CRB) causes the SC2 and SC1 signals to be programmed as outputs. However, if the corresponding multiplexed pins are programmed by the Port Control Register (PCR) to be GPIOs, the GPIO Port Direction Register (PRR) chooses their direction. The ESSI uses an external frame sync if GPIO is selected. To assure correct operation, either program the GPIO pins as outputs or configure the pins in the PCR as ESSI signals. The default selection for these signals after reset is GPIO. This note applies to both ESSI0 and ESSI1.
3	SCD1	0	**Serial Control Direction 1** 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.
2	SCD0	0	**Serial Control Direction 0** 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.
1	OF1	0	**Serial Output Flag 1** In Synchronous mode (SYN = 1), when transmitter 2 is disabled (TE2 = 0), the SC1 signal is configured as ESSI flag 1. When SCD1 is set, SC1 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.
0	OF0	0	**Serial Output Flag 0** In Synchronous mode (SYN = 1), when transmitter 1 is disabled (TE1 = 0), the SC0 signal is configured as ESSI flag 0. When 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.



Word Length: FSL1 = 0, FSL0 = 0

NOTE: Frame sync occurs while data is valid.

One Bit Length: FSL1 = 1, FSL0 = 0

NOTE: Frame sync occurs for one bit time preceding the data.

Mixed Frame Length: FSL1 = 0, FSL0 = 1

Mixed Frame Length: FSL1 = 1, FSL0 = 1

Figure 8-4. CRB FSL0 and FSL1 Bit Operation (FSR = 0)

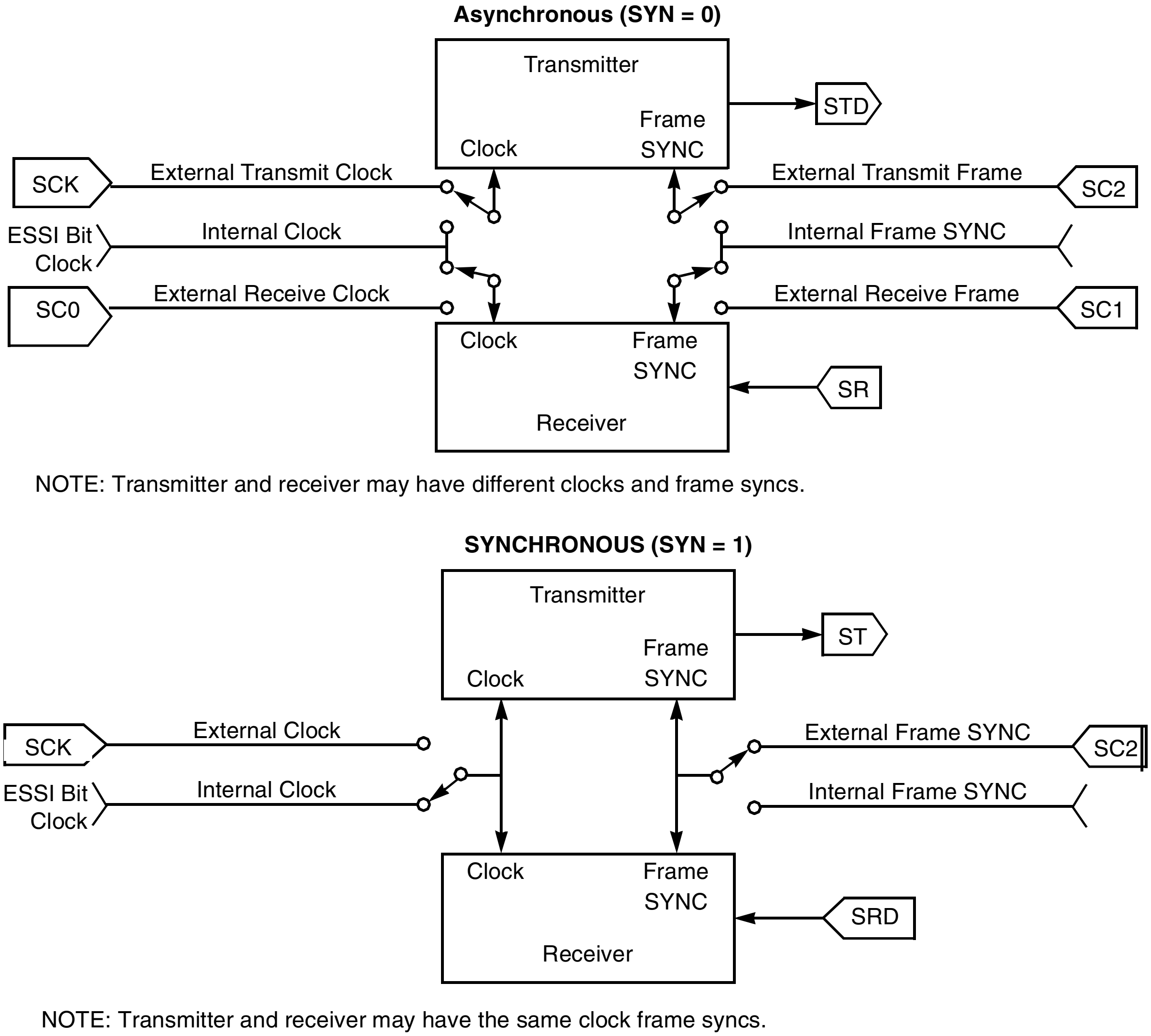
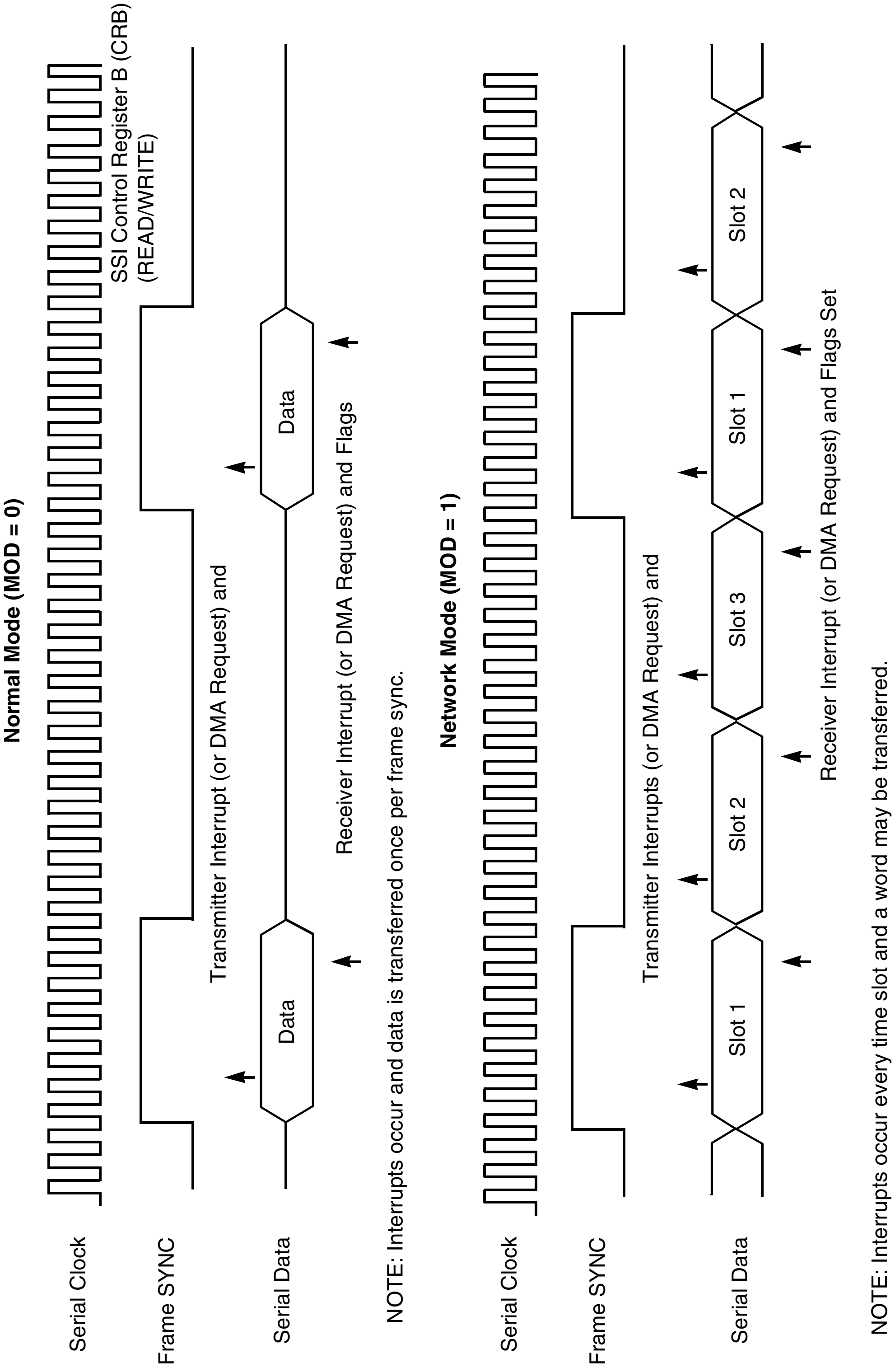


Figure 8-5. CRB SYN Bit Operation


Normal Mode (MOD = 0)
Serial Clock
Frame SYNC
Serial Data
Data
Data
SSI Control Register B (CRB)
(READ/WRITE)
Transmitter Interrupt (or DMA Request) and
Receiver Interrupt (or DMA Request) and Flags
Network Mode (MOD = 1)
Serial Clock
Frame SYNC
Serial Data
Slot 1
Slot 2
Slot 3
Slot 1
Slot 2
Transmitter Interrupts (or DMA Request) and
Receiver Interrupt (or DMA Request) and Flags Set

NOTE: Interrupts occur and data is transferred once per frame sync.

NOTE: Interrupts occur every time slot and a word may be transferred.

Figure 8-6. CRB MOD Bit Operation

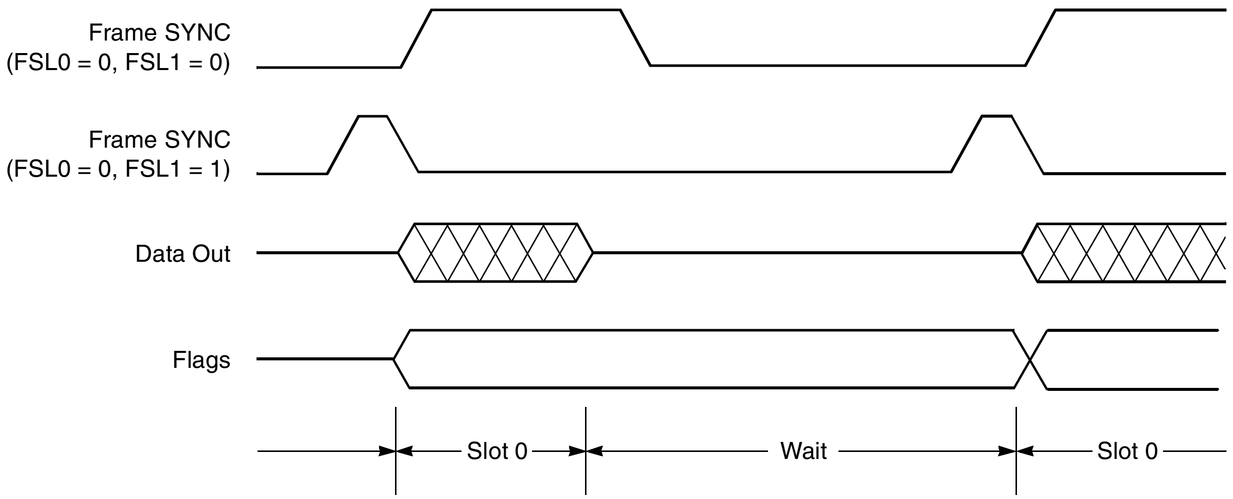


Figure 8-7. Normal Mode, External Frame Sync (8 Bit, 1 Word in Frame)

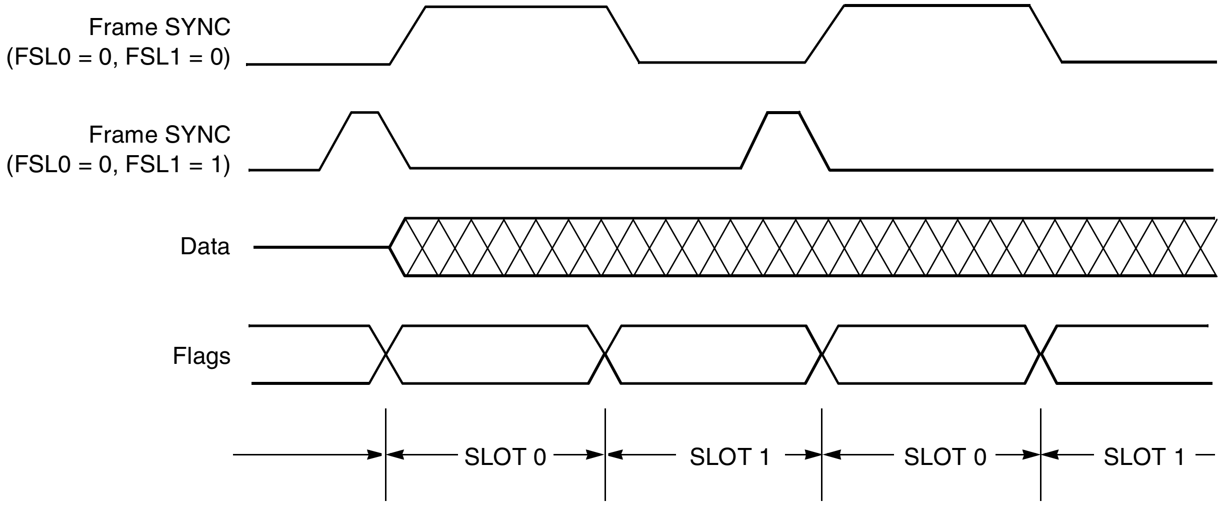


Figure 8-8. Network Mode, External Frame Sync (8 Bit, 2 Words in Frame)

8.4.3 ESSI Status Register (SSISR)

The SSISR is a read-only status register by which the DSP reads the ESSI status and serial input flags.

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
▨	▨	▨	▨	RDF	TDE	ROE	TUE	RFS	TFS	IF1	IF0

▨ —Reserved bit; read as 0; write to 0 0 for future compatibility.

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

Table 8-5. ESSI Status Register (SSISR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23–8		0	Reserved. Write to 0 for future compatibility.
7	RDF	0	**Receive Data Register Full** Set when the contents of the receive shift register transfer to the receive data register. RDF 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.
6	TDE	0	**Transmit Data Register Empty** 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 TDE is set, TDE data is written to all the TX registers of the enabled transmitters or to the TSR. The TDE bit is cleared when the DSP 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.
5	ROE	0	**Receiver Overrun Error Flag** Set when the serial receive shift register is filled and ready to transfer to the receive data register (RX) but RX is already full (that is, 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. The programmer clears ROE by reading the SSISR with the ROE bit set and then reading the RX.
4	TUE	0	**Transmitter Underrun Error Flag** Set when at least one of the enabled serial transmit shift registers is empty (that is, 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. 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.
3	RFS	0	**Receive Frame Sync Flag** 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 (that is, if the RE bit is set). 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.
2	TFS	0	**Transmit Frame Sync Flag** 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 that is, when TE0, TE1, or TE2 is set). 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.

Table 8-5. ESSI Status Register (SSISR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
1	IF1	0	**Serial Input Flag 1** The ESSI latches any data on the SC1 signal during reception of the first received bit after the frame sync is detected. IF1 is updated with this data when the data in the receive shift register transfers into the receive data register. IF1 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. If it is not enabled, IF1 is cleared.
0	IF0	0	**Serial Input Flag 0** The ESSI latches any data 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 transfers into the receive data register. IF0 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. If it is not enabled, the IF0 bit is cleared.

8.4.4 ESSI Receive Shift Register

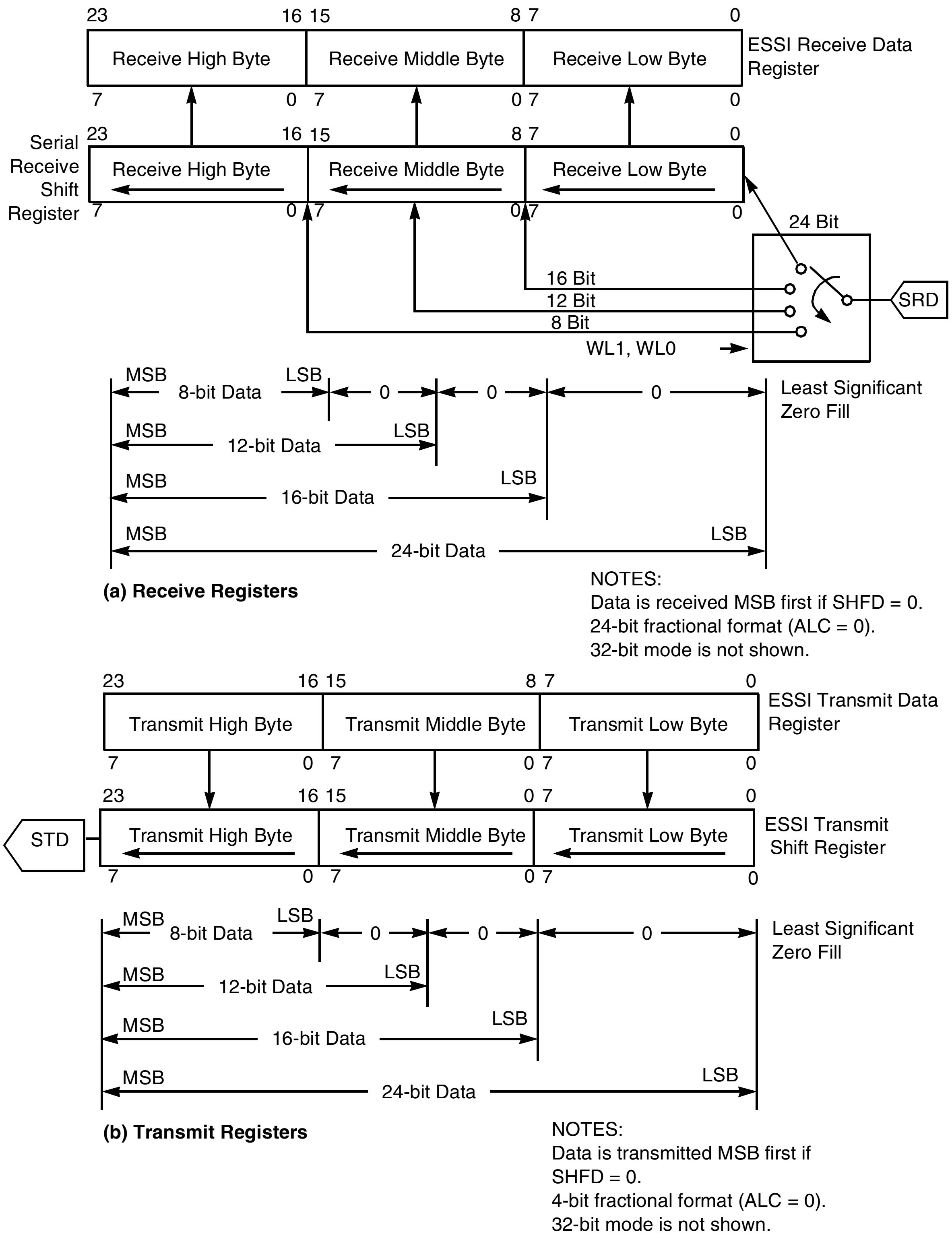
The 24-bit Receive Shift Register (see **Figure 8-8** and **Figure 8-9**) receives incoming data from the serial receive data signal. The selected (internal/external) bit clock shifts data in when the associated frame sync I/O is asserted. Data is received MSB first if SHFD is cleared and LSB first if SHFD is set. Data transfers to the ESSI Receive Data Register (RX) after 8, 12, 16, 24, or 32 serial clock cycles are counted, depending on the word length control bits in the CRA.

8.4.5 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 8-8** and **Figure 8-9**. 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.

8.4.6 ESSI Transmit Shift Registers

The three 24-bit transmit shift registers contain the data being transmitted, as in **Figure 8-8** and **Figure 8-9**. 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. Transmitted data is aligned according to the value of the ALC bit. When ALC 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 shifts out of these registers MSB first if the SHFD bit is cleared and LSB first if SHFD is set.



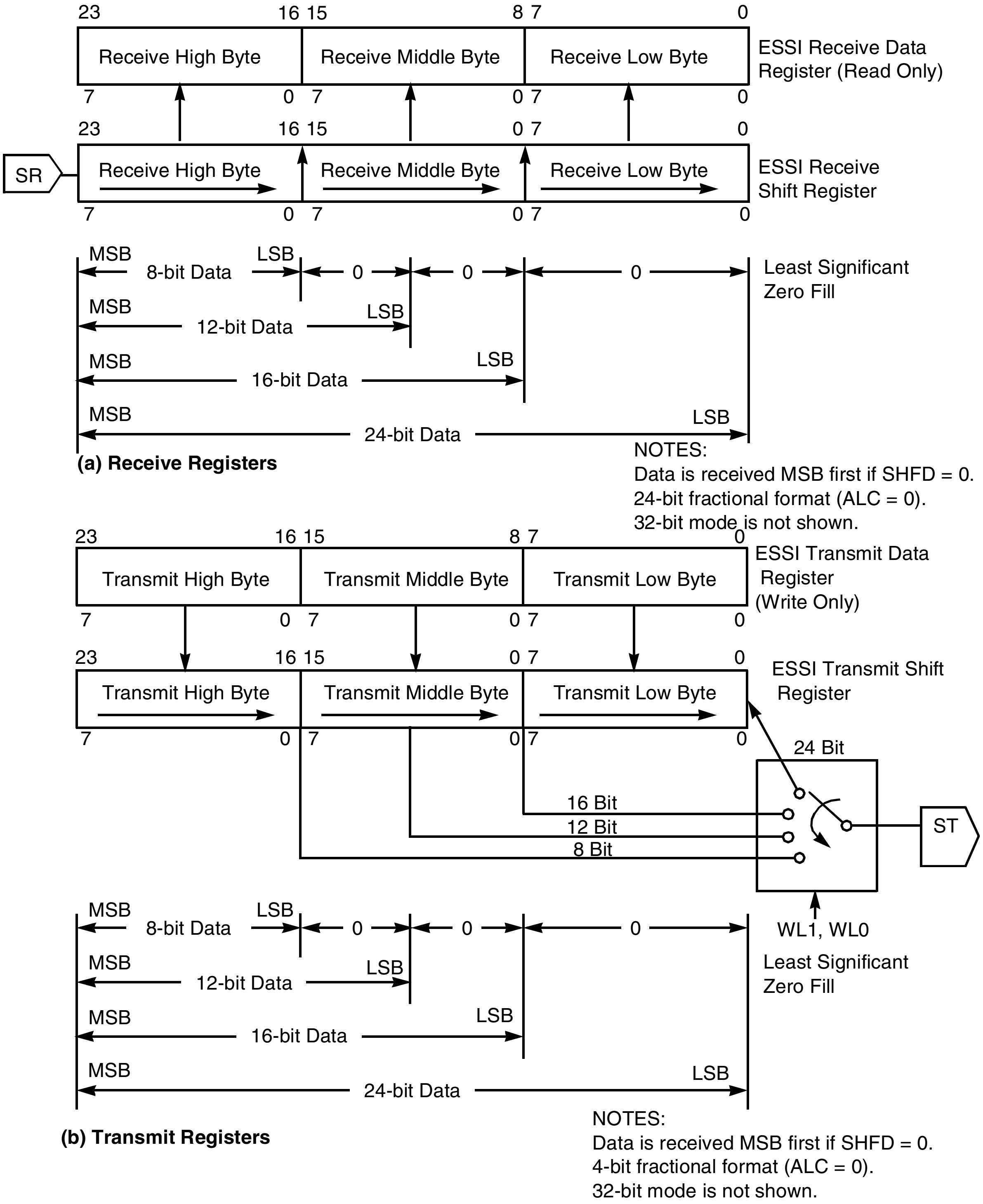
(a) Receive Registers

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

(b) Transmit Registers

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

Figure 8-9. ESSI Data Path Programming Model (SHFD = 0)



(a) Receive Registers

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

(b) Transmit Registers

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

Figure 8-10. ESSI Data Path Programming Model (SHFD = 1)

8.4.7 ESSI Transmit Data Registers (TX[2–0])

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

TX2, TX1, and TX0 are 24-bit write-only registers. Data written into these registers automatically transfers to the transmit shift registers. (See **Figure 8-8** and **Figure 8-9**.) 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. When data is written to a peripheral device, there is a two-cycle pipeline delay while any status bits affected by this operation are updated. If any of those status bits are read during the two-cycle delay, the status bit may not reflect the current status.

8.4.8 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 as 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.

8.4.9 Transmit Slot Mask Registers (TSMA, TSMB)

Both transmit slot mask registers are 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 bit settings, the transmitter(s) either tri-state the transmitter(s) data signal(s) or transmit a data word and generate a transmitter empty condition.

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	TS15	TS14	TS13	TS12
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TS11	TS10	TS9	TS8	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0

▨ —Reserved bit; read as 0; write to 0 0 for future compatibility.

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

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	TS31	TS30	TS29	TS28
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TS27	TS26	TS25	TS24	TS23	TS22	TS21	TS20	TS19	TS18	TS17	TS16

▨—Reserved bit; read as 0; write to 0 0 for future compatibility.

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

TSMA and TSMB (as in **Figure 8-8** and **Figure 8-9**) 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 data signals of the enabled transmitters are tri-stated during transmit time slot number N. The data still transfers 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 serviced transmits in the next enabled transmit time slot. When TSn is set, the transmit sequence proceeds normally. Data transfers 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, enabling all 32 slots for data transmission.

8.4.10 Receive Slot Mask Registers (RSMA, RSMB)

Both receive slot mask registers are 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.

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	RS15	RS14	RS13	RS12
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RS11	RS10	RS9	RS8	RS7	RS6	RS5	RS4	RS3	RS2	RS1	RS0

▨—Reserved bit; read as 0; write to 0 0 for future compatibility.

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

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	RS31	RS30	RS29	RS28
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RS27	RS26	RS25	RS24	RS23	RS22	RS21	RS20	RS19	RS18	RS17	RS16

▨–Reserved. Read as zero. Write with zero for future compatibility.

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

RSMA and RSMB 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 transfers from the receive data register(s) to the receive shift register(s), but the RDF and ROE flags are not set. Consequently, 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 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, enabling all 32 time slots for data transmission.

8.5 GPIO Signals and Registers

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

8.5.1 Port Control Registers (PCRC and PCRD)

The read/write 24-bit PCRs control the functionality of the signal lines for ESSI0 and ESSI1. Each of the PCR bits 5–0 controls the functionality of the corresponding signal line. When a PCR[i] bit is set, the corresponding port signal is configured as an ESSI signal. When a PCR[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.

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
▨	▨	▨	▨	▨	▨	PCx5	PCx4	PCx3	PCx2	PCx1	PCx0

Note: For Px[5–0], a 0 selects Pxn as the signal and a 1 selects the specified ESSI signal. For ESSI0, the GPIO signals are PC[5–0] and the ESSI signals are STD0, SRD0, SCK0, and SC0[2–0]. For ESSI1, the GPIO signals are PD[5–0] and the ESSI signals are STD1, SRD1, SCK1, and SC1[2–0].

▨ = Reserved. Read as zero. Write with zero for future compatibility.

8.5.2 Port Direction Registers (PRRC and PRRD)

The read/write PRRC and PRRD control the data direction of the ESSI0 and ESSI1 GPIO signals when they are enabled by the associated Port Control Register (PCRC or PCRD, respectively). When PRRC[i] or PRRD[i] is set, the corresponding signal is an output (GPO) signal. When PRRC[i] or PRRD[i] is cleared, the corresponding signal is an input (GPI) signal. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PRRC and PRRD bits.

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
▨	▨	▨	▨	▨	▨	PRx5	PRx4	PRx3	PRx2	PRx1	PRx0

Note: For bits 5–0, a 0 configures PRxn as a GPI and a 1 configures PRxn as a GPO. For ESSI0, the GPIO signals are PC[5–0]. For ESSI1, the GPIO signals are PD[5–0]. The corresponding direction bits for Port C GPIOs are PRC[5–0]. The corresponding direction bits for Port D GPIOs are PRD[5–0].

▨ = Reserved. Read as zero. Write with zero for future compatibility.

Table 8-6 summarizes the ESSI port signal configurations

Table 8-6. ESSI Port Signal Configurations

PCRC/PCRD[i]	PRRC/PRRD[i]	Port Signal[i] Function
1	X	ESSI0/ESSI1
0	0	Port C/Port D GPI
0	1	Port C/Port D GPO
X: The signal setting is irrelevant to the Port Signal[i] function.		

8.5.3 Port Data Registers (PDRC and PDRD)

Bits 5–0 of the read/write PDRs write data to or read data from the associated ESSI GPIO signal lines if they are configured as GPIO signals. If a port signal PC[i] or PD[i] is configured as an input (GPI), the corresponding PDRC[i] pr PDRD[i] bit reflects the value present on the input signal line. If a port signal PC[i] or PD[i] is configured as an output (GPO), a value written to the corresponding PDRC[i] pr PDRD[i] bit is reflected as a value on the output signal line. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PDRC and PDRD bits.

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

11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	PDRx5	PDRx4	PDRx3	PDRx2	PDRx1	PDRx0

Note: For bits 5–0, the value represents the level that is written to or read from the associated signal line if it is enabled as a GPIO signal by the respective port control register (PCRC or PCRD) bits. For ESSI0, the GPIO signals are PC[5–0]. For ESSI1, the GPIO signals are PD[5–0]. The corresponding data bits for Port C GPIOs are PDRC[5–0]. The corresponding data bits for Port D GPIOs are PDRD[5–0].

▨ = Reserved. Read as zero. Write with zero for future compatibility.

Serial Communication Interface 9

The DSP56321 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 RS-232, RS-422, and so on. 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. SCI asynchronous protocols include a multi-drop mode for master/slave operation with wake-up on idle line and wake-up on address bit capability. This mode allows the DSP56321 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 supplies the transmit and receive clocks. An enable vector and an interrupt vector are included so that the baud-rate generator can function as a general-purpose timer when the SCI is not using it, or when the interrupt timing is the same as that used by the SCI.

9.1 Operating Modes

The operating modes for the DSP56321 SCI are as follows:

- 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 multi-drop asynchronous (1 start, 8 data, 1 data type, 1 stop)
  This mode is used for master/slave operation with wake-up on idle line and wake-up on address bit capability. It allows the DSP56321 to share a single serial line efficiently with other peripherals.

These modes are selected by the SCR WD[2–0] bits. Synchronous data mode is essentially a high-speed shift register 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 RS-232 communication links are supported by these modes. Multi-drop Asynchronous mode is compatible with the MC68681 DUART, the M68HC11 SCI interface, and the Intel 8051 serial interface.

9.1.1 Synchronous Mode

Synchronous mode (SCR[WD2–0]=000, Shift Register mode) handles serial-to-parallel and parallel-to-serial conversions. In Synchronous mode, the clock is always common to the transmit and receive shift registers. As a controller (synchronous master), the DSP puts out a clock on the SCLK pin. To select master mode, choose the internal transmit and receive clocks (set TCM and RCM=0).

As a peripheral (synchronous slave), the DSP accepts an input clock from the SCLK pin. To select the slave mode, choose the external transmit and receive clocks (TCM and RCM=1). Since there is no frame signal, if a clock is missed because of noise or any other reason, the receiver loses synchronization with the data without any error signal being generated. You can detect an error of this type with an error detecting protocol or with external circuitry such as a watchdog timer. The simplest way to recover synchronization is to reset the SCI.

9.1.2 Asynchronous Mode

Asynchronous data uses a data format with embedded word sync, which allows an unsynchronized data clock to be synchronized with the word if the clock rate and number of bits per word is known. Thus, the clock can be generated by the receiver rather than requiring a separate clock signal. The transmitter and receiver both use an internal clock that is 16 times the data rate to allow the SCI to synchronize the data. The data format requires that each data byte have an additional start bit and stop bit. Also, two of the word formats have a parity bit. The Multi-drop mode used when SCIs are on a common bus has an additional data type bit. The SCI can operate in full-duplex or half-duplex modes since the transmitter and receiver are independent.

9.1.3 Multi-Drop Mode

Multi-drop is a special case of asynchronous data transfer. The key difference is that a protocol allows networking transmitters and receivers on a single data-transmission line. Inter-processor messages in a multi-drop network typically begin with a destination address. All receivers check for an address match at the start of each message. Receivers with no address match can ignore the remainder of the message and use a wake-up mode to enable the receiver at the start of the next message. Receivers with an address match can receive the message and optionally transmit an acknowledgment to the sender. The particular message format and protocol used are determined by the user’s software.

9.1.3.1 Transmitting Data and Address Characters

To send data, the 8-bit data character must be written to the STX register. Writing the data character to the STX register sets the ninth bit in the frame to zero, which indicates that this frame contains data. To send an 8-bit address, the address data is written to the STXA register, and the ninth bit in the frame is set to one, indicating that this frame contains an address.

9.1.3.2 Wired-OR Mode

Building a multi-drop bus network requires connecting multiple transmitters to a common wire. The Wired-OR mode allows this to be done without damaging the transmitters when the transmitters are not in use. A protocol is still needed to prevent two transmitters from simultaneously driving the bus. The SCI multi-drop word format provides an address field to support this protocol.

9.1.3.3 Idle Line Wake-up

A wake-up mode frees a DSP from reading messages intended for other processors. The usual operational procedure is for each DSP to suspend SCI reception (the DSP can continue processing) until the beginning of a message. Each DSP compares the address in the message header with the DSP address. If the addresses do not match, the SCI again suspends reception until the next address. If the address matches, the DSP reads and processes the message and then suspends reception until the next address. The Idle Line Wake-up mode wakes up the SCI to read a message before the first character arrives.

9.1.3.4 Address Mode Wake-up

The purpose and basic operational procedure for Address Mode Wake-up is the same as for Idle Line Wake-up. The difference is that Address Mode Wake-up re-enables the SCI when the ninth bit in a character is set to one (if cleared, this bit marks a character as data; if set, an address). As a result, an idle line is not needed, which eliminates the dead time between messages.

9.2 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.

To enable SCI interrupts, program 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.

9.2.1 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 the SCI Clock Polarity (SCKP) bit is cleared. RXD can be configured as a GPIO signal (PE0) when the SCI RXD function is not in use.

9.2.2 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 in use.

9.2.3 SCI Serial Clock (SCLK)

SCLK is a bidirectional signal that 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 in use. This signal can be programmed as PE2 when data is 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.

9.3 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 all be cleared (that is, 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 remain unaffected so the DSP program can 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 is unknown.
- Stop processing state reset (that is, the STOP instruction). Executing the STOP instruction halts operation of the SCI until the DSP is restarted, causing the SCI Status Register (SSR) to be reset. No other SCI registers are affected by the STOP instruction.

Table 9-1 illustrates how each type of reset affects each register in the SCI.

Table 9-1. SCI Registers After Reset

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

Table 9-1. SCI Registers After Reset (Continued)

Register	Bit Mnemonic	Bit Number	Reset Type			
			HW Reset	**SW Reset**	**IR Reset**	**ST Reset**
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.

9.4 SCI Initialization

The SCI is initialized as follows:

1. Ensure that the SCI is in its individual reset state (PCRE = \$0). Use a hardware $\overline{\text{RESET}}$ signal or software RESET instruction.
2. Program the SCI control registers.
3. Configure at least one SCI signal as an SCI signal.

If interrupts are to be used, the signals must be selected, and global 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 enable the SCI, but the interrupts should be unmasked last (that is, I[1–0] bits in the Status Register (SR) should be changed last). Synchronous applications usually require exact frequencies, so 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. When the SCI is configured in Synchronous mode, internal clock, and all the SCI pins are simultaneously enabled, an extra pulse of one DSP clock length is provided on the SCLK pin. There are two workarounds for this issue:

- Enable an SCI pin other than SCLK.
- In the next instruction, enable the remaining SCI pins, including the SCLK pin.

Following is an example of one way to initialize the SCI:

1. Ensure that the SCI is in its individual reset state (PCRE = \$0).
2. Configure the control registers (SCR, SCCR) according to the operating mode, but do not enable transmitter (TE = 0) or receiver (RE = 0).

   You can now set the interrupt enable bits that are used during the operation. No interrupt occurs yet.

3. Enable the SCI by setting the PCRE 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. The DMA channel services the SCI transmit request 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 (that is, 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 (that is, 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).

9.4.1 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: preamble, break, data.

9.4.2 Bootstrap Loading Through the SCI (Boot Mode 2 or A)

When the DSP comes out of reset, it checks the MODD, MODC, MODB, and MODA pins and sets the corresponding mode bits in the Operating Mode Register (OMR). If the mode bits are write to 0010 or 1010, respectively, the DSP loads the program RAM from the SCI. **Appendix A**, *Bootstrap Program* shows the complete bootstrap code. This program (1) configures the SCI, (2) loads the program size, (3) loads the location where the program begins loading in program memory, and (4) loads the program. First, the SCI Control Register is set to $000302, which enables the transmitter and receiver and configures the SCI for 10 bits asynchronous with one start bit, 8 data bits, one stop bit, and no parity. Next, the SCI Clock Control Register is set to $00C000, which configures the SCI to use external receive and transmit clocks from the SCLK pin input. This external clock must be 16 times the desired serial data rate.

The next step is to receive the program size and then the starting address to load the program. These two numbers are three bytes each loaded least significant byte first. Each byte is echoed back as it is received. After both numbers are loaded, the program size is in A0 and the starting address is in A1.

The program is then loaded one byte at a time, least significant byte first. After the program is loaded, the operating mode is set to zero, the CCR is cleared, and the DSP begins execution with the first instruction loaded

9.5 Exceptions

The SCI can cause five different exceptions in the DSP, discussed here from the highest to the lowest priority:

1. SCI receive data with exception status occurs when the receive data register is full with a receiver error (parity, framing, or overrun error). To clear the pending interrupt, read the SCI status register; then read SRX. Use a long interrupt service routine to handle the error condition. This interrupt is enabled by SCR[16] (REIE).
2. SCI receive data occurs when the receive data register is 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[11] (RIE).
3. SCI transmit data occurs when the transmit data register is 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[12] (TIE).
4. SCI idle line occurs when the receive line enters the idle state (10 or 11 bits of ones). This interrupt is latched and then automatically reset when the interrupt is accepted. This interrupt is enabled by SCR[10] (ILIE).
5. SCI timer occurs when the baud rate counter reaches zero. This interrupt is automatically reset when the interrupt is accepted. This interrupt is enabled by SCR[13] (TMIE).

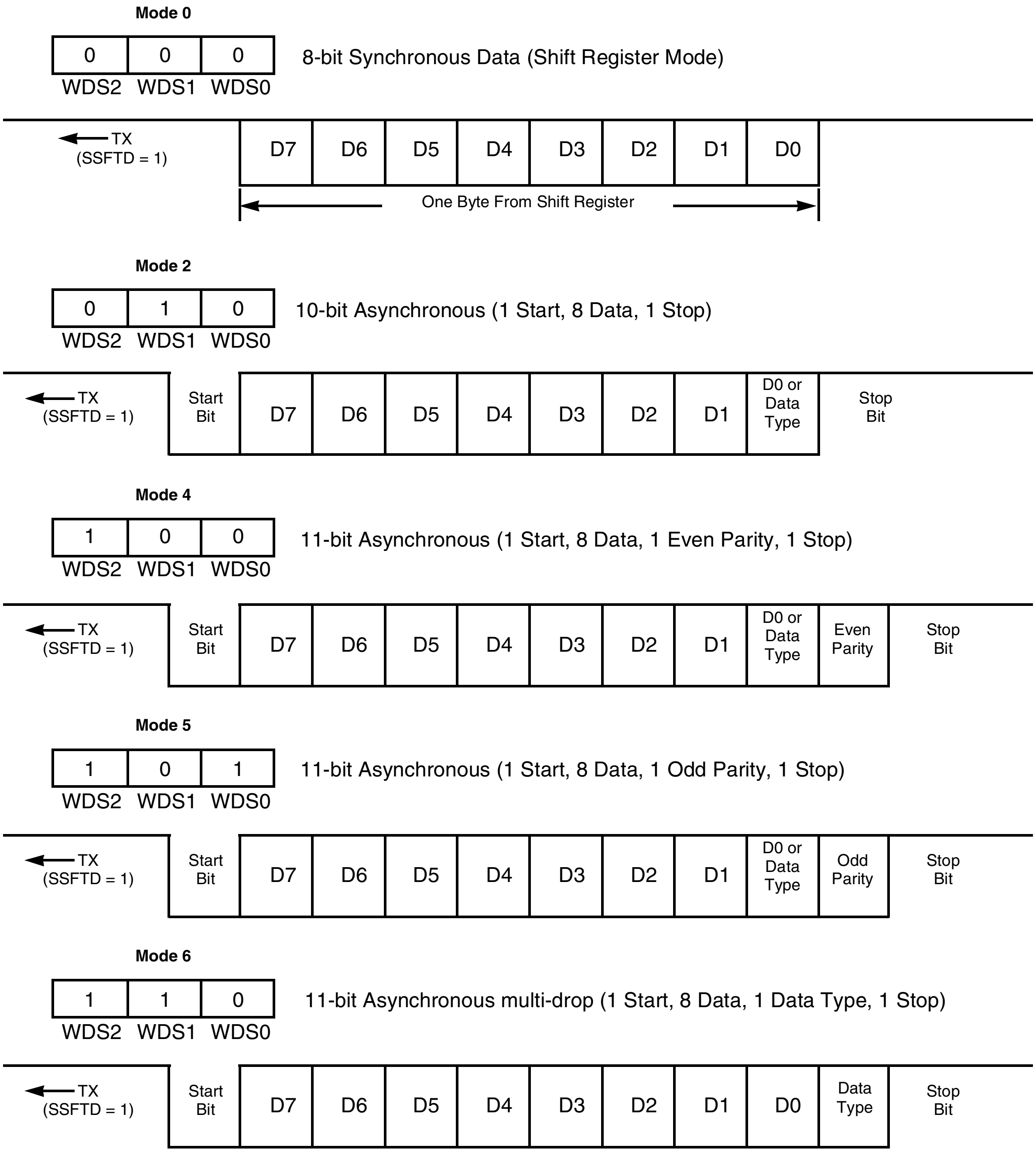
9.6 SCI Programming Model

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

- Control
  - SCI Control Register (SCR), **page 9-11**
  - SCI Clock Control Register (SCCR) in **page 9-16**
- Status
  - SCI Status Register (SSR) in **page 9-14**
- Data transfer
  - SCI Receive Data Registers (SRX) in **page 9-20**

— SCI Transmit Data Registers (STX) in **Figure 9-1**
— SCI Transmit Data Address Register (STXA) in **Figure 9-1**

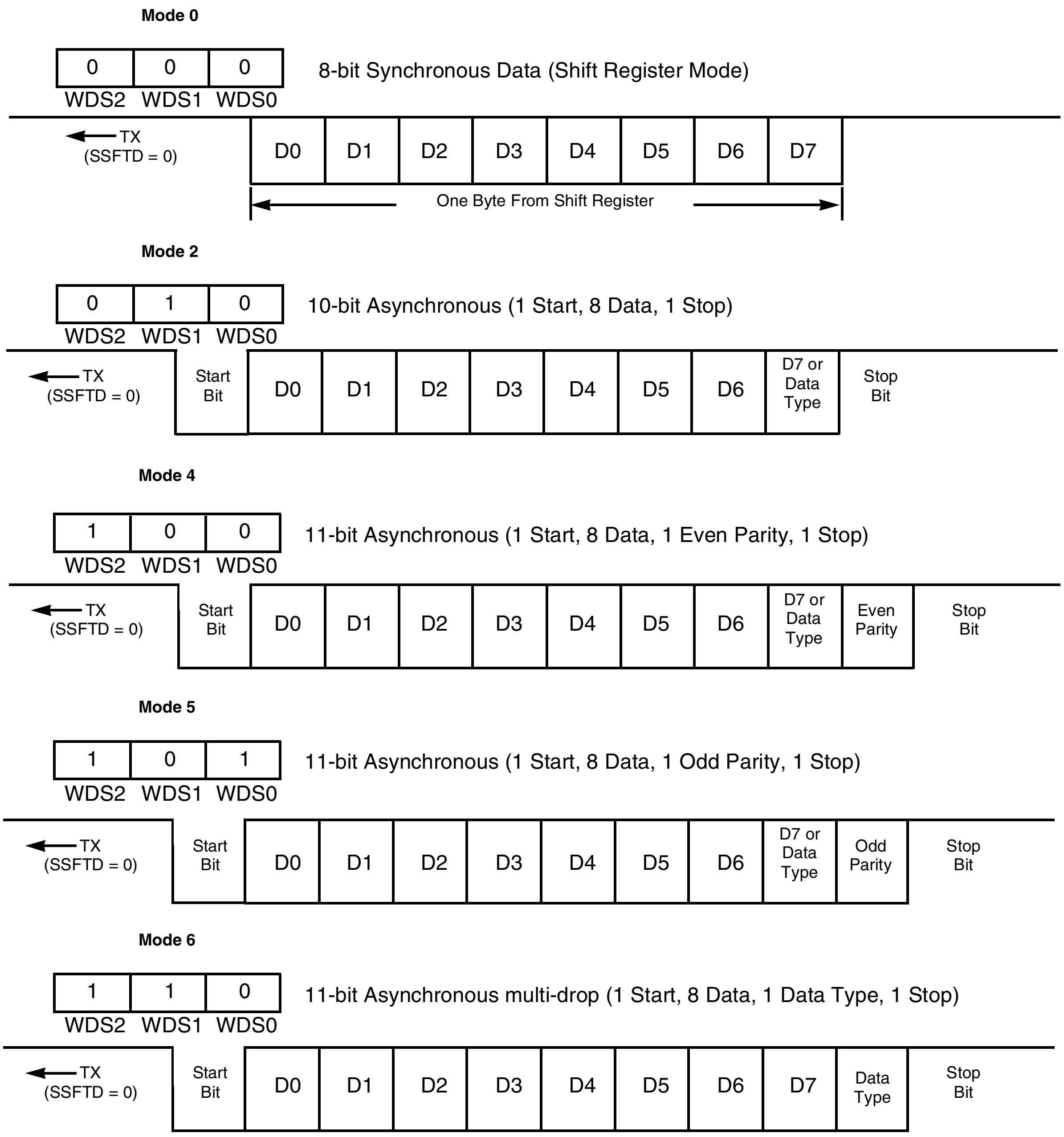
The SCI includes the GPIO functions described in **Section 9.7**, *GPIO Signals and Registers*, on page 9-22. The next subsections describe the registers and their bits.



Data Type: 1 = Address Byte
0 = Data Byte

Notes:
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 SSFTD = 1

Figure 9-1. SCI Data Word Formats (SSFTD = 1), 1



Data Type: 1 = Address Byte
0 = Data Byte

Notes:
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 SSFTD = 1.

Figure 9-2. SCI Data Word Formats (SSFTD = 0), 2

9.6.1 SCI Control Register (SCR)

The SCR is a read/write register that controls the serial interface operation.

23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	REIE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SCKP	STIR	TMIE	TIE	RIE	ILIE	TE	RE
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WOMS	RWU	WAKE	SBK	SSFTD	WDS2	WDS1	WDS0

▨—Reserved bit; read as 0; write to 0 for future compatibility.

Table 9-2. SCI Control Register (SCR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23–17	▨	0	Reserved. Write to 0 for future compatibility.
16	REIE	0	**Receive with Exception Interrupt Enable** Enables/disables 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.
15	SCKP	0	**SCI Clock Polarity** 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.
14	STIR	0	**Timer Interrupt Rate** 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 ensure proper operation of the timer, STIR must not be changed during timer operation (that is, if TMIE = 1).
13	TMIE	0	**Timer Interrupt Enable** Enables/disables 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.

Table 9-2. SCI Control Register (SCR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
12	TIE	0	**SCI Transmit Interrupt Enable** Enables/disables 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.
11	RIE	0	**SCI Receive Interrupt Enable** Enables/disables the SCI receive data interrupt. If RIE is cleared, the receive data interrupt is disabled, and 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 (that is, 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.
10	ILIE	0	**Idle Line Interrupt Enable** 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 is 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.
9	TE	0	**Transmitter Enable** 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 (that is, 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 10 or 11 consecutive ones (depending on WDS), giving you a convenient way to ensure that the line goes idle before a new message starts. 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.

Table 9-2. SCI Control Register (SCR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
8	RE	0	**Receiver Enable** 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 completes 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.
7	WOMS	0	**Wired-OR Mode Select** When WOMS 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 multi-drop configuration. An external pull-up resistor is required on the bus. When WOMS is cleared, the TXD signal uses an active internal pull-up. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears WOMS.
6	RWU	0	**Receiver Wake-up Enable** When RWU is set and the SCI is in Asynchronous mode, the wake-up function is enabled; that is, 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 wake-up hardware. You can also clear the RWU bit to wake up the receiver. You can use RWU to ignore messages that are for other devices on a multi-drop serial network. Wake up on idle line (WAKE is cleared) or wake up on address bit (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.
5	WAKE	0	**Wake-up Mode Select** When WAKE is cleared, the wake up on Idle Line mode is selected and the SCI receiver is re-enabled by an idle string of at least 10 or 11 (depending on WDS mode) consecutive ones. The transmitter 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 wake up on address bit mode is selected and 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 multi-drop 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. Each processor must compare the received character with its own address and determine whether to receive or ignore all following characters.
4	SBK	0	**Send Break** A break is an all-zero word frame—a start bit 0, characters of all zeros (including any parity), and a stop bit 0 (that is, ten or eleven zeros, depending on the mode selected). If SBK is set and then cleared, the transmitter finishes transmitting the current frame, sends 10 or 11 0s, 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 end 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 signal an unusual condition, message, and so on, by forcing a frame error; the frame error is caused by a missing stop bit.

Table 9-2. SCI Control Register (SCR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="5">Description</th></tr>
<tr><td>3</td><td>SSFTD</td><td>0</td><td colspan="5"><b>SCI Shift Direction</b>
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 they remain adjacent to the stop bit.</td></tr>
<tr><td rowspan="10">2–0</td><td rowspan="10">WDS[2–0]</td><td rowspan="10">0</td><td colspan="5"><b>Word Select</b>
Select the format of transmitted and received data. Asynchronous modes are compatible with most UART-type serial devices, and they support standard RS-232 communication links. Multi-drop 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 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 ones 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 ones, 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.</td></tr>
<tr><td><b>WDS2</b></td><td><b>WDS1</b></td><td><b>WDS0</b></td><td><b>Mode</b></td><td><b>Word Formats</b></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>8-Bit Synchronous Data (shift register mode)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td><td>10-Bit Asynchronous (1 start, 8 data, 1 stop)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>3</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td><td>11-Bit Asynchronous
(1 start, 8 data, 1 even parity, 1 stop)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td><td>11-Bit Asynchronous
(1 start, 8 data, 1 odd parity, 1 stop)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td><td>11-Bit Multi-drop Asynchronous
(1 start, 8 data, 1 data type, 1 stop)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>7</td><td>Reserved</td></tr>
</table>

9.6.2 SCI Status Register (SSR)

The SSR is a read-only register that indicates the status of the SCI.

23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
▨	▨	▨	▨	▨	▨	▨	▨
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R8	FE	PE	OR	IDLE	RDRF	TDRE	TRNE

▨—Reserved bit; read as 0; write to 0 for future compatibility.

Table 9-3. SCI Status Register (SSR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23–8		0	Reserved. Write to 0 for future compatibility.
7	R8	0	**Received Bit 8** In 11-bit Asynchronous Multi-drop mode, the R8 bit indicates 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 RESET signal, a software RESET instruction, an SCI individual reset, or a STOP instruction clears R8.
6	FE	0	**Framing Error Flag** In Asynchronous mode, FE is set 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 a read of the SRX. A hardware 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.
5	PE	0	**Parity Error** In 11-bit Asynchronous modes, PE is set when an incorrect parity bit is detected in the received character. PE and RDRF are set simultaneously 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 RESET signal, a software RESET instruction, an SCI individual reset, or a STOP instruction also clears PE. In 10-bit Asynchronous mode, 11-bit multi-drop 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.
4	OR	0	**Overrun Error Flag** 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 RESET signal, a software RESET instruction, an SCI individual reset, or a STOP instruction clears OR.
3	IDLE	0	**Idle Line Flag** Set when 10 (or 11) consecutive ones 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).
2	RDRF	0	**Receive Data Register Full** 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.

Table 9-3. SCI Status Register (SSR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
1	TDRE	1	**Transmit Data Register Empty** 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.
0	TRNE	1	**Transmitter Empty** This flag bit is set when both the transmit shift register and transmit data register (STX) are empty, indicating 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 a write into STX or STXA clears TDRE or when an idle, preamble, or break is transmitted. When set, TRNE 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 being transmitted. This procedure is useful when initiating the transfer of a message (that is, a string of characters).

9.6.3 SCI Clock Control Register (SCCR)

The SCCR is a read/write register that controls the selection of clock modes and baud rates for the transmit and receive sections of the SCI interface. The SCCR is cleared by a hardware $\overline{\text{RESET}}$ signal.

23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
▨	▨	▨	▨	▨	▨	▨	▨
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TCM	RCM	SCP	COD	CD11	CD10	CD9	CD8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0

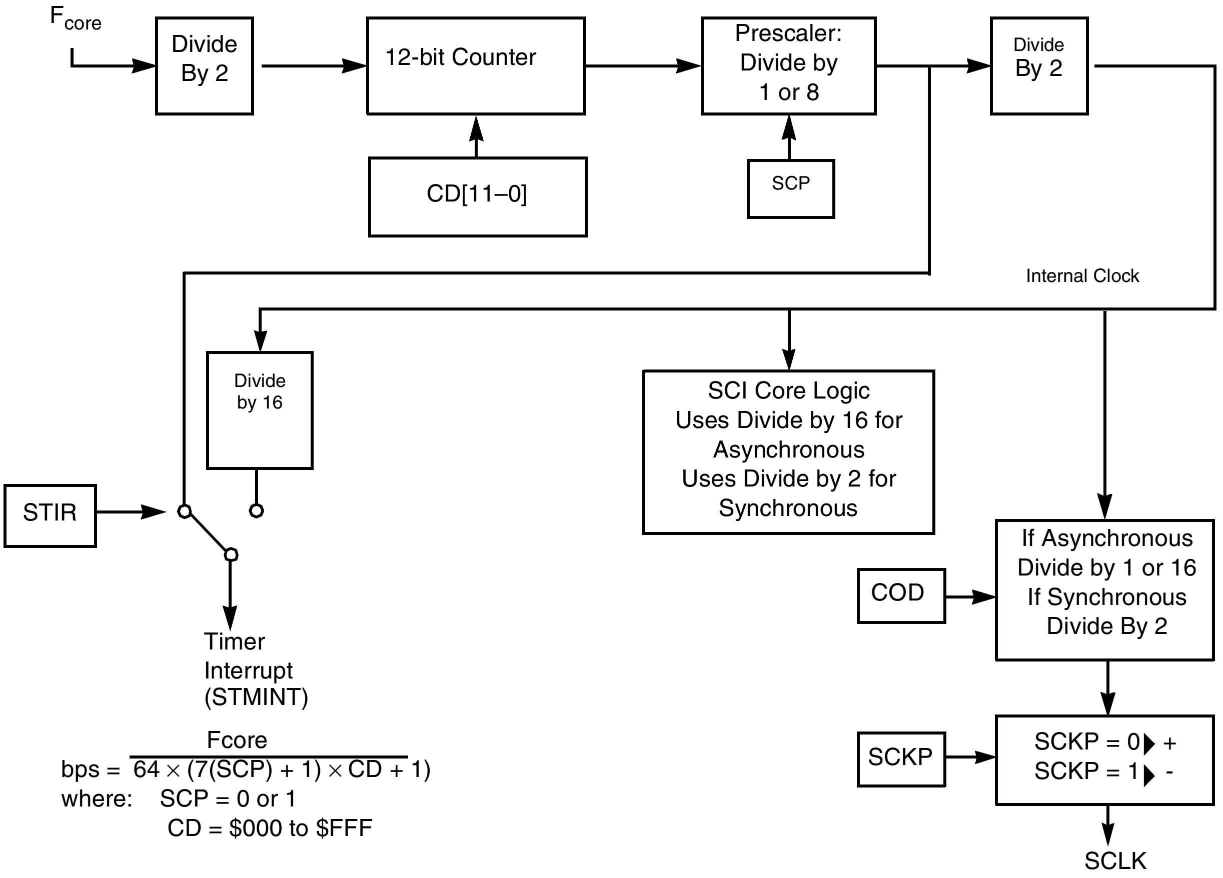
▨ Reserved. Read as 0. Write to 0 for future compatibility.

Table 9-4. SCI Clock Control Register (SCCR) Bit Definitions

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="6">Description</th></tr>
<tr><td>23–16</td><td></td><td>0</td><td colspan="6">Reserved. Write to 0 for future compatibility.</td></tr>
<tr><td>15</td><td>TCM</td><td>0</td><td colspan="6"><b>Transmit Clock Source</b>
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.</td></tr>
<tr><td rowspan="6">14</td><td rowspan="6">RCM</td><td rowspan="6">0</td><td colspan="6"><b>Receive Clock Mode Source</b>
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.</td></tr>
<tr><th>TCM</th><th>RCM</th><th>TX Clock</th><th>RX Clock</th><th>SCLK</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>Internal</td><td>Internal</td><td>Output</td><td>Synchronous/asynchronous</td></tr>
<tr><td>0</td><td>1</td><td>Internal</td><td>External</td><td>Input</td><td>Asynchronous only</td></tr>
<tr><td>1</td><td>0</td><td>External</td><td>Internal</td><td>Input</td><td>Asynchronous only</td></tr>
<tr><td>1</td><td>1</td><td>External</td><td>External</td><td>Input</td><td>Synchronous/asynchronous</td></tr>
<tr><td>13</td><td>SCP</td><td>0</td><td colspan="6"><b>Clock Prescaler</b>
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.</td></tr>
<tr><td>12</td><td>COD</td><td>0</td><td colspan="6"><b>Clock Out Divider</b>
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 (that is, 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.</td></tr>
<tr><td>11–0</td><td>CD[11–0]</td><td>0</td><td colspan="6"><b>Clock Divider</b>
Specifies 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.</td></tr>
</table>

The SCI clock determines the data transmission (baud) rate and can also establish a periodic interrupt that can act as an event timer or be used in any other timing function. Bits CD11– CD0, SCP, and SCR[STIR] work together to determine the time base. If SCR[TMIE] = 1 when the periodic time-out occurs, the SCI timer interrupt is recognized and pending. The SCI timer interrupt is automatically cleared when the interrupt is serviced. This interrupt occurs every time the periodic timer times out.

Figure 9-3 shows the block diagram of the internal clock generation circuitry with the formula to compute the bit rate when the internal clock is used.



F core
Divide By 2
12-bit Counter
Prescaler: Divide by 1 or 8
Divide By 2
CD[11–0]
SCP
Internal Clock
Divide by 16
SCI Core Logic Uses Divide by 16 for Asynchronous Uses Divide by 2 for Synchronous
STIR
Timer Interrupt (STMINT)
If Asynchronous Divide by 1 or 16 If Synchronous Divide By 2
COD
SCKP
SCKP = 0 +
SCKP = 1 -
SCLK
bps = Fcore / 64 × (7(SCP) + 1) × CD + 1)
where: SCP = 0 or 1
CD = $000 to $FFF

Figure 9-3. SCI Baud Rate Generator

As noted in **Section 9.6.1**, the SCI can be configured to operate in a single Synchronous mode or one of five Asynchronous modes. Synchronous mode requires that the TX and RX clocks use the same source, but that source may be the internal SCI clock if the SCI is configured as a master device or an external clock if the SCI is configured as a slave device. Asynchronous modes may use clocks from the same source (internal or external) or different sources for the TX clock and the RX clock.

For synchronous operation, the SCI uses a clock that is equal to the two times the desired bit rate (designated as the 2 × clock) for both internal and external clock sources. It must use the same source for both the TX and RX clock. The internal clock is used if the SCI is the master device and the external clock is used if the SCI is the slave device, as noted above. The clock is gated and limited to a maximum frequency equal to one eighth of the DSP core operating frequency (that is, 12.5 MHz for a DSP core frequency of 100 MHz).

For asynchronous operation, the SCI can use the internal and external clocks in any combination as the source clocks for the TX clock and RX clock. If an external clock is used for the SCLK input, it must be sixteen times the desired bit rate (designated as the 16 × clock), as indicated in

Figure 9-4. When the internal clock is used to supply a clock to an external device, the clock can use the actual bit rate (designated as the 1 × clock) or the 16 × clock rate, as determined by the COD bit. The output clock is continuous.

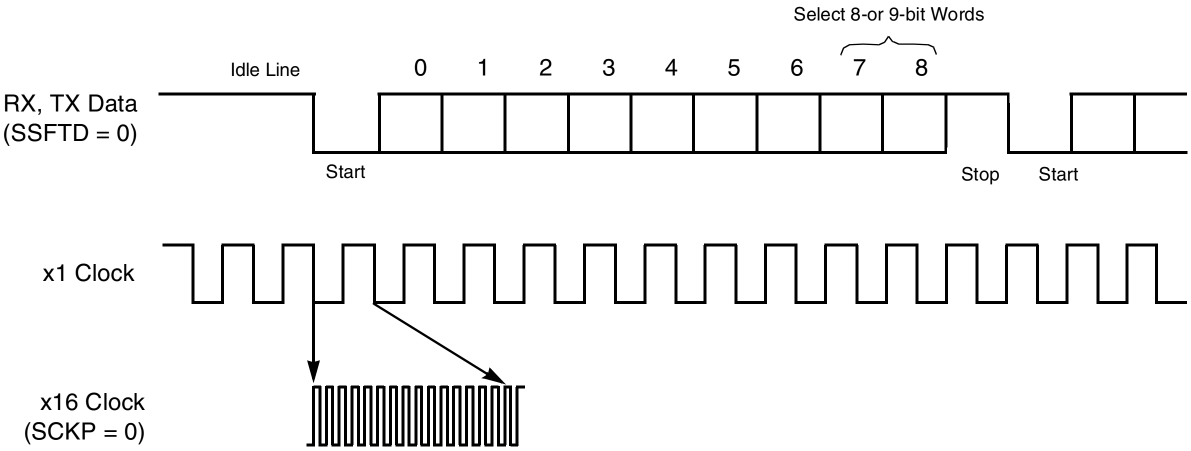
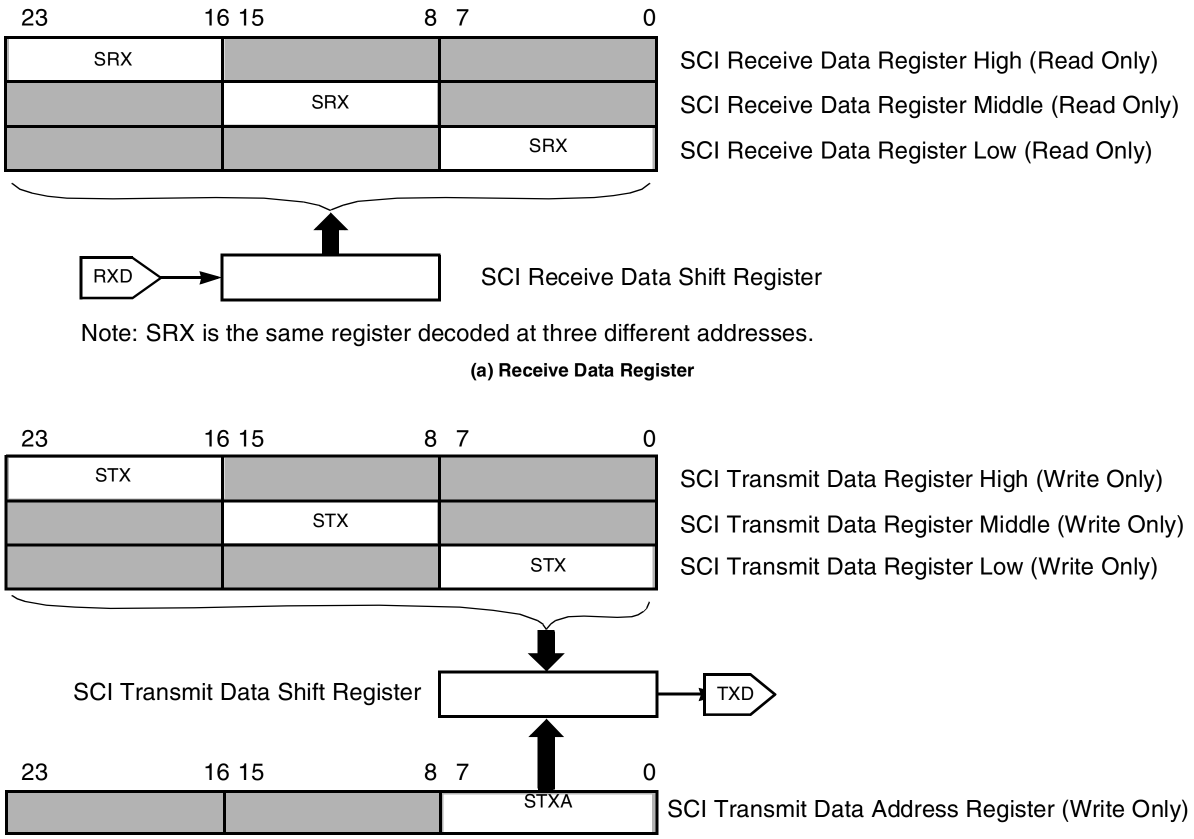


Figure 9-4. 16 x Serial Clock

When SCKP is cleared, the transmitted data on the TXD signal changes on the negative edge of the 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 serial clock.

9.6.4 SCI Data Registers

The SCI data registers are divided into two groups: receive and transmit, as shown in **Figure 9-1**. 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
SRX
SRX
SCI Receive Data Register High (Read Only)
SCI Receive Data Register Middle (Read Only)
SCI Receive Data Register Low (Read Only)
RXD
SCI Receive Data Shift Register
23
16 15
8 7
0
STX
STX
STX
SCI Transmit Data Register High (Write Only)
SCI Transmit Data Register Middle (Write Only)
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: SRX is the same register decoded at three different addresses.

(a) Receive Data Register

Notes: **1.** Bytes are masked on the fly.

2. STX is the same register decoded at four different addresses.

(b) Transmit Data Register

Figure 9-5. SCI Programming Model—Data Registers

9.6.4.1 SCI Receive Register (SRX)

Data bits received on the RXD signal are shifted into the SCI receive shift register. When a complete word is received, the data portion of the word is transferred to the byte-wide SRX. This process converts serial data to parallel data and provides double buffering. Double buffering promotes flexibility 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 zeros. Similarly, when SRXM is read, the contents of SRX are placed into the middle byte of the bus, and when SRXH is read, the contents of SRX are placed into 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 ORing three data bytes read from the three addresses.

The SCR WDS0, WDS1, and WDS2 control bits define the length and format of the serial word. The SCR receive clock mode (RCM) defines the clock source.

In Asynchronous mode, the start bit, the eight data bits, the address/data indicator bit or the parity bit, and the stop bit are received, respectively. 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 is 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.

9.6.4.2 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 multi-drop 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 multi-drop 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, because the second byte might overwrite the first byte. Thus, you should always poll the TDRE flag prior to writing STX or STXA to prevent overruns unless transmit interrupts are 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 SSR[TRNE] bit.

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.

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*.

9.7 GPIO Signals and Registers

Three registers control the GPIO functionality of the SCI pins: Port E control register (PCRE), Port E direction register (PRRE) and Port E data register (PDRE).

9.7.1 Port E Control Register (PCRE)

The read/write PCRE controls the functionality of SCI GPIO signals. Each of the PCRE[2–0] bits controls the functionality of the corresponding port signal. When a PCRE[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. A hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PCRE bits.

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

11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	PE2/ SCLK	PE1/ TXD	PE0/ RXD

Note: For bits 2–0, a 0 selects PEn as the signal and a 1 selects the specified SCI signal.

▨ = Reserved. Read as zero. Write to zero for future compatibility.

9.7.2 Port E Direction Register (PRRE)

The read/write PRRE controls the direction of SCI GPIO signals. When port signal[i] is configured as GPIO, PRRE[i] controls the port signal direction. When PRRE[i] is set, the GPIO port signal[i] is configured as output. When PRRE[i] is cleared, the GPIO port signal[i] is configured as input. A hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PRRE bits

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
▨	▨	▨	▨	▨	▨	▨	▨	▨	PRRE2	PRRE1	PRRE0

Note: For bits 2–0, a 0 configures PEn as a GPI and a 1 configures PEn as a GPO. For the SCI, the GPIO signals are PE[2–0]. The corresponding direction bits for Port E GPIOs are PRRE[2–0].

▨ = Reserved. Read as zero. Write with zero for future compatibility.

9.7.3 Port E Data Register (PDRE)

Bits 2–0 of the read/write 24-bit PDRE writes data to or reads data from the associated SCI signal lines when configured as GPIO signals. If a port signal PE[i] is configured as an input (GPI), the corresponding PDRE[i] bit reflects the value present on the input signal line. If a port signal PE[i] is configured as an output (GPO), a value written to the corresponding PDRE[i] bit is reflected as a value on the output signal line. Either a hardware $\overline{\text{RESET}}$ signal or a software RESET instruction clears all PDR bits.

23	22	21	20	19	18	17	16	15	14	13	12
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
▨	▨	▨	▨	▨	▨	▨	▨	▨	PDRE2	PDRE1	PDRE0

Note: For bits 2–0, the value represents the level that is written to or read from the associated signal line if enabled as a GPIO signal by the PCRE bits. For SCI, the GPIO signals are PE[2–0]. The corresponding data bits are PDRE[2–0].

▨ = Reserved. Read as zero. Write with zero for future compatibility.

Triple Timer Module 10

The timers in the DSP56321 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.

10.1 Overview

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 has the following capabilities:

- Uses internal or external clocking
- Interrupts the DSP56321 after a specified number of events (clocks) or signals an external device after counting internal events
- Triggers DMA transfers after a specified number of events (clocks) occurs
- Connects to the external world through one bidirectional signal, designated TIO[0– 2] for timers 0–2.

When TIO is configured as an input, the timer functions as an external event counter or measures external pulse width/signal period. When TIO is configured as an output, the timer functions as a timer, a watchdog timer, or a pulse-width modulator. When the timer does not use TIO, it can be used as a GPIO signal (also called TIO[0–2]).

10.1.1 Triple Timer Module Block Diagram

Figure 10-1 shows a block diagram of the triple timer module. This module includes a 24-bit Timer Prescaler Load Register (TPLR), a 24-bit Timer Prescaler Count Register (TPCR), and three timers. Each timer can use the prescaler clock as its clock source.

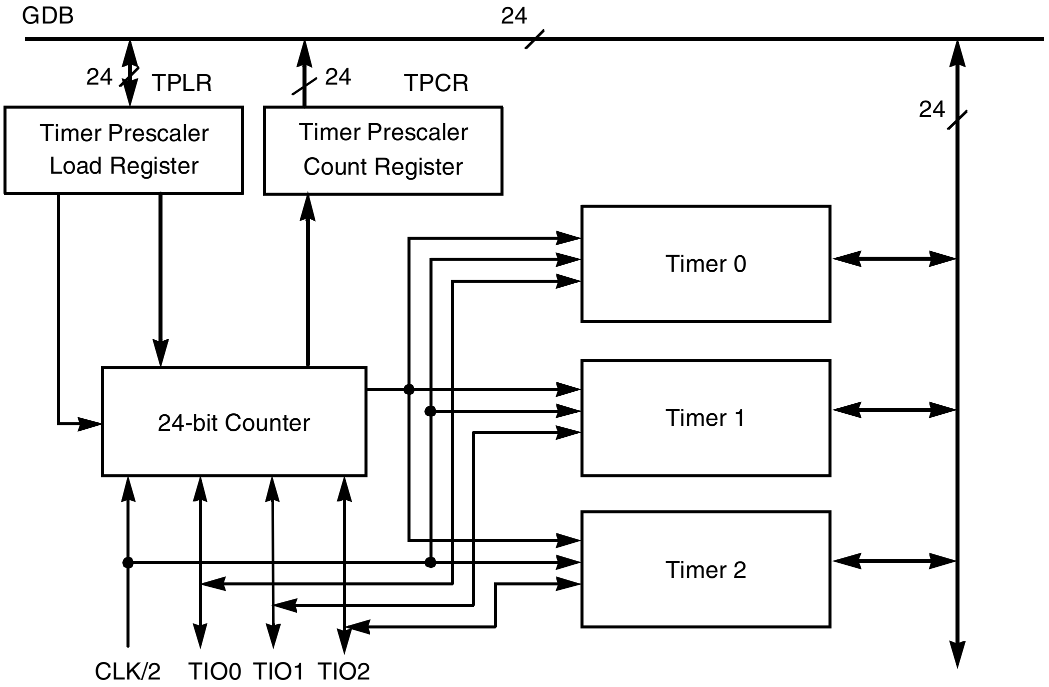


Figure 10-1. Triple Timer Module Block Diagram

10.1.2 Individual Timer Block Diagram

Figure 10-2 shows the structure of an individual timer block. The DSP56321 treats each timer as a memory-mapped peripheral with four registers occupying four 24-bit words in the X data memory space. The three timers are identical in structure and function. Either standard polled or interrupt programming techniques can be used to service the timers. A single, generic timer is discussed in this chapter. Each timer includes the following:

- 24-bit counter
- 24-bit read/write Timer Control and Status Register (TCSR)
- 24-bit read-only Timer Count Register (TCR)
- 24-bit write-only Timer Load Register (TLR)
- 24-bit read/write Timer Compare Register (TCPR)
- Logic for clock selection and interrupt/DMA trigger generation.

The timer mode is controlled by the TC[3–0] bits which are TCSR[7–4]. For a listing of the timer modes and descriptions of their operations, see **Section 10.3**, *Operating Modes*, on page 10-5.

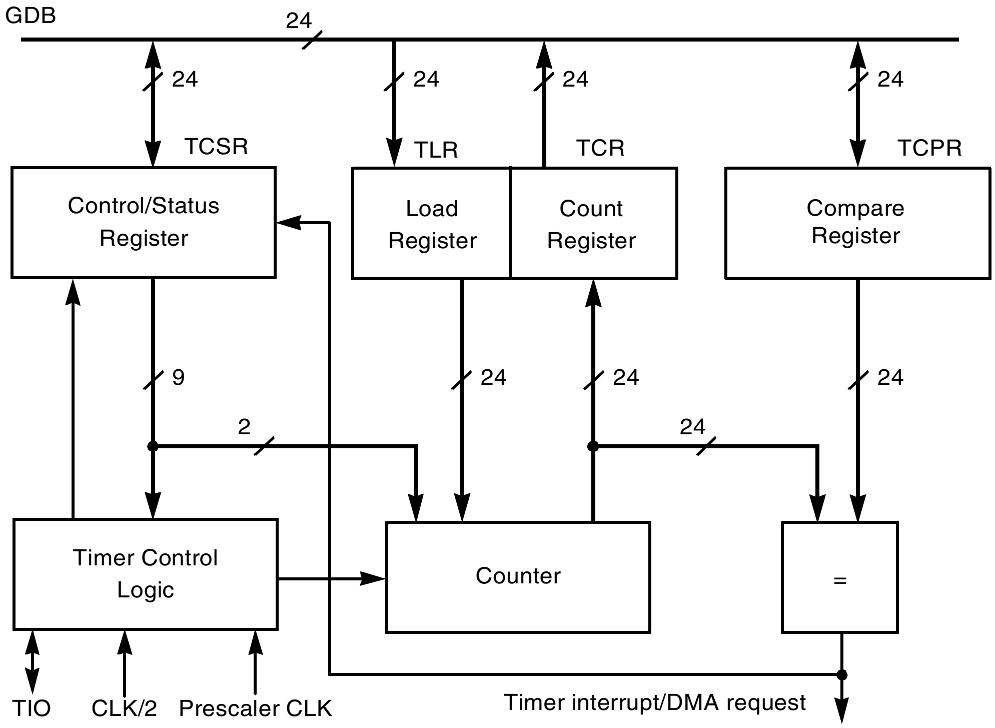


Figure 10-2. Timer Module Block Diagram

10.2 Operation

This section discusses the following timer basics: reset, initialization, and exceptions.

10.2.1 Timer After Reset

A hardware $\overline{\text{RESET}}$ signal or software RESET instruction clears the Timer Control and Status Register for each timer, thus configuring each timer as a GPIO. A timer is active only if the timer enable bit 0 (TCSR[TE]) in the specific timer TCSR is set.

10.2.2 Timer Initialization

To initialize a timer, do the following:

1. Ensure that the timer is not active either by sending a reset or clearing the TCSR[TE] bit.
2. Configure the control register (TCSR) to set the timer operating mode. Set the interrupt enable bits as needed for the application.
3. Configure other registers: Timer Prescaler Load Register (TPLR), Timer Load Register (TLR), and Timer Compare Register (TCPR) as needed for the application.
4. Enable the timer by setting the TCSR[TE] bit.

10.2.3 Timer Exceptions

Each timer can generate two different exceptions:

- Timer Overflow (highest priority) — Occurs when the timer counter reaches the overflow value. This exception sets the TOF bit. TOF is cleared when a value of one is written to it or when the timer overflow exception is serviced.
- Timer Compare (lowest priority) — Occurs when the timer counter reaches the value given in the Timer Compare Register (TCPR) for all modes except measurement modes. In measurement modes 4–6, a compare exception occurs when the appropriate transition occurs on the TIO signal. The Compare exception sets the TCF bit. TCF is cleared when a value of one is written to it or when the timer compare interrupt is serviced.

To configure a timer exception, perform the following steps. The example at the right of each step shows the register settings for configuring a Timer 0 compare interrupt. The order of the steps is optional except that the timer should not be enabled (step 2e) until all other exception configuration is complete:

1. Configure the 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:TIM0C`
2. Configure the interrupt trigger:
   - **a.** Enable and prioritize overall peripheral interrupt functionality.
     `IPRP (TOL[1-0])`
   - **b.** Enable a specific peripheral interrupt.
     `TCSR0 (TCIE)`
   - **c.** Unmask interrupts at the global level.
     `SR (I[1-0])`
   - **d.** Configure a peripheral interrupt-generating function.
     `TCSR0 (TC[7-4])`
   - **e.** Enable peripheral and associated signals. `TCSR0 (TE)`

10.3 Operating Modes

Each timer has operating 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 period, 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 ensure proper operation, the TCSR TC[3–0] bits should be changed only when the timer is disabled (that is, when TCSR[TE] is cleared).

10.3.1 Triple Timer Modes

For all triple timer modes, the following points are true:

- The TCSR[TE] bit is set to clear the counter and enable the timer. Clearing TCSR[TE] disables the timer.
- The value to which the timer is to count is loaded into the TCPR. (This is true for all modes except the measurement modes (modes 4 through 6).
- The counter is loaded with the TLR value on the first clock.
- If the counter overflows, TCSR[TOF] is set, and if TCSR[TOIE] is set, an overflow interrupt is generated.
- You can read the counter contents at any time from the Timer Count Register (TCR).

10.3.1.1 Timer GPIO (Mode 0)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	0	0	0	0	GPIO	Timer	GPIO	Internal

In Mode 0, the timer generates an internal interrupt when a counter value is reached, if the timer compare interrupt is enabled (see **Figure 10-3** and **Figure 10-4**). When the counter equals the

TCPR value, TCSR[TCF] is set and a compare interrupt is generated if the TCSR[TCIE] bit is set. If the TCSR[TRM] bit is set, the counter is reloaded with the TLR value at the next timer clock and the count is resumed. If TCSR[TRM] is cleared, the counter continues to increment on each timer clock signal. This process repeats until the timer is disabled.

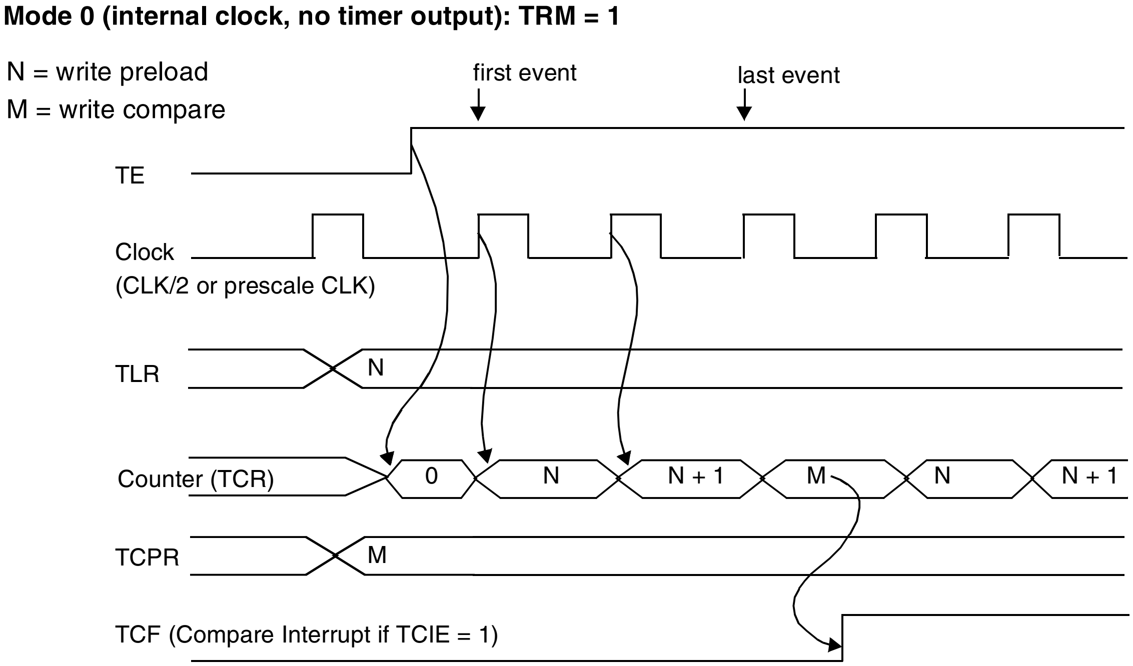


Figure 10-3. Timer Mode (TRM = 1)

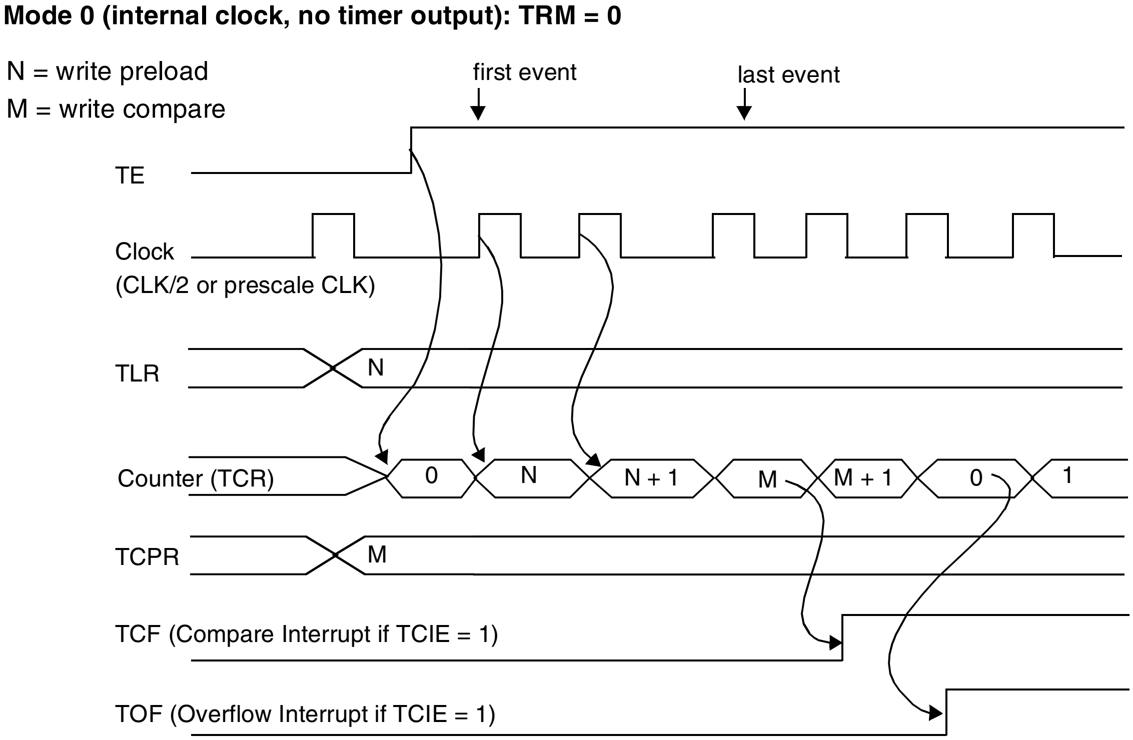


Figure 10-4. Timer Mode (TRM = 0)

10.3.1.2 Timer Pulse (Mode 1)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	0	0	1	1	Timer Pulse	Timer	Output	Internal

In Mode 1, the timer generates an external pulse on its TIO signal when the timer count reaches a pre-set value. The TIO signal is loaded with the value of the TCSR[INV] bit. When the counter matches the TCPR value, TCSR[TCF] is set and a compare interrupt is generated if the TCSR[TCIE] bit is set. The polarity of the TIO signal is inverted for one timer clock period. If TCSR[TRM] is set, the counter is loaded with the TLR value on the next timer clock and the count is resumed. If TCSR[TRM] is cleared, the counter continues to increment on each timer clock. This process repeats until TCSR[TE] is cleared (disabling the timer).

The TLR value in the TCPR sets the delay between starting the timer and generating the output pulse. To generate successive output pulses with a delay of X clock cycles between signals, set the TLR value to X/2 and set the TCSR[TRM] bit. This process repeats until the timer is disabled.

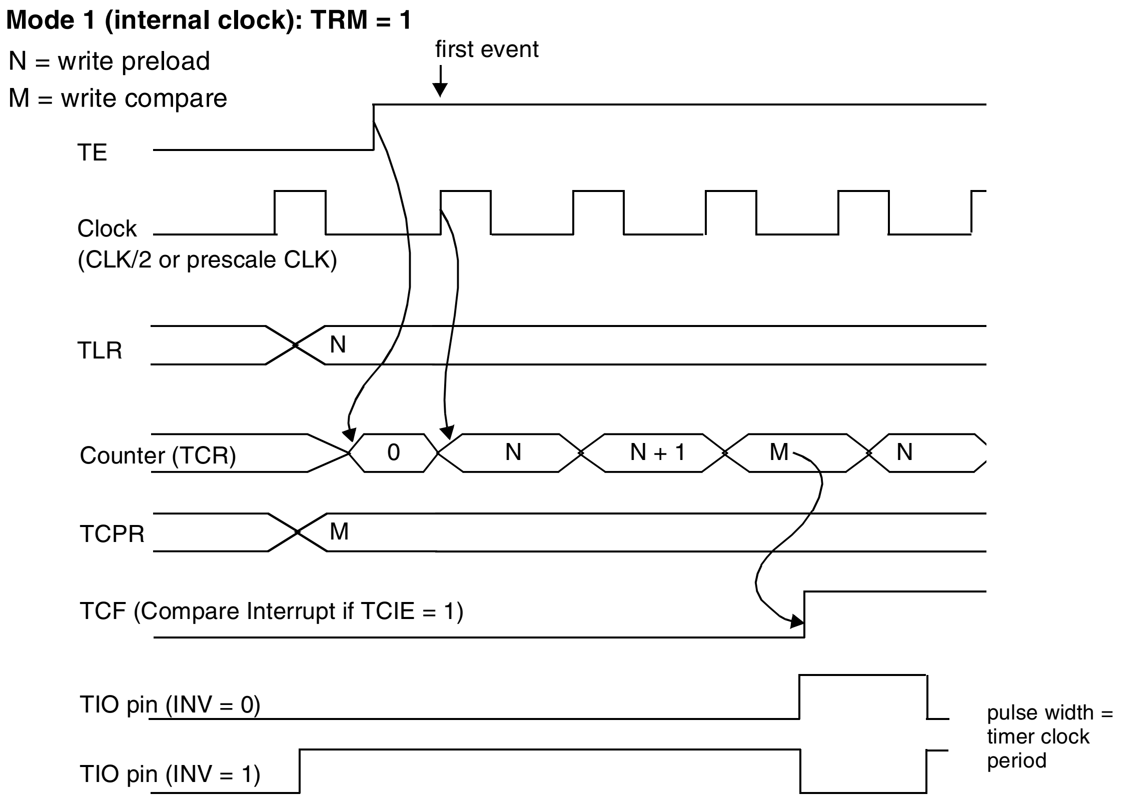


Figure 10-5. Pulse Mode (TRM = 1)

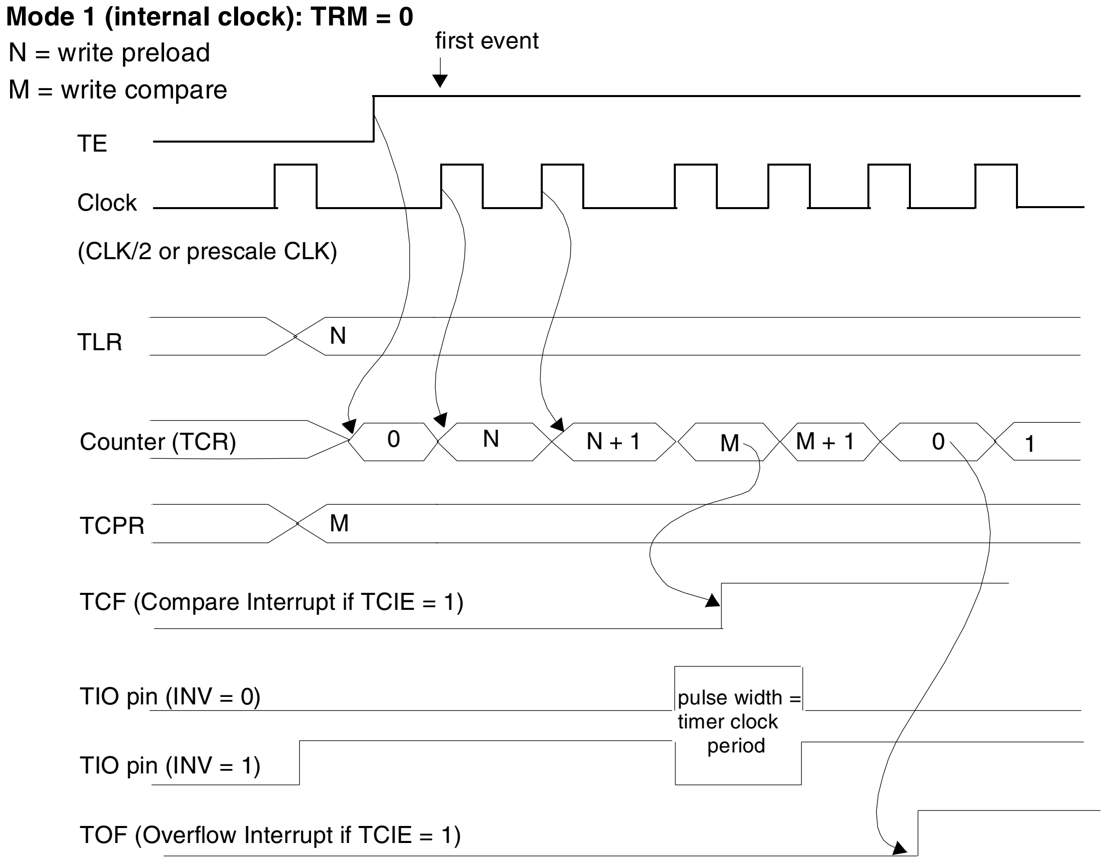


Figure 10-6. Pulse Mode (TRM = 0)

10.3.1.3 Timer Toggle (Mode 2)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	0	1	0	2	Toggle	Timer	Output	Internal

In Mode 2, the timer periodically toggles the polarity of the TIO signal. When the timer is enabled, the TIO signal is loaded with the value of the TCSR[INV] bit. When the counter value matches the value in the TCPR, the polarity of the TIO output signal is inverted. TCSR[TCF] is set, and a compare interrupt is generated if the TCSR[TCIE] bit is set. If the TCSR[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 repeats until the timer is cleared (disabling the timer). The TCPR[TLR] value 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 TCSR[TRM] bit. This process repeats until the timer is disabled (that is, TCSR[TE] is cleared).

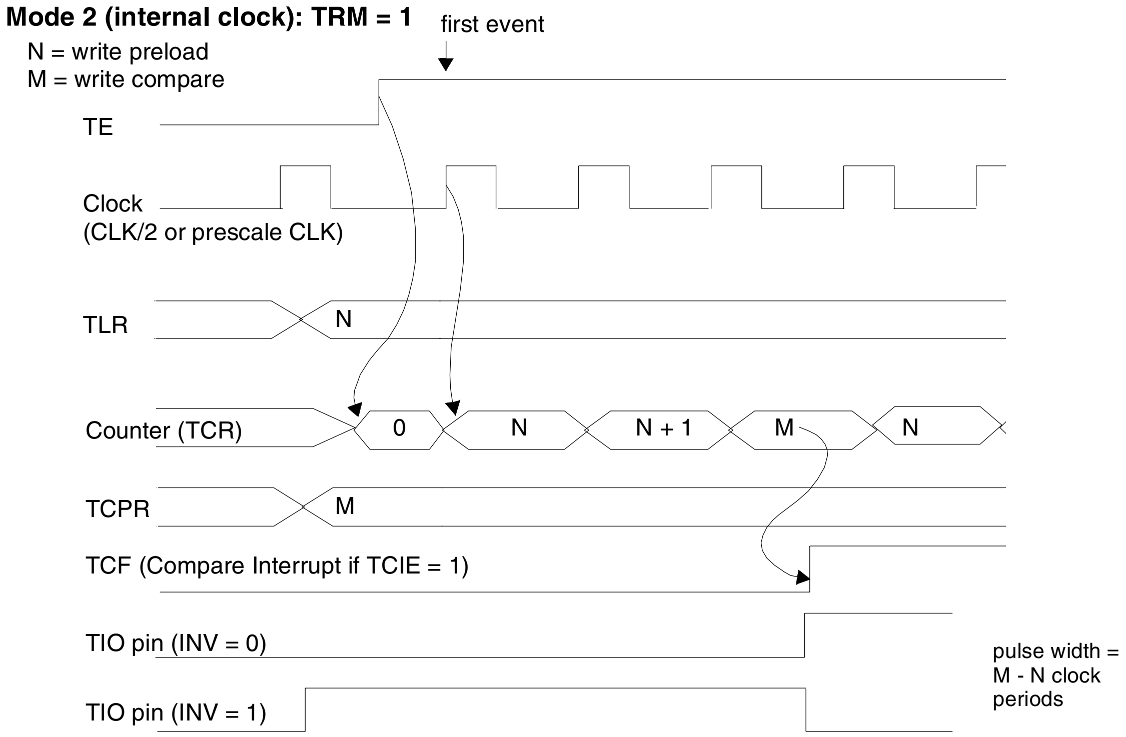


Figure 10-7. Toggle Mode, TRM = 1

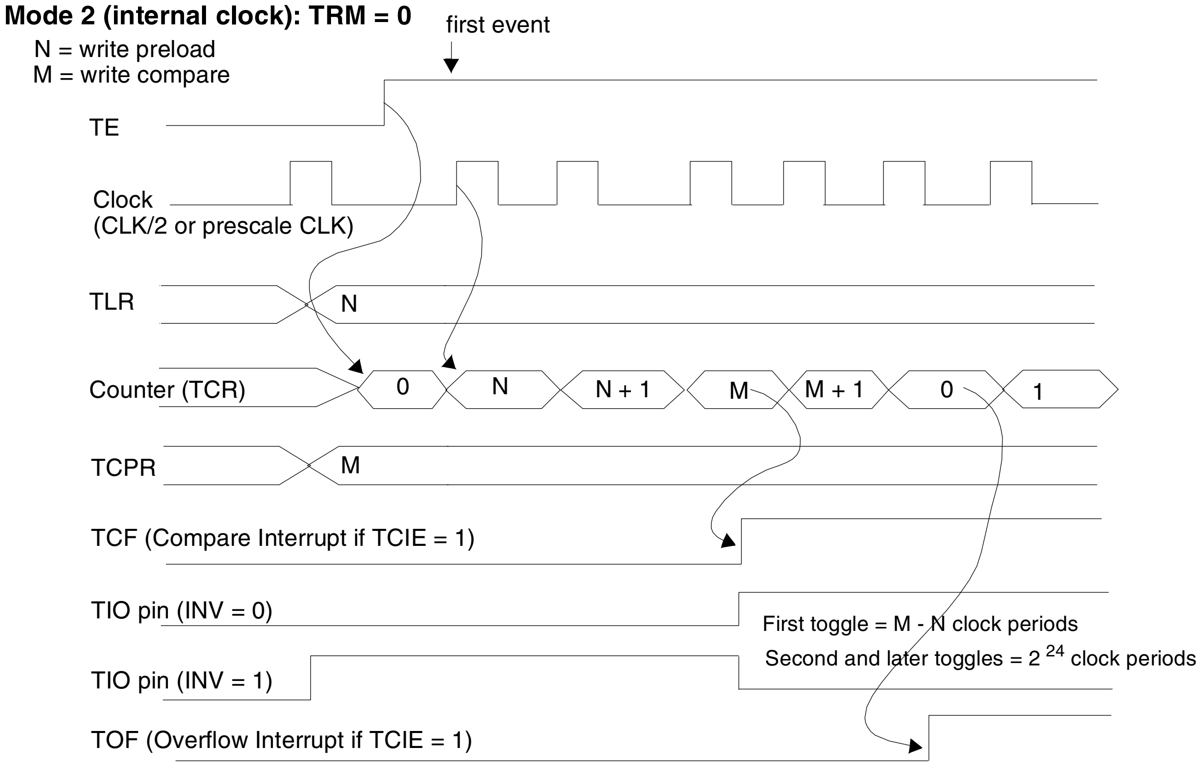

Mode 2 (internal clock): TRM = 0
N = write preload
M = write compare
first event
TE
Clock
(CLK/2 or prescale CLK)
TLR
N
Counter (TCR)
0
N
N + 1
M
M + 1
0
1
TCPR
M
TCF (Compare Interrupt if TCIE = 1)
TIO pin (INV = 0)
First toggle = M - N clock periods
Second and later toggles = 2 24 clock periods
TIO pin (INV = 1)
TOF (Overflow Interrupt if TCIE = 1)

Figure 10-8. Toggle Mode, TRM = 0

10.3.1.4 Timer Event Counter (Mode 3)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	0	1	1	3	Event Counter	Timer	Input	External

In Mode 3, the timer counts external events and issues an interrupt (if interrupt enable bits are set) when the timer counts a preset number of events. The timer clock signal can be taken from either the TIO input signal or the prescaler clock output. If an external clock is used, it is synchronized internally to the internal clock, and its frequency must be less than the DSP56321 internal operating frequency divided by 4. The value of the TCSR[INV] bit 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, TCSR[TCF] is set and a compare interrupt is generated if the TCSR[TCIE] bit is set. If the TCSR[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 TCSR[TRM] bit is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is disabled

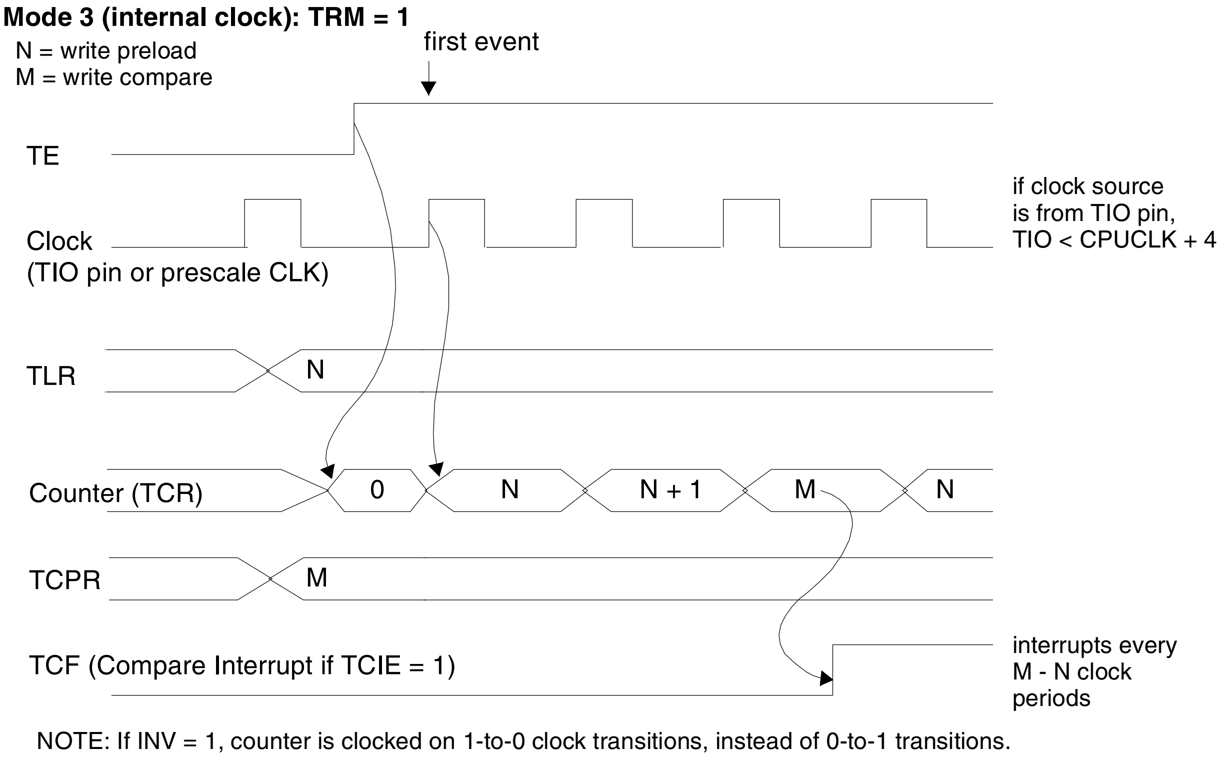
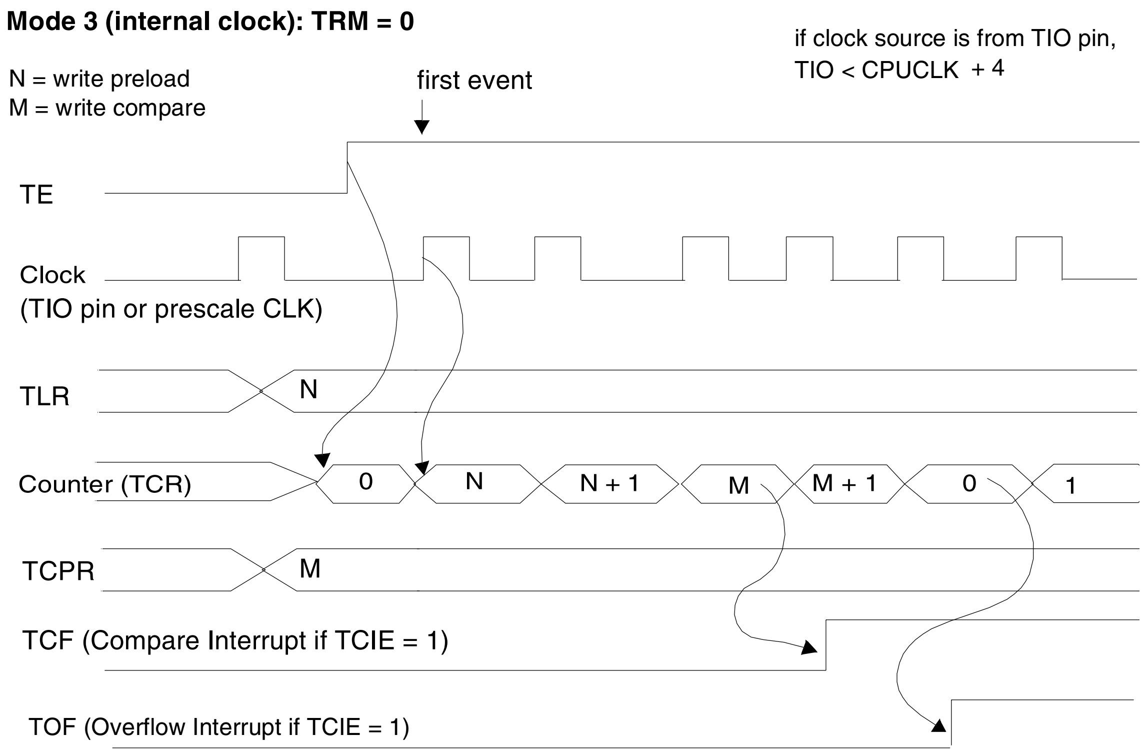

Mode 3 (internal clock): TRM = 1
N = write preload
M = write compare
first event
TE
Clock
(TIO pin or prescale CLK)
if clock source
is from TIO pin,
TIO < CPUCLK + 4
TLR
N
Counter (TCR)
0
N
N + 1
M
N
TCPR
M
TCF (Compare Interrupt if TCIE = 1)
interrupts every
M - N clock
periods
NOTE: If INV = 1, counter is clocked on 1-to-0 clock transitions, instead of 0-to-1 transitions.

Figure 10-9. Event Counter Mode, TRM = 1


Mode 3 (internal clock): TRM = 0
N = write preload
M = write compare
first event
if clock source is from TIO pin,
TIO < CPUCLK + 4
TE
Clock
(TIO pin or prescale CLK)
TLR
N
Counter (TCR)
0
N
N + 1
M
M + 1
0
1
TCPR
M
TCF (Compare Interrupt if TCIE = 1)
TOF (Overflow Interrupt if TCIE = 1)

NOTE: If INV = 1, counter is clocked on 1-to-0 clock transitions, instead of 0-to-1 transitions.

Figure 10-10. Event Counter Mode, TRM = 0

10.3.2 Signal Measurement Modes

The following signal measurement and pulse width modulation modes are provided:

- Measurement input width (Mode 4)
- Measurement input period (Mode 5)
- Measurement capture (Mode 6)
- Pulse width modulation (PWM) mode (Mode 7)

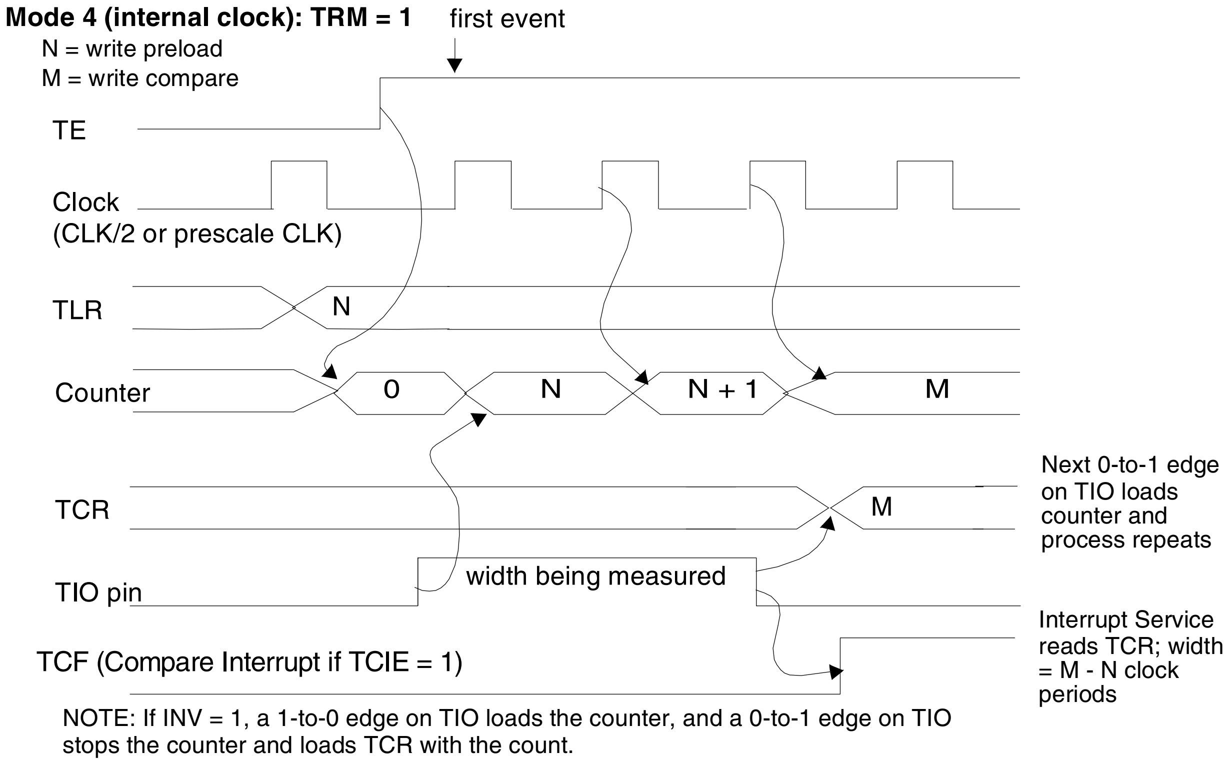
The external signal synchronizes with the internal clock that increments the counter. This synchronization process can cause the number of clocks measured for the selected signal value to vary from the actual signal value by plus or minus one counter clock cycle.

10.3.2.1 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 Mode 4, the timer counts the number of clocks that occur between opposite edges of an input signal. After the first appropriate transition (as determined by the TCSR[INV] bit) occurs on the

TIO input signal, the counter is loaded with the TLR value. If TCSR[INV] 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 (that is, 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. TCSR[TCF] is set and a compare interrupt is generated if the TCSR[TCIE] bit is set. The value of the counter (which measures the width of the TIO pulse) is loaded into the TCR, which can be read to determine the external signal pulse width. If the TCSR[TRM] bit is set, the counter is loaded with the TLR value on the first timer clock received following the next valid transition on the TIO input signal, and the count resumes. If TCSR[TRM] is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is disabled.



Next 0-to-1 edge on TIO loads counter and process repeats

Interrupt Service reads TCR; width = M - N clock periods

NOTE: If INV = 1, a 1-to-0 edge on TIO loads the counter, and a 0-to-1 edge on TIO stops the counter and loads TCR with the count.

Figure 10-11. Pulse Width Measurement Mode, TRM = 1

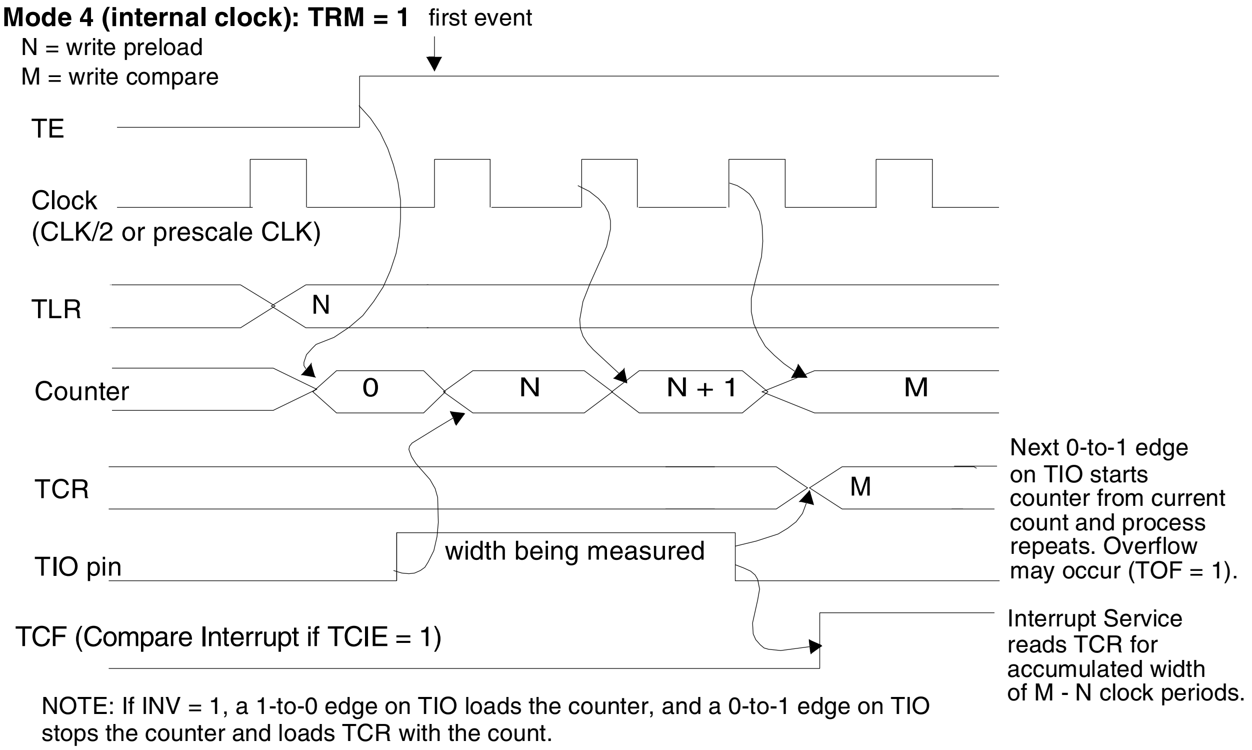
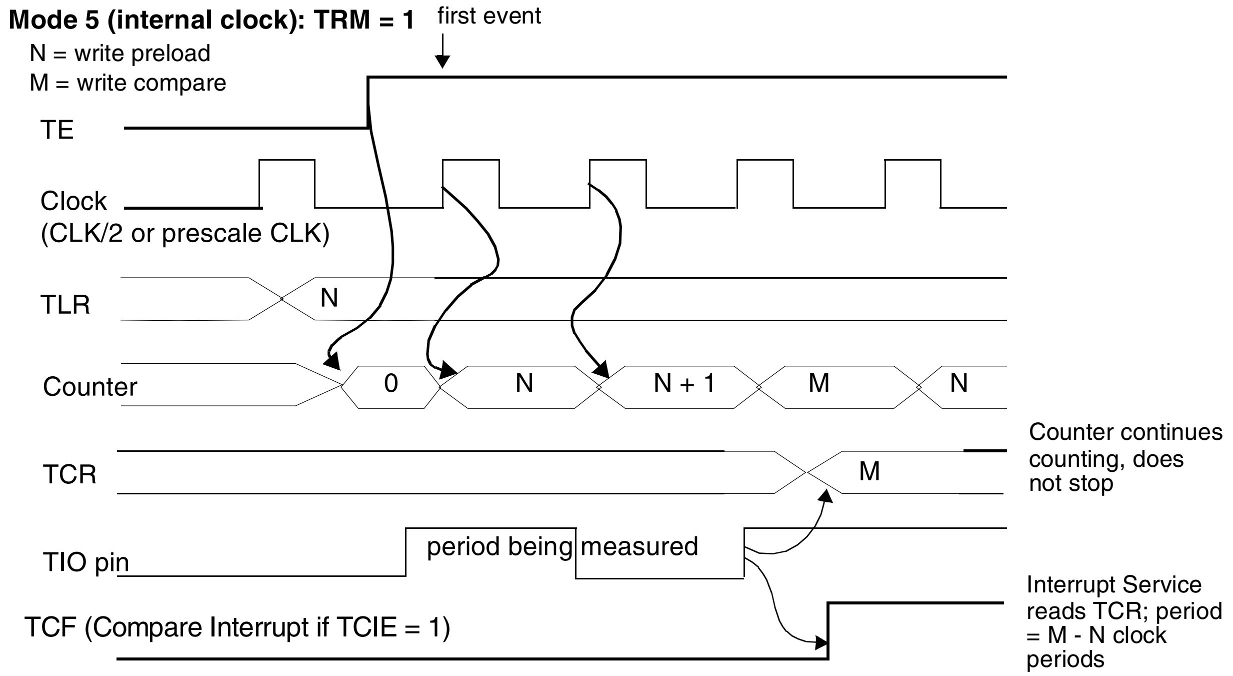

Mode 4 (internal clock): TRM = 1
N = write preload
M = write compare
first event
TE
Clock
(CLK/2 or prescale CLK)
TLR
N
Counter
0
N
N + 1
M
TCR
M
TIO pin
width being measured
TCF (Compare Interrupt if TCIE = 1)
Next 0-to-1 edge on TIO starts counter from current count and process repeats. Overflow may occur (TOF = 1).
Interrupt Service reads TCR for accumulated width of M - N clock periods.
NOTE: If INV = 1, a 1-to-0 edge on TIO loads the counter, and a 0-to-1 edge on TIO stops the counter and loads TCR with the count.

Figure 10-12. Pulse Width Measurement Mode, TRM = 0

10.3.2.2 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 Mode 5, the timer counts the period between the reception of signal edges of the same polarity across the TIO signal. 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 next signal transition of the same polarity that occurs on TIO, TCSR[TCF] is set, and a compare interrupt is generated if the TCSR[TCIE] bit is set. The contents of the counter load 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 TCSR[TRM] bit is set, the TCSR[TE] bit is set to clear the counter and enable the timer. The counter is repeatedly loaded and incremented until the timer is disabled. If the TCSR[TRM] bit is cleared, the counter continues to increment until it overflows.

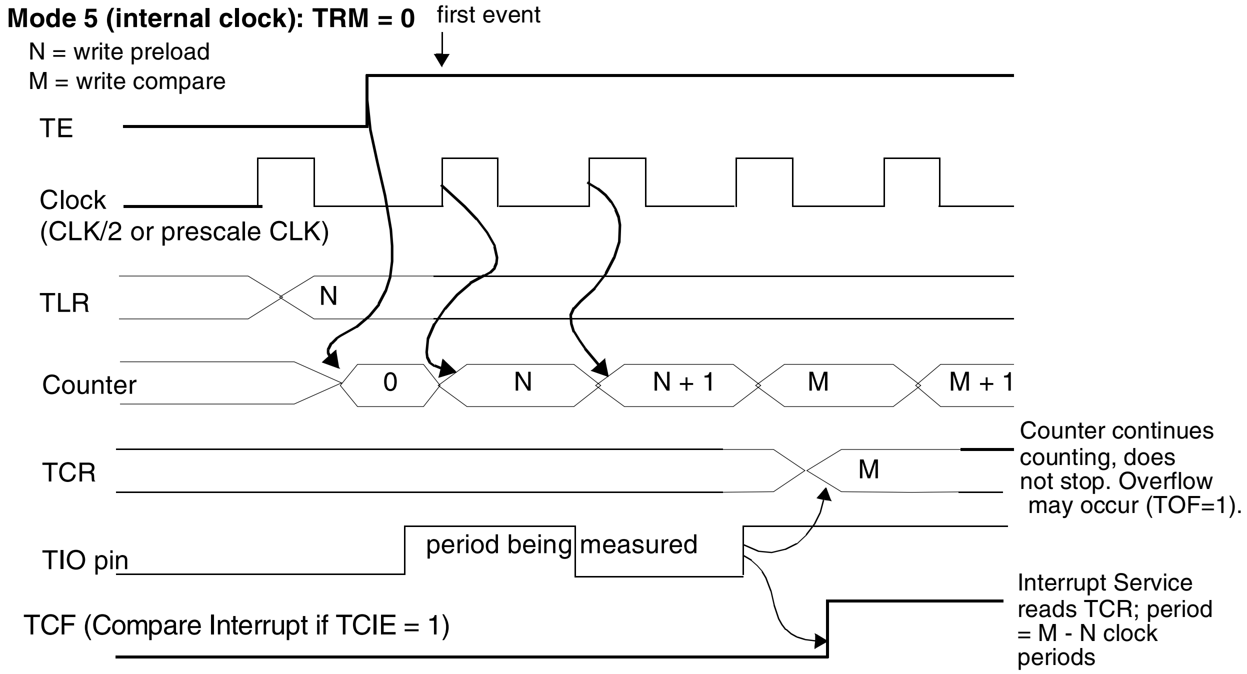


Counter continues counting, does not stop

Interrupt Service reads TCR; period = M - N clock periods

NOTE: If INV = 1, a 1-to-0 edge on TIO loads the counter, and a 0-to-1 edge on TIO loads TCR with count and the counter with N.

Figure 10-13. Period Measurement Mode, TRM = 1



Counter continues counting, does not stop. Overflow may occur (TOF=1).

Interrupt Service reads TCR; period = M - N clock periods

NOTE: If INV = 1, a 1-to-0 edge on TIO loads the counter, and a 0-to-1 edge on TIO loads TCR with count and the counter with N.

Figure 10-14. Period Measurement Mode, TRM = 0

10.3.2.3 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 Mode 6, the timer counts the number of clocks that elapse between when the timer starts and when an external signal is received. At the first appropriate transition of the external clock detected on the TIO signal, TCSR[TCF] is set and, if the TCSR[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 TCSR[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.

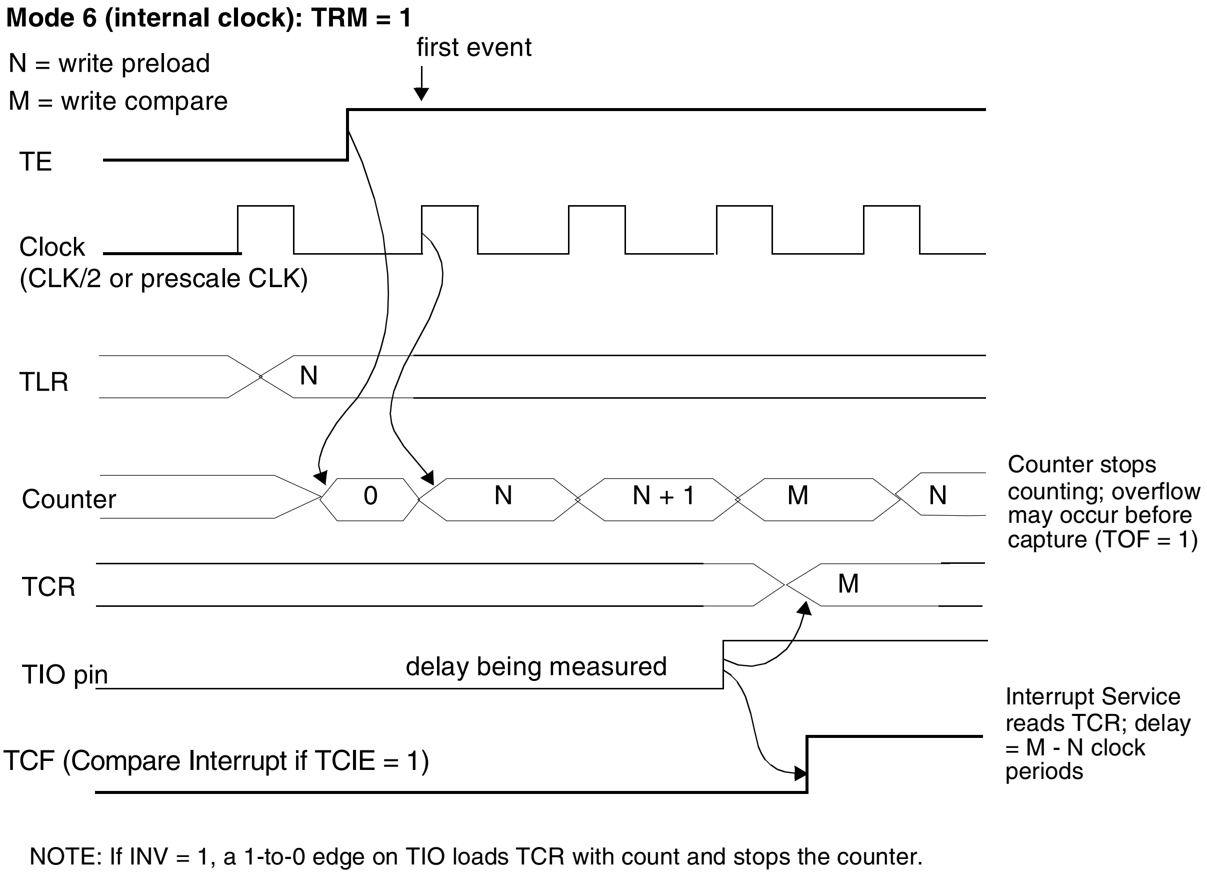

Mode 6 (internal clock): TRM = 1
N = write preload
M = write compare
first event
TE
Clock
(CLK/2 or prescale CLK)
TLR
N
Counter
0
N
N + 1
M
N
Counter stops counting; overflow may occur before capture (TOF = 1)
TCR
M
TIO pin
delay being measured
TCF (Compare Interrupt if TCIE = 1)
Interrupt Service reads TCR; delay = M - N clock periods
NOTE: If INV = 1, a 1-to-0 edge on TIO loads TCR with count and stops the counter.

Figure 10-15. Capture Measurement Mode, TRM = 0

10.3.3 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 Mode 7, the timer generates periodic pulses of a preset width. When the counter equals the value in the TCPR, the TIO output signal is toggled and TCSR[TCF] is set. The contents of the counter are placed into the TCR. If the TCSR[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, TCSR[TOF] is set, and an overflow interrupt is generated if the TCSR[TOIE] bit is set. If the TCSR[TRM] bit is set, the counter is loaded with the TLR value on the next timer clock and the count resumes. If the TCSR[TRM] bit is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is disabled.

When the TCSR[TE] bit is set and the counter starts, the TIO signal assumes the value of INV. 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.

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.

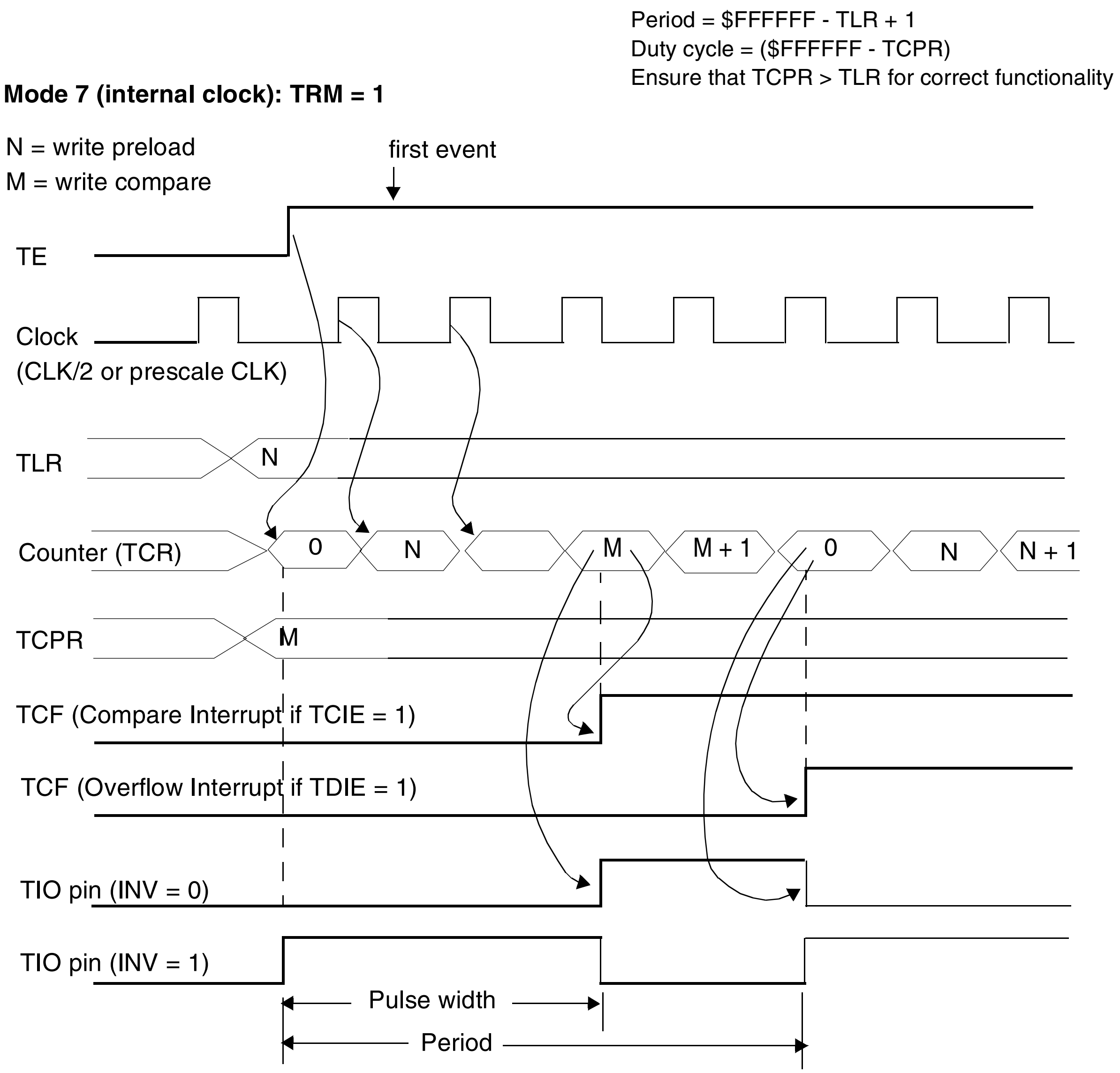

Period = $FFFFFF - TLR + 1
Duty cycle = ($FFFFFF - TCPR)
Ensure that TCPR > TLR for correct functionality
Mode 7 (internal clock): TRM = 1
N = write preload
M = write compare
first event
TE
Clock
(CLK/2 or prescale CLK)
TLR
N
Counter (TCR)
0
N
M
M + 1
0
N
N + 1
TCPR
M
TCF (Compare Interrupt if TCIE = 1)
TCF (Overflow Interrupt if TDIE = 1)
TIO pin (INV = 0)
TIO pin (INV = 1)
Pulse width
Period

Figure 10-16. Pulse Width Modulation Toggle Mode, TRM = 1

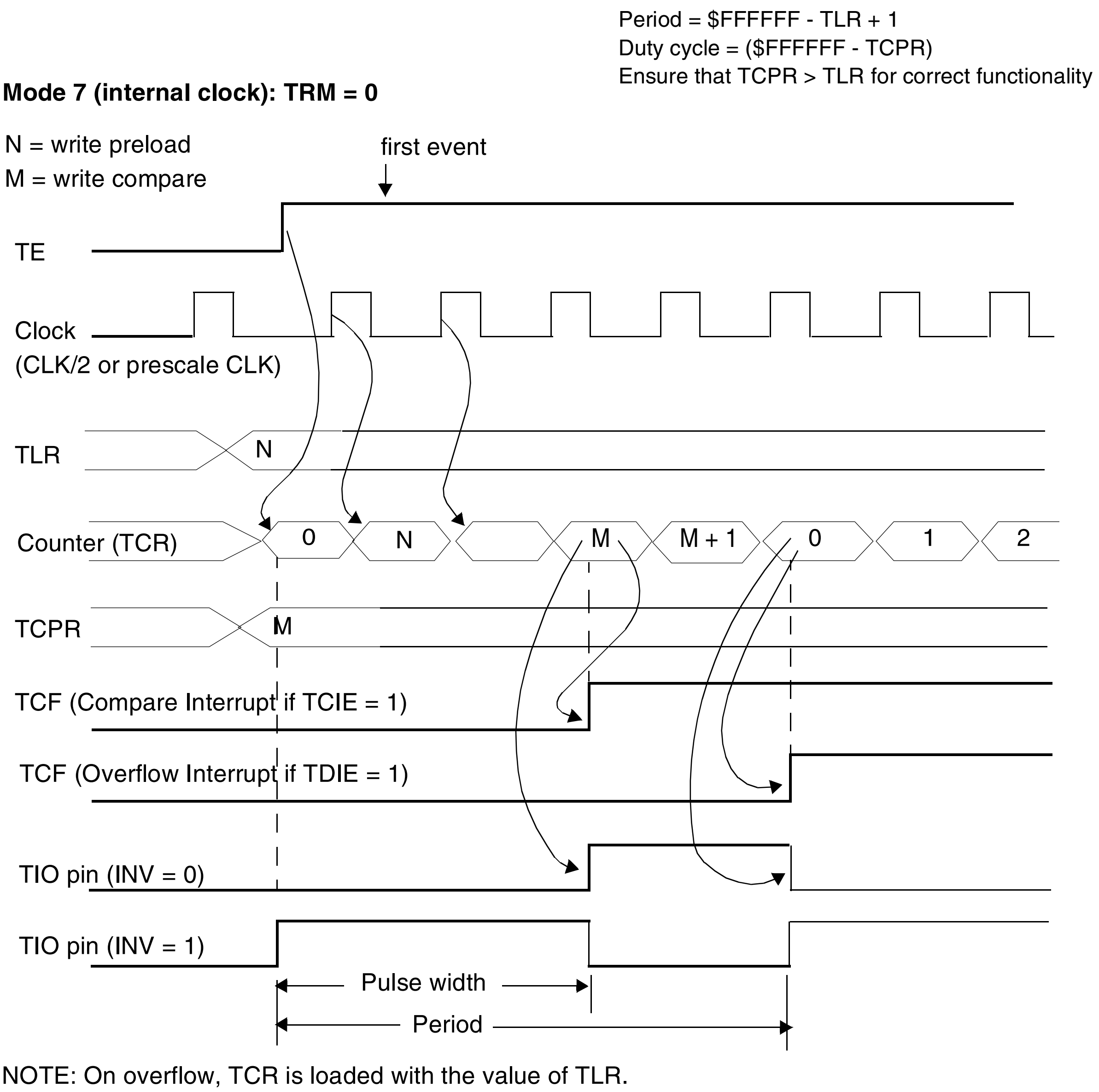

Period = $FFFFFF - TLR + 1
Duty cycle = ($FFFFFF - TCPR)
Ensure that TCPR > TLR for correct functionality
Mode 7 (internal clock): TRM = 0
N = write preload
M = write compare
first event
TE
Clock
(CLK/2 or prescale CLK)
TLR
N
Counter (TCR)
0
N
M
M + 1
0
1
2
TCPR
M
TCF (Compare Interrupt if TCIE = 1)
TCF (Overflow Interrupt if TDIE = 1)
TIO pin (INV = 0)
TIO pin (INV = 1)
Pulse width
Period
NOTE: On overflow, TCR is loaded with the value of TLR.

Figure 10-17. Pulse Width Modulation Toggle Mode, TRM = 0

10.3.4 Watchdog Modes

The following watchdog timer modes are provided:

- Watchdog Pulse
- Watchdog Toggle

10.3.4.1 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 Mode 9, the timer generates an external signal at a preset rate. The signal period is equal to the period of one timer clock. After the counter reaches the value in the TCPR, if the TCSR[TRM] bit is set, the counter is loaded with the TLR value on the next timer clock and the count resumes. Therefore TRM = 1 is not useful for watchdog functions. If the TCSR[TRM] bit is cleared, the counter continues to increment on each subsequent timer clock. This process repeats until the timer is disabled (that is, TCSR[TE] is cleared). If the counter overflows, a pulse is output on the TIO signal with a pulse width equal to the timer clock period. If the INV bit is set, the pulse polarity is high (logical 1). If INV is cleared, the pulse polarity is low (logical 0). The counter reloads when the TLR is written with a new value while the TCSR[TE] bit is set. In Mode 9, internal logic preserves the TIO value and direction for an additional 2.5 internal clock cycles after the hardware $\overline{\text{RESET}}$ signal is asserted. This convention ensures that a valid $\overline{\text{RESET}}$ signal is generated when the TIO signal resets the DSP56321.

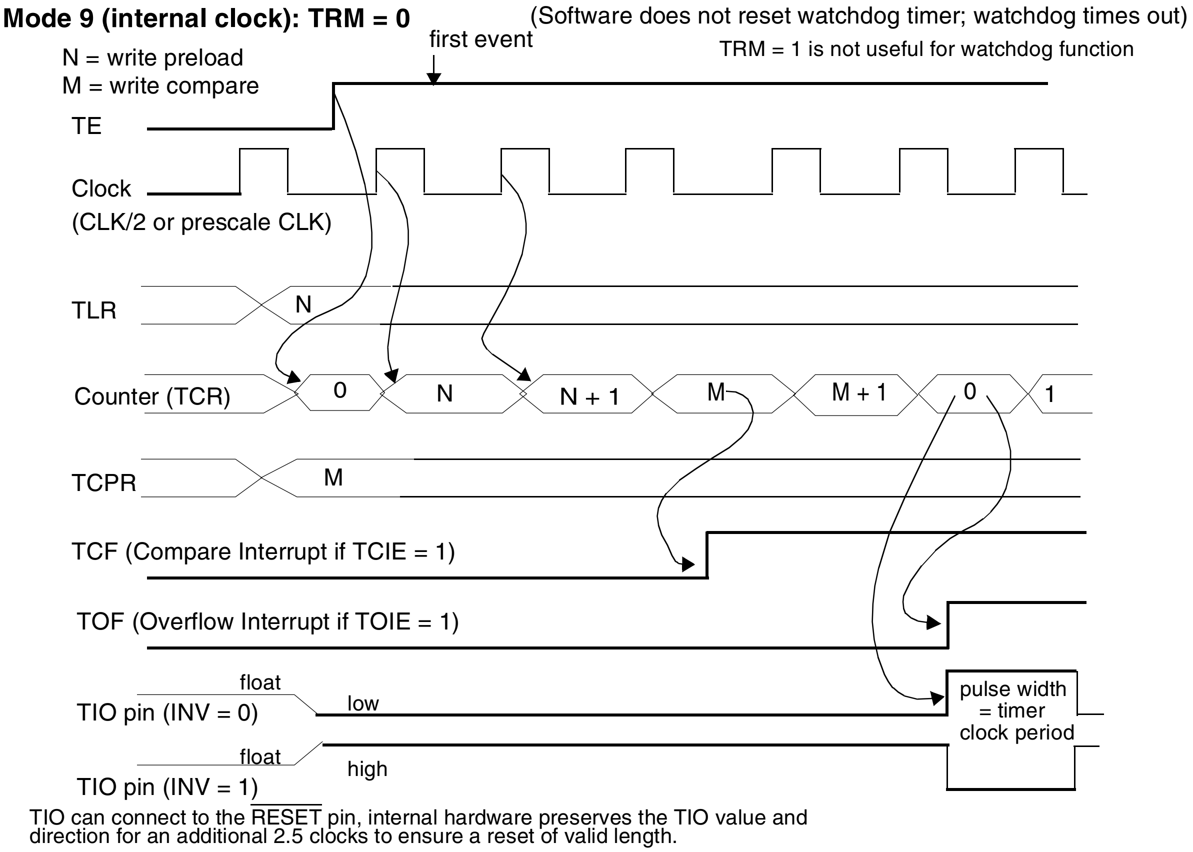
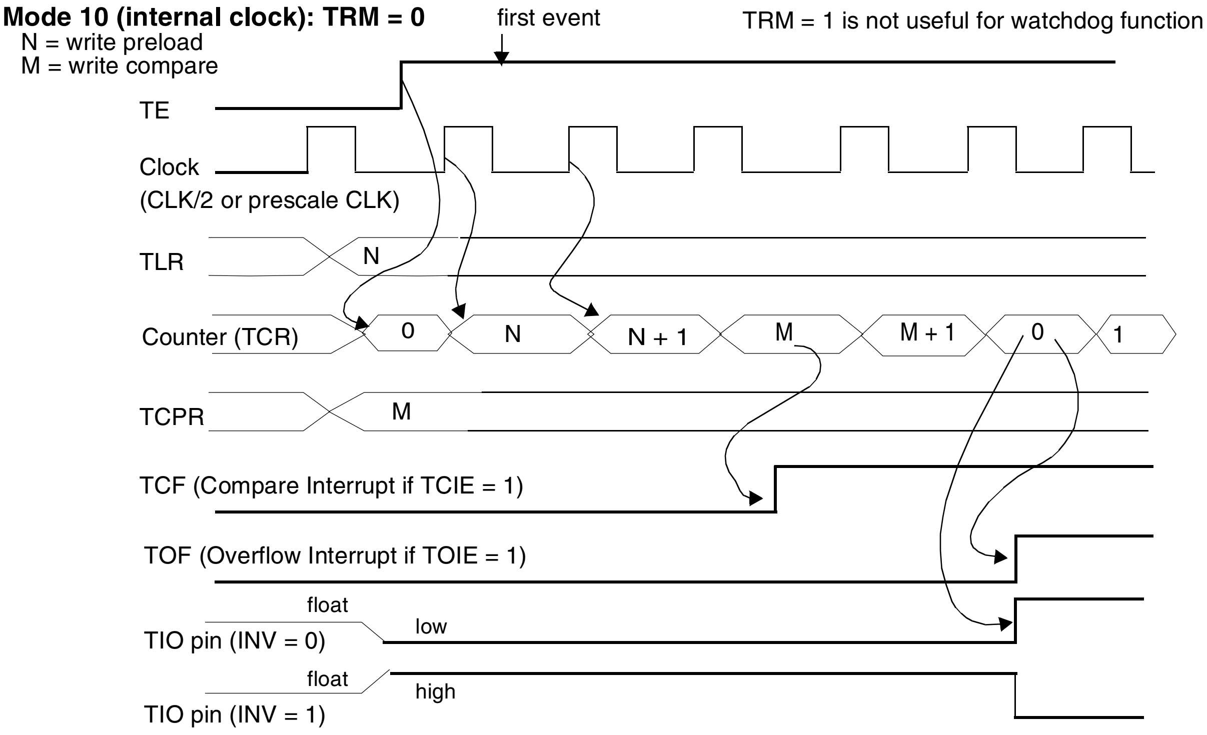

Mode 9 (internal clock): TRM = 0
(Software does not reset watchdog timer; watchdog times out)
first event
TRM = 1 is not useful for watchdog function
N = write preload
M = write compare
TE
Clock
(CLK/2 or prescale CLK)
TLR
N
Counter (TCR)
0
N
N + 1
M
M + 1
0
1
TCPR
M
TCF (Compare Interrupt if TCIE = 1)
TOF (Overflow Interrupt if TOIE = 1)
float
TIO pin (INV = 0)
low
pulse width = timer clock period
float
high
TIO pin (INV = 1)
TIO can connect to the $\overline{\text{RESET}}$ pin, internal hardware preserves the TIO value and direction for an additional 2.5 clocks to ensure a reset of valid length.

Figure 10-18. Watchdog Pulse Mode

10.3.4.2 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 Mode 10, the timer toggles an external signal after a preset period. The TIO signal is set to the value of the INV bit.When the counter equals the value in the TCPR, TCSR[TCF] is set, and a compare interrupt is generated if the TCSR[TCIE] bit is also set. If the TCSR[TRM] bit is set, the counter loads with the TLR value on the next timer clock and the count resumes. Therefore, TRM = 1 is not useful for watchdog functions. If the TCSR[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 counter is reloaded whenever the TLR is written with a new value while the TCSR[TE] bit is set. This process repeats until the timer is disabled. In Mode 10, internal logic preserves the TIO value and direction for an additional 2.5 internal clock cycles after the hardware $\overline{\text{RESET}}$ signal is asserted. This convention ensures that a valid reset signal is generated when the TIO signal resets the DSP56321.



TIO can connect to the RESET pin, internal hardware preserves the TIO value and direction for an additional 2.5 clocks to ensure a reset of valid length.

Figure 10-19. Watchdog Toggle Mode

10.3.4.3 Reserved Modes

Modes 8, 11, 12, 13, 14, and 15 are reserved.

10.3.5 Special Cases

The following special cases apply during wait and stop state.

- 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 DSP56321 leaves the wait state and services the interrupt.
- Timer behavior during stop — During execution of the STOP instruction, the timer clocks are disabled, timer activity stops, and the TIO signals are disconnected. Any external changes that happen to the TIO signals are ignored when the DSP56321 is in stop state. To ensure correct operation, disable the timers before the DSP56321 is placed in stop state.

10.3.6 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. 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.

10.4 Triple Timer Module Programming Model

The timer programmer’s model in **Figure 10-20** shows the structure of the timer registers.

10.4.1 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 (that is, one or more of the timer enable bits are set) and is using the prescaler output as its source (that is, one or more of the PCE bits are set).

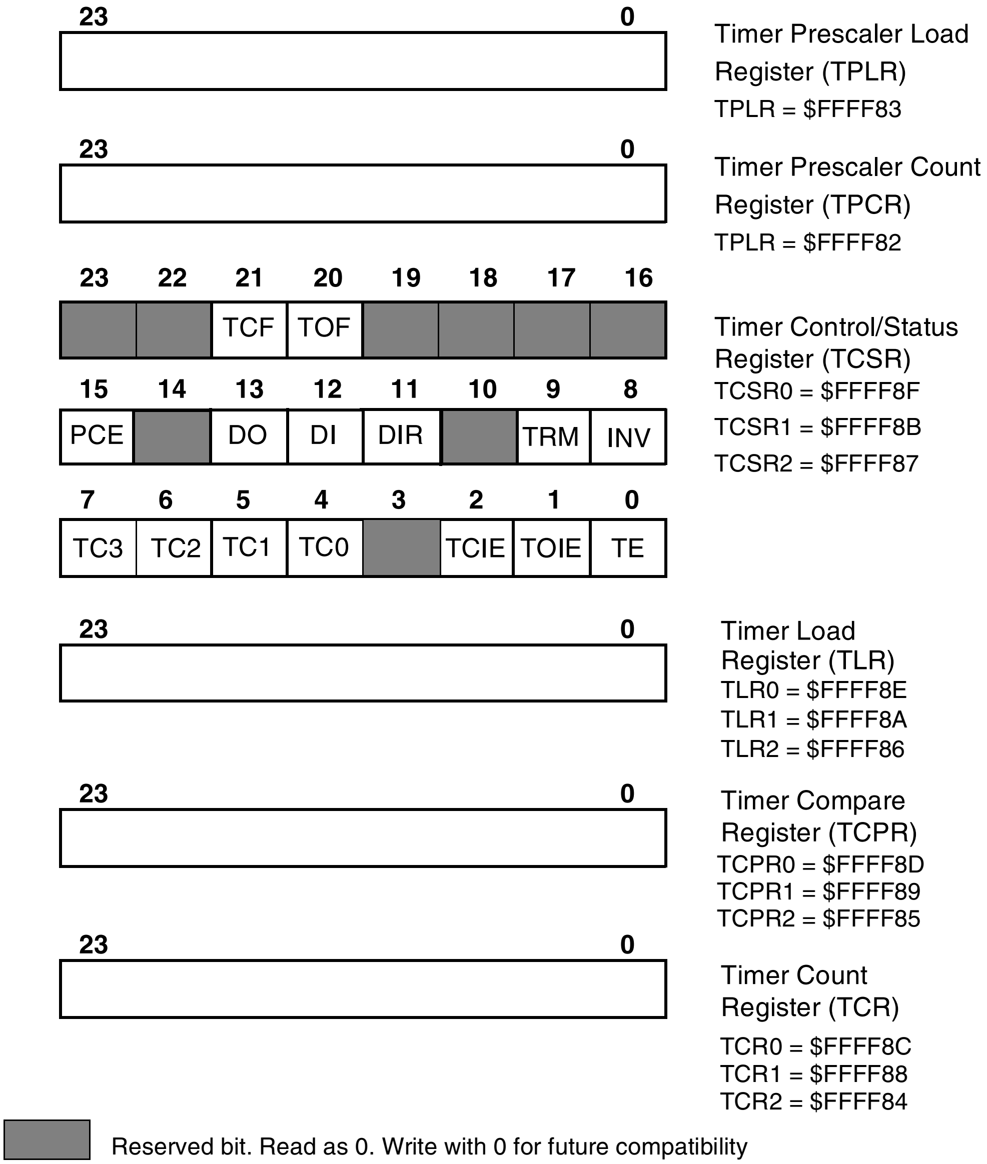


Figure 10-20. Timer Module Programmer's Model

10.4.2 Timer Prescaler Load Register (TPLR)

The TPLR is a read/write register that controls the prescaler divide factor (that is, the number that the prescaler counter loads and begins counting from) and the source for the prescaler input clock.

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. Write to 0 for future compatibility										

Table 10-1. Timer Prescaler Load Register (TPLR) Bit Definitions

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th colspan="3">Description</th></tr>
<tr><td>23</td><td></td><td>0</td><td colspan="3">Reserved. Write to zero for future compatibility.</td></tr>
<tr><td rowspan="6">22–21</td><td rowspan="6">PS[1–0]</td><td rowspan="6">0</td><td colspan="3"><b>Prescaler Source</b>
Control the source of the prescaler clock. 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 DSP56321 internal operating frequency divided by 4 (that is, CLK/4).

NOTE: To ensure proper operation, change the PS[1–0] bits only when the prescaler counter is disabled. Disable the prescaler counter by clearing TCSR[TE] of each of three timers.</td></tr>
<tr><th>PS1</th><th>PS0</th><th>Prescaler Clock Source</th></tr>
<tr><td>0</td><td>0</td><td>Internal CLK/2</td></tr>
<tr><td>0</td><td>1</td><td>TIO0</td></tr>
<tr><td>1</td><td>0</td><td>TIO1</td></tr>
<tr><td>1</td><td>1</td><td>TIO2</td></tr>
<tr><td>20–0</td><td>PL[20–0]</td><td>0</td><td colspan="3"><b>Prescaler Preload Value</b>
Contains 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.</td></tr>
</table>

10.4.3 Timer Prescaler Count Register (TPCR)

The TPCR is a read-only register that reflects the current value in the prescaler counter.

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; write to 0 for future compatibility

Table 10-2. Timer Prescaler Count Register (TPCR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23–21		0	Reserved. Write to zero for future compatibility.
20–0	PC[20–0]	0	**Prescaler Counter Value** Contain the current value of the prescaler counter.

10.4.4 Timer Control/Status Register (TCSR)

The TCSR is a read/write register controlling the timer and reflecting its status.

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

▨ Reserved. Read as 0. Write to 0 for future compatibility

Table 10-3. Timer Control/Status Register (TCSR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23–22		0	Reserved. Write to zero for future compatibility.

Table 10-3. Timer Control/Status Register (TCSR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
21	TCF	0	**Timer Compare Flag** Indicate that the event count is complete. In timer, PWM, and watchdog modes, the TCF bit is set after (M – N + 1) events are counted. (M is the value in the compare register and N is the TLR value.) In measurement modes, the TCF bit is set when the measurement completes. Writing a one to the TCF bit clears it. A zero 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 TCSR[TE] bit to disable the timer. NOTE: The TOF and TCF bits are cleared by a 1 written to the specific bit. To ensure 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.
20	TOF	0	**Timer Overflow Flag** Indicates that a counter overflow has occurred. This bit is cleared by writing a one to the TOF bit. Writing a zero to TOF 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 TCSR[TE] bit to disable the timer.
19–16		0	Reserved. Write to zero for future compatibility.
15	PCE	0	**Prescaler Clock Enable** Selects the prescaler clock as the timer source clock. When PCE is cleared, the timer uses either an internal (CLK/2) signal or an external (TIO) signal as its source clock. When PCE is set, the prescaler output is the timer source clock for the counter, regardless of the timer operating mode. To ensure proper operation, the PCE bit is changed only when the timer is disabled. 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.
14		0	Reserved. Write to zero for future compatibility.
13	DO	0	**Data Output** The source of the TIO value when it is a data output signal. The TIO signal is a 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 to the DO bit has no effect.
12	DI	0	**Data Input** 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.
11	DIR	0	**Direction** Determines the behavior of the TIO signal when it functions as a GPIO signal. When DIR is set, the TIO signal is an output; when DIR 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.

Table 10-3. Timer Control/Status Register (TCSR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
10		0	Reserved. Write to zero for future compatibility.
9	TRM	0	**Timer Reload Mode** Controls the counter preload operation. In timer (0–3) and watchdog (9–10) modes, the counter is preloaded with the TLR value after the TCSR[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 TCSR[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.
8	INV	0	**Inverter** 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. See **Table 10-4**, *Inverter (INV) Bit Operation,* on page 10-28. The INV bit does not affect the polarity of the prescaler source when the TIO is input to the prescaler. 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 TCSR[TE] bit is cleared). The timer is in GPIO mode.

Table 10-3. Timer Control/Status Register (TCSR) Bit Definitions (Continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th>Description</th></tr>
<tr><td>7–4</td><td>TC[3–0]</td><td>0</td><td><b>Timer Control</b>
Control the source of the timer clock, the behavior of the TIO signal, and the Timer mode of operation. <b>Section 10.3</b>, <i>Operating Modes</i>, on page 10-5 describes the timer operating modes in detail. To ensure proper operation, the TC[3–0] bits should be changed only when the timer is disabled (that is, when the TCSR[TE] bit is cleared). 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 (that is, CLK/4).
<table>
<tr><th colspan="4">Bit Settings</th><th colspan="4">Mode Characteristics</th></tr>
<tr><th>TC3</th><th>TC2</th><th>TC1</th><th>TC0</th><th>Mode Number</th><th>Mode Function</th><th>TIO</th><th>Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Timer and GPIO</td><td>GPIO [1]</td><td>Internal</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>Timer pulse</td><td>Output</td><td>Internal</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td><td>Timer toggle</td><td>Output</td><td>Internal</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>3</td><td>Event counter</td><td>Input</td><td>External</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>4</td><td>Input width measurement</td><td>Input</td><td>Internal</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>5</td><td>Input period measurement</td><td>Input</td><td>Internal</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>6</td><td>Capture event</td><td>Input</td><td>Internal</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>7</td><td>Pulse width modulation</td><td>Output</td><td>Internal</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>8</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>9</td><td>Watchdog pulse</td><td>Output</td><td>Internal</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>10</td><td>Watchdog Toggle</td><td>Output</td><td>Internal</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>11</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>12</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>13</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>14</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>15</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td colspan="8"><b>Notes:</b> <b>1.</b> The GPIO function is enabled only if all of the TC[3–0] bits are 0.</td></tr>
</table>
</td></tr>
<tr><td>3</td><td></td><td>0</td><td>Reserved. Write to zero for future compatibility.</td></tr>
</table>

Table 10-3. Timer Control/Status Register (TCSR) Bit Definitions (Continued)

Bit Number	Bit Name	Reset Value	Description
2	TCIE	0	**Timer Compare Interrupt Enable** Enables/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 M, an interrupt occurs after (M – N + 1) events, where N is the value of TLR. When cleared, the TCSR[TCIE] bit disables the compare interrupts.
1	TOIE	0	**Timer Overflow Interrupt Enable** 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, the TOIE bit disables overflow interrupt generation.
0	TE	0	**Timer Enable** Enables/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. NOTE: When all three timers are disabled and the signals are not in GPIO mode, all three TIO signals are tri-stated. To prevent undesired spikes on the TIO signals when 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.

Table 10-4. 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.	Initial output put on TIO signal directly.	Initial output inverted and put on TIO signal.
3	Counter is incremented on the **rising** edge of the signal from the TIO signal.	Counter is incremented on the **falling** edge of the signal from the TIO signal.	—	—
4	Width of the **high** input pulse is measured.	Width of the **low** input pulse is measured.	—	—

Table 10-4. Inverter (INV) Bit Operation (Continued)

Mode	TIO Programmed as Input		TIO Programmed as Output	
	INV = 0	INV = 1	INV = 0	INV = 1
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.

10.4.5 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 TCSR[TE] bit is set and a first event occurs.

- In timer modes, if the TCSR[TRM] bit 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 measurement modes, if TCSR[TRM] and TCSR[TE] are set, the counter is reloaded with the value in the TLR on each appropriate edge of the input signal.
- In PWM modes, if TCSR[TRM] is set, the counter is reloaded each time after it overflows and the new event occurs.
- In watchdog modes, if TCSR[TRM] 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 TCSR[TE] is set.
- In all modes, if TCSR[TRM] is cleared (TRM = 0), the counter operates as a free-running counter.

10.4.6 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 TCSR[TE] is set. When the values match, the timer compare flag bit is set and an interrupt is generated if interrupts are enabled (that is, the timer compare interrupt enable bit in the TCSR is set). The TCPR is ignored in measurement modes.

10.4.7 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.

Enhanced Filter Coprocessor 11

The enhanced filter coprocessor (EFCOP) peripheral module functions as a general-purpose, fully programmable filter. It has optimized modes of operation to perform single-channel and multichannel real and complex finite impulse response (FIR) filtering with and without adaptive FIR filtering and decimation or single-channel and multichannel infinite impulse response (IIR) filtering. EFCOP filter operations complete 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 filter coprocessor 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. This chapter describes the EFCOP features, architecture, operation, and programming model.

11.1 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
  - Three optional output scaling factors (1, 8, or 16)

- 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 12 K-word filter data memory bank and 12 K-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

11.2 Architecture Overview

As **Figure 11-1** shows, 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

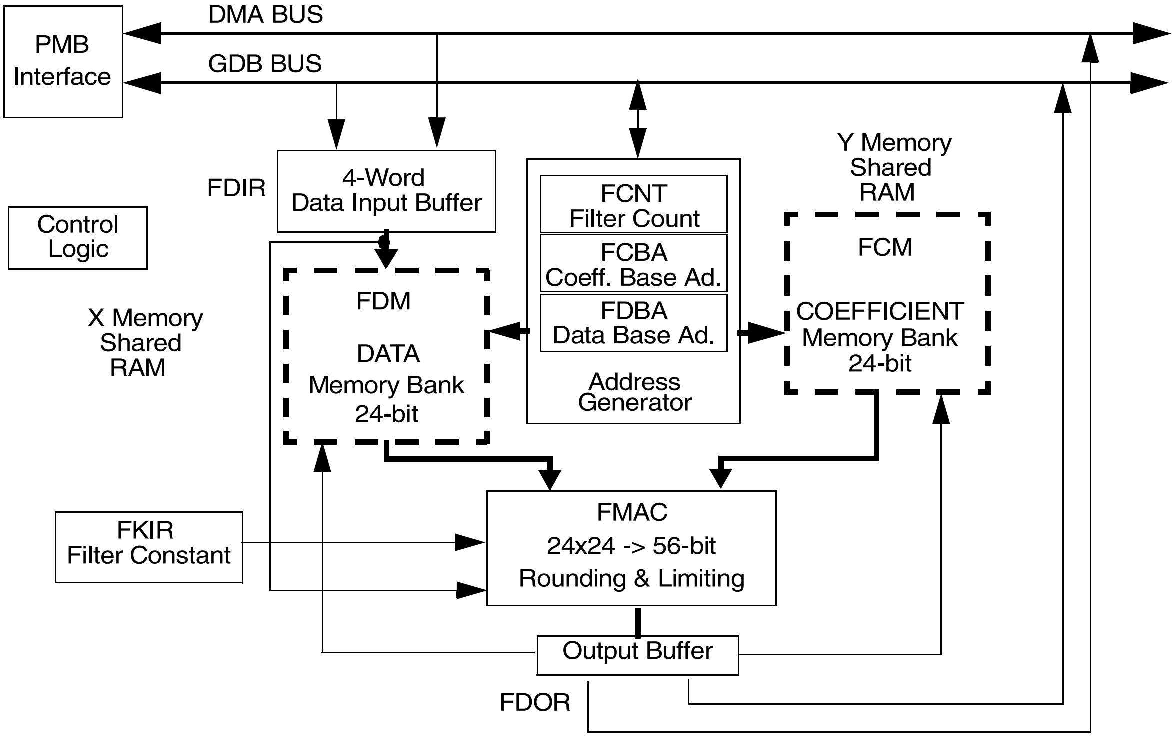


Figure 11-1. EFCOP Block Diagram

11.2.1 PMB Interface

The PMB interface block contains control and status registers, buffers the internal bus from the PMB, decodes and generates addresses, and controls the handshake signals required for DMA and interrupt operations. The block generates interrupt and DMA trigger signals for data transfers. The interface registers accessible to the DSP56300 core through the PMB are summarized in **Table 11-1**.

Table 11-1. EFCOP Registers Accessible Through the PMB

Register Name	Description
Filter Data Input Register (FDIR)	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)	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)	A 24-bit register for DSP-to-EFCOP constant transfers.
Filter Count (FCNT) Register	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)	A 24-bit read/write register used by the DSP56300 core to program the EFCOP and to examine the status of the EFCOP module.

Table 11-1. EFCOP Registers Accessible Through the PMB (Continued)

Register Name	Description
EFCOP ALU Control Register (FACR)	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)	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)	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)	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.

11.2.2 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 12 K lowest locations ($0–$2FFF) 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 12 K lowest locations ($0–$2FFF) 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 11-2**.

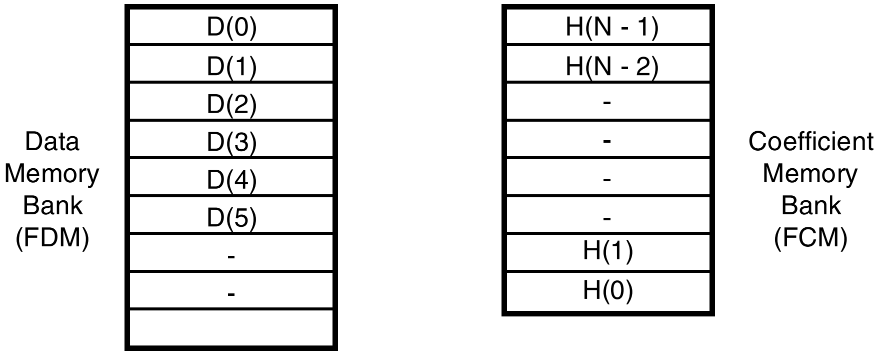


Figure 11-2. Storage of Filter Coefficients

The EFCOP connects to the shared memory in place of the DMA bus. Simultaneous core and EFCOP accesses to the same memory module block (1024 locations) of shared memory are not permitted. It is your responsibility to prevent such simultaneous accesses. **Figure 11-3** illustrates the memory shared between the core and the EFCOP.

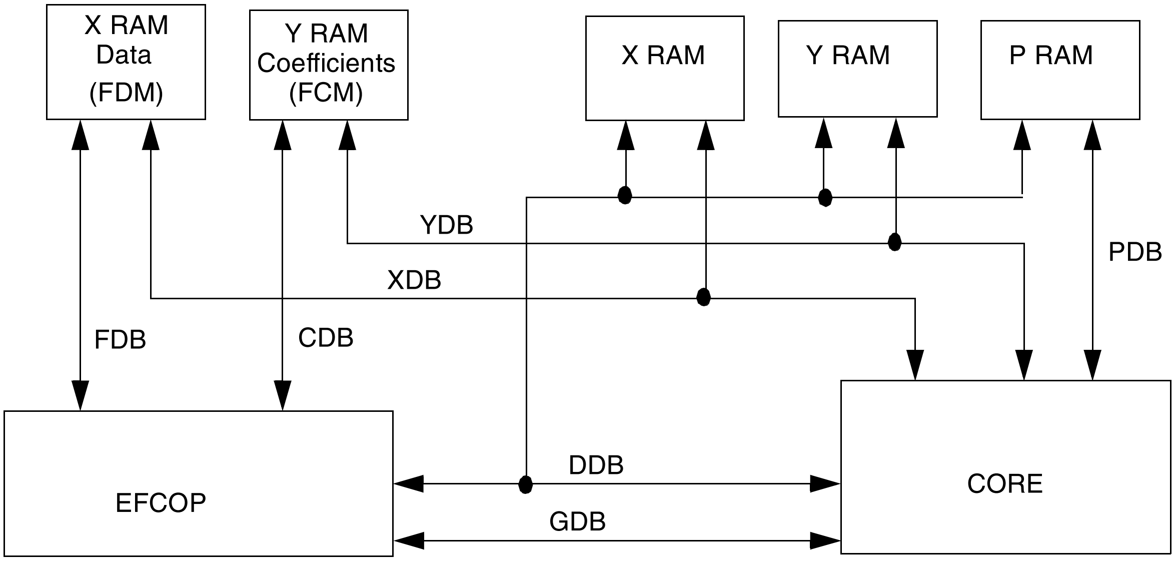


Figure 11-3. EFCOP Memory Organization

11.2.3 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 with saturation mode 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.

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.

11.3 EFCOP Operation

DSP56L307 EFCOP operation is determined by the control bits in the EFCOP Control/Status Register (FCSR), described in **Section 11.4.5**. 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 FCSR[FEN].

Note: To ensure proper EFCOP operation, most FCSR bits must be changed only while the EFCOP is enabled.

Table 11-2 summarizes the EFCOP operating modes.

Table 11-2. 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

Notes:
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 (that is, 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.

11.3.1 EFCOP Operation Summary

The EFCOP is very easy to use. To define the type of filtering to perform, you need only set the following registers (the settings in the FDCH and FACR are optional) and then enable the EFCOP by setting FCSR[FEN]:

- FCNT
- FDBA
- FCBA
- FCSR

Polling, DMA, or interrupts can then be used to write data to the FDIR and read data from the FDOR. As **Table 11-2** shows, the EFCOP operates in many different modes based on the settings of the control registers. However, the EFCOP performs only two basic types of processing, FIR filter type and IIR filter type processing. Various sub-options are available with each filter type, as described in the following sections.

11.3.2 EFCOP Initialization

Before the first sample is processed, the EFCOP filter must be initialized; that is, the input samples for times before n = 0 (assuming that time starts at 0) must be loaded into the FDM. The method by which this is done depends on the Filter Type selected.

11.3.2.1 FIR Initialization

The number of samples needed to initialize the filter is the number of filter coefficients minus one. To select Initialization mode, clear the FCSR[FPRC] bit. If FCSR[FPRC] is set, initialization is disabled and the EFCOP assumes that the core wrote the initial input values to the FDM before the EFCOP was enabled. Thus, the first value written to FDIR is the first sample to be filtered.

If FCSR[FPRC] is clear, initialization mode is enabled and the EFCOP initializes the FDM by receiving the number of coefficients minus one samples through the FDIR. After samples are loaded, the next value written to the FDIR is the first sample to be filtered.

11.3.2.2 IIR Initialization

Initialization is always disabled with the IIR filter type, and the FCSR[FPRC] bit is ignored. Thus, the DSP56300 core must write the initial input values before the EFCOP is enabled. The first value written to FDIR is always the first sample to be filtered.

11.3.3 FIR Filter Type

To select the FIR filter type clear FCSR[FLT] and perform the processing shown in **Figure 11-4** based on the equation shown below. The EFCOP takes an input, *x(n)*, from the FDIR, saves the

$$w(n) = \sum_{i=0}^{N} B_i x(n-i)$$

input while shifting the previous inputs down in the FDM, multiplies each input in the FDM by the corresponding coefficient, B_i, stored in the FCM, accumulates the multiplication results, and places accumulation result, *w(n)*, in the FDOR. This is done for each sample input to the FDIR.

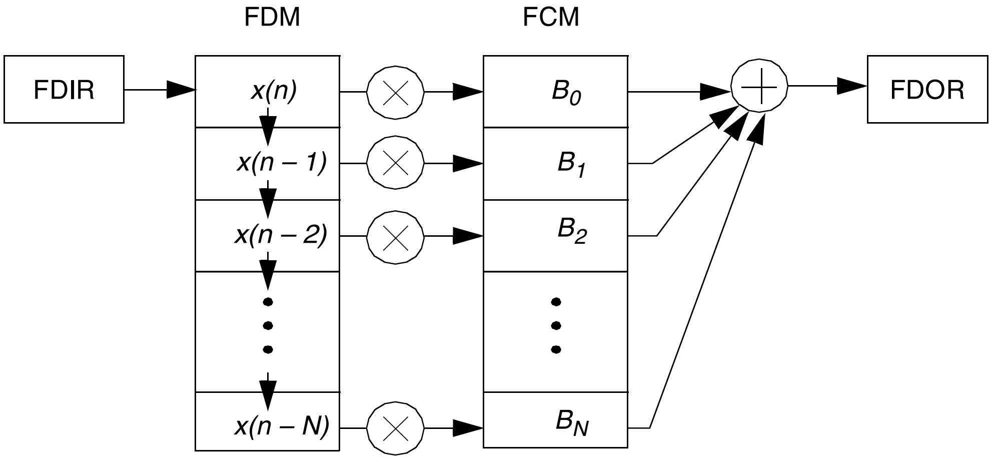


Figure 11-4. FIR Filter Type Processing

11.3.3.1 FIR Operating Modes

There are four operating modes available with the FIR filter type:

- Real
- Complex
- Alternating complex
- Magnitude mode.

11.3.3.1.1 Real Mode

Real mode performs FIR type filtering with real data and is selected by clearing both FOM bits in the FCSR. One sample, the real input, is written to the FDIR, and the EFCOP processes the data. Then one sample, the real output, is read from the FDOR. Four options are available with the real FIR filter type: coefficient update, adaptive mode, multichannel mode, and decimation. The first three options can be used individually or together. Decimation cannot be used with the adaptive and multichannel mode options

11.3.3.1.2 Complex Mode

Complex mode performs FIR type filtering with complex data based on the following equations:

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

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

where *H(n)* is the coefficients, *D(n)* is the input data, and *F(n)* is the output data at time *n*. Two samples, the real part then the imaginary part of the input, are written to the FDIR. The EFCOP processes the data, and then two samples—the real and then the imaginary part of the output—are read from the FDOR.

Complex mode is selected by setting the FCSR[FOM] bits to 01. In Complex mode, the number written to the FCNT register should be twice the number of filter coefficients. Also, the coefficients are stored in the FCM with the real part of the coefficient in the memory location preceding the memory location holding the imaginary part of the coefficient. Complex mode can be used with the decimation option.

11.3.3.1.3 Alternating Complex Mode

Alternating Complex mode performs FIR type filtering with complex data, providing alternating real and complex results based on the following equations

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

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

where *H(n)* is the coefficients, *D(n)* is the input data, and *F(n)* is the output data at time *n*. Two samples, the real part then the imaginary part of the input, are written to the FDIR. The EFCOP processes the data. Then one sample, alternating between the real part and the imaginary part of the output, is read from the FDOR.

Alternating Complex mode is selected by setting the FCSR[FOM] bits to 10. In Alternating Complex mode, the number written to the FCNT register should be twice the number of filter coefficients. Also, the coefficients should be stored in the FCM with the real part of the coefficient in the memory location preceding the memory location holding the imaginary part of the coefficient. Alternating Complex mode can be used with the decimation option.

11.3.3.1.4 Magnitude Mode

Magnitude mode calculates the magnitude of an input signal based on the following equation:

$$F(n) = \sum_{i=0}^{N-1} D(n-i)^2$$

where *D(n)* is the input data and *F(n)* is the output data at time *n*. One sample, the real input, is written to the FDIR. The EFCOP processes the data. Then one sample, the real magnitude of the input signal, is read from the FDOR. Magnitude mode is selected by setting both the FCSR[FOM] bits. Magnitude mode can be used with the decimation option.

11.3.3.2 FIR Filter Type Processing Options

There are four processing option available for the FIR filter type:

- Coefficient Update
- Adaptive Mode
- Multichannel Mode
- Decimation

11.3.3.2.1 Coefficient Update Option

The Coefficient Update option is only available in Real operating mode. If the user sets the FUPD bit, the EFCOP updates the coefficients and clears the FUPD bit. When used with adaptive mode, this only sets the initial update, since the mode sets the FUPD bit dynamically during normal operation to update the coefficients.

11.3.3.2.2 Adaptive Mode Option

Adaptive mode is only available in Real operating mode. It provides a way to update the coefficients based on filter input, *x(n)*, using the following equation,

$$h_{n+1}(i) = h_n(i) + K_e(n)x(n-i)$$

where $h_n(i)$ is the *i*th coefficient at time *n*. The coefficients are updated when FSCR[FUPD] is set. The EFCOP checks to see if a value has been written to the FKIR. If no value is written, the EFCOP halts processing until a value is written to the FKIR. When a value is written to the FKIR, the EFCOP updates all the coefficients based on the above equation using the value in the

FKIR for $K_e(n)$. The EFCOP automatically clears FSCR[FUPD] when the coefficient update is complete.

If the coefficients are to be updated after every input sample, Adaptive mode is enabled by setting the FCSR[FADP]. In Adaptive mode, the EFCOP automatically sets the FUDP bit after each input sample is processed. This allows for continuous processing using interrupts that includes a filter session and a coefficient update session with minimal core intervention.

11.3.3.2.3 Multichannel Mode Option

Multichannel mode is only available in Real operating mode. It allows several channels of data to be processed concurrently and is selected by setting the FCSR[FMLC]. The number of channels to process is one plus the number in the FDCH[FDCM] bits. For each time period, the EFCOP expects to receive the samples for each channel sequentially. This is repeated for consecutive time periods.

Filtering can be done with the same filter or different filters for each channel by using the FCSR[FSCO] bit. If FCSR[FSCO] is set, the same set of coefficients are used for all channels. If FSCO is clear, the coefficients for each filter are stored sequentially in memory for each channel.

11.3.3.2.4 Decimation Option

Decimation can be used with any four of the available FIR filter type modes. It cannot be used in conjunction with Adaptive and Multichannel modes. Decimation decreases (downsamples) the sampling rate. The decimation ratio defines the number of input samples per output sample. The decimation ratio is one plus the number in the FDCH[FDCM] bits. The decimation ratio can be programmed from 1 to 16.

For Real and Magnitude modes the decimation ratio number of samples must be written to the FDIR before an output sample is read from the FDOR. For Complex mode, two times the decimation ratio number of samples (one for the real part and one for the imaginary part of the input) must be written to the FDIR before two output samples (one for the real part and one for the imaginary part of the output) can be read from the FDOR. For Alternating Complex mode, two times the decimation ratio number of samples must be written to the FDIR (one for the real part and one for the imaginary part of the input) before one output sample (alternating between the real part and the imaginary part of the output) can be read from the FDOR.

11.3.4 IIR Filter Type

To select the IIR filter type, set the FCSR[FLT] bit and perform the processing shown in **Figure 11-5** based on the equation shown here. The EFCOP multiplies each previous output value in the

$$y(n) = S\left(w(n) + \sum_{j=1}^{M} A_j y(n-j)\right)$$

FDM by the corresponding coefficient, *A*, stored in the FCM; accumulates the multiplication results; adds the input, *w(n)*, from the FDIR (which is optionally not scaled by S, depending on the FACR[FISL] bit setting); places the accumulation result, *y(n)*, in the FDOR; and saves the output while shifting the previous outputs down in the FDM. This is done for each sample input to the FDIR. To process a complete IIR filter, a FIR filter type session followed by an IIR filter type session is needed.

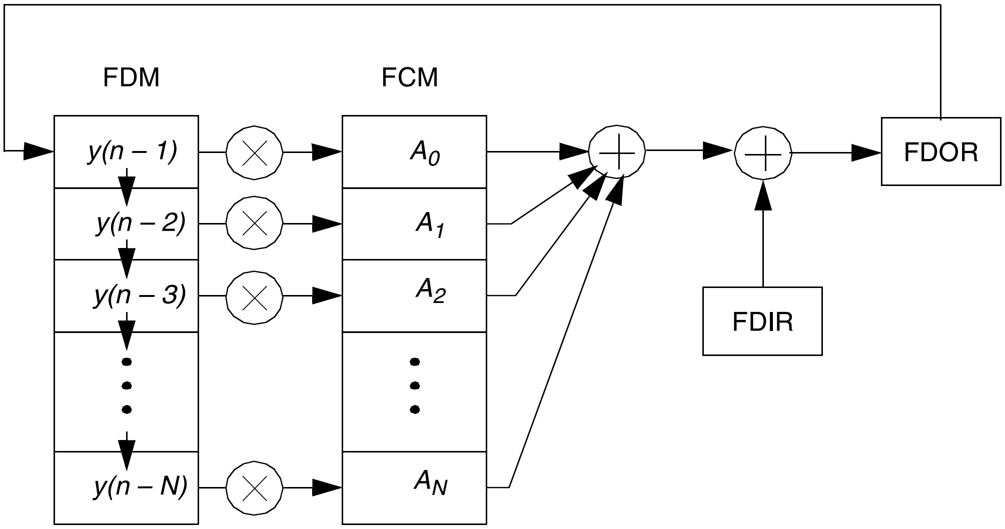


Figure 11-5. IIR Filter Type Processing

The IIR filter type only operates in Real mode. Thus, the FCSR[FOM] bits are ignored when the IIR filter type is in use. Real mode performs IIR type filtering with real data. One sample, the real input, is written to the FDIR, and the EFCOP processes the data. Then one sample, the real output, is read from the FDOR.

The default operation is single-channel, but the user can also select the Multi-channel Mode option. Multichannel mode for IIR filter type works exactly the same as it does for FIR filter type, as explained in **Section 11.3.3.2.3**, *Multichannel Mode Option*, on page 11-11. The Coefficient Update, Adaptive mode, and Decimation options are not available with the IIR filter type.

11.3.5 EFCOP Data Transfer Examples

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 **Section 11.3.6**, *EFCOP Operation Examples*, on page 11-14. The examples employ the following notations:

- 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

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 data transfer method uses 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 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 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 (that is, 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 FCSR[FPRC] bit is set), the EFCOP starts computation as soon as the first data sample is available in the input buffer. The filter coefficient bank must therefore be initialized before 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.

11.3.6 EFCOP Operation Examples

The following sections provide examples of how to use the EFCOP in Real FIR Filter and Adaptive FIR filter mode. **Section 11.3.6.4**, *Verification for Filter Examples*, on page 11-32 lists the programming inputs and outputs for the examples in the following sections.

11.3.6.1 Real FIR Filter

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.[1]

11.3.6.1.1 DMA Input/DMA Output

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

Set-up:

1. Set the filter count register (FCNT) to the length of the filter coefficients –1 (that is, N – 1).
2. Set the Data and Coefficient Base Address pointers (FDBA, FCBA).
3. Set the operation mode (FCSR[5:4] = FOM[00]).
4. Set Initialization mode (FCSR[7] = FPRC = 0).

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

5. Set DMA registers:

   DMA input: A two-dimensional (2D) DMA transfer fills up the FDM bank via channel 0. The DMA input control registers are initialized as shown in **Table 11-3**.

Table 11-3. DMA Channel 0 Register Initialization

Register Setting	Description
DCR0 bit values are as follows:	DMA Control Register 0
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
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 the EFCOP is in Y memory).
DSS = 0	Source in X memory space.
DOR0=1	DMA Offset Register 0
DCO0= $006003	DMA Counter Register 0 Gives transfer of 7 * 4 = 28 items (input sequence length).
DSR0 = Address of source data	DMA Source Address Register 0
DDR0 = $FFFFB0	DMA Destination Address Register for Channel 0

DMA output: Channel 1 is used, with a configuration similar to that of the DMA input channel, except for a 1D transfer. The DMA output control registers are initialized as shown in **Table 11-4**.

Table 11-4. DMA Channel 1 Register Initialization

Register Setting	Description
DCR1 bit values are as follows:	DMA Control Register 1
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.

Table 11-4. DMA Channel 1 Register Initialization (Continued)

Register Setting	Description
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	DMA Counter Register 1: Gives transfer of 9 items.
DSR1 = address of FDOR = $FFFFFB1	DMA Source Address Register 1
DDR1 = address of destination memory space	DMA Destination Address Register 1

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.

6. 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
7. Processing:
   - Whenever the Input Data Buffer (FDIR) is empty (that is, 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.

The filter coefficients are stored in “reverse order,” as **Figure 11-6** shows.

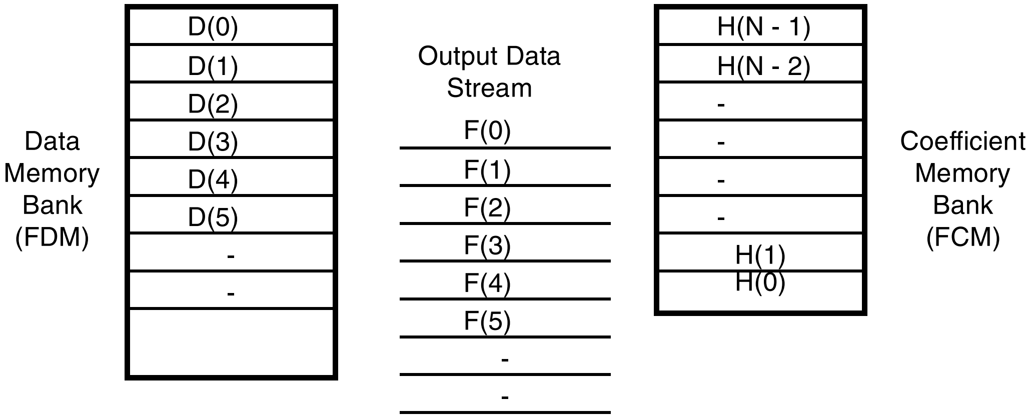


Figure 11-6. Real FIR Filter Data Stream

Example 11-1. 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     $3040; ; DMA source address point to DATA bank
DST_ADDRS       equ     $3000; 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
   movex0,x:(r0)+      ; clear FDM memory area
; ** DMA channel 1 initialization  - 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 initialization **
   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 initialization - 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'
```

11.3.6.1.2 DMA Input/Polling Output

The different stages of input/polling are as follows:

1. Set up:
   - Set the filter count register (FCNT) to the length of the filter coefficients – 1 (that is, 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 11.3.6.1.1**
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 (that is, 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 11-2. Real FIR Filtering using DMA input/Polling output

```
INCLUDE 'ioequ.asm'
;;******************************************************************
; equates
;;******************************************************************
Startequ$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     $3040; DMA source address point to DATA bank
DST_ADDRS       equ     $3000; 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

   movex0,x:(r0)+      ; clear FDM memory area

   move    #DST_ADDRS,r0 ; Destination address

; ** EFCOP initialization **

   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 initialization - 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’
```

11.3.6.1.3 DMA Input/Interrupt Output

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

1. Set up:
   - Set the filter count register (FCNT) to the length of the filter coefficients –1 (that is, 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 11.3.6.1.1**.
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 (that is, 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 11-3. Real FIR Filter DMA Input/Interrupt Output

```
INCLUDE ’ioequ.asm’

;;*******************************************************************

; equates

;;*******************************************************************

Startequ$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     $3040; DMA source address point to DATA bank
DST_ADDRS equ    $3000 ; 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
   jmpStart
   ORG p:$6a
   jsr     >kdo
   nop
   nop
;;*****************************************************************
; main program
;;*****************************************************************

   ORG p:Start
; ** interrupt initialization **
   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
   movex0,x:(r0)+      ; clear FDM memory area
   move    #DST_ADDRS,r0 ; Destination address
; ** EFCOP initialization **
```

```
   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 initialization - 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’
```

11.3.6.2 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,

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

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 11.3.6.1.1**. The difference is: set FDCH[11:8] = FDCM =M.

Processing:

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

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

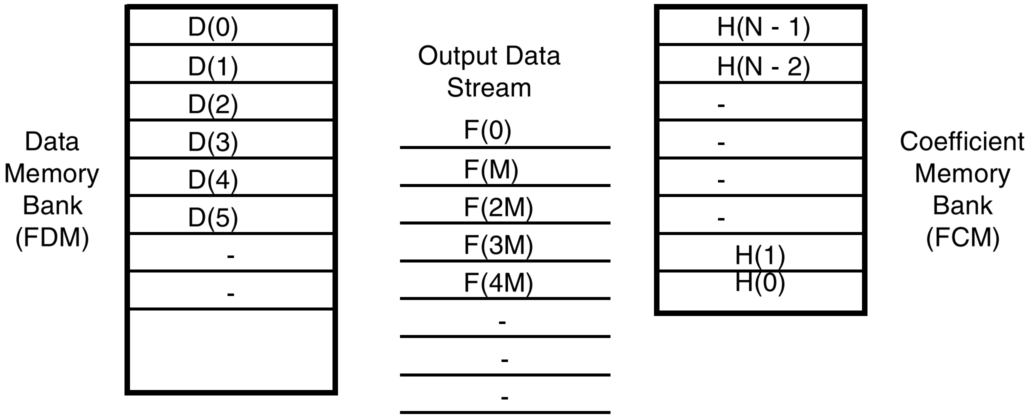


Figure 11-7. Real FIR Filter Data Stream With Decimation by M

11.3.6.3 Adaptive FIR Filter

An adaptive FIR filter is represented in **Figure 11-8**. 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.

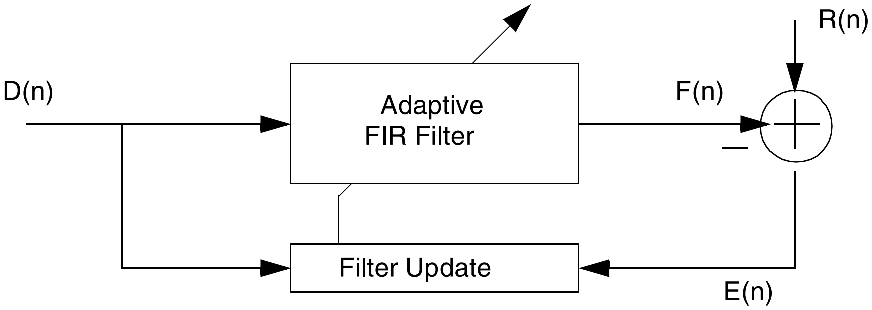


Figure 11-8. Adaptive FIR Filter

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).

11.3.6.3.1 Implementation Using Polling

Figure 11-9 shows a flowchart for an adaptive FIR filter that uses polling to transfer data.

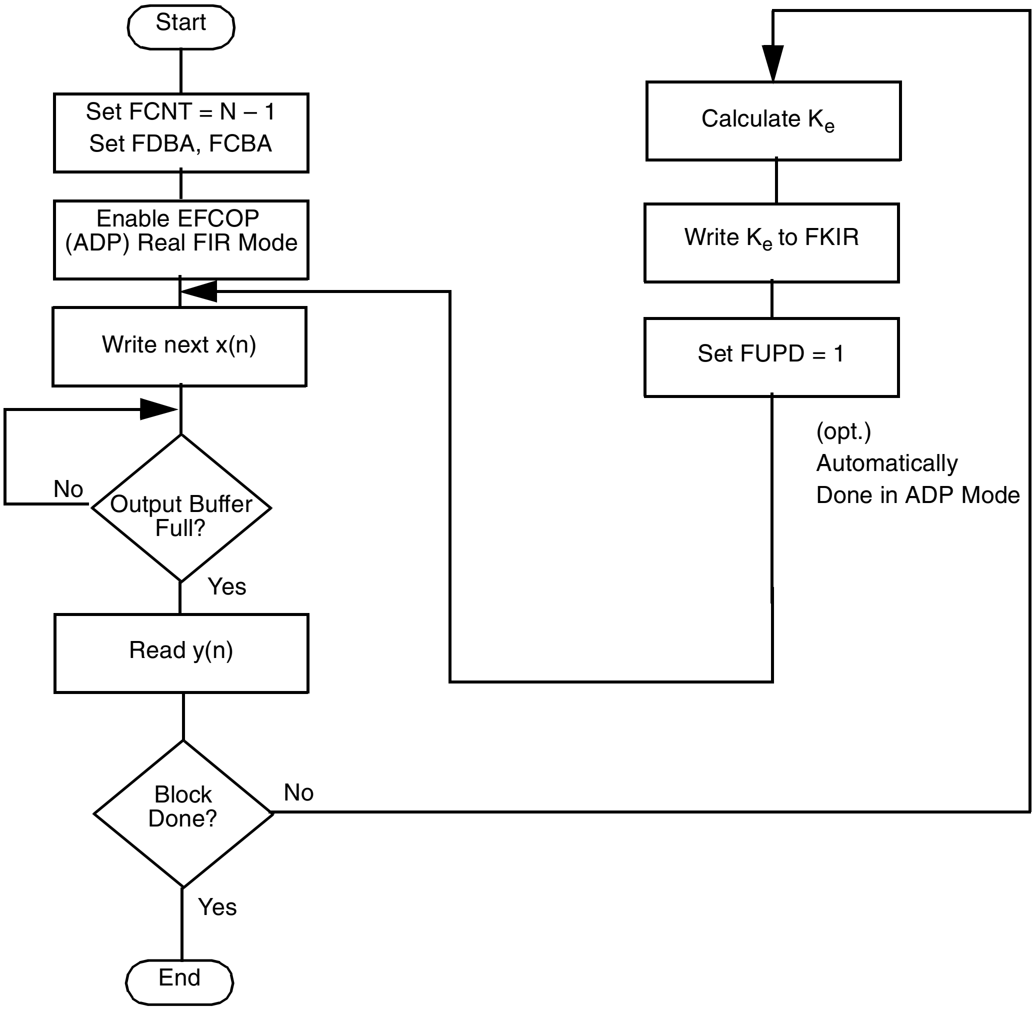


Figure 11-9. Adaptive FIR Filter Using Polling

11.3.6.3.2 Implementation Using DMA Input and Interrupt Output

Figure 11-10 shows a flowchart for an adaptive FIR filter that uses DMA and an interrupt to transfer data.

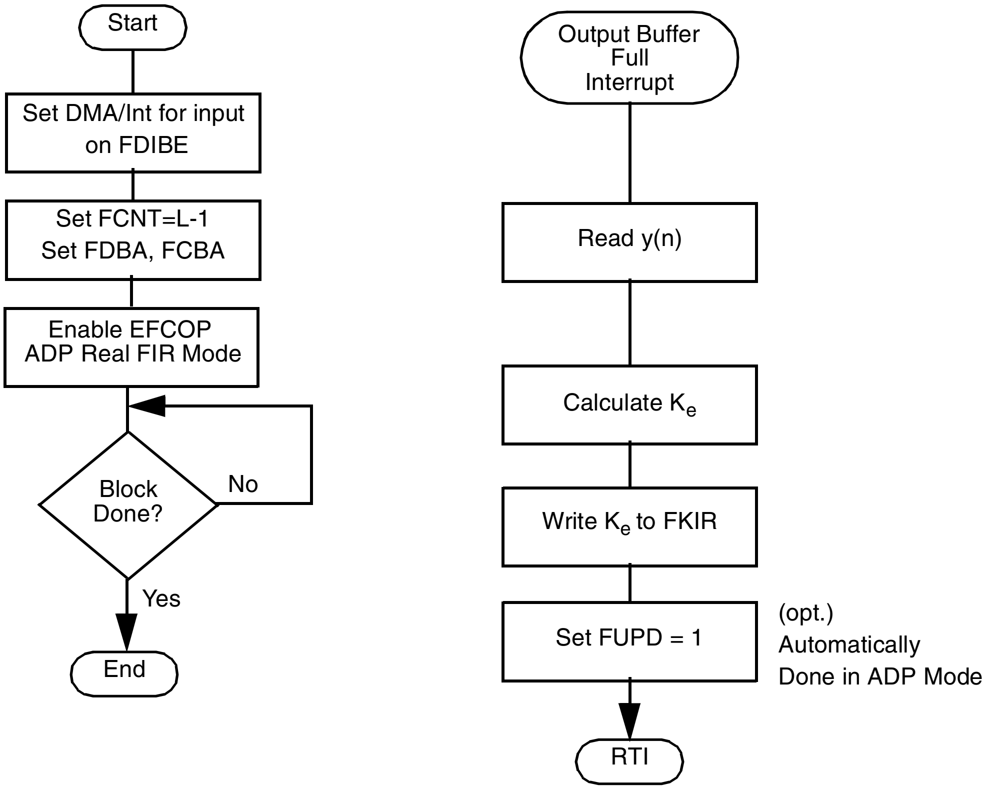


Figure 11-10. Adaptive FIR Filter Using DMA Input and Interrupt Output

11.3.6.3.3 Updating an FIR Filter

The following example shows an FIR adaptive filter that is updated using the LMS algorithm.

Example 11-4. 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         equu    20              ; EFCOP FIR length
DES_ADDRS       equ     $3200           ; Desired signal R(n)
SRC_ADDRS       equ     $3100           ; Reference signal D(n)
DST_ADDRS       equ     $3000           ; 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 (that is 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 initialization **

    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 initialization **

   move    #FDBA_ADDRS,r0           ; FDM memory area
   move    #0,x0
   rep     #FIR_LEN
   move    x0,x:(r0)+               ; clear FDM memory area

; ** address register initialization **

   move    #DST_ADDRS,r0            ; Destination address
   move    #DES_ADDRS,r1            ; Desired signal address

   rep     #FIR_LEN-1
   move    (r1)+                    ; Set reference pointer correctly

; ** EFCOP initialization **

   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 initialization - 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)

;******************************
   movepa1,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)
```

11.3.6.4 Verification for Filter Examples

The following sections provide input and output program listings for the examples given in **Section 11.3.6.1** through **Section 11.3.6.3**.

11.3.6.4.1 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
```

11.3.6.4.2 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
```

11.3.6.4.3 Output Sequence for Examples 11-1, 11-2, and 11-3

```
$d69ea9
$ccae36
$c48f2a
$be8b28
$bad8c5
$b9998c
$bad8cb
$be8b2d
$c7b906
```

11.3.6.4.4 Desired Signal for Example 11-4

```
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
```

11.3.6.4.5 Output Sequence for Example 11-4

```
$000000
$f44c4c
$ee54b3
$e7cd6b
$daed26
```

```
$cc1071
$c7db1c
$cdfe45
```

11.4 EFCOP Programming Model

This section documents the registers for configuring and operating the EFCOP. The EFCOP registers available to the DSP programmer are listed in **Table 11-5**. The following paragraphs describe these registers in detail.

Table 11-5. 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.	

11.4.1 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.

11.4.2 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.

11.4.3 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
```

11.4.4 Filter Count (FCNT) Register

The FCNT register is a read/write register that selects the filter length (number of filter taps). Always write the initial count into the FCNT register before you enable the EFCOP (that is, before you set FEN). The number stored in FCNT is used to generate the correct addressing for the FDM and for the FCM.

Note: To ensure correct operation, never change the contents of the FCNT register unless the EFCOP is in the individual reset state (that is, FEN = 0). In the individual reset state (that is, FEN = 0), the EFCOP module is inactive, but the contents of the FCNT 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**
FCNT11	FCNT10	FCNT9	FCNT8	FCNT7	FCNT6	FCNT5	FCNT4	FCNT3	FCNT2	FCNT1	FCNT0

▨ = Reserved bit; read as 0; write with 0 for future compatibility

Table 11-6. Filter Count FCNT Register Bits

Bit #	Abbr.	Description
23–12	▨	These bits are reserved and unused. They read as 0; write 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.

11.4.5 EFCOP Control Status Register (FCSR)

The FCSR is a read/write register by which the DSP56300 core controls the main operation modes of the EFCOP and monitors the EFCOP status.

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; write with 0 for future compatibility

Table 11-7. FCSR Bits

Bit Number	Bit Name	Reset Value	Description
23–16	▨	0	These bits are reserved and unused. They read as 0; write with 0 for future compatibility.
15	FDOBF	0	**Filter Data Output Buffer Full** When set, this read-only status bit indicates 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. When FDOBF is set, the EFCOP generates an FDOBF interrupt request to the DSP56300 core if that interrupt is enabled (that is, 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. A read from the FDOR clears the FDOBF bit.
14	FDIBE	0	**Filter Data Input Buffer Empty** When set, this read-only status bit indicates 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 FDIBE is set. After the EFCOP is enabled by setting FEN, FDIBE is set, indicating that the FDIR is empty. When FDIBE is set, the EFCOP generates an FDIR empty interrupt request to the DSP56300 core, if enabled (that is, 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. A write to the FDIR clears the FDIBE bit.
13	FCONT	0	**Filter Contention** When set, this read-only status bit indicates an attempt by both the DSP56300 core and the EFCOP to access the same 1024-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 RESET signal, a software RESET instruction, or an individual reset.
12	FSAT	0	**Filter Saturation** When set, this read-only status bit indicates 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 (that is, $7FFFFF). When an underflow occurs, the FSAT bit is set, and the result is saturated to the most negative number (that is, $800000). FSAT is a sticky status bit that is set by hardware and can be cleared only by a hardware RESET signal, a software RESET instruction, or an individual reset.

Table 11-7. FCSR Bits (Continued)

Bit Number	Bit Name	Reset Value	Description
11	FDOIE	0	**Filter Data Output Interrupt Enable** This read/write control bit enables 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. 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	0	**Filter Data Input Interrupt Enable** This read/write control bit enables 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. 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		0	Reserved. It is read as 0 and write with 0 for future compatibility.
8	FSCO	0	**Filter Shared Coefficients Mode** This read/write control bit is valid only when the EFCOP is operating in multichannel mode (that is, FMLC is set). When FSCO is set, the EFCOP uses the coefficients in the same memory area (that is, the same coefficients) to implement the filter for each channel. This mode is used when several channels are filtered through the same filter. When the FSCO bit is cleared, the EFCOP filter coefficients are stored sequentially in memory for each channel. To ensure proper operation, never change the FSCO bit unless the EFCOP is in individual reset state (that is, FEN = 0).
7	FPRC	0	**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.) To ensure proper operation, never change the FPRC bit unless the EFCOP is in individual reset state (that is, FEN = 0).
6	FMLC	0	**Filter Multichannel (FMLC) Mode** This read/write control bit 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. To ensure proper operation, never change the FMLC bit unless the EFCOP is in individual reset state (that is, FEN = 0).

Table 11-7. FCSR Bits (Continued)

Bit Number	Bit Name	Reset Value	Description
5–4	FOM[1–0]	0	**Filter Operation Mode** This pair of read/write control bits defines one of four operation modes if the FIR filter is selected (that is, 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 To ensure proper operation, never change the FOM bits unless the EFCOP is in the individual reset state (that is, FEN = 0).
3	FUPD	0	**Filter Update** 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 automatically set when the EFCOP is in adaptive mode (that is, FADP = 1).
2	FADP	0	**Filter Adaptive (FADP) Mode** This read/write control bit enables adaptive mode. Adaptive mode is 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 FADP 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 is 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.
1	FLT	0	**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 (that is, FEN = 0).
0	FEN	0	**Filter Enable** This read/write control bit enables the operation of the EFCOP. When FEN is cleared, operation is disabled and the EFCOP is in the individual reset state. 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.

11.4.6 EFCOP ALU Control Register (FACR)

The FACR is a read/write register by which the DSP56300 core controls the main operation modes of the EFCOP ALU.

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; write with 0 for future compatibility

▣ = Reserved for internal use; read as 0; write with 0 for proper use.

Table 11-8. EFCOP ALU Control Register (FACR) Bits

Bit Number	Bit Name	Reset Value	Description
23–16		0	Reserved. They read as 0; write with 0 for future compatibility.
15–12		0	Reserved for internal use. Written as 0 for proper operation.
11–7		0	Reserved and unused. They read as 0; write with 0 for future compatibility.
6	FISL	0	**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.
5	FSA	0	**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.
4	FSM	0	**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.
3–2	FRM[1–0]	0	**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.

Table 11-8. EFCOP ALU Control Register (FACR) Bits (Continued)

Bit Number	Bit Name	Reset Value	Description
1–0	FSCL[1–0]	0	**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 To ensure proper operation, never change the FSCL bits unless the EFCOP is in the individual reset state (that is, FEN = 0).

11.4.7 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 (that is, 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.

11.4.8 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 (that is, 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 (that is, 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.

11.4.9 Decimation/Channel Count Register (FDCH)

The FDCH is a read/write register that sets the number of channels used in multichannel mode (FCHL) and sets the decimation ratio in FIR filter mode. FDCH must be written before the FEN enables the EFCOP. FDCH should be changed only when the 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; write with 0 for future compatibility

Table 11-9. Decimation/Channel Count Register (FDCH) Bits

Bit Number	Bit Name	Reset Value	Description
23–12	▨	0	These bits are reserved and unused. They read as 0; write with 0 for future compatibility.
11–8	FDCM[3–0]	0	**Filter Decimation** These read/write control bits select the decimation function. There are 16 decimation factor options (from 1 to 16). To ensure proper operation, never change the FDCM bits unless the EFCOP is in the individual reset state (FEN = 0).
7–6	▨	0	Reserved and unused. They read as 0; write with 0 for future compatibility.
5–0	FCHL[5–0]	0	**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. To ensure proper operation, never change the FCHL bits unless the EFCOP is in the individual reset state (FEN = 0).

11.4.10 EFCOP Interrupt Vectors

Table 11-10 shows the EFCOP interrupt vectors, and **Table 11-11** shows the DMA request sources.

Table 11-10. 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 11-11. 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

Bootstrap Program

A

```
; BOOTSTRAP CODE FOR DSP56321 - (C) Copyright 2000 Freescale Semiconductor, Inc.

; Revised March, 18 1997. And again Dec 1999
;
; Bootstrap through the Host Interface, External EPROM or SCI.
;
; This is the Bootstrap program contained in the DSP56321 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 DSP

; will start fetching instructions  beginning  with address 0xC00000 (MD=0)

; or 0x008000  (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:0xD00000 (bits 7-0).
; The memory is selected by the Address Attribute AA1 and is accessed with
; 31 wait states.
; The EPROM bootstrap code expects to read 3 bytes
; specifying the number of program words, 3 bytes specifying the address
; to start loading the program words and then 3 bytes for each program
; word to be loaded. The number of words, the starting address and the
; program words are read least significant byte first followed by the
; mid and then by the most significant byte.
; The program words  will be condensed into 24-bit words and stored in
; contiguous PRAM memory locations starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1010, then it loads the program RAM from the SCI interface.
; The number of program words to be loaded and the starting address must
; be specified. The SCI bootstrap code expects to receive 3 bytes
; specifying the number of program words, 3 bytes specifying the address
; to start loading the program words and then 3 bytes for each program
; word to be loaded. The number of words, the starting address and the
; program words are received least significant byte first followed by the
; mid and then by the most significant byte. After receiving the
; program words, program execution starts in the same address where
; loading started. The SCI is programmed to work in asynchronous mode
; with 8 data bits, 1 stop bit and no parity. The clock source is
; external and the clock frequency must be 16x the baud rate.
; After each byte is received, it is echoed back through the SCI
; transmitter.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Operation mode MD:MC:MB:MA=1011 is reserved.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1100, then it loads the program RAM from the Host
; Interface programmed to operate in the ISA mode.
; The HOST ISA bootstrap code expects to read a 24-bit word
; specifying the number of program words, a 24-bit word specifying the address
; to start loading the program words and then a 24-bit word for each program
; word to be loaded. The program words  will be stored in
; contiguous PRAM memory locations starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by
; setting the Host Flag 0 (HF0). This will start execution of the loaded
; program from the specified starting address.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1101, then it loads the program RAM from the Host
; Interface programmed to operate in the HC11 non multiplexed mode.
;
```

```
; The HOST HC11  bootstrap code expects to read a 24-bit word
; specifying the number of program words, a 24-bit word specifying the address
; to start loading the program words and then a 24-bit word for each program
; word to be loaded. The program words  will be stored in
; contiguous PRAM memory locations starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by
; setting the Host Flag 0 (HF0). This will start execution of the loaded
; program from the specified starting address.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1110, then it loads the program RAM from the Host
; Interface programmed to operate in the 8051 multiplexed bus mode,
; in double-strob pin configuration.
; The HOST 8051 bootstrap code expects accesses that are byte wide.
; The HOST 8051 bootstrap code expects to read 3 bytes forming a 24-bit word
; specifying the number of program words, 3 bytes forming a 24-bit word
; specifying the address to start loading the program words and then 3 bytes
; forming 24-bit words for each program word to be loaded.
; The program words  will be stored in contiguous PRAM memory locations
; starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0). This will start execution of the loaded program from
; the specified starting address.
;
; The base address of the HI08 in multiplexed mode is 0x80 and is not
; modified by the bootstrap code. All the address lines are enabled
; and should be connected accordingly.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; If MD:MC:MB:MA=1111, then it loads the program RAM from the Host
; Interface programmed to operate in the MC68302 (IMP) bus mode,
; in single-strob pin configuration.
; The HOST MC68302 bootstrap code expects accesses that are byte wide.
; The HOST MC68302 bootstrap code expects to read 3 bytes forming a 24-bit word
; specifying the number of program words, 3 bytes forming a 24-bit word
; specifying the address to start loading the program words and then 3 bytes
; forming 24-bit words for each program word to be loaded.
; The program words  will be stored in contiguous PRAM memory locations
; starting at the specified starting address.
; After reading the program words, program execution starts from the same
; address where loading started.
; The Host Interface bootstrap load program may be stopped by setting the
; Host Flag 0 (HF0). This will start execution of the loaded program from
; the specified starting address.
;
;;;;;;;;;;;;;;;;;;;;; MEMORY EQUATES ;;;;;;;;;;;;;;;;;;;;;;;;
        page    132,55,0,0,0
                opt     mex



EQUALDATA       equ     1       ;; 1 if xram and yram are of equal
                                ;; size and addresses, 0 otherwise.

        if      (EQUALDATA)
start_dram      equ     0       ;; 24k X and Y RAM
end_dram     equ     0x6000   ;; same addresses
        else
start_xram      equ     0       ;; 24k XRAM
end_xram     equ     0x6000
start_yram      equ     0       ;; 24k YRAM
end_yram     equ     0x6000
        endif

start_pram      equ     0       ;; 16k PRAM
end_pram     equ     0x4000
```

```
DATA_START      equ 0x1000       ;; for dma, first 4k are shared memory
PSTART          equ 0
BLOCK           equ 0x400
ONE_BLOCK       equ (BLOCK-1) ;; for dma counter
PBLOCKS         equ (end_pram-PSTART)/BLOCK
DBLOCKS         equ (end_dram-DATA_START)/BLOCK
DMA_YP          equ 0x9882d9 ; de triggered  y+1->p+1
DMA_YX          equ 0x9882d1 ; de triggered  y+1->x+1
DMA_YY          equ 0x9882d5 ; de triggered  y+1->y+1
;; dma equates
M_DSR0   EQU     0xFFFFEF         ; DMA0 Source Address Register
M_DDR0   EQU     0xFFFFEE         ; DMA0 Destination Address Register
M_DCO0   EQU     0xFFFFED         ; DMA0 Counter
M_DCR0   EQU     0xFFFFEC         ; DMA0 Control Register
M_DTD0   EQU     0                ; DMA Channel Transfer Done Status 0
M_DSTR   EQU     0xFFFFF4         ; DMA Status Register

;;
;;;;;;;;;;;;;;;;;;;; GENERAL EQUATES ;;;;;;;;;;;;;;;;;;;;;;;;
;;

BOOT    equ     0xD00000         ; this is the location in P memory
                                ; on the external memory bus
                                ; where the external byte-wide
                                ; EPROM would be located
AARV    equ     0xD00409         ; AAR1 selects the EPROM as CE~
                                ; mapped as P from 0xD00000 to
                                ; 0xDFFFFF, active low

;;
;;;;;;;;;;;;;;;;;;;; DSP I/O REGISTERS ;;;;;;;;;;;;;;;;;;;;;;;;
;;

M_PDRC  EQU     0xFFFFBD         ;; Port C GPIO Data Register
M_PRRC  EQU     0xFFFFBE         ;; Port C Direction Register
```

```
M_SSR   EQU     0xFFFF93          ; SCI Status Register
M_STXL  EQU     0xFFFF95          ; SCI Transmit Data Register (low)
M_SRXL  EQU     0xFFFF98          ; SCI Receive Data Register (low)
M_SCCR  EQU     0xFFFF9B          ; SCI Clock Control Register
M_SCR   EQU     0xFFFF9C          ; SCI Control Register
M_PCRE  EQU     0xFFFF9F          ; Port E Control register
M_AAR1  EQU     0xFFFFF8          ; Address Attribute Register 1
M_HPCR  EQU     0xFFFFC4          ; Host Polarity Control Register
M_HSR   EQU     0xFFFFC3          ; Host Status Rgister
M_HRX   EQU     0xFFFFC6          ; Host Recceive Register
HRDF    EQU     0x0               ; Host Receive Data Full
HF0     EQU     0x3               ; Host Flag 0
HEN     EQU     0x6               ; Host Enable
SCK0    EQU     0x3               ;; SCK0 is bit #3 as GPIO

        ORG PL:0xff0000,PL:0xff0000    ; bootstrap code starts at 0xff0000


START
        clr a                    ; clear a
        movec   omr,a1
        and #0xf,a
        move    a1,n0
        move    #TABLE,r0
TABLE   ;; Table is here because it should actuallly start from 0
        jmp     (r0)+n0
one
        bra     <EPROMLD         ; MD:MC:MB:MA=0001 , load from eprom
two
        bra     <SCILD           ; MD:MC:MB:MA=0010 , load from SCI
three
        bra     <SCILD           ; MD:MC:MB:MA=0011 , load from SCI
four
        bra     <ISAHOSTLD       ; If MD:MC:MB:MA=0100, load from ISA HOST
five
        bra     <HC11HOSTLD      ; If MD:MC:MB:MA=0101, load from HC11 Host
```

```
six
        bra     <I8051HOSTLD    ; If MD:MC:MB:MA=0110, load from 8051 Host
seven
        bra     <BURN           ; If MD:MC:MB:MA=0111, burn in
eight
        nop
nine
        bra     <EPROMLD        ; MD:MC:MB:MA=1001 , load from eprom
ten
        bra     <SCILD          ; MD:MC:MB:MA=1010 , load from SCI
eleven
        bra     <SCILD          ; MD:MC:MB:MA=1011 ,reserved, SCI for now
twelve
        bra     <ISAHOSTLD      ; If MD:MC:MB:MA=1100, load from ISA HOST
thirteen
        bra     <HC11HOSTLD     ; If MD:MC:MB:MA=1101, load from HC11 Host
fourteen
        bra     <I8051HOSTLD    ; If MD:MC:MB:MA=1110, load from 8051 Host
fifteen
        bra     <MC68302HOSTLD  ; 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
```

```
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


;========================================================================
; This is the routine that loads from the SCI.
; MD:MC:MB:MA=1010 - external SCI clock


SCILD
        movep #0x0302,X:M_SCR    ; Configure SCI Control Reg
                                ; mode 2 (10 bit async)
                                ; LSB first, wakeup disabled,
                                ; no wired OR, receiver enable,
                                ; trasmit enable, clk divide by 32,
                                ; positive clk, no recieve interrupt

        movep #0xC000,X:M_SCCR   ; Configure SCI Clock Control Reg
                                ; external tx clock, external rx clock,
                                ; prescaler divide by1, 1x clock
                                ; clock divide ratio = 1

        movep #7,X:M_PCRE       ; Configure SCLK, TXD and RXD
                                ; as SCI pins

        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 #0x0,ccr            ; Clear CCR as if RESET to 0.
        jmp (r1)                ; Then go to starting Prog addr.
```

```
;=========================================================================
; The following modes are reserved, some of which are used for internal testing
; Can be implemented in future.
;
;=========================================================================
; This mode is reserved for internal testing purposes
; MD:MC:MB:MA=0111
;;----------------------------------------------------------
;;
;;  burnin mode
;;
;;  Intended to be used for burn-in test. Wake up from reset
;;  with PINIT set for execution in maximum frequency.
;;  All RAM locations are validated, arithmetic/logic operations
;;  are validated (add, eor) and exercised (mac).
;;  While all tests pass, the SCK0 pin will continue to toggle.
;;  When the test fails the DSP enters DEBUG and stops execution.
;;
;;----------------------------------------------------------
BURN
                ;;  prepare patterns
                clr b
                move    #0xaaaaaa,x0
                move    #0x555555,x1
                move    #0xcccccc,y0

                ;; configure SCK0 as gpio output. PRRC register is cleared at reset.
                movep   b,x:M_PDRC              ;; clear GPIO data register
                bset    #SCK0,x:M_PRRC          ;; Define SCK0 as GPIO output
                                                ;; SCK0 toggles means test pass
                dor #9,burn1
                ;;----------------------------
                ;; test RAM
                ;;----------------------------
;; write pattern to all memory locations
;; assume EQUALDATA
                ;; write x and y memory
                clr a   #start_dram,r0          ;; start of x/y ram
                move    #>end_dram,n0        ;; length of x/y ram
                rep     n0
                mac x0,x1,a x,l:(r0)+           ;; exercise mac, write x/y ram

                ;; write p memory
                clr a   #>start_pram,r2          ;; start of pram
                move    #>end_pram,n2        ;; length of pram
                rep     n2
                move    y0,p:(r2)+              ;; write pram

                ;; check memory contents
;; assume EQUALDATA


                ;; check dram
                clr a   #start_dram,r0          ;; restore pointer, clear a
                do      n0,_loopd
                move    x:(r0),a1               ;; a0=a2=0
                eor     x1,a
                add     a,b                     ;; accumulate error in b
                move    y:(r0)+,a1              ;; a0=a2=0
                eor     x0,a
                add     a,b                     ;; accumulate error in b
_loopd

                ;; check pram
                clr a   #start_pram,r2          ;; restore pointer, clear a
                do      n2,_loopp
                move    p:(r2)+,a1              ;; a0=a2=0
                eor     y0,a
                add     a,b                     ;; accumulate error in b
_loopp
```

```
                ;;-----------------------------------------------------
                ;; toggle pin if no errors, stop execution otherwise.
                ;;-----------------------------------------------------
                beq     label1
                bclr    #SCK0,x:M_PDRC          ;; clear sck0 if error,
                enddo                           ;; terminate the loop normally
                bra     <burn1                  ;; and stop execution
label1                                          ;; if no error
                bchg    #SCK0,x:M_PDRC        ;; toggle pin and keep on looping
;; exercise dma during burn-in
                move #DATA_START,n0
          clr    a #>0x010101,x1
;; fill one block in Y memory as source
          move   n0,r0
          rep    #BLOCK
          add    x1,a   a1,y:(r0)+
        ;; exercise Pram dma data path, no checking of results
                move n0,r0
                move #PSTART,r1
                move #DMA_YP,r2
                move #PBLOCKS,r3
                bsr <do_mem_dma
        ;; exercise Xram dma data path, no checking of results
                move n0,r0
                move #DATA_START,r1
                move #DMA_YX,r2
                move #DBLOCKS,r3
                bsr <do_mem_dma
        ;; exercise Yram dma data path, no checking of results
                move n0,r0
;; no change in r1 or r3
;;              move #PSTART,r1
;;              move #DBLOCKS,r3
                move #DMA_YY,r2
                bsr <do_mem_dma
                nop
                nop
burn1
                wait                         ;; enter wait after test completion

; subroutine to copy source from  DATA_START
;; to dest mem at r1
;; r2 has value for dma control reg
;; r3 has number of iterations (size/block)
do_mem_dma
        movep  r1,x:M_DDR0    ; memory initial address
        dor r3,loopoo
        movep  n0,x:M_DSR0
        movep  #>ONE_BLOCK,x:M_DCO0    ; BLOCK transfers.
        movep r2,x:M_DCR0               ; y->p,y->x or y->y
        ;; time required for transfer is 2*BLOCK+overhead
        ;; overhead=~ 10 cycles
        nop
;; last dma xfer
        rep #(2*BLOCK+10)
        nop
loopoo
        nop
        rts ;; do_mem_dma
```

Programming Reference B

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, Clock Generator (CLKGEN), 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 B-2**, *Internal I/O Memory Map (X Data Memory),* on page B-2 and **Table B-3**, *Internal I/O Memory Map (Y Data Memory),* on page B-7 list the memory addresses of all on-chip peripherals.
- **Table B-4**, *Interrupt Sources,* on page B-7 lists the interrupt starting addresses and sources.
- **Table B-5**, *Interrupt Source Priorities Within an IPL,* on page B-9 lists the priorities of specific interrupts within interrupt priority levels.
- The programming sheets appear in this manual as figures (listed in **Table B-1**); they show the major programmable registers on the DSP56321.

Table B-1. Guide to Programming Sheets

Module	Programming Sheet	Page
Central Processor	**Figure B-1**, *Status Register (SR)*	**page B-11**
	Figure B-2, *Digital Phase-Lock Loop Control (PCTL) Register*	**page B-13**
IPR	**Figure B-3**, *DPLL Static Control Register (DSCR)*	**page B-14**
	Figure B-4, *DMA Control Registers 5–0 (DCR[5–0])*	**page B-17**
CLKGEN	**Figure B-2**, *Digital Phase-Lock Loop Control (PCTL) Register*	**page B-13**
	Figure B-3, *DPLL Static Control Register (DSCR)*	**page B-14**
BIU	**Figure B-7**, *Host Control Register*	**page B-20**
	Figure B-8, *Interrupt Control and Command Vector Registers*	**page B-21**
DMA	**Figure B-4**, *DMA Control Registers 5–0 (DCR[5–0])*	**page B-17**

Table B-1. Guide to Programming Sheets (Continued)

HI08	**Figure B-5**, *Host Transmit Data Register*	**page B-18**
	Figure B-6, *Host Base Address and Host Port Control Registers*	**page B-19**
	Figure B-7, *Host Control Register*	**page B-20**
	Figure B-8, *Interrupt Control and Command Vector Registers*	**page B-21**
	Figure B-9, *Interrupt Vector and Host Transmit Data Registers*	**page B-22**
ESSI	**Figure B-10**, *ESSI Control Register A (CRA)*	**page B-23**
	Figure B-11, *ESSI Control Register B (CRB)*	**page B-24**
	Figure B-12, *ESSI Transmit and Receive Slot Mask Registers (TSM, RSM)*	**page B-25**
SCI	**Figure B-13**, *SCI Control Register (SCR)*	**page B-26**
	Figure B-14, *SCI Clock Control Registers (SCCR)*	**page B-27**
Timers	**Figure B-15**, *Timer Prescaler Load Register (TPLR)*	**page B-28**
	Figure B-16, *Timer Control/Status Register (TCSR)*	**page B-29**
	Figure B-22, *Timer Load, Compare, and Count Registers (TLR, TCPR, TCR)*	**page B-30**
GPIO	**Figure B-17**, *Host Data Direction and Host Data Registers (HDDR, HDR)*	**page B-31**
	Figure B-18, *Port C Registers (PCRC, PRRC, PDRC)*	**page B-32**
	Figure B-19, *Port D Registers (PCRD, PRRD, PDRD)*	**page B-33**
	Figure B-20, *Port E Registers (PCRE, PRRE, PDRE)*	**page B-34**
EFCOP	**Figure B-21**, *EFCOP Counter and Control Status Registers (FCNT and FCSR)*	**page B-35**
	Figure B-22, *EFCOP FACR, FDBA, FCBA, and FDCH Registers*	**page B-36**

B.1 Internal I/O Memory Map

Table B-2. Internal I/O Memory Map (X Data Memory)

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)
	$FFFD	$FFFFFD	Reserved
OnCE	$FFFC	$FFFFFC	OnCE GDB Register (OGDB)
BIU	$FFFB	$FFFFFB	Bus Control Register (BCR)
	$FFFA	$FFFFFA	Reserved
	$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)

Table B-2. Internal I/O Memory Map (X Data Memory) (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
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)
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 B-2. Internal I/O Memory Map (X Data Memory) (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
CLKGEN/DPLL	$FFD1	$FFFFD1	DPLL Control (PCTL) Register
	$FFD0	$FFFFD0	DPLL Static Control Register (DSCR)
	$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
Port C	$FFBF	$FFFFBF	Port C Control Register (PCRC)
	$FFBE	$FFFFBE	Port C Direction Register (PRRC)
	$FFBD	$FFFFBD	Port C GPIO Data Register (PDRC)

Table B-2. Internal I/O Memory Map (X Data Memory) (Continued)

Peripheral	16-Bit Address	24-Bit Address	Register Name
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 D GPIO Data Register (PDRD)
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 B-2. Internal I/O Memory Map (X Data Memory) (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 Receive 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
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 B-3. Internal I/O Memory Map (Y Data Memory)

Peripheral	16-Bit Address	24-Bit Address	Register Name
	$FFBF	$FFFFBF	Reserved
	$FFBE	$FFFFBE	Reserved
	$FFBD	$FFFFBD	Reserved
	$FFBC	$FFFFBC	Reserved
	$FFBB	$FFFFBB	Reserved
	$FFBA	$FFFFBA	Reserved
	$FFB9	$FFFFB9	Reserved
Enhanced Filter Coprocessor (EFCOP)	$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

B.2 Interrupt Sources and Priorities

Table B-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

Table B-4. Interrupt Sources (Continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
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
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

Table B-4. Interrupt Sources (Continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
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

Table B-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

Table B-5. Interrupt Source Priorities Within an IPL (Continued)

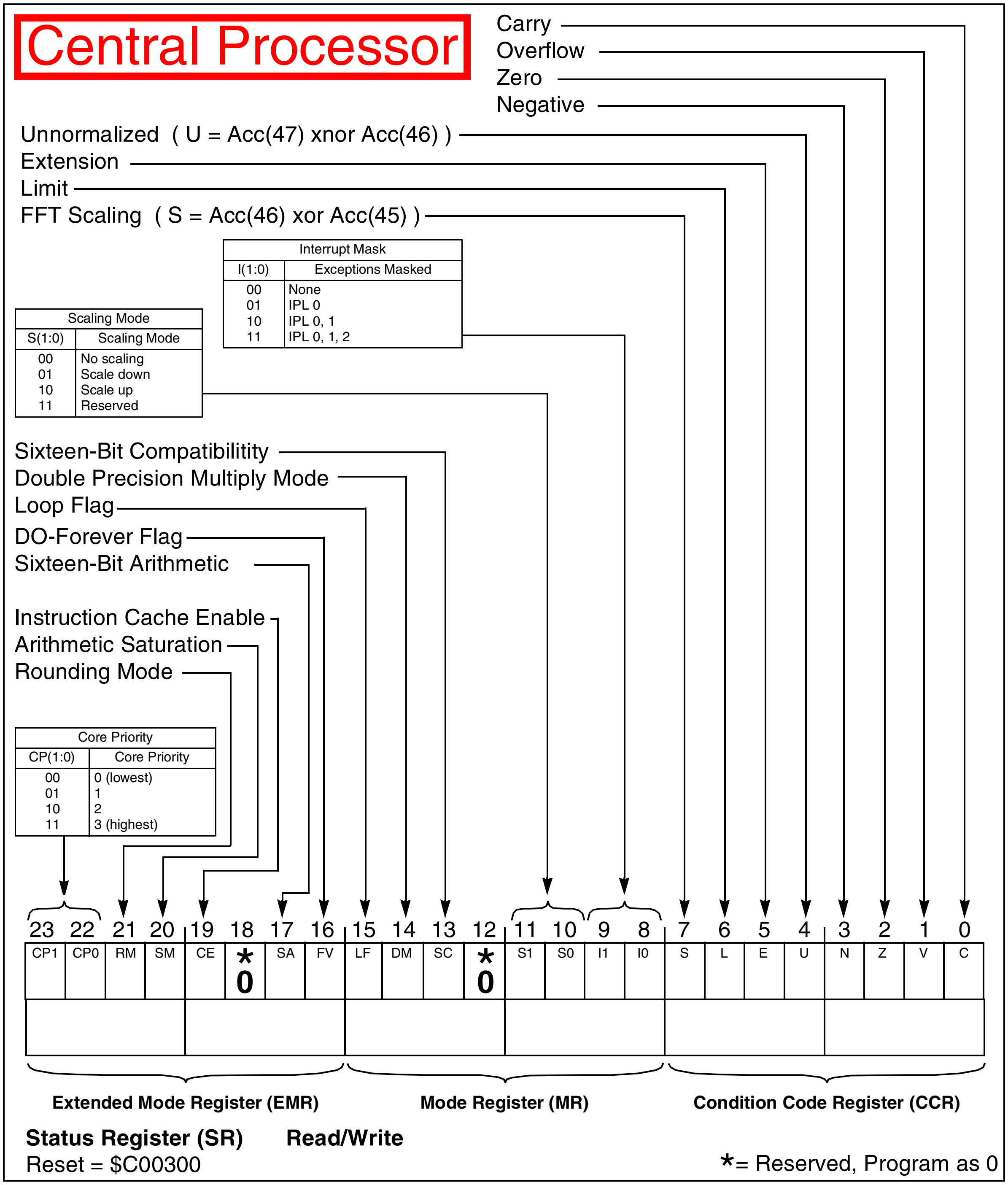
Priority	Interrupt Source
	DMA Channel 4 Interrupt
	DMA Channel 5 Interrupt
	Host Command Interrupt
	Host Transmit Data Empty
	Host Receive Data Full
	ESSI0 RX Data with Exception Interrupt
	ESSI0 RX Data Interrupt
	ESSI0 Receive Last Slot Interrupt
	ESSI0 TX Data With Exception Interrupt
	ESSI0 Transmit 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
	Timer2 Overflow Interrupt
	Timer2 Compare Interrupt
	EFCOP Data Input Buffer Empty
Lowest	EFCOP Data Output Buffer Full

B.3 Programming Sheets

Application:____ Date:____________________

____ Programmer:____________________

Sheet 1 of 2


Central Processor
Carry
Overflow
Zero
Negative
Unnormalized (U = Acc(47) xnor Acc(46))
Extension
Limit
FFT Scaling (S = Acc(46) xor Acc(45))
Sixteen-Bit Compatibilitity
Double Precision Multiply Mode
Loop Flag
DO-Forever Flag
Sixteen-Bit Arithmetic
Instruction Cache Enable
Arithmetic Saturation
Rounding Mode
Extended Mode Register (EMR)
Mode Register (MR)
Condition Code Register (CCR)

Interrupt Mask	
I(1:0)	Exceptions Masked
00	None
01	IPL 0
10	IPL 0, 1
11	IPL 0, 1, 2

Scaling Mode	
S(1:0)	Scaling Mode
00	No scaling
01	Scale down
10	Scale up
11	Reserved

Core Priority	
CP(1:0)	Core Priority
00	0 (lowest)
01	1
10	2
11	3 (highest)

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

Status Register (SR) **Read/Write**
Reset = $C00300

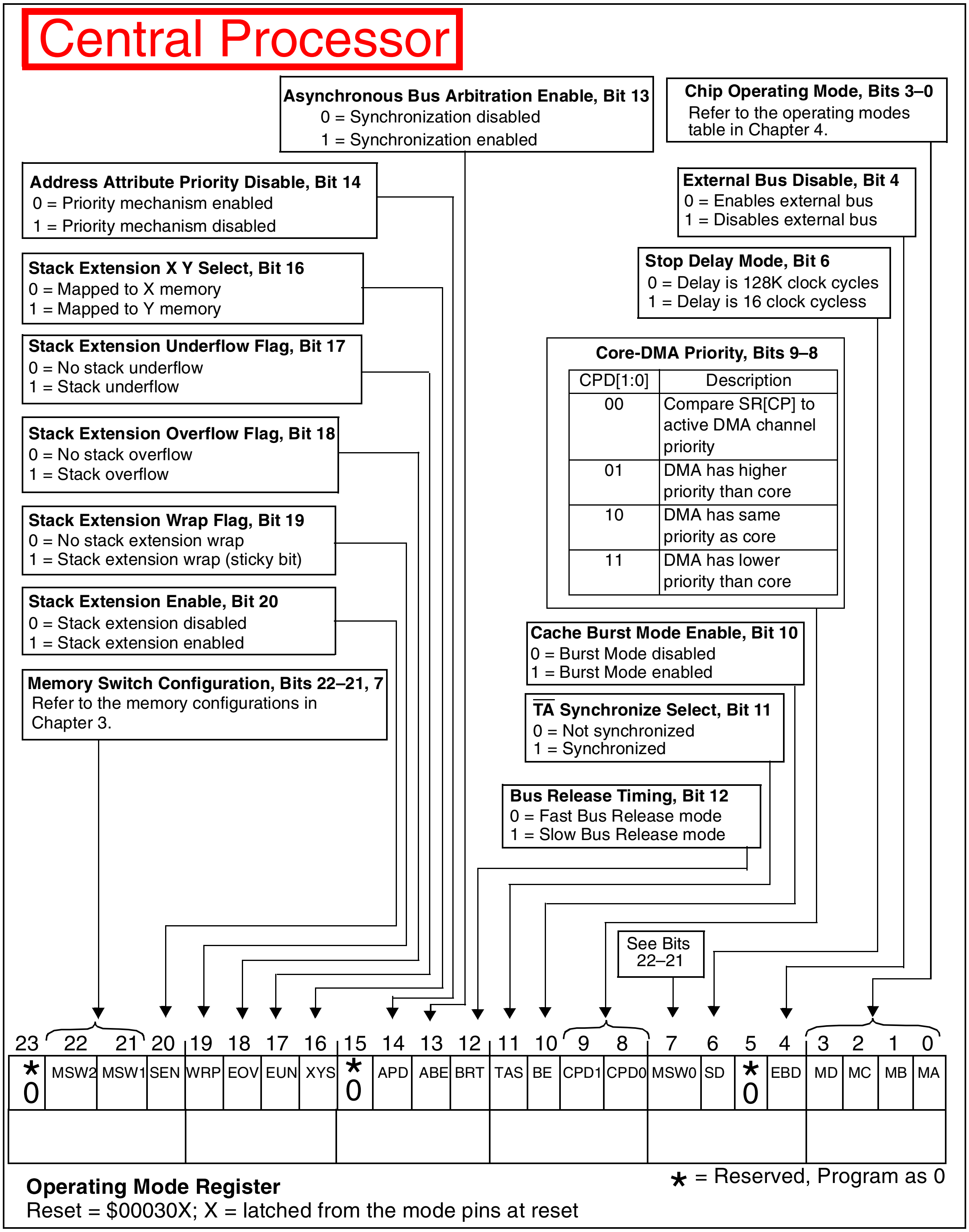
*= Reserved, Program as 0

Figure B-1. Status Register (SR)

Application:____ Date:________________

____ Programmer:________________

Sheet 2 of 2



Central Processor

Asynchronous Bus Arbitration Enable, Bit 13
0 = Synchronization disabled
1 = Synchronization enabled

Chip Operating Mode, Bits 3–0
Refer to the operating modes table in Chapter 4.

Address Attribute Priority Disable, Bit 14
0 = Priority mechanism enabled
1 = Priority mechanism disabled

Stack Extension X Y Select, Bit 16
0 = Mapped to X memory
1 = Mapped to Y memory

Stack Extension Underflow Flag, Bit 17
0 = No stack underflow
1 = Stack underflow

Stack Extension Overflow Flag, Bit 18
0 = No stack overflow
1 = Stack overflow

Stack Extension Wrap Flag, Bit 19
0 = No stack extension wrap
1 = Stack extension wrap (sticky bit)

Stack Extension Enable, Bit 20
0 = Stack extension disabled
1 = Stack extension enabled

Memory Switch Configuration, Bits 22–21, 7
Refer to the memory configurations in Chapter 3.

External Bus Disable, Bit 4
0 = Enables external bus
1 = Disables external bus

Stop Delay Mode, Bit 6
0 = Delay is 128K clock cycles
1 = Delay is 16 clock cycless

Core-DMA Priority, Bits 9–8

CPD[1:0]	Description
00	Compare SR[CP] to active DMA channel priority
01	DMA has higher priority than core
10	DMA has same priority as core
11	DMA has lower priority than core

Cache Burst Mode Enable, Bit 10
0 = Burst Mode disabled
1 = Burst Mode enabled

$\overline{\text{TA}}$ Synchronize Select, Bit 11
0 = Not synchronized
1 = Synchronized

Bus Release Timing, Bit 12
0 = Fast Bus Release mode
1 = Slow Bus Release mode

See Bits 22–21

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	MSW2	MSW1	SEN	WRP	EOV	EUN	XYS	* 0	APD	ABE	BRT	TAS	BE	CPD1	CPD0	MSW0	SD	* 0	EBD	MD	MC	MB	MA

Operating Mode Register
Reset = $00030X; X = latched from the mode pins at reset

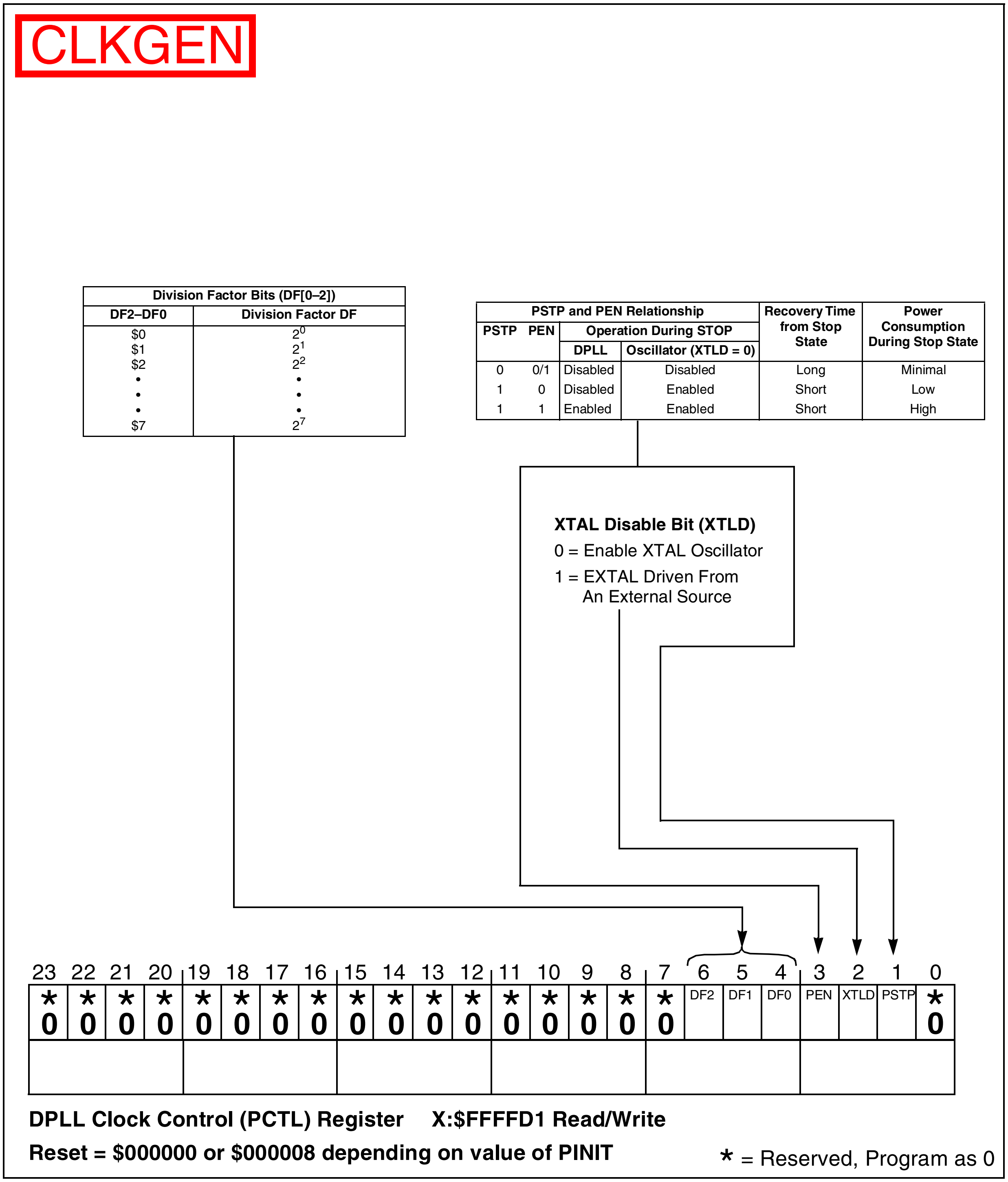
* = Reserved, Program as 0

Figure B-2. Operating Mode Register (OMR)

Application:______________________________________ Date:________________

______________________________________ Programmer:______________

Sheet 1 of 2



CLKGEN

Division Factor Bits (DF[0–2])	
DF2–DF0	Division Factor DF
$0	2^0
$1	2^1
$2	2^2
•	•
•	•
•	•
$7	2^7

PSTP and PEN Relationship				Recovery Time from Stop State	Power Consumption During Stop State
PSTP	PEN	Operation During STOP			
		DPLL	Oscillator (XTLD = 0)		
0	0/1	Disabled	Disabled	Long	Minimal
1	0	Disabled	Enabled	Short	Low
1	1	Enabled	Enabled	Short	High

XTAL Disable Bit (XTLD)

0 = Enable XTAL Oscillator

1 = EXTAL Driven From
An External Source

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	* 0	* 0	* 0	DF2	DF1	DF0	PEN	XTLD	PSTP	* 0

DPLL Clock Control (PCTL) Register X:$FFFFD1 Read/Write

Reset = $000000 or $000008 depending on value of PINIT

* = Reserved, Program as 0

Figure B-2. Digital Phase-Lock Loop Control (PCTL) Register

Application:____ Date:__________________

____ Programmer:________________

Sheet 2 of 2

CLKGEN

Binary Rate Modulation Order (BRMO)

0 = Multiplication Factor Denominator (MFD) < 8

1 = Multiplication Factor Denominator (MFD) ≥ 8

Multiplication Factor Denominator MFD[6–0]

MFD[6–0]	Multiplication Factor Denominator (MFD)
$00	1
$01	2
$02	3
•	•
•	•
•	•
$7E	127
$7F	128

Phase Lock Mode (PLM)

0 = Frequency Only Lock (FOL) mode

1 = Frequency and Phase Lock (FPL) mode

Multiplication Factor Numerator MFN[6–0]

MFN[6–0]	Multiplication Factor Numerator (MFN]
$00	0
$01	1
$02	2
•	•
•	•
•	•
$7E	126
$7F	127

Multiplication Factor Integer (MFI[3–0])

MFI[3–0]	Multiplication Factor Integer (MFI)
$0	5
$1	5
$2	5
$3	5
$4	5
$5	5
$6	6
$7	7
•	•
•	•
•	•
$E	14
$F	15

Predivision Factor Bits (PD[3–0])

PD[3–0]	Predivision Factor (PDF)
$0	1
$1	2
$2	3
•	•
•	•
•	•
$F	16

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRMO	PLM	PDF3	PDF2	PDF1	PDF0	MFD6	MDF5	MFD4	MFD3	MFD2	MFD1	MFD0	MFN6	MFN5	MFN4	MFN3	MFN2	MFN1	MFN0	MFI3	MFI2	MFI1	MFI0

DPLL Static Control Register (DSCR) **X:$FFFFD0 Read/Write**

Reset = $C00008

Note: If MFI = $F, MFN must be $00.

Figure B-3. DPLL Static Control Register (DSCR)

Application:____ Date:________________

____ Programmer:____________

Sheet 1 of 2

Bus Interface Unit

NOTE: All BCR bits are read/write control bits.

Bus Request Hold, Bit 23

0 = BR pin is asserted only for attempted or pending access

1 = BR pin is always asserted

Default Area Wait Control, Bits 20–16

Area 3 Wait Control, Bits 15–13

Area 2 Wait Control, Bits 12–10

Area 1 Wait Control, Bits 9–5

Area 0 Wait Control, Bits 4– 0

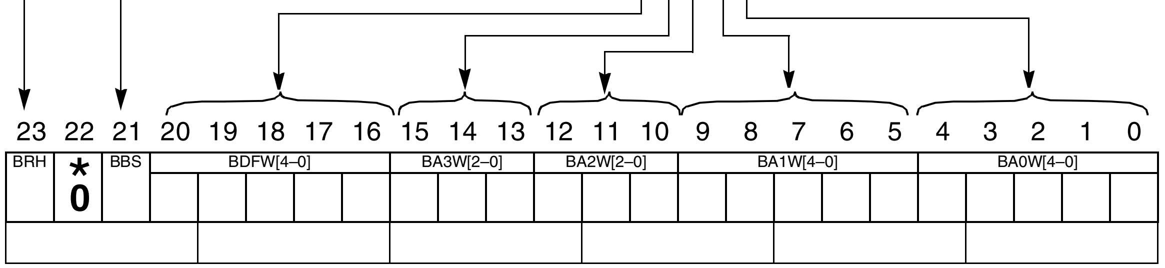
These read/write control bits define the number of wait states inserted into each external SRAM access to the designated area. The value of these bits should not be programmed as zero.

Bits	Bit Name	# of Wait States
20–16	BDFW[4–0]	0–31
15–13	BA3W[2–0]	0–7
12–10	BA2W[2–0]	0–7
9–5	BA1W[4–0]	0–31
4–0	BA0W[4–0]	0–31

Bus State, Bit 21

0 = DSP is not bus master

1 = DSP is bus master



Bus Control Register (BCR) **X:$FFFFFB Read/Write**

Reset = $1FFFFF

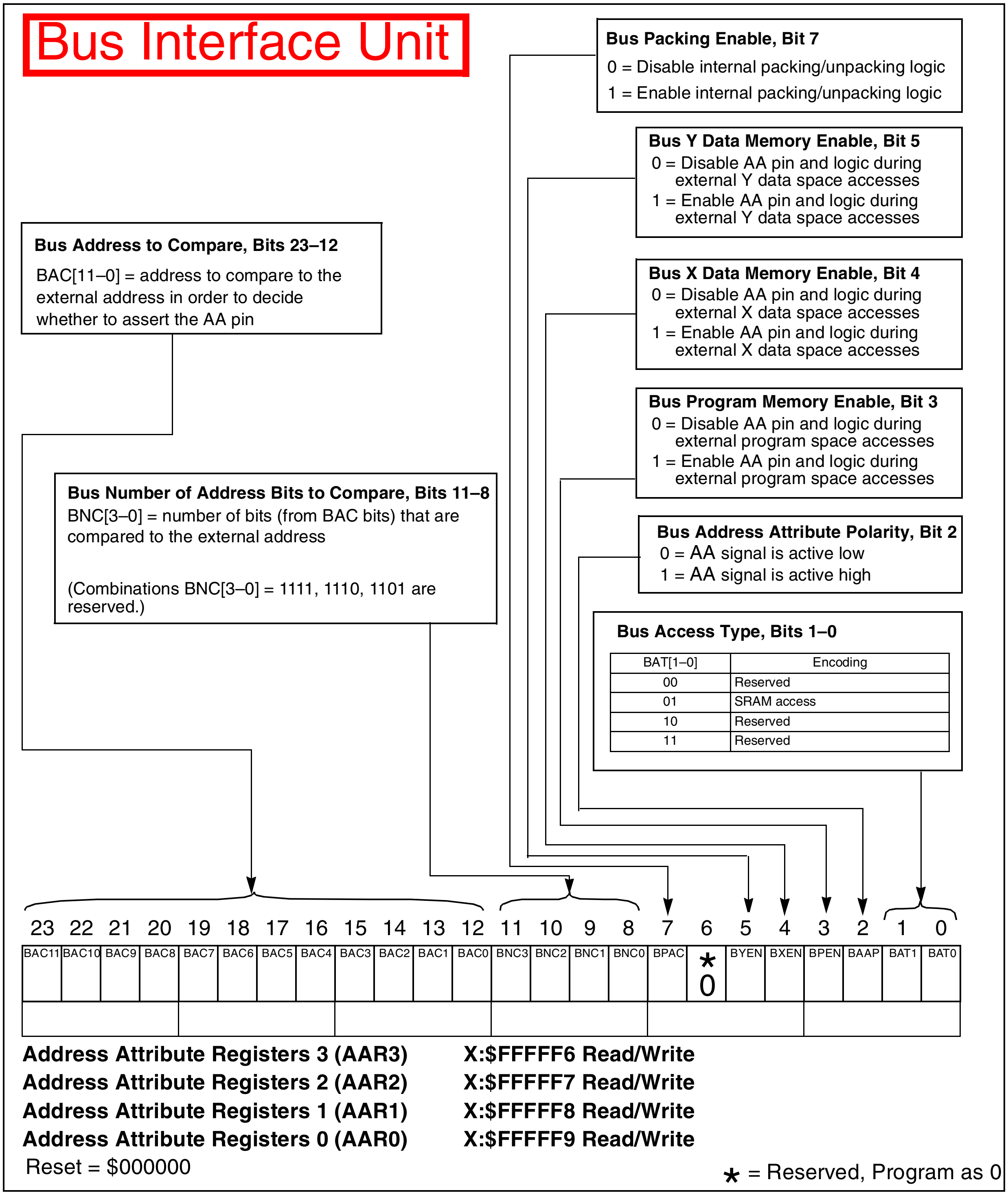
* = Reserved, Program as 0

Figure B-7. Bus Control Register (BCR)

Application:____________________________________ Date:______________

____________________________________ Programmer:______________

Sheet 2 of 2



Bus Interface Unit

Bus Packing Enable, Bit 7
0 = Disable internal packing/unpacking logic
1 = Enable internal packing/unpacking logic

Bus Y Data Memory Enable, Bit 5
0 = Disable AA pin and logic during external Y data space accesses
1 = Enable AA pin and logic during external Y data space accesses

Bus X Data Memory Enable, Bit 4
0 = Disable AA pin and logic during external X data space accesses
1 = Enable AA pin and logic during external X data space accesses

Bus Program Memory Enable, Bit 3
0 = Disable AA pin and logic during external program space accesses
1 = Enable AA pin and logic during external program space accesses

Bus Address Attribute Polarity, Bit 2
0 = AA signal is active low
1 = AA signal is active high

Bus Access Type, Bits 1–0

BAT[1–0]	Encoding
00	Reserved
01	SRAM access
10	Reserved
11	Reserved

Bus Address to Compare, Bits 23–12

BAC[11–0] = address to compare to the external address in order to decide whether to assert the AA pin

Bus Number of Address Bits to Compare, Bits 11–8

BNC[3–0] = number of bits (from BAC bits) that are compared to the external address

(Combinations BNC[3–0] = 1111, 1110, 1101 are reserved.)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAC11	BAC10	BAC9	BAC8	BAC7	BAC6	BAC5	BAC4	BAC3	BAC2	BAC1	BAC0	BNC3	BNC2	BNC1	BNC0	BPAC	* 0	BYEN	BXEN	BPEN	BAAP	BAT1	BAT0

Address Attribute Registers 3 (AAR3) **X:$FFFFF6 Read/Write**
Address Attribute Registers 2 (AAR2) **X:$FFFFF7 Read/Write**
Address Attribute Registers 1 (AAR1) **X:$FFFFF8 Read/Write**
Address Attribute Registers 0 (AAR0) **X:$FFFFF9 Read/Write**
Reset = $000000

* = Reserved, Program as 0

Figure B-8. Address Attribute Registers (AAR[3–0])

Application:____

____

Date:________________

Programmer:____________

Sheet 1 of 1

DMA

DMA Channel Enable, Bit 23
0 = Disables channel operation
1 = Enables channel operation

Three-Dimensional Mode, Bit 10
0 = Three-Dimensional mode disabled
1 = Three-Dimensional mode enabled

DMA Interrupt Enable, Bit 22
0 = Disables DMA Interrupt
1 = Enables DMA interrupt

DMA Transfer Mode, Bits 21–19

DTM[2:0]	Triggered By	DE Cleared	Transfer Mode
000	request	yes	block transfer
001	request	yes	word transfer
010	request	yes	line transfer
011	DE	yes	block transfer
100	request	no	block transfer
101	request	no	word transfer
110–111	reserved		

DMA Channel Priority, Bits 18–17

DPR[1:0]	Channel Priority
00	Priority level 0 (lowest)
01	Priority level 1
10	Priority level 2
11	Priority level 3 (highest)

DMA Continuous Mode Enable, Bit 16
0 = Disables continuous mode
1 = Enables continuous mode

DMA Request Source, Bits 15–11

DRS[4:0]	Requesting Device
00000–00011	External (IRQA, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$, $\overline{\text{IRQD}}$)
00100–01001	Transfer done from channel 0,1,2,3,4,5
01010–01011	ESSI0 Receive, Transmit Data
01100–01101	ESSI1 Receive, Transmit Data
01110–01111	SCI Receive, Transmit Data
10000–10010	Timer0, Timer1, Timer2
10011	Host Receive Data Full
10100	Host Transmit Data Empty
10101	EFCOP FDIBE = 1
10110	EFCOP FDOBF = 1
10111 - 11111	Reserved

DMA Address Mode, Bits 9–4

Non-Three-Dimensional Addressing Modes (D3D=0)

DAM[2–0] = source DAM[5–3] = Destination

DAM[5–3] DAM[2–0]	Addressing Mode	Counter Mode	Offset Register Selection
000	2D	B	DOR0
001	2D	B	DOR1
010	2D	B	DOR2
011	2D	B	DOR3
100	No update	A	None
101	Postincrement-by-1	A	None
110–111	reserved		

Three-Dimensional Addressing Modes (D3D=1)

DAM[5–3]	Addressing Mode	Offset Selection
000	2D	DOR0
001	2D	DOR1
010	2D	DOR2
011	2D	DOR3
100	No update	None
101	Postincrement-by-1	None
110	3D	DOR[0–1]
111	3D	DOR[2–3]

DAM2	Addressing Mode	Offset Selection
0	Source: 3D	Source: DOR[0–1]
	Destination: Defined by DAM[5–3]	
1	Source: Defined by DAM[5–3]	
	Destination: 3D	Destination: DOR[2–3]

DAM [1–0]	Counter	DCO Layout		
00	Mode C	DCOH[23–12]	DCOM[11–6]	DCOL[5–0]
01	Mode D	DCOH[23–18]	DCOM[17–6]	DCOL[5–0]
10	Mode E	DCOH[23–18]	DCOM[17–12]	DCOL[11–0]
11	—	Reserved		

DMA Destination Space, Bits 3–2

DSS[1:0]	DMA Destination Memory
00	X Memory Space
01	Y Memory Space
10	P Memory Space
11	Reserved

DMA Source Space, Bits 1–0

DSS[1:0]	DMA Source Memory
00	X Memory Space
01	Y Memory Space
10	P Memory Space
11	Reserved

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DE	DIE	DTM[2–0]			DPR[1–0]		DCON	DRS[4–0]					D3D	DAM[5–0]						DDS[1–0]		DSS[1–0]	

DMA Control Registers (DCR5–DCR0) **X:$FFFFD8, X:$FFFFDC, X:$FFFFE0,**
Reset = $000000 **X:$FFFFE4, X:$FFFFE8, X:$FFFFEC Read/Write**

Figure B-4. DMA Control Registers 5–0 (DCR[5–0])

Application:____ Date:_______________

____ Programmer:__________

Sheet 1 of 5

HOST

Host Transmit Data (usually Loaded by program)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Transmit High Byte								Transmit Middle Byte								Transmit Low Byte							

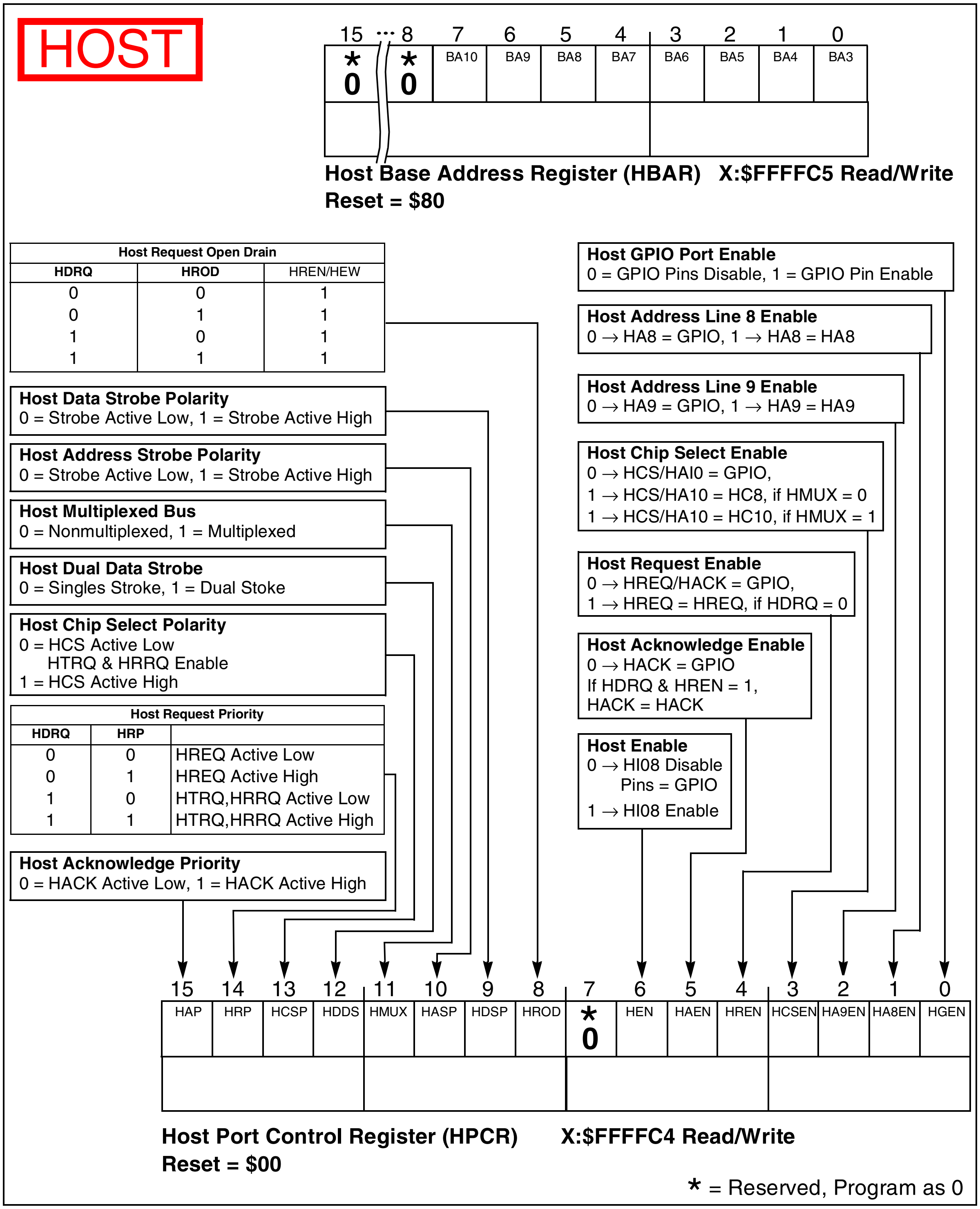
Host Transmit Data Register (HTX) X:$FFFFC7 Write Only
Reset = empty

Figure B-5. Host Transmit Data Register

Application:____ Date:______________

____ Programmer:____________

Sheet 2 of 5



Host Base Address Register (HBAR) X:$FFFFC5 Read/Write
Reset = $80

Host Request Open Drain		
HDRQ	HROD	HREN/HEW
0	0	1
0	1	1
1	0	1
1	1	1

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 = Singles Stroke, 1 = Dual Stoke

Host Chip Select Polarity
0 = HCS Active Low
HTRQ & HRRQ Enable
1 = HCS Active High

Host Request Priority		
HDRQ	HRP	
0	0	HREQ Active Low
0	1	HREQ Active High
1	0	HTRQ,HRRQ Active Low
1	1	HTRQ,HRRQ Active High

Host Acknowledge Priority
0 = HACK Active Low, 1 = HACK Active High

Host GPIO Port Enable
0 = GPIO Pins Disable, 1 = GPIO Pin Enable

Host Address Line 8 Enable
0 → HA8 = GPIO, 1 → HA8 = HA8

Host Address Line 9 Enable
0 → HA9 = GPIO, 1 → HA9 = HA9

Host Chip Select Enable
0 → HCS/HAI0 = GPIO,
1 → HCS/HA10 = HC8, if HMUX = 0
1 → HCS/HA10 = HC10, if HMUX = 1

Host Request Enable
0 → HREQ/HACK = GPIO,
1 → HREQ = HREQ, if HDRQ = 0

Host Acknowledge Enable
0 → HACK = GPIO
If HDRQ & HREN = 1,
HACK = HACK

Host Enable
0 → HI08 Disable
Pins = GPIO
1 → HI08 Enable

Host Port Control Register (HPCR) X:$FFFFC4 Read/Write
Reset = $00

* = Reserved, Program as 0

Figure B-6. Host Base Address and Host Port Control Registers

Application:____ Date:________________

____ Programmer:__________

Sheet 3 of 5

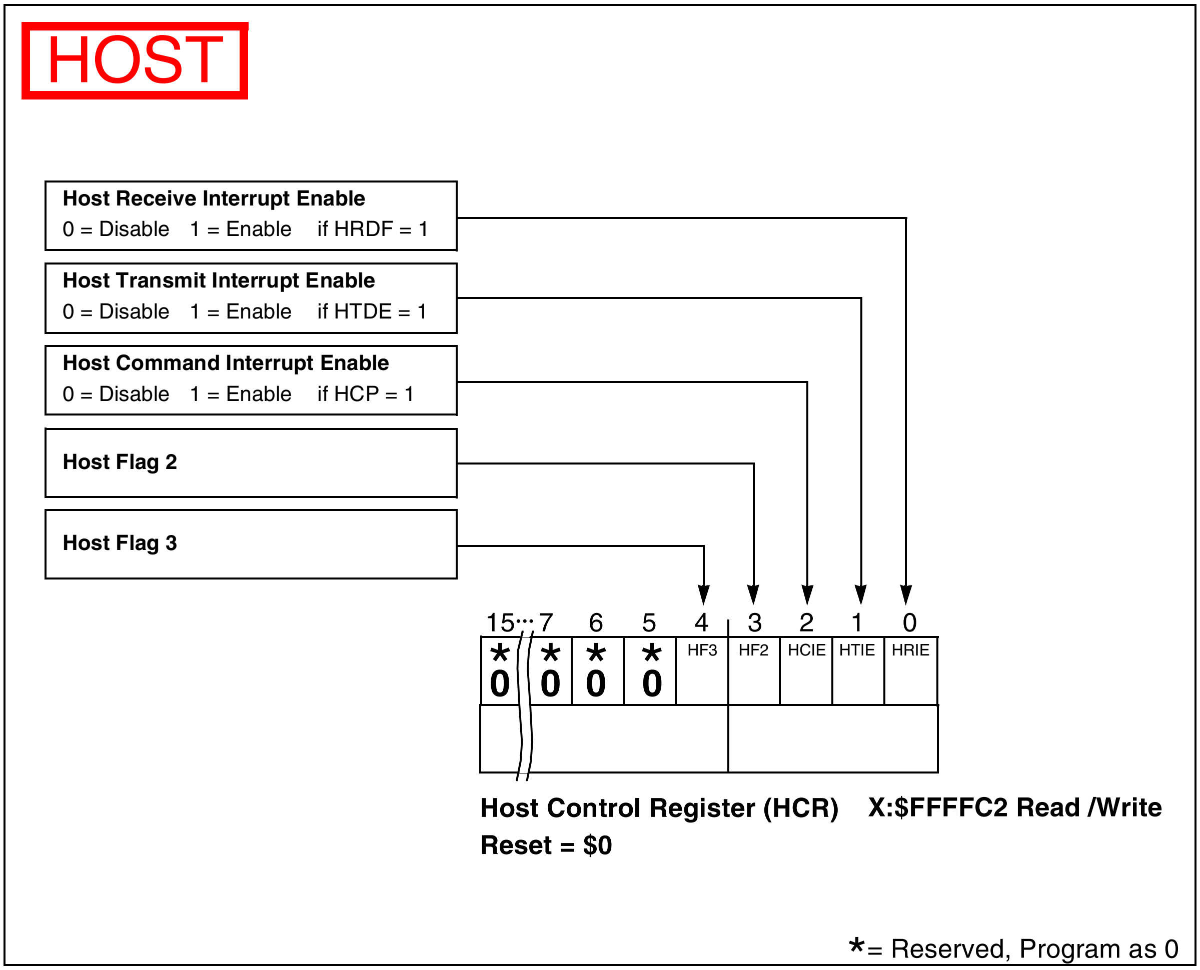

HOST
Host Receive Interrupt Enable
0 = Disable 1 = Enable if HRDF = 1
Host Transmit Interrupt Enable
0 = Disable 1 = Enable if HTDE = 1
Host Command Interrupt Enable
0 = Disable 1 = Enable if HCP = 1
Host Flag 2
Host Flag 3
15 ··· 7 6 5 4 3 2 1 0
* 0 * 0 * 0 * 0
HF3 HF2 HCIE HTIE HRIE
Host Control Register (HCR) X:$FFFFC2 Read /Write
Reset = $0
* = Reserved, Program as 0

Figure B-7. Host Control Register

Application:____ Date:________________

____ Programmer:________________

Sheet 4 of 5

HOST

Host Side

Receive Request Enable

DMA Off	0 = Interrupts Disabled	1 = Interrupts Enabled
DMA On	0 = Host → DSP	1 = DSP → Host

Transmit Request Enable

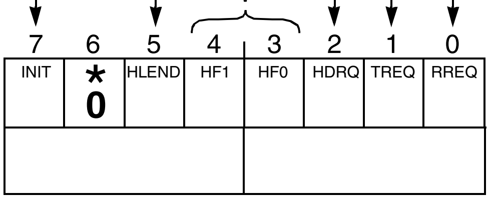
DMA Off	0 = Interrupts Disabled	1 = Interrupts Enabled
DMA On	0 = DSP → Host	1 = Host → DSP

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

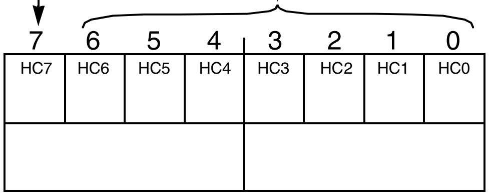


Interrupt Control Register (ICR) Host Address: $0 Read/Write
Reset = $00

Host Vector
Contains Host Command Interrupt Address ÷ 2

Host Command
Handshakes Executing Host Command Interrupts

Contains the host command interrupt address



Command Vector Register (CVR) Host Address: $1 Read/Write
Reset = $32

*= Reserved, Program as 0

Figure B-8. Interrupt Control and Command Vector Registers

Application:____ Date:________________

____ Programmer:____________

Sheet 5 of 5

HOST

Host Side

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

Contains the interrupt vector or number

Interrupt Vector Register (IVR) **Host Address: $3 Read/Write**
Reset = $0F

Host Transmit Data (usually loaded by program)

7 … 0	7 … 0	7 … 0	7 … 0
Transmit Low Byte	Transmit Middle Byte	Transmit High Byte	Not Used
			0 0 0 0 0 0 0 0
$7	$6	$5	$4

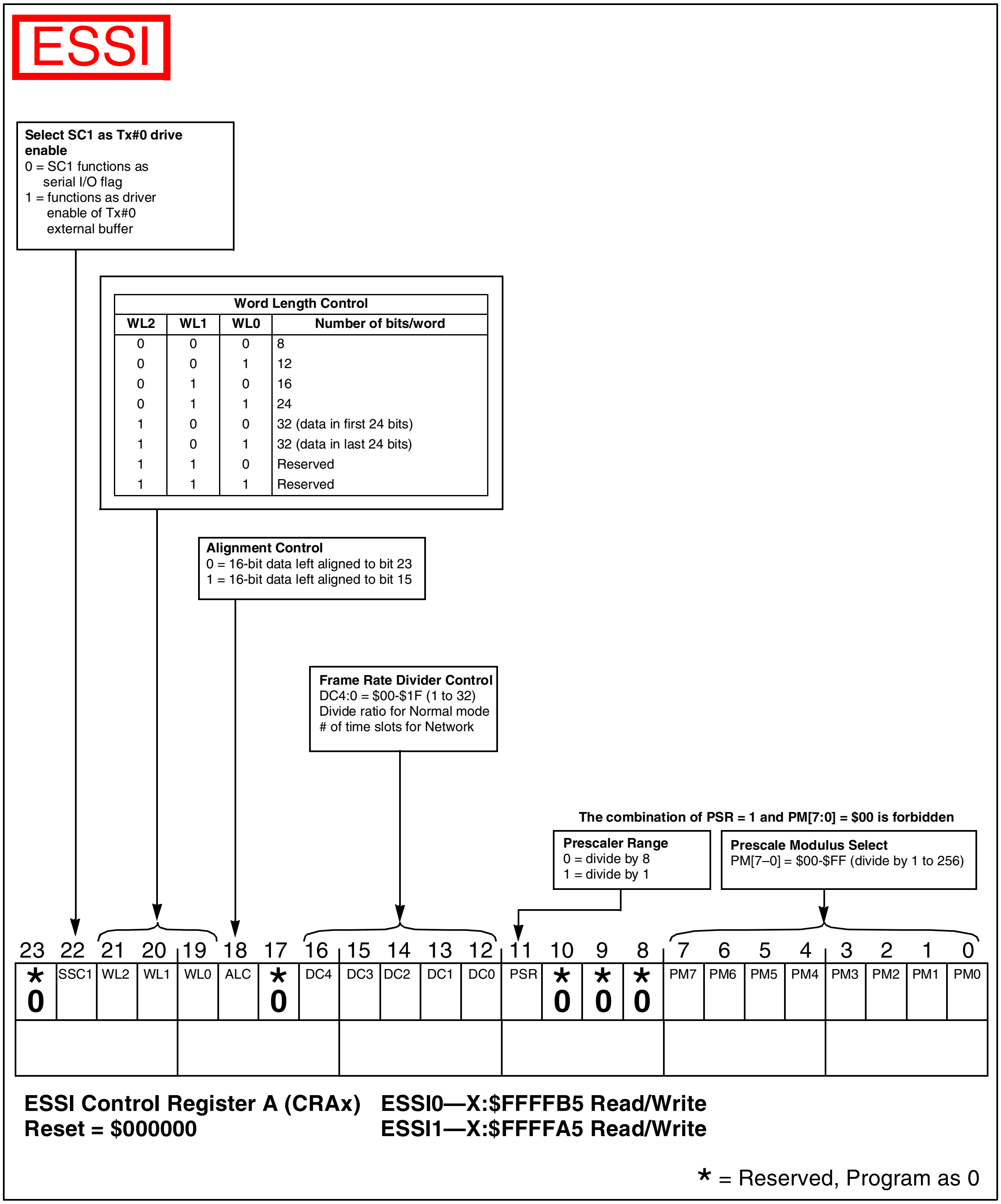
Transmit Byte Registers **Host Addresses: $7, $6, $5, $4 Write Only**
Reset = $00

Figure B-9. Interrupt Vector and Host Transmit Data Registers

Application:______________________________________ Date:________________

______________________________________ Programmer:________________

Sheet 1 of 3



ESSI

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

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

Alignment Control
0 = 16-bit data left aligned to bit 23
1 = 16-bit data left aligned to bit 15

Frame Rate Divider Control
DC4:0 = $00-$1F (1 to 32)
Divide ratio for Normal mode
# of time slots for Network

The combination of PSR = 1 and PM[7:0] = $00 is forbidden

Prescaler Range
0 = divide by 8
1 = divide by 1

Prescale Modulus Select
PM[7–0] = $00-$FF (divide by 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

ESSI Control Register A (CRAx) **ESSI0—X:$FFFFB5 Read/Write**
Reset = $000000 **ESSI1—X:$FFFFA5 Read/Write**

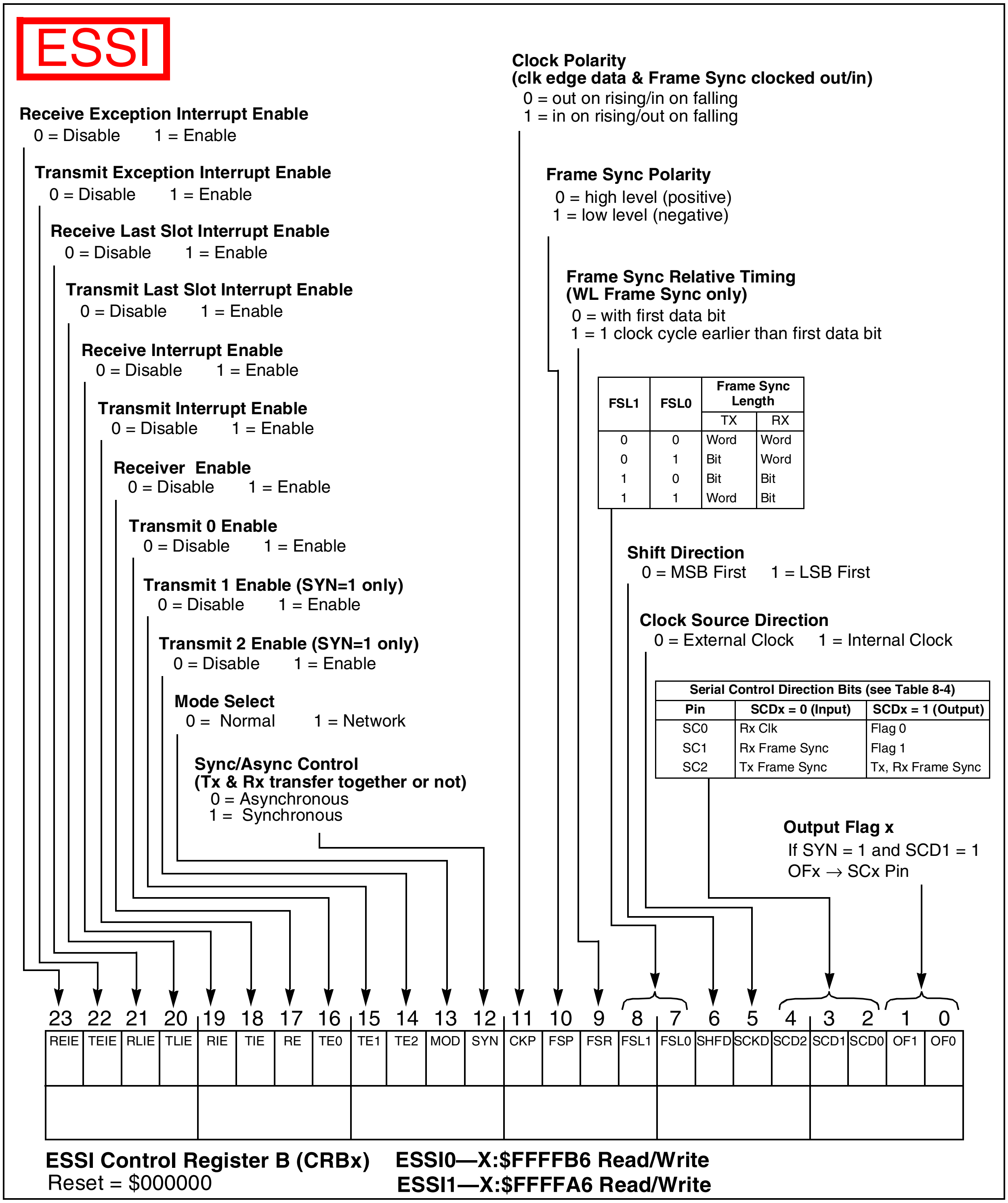
* = Reserved, Program as 0

Figure B-10. ESSI Control Register A (CRA)

Application:__________________________________ Date:______________

__________________________________ Programmer:__________

Sheet 2 of 3



ESSI

Receive Exception Interrupt Enable
0 = Disable 1 = Enable

Transmit Exception Interrupt Enable
0 = Disable 1 = Enable

Receive Last Slot Interrupt Enable
0 = Disable 1 = Enable

Transmit Last Slot Interrupt Enable
0 = Disable 1 = Enable

Receive Interrupt Enable
0 = Disable 1 = Enable

Transmit Interrupt Enable
0 = Disable 1 = Enable

Receiver Enable
0 = Disable 1 = Enable

Transmit 0 Enable
0 = Disable 1 = Enable

Transmit 1 Enable (SYN=1 only)
0 = Disable 1 = Enable

Transmit 2 Enable (SYN=1 only)
0 = Disable 1 = Enable

Mode Select
0 = Normal 1 = Network

Sync/Async Control
(Tx & Rx transfer together or not)
0 = Asynchronous
1 = Synchronous

Clock Polarity
(clk edge data & Frame Sync clocked out/in)
0 = out on rising/in on falling
1 = in on rising/out on falling

Frame Sync Polarity
0 = high level (positive)
1 = low level (negative)

Frame Sync Relative Timing
(WL Frame Sync only)
0 = with first data bit
1 = 1 clock cycle earlier than first data bit

FSL1	FSL0	Frame Sync Length	
		TX	RX
0	0	Word	Word
0	1	Bit	Word
1	0	Bit	Bit
1	1	Word	Bit

Shift Direction
0 = MSB First 1 = LSB First

Clock Source Direction
0 = External Clock 1 = Internal Clock

Serial Control Direction Bits (see Table 8-4)		
Pin	**SCDx = 0 (Input)**	**SCDx = 1 (Output)**
SC0	Rx Clk	Flag 0
SC1	Rx Frame Sync	Flag 1
SC2	Tx Frame Sync	Tx, Rx Frame Sync

Output Flag x
If SYN = 1 and SCD1 = 1
OFx → SCx Pin

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

ESSI Control Register B (CRBx) **ESSI0—X:$FFFFB6 Read/Write**
Reset = $000000 **ESSI1—X:$FFFFA6 Read/Write**

Figure B-11. ESSI Control Register B (CRB)

Application:______________________________________ Date:________________

______________________________________ Programmer:____________

Sheet 3 of 3

ESSI

SSI Transmit Slot Mask
0 = IgnoreTime Slot
1 = Active Time Slot

23	···	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0		* 0	TS15	TS14	TS13	TS12	TS11	TS10	TS9	TS8	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0

ESSI Transmit Slot Mask A (TSMA[0–1]) **ESSI0—X:$FFFFB4 Read/Write**
Reset = $FFFF **ESSI1—X:$FFFFA4 Read/Write**

SSI Transmit Slot Mask
0 = IgnoreTime Slot
1 = ActiveTime Slot

23	···	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0		* 0	TS31	TS30	TS29	TS28	TS27	TS26	TS25	TS24	TS23	TS22	TS21	TS20	TS19	TS18	TS17	TS16

ESSI Transmit Slot Mask B (TSMB[0–1]) **ESSI0—X:$FFFFB3 Read/Write**
Reset = $FFFF **ESSI1—X:$FFFFA3 Read/Write**

SSI Receive Slot Mask
0 = IgnoreTime Slot
1 = ActiveTime Slot

23	···	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0		* 0	RS15	RS14	RS13	RS12	RS11	RS10	RS9	RS8	RS7	RS6	RS5	RS4	RS3	RS2	RS1	RS0

ESSI Receive Slot Mask A (RSMA[0–1] **ESSI0—X:$FFFFB2 Read/Write**
Reset = $FFFF **ESSI1—X:$FFFFA2 Read/Write**

SSI 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
* 0		* 0	RS31	RS30	RS29	RS28	RS27	RS26	RS25	RS24	RS23	RS22	RS21	RS20	RS19	RS18	RS17	RS16

ESSI Receive Slot Mask B (RSMB[0–1]) **ESSI0—X:$FFFFB1 Read/Write**
Reset = $FFFF **ESSI1—X:$FFFFA1 Read/Write**

* = Reserved, Program as 0

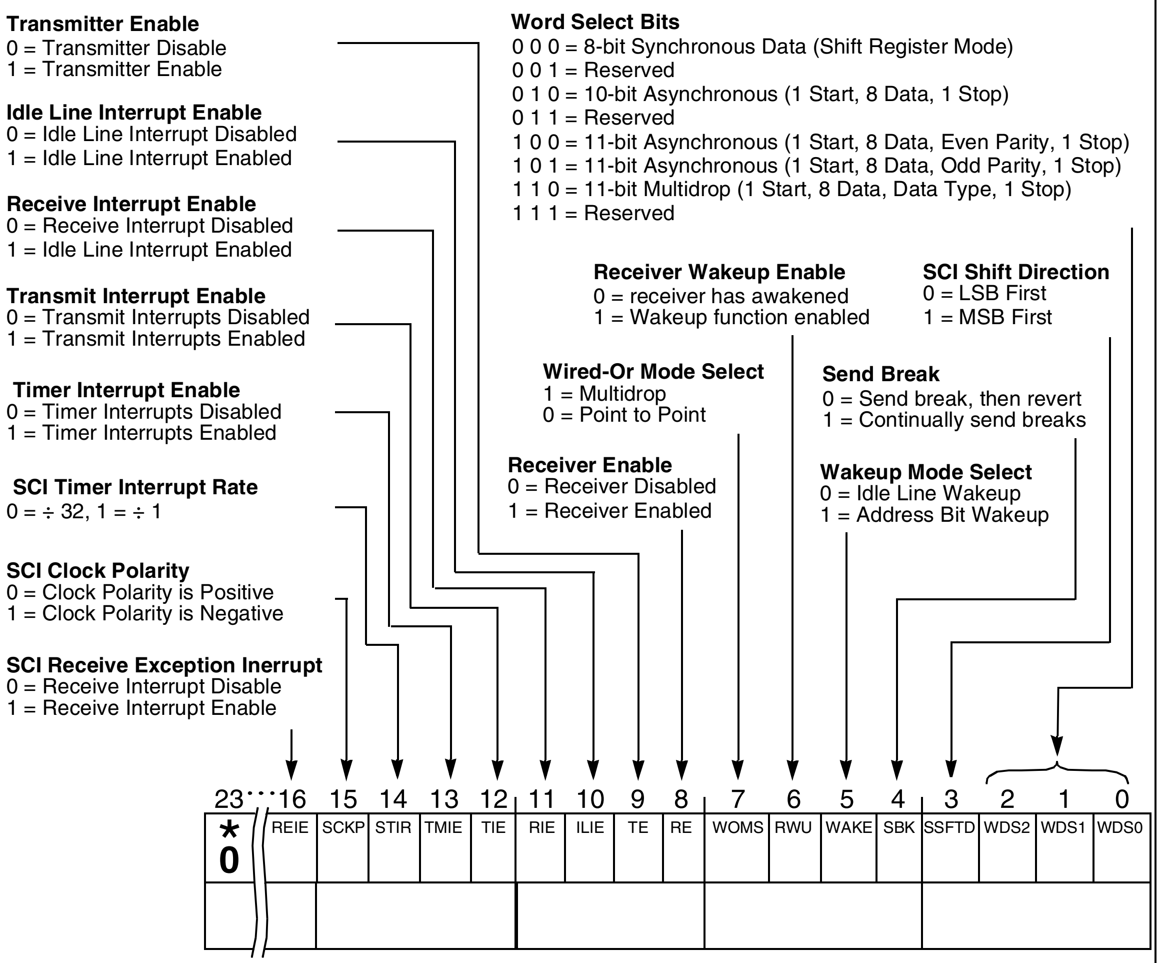
Figure B-12. ESSI Transmit and Receive Slot Mask Registers (TSM, RSM)

Application:____ Date:________________

____ Programmer:________________

Sheet 1 of 2

SCI



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

23···16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	REIE	SCKP	STIR	TMIE	TIE	RIE	ILIE	TE	RE	WOMS	RWU	WAKE	SBK	SSFTD	WDS2	WDS1	WDS0

SCI Control Register (SCR) **X:$FFFF9C Read/Write**
Reset $000000

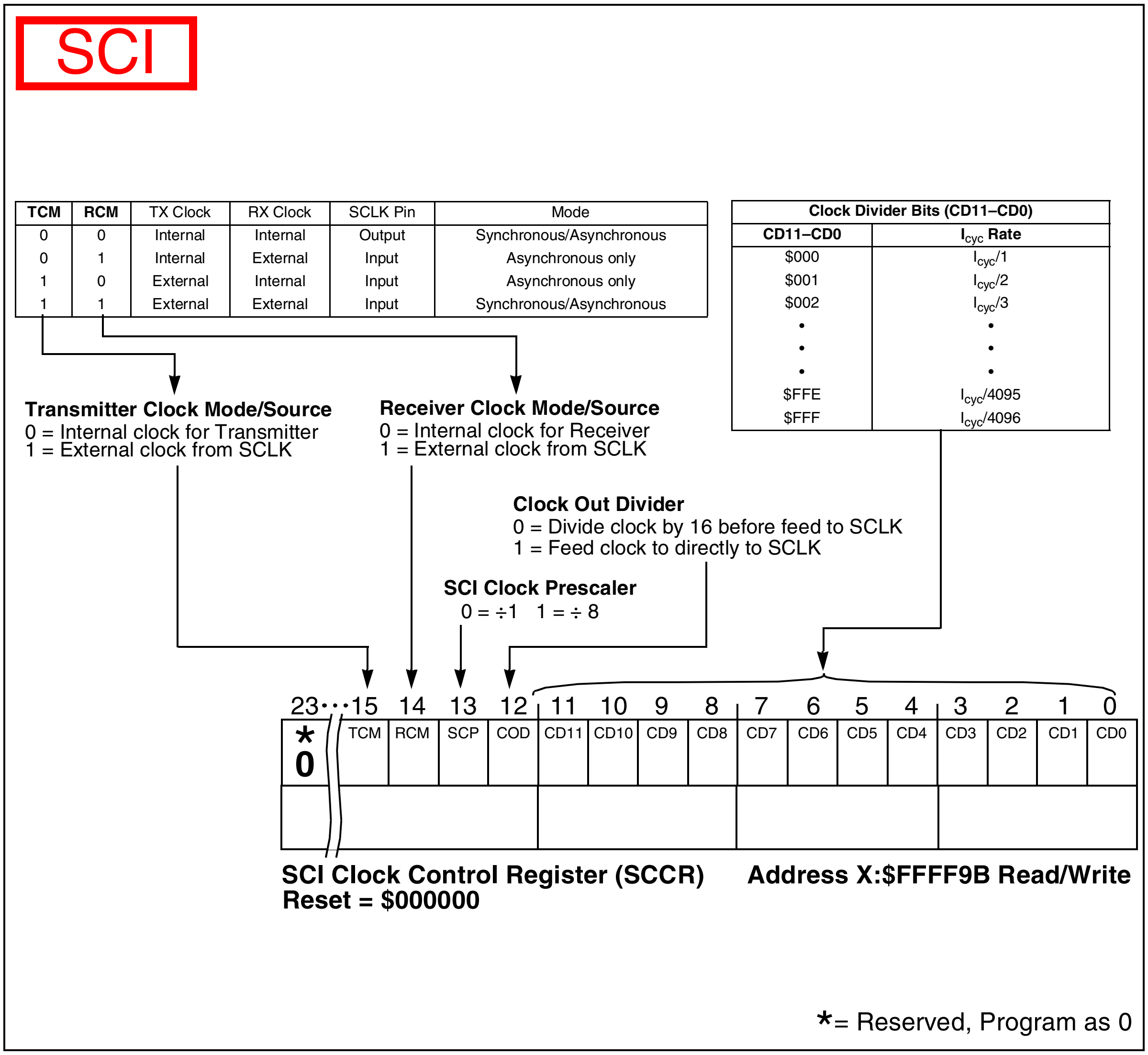
* = Reserved, Program as 0

Figure B-13. SCI Control Register (SCR)

Application:____ Date:______________

____ Programmer:______________

Sheet 2 of 2



SCI

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	$I_{cyc}/4096$

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

SCI Clock Control Register (SCCR) **Address X:$FFFF9B Read/Write**
Reset = $000000

*= Reserved, Program as 0

Figure B-14. SCI Clock Control Registers (SCCR)

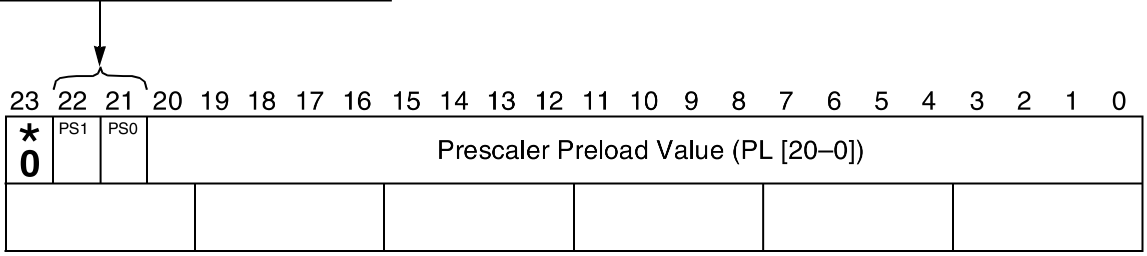
Application:____ Date:________________

____ Programmer:________________

Sheet 1 of 3

Timers

PS (1–0)	Prescaler Clock Source
00	Internal CLK/2
01	TIO0
10	TIO1
11	TIO2



Timer Prescaler Load Register (TPLR) **X:$FFFF83 Read/Write**
Reset = $000000

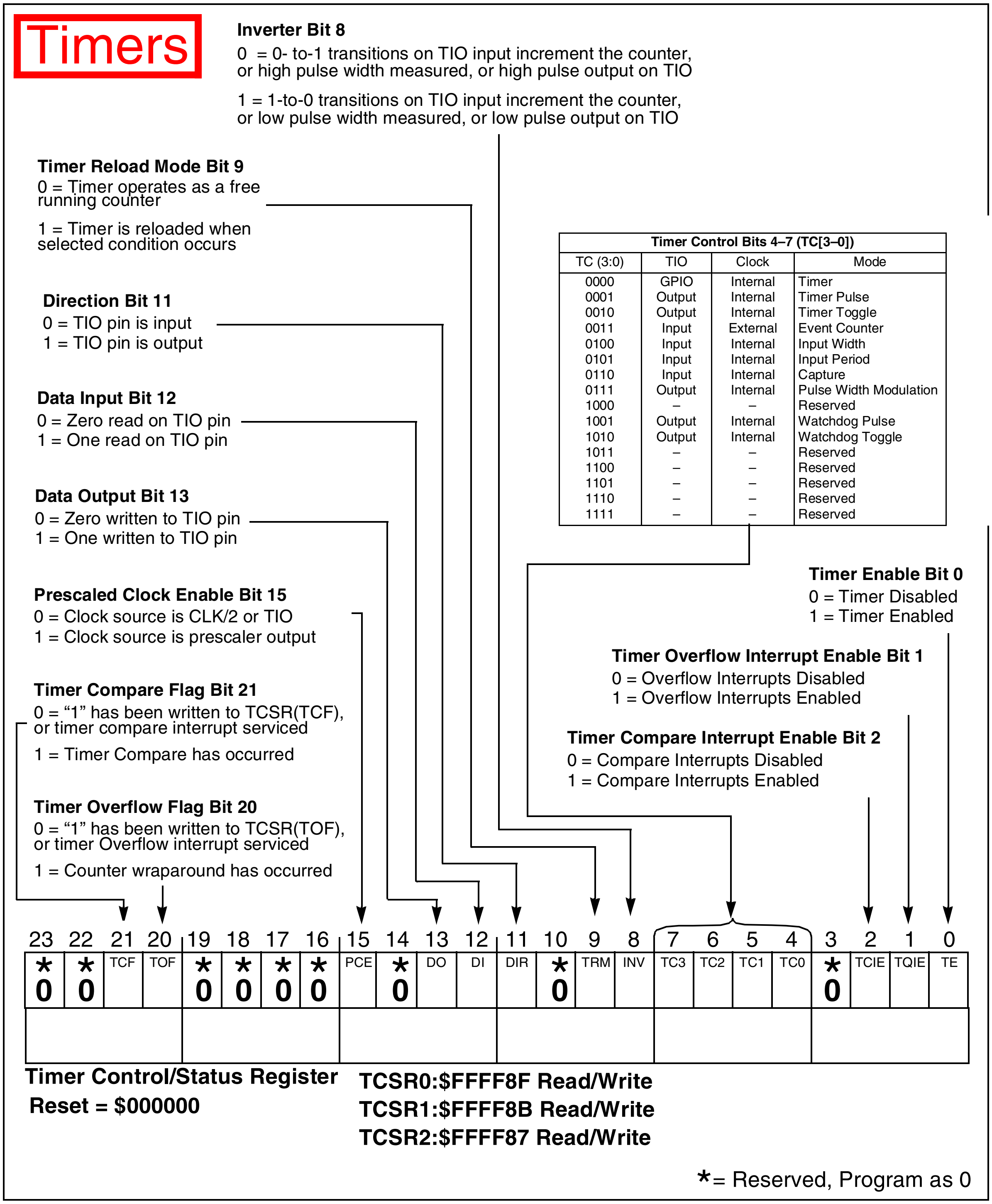
*= Reserved, Program as 0

Figure B-15. Timer Prescaler Load Register (TPLR)

Application:____ Date:________________

____ Programmer:________________

Sheet 2 of 3



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 (TC[3–0])

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
Reset = $000000

TCSR0:$FFFF8F Read/Write
TCSR1:$FFFF8B Read/Write
TCSR2:$FFFF87 Read/Write

* = Reserved, Program as 0

Figure B-16. Timer Control/Status Register (TCSR)

Application:____ Date:________________

____ Programmer:________________

Sheet 3 of 3

Timers

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timer Reload Value																							

Timer Load Register (TLR[0–2])
Reset = $xxxxxx, value is indeterminate after reset

TLR0—X:$FFFF8E Write Only
TLR1—X:$FFFF8A Write Only
TLR2—X:$FFFF86 Write Only

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 (TCPR[0–2])
Reset = $xxxxxx, value is indeterminate after reset

TCPR0—X:$FFFF8D Read/Write
TCPR1—X:$FFFF89 Read/Write
TCPR2—X:$FFFF85 Read/Write

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 (TCR[0–2])
Reset = $000000

TCR0—X:$FFFF8C Read Only
TCR1—X:$FFFF88 Read Only
TCR2—X:$FFFF84 Read Only

Figure B-22. Timer Load, Compare, and Count Registers (TLR, TCPR, TCR)

Application:____ Date:________________

____ Programmer:____________

Sheet 1 of 4

GPIO

Port B (HI08)

DRx = 1 → HIx is Output
DRx = 0 → HIx is Input

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

Host Data Direction Register (HDDR) X:$FFFFC8 Write
Reset = $00

DRx holds value of corresponding HI08 GPIO pin.
Function depends on HDDR.

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

Host Data Register (HDR) X:$FFFFC9 Write
Reset = Undefined

Figure B-17. Host Data Direction and Host Data Registers (HDDR, HDR)

Application:____ Date:_______________

____ Programmer:_______________

Sheet 2 of 4

GPIO

Port C (ESSI0)

PCn = 1 → Port Pin configured as ESSI
PCn = 0 → Port Pin configured as GPIO

23···	6	5	4	3	2	1	0
* 0	* 0	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0

Port C Control Register (PCRC) **X:$FFFFBF Read/Write**
Reset = $000000

PDCn = 1 → Port Pin is Output
PDCn = 0 → Port Pin is Input

23···	6	5	4	3	2	1	0
* 0	* 0	PRC5	PRC4	PRC3	PRC2	PRC1	PRC0

Port C Direction Register (PRRC) **X:$FFFFBE Read/Write**
Reset = $000000

if 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

23···	6	5	4	3	2	1	0
* 0	* 0	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0

Port C GPIO Data Register (PDRC) **X:$FFFFBD Read/Write**
Reset = $000000

*= Reserved, Program as 0

Figure B-18. Port C Registers (PCRC, PRRC, PDRC)

Application:____________________________________ Date:_______________

____________________________________ Programmer:__________

Sheet 3 of 4

GPIO

Port D (ESSI1)

PCn = 1 → Port Pin configured as ESSI
PCn = 0 → Port Pin configured as GPIO

23···6		5	4	3	2	1	0
*0	*0	PCD5	PCD4	PCD3	PCD2	PCD1	PCD0

Port D Control Register (PCRD) X:$FFFFAF Read/Write
Reset = $000000

PDCn = 1 → Port Pin is Output
PDCn = 0 → Port Pin is Input

23···6		5	4	3	2	1	0
*0	*0	PRD5	PRD4	PRD3	PRD2	PRD1	PRD0

Port D Direction Register (PRRD) X:$FFFFAE Read/Write
Reset = $000000

if 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

23···6		5	4	3	2	1	0
*0	*0	PDD5	PDD4	PDD3	PDD2	PDD1	PDD0

Port D GPIO Data Register (PDRD) X:$FFFFAD Read/Write
Reset = $000000

*= Reserved, Program as 0

Figure B-19. Port D Registers (PCRD, PRRD, PDRD)

Application:____ Date:________________

____ Programmer:______________

Sheet 4 of 4

GPIO

Port E (SCI)

PCn = 1 → Port Pin configured as ESSI
PCn = 0 → Port Pin configured as GPIO

23···	···6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	PCE2	PCE1	PCE0

Port E Control Register (PCRE) **X:$FFFF9F Read/Write**
Reset = $000000

PDCn = 1 → Port Pin is Output
PDCn = 0 → Port Pin is Input

23···	···6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	PRE2	PRE1	PRE0

Port E Direction Register (PRRE) **X:$FFFF9E Read/Write**
Reset = $000000

if 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

23···	···6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	PDE2	PDE1	PDE0

Port E GPIO Data Register (PDRE) **X:$FFFF9D Read/Write**
Reset = $000000

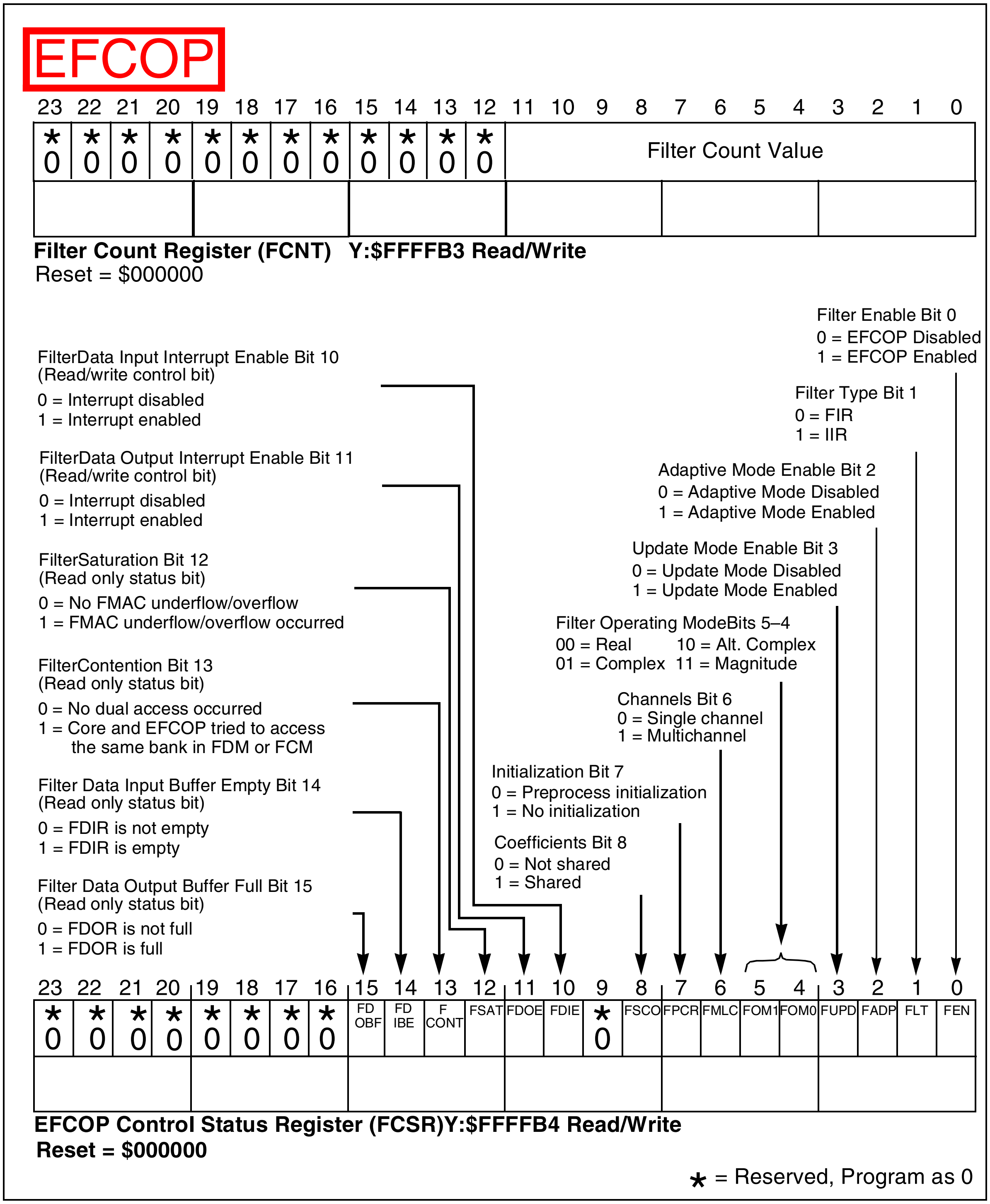
*= Reserved, Program as 0

Figure B-20. Port E Registers (PCRE, PRRE, PDRE)

Application:____ Date:____________________

____ Programmer:______________

Sheet 1 of 2



EFCOP

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	Filter Count Value											

Filter Count Register (FCNT) Y:$FFFFB3 Read/Write
Reset = $000000

FilterData Input Interrupt Enable Bit 10
(Read/write control bit)
0 = Interrupt disabled
1 = Interrupt enabled

FilterData Output Interrupt Enable Bit 11
(Read/write control bit)
0 = Interrupt disabled
1 = Interrupt enabled

FilterSaturation Bit 12
(Read only status bit)
0 = No FMAC underflow/overflow
1 = FMAC underflow/overflow occurred

FilterContention Bit 13
(Read only status bit)
0 = No dual access occurred
1 = Core and EFCOP tried to access
the same bank in FDM or FCM

Filter Data Input Buffer Empty Bit 14
(Read only status bit)
0 = FDIR is not empty
1 = FDIR is empty

Filter Data Output Buffer Full Bit 15
(Read only status bit)
0 = FDOR is not full
1 = FDOR is full

Filter Enable Bit 0
0 = EFCOP Disabled
1 = EFCOP Enabled

Filter Type Bit 1
0 = FIR
1 = IIR

Adaptive Mode Enable Bit 2
0 = Adaptive Mode Disabled
1 = Adaptive Mode Enabled

Update Mode Enable Bit 3
0 = Update Mode Disabled
1 = Update Mode Enabled

Filter Operating ModeBits 5–4
00 = Real 10 = Alt. Complex
01 = Complex 11 = Magnitude

Channels Bit 6
0 = Single channel
1 = Multichannel

Initialization Bit 7
0 = Preprocess initialization
1 = No initialization

Coefficients Bit 8
0 = Not shared
1 = Shared

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	FD OBF	FD IBE	F CONT	FSAT	FDOE	FDIE	* 0	FSCO	FPCR	FMLC	FOM1	FOM0	FUPD	FADP	FLT	FEN

EFCOP Control Status Register (FCSR)Y:$FFFFB4 Read/Write
Reset = $000000

* = Reserved, Program as 0

Figure B-21. EFCOP Counter and Control Status Registers (FCNT and FCSR)

Application:____ Date:________________

____ Programmer:________________

Sheet 2 of 2

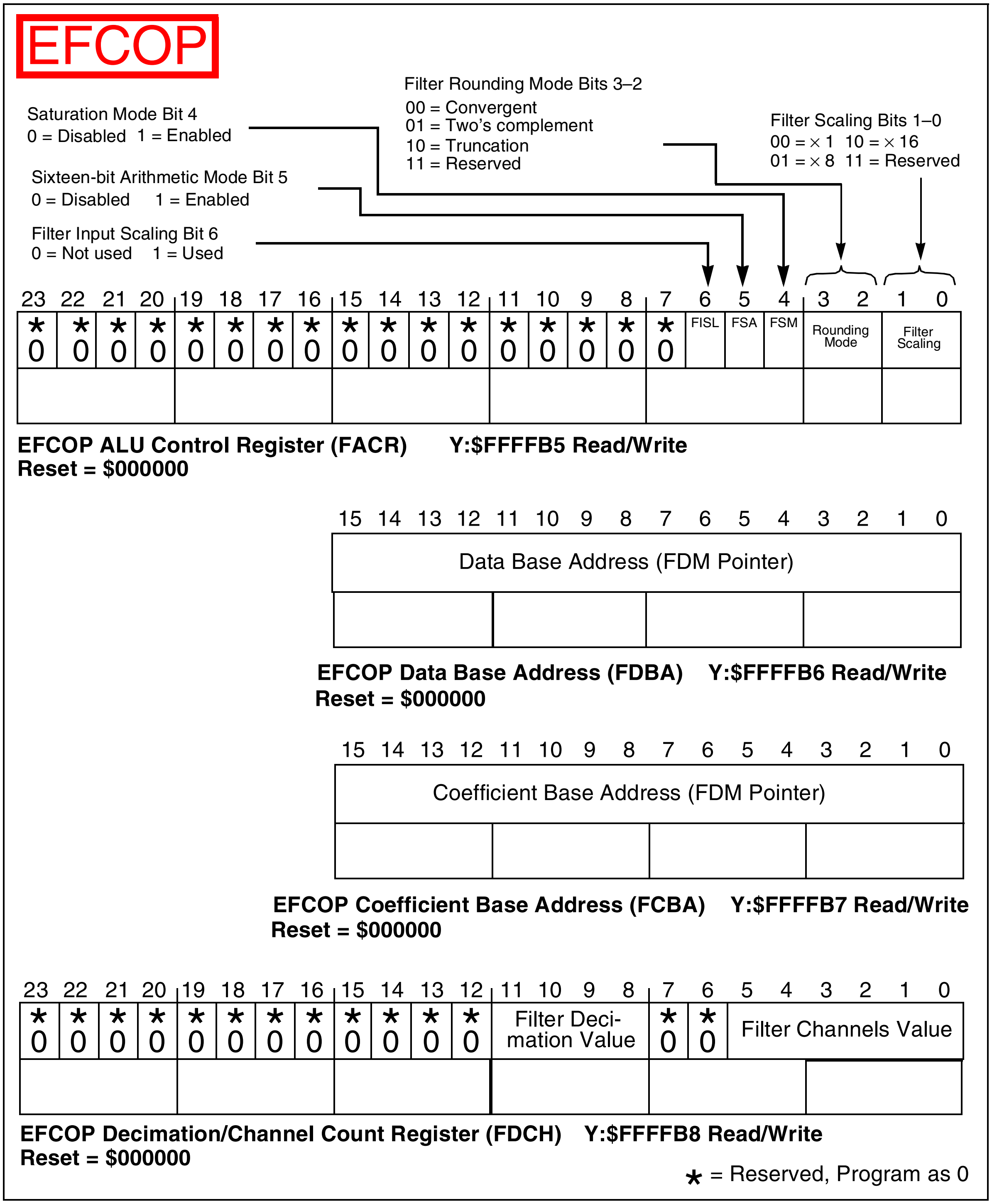


Figure B-22. EFCOP FACR, FDBA, FCBA, and FDCH Registers

Index