DSP56003/005

24-BIT
DIGITAL SIGNAL PROCESSOR
USER’S MANUAL

Motorola, Inc.
Semiconductor Products Sector
DSP Division
6501 William Cannon Drive, West
Austin, Texas 78735-8598

MOTOROLA

Order this document by DSP56003UM/AD

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

OnCE™ is a trade mark of Motorola, Inc.

TABLE OF CONTENTS

Paragraph Number | Title | Page Number

SECTION 1
INTRODUCTION TO THE DSP56003/005

Table of Contents (Continued)

Paragraph Number — Title — Page Number

SECTION 2
PIN DESCRIPTIONS

Table of Contents (Continued)

Paragraph Number | Title | Page Number

Table of Contents (Continued)

Paragraph Number | Title | Page Number

SECTION 3
MEMORY, OPERATING MODES, AND INTERRUPTS

SECTION 4
EXTERNAL MEMORY INTERFACE

SECTION 5
HOST INTERFACE

Table of Contents (Continued)

Table of Contents (Continued)

SECTION 6
SERIAL COMMUNICATIONS INTERFACE

Table of Contents (Continued)

Table of Contents (Continued)

Paragraph Number | Title | Page Number

SECTION 7
SYNCHRONOUS SERIAL INTERFACE

Table of Contents (Continued)

SECTION 8
TIMER/EVENT COUNTER

Table of Contents (Continued)

Paragraph Number | Title | Page Number

SECTION 9
PULSE WIDTH MODULATORS

Table of Contents (Continued)

SECTION 10
WATCHDOG TIMER

APPENDIX A
BOOTSTRAP PROGRAM AND
DATA ROM LISTINGS

Table of Contents (Continued)

Paragraph Number — Title — Page Number

APPENDIX B
PROGRAMMING SHEETS

APPENDIX C
DSP56003 AND DSP56005 DIFFERENCES

Table of Contents (Continued)

Paragraph Number	Title	Page Number

LIST of FIGURES

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

Figure Number | Title | Page Number

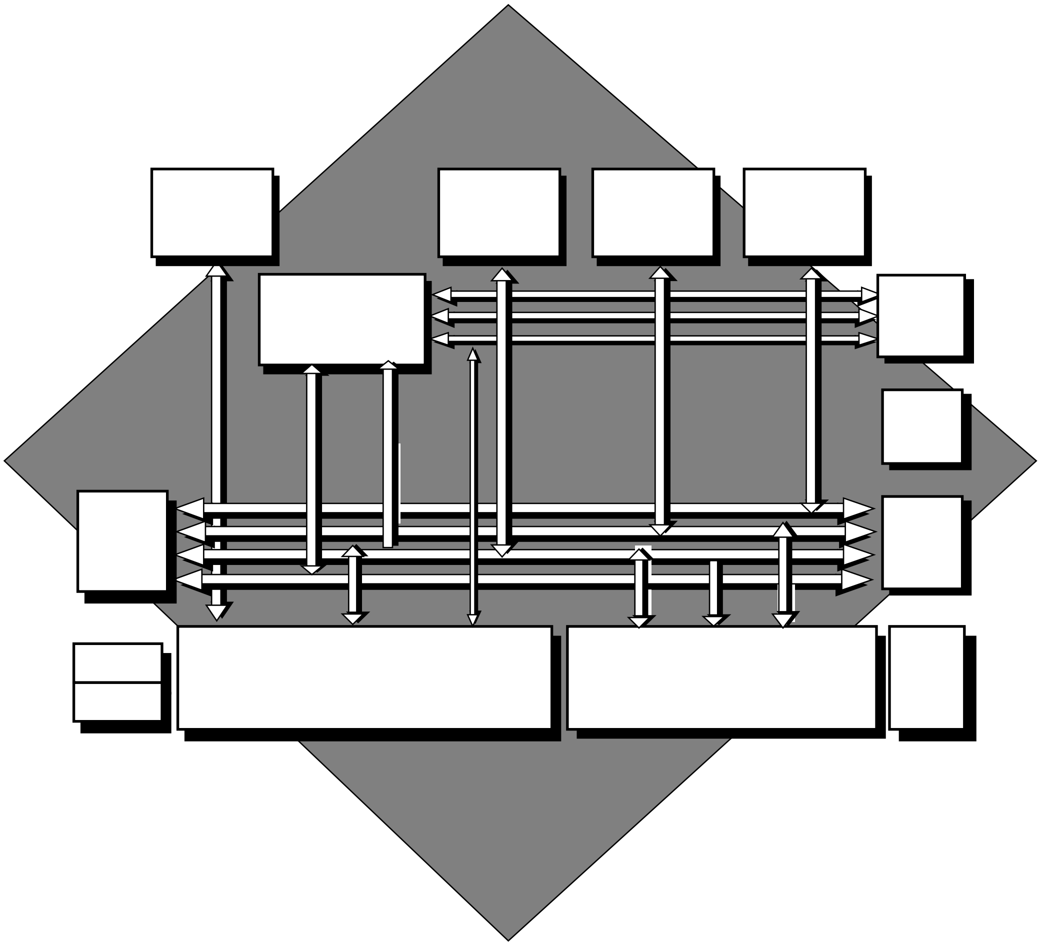
LIST of TABLES

Table Number **Title** **Page Number**

List of Tables (Continued)

SECTION 1

INTRODUCTION TO THE DSP56003/005



SECTION CONTENTS

1.1 MANUAL INTRODUCTION

This manual describes the DSP56003 and the DSP56005 24-bit digital signal processors, their memory, operating modes, and peripheral modules. All of the documentation listed in Table 1-1 is required for a complete description of the DSP56003/005, and is necessary to properly design with the part.

Table 1-1 Documentation Required for a Complete Description

Document Name	Description	Order Number
DSP56000 Family Manual	Detailed description of the 56000-family architecture and the 24-bit core processor and instruction set	DSP56KFAMUM/AD
DSP56003/005 User's Manual	Detailed description of memory, peripherals, and interfaces	DSP56003UM/AD
DSP56003/005 Technical Data Sheet	Pin and package descriptions, and electrical and timing specifications	DSP56005/D

1.1.1 Related Literature

Additional supporting literature discussing theory, algorithms, systems, and applications of DSP or the DSP56003/005 is listed in Table 1-2. Documentation is available from a local Motorola distributor or semiconductor sales office, or through these Motorola Literature Distribution Centers:

1. USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
2. EUROPE: Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, Great Britain.
3. JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
4. ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbor Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

Table 1-2 Related Motorola Documentation

Document Name	Description	Order Number
Motorola's 16-, 24-, and 32-bit Digital Signal Processing Families	Overview of all of the DSP product families	BR1105/D
DSP56005 Product Brief	Product overview, block diagram, features	DSP56005P/D
DSP56000/001 ADS Brochure	Overview of the chip's Application Development System hardware	BR517/D
DSP56000/001 C Cross Compiler	Product summary	BR541/D
DSP56000CLASx Design-In Software Package	Product summary. Simulator, Libraries, Assembler, Linker	BR526/D
Digital Sine-Wave Synthesis	Application Report. Uses the DSP56001 look-up table	APR1/D
Digital Stereo 10-band Graphic Equalizer	Application Report. Includes code and circuitry; features the DSP56001	APR2/D
Fractional And Integer Arithmetic	Application Report. Includes code	APR3/D
Implementation of Fast Fourier Transforms	Application Report. Comprehensive FFT algorithms and code for DSP56001, DSP56156, and DSP96002	APR4/D
Implementation of PID Controllers	Application Report. PWM using the SCI timer and three phase output using modulo addressing	APR5/D
Convolutional Encoding and Viterbi Decoding with a V.32 Modem Trellis Example	Application Report. Theory and code; features the DSP56001	APR6/D
Implementing IIR/FIR Filters	Application Report. Comprehensive example using the DSP56001	APR7/D
Full-Duplex 32-kbit/s CCITT ADPCM Speech Coding	Application Report. Features the DSP56001	APR9/D

Table 1-2 Related Motorola Documentation (Continued)

Document Name	Description	Order Number
DSP56001 Interface Techniques and Examples	Application Report. Interfaces for pseudo static RAM, dynamic RAM, ISA bus, Host	APR11/D
Twin CODEC Expansion Board for the DSP56000 ADS	Application Report. Circuit, code, FIR filter design for two voice band CODECs connecting to the SSI	APR12/D
Conference Bridging in the Digital Telecommunications Environment	Application Report. Theory and code; features the DSP56001/002	APR14/D
Implementation of Adaptive Controllers	Application Report. Adaptive control using reference models; generalized predictive control; includes code	APR15/D
Low Cost Controller for DSP56001	Application Report. Circuit and code to connect two DSP56001s to an MC68008	APR402/D
G.722 Audio Processing	Application Report. Theory and code using SB-ADPCM	APR404/D
Minimal Logic DRAM Interface	Application Report. 1M x 4 80 nS DRAM, 1 PAL, code	APR405/D
Logarithmic/Linear Conversion Routines	Application Report. m-law and A-law companding routines for PCM mono-circuits	ANE408/D
Dr. BuB Bulletin Board	Flyer. Motorola's electronic bulletin board where free DSP software is available	BR297/D
Third Party Compendium	Brochures from companies selling hardware and software that supports Motorola DSPs	DSP3RDPTYPAK/D
University Support Program	Flyer. Motorola's program supporting Universities in DSP research and education	BR382/D
Real Time Signal Processing Applications with Motorola's DSP56000 Family	Textbook by Mohamed El-Sharkawy; 398+ pages	Prentice-Hall, 1990; ISBN 0-13-767138-5

1.1.2 Training

Both self-paced audio courses and instructor led class room courses are available for the Motorola digital signal processors. A technical training catalog (order number BR348/D) is available which describes these courses and gives the current training schedule and prices. Information about these courses and registration is available by calling (602) 897-3665. To register for Toronto and Ottawa classes, call (416) 497-8181.

1.1.3 Technical Assistance

Information and assistance for DSP applications is available through your local Motorola field office. See your local telephone directory for telephone numbers.

1.1.4 Manual Conventions

The following conventions are used in this manual:

- This manual describes both the DSP56003 and the DSP56005. The chip is identical in both of these DSPs. However, the DSP56003 is in a larger package with more pins which provide more functions than the DSP56005. Vertical bars in the margin of this manual have been used to flag portions of the text that describe these signals that apply only to the DSP56003. Additionally, Appendix C details in one place all of the differences between the two parts.
- overbars are used to indicate a signal that is active when pulled to ground (see Table 1-3) e.g. the reset pin, $\overline{\text{RESET}}$, is active when pulled to ground. Therefore, references to the $\overline{\text{RESET}}$ pin will always have an overbar.
- The word “assert” (see Table 1-3) 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” (see Table 1-3) means that a high true signal is pulled low to ground or that a low true signal is pulled high to V_{CC}.
- The word “reset” is used in three different contexts in this manual. There is a reset pin which is always written as “$\overline{\text{RESET}}$”, there is a reset instruction which is always written as “RESET”, and the word reset, used to refer to the reset function, is written in lower case with a leading capital letter as grammar dictates.

Table 1-3 High True / Low True Signal Conventions

Signal/Symbol	Logic State	Signal State	Voltage
$\overline{\text{PIN}}$	True	Asserted	Ground
$\overline{\text{PIN}}$	False	Deasserted	V_{CC}
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).

1.1.5 Manual Organization

This manual includes the following sections:

SECTION 1 — INTRODUCTION

- introduces the manual and gives references to related literature
- provides a brief description of the blocks in the chip block diagram. Detailed information on these blocks can be found in either the *DSP56000 Family Manual* or in this manual.

SECTION 2 — PIN DESCRIPTIONS

- presents the DSP56003/005 pin descriptions
  Note that the DSP56003 is a version of the DSP56005 with extra functions. The additional signals on the DSP56003 are bus arbitration signals, a PLL lock signal, and a PLL clock output polarity control signal.

SECTION 3 — MEMORY, OPERATING MODES, AND INTERRUPTS

- presents the details of the DSP56003/005 memory maps
- describes the interrupt vector locations
- describes operation of the interrupt priority register
- explains the various operating modes that affect the processor's program and data memories

SECTION 4 — EXTERNAL MEMORY INTERFACE

- describes the external memory port, its registers, and its controls

SECTION 5 — HOST INTERFACE

- describes operation, registers, and control of the parallel Host Interface (HI) and Port B General Purpose I/O

SECTION 6 — SERIAL COMMUNICATIONS INTERFACE

- describes the Port C parallel I/O, the Synchronous Communication Interface, its registers, and its controls

SECTION 7 — SYNCHRONOUS SERIAL INTERFACE

- describes the Port C parallel I/O, the Synchronous Serial Interface, its registers, and its controls

SECTION 8 — TIMER AND EVENT COUNTER

- describes the timer/event counter, its registers, and controls

SECTION 9 — PULSE WIDTH MODULATORS

- describes the five pulse width modulators available on the DSP56003/005, its registers, and controls

SECTION 10 — WATCHDOG TIMER

- describes a timer that can interrupt the DSP56003/005 after a specified number of clocks, its registers, and controls

APPENDIX A — BOOTSTRAP CODE AND DATA ROM LISTINGS

- provides the code used to bootstrap the DSP56003/005 and the listings for the sine table and arctangent table available in on-chip ROM

APPENDIX B — PROGRAMMING SHEETS

- provide a fast reference section for the instructions and registers used by the DSP56003/005. These sheets are intended to be copied and used while programming the registers.

APPENDIX C — DIFFERENCES BETWEEN THE DSP56003 AND DSP56005

- provides a description of the specific differences between the two parts

USER'S COMMENTS

- allows the reader to notify Motorola of any errors or discrepancies discovered in this manual

1.2 PRODUCT USE

The DSP56003/005 is a general purpose digital signal processor designed for control and embedded processor applications such as a disk drive controller. It is based on the DSP56002 in that is has the same core processor and peripherals (Host Interface, SCI, SSI, and Timer/Event Counter) but has two new peripherals and extra memory.

A Timer/Event Counter, Pulse Width Modulators, and a Watchdog Timer provide the tools needed to design sophisticated yet cost effective control applications using the DSP56003/005. The Timer/Event Counter provides a versatile tool for both monitoring signals and generating them. The five pulse width modulators provide a convenient means to generate signals and control motors. Critical applications require a foolproof method of insuring proper DSP operation. The Watchdog Timer is a tool to detect some software and hardware failures and provide a failure recovery path by resetting the system or running a corrective program. The general purpose I/O pins provide up to 25 additional input or output signals that are individually controllable.

While the DSP56003/005 has the power and ease of programming required for stand alone, embedded applications, the three communication ports (Host Interface, SCI, and SSI) allow this DSP to be simply connected to almost any other electronic device for attached processor or distributed processing applications with little or no additional logic.

1.3 DSP56003/005 ARCHITECTURAL OVERVIEW

The DSP56003 and DSP56005 are expanded versions of the DSP56002 and are members of the 24-bit 56000 family. They are composed of the 24-bit 56000 DSP core, memory, and unique set of peripheral modules. The 24-bit 56000 DSP core is composed of a data ALU, an address generation unit, a program controller, an On-Chip Emulator (OnCE Port), and a PLL-based clock oscillator. The DSP56000-family architecture, on which the DSP56003/005 is built, was designed to maximize throughput in data-intensive digital signal processing applications. The result is a dual-natured, expandable architecture with sophisticated on-chip peripherals and general purpose I/O. It is dual-natured in that there are two independent, expandable data memory spaces, two address arithmetic units, and a data ALU which has two accumulators and two shifter/limiters. The duality of the architecture makes it easier to write software for DSP applications. For example, data is naturally partitioned into X and Y spaces for graphics and image processing applications, into coefficient and data spaces for filtering and transformations, and into real and imaginary spaces for performing complex arithmetic.

The following memory and peripheral modules are contained in the DSP56003/005:

- Program RAM Memory Module — Long and varied programs can be run from the 4608 words of 24-bit wide fully static program RAM that resides on the DSP56003/005
- Bootstrap ROM Memory Module — Bootstrap code runs at reset time from a 96-word on-chip ROM to load the DSP's operating program. This ROM does not occupy any of the 64k program memory space and is not accessible by the user.
- Data RAM Memory Modules — There are two on-chip data RAMs: 256 words of X RAM and 256 words of Y RAM. These are general purpose memories that can be used for intermediate values, coefficients, stacks, queues, variables, etc.
- Data ROM Memory Modules — There are two on-chip data ROMs. A 256 word arctangent table is located from X:100 to X:1FF and a 256 word sine table is located from Y:100 to Y:1FF. These tables are useful in calculating trigonometric functions in control operations.
- Host Interface Module (HI) — The 8-bit Host Interface module provides a fast, yet simple parallel interface to connect the DSP56003/005 to a host processor or bus. The Host Interface is identical to those found in the DSP56000, DSP56001, and DSP56002.
- Serial Communications Interface Peripheral Module (SCI) — The SCI peripheral module provides a full-duplex asynchronous serial interface which allows the DSP56003/005 to communicate using standard universal asynchronous receiver and transmitter (UART) protocols at bit rates up to CLK/4, i.e. 12.5 MHz for a 50 MHz clock, which are commonly used in modems, terminals, microcontrollers, computer serial ports, etc. The SCI is identical to those found in the DSP56000, DSP56001, and DSP56002.
- Synchronous Serial Interface Peripheral Module (SSI) — The SSI peripheral module is an extremely flexible, full-duplex synchronous serial interface. The SSI allows the DSP56003/005 to be used with standard codecs, other DSPs, microprocessors, and serial peripherals up to system clock/4; i.e. 12.5 Mb/s for a 50 MHz clock. The SSI is identical to those found in the DSP56000, DSP56001, and DSP56002.
- Timer/Event Counter Peripheral Module — The 24-bit Timer/Event Counter peripheral module can be used to interrupt the DSP at set intervals, output a fixed or modulated pulse or square wave, measure pulse widths (rising to falling or falling to rising edges), and measure signal periods (rising to rising or falling to falling edges). The maximum resolution is $^1/_2$ the DSP clock frequency, i.e. 40 ns for a 50 MHz clock. The Timer/Event Counter is identical to the one found in the DSP56002.

- Pulse Width Modulator Peripheral Module (PWM) — The PWM module contains three 16-bit signed data pulse width modulators and two 16-bit positive fractional data pulse width modulators. These are very flexible devices useful in many applications such as disk drive motor control and head positioning, heater controls, lighting controls, etc. The maximum resolution is $^1/_2$ the DSP clock frequency, i.e. 40 ns for a 50 MHz clock.

- Watchdog Timer Peripheral Module — The 16-bit Watchdog Timer peripheral module generates a non-maskable interrupt if it is allowed to time out. This can be used to reset the DSP when either software or hardware stops responding normally. The maximum resolution is $^1/_4$ the DSP clock frequency, i.e. 80 ns for a 50 MHz clock.

1.3.1 DSP56003/005 Features

24-bit 56000 Family Central Processing Unit (CPU) Features

- 25 Million Instructions per Second (MIPS) at 50 MHz
- On-chip Harvard Architecture Making Parallel Accesses to Program and Two Data Memories
- Single-Cycle 24 x 24 Bit Parallel Multiply-Accumulator
- Highly Parallel Instruction Set with Unique DSP Addressing Modes
- Zero Overhead Nested DO Loops
- Fast Auto-Return Interrupts
- Operation with 24-bit Data/16-bit Address Parallel Interface to Off-Chip Memory
- STOP and WAIT Low-power Standby Modes
- Fully Static Logic, Operation Frequency Down to DC
- Low-power CMOS Design

DSP56003/005 Features

- 4608 x 24-bit Program RAM
- Two 256 x 24-bit Data RAM
- Two 256 x 24-bit Data ROM (Arctangent and Sine Tables)
- Full Speed Memory Expansion Port with 16-bit Address and 24-bit Data Buses
- Byte-wide Host Interface with DMA Support
- Synchronous Serial Interface Port
- Asynchronous Serial Communication Interface Port
- Up to 25 General Purpose I/O Pins
- 24-bit Timer/Event Counter
- Five Pulse Width Modulators
  - Three Use Two’s Complement, Fractional Data
  - Two Use Positive Fractional Data
- Watchdog Timer
- On-chip Emulator Port (OnCE™ Port) for Unobtrusive, Full Speed Debugging

- PLL Based Clocking with Wide Input Frequency Range, Wide Range Frequency Multiplication (1 to 4096) and Power Saving Clock Divider (2^i, i=0,...,15) to Reduce Clock Noise

DSP56003 Features

- The DSP56003 has the same features as the DSP56005 with the following additions:
  - External Memory Bus Arbitration Signals
  - PLL Lock Signal
  - PLL Clock Output Polarity Signal

1.3.2 Block Diagram Description

The major components of the DSP56003/005 are (see Figure 1-1):

DSP56000 Family DSP Engine

- Data ALU
- Address Generation Unit
- Program Control Unit
- Data Buses
- Address Buses

Memory Modules

- Program Memory including bootstrap code
- X Data Memory
- Y Data Memory

Peripheral Modules

- External Memory Expansion Port
- Host Interface
- Serial Communications Interface
- Synchronous Serial Interface
- Timer/Event Counter
- Pulse Width Modulators
- Watchdog Timer
- General Purpose I/O (most unused peripheral pins can be assigned for general purpose Input/Output control)

These components are depicted in Figure 1-1 and described in the following paragraphs. The blocks shown in the Expansion Area in Figure 1-1 are described in detail in this manual. The blocks shown in the 24-bit 56000 core area are described in detail in the DSP56000 Family Manual.

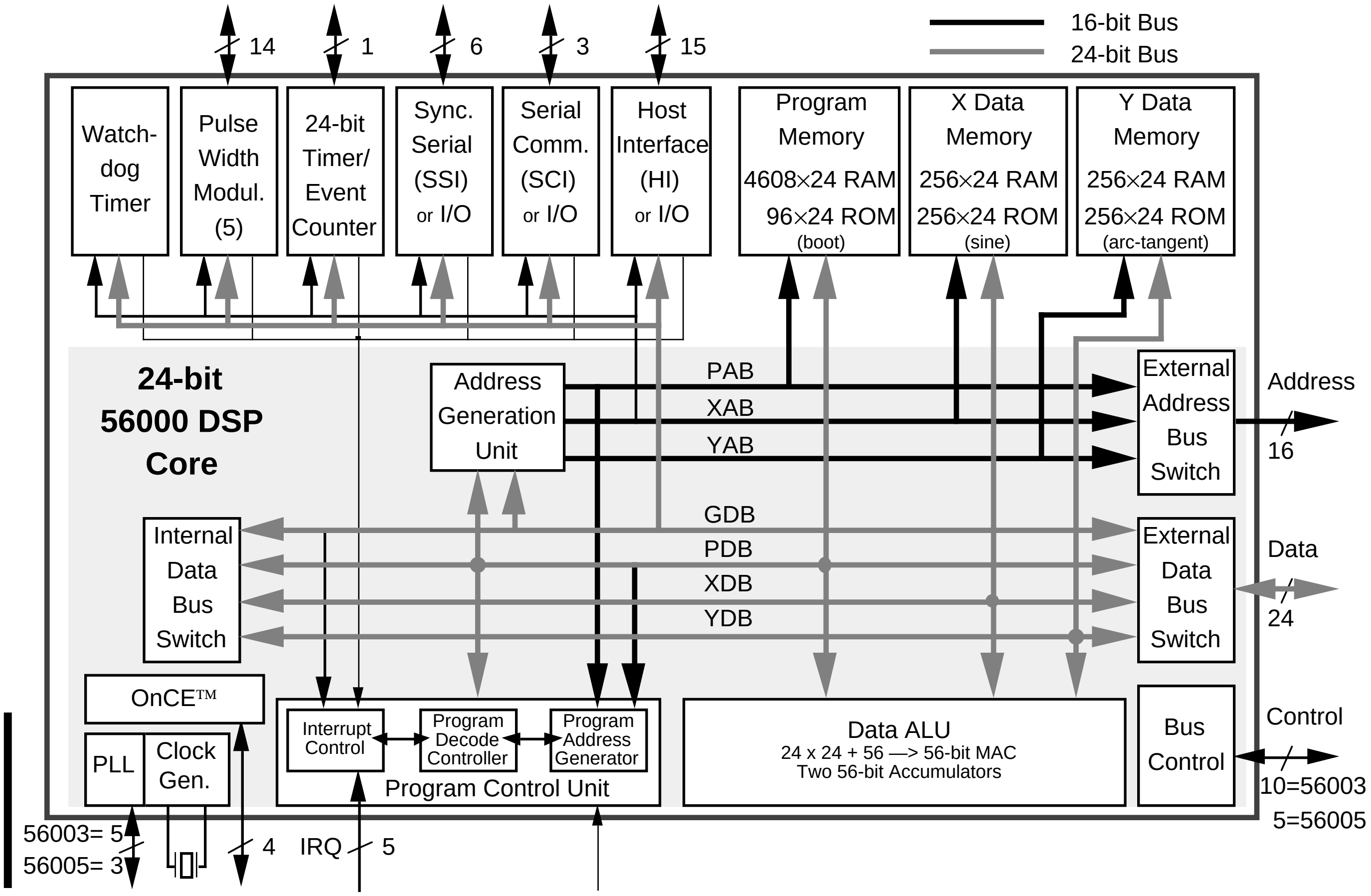


Figure 1-1 DSP56003/005 Block Diagram

1.3.2.1 Data Buses

Data movement on the chip occurs over four bidirectional 24-bit buses — the X data bus (XDB), the Y data bus (YDB), the program data bus (PDB), and the global data bus (GDB). Certain instructions concatenate XDB and YDB to form a 48-bit data bus. Data transfers between the data ALU and the two data memories, X data memory and Y data memory, occur over the XDB and YDB, respectively. These transfers can occur simultaneously on the chip, maximizing performance. All other data transfers such as I/O transfers to internal peripherals occur over the GDB. Instruction word pre-fetches take place over the PDB in parallel to data transfers. Transfers between buses are accomplished through the internal bus switch.

External memory transfers occur through the external memory port (Port A). A single transfer can occur through Port A in a single instruction cycle and can be a program memory, X memory, or Y memory transfer. The appropriate address and data bus is directed to Port A by the external address bus switch and external data bus switch.

1.3.2.2 Address Buses

Addresses are specified for internal X data memory and Y data memory on two unidirectional 16-bit buses — the X address bus (XAB) and the Y address bus (YAB). Program memory addresses are specified on the 16-bit program address bus (PAB). External memory spaces are addressed via a single 16-bit unidirectional external address bus driven by a three input multiplexer that can select addressing from either the XAB, YAB, or PAB. There is no processing delay† if only one external memory space is accessed in an instruction.

If two or three external memory spaces are accessed in a single instruction, there will be a one or two instruction cycle execution delay†, respectively. A bus arbitrator controls external accesses.

1.3.2.3 Data ALU

The data ALU has been designed to be fast and yet provide the capability to process signals having a wide dynamic range. Special circuitry has been provided to facilitate handling data overflows and round-off errors.

The data ALU performs all of the arithmetic and logical operations on data operands. The data ALU consists of four 24-bit input registers, two 48-bit accumulator registers, two 8-bit accumulator extension registers, an accumulator shifter, two data shifter/limiters, and a parallel single-cycle non-pipelined multiply-accumulator (MAC). Data ALU operations use fractional two's complement arithmetic. Data ALU registers may be read or written over the XDB and YDB as 24- or 48-bit operands. The data ALU is capable of performing any of the following operations in a single instruction cycle: multiplication, multiply-accumulate with positive or negative accumulation, convergent rounding, multiply-accumulation with positive or negative accumulation and convergent rounding, addition, subtraction, a divide iteration, a normalization iteration, shifting, and logical operations. Data ALU source operands may be 24, 48, or in some cases 56 bits, and originate from data ALU registers. The data ALU destination is always one of the two 56-bit accumulators.

The 24-bit data words provide 144 dB of dynamic range. This is sufficient for most real world applications including higher level audio applications since the majority of analog to digital (A/D) and digital to analog (D/A) converters are 16 bits or less, and certainly not greater than 24 bits. The 56-bit accumulation internal to the data ALU provides 336 dB of internal dynamic range assuring no loss of precision due to intermediate processing.

† when using fast external memories

Two data shifter/limiters provide special post-processing on data read from the ALU accumulator registers A and B and directed to the XDB or YDB. The data shifters are capable of shifting data one bit to the left or to the right as well as passing the data unshifted. Each data shifter has a 24-bit output with overflow indication. The data shifters are controlled by scaling mode bits. These shifters permit dynamic scaling of fixed point data without modifying the program code by simply programming the scaling mode bits. This permits block floating-point algorithms to be implemented in a regular fashion. For example, FFT routines can use this feature to selectively scale each butterfly pass.

Saturation arithmetic is accommodated to minimize errors due to overflow. Overflow occurs when a source operand requires more bits for accurate representation that there are available in the destination. To minimize the error due to overflow, the maximum (or minimum if negative) value, i.e. “limited”, is written to the destination with an error flag.

1.3.2.4 Address Generation Unit

The address generation unit performs all address storage and effective address calculations necessary to address data operands in memory. It implements three types of arithmetic to update addresses — linear, modulo, and reverse carry. This unit operates in parallel with other chip resources to minimize address generation overhead. The address generation unit contains eight address registers (R0-R7, i.e. Rn), eight offset registers (N0-N7, i.e. Nn), and eight modifier registers (M0-M7, i.e. Mn). The Rn are 16-bit registers which may contain an address or data. Each Rn register may provide addresses to the XAB, YAB, and PAB. The Nn and Mn registers are 16-bit registers which are normally used to control updating the Rn registers but can be used for data.

Address generation unit registers may be read or written via the global data bus as 16-bit operands. The address generation unit has two modulo arithmetic units which can generate two independent 16-bit addresses every instruction cycle for any two of the XAB, YAB, or PAB. The address generation unit can directly address 65,536 (64k) locations on the XAB, 65,536 locations on the YAB, and 65,536 locations on the PAB — a total capability of 196,608 24-bit words.

1.3.2.5 Memories

The three independent memory spaces of the DSP56003/005 — X data, Y data, and program — are shown in Figure 1-1, Figure 1-2a, and Figure 1-2b. These memory spaces are configured by control bits in the operating mode register.

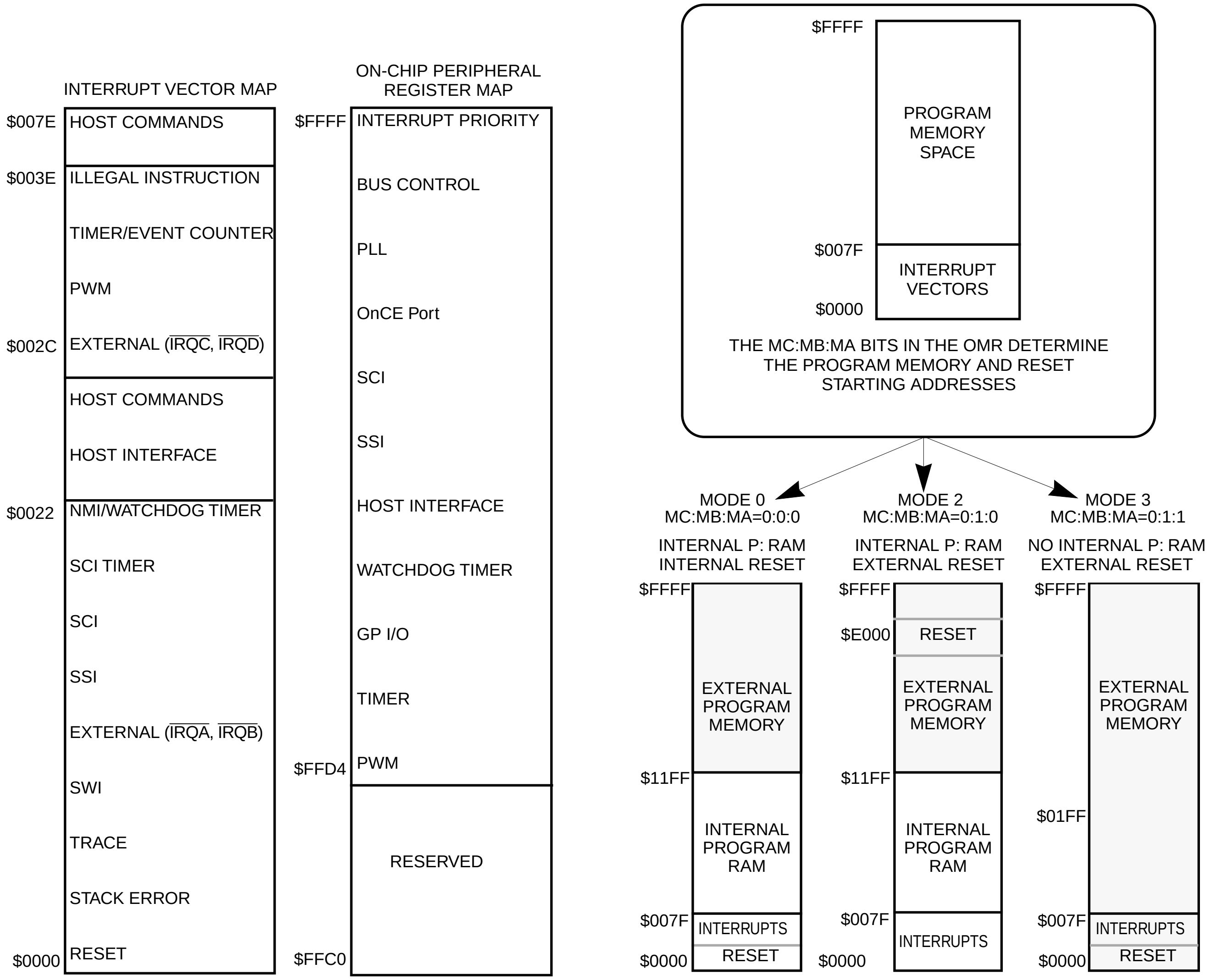
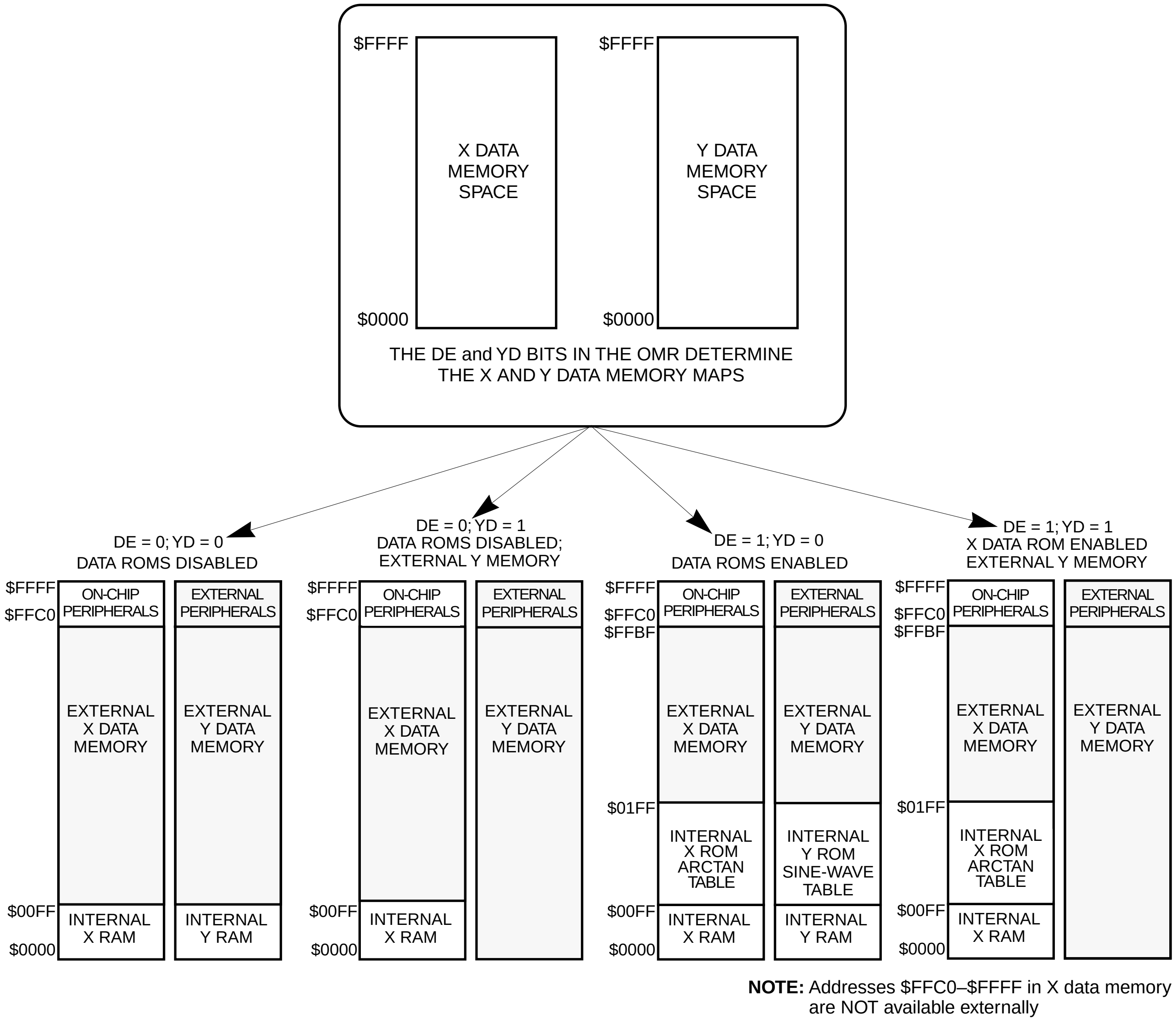


Figure 1-2a DSP56003/005 Memory Maps

1.3.2.5.1 Program Memory

On-chip program RAM memory, consists of a 4608 location by 24-bit high speed RAM which is enabled by bits in the OMR. Addresses are received from the program control logic (usually the program counter) over the PAB. Program memory may be written using MOVEM instructions. The interrupt vectors for the on-chip resources are located in the bottom 128 locations of program memory. Program memory may be expanded to 64k off-chip.

Program RAM has many advantages. It provides a means to develop code efficiently. The programs can be changed dynamically, allowing efficient overlaying of DSP software algorithms. In this way the on-chip program RAM operates as a fixed cache thereby minimizing contention with accesses to external data memory spaces.



NOTE: Addresses $FFC0–$FFFF in X data memory are NOT available externally

Figure 1-2b DSP56003/005 Memory Maps

The bootstrap mode, described in Appendix A, provides a convenient, low cost method to load the DSP56003/005 program RAM with a program after power-on reset. It allows loading the program RAM from a single, inexpensive EPROM, or serially through the SCI, or via the Host Interface using a host processor.

1.3.2.5.2 X Data Memory

On-chip X data RAM is a 24-bit wide internal memory which occupies the lowest 256 locations in X memory space. The on-chip X data ROM can occupy locations 256 through 511 in X data memory space. The X data ROM is factory programmed with arctangent tables, useful in control applications. The on-chip peripheral registers occupy the top 64 locations. Addresses are received from the XAB, and data transfers to the data ALU occur on the XDB. X memory may be expanded to the full 64k off-chip.

1.3.2.5.3 Y Data Memory

On-chip Y data RAM is a 24-bit wide internal memory which occupies the lowest 256 locations in Y memory space. The on-chip Y data ROM can occupy locations 256 through 511 in Y data memory space. The Y data ROM is factory programmed with a full four quadrant sine table, useful for FFTs, DFTs, and wave form generation. It is recommended that any off-chip peripheral registers be mapped into the top 64 Y-data memory locations (to take advantage of the MOVEP instruction). Addresses are received from the YAB, and data transfers to the data ALU occur on the YDB. Y memory may be expanded to the full 64k off-chip.

1.3.2.5.4 Bootstrap ROM

Bootstrap ROM is a 96 location by 24-bit factory programmed ROM which is used to load the on-chip program RAM with the desired code prior to normal operation. The Bootstrap ROM is not accessible by the user and is disabled in normal operating modes.

1.3.2.6 Program Control Unit

The program control unit performs instruction prefetch, instruction decoding, hardware DO loop control, and exception processing. It contains six directly addressable registers — the program counter (PC), loop address (LA), loop counter (LC), status register (SR), operating mode register (OMR), and stack pointer (SP). The Program Control Unit also contains a 15 level by 32-bit system stack memory. The 16-bit PC can address 65,536 (64k) locations in program memory space.

1.3.2.7 Phase-locked Loop (PLL)

The PLL allows the processor to operate at a high internal clock frequency using a low frequency clock input. Lower frequency clock inputs reduce the overall electromagnetic interference generated by a system, and the ability to oscillate at different frequencies allows greater flexibility while reducing costs by eliminating the need for additional oscillators in a system.

The PLL performs frequency multiplication to allow the processor to use almost any available external system clock for full speed operation, while also supplying an output clock synchronized to the synthesized internal core clock. It also improves the synchronous timing of the processor's external memory port, significantly reducing the timing skew between EXTAL and the internal chip phases. The PLL is unusual in that it provides a low power divider on its output, which can reduce or restore the chip operating frequency without losing the PLL lock.

1.3.2.8 On-chip Emulator (OnCE™) Port

OnCE Port circuitry provides a sophisticated debugging tool that allows simple, inexpensive, and speed independent access to the processor's internal registers and peripherals. OnCE Port tells the application programmer the exact status of the registers, memory locations, and buses, as well as storing the last five instructions that were executed. OnCE Port capabilities are accessible through a set of pins which are standard on most of the members of the DSP56000 processor families.

1.3.2.9 Input/Output

A variety of system configurations are facilitated by the DSP56003/005 I/O structure. These configurations include multiple DSP56003/005 systems with or without a host processor, global bus systems with bus arbitration (DSP56003 only), and many serial configurations; all with minimal glue logic. Each I/O interface has its own control, status, and double buffered data registers which are memory-mapped in the X-data memory space. Each interface has several dedicated interrupt vector addresses and control bits to enable/disable interrupts (see Figure 1-2a). These interrupt vector addresses minimize the overhead associated with servicing an interrupt by immediately executing the appropriate service routine, sometimes without a context switch. Each interrupt can be programmed to one of three maskable priority levels.

Specifically, the I/O structure consists of Port A, ten peripherals, and up to 25 additional I/O pins as well as four general-purpose interrupt pins, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$, and $\overline{\text{IRQD}}$. The 25 additional pins may be used as GPIO pins or allocated to four of the ten on-chip peripherals under software control. The ten peripherals provided on the DSP56003/005 are:

- one 8-bit parallel Host Interface (HI) — 15 GPIO pins
- one Serial Communications Interface (SCI) — three GPIO pins
- one Synchronous Serial Interface (SSI) — six GPIO pins
- one Timer/Event Counter — one GPIO pins
- five Pulse Width Modulators (PWMs) — no GPIO pins
- one Watchdog Timer — no pins

1.3.2.9.1 External Memory Interface (Port A)

The DSP56003/005 expansion port is designed to synchronously interface over a common 24-bit data bus with a wide variety of memory and peripheral devices such as high speed static RAMs, slower memory devices, and other DSPs and MPUs in master/slave configurations. This capability is possible because the expansion bus timing is programmable. The expansion bus timing is controlled by a bus control register (BCR). The BCR controls the timing of the bus interface signals $\overline{\text{RD}}$ and $\overline{\text{WR}}$, and the data lines. Each of four memory spaces X data, Y data, Program data, and I/O has its own 4-bit BCR which can be programmed for up to 15 WAIT states (one WAIT state is equal to a clock period or equivalently, one-half of an instruction cycle). In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces.

1.3.2.9.2 General Purpose I/O (HI, SCI, SSI, Timer/Event Counter)

Each Host Interface, SCI, SSI, and Timer/Event Counter pin may be programmed as a general purpose I/O pin or as a dedicated on-chip peripheral pin under software control. Associated with each general purpose port is a data direction register which programs each pin as an input or output, and a data register for data I/O. These registers are read/write making the use of bit manipulation instructions extremely effective.

1.3.2.9.3 Host Interface (HI)

The Host Interface is a byte-wide, full duplex parallel port which may be connected directly to the data bus of a host processor. The host processor may be any of a number of industry standard microcomputers or microprocessors, another DSP, or DMA hardware. This Host Interface is identical to the ones on the DSP56001 and DSP56002.

The Host Interface appears to the host processor as a memory mapped peripheral occupying eight bytes in the host processor address space. Separate transmit and receive data registers are double-buffered to allow the DSP56003/005 and host processor to transfer data efficiently at high speed. The host processor can use standard data move instructions and addressing modes to communicate with the Host Interface. Handshake flags are provided for polled or interrupt driven data transfers with the host processor or DMA hardware may be used to transfer data without host processor intervention.

One of the most innovative features of the Host Interface is the Host Command feature. The host processor can issue vectored exception requests to the DSP56003/005 and may select any one of 128 DSP56003/005 exception routines to be executed. This flexibility allows the host programmer to execute up to 128 functions preprogrammed in the DSP56003/005.

1.3.2.9.4 Serial Communication Interface (SCI)

The SCI provides an asynchronous, full-duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems. This interface uses three dedicated pins — transmit data (TXD), receive data (RXD), and SCI serial clock (SCLK). It supports industry standard asynchronous bit rates up to CLK/4, i.e. 12.5 MHz for a 50 MHz clock, and protocols as well as high speed (up to system clock/8; i.e. 6.25 Mb/s for a 50 MHz clock) synchronous data transmission. The asynchronous protocols include a multidrop mode for master/slave operation. The Serial Communication Interface consists of separate transmit and receive sections having operations which can be asynchronous with respect to each other. A programmable baud rate generator is included to generate the transmit and/or receive clocks.

The baud rate generator can function as a general purpose timer when it is not being used by the SCI peripheral. This Serial Communication Interface is identical to the ones on the DSP56001 and DSP56002.

1.3.2.9.5 Synchronous Serial Interface (SSI)

The SSI is an extremely flexible, full-duplex serial interface which allows the DSP56003/005 to communicate with a variety of serial devices. These include industry standard codecs, other DSPs, microprocessors, and peripherals. Each of the following characteristics of the SSI can be independently defined: the number of bits per word (8, 12, 16, or 24), the protocol or mode (Normal, Network, or On-demand), the clock (up to system clock/4; i.e. 12.5 Mb/s for a 50 MHz clock), and the transmit/receive synchronization (word or bit length frame sync).

The Normal mode is typically used to interface with devices on a regular or periodic basis. In this mode the SSI functions with one data word of I/O per frame.

The Network mode provides time slots in addition to a bit clock and frame synchronization pulse. The SSI functions with from 2 to 32 words of I/O per frame in the Network mode. This mode is typically used in star or ring Time Division Multiplex networks with other DSP56003/005s and/or codecs.

The On-Demand mode is a data driven mode. There are no time slots defined. This mode is intended to be used to interface to devices on a non-periodic basis. The clock can be programmed to be continuous or gated.

This Synchronous Serial Interface is identical to the ones on the DSP56001 and DSP56002.

1.3.2.9.6 Timer/Event Counter

The Timer/Event Counter can use internal or external clocking and has a resolution of CLK/2. The counter is 24 bits long. It can also interrupt the processor after a number of events (clocks) specified by a user program, or it can signal an external device after counting internal events.

The Timer/Event Counter can be used as an external event counter, to measure external pulse width/signal periods, or to generate a timer pulse.

This Timer/Event Counter is identical to the one on the DSP56002.

1.3.2.9.7 Pulse Width Modulators (PWM)

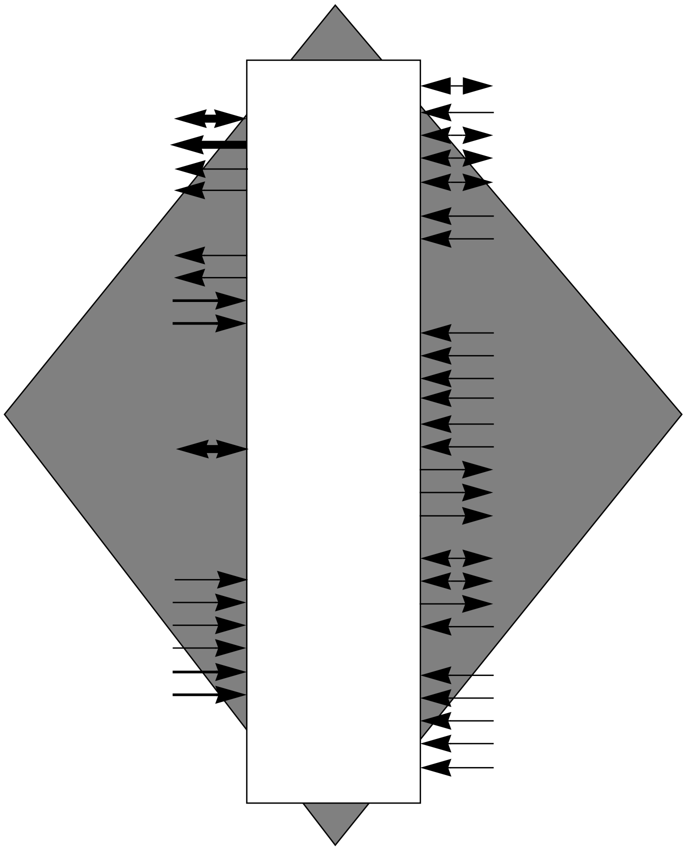
There are a total of five pulse width modulators on the DSP56003/005. Three of these use 9-bit to 16-bit signed two's complement fractional data and two use 9-bit to 16-bit positive fractional data. These modulators can be used to provide fixed pulse width signals. However, there is a separate interrupt vector location for each of the five PWMs which makes it easy to reprogram each PWM after every carrier cycle. Very short pulse widths and high repetition rates are possible since the maximum PWM clock rate is 1/2 of the DSP core clock rate.

1.3.2.9.8 Watchdog Timer

The Watchdog Timer uses the DSP core clock to run a count down timer. When that timer times out, the Watchdog Timer generates a non-maskable interrupt that uses the same vector address as the NMI exception vector. This timer can be used to detect a program that is caught in a loop or any other failure (software or hardware) that prevents the program from resetting the watchdog timer before it times out. The DSP can then either reset and reinitialize the system or run the appropriate error reporting/recovery program.

SECTION 2

PIN DESCRIPTIONS



SECTION CONTENTS

2.1 INTRODUCTION

This section introduces pins associated with the DSP56003/005 (see Figure 2-1). It divides the pins into their functional groups and explains the role each pin plays in the operation of the chip. It acts as a reference for following chapters which explain the chip's peripherals in detail.

2.2 PIN DESCRIPTIONS

The DSP56003 is available in a 176 pin thin quad flat pack (TQFP), see the *DSP56003/DSP56005 Data Sheet*. The DSP56005 is available in a 144 TQFP. The pins are organized into the functional groups indicated in Table 2-1. The signals are discussed in the paragraphs that follow.

Table 2-1 Functional Pin Groupings

Functional Group	DSP56003 Pins	DSP56005 Pins
Address Bus	16	16
Data Bus	24	24
Bus Control	11	6
Host Interface (HI)	15	15
Serial Communications Interface (SCI)	3	3
Synchronous Serial Interface (SSI)	6	6
Timer/Event Counter	1	1
Pulse Width Modulator A (PWMA)	10	10
Pulse Width Modulator B (PWMB)	4	4
On-chip Emulation (OnCE) Port	4	4
Power (V_{CC})	18	17
Ground (GND)	42	26
Interrupt and Mode Control	6	6
Phase-locked Loop (PLL) and Clock	7	5
Reserved	9	1
Total Number of Pins	176	144

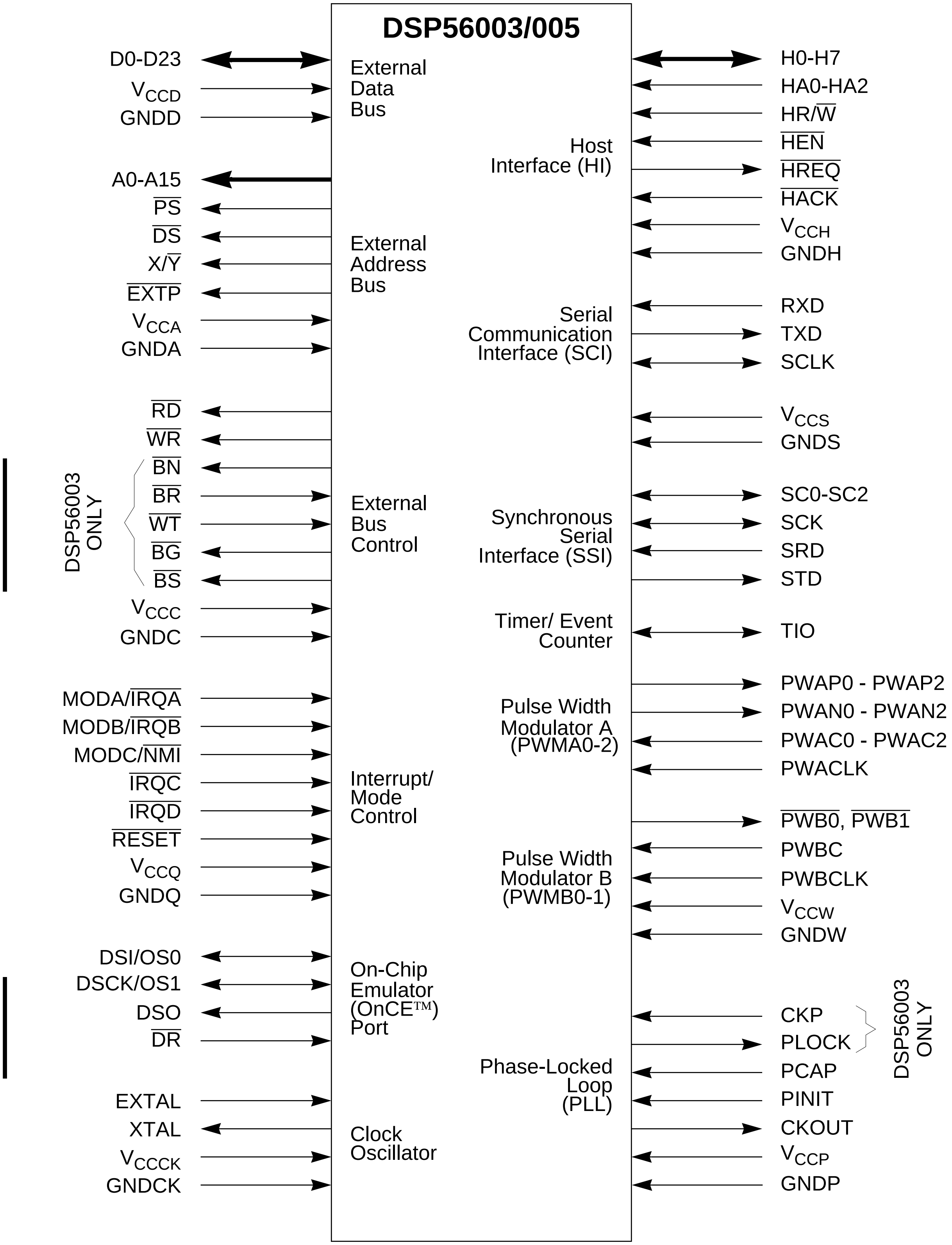


Figure 2-1 DSP56003/005 Signals

All unused inputs should have pull-up resistors for two reasons:

1. floating inputs draw excessive power
2. a floating input can cause erroneous operation

For example, during reset, all signals are three-stated. Without pull-up resistors, the $\overline{\text{BR}}$ and $\overline{\text{WT}}$ signals may appear to be active, causing two or more memory chips to try to simultaneously drive the external bus, which can damage the memory chips. A pull-up resistor in the 50K-ohm range should be sufficient.

Also, for future enhancements, all reserved, ‘no connect’ (NC), pins should be left unconnected.

2.2.1 Port A Address Bus, Data Bus, and Basic Bus Control

The Port A address and data bus signals control the access to external memory. These signals are three-stated during reset unless noted otherwise, and may require pull-up resistors to minimize power consumption and to prevent erroneous operation.

2.2.1.1 Address Bus (A0–A15) — three-state, outputs

A0-A15 specify the address for external program and data memory accesses. If there is no external bus activity, A0-A15 remain at their previous values. A0-A15 are three-stated during hardware reset.

2.2.1.2 Data Bus (D0–D23) — three-state, bidirectional input/outputs

Data for external memory I/O is presented on D0-D23. If there is no external bus activity, D0-D23 are three-stated. D0-D23 are also three-stated during hardware reset.

2.2.1.3 Program Memory Select ($\overline{\text{PS}}$) — three-state, active low output

This output is asserted only when external program memory is referenced (see Table 2-2). $\overline{\text{PS}}$ timing is the same as the A0-A15 address lines. If the external bus is not used during an instruction cycle, $\overline{\text{PS}}$ is driven high. $\overline{\text{PS}}$ is three-stated during hardware reset.

2.2.1.4 Data Memory Select ($\overline{\text{DS}}$) — three-state, active low output

This three-state output is asserted only when external data memory is referenced (see Table 2-2). If the external bus is not used during an instruction cycle, $\overline{\text{DS}}$ is driven high. $\overline{\text{DS}}$ is three-stated during hardware reset.

Table 2-2 Program and Data Memory Select Encoding

$\overline{\text{PS}}$	$\overline{\text{DS}}$	X/$\overline{\text{Y}}$	External Memory Reference
1	1	1	No Activity
1	0	1	X Data Memory on Data Bus
1	0	0	Y Data Memory on Data Bus
0	1	1	Program Memory on Data Bus (Not an Exception)
0	1	0	External Exception Fetch: Vector or Vector +1 (Development Mode Only)
0	0	X	Reserved
1	1	0	Reserved

2.2.1.5 X/$\overline{\text{Y}}$ Select (X/$\overline{\text{Y}}$) — three-state output

This three-state output selects which external data memory space (X or Y) is referenced by $\overline{\text{DS}}$ (see Table 2-2). X/$\overline{\text{Y}}$ is three-stated during hardware reset.

2.2.1.6 Read Enable ($\overline{\text{RD}}$) — three-state, active low output

This output is asserted during external memory read cycles. When $\overline{\text{RD}}$ is asserted, the data bus pins D0-D23 become inputs, and an external device is enabled onto the data bus. When $\overline{\text{RD}}$ is deasserted, the external data is latched inside the DSP. When $\overline{\text{RD}}$ is asserted, it qualifies the A0-A15, $\overline{\text{PS}}$ and $\overline{\text{DS}}$ pins. $\overline{\text{RD}}$ can be connected directly to the $\overline{\text{OE}}$ pin of a static RAM or ROM. $\overline{\text{RD}}$ is three-stated during hardware reset.

2.2.1.7 Write Enable ($\overline{\text{WR}}$) — three-state, active low output

This output is asserted during external memory write cycles. When $\overline{\text{WR}}$is asserted, the data bus pins D0-D23 become outputs, and the DSP puts data on the bus. When $\overline{\text{WR}}$is deasserted, the external data is latched inside the external device. When $\overline{\text{WR}}$ is asserted, it qualifies the A0-A15, $\overline{\text{PS}}$ and $\overline{\text{DS}}$ pins. $\overline{\text{WR}}$ can be connected directly to the $\overline{\text{WE}}$ pin of a static RAM. $\overline{\text{WR}}$ is three-stated during hardware reset.

2.2.1.8 External Peripheral ($\overline{\text{EXTP}}$) — active low output

The $\overline{\text{EXTP}}$ pin is an output asserted whenever the external Y memory I/O space (Y:$FFC0-$FFFF) is accessed. This signal simplifies generating peripheral enable signals. No additional circuitry is needed if only one external peripheral is used. For most applications, no more than one decode chip is needed and, as a result, decode delays are minimized. Using the Y memory I/O space allows the MOVEP instruction to be used to send and to receive data. Using the MOVEP instruction may allow the entire I/O routine to fit in a fast interrupt. $\overline{\text{EXTP}}$ is three-stated during hardware reset.

2.2.2 Enhanced Bus Control

These additional bus control pins are only available on the DSP56003. They provide a means to connect additional bus masters which may be additional DSPs, microprocessors, direct memory access (DMA) controllers, etc. through port A to the DSP56003. The bus control signals are three-stated during reset unless noted otherwise and require pull-up resistors to prevent erroneous operation.

2.2.2.1 Bus Needed ($\overline{\mathrm{BN}}$) — active low output — DSP56003 Only

The $\overline{\mathrm{BN}}$ output pin is asserted whenever the chip requires the external memory expansion port (Port A). During instruction cycles where the external bus is not required, $\overline{\mathrm{BN}}$ is deasserted. If an external device has requested the bus by asserting the $\overline{\mathrm{BR}}$ input and the DSP has granted the bus (by asserting $\overline{\mathrm{BG}}$), the DSP will continue processing as long as no external accesses are required. If an external access is required and the chip is not the bus master, it will stop processing and remain in wait states until bus ownership is returned. If the $\overline{\mathrm{BN}}$ pin is asserted when the chip is not the bus master, the chip's processing has stopped and the DSP is waiting to acquire bus ownership. An external arbiter may use this pin to help decide when to return bus ownership to the DSP. During hardware reset, $\overline{\mathrm{BN}}$ is deasserted.

Note: The $\overline{\mathrm{BN}}$ pin cannot be used as an early indication of imminent external bus access because it is valid later than the other bus control signal $\overline{\mathrm{BS}}$.

2.2.2.2 Bus Request ($\overline{\mathrm{BR}}$) — active low input — DSP56003 Only

The bus request $\overline{\mathrm{BR}}$ allows another device such as a processor or DMA controller to become the master of the DSP external data bus D0-D23 and external address bus A0-A15. The DSP asserts $\overline{\mathrm{BG}}$ after the $\overline{\mathrm{BR}}$ input is asserted. The DSP bus controller releases control of the external data bus D0-D23, address bus A0-A15 and bus control pins $\overline{\mathrm{PS}}$, $\overline{\mathrm{DS}}$, X/$\overline{\mathrm{Y}}$, $\overline{\mathrm{RD}}$, and $\overline{\mathrm{WR}}$ at the earliest time possible consistent with proper synchronization after the execution of the current instruction has been completed. These pins are then placed in the high impedance state and the $\overline{\mathrm{BG}}$ pin is asserted. The DSP continues executing instructions only if internal program and data memory resources are accessed. If the DSP requests the external bus while $\overline{\mathrm{BR}}$ input pin is asserted, the DSP bus controller inserts wait states until the external bus becomes available ($\overline{\mathrm{BR}}$ and $\overline{\mathrm{BG}}$ deasserted). When $\overline{\mathrm{BR}}$ is deasserted, the DSP will again assume bus mastership. $\overline{\mathrm{BR}}$ is an input during reset.

Notes:

1. Interrupts are not serviced when a DSP instruction is waiting for the bus controller.

2. $\overline{\mathrm{BR}}$ is prevented from interrupting the execution of a read/modify/write instruction.

3. To prevent erroneous operation, the $\overline{\mathrm{BR}}$ pin should be pulled up when it is not in use.

2.2.2.3 Bus Grant ($\overline{\mathrm{BG}}$) — active low output — DSP56003 Only

This pin is asserted to acknowledge an external bus request. It indicates that the DSP has released control of the external address bus A0-A15, data bus D0-D23 and bus control pins $\overline{\mathrm{PS}}$, $\overline{\mathrm{DS}}$, X/$\overline{\mathrm{Y}}$, $\overline{\mathrm{EXTP}}$, $\overline{\mathrm{RD}}$, and $\overline{\mathrm{WR}}$. The $\overline{\mathrm{BG}}$ output is asserted in response to a $\overline{\mathrm{BR}}$ input. When the $\overline{\mathrm{BG}}$ output is asserted, the external address bus A0-A15, data bus D0-D23 and bus control pins are in the high impedance state. $\overline{\mathrm{BG}}$ assertion may occur in the middle of an instruction which requires more than one external bus cycle for execution. Note that $\overline{\mathrm{BG}}$assertion will not occur during indivisible read-modify-write instructions (BSET, BCLR, BCHG). When $\overline{\mathrm{BR}}$ is deasserted, the $\overline{\mathrm{BG}}$ output is deasserted and the DSP regains control of the external address bus, data bus, and bus control pins. This output is deasserted during hardware reset.

2.2.2.4 Bus Strobe ($\overline{\mathrm{BS}}$) — active low output — DSP56003 Only

Bus Strobe is asserted at the start of a bus cycle and deasserted at the end of the bus cycle. This pin can be used as an “early bus start” signal by an address latch and as an “early bus end” signal by an external bus controller. It may also be used with the bus wait input, $\overline{\mathrm{WT}}$, to generate wait states, a feature which provides capabilities such as:

- connecting slower asynchronous devices to the DSP
- allowing devices with differing timing requirements to reside in the same memory space
- allowing a bus arbiter to provide a fast multiprocessor bus access
- providing an alternative to the WAIT and STOP instructions to halt the DSP at a known program location and have a fast restart

This output is deasserted during hardware reset.

2.2.2.5 Bus Wait ($\overline{\mathrm{WT}}$) — active low input — DSP56003 Only

This input allows an external device to force the DSP to generate wait states for as long as $\overline{\mathrm{WT}}$ is asserted. If $\overline{\mathrm{WT}}$ is asserted while $\overline{\mathrm{BS}}$ is asserted, wait states will be inserted into the current cycle. See the *DSP56003/005 Data Sheet* for timing details. $\overline{\mathrm{WT}}$ is an input during reset.

2.2.3 Host Interface

The following paragraphs discuss the host interface signals, which provide a convenient connection to another processor.

2.2.3.1 Host Data Bus (H0–H7) — bidirectional

This bidirectional data bus is used to transfer data between the host processor and the DSP. This bus is an input unless enabled by a host processor read. It is high impedance when $\overline{\text{HEN}}$ is deasserted. H0-H7 may be programmed as Port B general purpose parallel I/O pins called PB0-PB7 when the Host Interface (HI) is not being used. These pins are configured as GPIO input pins during hardware reset.

2.2.3.2 Host Address (HA0–HA2) — input*

These inputs provide the address selection for each HI register and must be stable when $\overline{\text{HEN}}$ is asserted. HA0-HA2 may be programmed as Port B general purpose parallel I/O pins called PB8-PB10 when the HI is not being used. These pins are configured as GPIO input pins during hardware reset.

2.2.3.3 Host Read/Write (HR/$\overline{\text{W}}$) — input*

This input selects the direction of data transfer for each host processor access. If HR/$\overline{\text{W}}$ is high and $\overline{\text{HEN}}$ is asserted, H0-H7 are outputs, and DSP data is transferred to the host processor. If HR/$\overline{\text{W}}$ is low and $\overline{\text{HEN}}$ is asserted, H0-H7 are inputs and host data is transferred to the DSP when $\overline{\text{HEN}}$ is deasserted. When $\overline{\text{HEN}}$ is asserted, HR/$\overline{\text{W}}$ must be stable. HR/$\overline{\text{W}}$ may be programmed as a general purpose I/O pin called PB11 when the HI is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.3.4 Host Enable ($\overline{\text{HEN}}$) — active low input*

This input enables a data transfer on the host data bus. When $\overline{\text{HEN}}$ is asserted and HR/$\overline{\text{W}}$ is high, H0-H7 becomes an output and DSP data may be latched by the host processor. When $\overline{\text{HEN}}$ is asserted and HR/$\overline{\text{W}}$ is low, H0-H7 is an input and host data is latched inside the DSP when $\overline{\text{HEN}}$ is deasserted. Normally a chip select signal derived from host address decoding and an enable clock is connected to the Host Enable. $\overline{\text{HEN}}$ may be programmed as a general purpose I/O pin called PB12 when the HI is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.3.5 Host Request ($\overline{\text{HREQ}}$) — active low output*

This open-drain output signal is used by the DSP to request service from the host processor. $\overline{\text{HREQ}}$ may be connected to a host processor interrupt request pin, a DMA controller transfer request pin, or a control input to external circuitry. $\overline{\text{HREQ}}$ is asserted when an enabled request occurs in the HI. $\overline{\text{HREQ}}$ is deasserted when the enabled request is cleared or masked, DMA $\overline{\text{HACK}}$ is asserted, or the DSP is reset. $\overline{\text{HREQ}}$ may be programmed as a general purpose I/O pin (not open-drain) called PB13 when the HI is not being used. This pin is configured as a GPIO input pin during hardware reset.

* Note that these pins can be inputs or outputs when programmed as general purpose I/O.

2.2.3.6 Host Acknowledge ($\overline{\text{HACK}}$) — active low input*

This input has two functions:

- to provide a host acknowledge signal for DMA transfers
- to control handshaking and to provide a host interrupt acknowledge compatible with MC68000 family processors

If programmed as a host acknowledge signal, $\overline{\text{HACK}}$ may be used as a data strobe for HI DMA data transfers. If programmed as an MC68000 host interrupt acknowledge, $\overline{\text{HACK}}$ enables the HI Interrupt Vector Register (IVR) onto the host data bus H0-H7 if the Host Request $\overline{\text{HREQ}}$ output is asserted. In this case, all other HI control pins are ignored and the HI state is not affected. $\overline{\text{HACK}}$ may be programmed as a general purpose I/O pin called PB14 when the HI is not being used. For more details about the programming options for this pin, see Section 5.3.4.6 — Host Acknowledge (HACK). This pin is configured as a GPIO input pin during hardware reset.

Note: $\overline{\text{HACK}}$ should always be pulled high when not in use.

2.2.4 Serial Communication Interface (SCI)

The following signals relate to the SCI. They are introduced briefly here and described in more detail in Section 6 — *Serial Communications Interface*.

2.2.4.1 Receive Data (RXD) — input*

This input receives byte-oriented data and transfers the data to the SCI receive shift register. Input data is sampled on the positive or the negative edge of the receive clock, depending on how the SCI control register is programmed. RXD may be programmed as a general-purpose I/O pin called PC0 when it is not being used as an SCI pin. This pin is configured as a GPIO input pin during hardware reset.

2.2.4.2 Transmit Data (TXD) — output*

This output transmits serial data from the SCI transmit shift register. Data changes on the negative edge of the transmit clock. This output is stable on the positive or the negative edge of the transmit clock, depending on how the SCI control register is programmed. TXD may be programmed as a general-purpose I/O pin called PC1 when the SCI TXD function is not being used. This pin is configured as a GPIO input pin during hardware reset.

* Note that these pins can be inputs or outputs when programmed as general purpose I/O.

2.2.4.3 SCI Serial Clock (SCLK) — bidirectional

This bidirectional pin provides an input or output clock from which the transmit and/or receive baud rate is derived in the asynchronous mode, and from which data is transferred in the synchronous mode. SCLK may be programmed as a general-purpose I/O pin called PC2 when the SCI SCLK function is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.5 Synchronous Serial Interface (SSI)

The SSI pins SC0, SC1, SC2, SCK, SRD, and STD are introduced briefly here and are described in more detail in Section 7 — *Synchronous Serial Interface*.

2.2.5.1 Serial Control 0 (SC0) — bidirectional

This bidirectional pin's function is determined by whether the SSI is in synchronous or asynchronous mode. In synchronous mode, this pin is used for serial flag I/O. In asynchronous mode, this pin receives clock I/O. SC0 and SC1 are independent serial I/O flags but may be used together for multiple serial device selection. SC0 may be programmed as a general-purpose I/O pin called PC3 when the SSI SC0 function is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.5.2 Serial Control 1 (SC1) — bidirectional

The SSI uses this bidirectional pin to control flag or frame synchronization. This pin's function is determined by whether the SSI is in synchronous or asynchronous mode. In asynchronous mode, this pin is frame sync I/O. For synchronous mode with continuous clock, this pin is serial flag SC1 and operates like the SC0. SC0 and SC1 are independent serial I/O flags but may be used together for multiple serial device selection. SC1 may be programmed as a general-purpose I/O pin called PC4 when the SSI SC1 function is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.5.3 Serial Control 2 (SC2) — bidirectional

The SSI uses this bidirectional pin to control frame synchronization only. As with SC0 and SC1, its function is defined by the SSI operating mode. SC2 may be programmed as a general-purpose I/O pin called PC5 when the SSI SC2 function is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.5.4 SSI Serial Clock (SCK) — bidirectional

This bidirectional pin provides the serial bit rate clock for the SSI when only one clock is being used. SCK may be programmed as a general-purpose I/O pin called PC6 when it is not needed as an SSI pin. This pin is configured as a GPIO input pin during hardware reset.

2.2.5.5 SSI Receive Data (SRD) — input*

This input pin receives serial data into the SSI receive shift register. SRD may be programmed as a general-purpose I/O pin called PC7 when the SRD function is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.5.6 SSI Transmit Data (STD) — output*

This output pin transmits serial data from the SSI transmit shift register. STD may be programmed as a general-purpose I/O pin called PC8 when the STD function is not being used. This pin is configured as a GPIO input pin during hardware reset.

2.2.6 Timer/Event Counter Pin

The following pin is dedicated to the Timer/Event Counter operation.

2.2.6.1 Timer/Event Counter Input/Output (TIO) — bidirectional

The TIO pin provides an interface to the Timer/Event Counter module. When the module functions as an external event counter or is used to measure an external pulse width/signal period, the TIO is used as an input. When the module functions as a timer, the TIO is an output and the signal on the TIO pin is the timer pulse. When not used by the timer module, the TIO can act as a general purpose I/O pin. Reset disables the TIO pin and causes it to be three-stated.

2.2.7 Pulse Width Modulator A (PWMA)

Pulse Width Modulator A is a set of three 16-bit signed two's complement fractional data pulse width modulators and has 10 dedicated external pins. These pulse width modulators are independent of the PWMB modulators.

2.2.7.1 Pulse Width Modulator A Positive (PWAP0 - PWAP2) — output

These three pins are the positive outputs for the three PWMA modulators (PWMA0, PWMA1, and PWMA2). When a positive two's complement number is loaded in one of the three PWMA Count Registers, an output signal will be generated on the respective pin (e.g., loading PWACR0 with a positive two's complement number will generate an output on PWAP0).

* Note that these pins can be inputs or outputs when programmed as general purpose I/O.

When a negative two's complement number is loaded in a PWMA Count Register, PWAP0-PWAP2 will be at its inactive logic level (as defined by the polarity bits in the PWMA Control/Status Register 1). These pins are driven at their inactive logic level (as defined by the polarity bits in the Control/Status Register 1) when the individual PWM modulator (PWMA0, PWMA1, or PWMA2) is not enabled. During hardware reset, these pins are driven to a high logic level.

2.2.7.2 Pulse Width Modulator A Negative (PWAN0 - PWAN2) — output

These three pins are the negative outputs for the three PWMA modulators (PWMA0, PWMA1, and PWMA2). When a negative two's complement number is loaded in one of the three PWMA Count Registers, an output signal will be generated on the respective pin (e.g. loading PWACR0 with a negative two's complement number will generate an output on PWAN0).

When a positive two's complement number is loaded in a PWMA Count Register, the N-output (PWAN0-PWAN2) of this PWMA block will be at its inactive logic level (as defined by the polarity bits in the PWMA Control/Status Register 1). These pins are driven at their inactive logic level (as defined by the polarity bits in the Control/Status Register 1) when the individual PWM modulator (PWMA0, PWMA1, or PWMA2) is not enabled. During hardware reset, these pins are driven to a high logic level.

2.2.7.3 Pulse Width Modulator A Carrier (PWAC0 - PWAC2) — input

These three pins are inputs that provide the external carrier signals for the three PWMAs (PWMA0, PWMA1 and PWMA2). When the carrier source for the respective PWMA block is programmed to be external, the modulator starts operation at each rising edge of its carrier signal. While a PWMA block is either disabled, or is enabled and programmed to operate with the internal carrier, its respective internal input buffer is disconnected from the pin and no external pull-up is necessary.

2.2.7.4 Pulse Width Modulator A Clock (PWACLK) — input

This input increments the prescaler which connects to the three PWMA blocks and increments the counter in each these blocks. If all of the PWMA blocks are either disabled, or are programmed to use the internal clock, the internal input buffer is disconnected from the pin and no external pull-up is necessary.

2.2.8 Pulse Width Modulator B (PWMB)

Pulse Width Modulator B is a pair 16-bit positive fractional data pulse width modulators and has four dedicated external pins. These two pulse width modulators are independent of the PWMA modulators.

2.2.8.1 Pulse Width Modulator B Carrier (PWBC) — input

This pin is an input that provides the external carrier signals for the two PWMB blocks (PWMB0 and PWMB1). When the carrier source for these blocks is programmed to be external, these blocks start operation at each rising edge of this signal. While a PWMB block is either disabled, or is enabled and programmed to operate with the internal carrier, its respective internal input buffer is disconnected from the pin and no external pull-up is necessary.

2.2.8.2 Pulse Width Modulator B Output ($\overline{\text{PWB0}}$-$\overline{\text{PWB1}}$) — active low output

These two pins are the outputs for pulse width modulators PWMB0 and PWMB1. These pins are either open drain or driven at TTL levels depending on the programming of PWBCSR1 bit 14 (WBR0). These pins are also in the high-impedance state or in a high logic state (depending on the value of the bit WBO in PWBCSR1) when PWMB0 and PWMB1 are disabled. During hardware reset, these pins are in the high-impedance state.

2.2.8.3 Pulse Width Modulator B Clock (PWBCLK) — input

This input increments the prescaler which increments the counter connected to the two PWMB blocks. While both PWMB blocks are disabled, the internal input buffer is disconnected from the pin and no external pull-up is necessary. While the PWMB blocks are programmed to use the internal clock, the internal input buffer is disconnected from the pin and no external pull-up is necessary.

2.2.9 On-Chip Emulation (OnCE™) Port

The following paragraphs describe the pins associated with the OnCE Port controller and its serial interface.

2.2.9.1 Debug Serial Input/Chip Status 0 (DSI/OS0) — bidirectional

The DSI/OS0 pin, when an input, is the pin through which serial data or commands are provided to the OnCE port controller. The data received on the DSI pin will be recognized only when the DSP has entered the debug mode of operation. Data must have valid TTL logic levels before the serial clock falling edge. Data is always shifted into the OnCE serial port most significant bit (MSB) first. When the DSP is not in the debug mode, the DSI/OS0 pin provides information about the chip status if it is an output and used in conjunction with the OS1 pin. When switching from output to input, the pin is three-stated. During hardware reset, this pin is defined as an output and it is driven low.

Note: To avoid possible glitches, an external pull-down resistor should be attached to this pin.

2.2.9.2 Debug Serial Clock/Chip Status 1 (DSCK/OS1) — bidirectional

The DSCK/OS1 pin, when an input, is the pin through which the serial clock is supplied to the OnCE port. The serial clock provides pulses required to shift data into and out of the OnCE serial port. Data is clocked into the OnCE port on the falling edge and is clocked out of the OnCE serial port on the rising edge. If the DSCK/OS1 pin is an output and used in conjunction with the OS0 pin, it provides information about the chip status when the DSP is not in the debug mode. The debug serial clock frequency must be no greater than $^1/_8$ of the processor clock frequency. The pin is three-stated when it is changing from input to output. During hardware reset, this pin is defined as an output and is driven low.

Note: To avoid possible glitches, an external pull-down resistor should be attached to this pin.

2.2.9.3 Debug Serial Output (DSO) — output

The debug serial output provides the data contained in one of the OnCE port controller registers as specified by the last command received from the command controller. The most significant bit (MSB) of the data word is always shifted out of the OnCE serial port first. Data is clocked out of the OnCE Port serial port on the rising edge of DSCK.

The DSO pin also provides acknowledge pulses to the external command controller. When the chip enters the debug mode, the DSO pin will be pulsed low to indicate (acknowledge) that the OnCE Port is waiting for commands. After receiving a read command, the DSO pin will be pulsed low to indicate that the requested data is available and the OnCE Port serial port is ready to receive clock pulses in order to deliver the data. After receiving a write command, the DSO pin will be pulsed low to indicate that the OnCE serial port is ready to receive the data to be written; after the data is written, another acknowledge pulse will be provided.

During hardware reset and when idle, the DSO pin is held high.

2.2.9.4 Debug Request ($\overline{\text{DR}}$) — active low input

The debug request input provides a means of entering the debug mode of operation. This pin, when asserted, will cause the DSP to finish the current instruction being executed, to save the instruction pipeline information, to enter the debug mode, and to wait for commands to be entered from the debug serial input line. While the DSP is in the debug mode, the user can reset the OnCE Port controller by asserting $\overline{\text{DR}}$, waiting for an acknowledge from DSO, and then deasserting $\overline{\text{DR}}$. It may be necessary to reset the OnCE Port controller in cases where synchronization between the OnCE Port controller and external circuitry is lost. Asserting $\overline{\text{DR}}$ when the DSP is in the Wait or the Stop state, and keeping it asserted until an acknowledge pulse in the DSP is produced, sends the DSP into the debug mode. After receiving the acknowledge, $\overline{\text{DR}}$ must be deasserted before sending the first OnCE Port command. For more information, see Section 10.6 —*Methods Of Entering The Debug Mode* in the *DSP56000 Family Manual*.

2.2.10 Power and Ground

The power and ground pins are presented in the following paragraphs. There are ten sets of power and ground pins (see Table 2-3). In accordance with good engineering practice, V_{CC} should be bypassed to ground (as needed) by a 0.1 μF capacitor located as close as possible to the chip package. The two circuits where this bypassing is most important are the PLL and the core processor internal logic circuits. Refer to the pin assignments and layout practices section in the *DSP56003/005 Data Sheet* for additional information.

Table 2-3 Power and Ground Pins

Function	Pin Names		DSP56003		DSP56005	
	V_{CC}	GND	V_{CC}	GND	V_{CC}	GND
Address Bus	V_{CCA}	GNDA	3	5	3	5
Data Bus	V_{CCD}	GNDD	3	6	3	6
Bus Control	V_{CCC}	GNDC	1	1	1	1
Host Interface (HI)	V_{CCH}	GNDH	2	4	2	4
Port C (Serial Communications Interface, Synchronous Serial Interface)	V_{CCS}	GNDS	1	2	1	2
Pulse Width Modulator (PWM)	V_{CCW}	GNDW	1	2	1	2
Internal Logic	V_{CCQ}	GNDQ	5	4	4	4
Phase-locked Loop (PLL)	V_{CCP}	GNDP	1	1	1	1
Clock	V_{CCCK}	GNDCK	1	1	1	1
Thermal	—	GND	0	16	0	0

2.2.10.1 Power

These pins provide power to the circuits listed below. The voltage should be well regulated and the pin should be provided with an extremely low impedance path to the power rail.

- Address Bus Output Buffer Power (V_{CCA})
- Data Bus Output Buffer Power (V_{CCD})
- Bus Control Power (V_{CCC})
- Host Interface Power (V_{CCH})
- Serial Power (V_{CCS})
- PWM Power (V_{CCW})
- Internal Logic Power (V_{CCQ}) — core processor internal logic circuits
- PLL Circuit Power (V_{CCP})
  This pin supplies a quiet power source to the Phase-Locked Loop (PLL) to provide greater frequency stability. The voltage should be well regulated and the pin should be provided with an extremely low impedance path to the power rail. V_{CCP} should be bypassed to GNDP by a 0.1 µF capacitor located as close as possible to the chip package.
- Clock Power (V_{CCCK}) — CKOUT circuitry

2.2.10.2 Ground

These pins provide grounds for the circuits listed below. The pins should be provided with an extremely low impedance path to ground.

- Address Bus Output Buffer Ground (GNDA)
- Data Bus Output Buffer Ground (GNDD)
- Bus Control Ground (GNDC)
- Host Interface Ground (GNDH)
- Serial Ground (GNDS) — SCI, SSI, and their GPIO circuits
- PWM Ground (GNDW)
- Internal Logic Ground (GNDQ) — core processor internal logic circuits
- PLL Circuit Ground (GNDP)
  This pin supplies a quiet ground source to the PLL to provide greater frequency stability. The pin should be provided with an extremely low impedance path to ground. V_{CCP} should be bypassed to GNDP by a 0.1 µF capacitor located as close as possible to the chip package.
- Clock Ground (GNDCK) — CKOUT circuitry
- Thermal Ground (GND) — DSP56003 Only
  These pins provide a thermal enhancement (i.e. a heat sink) to the chip. The pins should be directly connected to the ground plane layer to help dissipate heat from the chip. This thermal connection is not necessary for operation.

However, it will help keep the chip within the thermal specifications when thermal specification limits are otherwise being approached.

2.2.11 Interrupt and Mode Control

The interrupt and mode control pins select the chip's operating mode as it comes out of hardware reset and receive interrupt requests from external sources after reset.

2.2.11.1 Mode Select A/External Interrupt Request A (MODA/$\overline{\text{IRQA}}$) — input

This input pin has three functions:

- to work with the MODB and MODC pins to select the chip's initial operating mode
- to allow an external device to request a DSP interrupt after internal synchronization
- to turn on the internal clock generator when the DSP in the Stop processing state, causing the chip to resume processing

MODA is read and internally latched in the DSP when the processor exits the reset state. MODA, MODB, and MODC select the initial chip operating mode. Several clock cycles after leaving the reset state, the MODA pin changes to the external interrupt request $\overline{\text{IRQA}}$. The chip operating mode can be changed by software after reset.

The $\overline{\text{IRQA}}$ input is a synchronized external interrupt request. It may be programmed to be level sensitive or negative edge triggered. When the signal is edge triggered, triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{IRQA}}$ will generate multiple interrupts also increases.

While the DSP is in the Stop processing state, asserting $\overline{\text{IRQA}}$ gates on the oscillator and, after a clock stabilization delay, enables clocks to the processor and peripherals. Hardware reset causes this input to act as MODA.

2.2.11.2 Mode Select B/External Interrupt Request B (MODB/$\overline{\text{IRQB}}$) — input

This input pin has two functions:

- to work with the MODA and MODC pins to select the chip's initial operating mode
- to allow an external device to request a DSP interrupt after internal synchronization

MODB is read and internally latched in the DSP when the processor exits the reset state. MODA, MODB, and MODC select the initial chip operating mode. Several clock cycles after leaving the reset state, the MODB pin changes to the external interrupt request $\overline{\text{IRQB}}$. The chip operating mode can be changed by software after reset.

The $\overline{\text{IRQB}}$ input is a synchronized external interrupt request. It may be programmed to be level sensitive or negative edge triggered. When the signal is edge triggered, triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{IRQB}}$ will generate multiple interrupts also increases.

Hardware reset causes this input to act as MODB.

2.2.11.3 Mode Select C/Non-Maskable Interrupt Request (MODC/$\overline{\text{NMI}}$) — edge triggered input

This input pin has two functions:

- to work with the MODA and MODB pins to select the chip's initial operating mode
- to allow an external device to request a DSP interrupt after internal synchronization

MODC is read and internally latched in the DSP when the processor exits the reset state. MODA, MODB, and MODC select the initial chip operating mode. Several clock cycles after leaving the reset state, the MODC pin changes to the non-maskable interrupt request, $\overline{\text{NMI}}$. The chip operating mode can be changed by software after reset.

The $\overline{\text{NMI}}$ input is a negative-edge triggered external interrupt request. This is a level 3 interrupt that can not be masked out. Triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{NMI}}$ will generate multiple interrupts also increases.

Hardware reset causes this input to act as MODC.

2.2.11.4 External Interrupt Request C ($\overline{\text{IRQC}}$) — edge triggered input

This negative edge triggered input allows an external device to request a DSP interrupt after internal synchronization. Triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{IRQC}}$ will generate multiple interrupts also increases.

2.2.11.5 External Interrupt Request D ($\overline{\text{IRQD}}$) — edge triggered input

This negative edge triggered input allows an external device to request a DSP interrupt after internal synchronization. Triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{IRQD}}$ will generate multiple interrupts also increases.

2.2.11.6 Reset ($\overline{\text{RESET}}$) — input

This input is a direct hardware reset of the processor. When $\overline{\text{RESET}}$ is asserted, the DSP is initialized and placed in the reset state. A Schmitt trigger input is used for noise immunity. When the reset pin is deasserted, the initial chip operating mode is latched from the MODA, MODB, and MODC pins. The chip also samples the PINIT pin and writes its status into the PEN bit of the PLL Control Register. On the DSP56003 *only*, the DSP samples the CKP pin to determine the polarity of the CKOUT signal. When the chip comes out of the reset state, deassertion occurs at a voltage level and is not directly related to the rise time of the $\overline{\text{RESET}}$ signal. However, the probability that noise on $\overline{\text{RESET}}$ will generate multiple resets increases with increasing rise time of the $\overline{\text{RESET}}$ signal.

2.2.12 Clock, Oscillator, and PLL Pins

The following pins are dedicated to the PLL, clock, and oscillator operation.

2.2.12.1 Output Clock (CKOUT) — output

This output pin provides a 50% (refer to the *DSP56003/DSP56005 Data Sheet* for absolute timings) duty cycle output clock synchronized to the internal processor clock when the PLL is enabled and locked. When the PLL is disabled, the output clock at CKOUT is derived from, and has the same frequency and duty cycle as, EXTAL.

Note: If the PLL is enabled and the multiplication factor is less than or equal to 4, then CKOUT is synchronized to EXTAL. (For information on the DSP56003/005's PLL multiplication factor, see Section 3.6 — *PLL Multiplication Factor* in the *DSP56000 Family Manual*.

2.2.12.2 CKOUT Polarity Control (CKP) — input. — DSP56003 Only

This input pin defines the polarity of the CKOUT clock output. Strapping CKP through a resistor to GND will make the CKOUT polarity the same as the EXTAL polarity. Strapping CKP through a resistor to V_{CC} will make the CKOUT polarity the inverse of the EXTAL polarity. The CKOUT clock polarity is internally latched at the end of the hardware reset, so that any changes of the CKP pin logic state after deassertion of hardware reset will not affect the CKOUT clock polarity.

2.2.12.3 External Clock/Crystal (EXTAL) — input

This pin may be used in one of two ways:

- driven from an external clock
- interface the internal crystal oscillator input to an external crystal circuit

If the PLL is enabled, this pin is internally connected to the on-chip PLL. The PLL can multiply the frequency on the EXTAL pin to generate the internal DSP clock. The PLL output is divided by two to produce a four-phase instruction cycle clock, with the minimum instruction time being two PLL output clock periods. If the PLL is disabled, EXTAL is divided by two to produce the four-phase instruction cycle clock.

2.2.12.4 Crystal (XTAL) — output

This output connects the internal crystal oscillator output to an external crystal. If an external clock is used, XTAL should not be connected. It may be disabled through software control using the XTLD bit in the PLL control register.

2.2.12.5 PLL Filter Capacitor (PCAP) — input

This input is used to connect a high quality external capacitor needed for the PLL filter. The capacitor should be as close as possible to the chip with heavy, short traces connecting one terminal of the capacitor to PCAP and the other terminal to V_{CCP}. The capacitor value is specified in the *DSP56003/005 Data Sheet*.

2.2.12.6 PLL Initialization (PINIT) — input

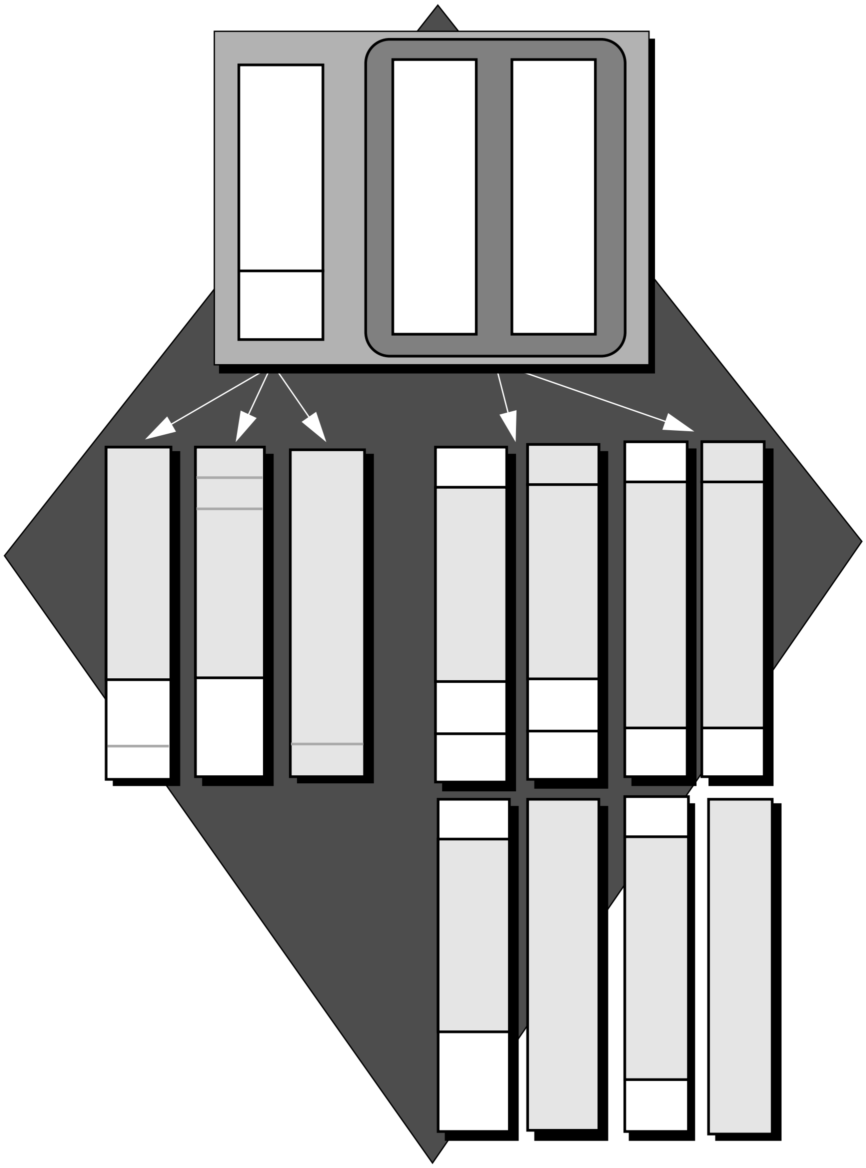
During the assertion of hardware reset, the value at the PINIT input pin is written into the PEN bit of the PLL control register. When high, the PEN bit enables the PLL by causing it to derive the internal clocks from the PLL voltage controlled oscillator output. When the bit is clear, the PLL is disabled and the chip’s internal clocks are derived from the clock connected to the EXTAL pin. After hardware reset is deasserted, the PINIT pin is ignored.

2.2.12.7 Phase and Frequency Locked (PLOCK) — output — DSP56003 Only

This signal originates from the PLL phase detector. The chip asserts PLOCK when the PLL is enabled and has locked on the proper phase and frequency of EXTAL. PLOCK is deasserted by the chip if the PLL is enabled and has not locked on the proper phase and frequency. The processor is halted when PLOCK is deasserted. PLOCK is asserted if the PLL is disabled. This signal is a reliable indicator of the PLL lock state only after the chip has exited the hardware reset state. During hardware reset, the PLOCK state is determined by PINIT and by the PLL lock condition.

SECTION 3

MEMORY, OPERATING MODES, AND INTERRUPTS



SECTION CONTENTS

3.1 MEMORY INTRODUCTION

The DSP56003/005 memory can be partitioned in several ways to provide high-speed parallel operation and additional off-chip memory expansion. Program and data memory are separate, and the data memory is, in turn, divided into two separate memory spaces, X and Y. Both the program and data memories can be expanded off-chip. There are also two on-chip data read-only memories (ROMs) that can overlay a portion of the X and Y data memories, and a bootstrap ROM that can overlay part of the program random-access memory (RAM). The data memories are divided into two independent spaces to work with the two address arithmetic logic units (ALUs) to feed two operands simultaneously to the data ALU.

The DSP operating modes determine the memory maps for program and data memories and the start-up procedure when the DSP leaves the reset state. This section describes the DSP56003/005 Operating Mode Register (OMR), its operating modes and their associated memory maps, and discusses how to set and reset operating modes.

This section also includes details of the interrupt vectors and priorities and describes the effect of a hardware reset on the PLL multiplication factor.

3.1.1 DSP56003/005 Data and Program Memory

The DSP56003/005 has 4608 words of program RAM, 96 words of bootstrap ROM, 256 words of RAM and 256 words of ROM for each of the X and Y internal data memories. The memory maps are shown in Figure 3-1a and Figure 3-1b.

3.1.1.1 Program Memory

The DSP56003/005 has 4608 words of program RAM and 96 words of factory-programmed bootstrap ROM.

The bootstrap ROM is programmed to perform the bootstrap operation from the memory expansion port (port A), from the host interface, or from the SCI. The bootstrap ROM provides a convenient, low cost method of loading the program RAM with a user program after power-on reset. The bootstrap ROM activity is controlled by the MA, MB, and MC bits in the OMR (see Section 3.2 — Operating Mode Register (OMR), for a complete explanation of the OMR and the DSP56003/005's operating modes and memory maps).

Addresses are received from the program control logic (usually the program counter) over the PAB. Program memory may be written using the program memory (MOVEM) instructions. The interrupt vectors are located in the bottom 128 locations ($0000-$007F) of program memory. Program memory may be expanded to 64K off-chip.

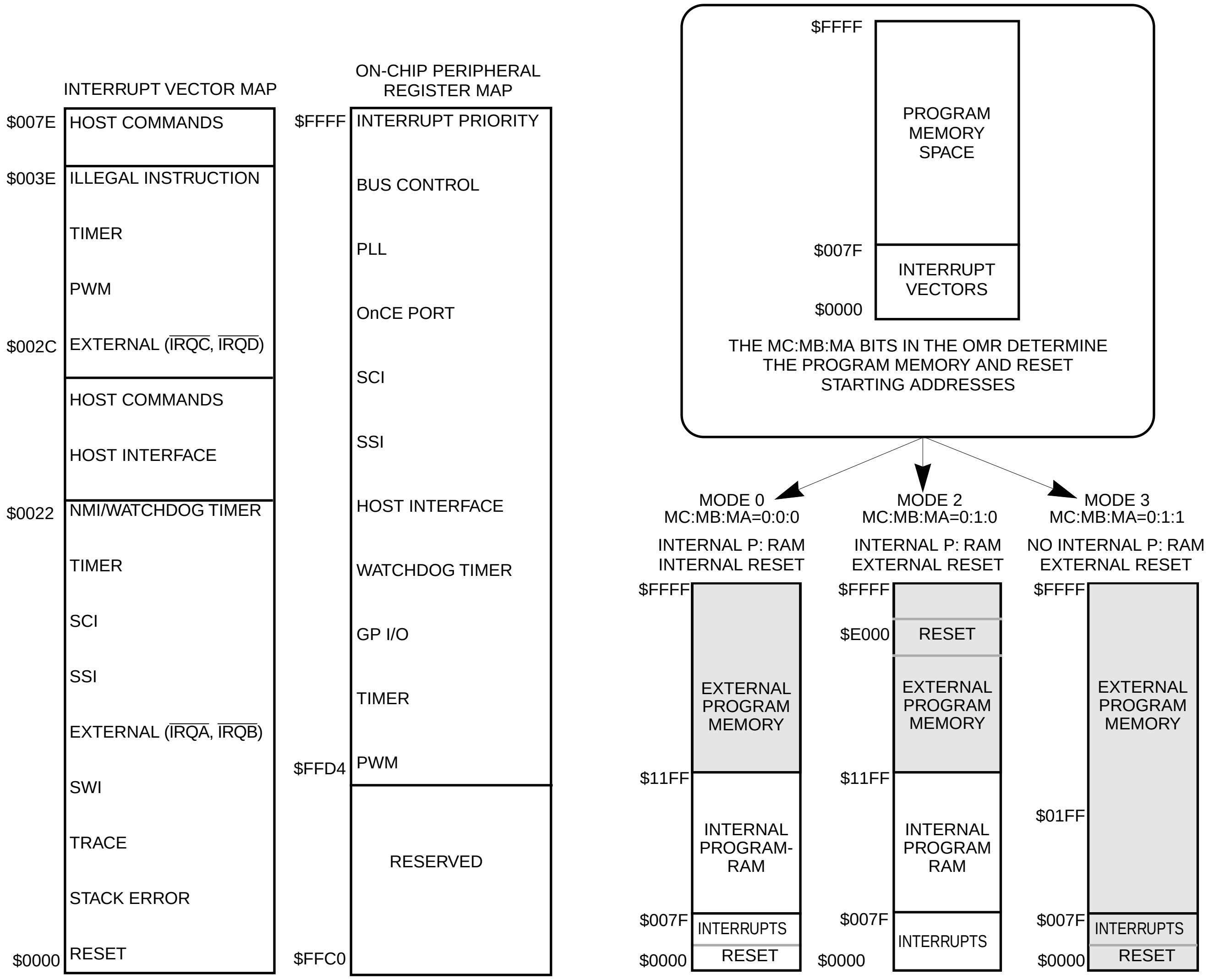


Figure 3-1a DSP56003/005 Memory Maps

3.1.1.2 X Data Memory

The on-chip X data RAM is a 24-bit-wide, internal static memory occupying the lowest 256 locations (0–255) in X memory space. The on-chip X data ROM occupies locations 256–511 in the X data memory space and is controlled by the DE bit in the OMR. (See the explanation of the DE bit in Section 3.2.2 — Data ROM Enable (DE) Bit 2. Also, see Figure 3-1a.) The on-chip peripheral registers occupy the top 64 locations of the X data memory ($FFC0–$FFFF). The 16-bit addresses are received from the X Address Bus, and 24-bit data transfers to the data ALU occur on the X Data Bus. The X memory may be expanded to 64K words off-chip.

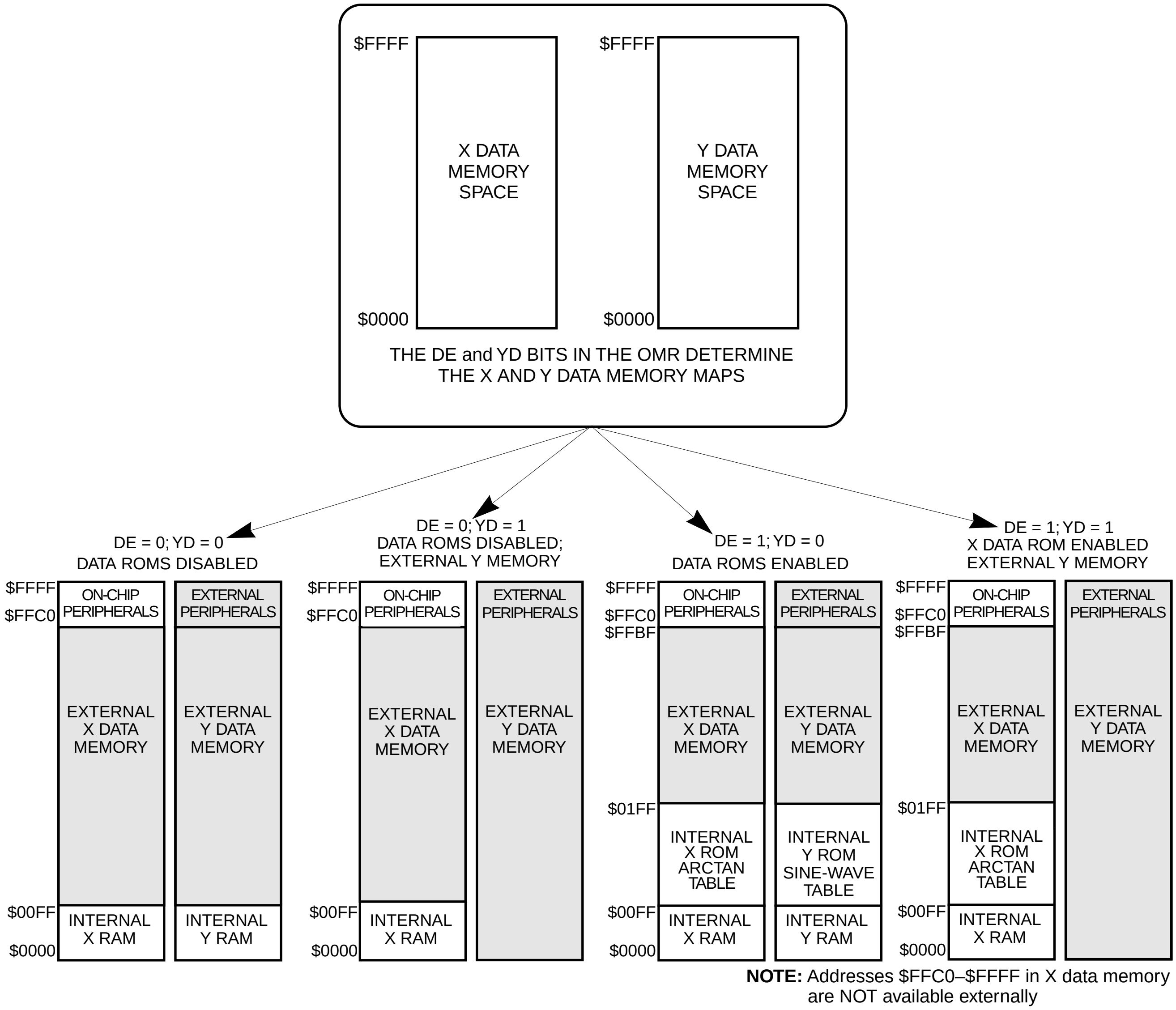


Figure 3-1b DSP56003/005 Memory Maps

3.1.1.3 Y Data Memory

The on-chip Y data RAM is a 24-bit-wide internal static memory occupying the lowest 256 locations (0–255) in the Y memory space. The on-chip Y data ROM occupies locations 256–511 in Y data memory space and is controlled by the DE and YD bits in the OMR. (See the explanations of the DE and YD bits in Section 3.2.2 — OMR Data ROM Enable (DE) Bit 2 and Section 3.2.3 — OMR Internal Y Memory Disable Bit (YD) Bit 3, respectively. Also, see Figure 3-1a.) The 16-bit addresses are received from the Y Address Bus, and 24-bit data transfers to the data ALU occur on the Y Data Bus. Y memory may be expanded to 64K off-chip.

Note: The off-chip peripheral registers should be mapped into the top 64 locations ($FFC0–$FFFF) to take advantage of the move peripheral data (MOVEP) instruction. The $\overline{\text{EXTP}}$ pin indicates when these memory locations are being accessed and can be used to reduce the address decode logic required to generate chip enable signals.

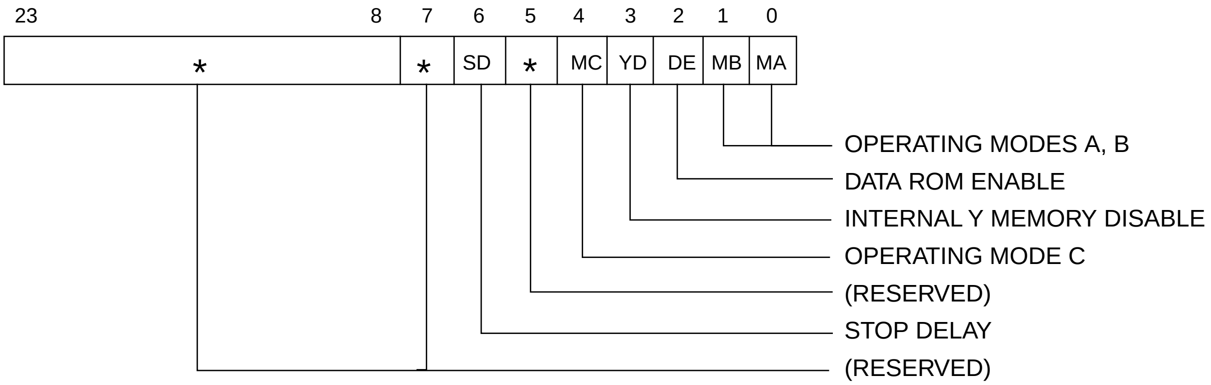


Figure 3-2 OMR Format

3.2 DSP56003/005 OPERATING MODE REGISTER (OMR)

Operating modes determine the memory maps for program and data memories, and the start-up procedure when the DSP leaves the reset state. The processor samples the MODA, MODB, and MODC pins as it leaves the reset state, establishes the initial operating mode, and writes the operating mode information to the Operating Mode Register. When the processor leaves the reset state, the MODA and MODB pins become general-purpose interrupt pins, $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$, respectively, and the MODC pin becomes the non-maskable interrupt pin $\overline{\text{NMI}}$.

The OMR is a 24-bit register (only six bits are defined) that controls the current operating mode of the processor. It is located in the DSP56003/005's Program Control Unit (described in Section 5 of the *DSP56000 Family Manual*). The OMR bits are only affected by processor reset and by the ANDI, ORI, MOVEC, BSET, BCLR, and BCHG instructions, which directly reference the OMR. The OMR format for the DSP56003/005 is shown in Figure 3-2.

3.2.1 OMR Chip Operating Mode (MC, MB, MA) Bits 4, 1, and 0

The chip operating mode bits, MC, MB, and MA define the program memory maps and the operating mode of the DSP56003/005 (see Table 3-2). On processor reset, MC, MB, and MA are loaded from the external mode select pins, MODC, MODB, and MODA, respectively. After the DSP leaves the reset state, MC, MB, and MA can be changed under software control.

3.2.2 OMR Data ROM Enable (DE) Bit 2

The DE bit enables the two, on-chip, 256 x 24 data ROMs located between addresses \$0100–\$01FF in the X and Y memory spaces (if the YD bit is set, Y data memory accesses are external and do not access the internal data ROM memory). When DE is cleared, the \$0100–\$01FF address space is part of the external X and Y data spaces, and the on-chip data ROMs are disabled. Hardware reset clears the DE bit.

Table 3-1 Memory Mode Bits

DE	YD	Data Memory
0	0	Internal ROMs Disabled and their addresses are part of External Memory
0	1	Internal X Data ROM is Disabled and is part of External Memory. Internal Y Data RAM and ROM are Disabled and are part of External Memory

3.2.3 OMR Internal Y Memory Disable (YD) Bit 3

Bit 3 is defined as Internal Y Memory Disable (YD). When set, all Y Data Memory addresses are considered to be external, disabling access to internal Y Data Memory. When cleared, internal Y Data Memory may be accessed according to the state of the DE control bit. The content of the internal Y Data Memory is not affected by the state of the YD bit. The YD bit is cleared during hardware reset.

Figure 3-1a shows a graphic representation of the DE and YD bit effects on the X and Y data memory maps. Table 3-1 also compares the DE and YD effects on the memory maps.

3.2.4 OMR Chip Operating Mode (MC) Bit 4

The MC bit, together with bits MA and MB, define the program memory map and the operating mode of the chip. See Paragraph 3.2.1 above for more information.

3.2.5 OMR Reserved Bit 5

This bit is reserved for future expansion and will be read as zero during read operations. This bit should be written as zero for future compatibility.

3.2.6 OMR Stop Delay (SD) Bit 6

The SD bit determines the length of the clock stabilization delay that occurs when the processor leaves the stop processing state. If the stop delay bit is zero when the chip leaves the stop state, a 64K clock cycle delay is selected before continuing the stop instruction cycle and exiting the stop mode. This long delay period is long enough to allow the internal clock to begin oscillating and to stabilize. When a stable external clock is used, setting the stop delay bit to one alows a shorter delay of only eight clock cycles for faster start-up of the DSP (see the *DSP56003/005 Data Sheet* for the actual timing values).

Table 3-2 DSP56003/005 Operating Mode Summary

Operating Mode	M C	M B	M A	Description
0	0	0	0	Single-Chip Mode - P: RAM enabled, reset at $0000
1	0	0	1	Bootstrap from EPROM at $C000, exit in Mode 0
2	0	1	0	Normal Expanded Mode - P: RAM enabled, reset at $E000
3	0	1	1	Development Mode - P: RAM disabled, reset at $0000
4	1	0	0	(Reserved)
5	1	0	1	Bootstrap from Host, exit in Mode 0
6	1	1	0	Bootstrap from SCI (external clock), exit in Mode 0
7	1	1	1	Bootstrap from EPROM at $8000, exit in Mode 0

3.2.7 OMR Reserved Bits 7–23

These bits are reserved for future expansion and will be read as zero during read operations. These bits should be written as zero for future compatibility.

3.3 DSP56003/005 OPERATING MODES

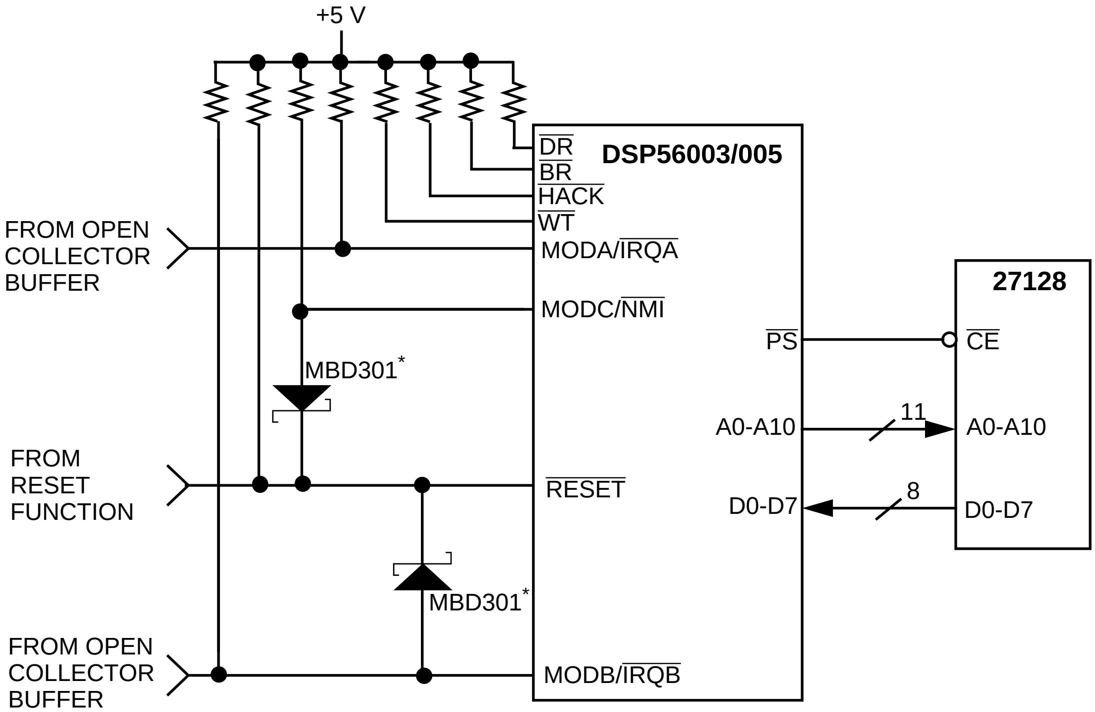
The user can set the chip operating mode through hardware by pulling the appropriate MODC, MODB, and MODA pins high, and then asserting the $\overline{\text{RESET}}$ pin. When the DSP leaves the reset state, it samples the mode pins and writes the results to the OMR to set the initial operating mode.

Chip operating modes can also be changed using software to write the operating mode bits (MC, MB, MA) in the OMR. Changing operating modes does not reset the DSP.

Note: The user should disable interrupts immediately before changing the OMR to prevent an interrupt from going to the wrong memory location. Also, one no-operation (NOP) instruction should be included after changing the OMR to allow for remapping to occur.

3.3.1 Single Chip Mode (Mode 0)

In the single-chip mode, all internal program and data RAM memories are enabled (see Figure 3-1a). A hardware reset causes the DSP to jump to internal program memory location $0000 and resume execution. The memory maps for mode 0 and mode 2 (see Figure 3-1a and Figure 3-1b) are identical. The difference between the two modes is that reset vectors to program memory location $0000 in mode 0 and vectors to location $E000 in mode 2.



Notes: 1. ***These diodes **must** be Schottky diodes.
2. All resistors are 15KΩ unless noted otherwise.
3. When in RESET, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$ and $\overline{\text{NMI}}$ must be deasserted by external peripherals.

ADDRESS OF EXTERNAL BYTE-WIDE P MEMORY	CONTENTS LOADED TO INTERNAL P: RAM AT:
P:$C000	P:$0000 LOW BYTE
P:$C001	P:$0000 MID BYTE
P:$C002	P:$0000 HIGH BYTE
•	•
•	•
•	•
P:$C5FD	P:$01FF LOW BYTE
P:$C5FE	P:$01FF MID BYTE
P:$C5FF	P:$01FF HIGH BYTE

Figure 3-3 Port A Bootstrap Circuit (Mode 1)

3.3.2 Bootstrap From EPROM at $C000 (Mode 1)

The bootstrap mode allows the DSP to load a program from an inexpensive byte-wide ROM into internal program memory during a power-on reset. On power-up, the wait-state generator adds 15 wait states to all external memory accesses (controlled by the BCR) so that slow memory can be used. The bootstrap program uses the bytes in three consecutive memory locations in the external ROM to build a single word in internal program memory (see Table 3-3).

Table 3-3 Organization of EPROM Data Contents

Address of External Byte-Wide Memory:	Contents Loaded to Internal Program RAM at:	
P:$C000	P:$0000	low byte
P:$C001	P:$0000	mid byte
P:$C002	P:$0000	high byte
•	•	
•	•	
•	•	
P:$C5FD	P:$01FF	low byte
P:$C5FE	P:$01FF	mid byte
P:$C5FF	P:$01FF	high byte

In the bootstrap mode, the chip enables the bootstrap ROM and executes the bootstrap program (the bootstrap program code is shown in Appendix A). The bootstrap ROM contains the bootstrap firmware program that performs initial loading of the DSP56003/005 program RAM. Written in DSP56003/005 assembly language, the program initializes the program RAM by loading from an external byte-wide EPROM starting at location P:$C000.

The EPROM is typically connected to the chip's address and data bus.The data contents of the EPROM must be organized as shown in Table 3-3.

After loading the internal memory, the DSP switches to the single-chip mode (Mode 0) and begins program execution at on-chip program memory location $0000.

If the user selects Mode 1 through hardware (MODA, MODB, MODC pins), the following actions occur once the processor comes out of the reset state.

1. The control logic maps the bootstrap ROM into the internal DSP program memory space starting at location $0000.
2. Program execution begins at location $0000 in the bootstrap ROM. The bootstrap ROM program loads program RAM from the external byte-wide EPROM starting at P:$C000.

3. The bootstrap ROM program ends the bootstrap operation and begins executing the user program. The processor enters Mode 0 by writing to the OMR. This action is timed to remove the bootstrap ROM from the program memory map and re-enable read/write access to the program RAM. The change to Mode 0 is timed to allow the bootstrap program to execute a single-cycle instruction (clear status register), then a JMP #<00, and begin execution of the user program at location $0000.

The user can also get into the bootstrap mode (Mode 1) through software by writing zero to MC and MB, and one to MA in the OMR. This selection initiates a timed operation to map the bootstrap ROM into the program address space (after a delay to allow execution of a single-cycle instruction), and then a JMP #<00 to begin the bootstrap process described previously in steps 1 through 4. This technique allows the user to reboot the system (with a different program, if desired).

The code to enter the bootstrap mode is as follows:

```
MOVEP          #0,X:$FFFF   ;Disable interrupts.

MOVEC          #1,OMR       ;The bootstrap ROM is mapped
                            ;into the lowest 96 locations
                            ;in program memory.

NOP                         ;Allow one cycle delay for the
                            ;remapping.

JMP            <$0          ;Begin bootstrap.
```

The code disables interrupts before executing the bootstrap code. Otherwise, an interrupt could cause the DSP to execute the bootstrap code out of sequence because the bootstrap program overlays the interrupt vectors.

3.3.3 Normal Expanded Mode (Mode 2)

In this mode, the internal program RAM is enabled and the hardware reset vectors to location $E000. (The memory maps for Mode 0 and Mode 2 are identical. The difference for Mode 2 is that, after reset, the instruction at location $E000 is executed instead of the instruction at $0000 — see Figure 3-1a and Table 3-2).

3.3.4 Development Mode (Mode 3)

In this mode, the internal program RAM is disabled and the hardware reset vector is set to location $0000. All references to program memory space are directed to external program memory. The reset vector points to location $0000. The memory map for this mode is shown in Figure 3-1a and Table 3-2.

3.3.5 Reserved (Mode 4)

This mode is reserved for future definition. If selected, it defaults to Mode 5.

3.3.6 Bootstrap From Host (Mode 5)

In this mode, the Bootstrap ROM is enabled and the bootstrap program is executed. This is similar to Mode 1 except that the bootstrap program loads internal program RAM from the Host Port.

Note: There is a difference between Modes 1 and 5 in the DSP56003/005 and Mode 1 in the DSP56001. A DSP56001 program that reloads the internal program RAM from the Host Port by setting MB:MA = 01 (assuming an external pull-up resistor on bit 23 of P:$C000) will not work as expected in the DSP56003/005. In the DSP56003/005, the program would trigger a bootstrap from the external EPROM. The solution is to modify the DSP56001 program to set MC:MB:MA = 101.

3.3.7 Bootstrap From SCI (Mode 6)

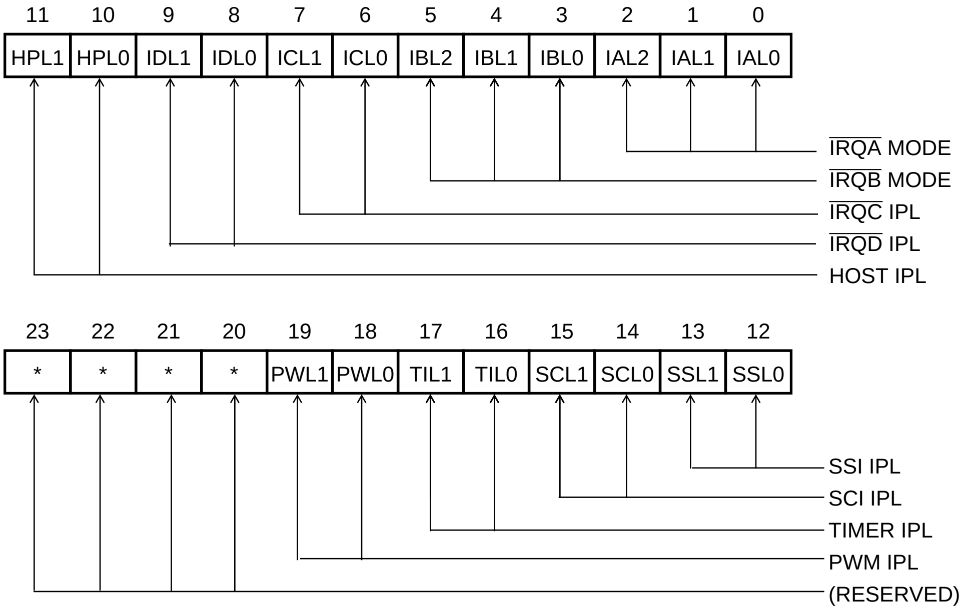
In this mode, the Bootstrap ROM is enabled and the bootstrap program is executed. The internal and/or external program RAM is loaded from the SCI serial interface. The number of program words to load and the starting address must be specified. The SCI bootstrap code expects to receive three bytes specifying the number of program words, three bytes specifying the address from which to start loading the program words, and then three 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 at the address where the first instruction was loaded. 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.

3.3.8 Bootstrap From EPROM at $8000 (Mode 7)

This mode is identical in operation to Mode 1 except that the mode pins and/or bits must be set to MC:MB:MA = 111 and the EPROM is loaded from memory location P: $8000.

3.4 DSP56003/005 INTERRUPT PRIORITY REGISTER

SECTION 7 of the *DSP56000 Family Manual* describes interrupt (exception) processing in detail. It discusses interrupt sources, interrupt types, and interrupt priority levels (IPL). The figures and table in this section updates the information in the family manual to include the specific information for the DSP56003/005.


11 10 9 8 7 6 5 4 3 2 1 0
HPL1 HPL0 IDL1 IDL0 ICL1 ICL0 IBL2 IBL1 IBL0 IAL2 IAL1 IAL0
$\overline{\text{IRQA}}$ MODE
$\overline{\text{IRQB}}$ MODE
$\overline{\text{IRQC}}$ IPL
$\overline{\text{IRQD}}$ IPL
HOST IPL
23 22 21 20 19 18 17 16 15 14 13 12
* * * * PWL1 PWL0 TIL1 TIL0 SCL1 SCL0 SSL1 SSL0
SSI IPL
SCI IPL
TIMER IPL
PWM IPL
(RESERVED)

Note: Bits 20 to 23 are reserved, read as zero and should be written with zero for future compatibility.

Figure 3-4 DSP56003/005 Interrupt Priority Register

Interrupt priority levels for each on-chip peripheral device and for each external interrupt source can be programmed under software control by writing to the interrupt priority register. Level 3 interrupts are nonmaskable, and interrupts of levels 0-2 are maskable.

The DSP56003/005 Interrupt Priority Register (IPR) configuration is shown in Figure 3-4. The starting addresses of interrupt vectors in the DSP56003/005 are defined as shown in Table 3-4, while the relative priorities of exceptions within the same IPL are defined as shown in Table 3-5).

3.5 DSP56003/005 PHASE-LOCKED LOOP (PLL) CONFIGURATION

Section 9 of the *DSP56000 Family Manual* discusses the details of the PLL. The PLL multiplication factor and the clock applied to EXTAL determine the frequency at which the Voltage Controlled Oscillator (VCO) will oscillate i.e. the output frequency of the PLL.

If the PLL is used as the DSP internal clock (PCTL PLL Enable Bit = 1):

- the PLL VCO output is used directly as the internal DSP clock if the PCTL Chip Clock Source Bit (CSRC) is set
- the PLL VCO frequency is divided by the Low Power Divider (LPD) and then used as the internal DSP clock if the CSRC bit is cleared

The DSP56003/005 PLL multiplication factor is set to a logic one during hardware reset, which means that the Multiplication Factor Bits MF0-MF11 in the PLL Control Register (PCTL) are set to $000.The DSP56003/005 LPD division factor bits in the PLL Control Register (PCTL) are cleared during hardware reset.

Table 3-4 Interrupt Vectors

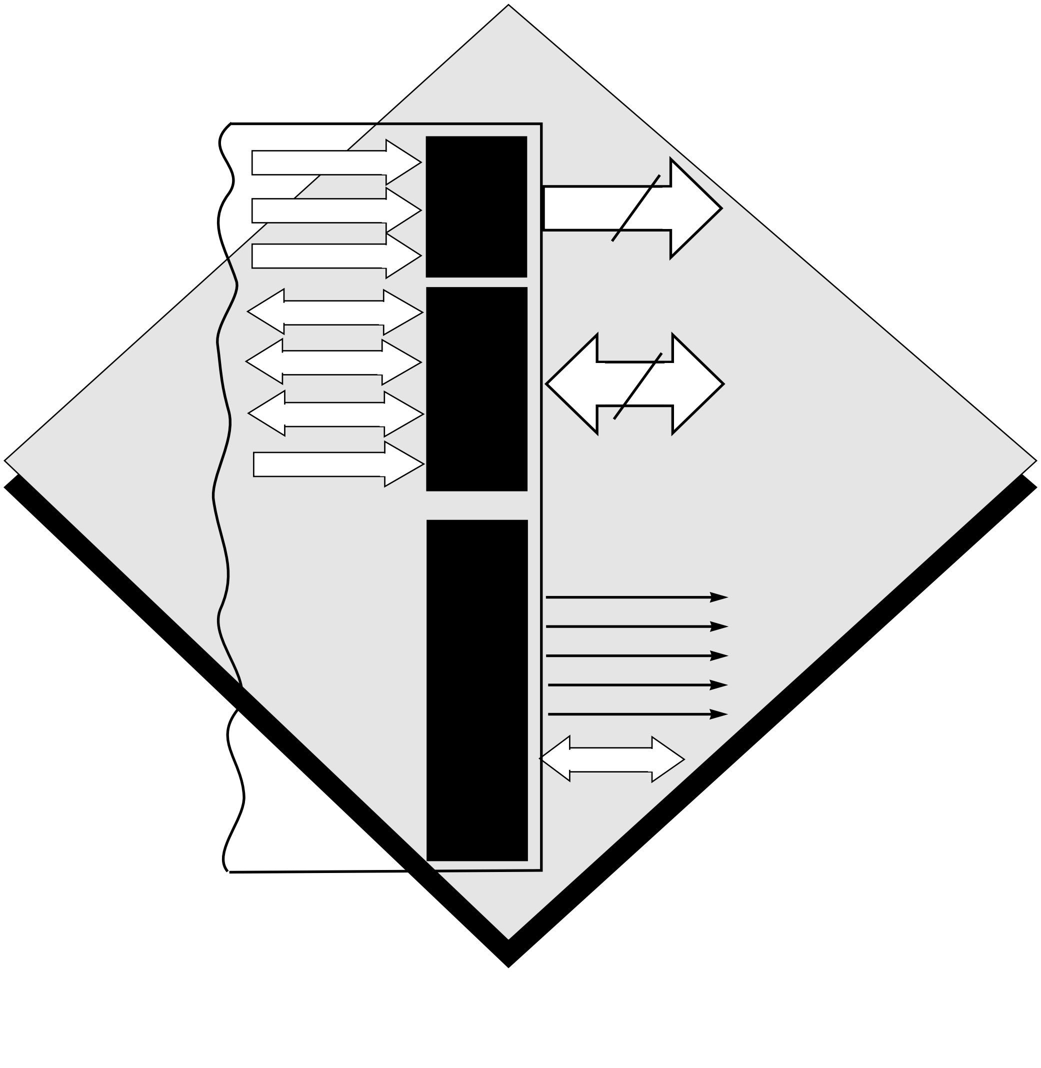
Interrupt Starting Address	IPL	Interrupt Source
P:$0000	3	Hardware $\overline{\text{RESET}}$
P:$0002	3	Stack Error
P:$0004	3	Trace
P:$0006	3	SWI
P:$0008	0 - 2	$\overline{\text{IRQA}}$
P:$000A	0 - 2	$\overline{\text{IRQB}}$
P:$000C	0 - 2	SSI Receive Data
P:$000E	0 - 2	SSI Receive Data With Exception Status
P:$0010	0 - 2	SSI Transmit Data
P:$0012	0 - 2	SSI Transmit Data with Exception Status
P:$0014	0 - 2	SCI Receive Data
P:$0016	0 - 2	SCI Receive Data with Exception Status
P:$0018	0 - 2	SCI Transmit Data
P:$001A	0 - 2	SCI Idle Line
P:$001C	0 - 2	SCI Timer
P:$001E	3	$\overline{\text{NMI}}$
P:$0020	0 - 2	Host Receive Data
P:$0022	0 - 2	Host Transmit Data
P:$0024	0 - 2	Host Command (Default)
P:$0026	0 - 2	Available for Host Command
P:$0028	0 - 2	Available for Host Command
P:$002A	0 - 2	Available for Host Command
P:$002C	0 - 2	$\overline{\text{IRQC}}$
P:$002E	0 - 2	$\overline{\text{IRQD}}$
P:$0030	0 - 2	PWMA0
P:$0032	0 - 2	PWMA1
P:$0034	0 - 2	PWMA2
P:$0036	0 - 2	PWMB0
P:$0038	0 - 2	PWMB1
P:$003A	0 - 2	PWM Error
P:$003C	0 - 2	Timer/Event Counter
P:$003E	3	Illegal Instruction
P:$0040	0 - 2	Available for Host Command
P:$007E	0 - 2	Available for Host Command

Table 3-5 Exception Priorities Within an IPL

Priority	Exception
Level 3 (Nonmaskable)	
Highest	Hardware $\overline{\text{RESET}}$
	Illegal Instruction
	$\overline{\text{NMI}}$ (External Interrupt)
	Stack Error
	Trace
Lowest	SWI
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)
	Host Command Interrupt
	Host Receive Data Interrupt
	Host Transmit Data Interrupt
	SSI RX Data with Exception Interrupt
	SSI RX Data Interrupt
	SSI TX Data with Exception Interrupt
	SSI TX Data Interrupt
	SCI RX Data with Exception Interrupt
	SCI RX Data Interrupt
	SCI TX Data with Exception Interrupt
	SCI TX Data Interrupt
	SCI Idle Line Interrupt
	SCI Timer Interrupt
	Timer/Event Counter Interrupt
	PWM Error
	PWMA0 Ready
	PWMA1 Ready
	PWMA2 Ready
	PWMB0 Ready
Lowest	PWMB1 Ready

SECTION 4

EXTERNAL MEMORY INTERFACE



SECTION CONTENTS

4.1 INTRODUCTION

The External Memory Interface (often refered to as Port A) provides a versatile interface to external memory, allowing economical connection with fast memories/devices, slow memories/devices, and multiple bus master systems.

The external memory interface has two power-reduction features. It can access internal memory spaces, toggling only the external memory signals that need to change, thereby eliminating unneeded switching current. Also, if conditions allow the processor to operate at a lower memory speed, wait states can be added to the external memory access to significantly reduce power while the processor accesses those memories.

4.2 INTERFACE

The DSP56003/005 processor can access one or more of its memory sources (X data memory, Y data memory, and program memory) while it executes an instruction. The memory sources may be either internal or external to the DSP. Three address buses (XAB, YAB, and PAB) and four data buses (XDB, YDB, PDB, and GDB) are available for internal memory accesses during one instruction cycle. The external memory interface's one address bus and one data bus are available for external memory accesses.

If all memory sources are internal to the DSP, one or more of the three memory sources may be accessed in one instruction cycle (i.e., program memory access or program memory access plus an X, Y, XY, or L memory reference). However, when one or more of the memories are external to the chip, memory references may require additional instruction cycles because only one external memory access can occur per instruction cycle.

If an instruction cycle requires more than one external access, the processor will make the accesses in the following priority: X memory, Y memory, and program memory. It takes one instruction cycle for each external memory access – i.e., one access can be executed in one instruction cycle, two accesses take two instruction cycles, etc. Since the external data bus is only 24 bits wide, one XY or long external access will take two instruction cycles. The 16-bit address bus can sustain a rate of one memory access per instruction cycle (using no-wait-state memory which is discussed in Section 4.4 — Wait States).

Figure 4-1 shows the external memory interface signals divided into their three functional groups: address bus signals (A0-A15), data bus signals (D0-D15), and bus control. The bus control signals can be subdivided into three additional groups: read/write control ($\overline{\text{RD}}$ and $\overline{\text{WR}}$), address space selection (including program memory select ($\overline{\text{PS}}$), data memory select ($\overline{\text{DS}}$), external peripheral select ($\overline{\text{EXTP}}$), and X/$\overline{\text{Y}}$ select) and bus access control ($\overline{\text{BN}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, $\overline{\text{WT}}$, $\overline{\text{BS}}$ — DSP56003 only).

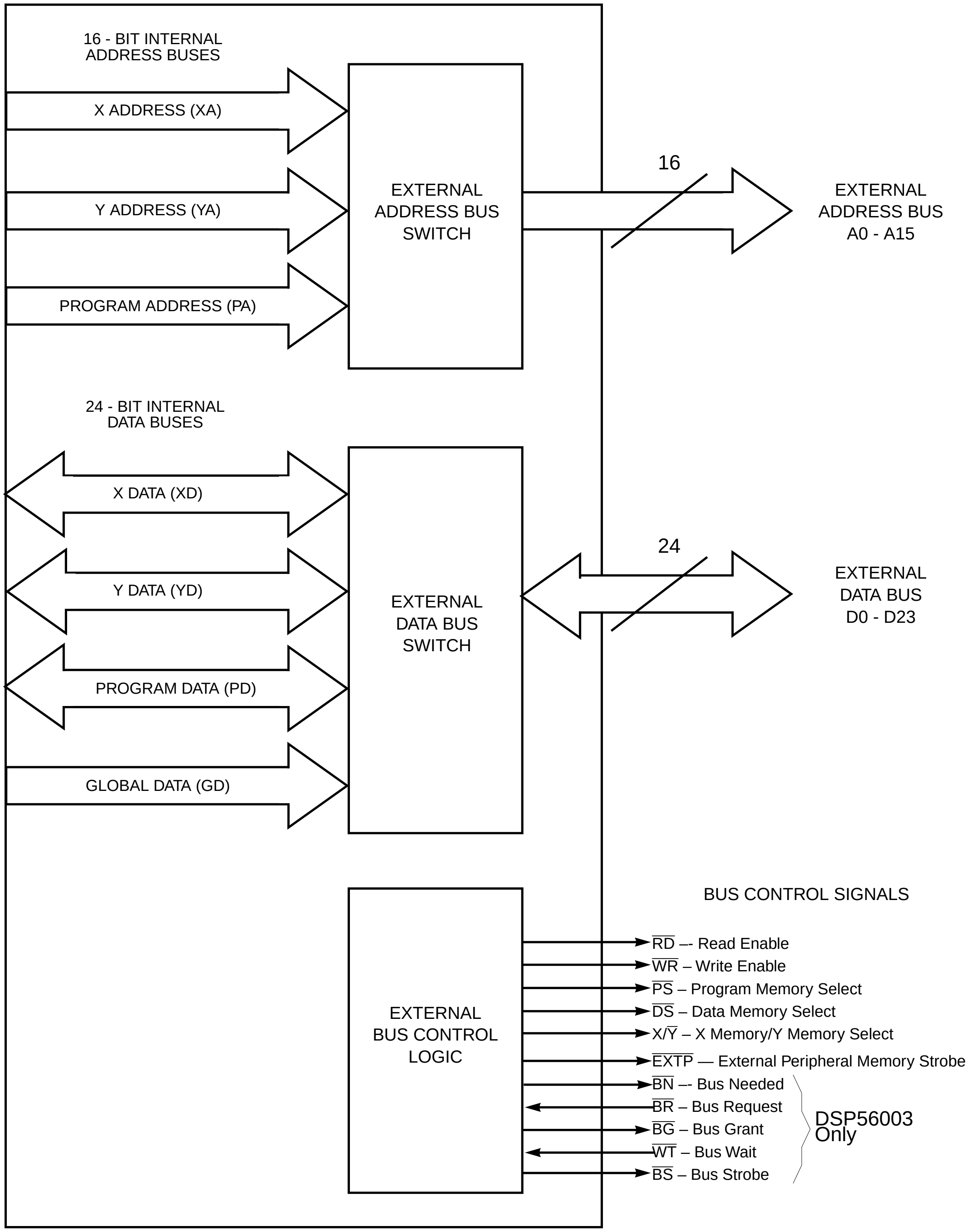


Figure 4-1 External Memory Interface Signals

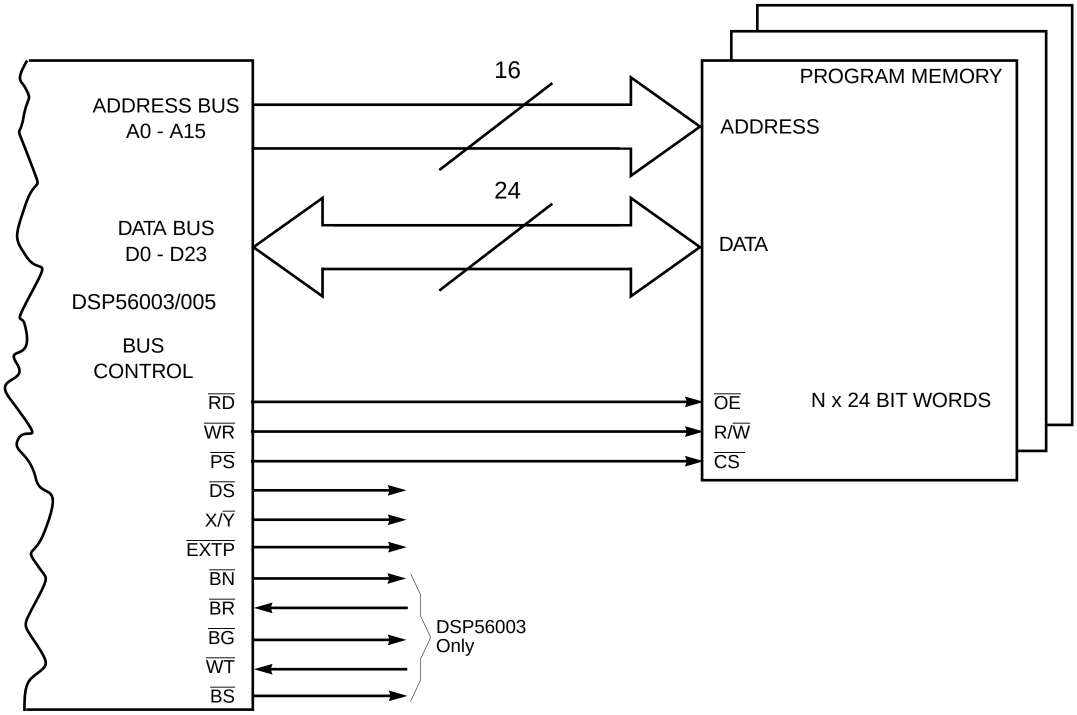


Figure 4-2 External Program Space

The read/write controls can act as decoded read and write controls, or, as seen in Figure 4-2, Figure 4-3, and Figure 4-4, the write signal can be used as the read/write control, and the read signal can be used as an output enable (or data enable) control for the memory. Decoding in such a way simplifies connection to high-speed random-access memories (RAMs). The program memory select, data memory select, and X/Y select can be considered additional address signals, which extend the directly addressable memory from 64K words to 192K words total.

Since external logic delay is large relative to RAM timing margins, timing becomes more difficult as faster DSPs are introduced. The separate read and write strobes used by the DSP56003/005 are mutually exclusive, with a guard time between them to avoid an instance where two data buffers are enabled simultaneously. Other methods using external logic gates to generate the RAM control inputs require either faster RAM chips or external data buffers to avoid data bus buffer conflicts.

Figure 4-2 shows an example of external program memory. A typical implementation of this circuit would use three-byte-wide static memories and would not require any additional logic. The $\overline{\mathrm{PS}}$ signal is used as the program-memory chip-select signal to enable the program memory at the appropriate time.

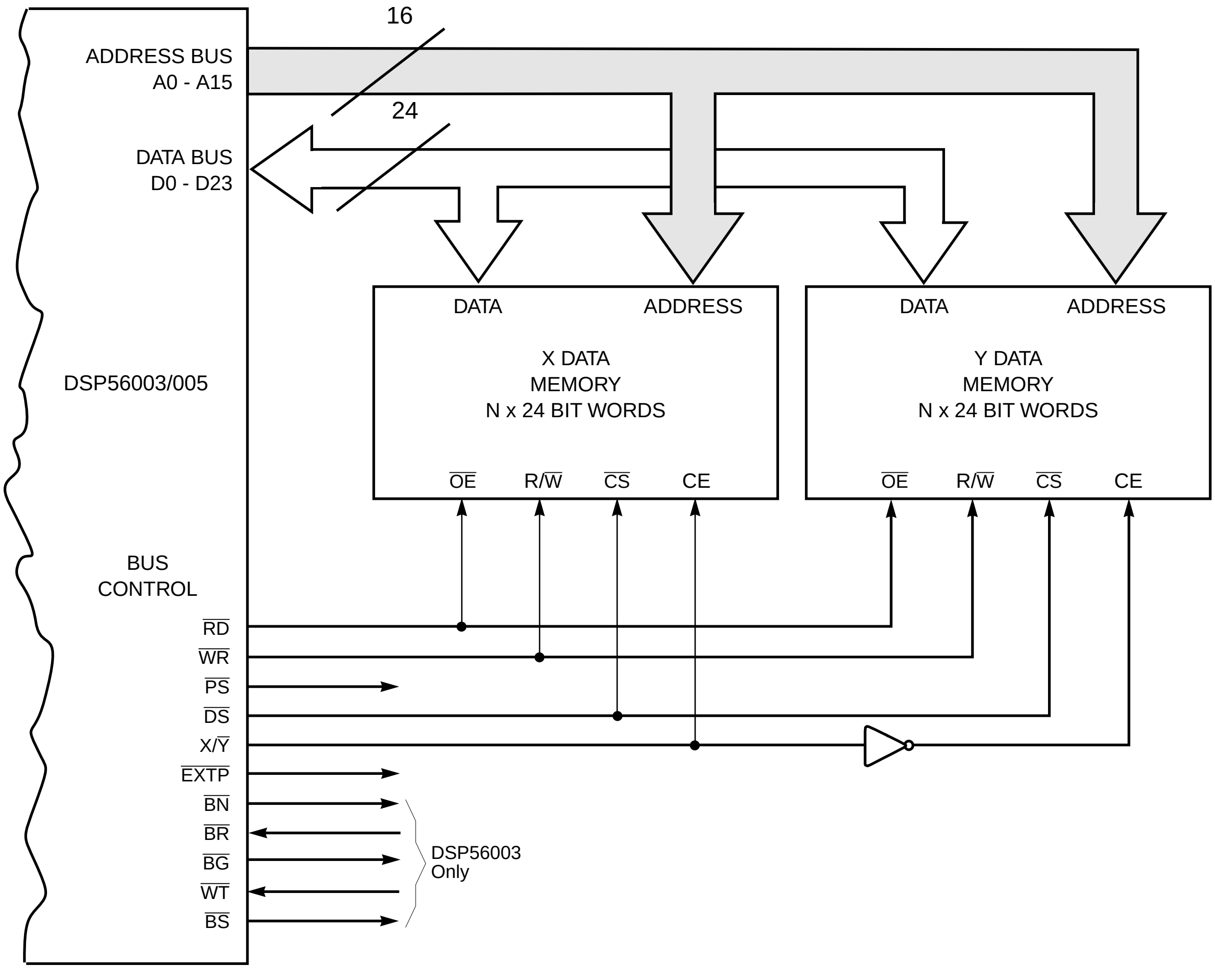


Figure 4-3 External X and Y Data Space

Figure 4-3 shows a similar circuit using the $\overline{\text{DS}}$ signal to enable two data memories and using the X/$\overline{\text{Y}}$ signal to select between them. The three external memory spaces (program, X data, and Y data) do not have to reside in separate physical memories; a single memory can be employed by using the $\overline{\text{PS}}$, $\overline{\text{DS}}$, and X/$\overline{\text{Y}}$ signals as additional address lines to segment the memory into three spaces (see Figure 4-4). Table 4-1 shows how the $\overline{\text{PS}}$, $\overline{\text{DS}}$, and X/$\overline{\text{Y}}$ signals are decoded.

If the DSP is in the development mode, an exception fetch to any interrupt vector location will cause the X/$\overline{\text{Y}}$ signal to go low when $\overline{\text{PS}}$ is asserted. This procedure is useful for debugging and for allowing external circuitry to track interrupt servicing.

Table 4-1 Program and Data Memory Select Encoding

$\overline{\text{PS}}$	$\overline{\text{DS}}$	X/$\overline{\text{Y}}$	External Memory Reference
1	1	1	No Activity
1	0	1	X Data Memory on Data Bus
1	0	0	Y Data Memory on Data Bus
0	1	1	Program Memory on Data Bus (Not an Exception)
0	1	0	External Exception Fetch: Vector or Vector +1 (Development Mode Only)
0	0	X	(Reserved)
1	1	0	(Reserved)

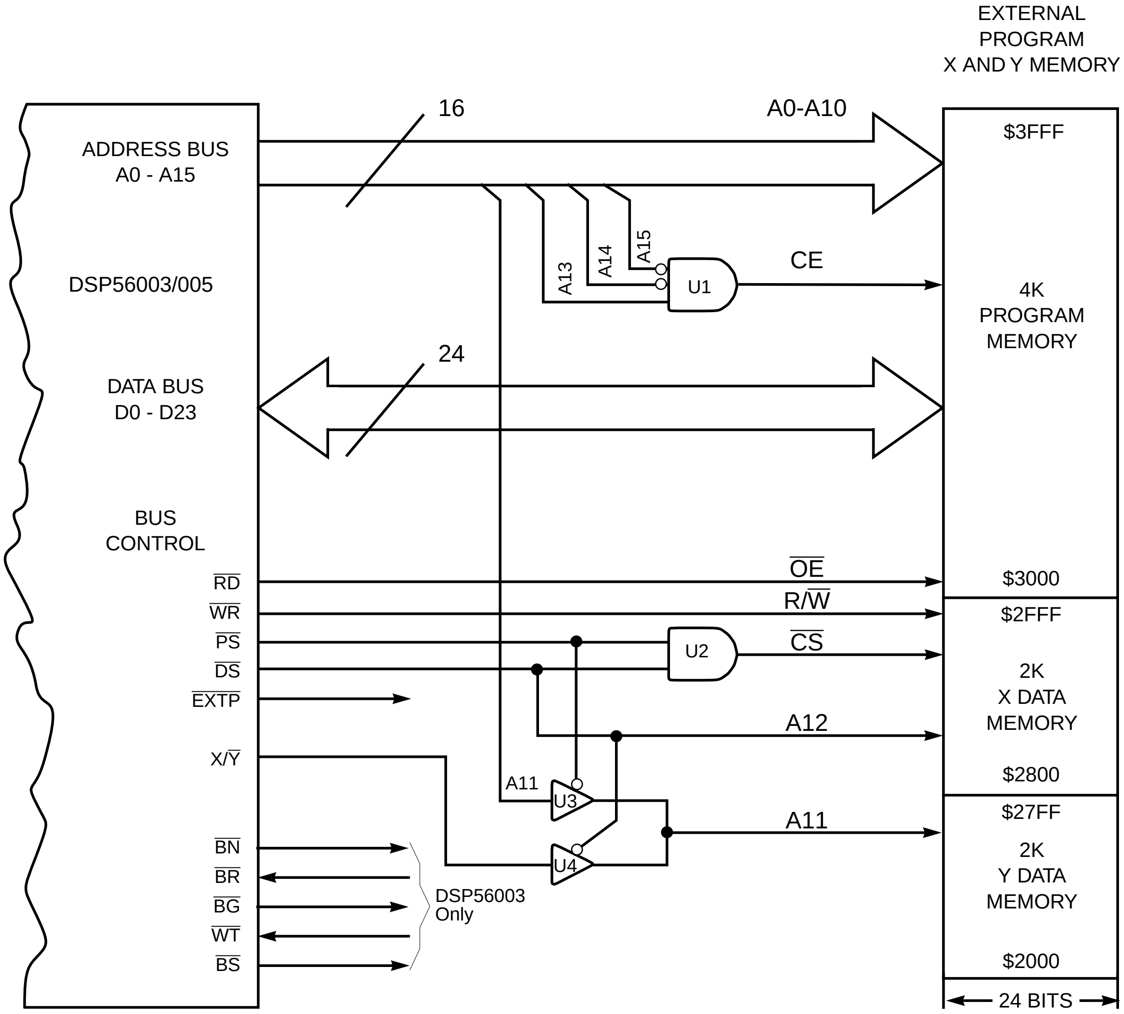
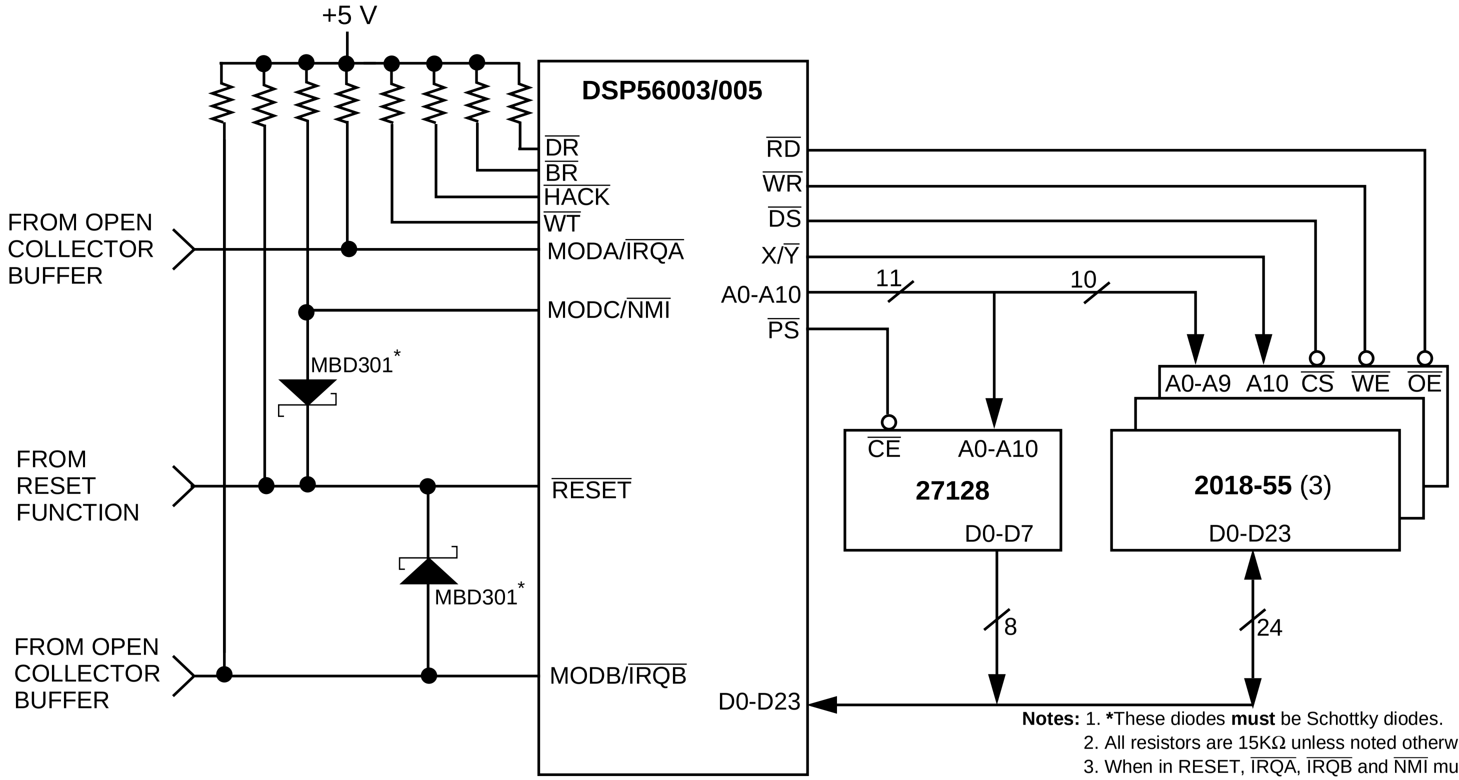


Figure 4-4 Memory Segmentation



Notes: 1. ***These diodes **must** be Schottky diodes.
2. All resistors are 15KΩ unless noted otherwise.
3. When in RESET, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$ and $\overline{\text{NMI}}$ must be deasserted by external peripherals.

Figure 4-5 External Memory Interface Bootstrap ROM with X and Y RAM

Figure 4-5 shows a system that uses internal program memory loaded from an external ROM during power-up and splits the data memory space of a single memory bank into X: and Y: memory spaces. Although external program memory must be 24 bits, external data memory does not. Of course, this is application specific. Many systems use 16 or fewer bits for A/D and D/A conversion and, therefore, they may only need to store 16, 12, or even eight bits of data. The 24/56 bits of internal precision is usually sufficient for intermediate results. This is a cost saving feature which can reduce the number of external memory chips.

4.3 TIMING

The external bus timing is defined by the operation of the address bus, data bus, and bus control pins. Reads or writes by the DSP to the external data bus are synchronous with the DSP clock. The timing A, B, and C relative to the edges of an external clock (see Figure 4-6 and Figure 4-7) are provided in the *DSP56003/005 Data Sheet*. This timing is essential for designing synchronous multiprocessor systems. Figure 4-6 shows the external memory interface timing with no wait states (wait-state control is discussed in Section 4.4). One instruction cycle equals two clock cycles or four clock phases.

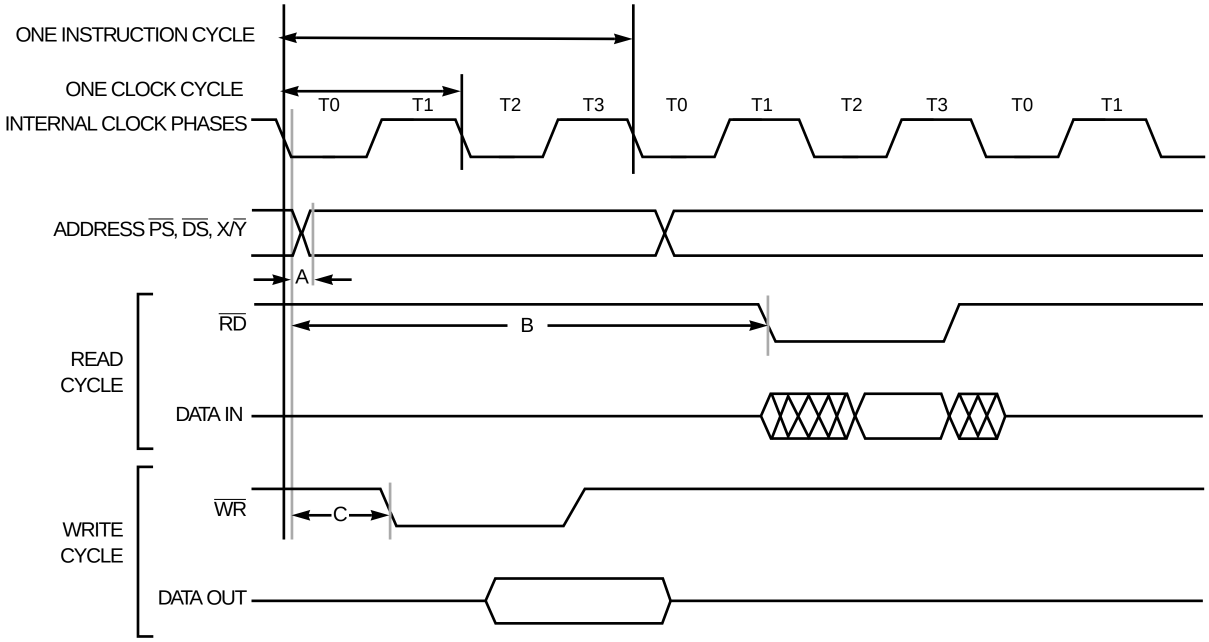


Figure 4-6 External Memory Interface Bus Operation with No Wait States

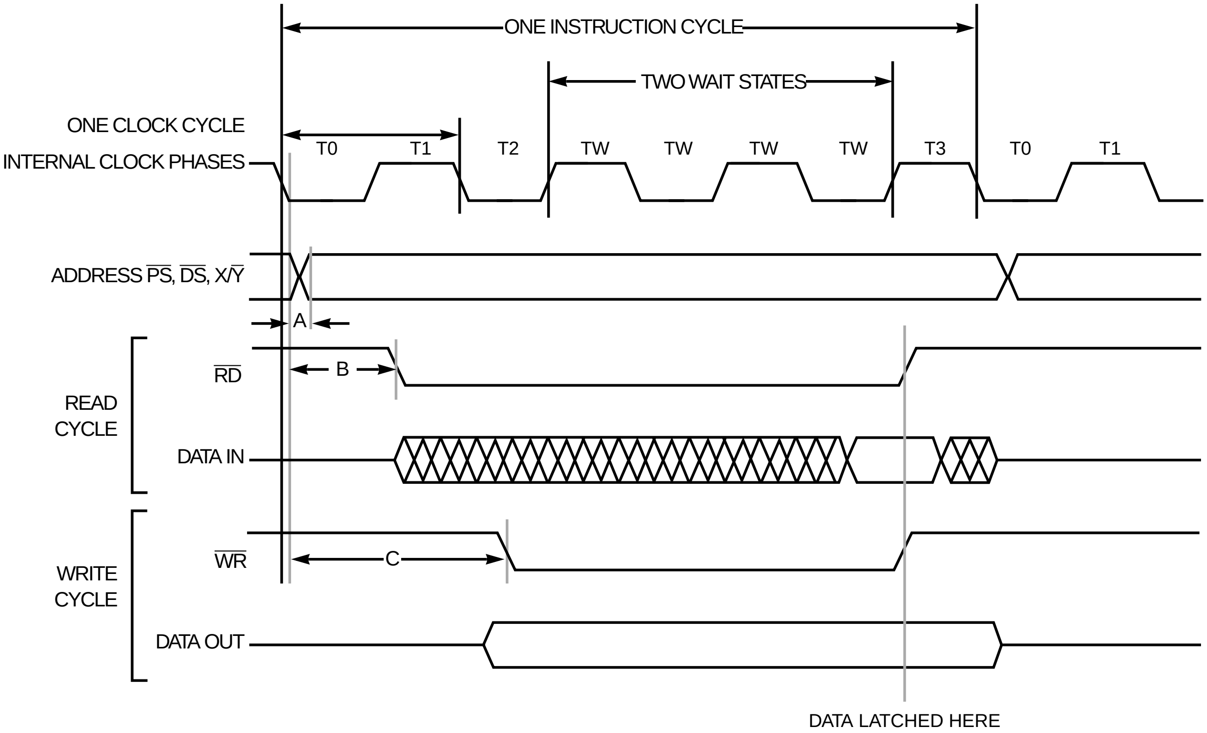


Figure 4-7 External Memory Interface Bus Operation with Two Wait States

The clock phases, which are numbered T0 – T3, are used for timing on the DSP. Figure 4-7 shows the same timing with two wait states added to the external X: memory access. Four TW clock phases have been added because one wait state adds two T phases and is equivalent to repeating the T2 and $\overline{\text{T2}}$ clock phases. The write signal is also delayed from the T1 to the T2 state when one or more wait states are added to ease interfacing to the port. Each external memory access requires the following procedure:

1. The external memory address is defined by the address bus (A0–A15), and the memory reference selects ($\overline{\text{PS}}$, $\overline{\text{DS}}$, and X/$\overline{\text{Y}}$). These signals change in the first phase (T0) of the bus cycle. Since the memory reference select signals have the same timing as the address bus, they may be used as additional address lines. The address and memory reference signals are also used to generate chip-select signals for the appropriate memory chips. These chip-select signals change the memory chips from low-power standby mode to active mode and begin the read access time. This mode change allows slower memories to be used since the chip-select signals can be address based rather than read or write enable based. Read and write enable do not become active until after the address is valid. See the timing diagrams in the *DSP56003/005 Data Sheet* for detailed timing information.

2. When the address and memory reference signals are stable, the data transfer is enabled by read enable ($\overline{\text{RD}}$) or write enable ($\overline{\text{WR}}$). $\overline{\text{RD}}$ or $\overline{\text{WR}}$ is asserted to "qualify" the address and memory reference signals as stable and to perform the read or write data transfer. $\overline{\text{RD}}$ and $\overline{\text{WR}}$ are asserted in the second phase of the bus cycle (if there are no wait states). Read enable is typically connected to the output enable ($\overline{\text{OE}}$) of the memory chips and simply controls the output buffers of the chip-selected memory. Write enable is connected to the write enable ($\overline{\text{WE}}$) or write strobe ($\overline{\text{WS}}$) of the memory chips and is the pulse that strobes data into the selected memory. For a read operation, $\overline{\text{RD}}$ is asserted and $\overline{\text{WR}}$ remains deasserted. Since write enable remains deasserted, a memory read operation is performed. The DSP data bus becomes an input, and the memory data bus becomes an output. For a write operation, $\overline{\text{WR}}$ is asserted and $\overline{\text{RD}}$ remains deasserted. Since read enable remains deasserted, the memory chip outputs remain in the high-impedance state even before write strobe is asserted. This state assures that the DSP and the chip-selected memory chips are not enabled onto the bus at the same time. The DSP data bus becomes an output, and the memory data bus becomes an input.

3. Wait states are inserted into the bus cycle by a wait-state counter or by asserting $\overline{\text{WT}}$. The wait-state counter is loaded from the bus control register. If the value loaded into the wait-state counter is zero, no wait states are inserted into the bus cycle, and $\overline{\text{RD}}$ and $\overline{\text{WR}}$ are asserted as shown in Figure 4-6. If a value $W \neq 0$ is loaded into the wait state counter, W wait states are inserted into the bus cycle. When wait states are inserted into an external write cycle, $\overline{\text{WR}}$ is delayed from T1 to T2. The timing for the case of two wait states (W=2) is shown in Figure 4-7.

4. When $\overline{\text{RD}}$ or $\overline{\text{WR}}$ are deasserted at the start of T3 in a bus cycle, the data is latched in the destination device – i.e., when $\overline{\text{RD}}$ is deasserted, the DSP latches the data internally; when $\overline{\text{WR}}$ is deasserted, the external memory latches the data on the positive-going edge. The address signals remain stable until the first phase of the next external bus cycle to minimize power dissipation. The memory reference signals ($\overline{\text{PS}}$, $\overline{\text{DS}}$, and X/$\overline{\text{Y}}$) are deasserted (held high) during periods of no bus activity, and the data signals are three-stated. For read-modify-write instructions such as BSET, the address and memory reference signals remain active for the complete composite (i.e., two I_{cyc}) instruction cycle.

4.4 WAIT STATES

The DSP56003/005 features two methods to allow the user to accommodate slow memory by changing the external memory interface bus timing. The first method uses the bus control register (BCR), which allows a fixed number of wait states to be inserted in a given memory access to all locations in each of the four memory spaces: X, Y, P, and I/O. The second method uses the bus strobe ($\overline{\text{BS}}$) and bus wait ($\overline{\text{WT}}$) facility (DSP56003 only), which allows an external device to insert an arbitrary number of wait states when accessing either a single location or multiple locations of external memory or I/O space. Wait states are executed until the external device releases the DSP to finish the external memory cycle.

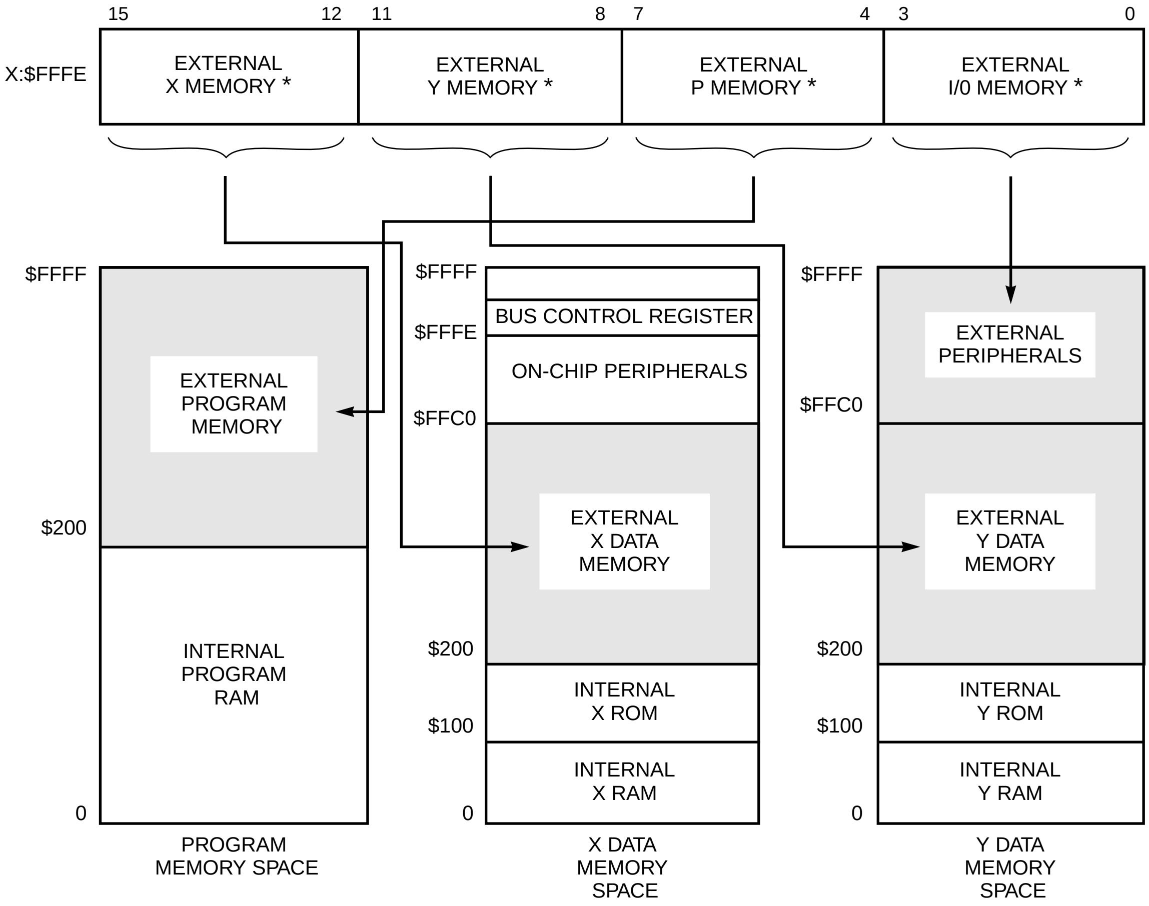
Table 4-2 Wait State Control

BCR Contents	$\overline{\text{WT}}$ (DSP56003 only)	Number of Wait States Generated
0	Deasserted	0
0	Asserted — DSP56003 only	2 (minimum)
> 0	Deasserted	Equals value in BCR
> 0	Asserted — DSP56003 only	Minimum equals 2 or value in BCR. Maximum is determined by BCR or $\overline{\text{WT}}$, whichever is larger.

4.5 BUS CONTROL REGISTER (BCR)

The BCR determines the expansion bus timing by controlling the timing of the bus interface signals, $\overline{\text{RD}}$ and $\overline{\text{WR}}$, and the data output lines. It is a memory mapped register located at X:$FFFE. Each of the memory spaces in Figure 4-8 (X data, Y data, program data, and I/O) has its own 4-bit BCR, which can be programmed for inserting up to 15 wait states (each wait state adds one-half instruction cycle to each memory access – i.e., 20 ns for a 50 Mhz clock). In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces. **On processor reset, the BCR is preset to all ones (15 wait states).** This allows slow memory to be used for boot strapping. **The BCR needs to be set appropriately for the memory being used or the processor will insert 15 wait states between each external memory fetch and cause the DSP to run slowly.**

Figure 4-8 illustrates which of the four BCR nibbles affect which external memory space. All the internal peripheral devices are memory mapped, and their control registers reside between X:$FFC0 and X:$FFFF.



* Zero to 15 wait states can be inserted into each external memory access.

Figure 4-8 Bus Control Register

To load the BCR the way it is shown in Figure 4-9, execute a "MOVEP #$48AD, X:$FFFE" instruction. Or, change the individual bits in one of the four subregisters by using the BSET and BCLR instructions which are detailed in the *DSP56000 Family Manual*, SECTION 6 and APPENDIX A.

Figure 4-9 shows an example of mixing different memory speeds and memory-mapped peripherals in different address spaces. The internal memory uses no wait states, X: memory uses two wait states, Y: memory uses four wait states, P: memory uses five wait states, and the analog converters use 14 wait states. Controlling five different devices at five different speeds requires only one additional logic package. Half the gates in that package are used to map the analog converters to the top 64 memory locations in Y: memory.

Adding wait states to external memory accesses can substantially reduce power requirements. Consult the *DSP56003/005 Data Sheet* for specific power consumption requirements.

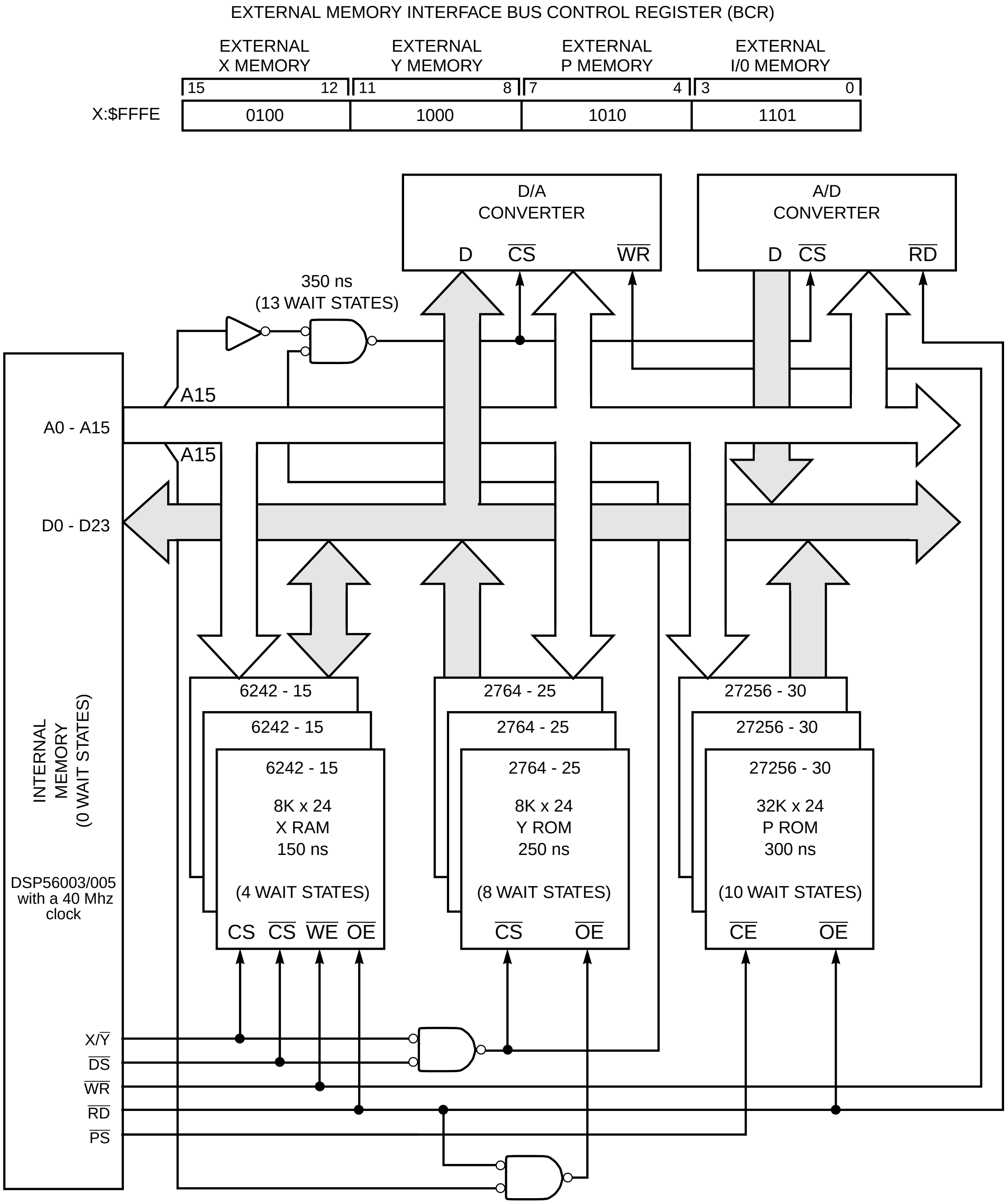


Figure 4-9 Mixed-Speed Expanded System

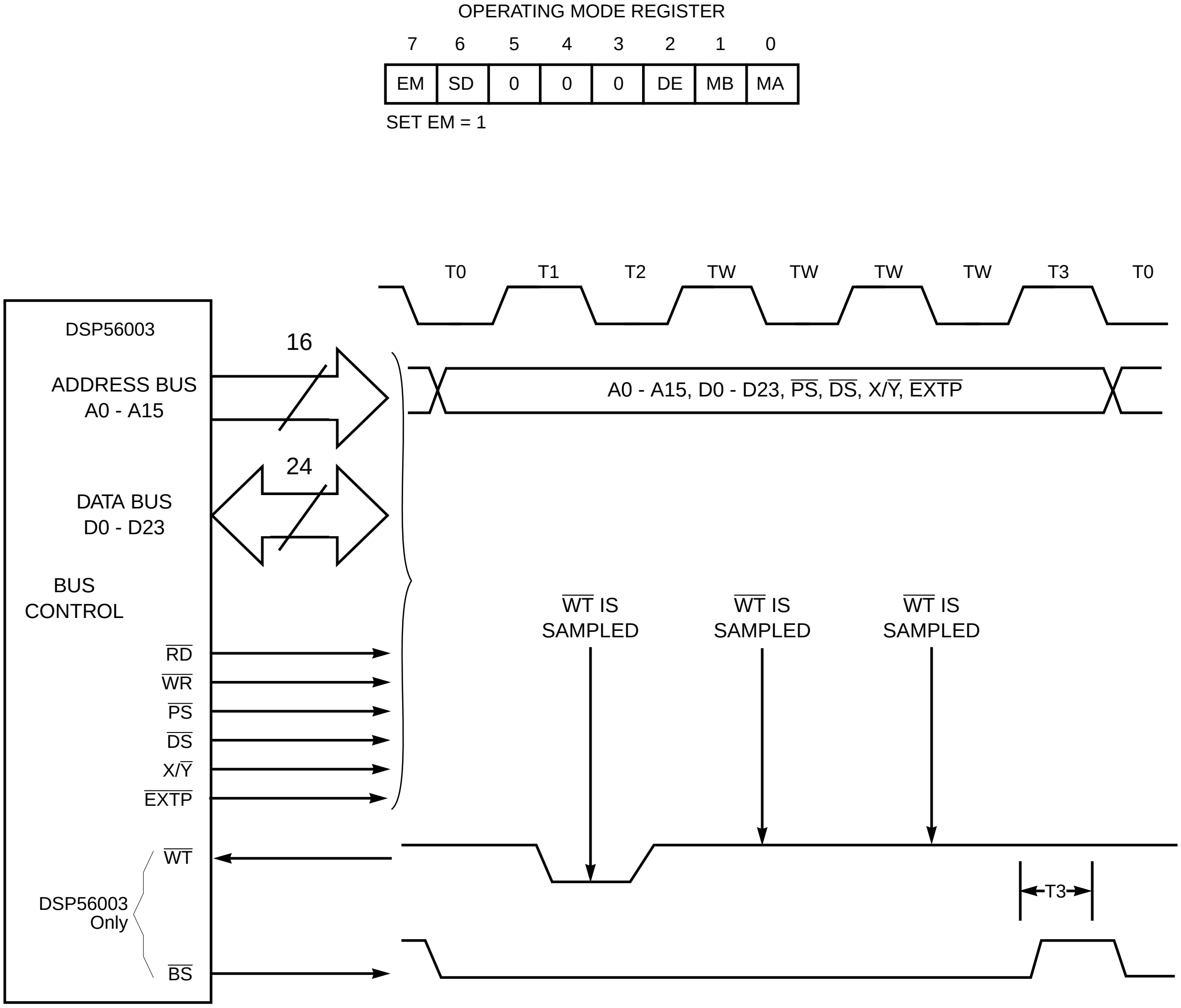


Figure 4-10 Bus Strobe/Wait Sequence — DSP56003 Only

4.6 BUS STROBE AND WAIT PINS — DSP56003 Only

The ability to insert wait states using $\overline{BS}$ and $\overline{WT}$ allows devices with differing timing requirements to reside in the same memory space, allows a bus arbiter to provide a fast multiprocessor bus access, and provides another means of halting the DSP at a known program location with a fast restart.

The timing of the $\overline{BS}$ and $\overline{WT}$ pins is illustrated in Figure 4-10. $\overline{BS}$ is asserted at the same time as the external address lines. $\overline{BS}$ can be used by external wait-state logic to establish the start of an external access. $\overline{BS}$ is deasserted in T3 of each external bus cycle, signaling that the current bus cycle will complete. Since the $\overline{WT}$ signal is internally synchronized, it can be asserted asynchronously with respect to the system clock.

The $\overline{\text{WT}}$ signal should only be asserted while $\overline{\text{BS}}$ is asserted. Asserting $\overline{\text{WT}}$ while $\overline{\text{BS}}$ is deasserted will give indeterminate results. However, for the number of inserted wait states to be deterministic, $\overline{\text{WT}}$ timing must satisfy setup and hold timing with respect to the negative-going edge of EXTAL. The setup and hold times are provided in the *DSP56003/005 Data Sheet*. The timing of $\overline{\text{WR}}$ is controlled by the BCR and is independent of $\overline{\text{WT}}$. The minimum number of wait states that can be inserted using the $\overline{\text{WT}}$ pin is two. The BCR is still operative when using $\overline{\text{BS}}$ and $\overline{\text{WT}}$ and defines the minimum number of wait states that are inserted. Table 4-2 summarizes the effect of the BCR and $\overline{\text{WT}}$ pin on the number of wait states generated.

4.7 BUS ARBITRATION AND SHARED MEMORY — DSP56003 Only

The DSP56003 has five pins that control the external memory interface. They are bus needed ($\overline{\text{BN}}$), bus request ($\overline{\text{BR}}$), bus grant ($\overline{\text{BG}}$), bus strobe ($\overline{\text{BS}}$) and bus wait ($\overline{\text{WT}}$) and they are described in Section 2 — DSP56003/005 Pin Descriptions.

The bus control signals provide the means to connect additional bus masters (which may be additional DSPs, microprocessors, direct memory access (DMA) controllers, etc.) to the external memory interface bus. They work together to arbitrate and determine what device gets access to the bus.

If an external device has requested the external bus by asserting the $\overline{\text{BR}}$ input, and the DSP has granted the bus by asserting $\overline{\text{BG}}$, the DSP will continue to process as long as it requires no external bus accesses itself. If the DSP **does** require an external access but is not the bus master, it will stop processing and remain in wait states until it regains bus ownership. The $\overline{\text{BN}}$ pin will be asserted, and an external device may use $\overline{\text{BN}}$ to help "arbitrate", or decide when to return bus ownership to the chip.

- Four examples of bus arbitration will be described later in this section:
- bus arbitration using only $\overline{\text{BR}}$ and $\overline{\text{BG}}$ with internal control
- bus arbitration using $\overline{\text{BN}}$, $\overline{\text{BR}}$, and $\overline{\text{BG}}$ with external control
- bus arbitration using $\overline{\text{BR}}$, $\overline{\text{BG}}$ and $\overline{\text{WT}}$, $\overline{\text{BS}}$ with no overhead
- signaling using semaphores.

The $\overline{\text{BR}}$ input allows an external device to request and be given control of the external bus while the DSP continues internal operations using internal memory spaces. This independent operation allows a bus controller to arbitrate a multiple bus-master system independent of operation of each DSP. (A bus master can issue addresses on the bus; a bus slave can respond to addresses on the bus. A single device can be both a master and a slave, but can only be one or the other at any given time.)

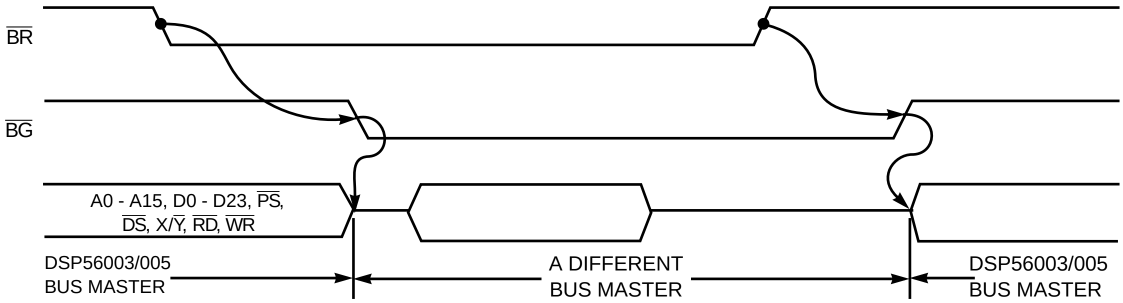

$\overline{\mathrm{BR}}$
$\overline{\mathrm{BG}}$
A0 - A15, D0 - D23, $\overline{\mathrm{PS}}$, $\overline{\mathrm{DS}}$, X/$\overline{\mathrm{Y}}$, $\overline{\mathrm{RD}}$, $\overline{\mathrm{WR}}$
DSP56003/005 BUS MASTER
A DIFFERENT BUS MASTER
DSP56003/005 BUS MASTER

Figure 4-11 Bus Request/Bus Grant Sequence — DSP56003 Only

Before $\overline{\mathrm{BR}}$ is asserted, all the external memory interface signals may be driven by the DSP. When $\overline{\mathrm{BR}}$ is asserted (see Figure 4-11), the DSP will assert $\overline{\mathrm{BG}}$ after the current external access cycle completes and will simultaneously three-state (high-impedance) the external memory interface signals (see the *DSP56003/005 Data Sheet* for exact timing of $\overline{\mathrm{BR}}$ and $\overline{\mathrm{BG}}$). The bus is then available to whatever external device has bus mastership. The external device will return bus mastership to the DSP by deasserting $\overline{\mathrm{BR}}$. After the DSP completes the current cycle (an internally executed instruction with or without wait states), $\overline{\mathrm{BG}}$ will be deasserted. When $\overline{\mathrm{BG}}$ is deasserted, the A0-A15, $\overline{\mathrm{PS}}$, $\overline{\mathrm{DS}}$, X/$\overline{\mathrm{Y}}$, $\overline{\mathrm{EXTP}}$, and $\overline{\mathrm{RD}}$, $\overline{\mathrm{WR}}$ lines will be driven. However, the data lines will remain in three-state. All signals are now ready for a normal external access.

During the wait state (see SECTION 7 in the *DSP56000 Family Manual*), the $\overline{\mathrm{BR}}$ and $\overline{\mathrm{BG}}$ circuits remain active. However, the port is inactive - the control signals are deasserted, the data signals are inputs, and the address signals remain as the last address read or written. When $\overline{\mathrm{BR}}$ is asserted, all signals are three-stated (high impedance). Table 4-3 shows the status of $\overline{\mathrm{BR}}$ and $\overline{\mathrm{BG}}$ during the wait state.

Table 4-3 BR and BG During Wait — DSP56003 Only

Signal	Before $\overline{\mathrm{BR}}$	While $\overline{\mathrm{BG}}$	After $\overline{\mathrm{BR}}$	After Return to Normal State	After First

4.7.1 Bus Arbitration Using Only $\overline{\text{BR}}$ and $\overline{\text{BG}}$ With Internal Control — DSP56003 Only

Perhaps the simplest example of a shared memory system using a DSP56003 is shown in Figure 4-12. The bus arbitration is performed within the DSP#2 by using software. DSP#2 controls all bus operations by using I/O pin OUT2 to three-state its own external memory interface and by never accessing the external memory interface without first calling the subroutine that arbitrates the bus. When the DSP#2 needs to use external memory, it uses I/O pin OUT1 to request bus access and I/O pin IN1 to read bus grant. DSP#1 does not need any extra code for bus arbitration since the $\overline{\text{BR}}$ and $\overline{\text{BG}}$ hardware handles its bus arbitration automatically. The protocol for bus arbitration is as follows:

At reset: DSP#2 sets OUT2=0 ($\overline{\text{BR}}$#2=0) and OUT1=1 ($\overline{\text{BR}}$#1=1), which gives DSP#1 access to the bus and suspends DSP#2 bus access.

When DSP#2 wants control of the memory, the following steps are performed (see Figure 4-13):

1. DSP# 2 sets OUT1=0 ($\overline{\text{BR}}$#1=0).
2. DSP# 2 waits for IN1=0 ($\overline{\text{BG}}$#1=0 and DSP#1 off the bus).
3. DSP#2 sets OUT2=1 ($\overline{\text{BR}}$#2=1 to let DSP#2 control the bus).
4. DSP#2 accesses the bus for block transfers, etc. at full speed.
5. To release the bus, DSP#2 sets OUT2=0 ($\overline{\text{BR}}$#2=0) after the last external access.
6. DSP#2 then sets OUT1=1 ($\overline{\text{BR}}$#1=1) to return control of the bus to DSP#1.
7. DSP#1 then acknowledges mastership by deasserting $\overline{\text{BG}}$#1.

4.7.2 Bus Arbitration Using $\overline{\text{BN}}$, $\overline{\text{BR}}$, and $\overline{\text{BG}}$ With External Control — DSP56003 Only

The system shown in Figure 4-14 can be implemented with external bus arbitration logic, which will save processing capacity on the DSPs and can make bus access much faster at a cost of additional hardware. The bus arbitration logic takes control of the external bus by deasserting an enable signal (E1, E2, and E3) to all DSPs, which will then acknowledge by granting the bus ($\overline{\text{BG}}$=0). When a DSP (DSP#1 in Figure 4-14) needs the bus, it will enter the wait state with $\overline{\text{BN}}$ asserted. If DSP#1 has highest priority of the pending bus requests, the arbitration logic grants the bus to DSP#1 by asserting E1 (E2 for DSP#2; E3 for DSP#3) to let the DSP know that it can have the bus. DSP#1 will then deassert $\overline{\text{BG}}$ to tell the arbiter it has taken control of the bus. When the DSP no longer needs to make an external access it will deassert $\overline{\text{BN}}$ and the arbiter deasserts E1, after which the DSP deasserts $\overline{\text{BG}}$.

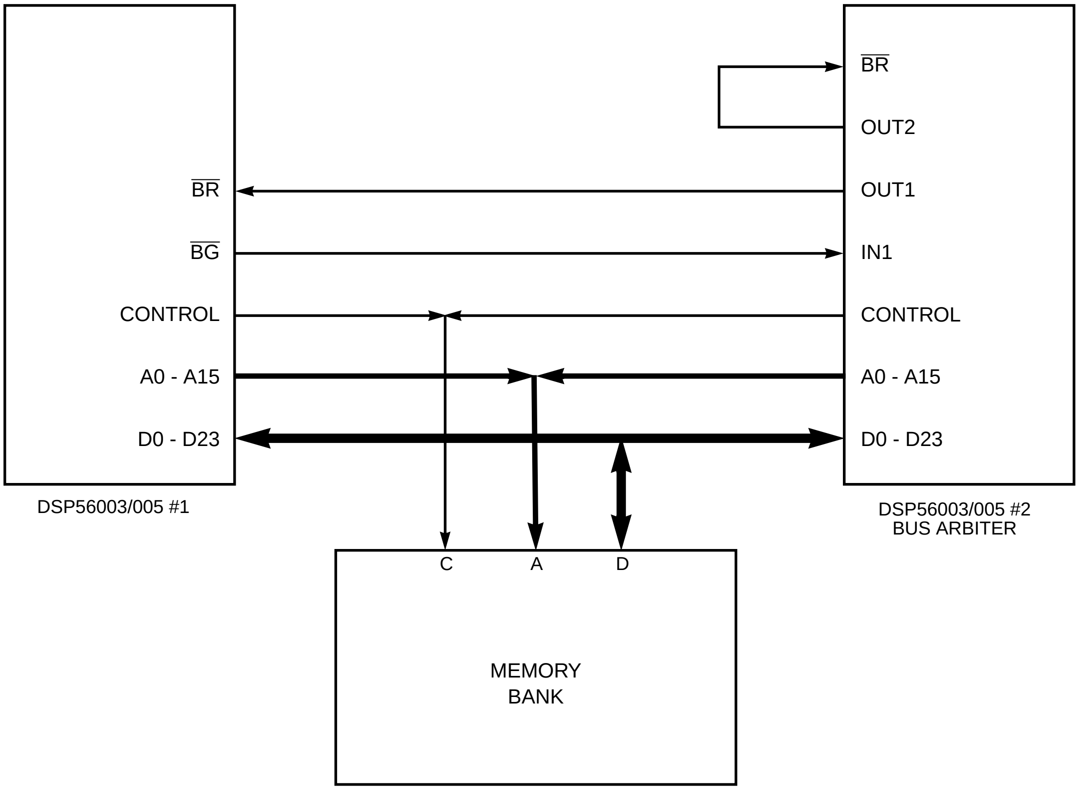


Figure 4-12 Bus Arbitration Using Only $\overline{\text{BR}}$ and $\overline{\text{BG}}$ with Internal Control — DSP56003 Only

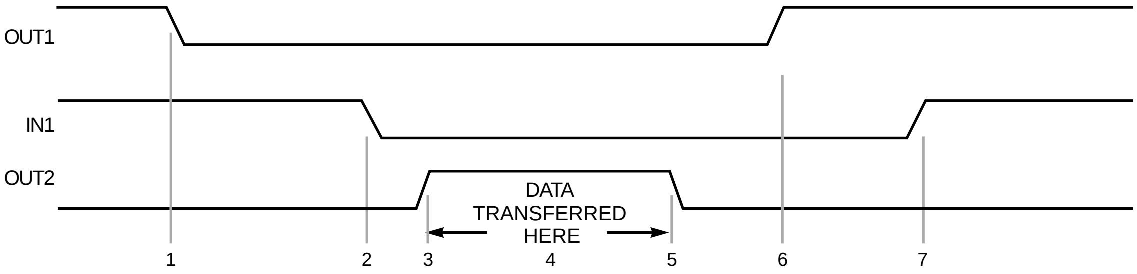


Figure 4-13 Two DSPs with External Bus Arbitration Timing

4.7.3 Arbitration Using $\overline{\text{BR}}$ and $\overline{\text{BG}}$, and $\overline{\text{WT}}$ and $\overline{\text{BS}}$ With No Overhead —

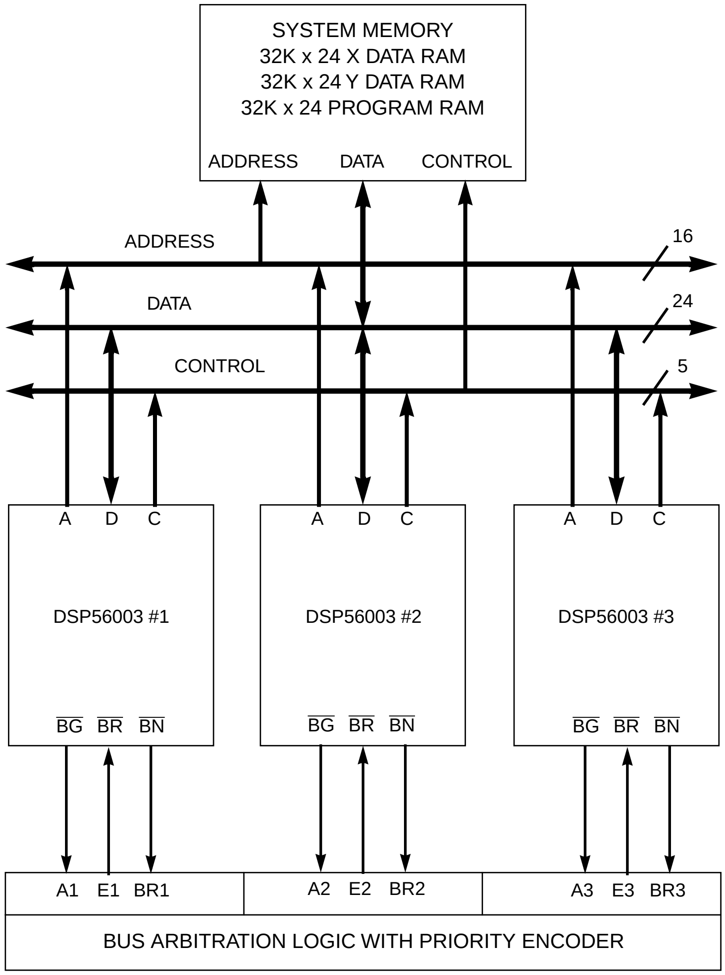


Figure 4-14 Bus Arbitration Using $\overline{\text{BN}}$, $\overline{\text{BR}}$, and $\overline{\text{BG}}$ with External Control — DSP56003 Only

DSP56003 Only

By using the circuit shown in Figure 4-15, two DSPs can share memory with hardware arbitration that requires no software on the part of the DSPs. The protocol for bus arbitration in Figure 4-15 is as follows:

At RESET assume DSP#1 is not making external accesses so that $\overline{\text{BR}}$ of DSP#2 is deasserted. Hence, $\overline{\text{BG}}$ of DSP#2 is deasserted, which three-states the buffers, giving DSP#2 control of the memory.

When DSP#1 wants control of the memory the following steps are performed (see Figure 4-16):

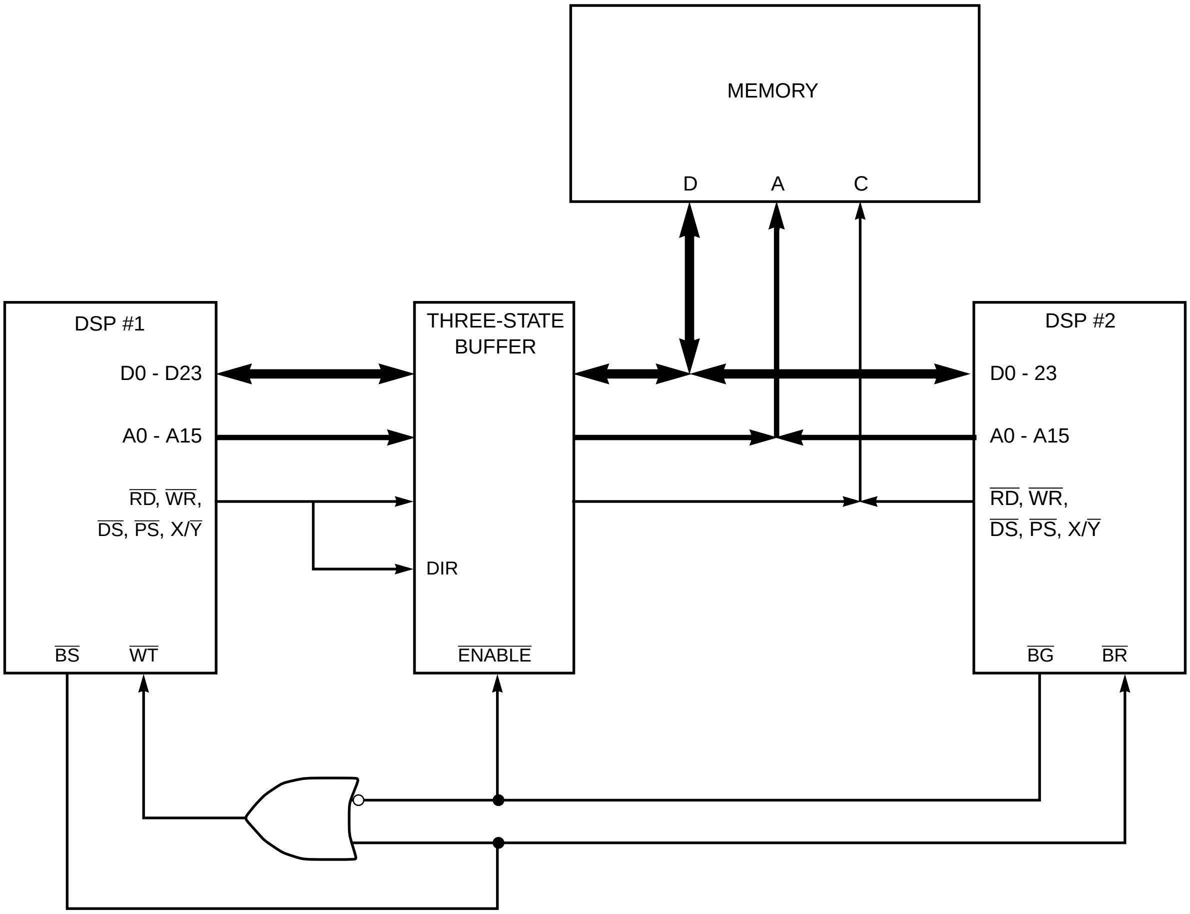

MEMORY
D
A
C
DSP #1
D0 - D23
A0 - A15
$\overline{RD}$, $\overline{WR}$,
$\overline{DS}$, $\overline{PS}$, X/$\overline{Y}$
$\overline{BS}$
$\overline{WT}$
THREE-STATE BUFFER
DIR
$\overline{ENABLE}$
DSP #2
D0 - 23
A0 - A15
$\overline{RD}$, $\overline{WR}$,
$\overline{DS}$, $\overline{PS}$, X/$\overline{Y}$
$\overline{BG}$
$\overline{BR}$

Figure 4-15 Bus Arbitration Using $\overline{\text{BR}}$ and $\overline{\text{BG}}$, and $\overline{\text{WT}}$ and $\overline{\text{BS}}$ with No Overhead — DSP56003 Only

1. DSP#1 makes an external access, thereby asserting $\overline{\text{BS}}$, which asserts $\overline{\text{WT}}$ (causing DSP#1 to execute wait states in the current cycle) and asserts DSP#2 $\overline{\text{BR}}$ (requesting that DSP#2 release the bus).

2. When DSP#2 finishes its present bus cycle, it three-states its bus drivers and asserts $\overline{\text{BG}}$. Asserting $\overline{\text{BG}}$ enables the three-state buffers, placing the DSP#1 signals on the memory bus. Asserting $\overline{\text{BG}}$ also deasserts $\overline{\text{WT}}$, which allows DSP#1 to finish its bus cycle.

3. When DSP#1's memory cycle is complete, it releases $\overline{\text{BS}}$, which deasserts $\overline{\text{BR}}$. DSP#2 then deasserts $\overline{\text{BG}}$, three-stating the buffers and allowing DSP#2 to access the memory bus.

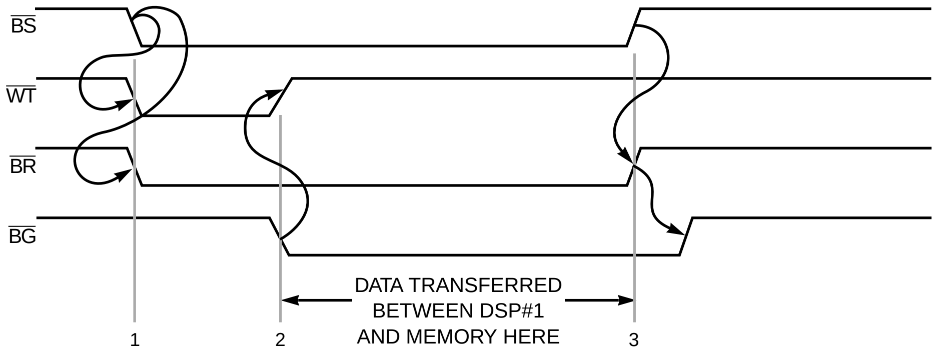


Figure 4-16 Two DSPs with External Bus Arbitration Timing — DSP56003 Only

4.7.4 Signaling Using Semaphores

Figure 4-17 shows a more sophisticated shared memory system that uses external arbitration with both local external memory and shared memory. The four semaphores are bits in one of the words in each shared memory bank used by software to arbitrate memory use. Semaphores are commonly used to indicate that the contents of the semaphore's memory blocks are being used by one processor and are not available for use by another processor. Typically, if the semaphore is cleared, the block is not allocated to a processor; if the semaphore is set, the block is allocated to a processor.

Without semaphores, one processor may try to use data while it is being changed by another processor, which may cause errors. This problem can occur in a shared memory system when separate test and set instructions are used to "lock" a data block for use by a single processor.

The **correct procedure** is to test the semaphore and then set the semaphore if it was clear to lock and gain exclusive use of the data block. The problem occurs when the second processor acquires the bus and tests the semaphore after the first processor tests the semaphore but before the first processor can lock the data block.

The **incorrect sequence** is:

1. the first processor tests the semaphore and sees that the block is available
2. the second processor then tests the bit and also sees that the block is available
3. both processors then set the bit to lock the data
4. both proceed to use the data on the assumption that the data cannot be changed by another processor

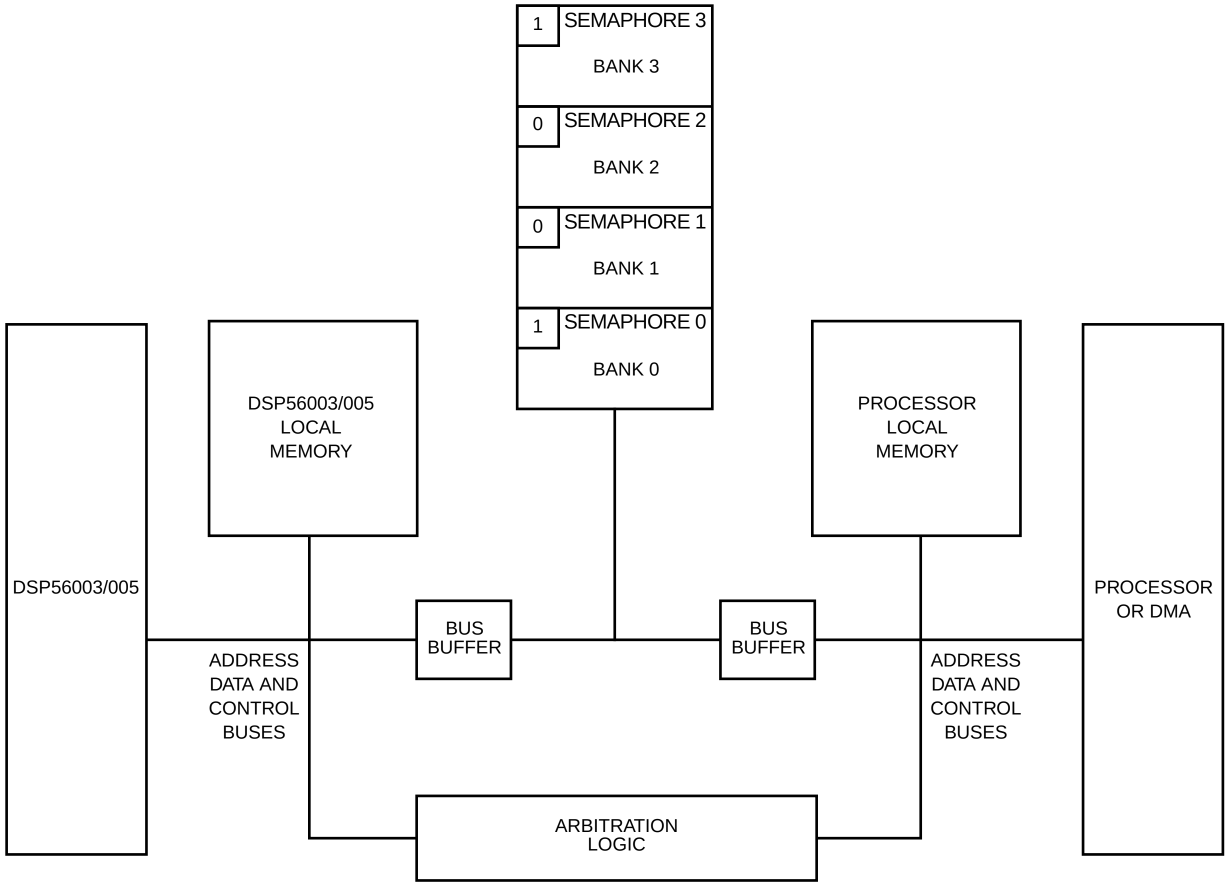
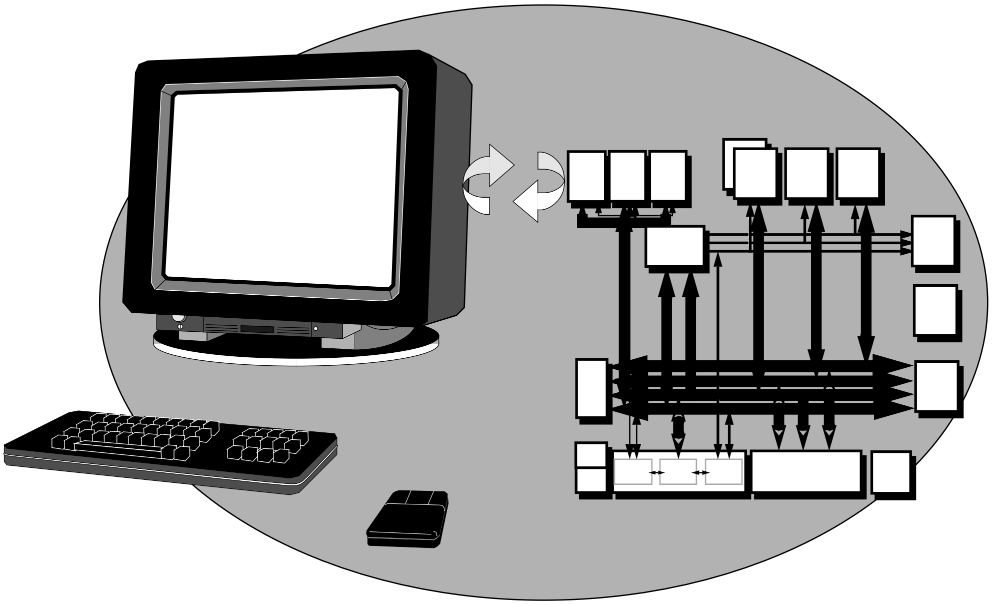


Figure 4-17 Signaling Using Semaphores

The solution is that the DSP56K processor series has a group of instructions designed specifically to prevent this problem. They perform an indivisible read-modify-write operation and do not release the bus between the read and write (specifically, A0–A15, $\overline{\text{DS}}$, $\overline{\text{PS}}$, and X/$\overline{\text{Y}}$ do not change state). **Using a read-modify-write operation allows these instructions to test the semaphore and then to set, clear, or change the semaphore without the possibility of another processor testing the semaphore before it is changed.** The instructions are bit test and change (BCHG), bit test and clear (BCLR), and bit test and set (BSET). (They are discussed in detail in the *DSP56000 Family Manual.*) The proper way to set the semaphore to gain exclusive access to a memory block is to use BSET to test the semaphore and to set it to one. After the bit is set, the result of the test operation will reveal if the semaphore was clear before it was set by BSET and if the memory block is available. If the bit was already set and the block is in use by another processor, the DSP must wait to access the memory block.

SECTION 5

HOST INTERFACE



SECTION CONTENTS

5.1 INTRODUCTION

Port B is a dual-purpose I/O port. It performs as 15 general-purpose I/O (GPIO) pins, each configurable as output or input, to be used for device control. Or, it can perform as an 8-bit bidirectional host interface (HI) (see Figure 5-1), where it provides a convenient connection to another processor. This section describes both configurations, including examples of how to configure and use the port. This Port B (GPIO and Host Interface) is identical to the Port B on the DSP56001 and DSP56002.

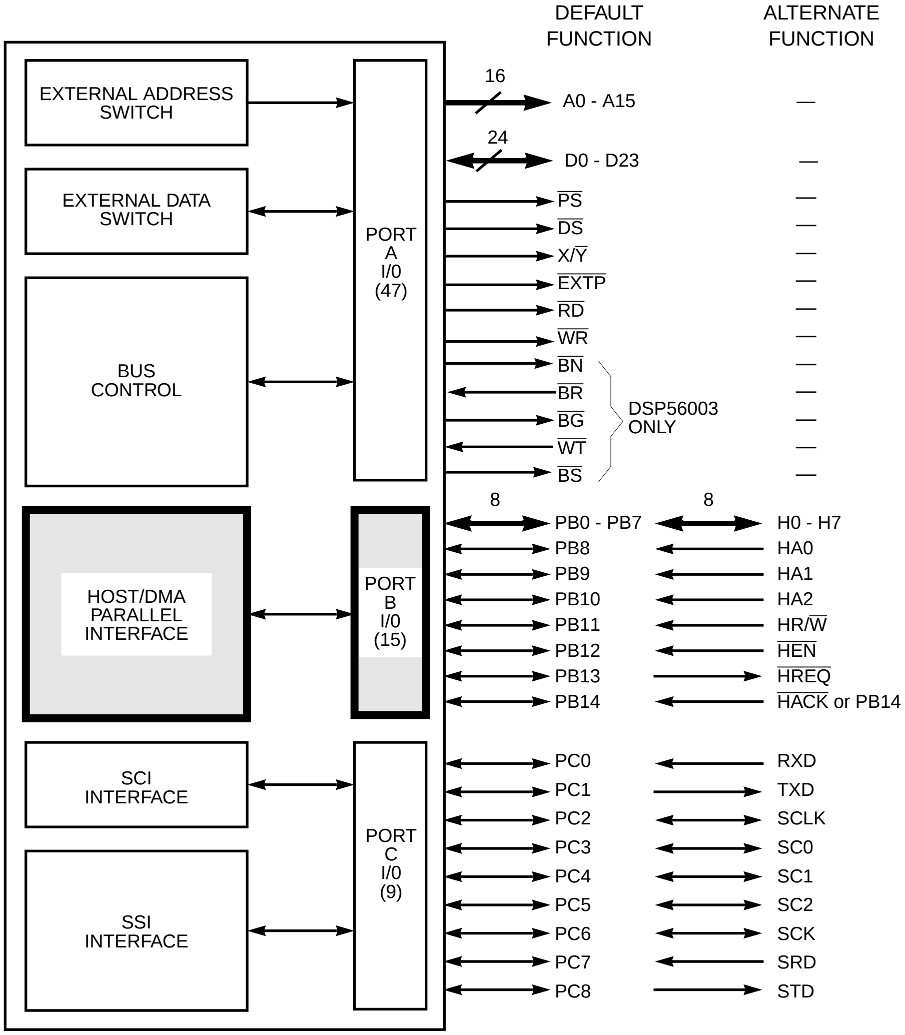


Figure 5-1 Port B Interface

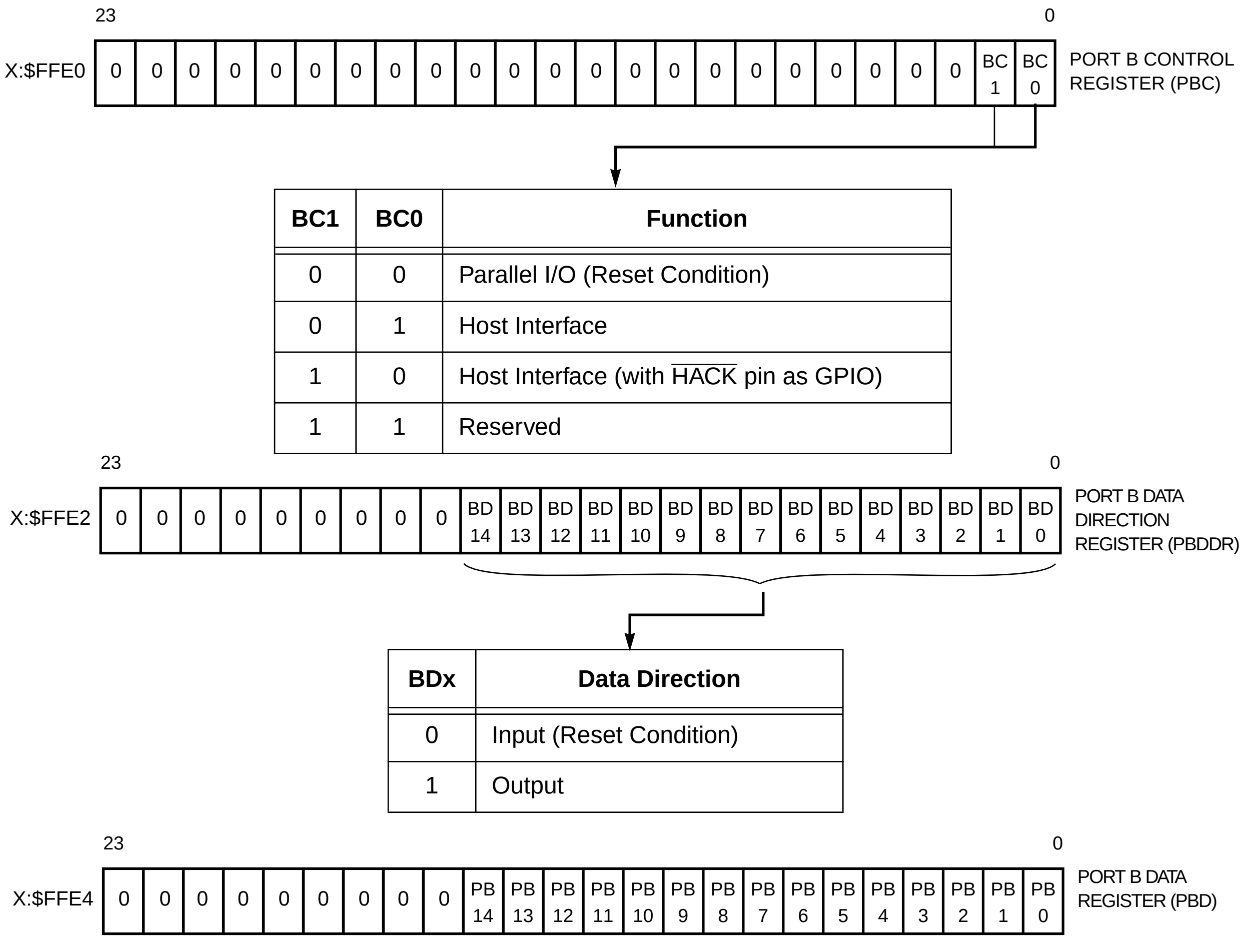
5.2 GENERAL PURPOSE I/O CONFIGURATION

When it is configured as general-purpose I/O, Port B acts as three memory-mapped registers (see Figure 5-2) that control 15 I/O pins (see Figure 5-3). They are the Port B control register (PBC), Port B data direction register (PBDDR), and Port B data register (PBD).

Reset configures Port B as general-purpose I/O with all 15 pins as inputs by clearing both the control (PBC), and data direction (PBDDR) registers (external circuitry connected to these pins may need pullups until the pins are configured for operation). There are three registers associated with each external pin.

To select between general purpose I/O and the HI, set PBC bits 0 and 1 as shown in Figure 5-2. Use the PBDDR to determine whether the corresponding bit in the PBD shall be an input pin (bit is set to zero) or an output pin (bit is set to one).

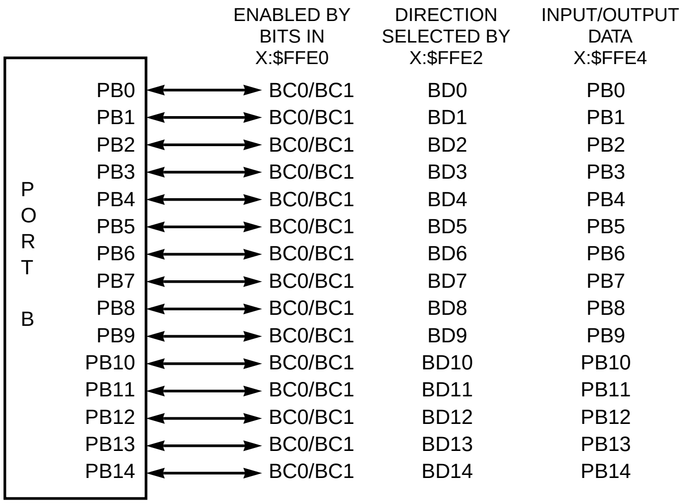
If a pin is configured as a GPIO **input** (as shown in Figure 5-4) and the processor reads the PBD, the processor sees the logic level on the pin. If the processor writes to the PBD, the data is latched there, but does not appear on the pin because the buffer is in the high-impedance state.


23
0
X:$FFE0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
BC 1
BC 0
PORT B CONTROL REGISTER (PBC)
23
0
X:$FFE2
0 0 0 0 0 0 0 0 0
BD 14 BD 13 BD 12 BD 11 BD 10 BD 9 BD 8 BD 7 BD 6 BD 5 BD 4 BD 3 BD 2 BD 1 BD 0
PORT B DATA DIRECTION REGISTER (PBDDR)
23
0
X:$FFE4
0 0 0 0 0 0 0 0 0
PB 14 PB 13 PB 12 PB 11 PB 10 PB 9 PB 8 PB 7 PB 6 PB 5 PB 4 PB 3 PB 2 PB 1 PB 0
PORT B DATA REGISTER (PBD)

BC1	BC0	Function
0	0	Parallel I/O (Reset Condition)
0	1	Host Interface
1	0	Host Interface (with $\overline{\text{HACK}}$ pin as GPIO)
1	1	Reserved

BDx	Data Direction
0	Input (Reset Condition)
1	Output

Figure 5-2 Parallel Port B Registers



	ENABLED BY BITS IN X:$FFE0	DIRECTION SELECTED BY X:$FFE2	INPUT/OUTPUT DATA X:$FFE4
PB0	BC0/BC1	BD0	PB0
PB1	BC0/BC1	BD1	PB1
PB2	BC0/BC1	BD2	PB2
PB3	BC0/BC1	BD3	PB3
PB4	BC0/BC1	BD4	PB4
PB5	BC0/BC1	BD5	PB5
PB6	BC0/BC1	BD6	PB6
PB7	BC0/BC1	BD7	PB7
PB8	BC0/BC1	BD8	PB8
PB9	BC0/BC1	BD9	PB9
PB10	BC0/BC1	BD10	PB10
PB11	BC0/BC1	BD11	PB11
PB12	BC0/BC1	BD12	PB12
PB13	BC0/BC1	BD13	PB13
PB14	BC0/BC1	BD14	PB14

Figure 5-3 Parallel Port B Pinout

If a pin is configured as a GPIO **output** and the processor reads the PBD, the processor sees the contents of the PBD rather the logic level on the pin, which allows the PBD to be used as a general purpose 15-bit register. If the processor writes to the PBD, the data is latched there and appears on the pin during the following instruction cycle (see **Section 5.2.2 Port B General Purpose I/O Timing**).

If a pin is configured as a **host** pin, the Port B GPIO registers can be used to help in debugging the HI. If the PBDDR bit for a given pin is cleared (configured as an input), the PBD will show the logic level on the pin, regardless of whether the HI function is using the pin as an input or an output.

If the PBDDR is set (configured as an output) for a given pin that is configured as a**host** pin, when the processor reads the PBD, it sees the contents of the PBD rather than the logic level on the pin - another case which allows the PBD to act as a general purpose register.

Note: The external host processor should be carefully synchronized to the DSP56003/005 to assure that the DSP and the external host will properly read status bits transmitted between them. There is more discussion of such port use considerations in sections **Section 5.3.2.7 Host Port Use Considerations – DSP Side** and **Section 5.3.6.5 Host Port Use Considerations — Host Side**.

Port Control Register Bit	Data Direction Register Bit	Pin Function
0	0	Port Input Pin

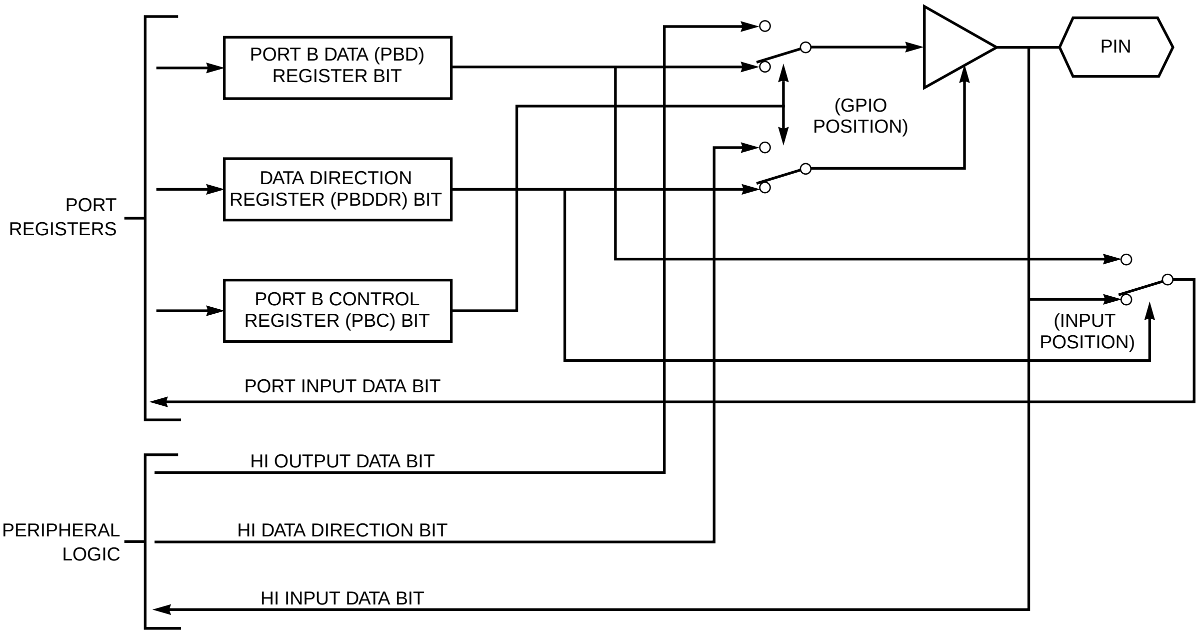
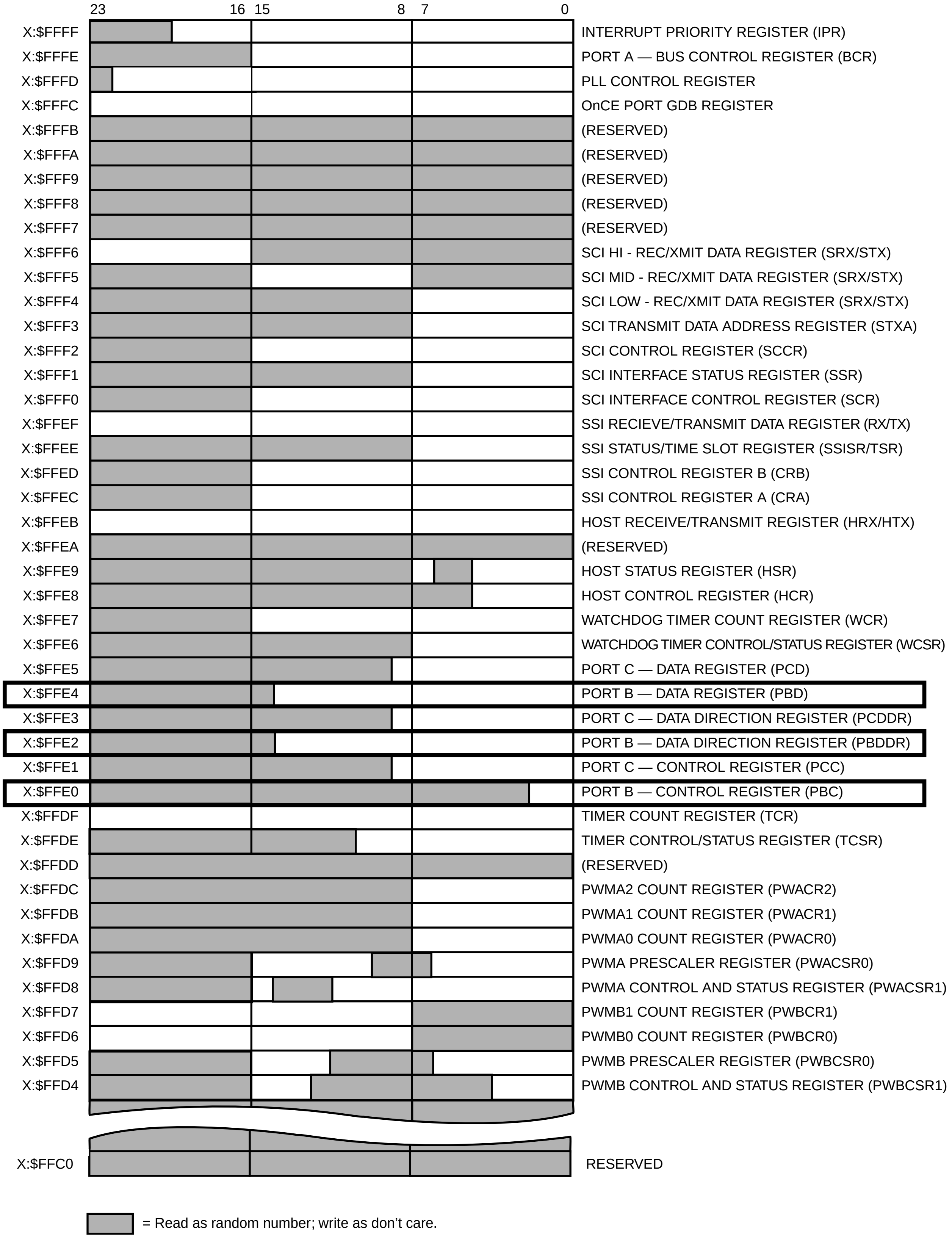


Figure 5-4 Port B I/O Pin Control Logic

5.2.1 Programming General Purpose I/O

Port B is a memory-mapped peripheral as are all of the DSP56003/005 peripherals (see Figure 5-5). The standard MOVE instruction transfers data between Port B and a register; as a result, MOVE takes two instructions to perform a memory-to-memory data transfer and uses a temporary holding register. The MOVEP instruction is specifically designed for I/O data transfer as shown in Figure 5-6. Although the MOVEP instruction may take twice as long to execute as a MOVE instruction, only one MOVEP is required for a memory-to-memory data transfer, and MOVEP does not use a temporary register. Using the MOVEP instruction allows a fast interrupt to move data to/from a peripheral to memory and execute one other instruction or move the data to an absolute address. MOVEP is the only memory-to-memory move instruction; however, one of the operands must be in the top 64 locations of either X: or Y: memory.

The bit-oriented instructions that use I/O short addressing (BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, and JSSET) can also be used to address individual bits for faster I/O processing. The digital signal processor (DSP) does not have a hardware data strobe to strobe data out of the GPIO port. If a strobe is needed, it can be implemented using software to toggle one of the GPIO pins.



X:$FFFF	INTERRUPT PRIORITY REGISTER (IPR)
X:$FFFE	PORT A — BUS CONTROL REGISTER (BCR)
X:$FFFD	PLL CONTROL REGISTER
X:$FFFC	OnCE PORT GDB REGISTER
X:$FFFB	(RESERVED)
X:$FFFA	(RESERVED)
X:$FFF9	(RESERVED)
X:$FFF8	(RESERVED)
X:$FFF7	(RESERVED)
X:$FFF6	SCI HI - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF5	SCI MID - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF4	SCI LOW - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF3	SCI TRANSMIT DATA ADDRESS REGISTER (STXA)
X:$FFF2	SCI CONTROL REGISTER (SCCR)
X:$FFF1	SCI INTERFACE STATUS REGISTER (SSR)
X:$FFF0	SCI INTERFACE CONTROL REGISTER (SCR)
X:$FFEF	SSI RECIEVE/TRANSMIT DATA REGISTER (RX/TX)
X:$FFEE	SSI STATUS/TIME SLOT REGISTER (SSISR/TSR)
X:$FFED	SSI CONTROL REGISTER B (CRB)
X:$FFEC	SSI CONTROL REGISTER A (CRA)
X:$FFEB	HOST RECEIVE/TRANSMIT REGISTER (HRX/HTX)
X:$FFEA	(RESERVED)
X:$FFE9	HOST STATUS REGISTER (HSR)
X:$FFE8	HOST CONTROL REGISTER (HCR)
X:$FFE7	WATCHDOG TIMER COUNT REGISTER (WCR)
X:$FFE6	WATCHDOG TIMER CONTROL/STATUS REGISTER (WCSR)
X:$FFE5	PORT C — DATA REGISTER (PCD)
X:$FFE4	PORT B — DATA REGISTER (PBD)
X:$FFE3	PORT C — DATA DIRECTION REGISTER (PCDDR)
X:$FFE2	PORT B — DATA DIRECTION REGISTER (PBDDR)
X:$FFE1	PORT C — CONTROL REGISTER (PCC)
X:$FFE0	PORT B — CONTROL REGISTER (PBC)
X:$FFDF	TIMER COUNT REGISTER (TCR)
X:$FFDE	TIMER CONTROL/STATUS REGISTER (TCSR)
X:$FFDD	(RESERVED)
X:$FFDC	PWMA2 COUNT REGISTER (PWACR2)
X:$FFDB	PWMA1 COUNT REGISTER (PWACR1)
X:$FFDA	PWMA0 COUNT REGISTER (PWACR0)
X:$FFD9	PWMA PRESCALER REGISTER (PWACSR0)
X:$FFD8	PWMA CONTROL AND STATUS REGISTER (PWACSR1)
X:$FFD7	PWMB1 COUNT REGISTER (PWBCR1)
X:$FFD6	PWMB0 COUNT REGISTER (PWBCR0)
X:$FFD5	PWMB PRESCALER REGISTER (PWBCSR0)
X:$FFD4	PWMB CONTROL AND STATUS REGISTER (PWBCSR1)
X:$FFC0	RESERVED

▨ = Read as random number; write as don't care.

Figure 5-5 On-Chip Peripheral Memory Map

```
           .
MOVE    #$0,X:$FFE0      ;Select Port B to be general-purpose I/O
MOVE    #$7F00,X:$FFE2   ;Select pins PB0–PB7 to be inputs
           .             ;and pins PB8–PB14 to be outputs
           .
MOVEP   #data_out,X:$FFE4 ;Put bits 8–14 of “data_out” on pins
                          ;PB8–PB14 bits 0–7 are ignored
MOVEP   X:$FFE4,#data_in  ;Put PB0–PB7 in bits 0–7 of “data_in”
```

Figure 5-6 Instructions to Write/Read Parallel Data with Port B

Figure 5-7 details the process of programming Port B as GPIO. Normally, it is not good programming practice to activate a peripheral before programming it. However, reset activates the Port B general-purpose I/O as all inputs; the alternative is to configure Port B as an HI, which may not be desirable. In this case, it is probably better to insure that Port B is initially configured for general-purpose I/O, and then configure the data direction and data registers. It may be better in some situations to program the data direction or the data registers first to prevent two devices from driving one signal. The order of steps 1, 2, and 3 in Figure 5-7 is optional and can be changed as needed.

5.2.2 Port B General Purpose I/O Timing

General purpose data written to Port B is synchronized to the central processing unit (CPU) but delayed by one instruction cycle. For example, the instruction

```
MOVE       DATA15,X:PORTB   DATA24,Y:EXTERN
```

1. writes 15 bits of data to the Port B register, but the output pins do not change until the following instruction cycle
2. writes 24 bits of data to the external Y memory, which appears on Port A during T2 and T3 of the current instruction

As a result, if it is desirable to synchronize Port A and Port B outputs, two instructions must be used:

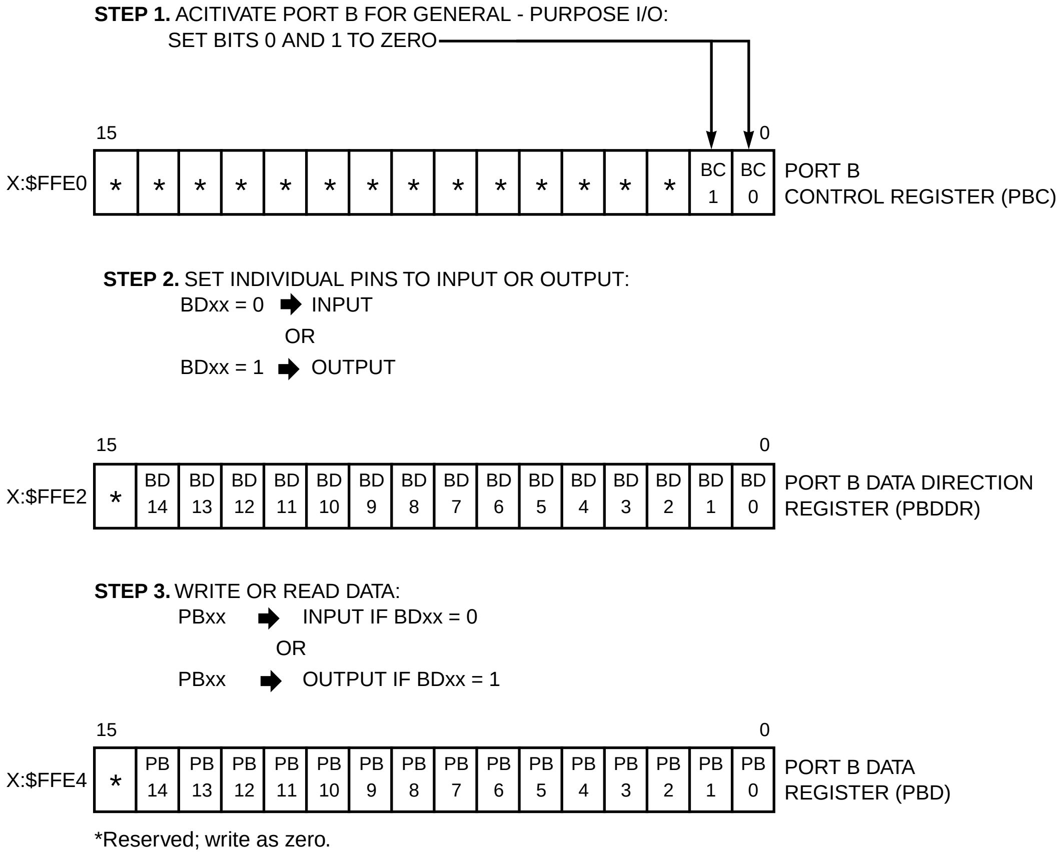
```
MOVE       DATA15,X:PORTB

NOP                         DATA24,Y:EXTERN
```

The NOP can be replaced by any instruction that allows parallel moves. Inserting one or more “MOVE DATA15,X:PORTB DATA24,Y:EXTERN” instructions between the first and second instruction effectively produces an external 39-bit write each instruction cycle with only one instruction cycle lost in setup time:

```
MOVE    DATA15,X:PORTB
MOVE    DATA15,X:PORTB     DATA24,Y:EXTERN
MOVE    DATA15,X:PORTB     DATA24,Y:EXTERN
:
:
MOVE    DATA15,X:PORTB     DATA24,Y:EXTERN
NOP                        DATA24,Y:EXTERN
```

One application of this technique is to create an extended address for Port A by concatenating the Port A address bits (instead of data bits) to the Port B general-purpose output bits. The Port B general-purpose I/O register would then work as a base address register, allowing the address space to be extended from 64K words (16 bits) to two billion words (16 bits +15 bits = 31 bits).



STEP 1. ACITIVATE PORT B FOR GENERAL - PURPOSE I/O:
SET BITS 0 AND 1 TO ZERO

X:$FFE0 — PORT B CONTROL REGISTER (PBC)

15															0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	BC 1	BC 0

STEP 2. SET INDIVIDUAL PINS TO INPUT OR OUTPUT:
BDxx = 0 ➡ INPUT
OR
BDxx = 1 ➡ OUTPUT

X:$FFE2 — PORT B DATA DIRECTION REGISTER (PBDDR)

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

STEP 3. WRITE OR READ DATA:
PBxx ➡ INPUT IF BDxx = 0
OR
PBxx ➡ OUTPUT IF BDxx = 1

X:$FFE4 — PORT B DATA REGISTER (PBD)

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

*Reserved; write as zero.

Figure 5-7 I/O Port B Configuration

Port B uses the DSP CPU four-phase clock for its operation. Therefore, if wait states are inserted in the DSP CPU timing, they also affect Port B timing. The result is that ports A and B in the previous synchronization example will always stay synchronized, regardless of how many wait states are used.

5.3 HOST INTERFACE (HI)

The HI is a byte-wide, full-duplex, double-buffered, parallel port which may be connected directly to the data bus of a host processor. The host processor may be any of a number of industry standard microcomputers or microprocessors, another DSP, or DMA hardware because this interface looks like static memory. The HI is asynchronous and consists of two banks of registers – one bank accessible to the host processor and a second bank accessible to the DSP CPU (see Figure 5-8). A brief description of the HI features is presented in the following listing:

Speed

3.3 Million Word/Sec Interrupt Driven Data Transfer Rate (This is the maximum interrupt rate for the DSP56003/005 running at 40 MHz – i.e., one interrupt every six instruction cycles.)

Signals (15 Pins)

H0–H7	Host Data Bus
HA0-HA2	Host Address Select
HR/$\overline{\text{W}}$	Host Read/Write Control
$\overline{\text{HEN}}$	Host Transfer Enable
$\overline{\text{HREQ}}$	Host Request
$\overline{\text{HACK}}$	Host Acknowledge

Interface – DSP CPU Side

Mapping: Three X: Memory Locations
Data Word: 24 Bits

Transfer Modes:
- DSP to Host
- Host to DSP
- Host Command

Handshaking Protocols:
- Software Polled
- Interrupt Driven (Fast or Long Interrupts)
- Direct Memory Access

Instructions:
- Memory-mapped registers allow the standard MOVE instruction to be used.
- Special MOVEP instruction provides for I/O service capability using fast interrupts.
- Bit addressing instructions
- simplify I/O service routines.
- I/O short addressing provides faster execution with fewer instruction words.

Interface – Host Side

Mapping:
- Eight Consecutive Memory Locations
- Memory-Mapped Peripheral for Microprocessors, DMA Controllers, etc.

Data Word: Eight Bits

Transfer Modes:
- DSP to Host
- Host to DSP
- Host Command
- Mixed 8-, 16-, and 24-Bit Data Transfers

Handshaking Protocols:
- Software Polled
- Interrupt Driven and Compatible with MC68000
- Cycle Stealing DMA with Initialization

Dedicated Interrupts:
- Separate Interrupt Vectors for Each Interrupt Source
- Special host commands force DSP CPU interrupts under host processor control, which are useful for:
  - Real-Time Production Diagnostics
  - Debugging Window for Program Development
  - Host Control Protocols and DMA Setup

Figure 5-8 is a block diagram showing the registers in the HI. These registers can be divided vertically down the middle into registers visible to the host processor on the left and registers visible to the DSP on the right. They can also be divided horizontally into control at the top, DSP-to-host data transfer in the middle (HTX, RXH, RXM, and RXL), and host-to-DSP data transfer at the bottom (THX, TXM, TXL, and HRX).

5.3.1 Host Interface – DSP CPU Viewpoint

The DSP CPU views the HI as a memory-mapped peripheral occupying three 24-bit words in data memory space. The DSP may use the HI as a normal memory-mapped peripheral, using either standard polled or interrupt programming techniques. Separate transmit and receive data registers are double buffered to allow the DSP and host processor to efficiently transfer data at high speed. Memory mapping allows DSP CPU communication with the HI registers to be accomplished using standard instructions and addressing modes. In addition, the MOVEP instruction allows HI-to-memory and memory-to-HI data transfers without going through an intermediate register. Both hardware and software reset disable the HI and change Port B to general-purpose I/O with all pins designated as inputs.

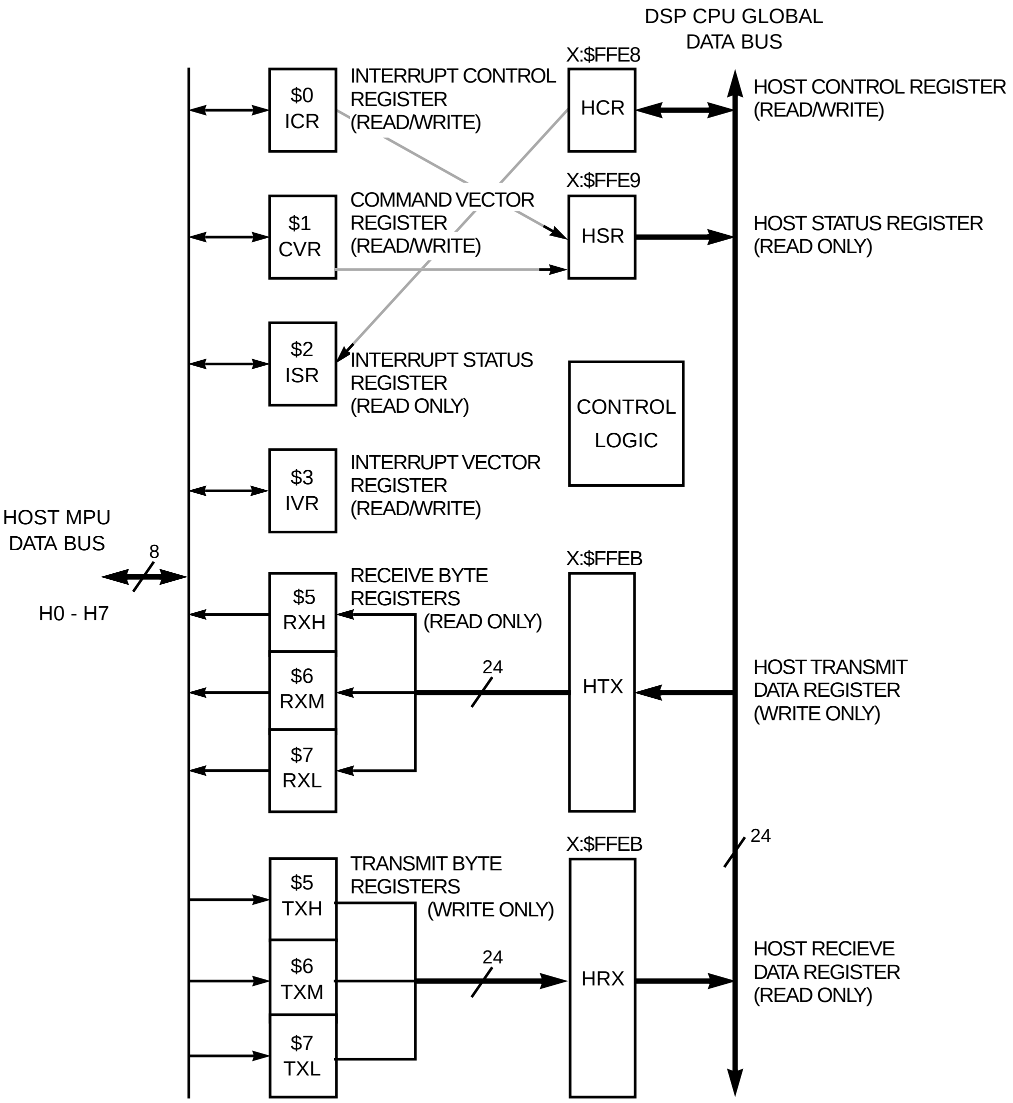
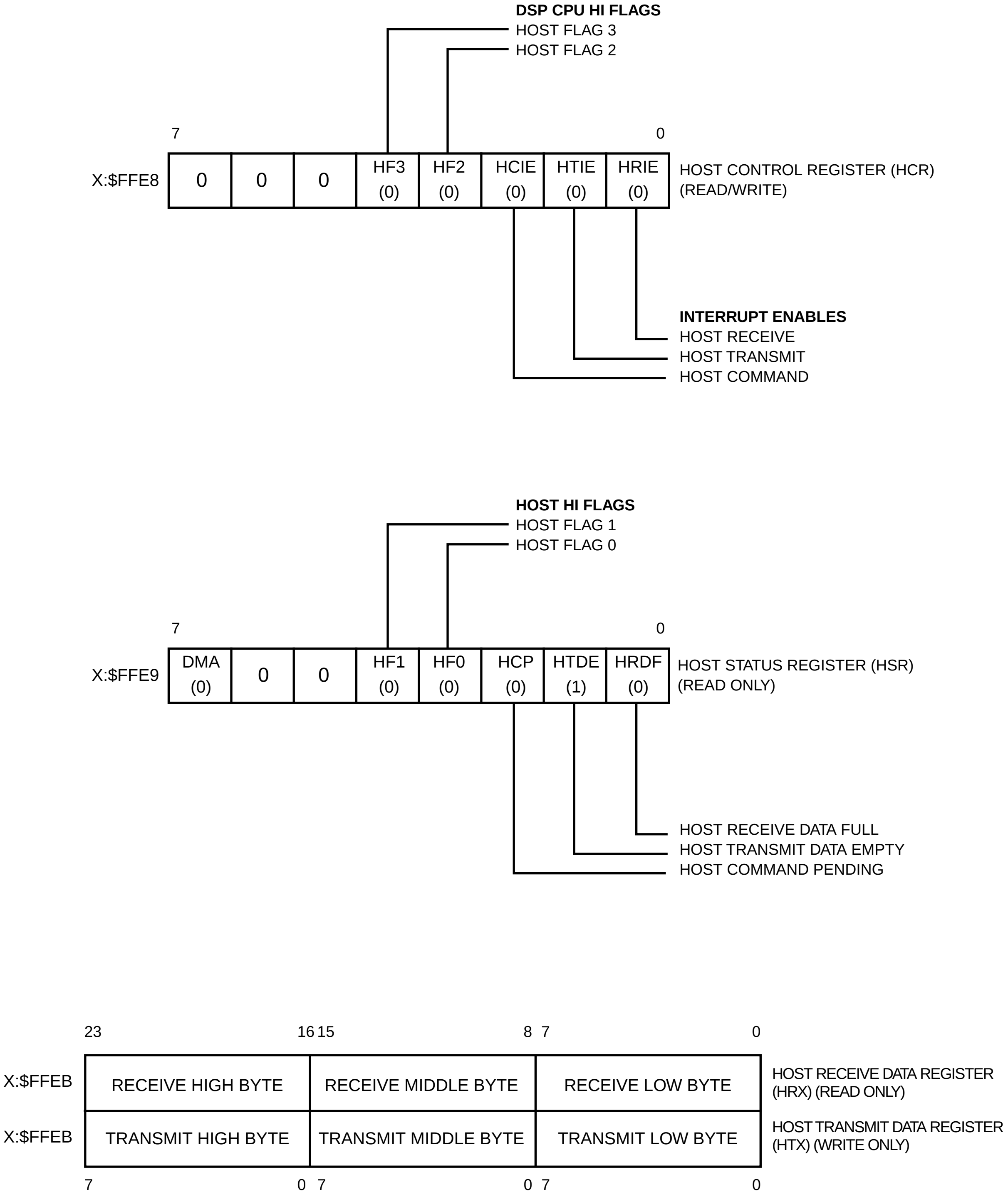


Figure 5-8 HI Block Diagram

5.3.2 Programming Model – DSP CPU Viewpoint

The HI has two programming models: one for the DSP programmer and one for the host processor programmer. In most cases, the notation used reflects the DSP perspective. The HI – DSP programming model is shown in Figure 5-9. There are three registers: a control register (HCR), a status register (HSR), and a data transmit/receive register (HTX/HRX). These registers can only be accessed by the DSP56003/005; they can not be accessed by the host processor. The HI host processor programming model is shown in Figure 5-12.



NOTE: The numbers in parentheses are reset values.

Figure 5-9 Host Interface Programming Model — DSP Viewpoint

The following paragraphs describe the purpose and operation of each bit in each register of the HI visible to the DSP CPU. The effects of the different types of reset on these registers are shown. A brief discussion of interrupts and operation of the DSP side of the HI complete the programming model from the DSP viewpoint. The programming model from the host viewpoint begins at **Section 5.3.3.1 Programming Model – Host Processor Viewpoint**.

5.3.2.1 Host Control Register (HCR)

The HCR is an 8-bit read/write control register used by the DSP to control the HI interrupts and flags. The HCR cannot be accessed by the host processor. It occupies the low-order byte of the internal data bus; the high-order portion is zero filled. Any reserved bits are read as zeros and should be programmed as zeros for future compatibility. (The bit manipulation instructions are useful for accessing the individual bits in the HCR.) The contents of the HCR are cleared on hardware or software reset. The control bits are described in the following paragraphs.

5.3.2.1.1 HCR Host Receive Interrupt Enable (HRIE) Bit 0

The HRIE bit is used to enable a DSP interrupt when the host receive data full (HRDF) status bit in the host status register (HSR) is set. When HRIE is cleared, HRDF interrupts are disabled. When HRIE is set, a host receive data interrupt request will occur if HRDF is also set. Hardware and software resets clear HRIE.

5.3.2.1.2 HCR Host Transmit Interrupt Enable (HTIE) Bit 1

The HTIE bit is used to enable a DSP interrupt when the host transmit data empty (HTDE) status bit in the HSR is set. When HTIE is cleared, HTDE interrupts are disabled. When HTIE is set, a host transmit data interrupt request will occur if HTDE is also set. Hardware and software resets clear the HTIE.

5.3.2.1.3 HCR Host Command Interrupt Enable (HCIE) Bit 2

The HCIE bit is used to enable a vectored DSP interrupt when the host command pending (HCP) status bit in the HSR is set. When HCIE is cleared, HCP interrupts are disabled. When HCIE is set, a host command interrupt request will occur if HCP is also set. The starting address of this interrupt is determined by the host vector (HV). Hardware and software resets clear the HCIE.

5.3.2.1.4 HCR Host Flag 2 (HF2) Bit 3

The HF2 bit is used as a general-purpose flag for DSP-to-host communication. HF2 may be set or cleared by the DSP. HF2 is visible in the interrupt status register (ISR) on the host processor side (see Figure 5-10). Hardware and software resets clear HF2.

5.3.2.1.5 HCR Host Flag 3 (HF3) Bit 4

The HF3 bit is used as a general-purpose flag for DSP-to-host communication. HF3 may be set or cleared by the DSP. HF3 is visible in the ISR on the host processor side (see Figure 5-10). Hardware and software resets clear HF3.

Note: There are four host flags: two used by the host to signal the DSP (HF0 and HF1) and two used by the DSP to signal the host processor (HF2 and HF3). They are general purpose flags and are not designated for any specific purpose. The host flags do not cause interrupts; they must be polled to see if they have changed. These flags can be used individually or as encoded pairs. See Section 5.3.2.7 Host Port Use Considerations – DSP Side for additional information. An example of the use of host flags is the bootstrap loader, which is listed in Appendix A. Host flags are used to tell the bootstrap program whether or not to terminate early.

5.3.2.1.6 HCR Reserved Bits 5, 6, and 7

These unused bits are reserved for future expansion and should be written with zeros for upward compatibility.

5.3.2.2 Host Status Register (HSR)

The HSR is an 8-bit read-only status register used by the DSP to interrogate status and flags of the HI. It can not be directly accessed by the host processor. When the HSR is read to the internal data bus, the register contents occupy the low-order byte of the data bus; the high-order portion is zero filled. The status bits are described in the following paragraphs.

5.3.2.2.1 HSR Host Receive Data Full (HRDF) Bit 0

The HRDF bit indicates that the host receive data register (HRX) contains data from the host processor. HRDF is set when data is transferred from the TXH:TXM:TXL registers to the HRX register. HRDF is cleared when HRX is read by the DSP. HRDF can also be cleared by the host processor using the initialize function. Hardware, software, individual, and STOP resets clear HRDF.

5.3.2.2.2 HSR Host Transmit Data Empty (HTDE) Bit 1

The HTDE bit indicates that the host transmit data register (HTX) is empty and can be written by the DSP. HTDE is set when the HTX register is transferred to the RXH:RXM:RXL registers. HTDE is cleared when HTX is written by the DSP. HTDE can also be set by the host processor using the initialize function. Hardware, software, individual, and STOP sets HTDE.

5.3.2.2.3 HSR Host Command Pending (HCP) Bit 2

The HCP bit indicates that the host has set the HC bit and that a host command interrupt is pending. The HCP bit reflects the status of the HC bit in the command vector register (CVR). HC and HCP are cleared by the DSP exception hardware when the exception is taken. The host can clear HC, which also clears HCP. Hardware, software, individual, and STOP resets clear HCP.

5.3.2.2.4 HSR Host Flag 0 (HF0) Bit 3

The HF0 bit in the HSR indicates the state of host flag 0 in the ICR on the host processor side. HF0 can only be changed by the host processor (see Figure 5-10). Hardware, software, individual, and STOP resets clear HF0.

5.3.2.2.5 HSR Host Flag 1 (HF1) Bit 4

The HF1 bit in the HSR indicates the state of host flag 1 in the ICR on the host processor side. HF1 can only be changed by the host processor (see Figure 5-10). Hardware, software, individual, and STOP resets clear HF1.

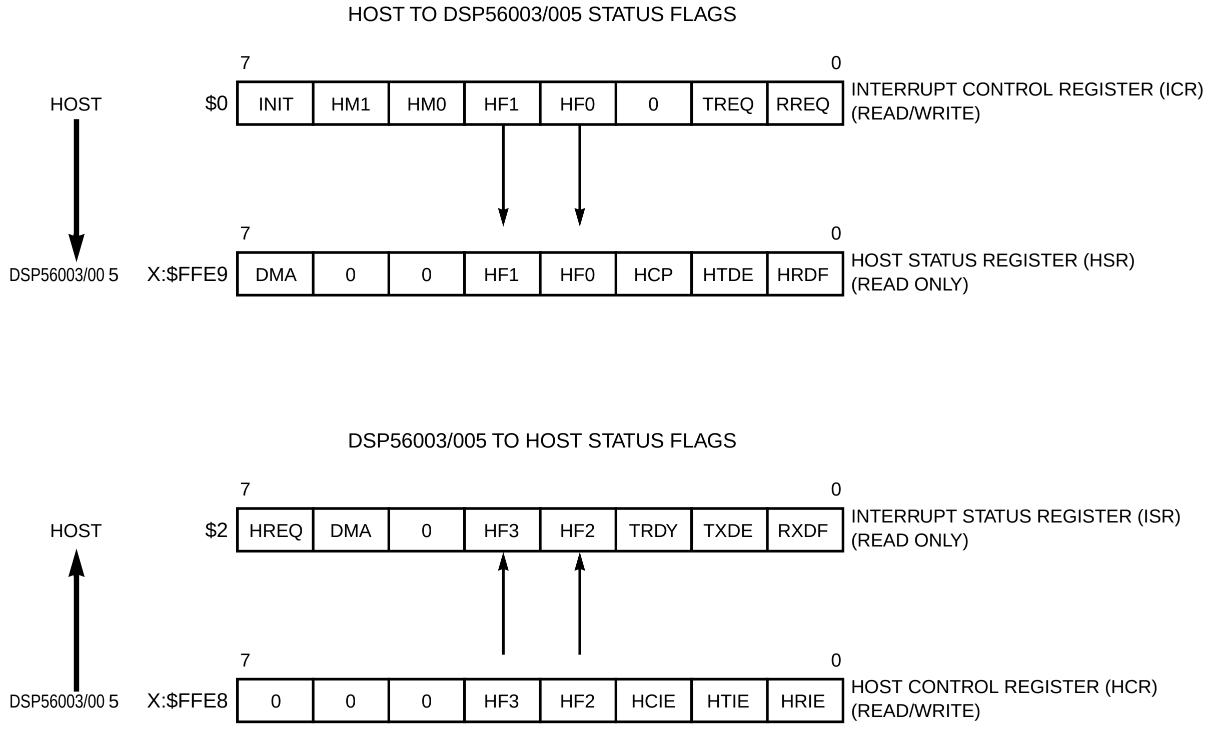


Figure 5-10 Host Flag Operation

5.3.2.2.6 HSR Reserved Bits 5 and 6

These status bits are reserved for future expansion and read as zero during DSP read operations.

5.3.2.2.7 HSR DMA Status (DMA) Bit 7

The DMA bit indicates that the host processor has enabled the DMA mode of the HI by setting HM1 or HM0 to one. When the DMA bit is zero, it indicates that the DMA mode is disabled by the HM0 and HM1 bits in the ICR and that no DMA operations are pending. When the DMA bit is set, the DMA mode has been enabled if one or more of the host mode bits have been set to one. The channel not in use can be used for polled or interrupt operation by the DSP. Hardware, software, individual, and STOP resets clear the DMA bit.

5.3.2.3 Host Receive Data Register (HRX)

The HRX register is used for host-to-DSP data transfers. The HRX register is viewed as a 24-bit read-only register by the DSP CPU. The HRX register is loaded with 24-bit data from the transmit data registers (TXH:TXM:TXL) on the host processor side when both the transmit data register empty TXDE (host processor side) and DSP host receive data full (HRDF) bits are cleared. This transfer operation sets TXDE and HRDF. The HRX register contains valid data when the HRDF bit is set. Reading HRX clears HRDF. The DSP may program the HRIE bit to cause a host receive data interrupt when HRDF is set. Resets do not affect HRX.

5.3.2.4 Host Transmit Data Register (HTX)

The HTX register is used for DSP-to-host data transfers. The HTX register is viewed as a 24-bit write-only register by the DSP CPU. Writing the HTX register clears HTDE. The DSP may program the HTIE bit to cause a host transmit data interrupt when HTDE is set. The HTX register is transferred as 24-bit data to the receive byte registers (RXH:RXM:RXL) if both the HTDE bit (DSP CPU side) and receive data full (RXDF) status bits (host processor side) are cleared. This transfer operation sets RXDF and HTDE. Data should not be written to the HTX until HTDE is set to prevent the previous data from being overwritten. Resets do not affect HTX.

5.3.2.5 Register Contents After Reset

Table 5-1 shows the results of four reset types on bits in each of the HI registers seen by the DSP CPU. The hardware reset (HW) is caused by the $\overline{\text{RESET}}$ signal; the software reset (SW) is caused by executing the RESET instruction; the individual reset (IR) is caused by clearing PBC register bits 0 and 1, and the stop reset (ST) is caused by executing the STOP instruction.

Table 5-1 Host Registers after Reset — DSP CPU Side

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	IR Reset	ST Reset
HCR	HF(3 - 2)	0	0	—	—
	HCIE	0	0	—	—
	HTIE	0	0	—	—
	HRIE	0	0	—	—
HSR	DMA	0	0	0	0
	HF(1 - 0)	0	0	0	0
	HCP	0	0	0	0
	HTDE	1	1	1	1
	HRDF	0	0	0	0
HRX	HRX (23 - 0)	—	—	—	—
HTX	HTX (23 - 0)	—	—	—	—

5.3.2.6 Host Interface DSP CPU Interrupts

The HI may request interrupt service from either the DSP or the host processor. The DSP CPU interrupts are internal and do not require the use of an external interrupt pin (see Figure 5-11). When the appropriate mask bit in the HCR is set, an interrupt condition caused by the host processor sets the appropriate bit in the HSR, which generates an interrupt request to the DSP CPU. The DSP acknowledges interrupts caused by the host processor by jumping to the appropriate interrupt service routine. The three possible interrupts are 1) receive data register full, 2) transmit data register empty, and 3) host command. The host command can access any interrupt vector in the interrupt vector table although it has a set of vectors reserved for host command use. The DSP interrupt service routine must read or write the appropriate HI register (clearing HRDF or HTDE, for example) to clear the interrupt. In the case of host command interrupts, the interrupt acknowledge from the program controller will clear the pending interrupt condition.

5.3.2.7 Host Port Use Considerations – DSP Side

Synchronization is a common problem when two asynchronous systems are connected, and careful synchronization is required when reading multi-bit registers that are written by another asynchronous system. The considerations for proper operation on the DSP CPU side are discussed in the following paragraphs, and considerations for the host processor side are discussed in **Section 5.3.6.5 Host Port Use Considerations — Host Side**.

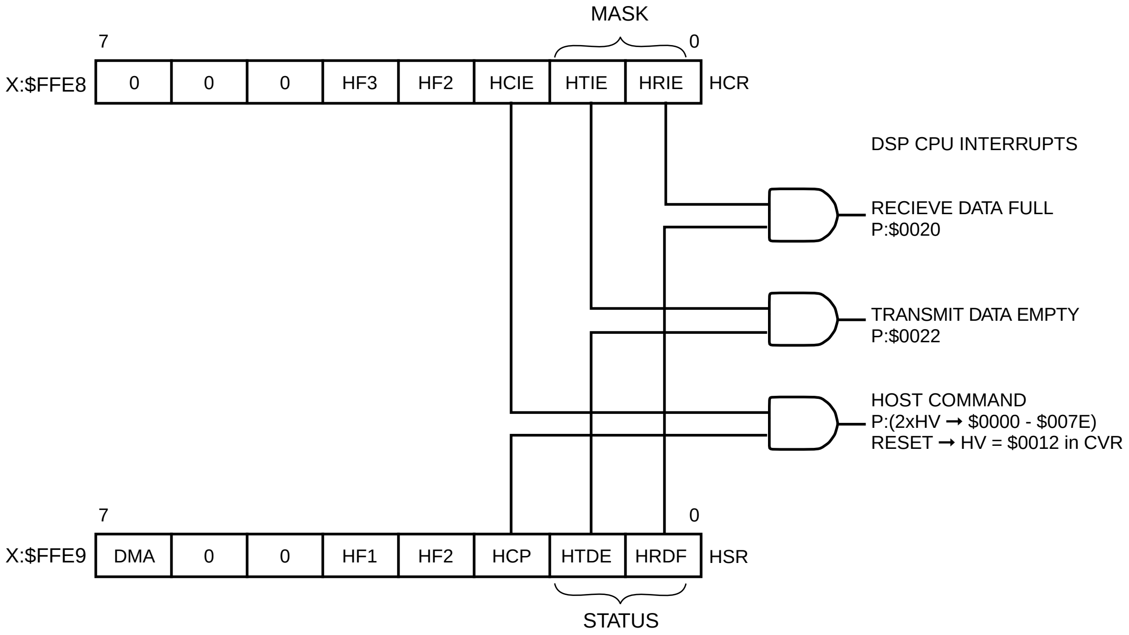


Figure 5-11 HSR–HCR Operation

DMA, HF1, HF0, HCP, HTDE, and HRDF status bits are set or cleared by the host processor side of the interface. These bits are individually synchronized to the DSP clock.

The only system problem with reading status occurs if HF1 and HF0 are encoded as a pair because each of their four combinations (00, 01, 10, and 11) has significance. There is a small possibility that the DSP will read the status bits during the transition and receive “01” or “10” instead of “11”. The solution to this potential problem is to read the bits twice for consensus (See **Section 5.3.6.5 Host Port Use Considerations — Host Side** for additional information).

5.3.3 Host Interface – Host Processor Viewpoint

The HI appears to the host processor as eight words of byte-wide static memory. The host may access the HI asynchronously by using polling techniques or interrupt-based techniques. Separate transmit and receive data registers are double buffered to allow the DSP CPU and host processor to transfer data efficiently at high speed. The HI contains a rudimentary DMA controller, which makes generating addresses (HA0–HA2) for the TX/RX registers in the HI unnecessary.

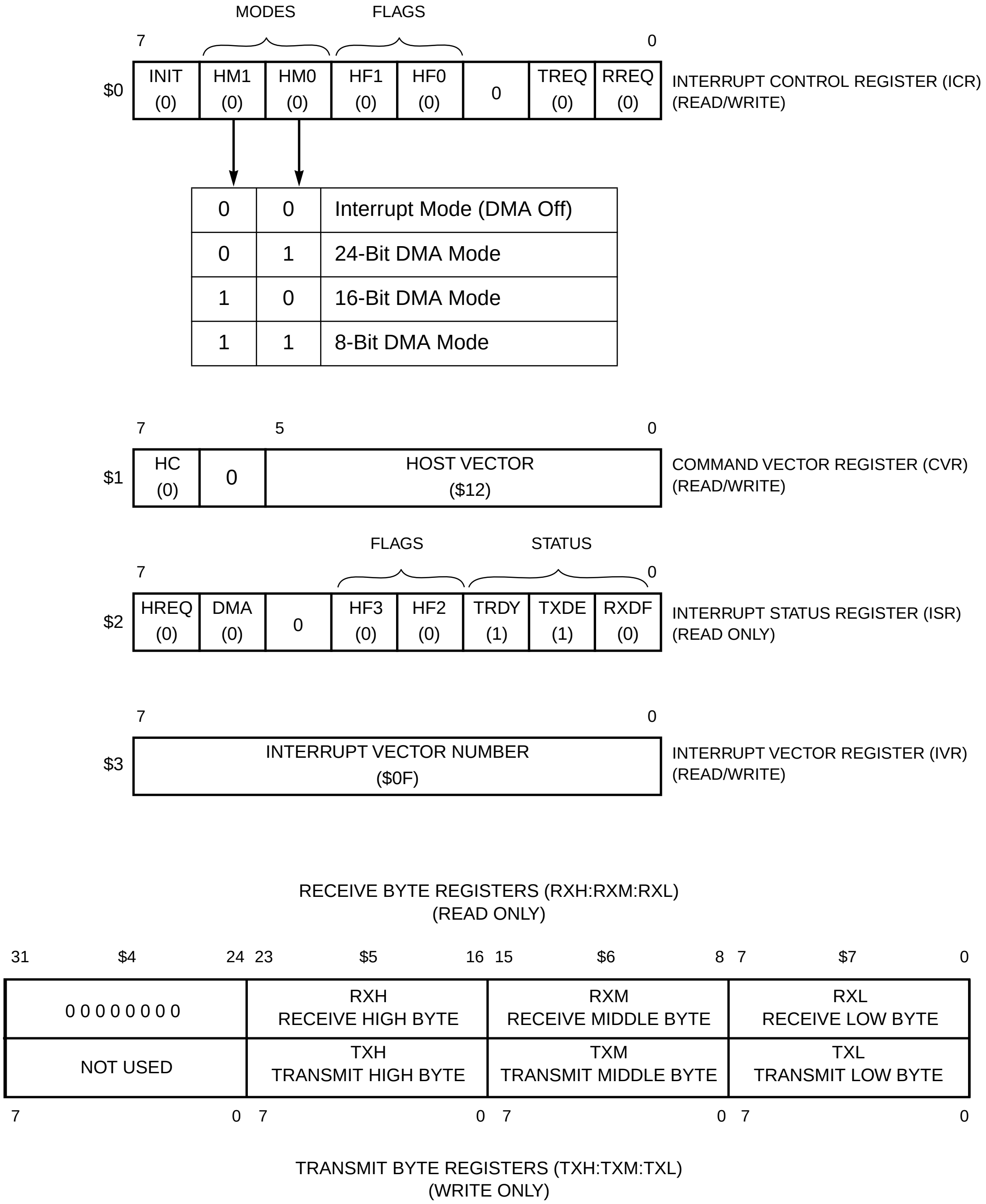
5.3.3.1 Programming Model – Host Processor Viewpoint

The HI appears to the host processor as a memory-mapped peripheral occupying eight bytes in the host processor address space (see Figure 5-12 and Figure 5-13). These registers can be viewed as one control register (ICR), one status register (ISR), three data registers (RXH/TXH, RXM/TXM, and RXL/TXL), and two vector registers (IVR and CVR). The CVR is a special command register that is used by the host processor to issue commands to the DSP. These registers can be accessed only by the host processor; they can not be accessed by the DSP CPU. Host processors may use standard host processor instructions (e.g., byte move) and addressing modes to communicate with the HI registers. The HI registers are addressed so that 8-bit MC6801-type host processors can use 16-bit load (LDD) and store (STD) instructions for data transfers. The 16-bit MC68000/MC68010 host processor can address the HI using the special MOVEP instruction for word (16-bit) or long-word (32-bit) transfers. The 32-bit MC68020 host processor can use its dynamic bus sizing feature to address the HI using standard MOVE word (16-bit), long-word (32-bit) or quad-word (64-bit) instructions. The $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ handshake flags are provided for polled or interrupt-driven data transfers with the host processor. Because the DSP interrupt response is sufficiently fast, most host microprocessors can load or store data at their maximum programmed I/O (non-DMA) instruction rate without testing the handshake flags for each transfer. If the full handshake is not needed, the host processor can treat the DSP as fast memory, and data can be transferred between the host processor and the DSP at the fastest host processor data rate. DMA hardware may be used with the handshake flags to transfer data without host processor intervention.

One of the most innovative features of the host interface is the host command feature. With this feature, the host processor can issue vectored exception requests to the DSP56003/005. The host may select any one of 128 DSP56003/005 exception routines to be executed by writing a vector address register in the HI. This flexibility allows the host programmer to execute up to 128 preprogrammed functions inside the DSP56003/005. For example, host exceptions can allow the host processor to read or write DSP56003/005 registers (X, Y, or program memory locations), force exception handlers (e.g., SSI, SCI, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$ exception routines), and perform control and debugging operations if exception routines are implemented in the DSP56003/005 to perform these tasks.

5.3.3.2 Interrupt Control Register (ICR)

The ICR is an 8-bit read/write control register used by the host processor to control the HI interrupts and flags. ICR cannot be accessed by the DSP CPU. ICR is a read/write register, which allows the use of bit manipulation instructions on control register bits. The control bits are described in the following paragraphs.



HM1	HM0	
0	0	Interrupt Mode (DMA Off)
0	1	24-Bit DMA Mode
1	0	16-Bit DMA Mode
1	1	8-Bit DMA Mode

NOTE: The numbers in parentheses are reset values.

Figure 5-12 Host Processor Programming Model — Host Side

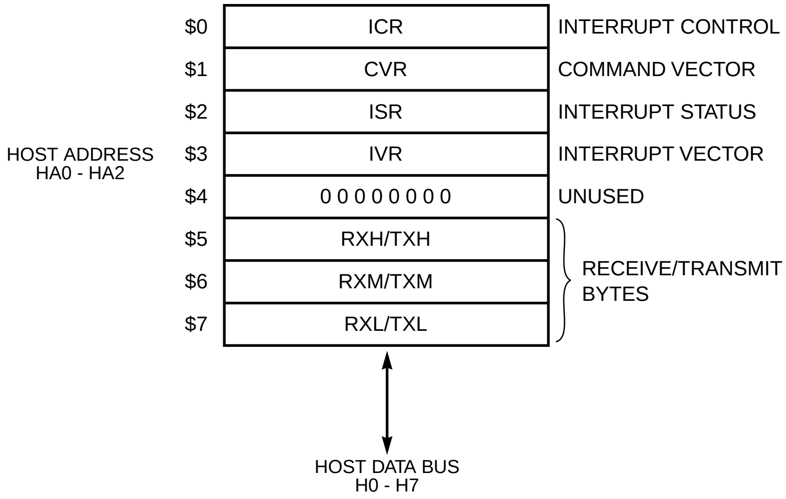


Figure 5-13 HI Register Map

5.3.3.2.1 ICR Receive Request Enable (RREQ) Bit 0

The RREQ bit is used to control the $\overline{\text{HREQ}}$ pin for host receive data transfers.

In interrupt mode (DMA off), RREQ is used to enable interrupt requests via the external host request ($\overline{\text{HREQ}}$) pin when the receive data register full (RXDF) status bit in the ISR is set. When RREQ is cleared, RXDF interrupts are disabled. When RREQ is set, the external $\overline{\text{HREQ}}$ pin will be asserted if RXDF is set.

In DMA modes, RREQ must be set or cleared by software to select the direction of DMA transfers. Setting RREQ sets the direction of DMA transfer to be DSP to host and enables the $\overline{\text{HREQ}}$ pin to request data transfer. Hardware, software, individual, and STOP resets clear RREQ.

5.3.3.2.2 ICR Transmit Request Enable (TREQ) Bit 1

The TREQ bit is used to control the $\overline{\text{HREQ}}$ pin for host transmit data transfers.

In interrupt mode (DMA off), TREQ is used to enable interrupt requests via the external $\overline{\text{HREQ}}$ pin when the transmit data register empty (TXDE) status bit in the ISR is set. When TREQ is cleared, TXDE interrupts are disabled. When TREQ is set, the external $\overline{\text{HREQ}}$pin will be asserted if TXDE is set.

In DMA modes, TREQ must be set or cleared by software to select the direction of DMA transfers. Setting TREQ sets the direction of DMA transfer to be host to DSP and enables the $\overline{\text{HREQ}}$ pin to request data transfer. Hardware, software, individual, and STOP resets clear TREQ.

Table 5-2 summarizes the effect of RREQ and TREQ on the $\overline{\text{HREQ}}$ pin.

Table 5-2 $\overline{\text{HREQ}}$ Pin Definition

TREQ	RREQ	$\overline{\text{HREQ}}$ Pin
Interrupt Mode		
0	0	No Interrupts (Polling)
0	1	RXDF Request (Interrupt)
1	0	TXDE Request (Interrupt)
1	1	RXDF and TXDE Request (Interrupts)
DMA Mode		
0	0	No DMA
0	1	DSP to Host Request (RX)
1	0	Host to DSP Request (TX)
1	1	Undefined (Illegal)

5.3.3.2.3 ICR Reserved Bit 2

This bit, which is reserved and unused, reads as a logic zero.

5.3.3.2.4 ICR Host Flag 0 (HF0) Bit 3

The HF0 bit is used as a general-purpose flag for host-to-DSP communication. HF0 may be set or cleared by the host processor and cannot be changed by the DSP. HF0 is visible in the HSR on the DSP CPU side of the HI (see Figure 5-10). Hardware, software, individual, and STOP resets clear HF0.

5.3.3.2.5 ICR Host Flag 1 (HF1) Bit 4

The HF1 bit is used as a general-purpose flag for host-to-DSP communication. HF1 may be set or cleared by the host processor and cannot be changed by the DSP. Hardware, software, individual, and STOP resets clear HF1.

5.3.3.2.6 ICR Host Mode Control (HM1 and HM0 bits) Bits 5 and 6

The HM0 and HM1 bits select the transfer mode of the HI (see Table 5-3). HM1 and HM0 enable the DMA mode of operation or interrupt (non-DMA) mode of operation.

When both HM1 and HM0 are cleared, the DMA mode is disabled, and the TREQ and RREQ control bits are used for host processor interrupt control via the external $\overline{\text{HREQ}}$ output pin. Also, in the non-DMA mode, the $\overline{\text{HACK}}$ input pin is used for the MC68000 Family vectored interrupt acknowledge input.

Table 5-3 Host Mode Bit Definition

HM1	HM0	Mode
0	0	Interrupt Mode (DMA Off)
0	1	DMA Mode (24 Bit)
1	0	DMA Mode (16 Bit)
1	1	DMA Mode (8 Bit)

When HM1 or HM0 are set, the DMA mode is enabled, and the $\overline{\text{HREQ}}$ pin is used to request DMA transfers. When the DMA mode is enabled, the TREQ and RREQ bits select the direction of DMA transfers. The $\overline{\text{HACK}}$ input pin is used as a DMA transfer acknowledge input. If the DMA direction is from DSP to host, the contents of the selected register are enabled onto the host data bus when $\overline{\text{HACK}}$ is asserted. If the DMA direction is from host to DSP, the selected register is written from the host data bus when $\overline{\text{HACK}}$ is asserted.

The size of the DMA word to be transferred is determined by the DMA control bits, HM0 and HM1. The HI register selected during a DMA transfer is determined by a 2-bit address counter, which is preloaded with the value in HM1 and HM0. The address counter substitutes for the HA1 and HA0 bits of the HI during a DMA transfer. The host address bit (HA2) is forced to one during each DMA transfer. The address counter can be initialized with the INIT bit feature. After each DMA transfer on the host data bus, the address counter is incremented to the next register. When the address counter reaches the highest register (RXL or TXL), the address counter is not incremented but is loaded with the value in HM1 and HM0. This allows 8-, 16- or 24-bit data to be transferred in a circular fashion and eliminates the need for the DMA controller to supply the HA2, HA1, and HA0 pins. For 16- or 24-bit data transfers, the DSP CPU interrupt rate is reduced by a factor of 2 or 3, respectively, from the host request rate – i.e., for every two or three host processor data transfers of one byte each, there is only one 24-bit DSP CPU interrupt.

Hardware, software, individual, and STOP resets clear HM1 and HM0.

5.3.3.2.7 ICR Initialize Bit (INIT) Bit 7

The INIT bit is used by the host processor to force initialization of the HI hardware. Initialization consists of configuring the HI transmit and receive control bits and loading HM1 and HM0 into the internal DMA address counter. Loading HM1 and HM0 into the DMA address counter causes the HI to begin transferring data on a word boundary rather than transferring only part of the first data word.

Table 5-4 $\overline{\text{HREQ}}$ Pin Definition

TREQ	RREQ	After INIT Execution	Transfer Direction Initialized
Interrupt Mode (HM1 = 0, HM0 = 0) INIT Execution			
0	0	INIT = 0; Address Counter = 00	None
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = 00	DSP to Host
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = 00	Host to DSP
1	1	INIT = 0; RXDF = 0; HTDE = 1; TXDE = 1; HRDF = 0; Address Counter = 00	Host to/from DSP
DMA Mode (HM1 or HM0 = 1) INIT Execution			
0	0	INIT = 0; Address Counter = HM1, HM0	None
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = HM1, HM0	DSP to Host
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = HM1, HM0	Host to DSP
1	1	Undefined (Illegal)	Undefined

There are two methods of initialization:

1. allowing the DMA address counter to be automatically set after transferring a word
2. setting the INIT bit, which sets the DMA address counter.

Using the INIT bit to initialize the HI hardware may or may not be necessary, depending on the software design of the interface.

The type of initialization done when the INIT bit is set depends on the state of TREQ and RREQ in the HI. The INIT command, which is local to the HI, is designed to conveniently configure the HI into the desired data transfer mode. The commands are described in the following paragraphs and in Table 5-4. The host sets the INIT bit, which causes the HI hardware to execute the INIT command. The interface hardware clears the INIT bit when the command has been executed. Hardware, software, individual, and STOP resets clear INIT.

INIT execution always loads the DMA address counter and clears the channel according to TREQ and RREQ. INIT execution is not affected by HM1 and HM0.

The internal DMA counter is incremented with each DMA transfer (each $\overline{\text{HACK}}$ pulse) until it reaches the last data register (RXL or TXL). When the DMA transfer is completed, the counter is loaded with the value of the HM1 and HM0 bits. When changing the size of the DMA word (changing HM0 and HM1 in the ICR), the DMA counter is not automatically updated, and, as a result, the DMA counter will point to the wrong data register immediately after HM1 and HM0 are changed. The INIT function must be used to preset the internal DMA counter correctly. Always set INIT after changing HM0 and HM1. However, the DMA counter can not be initialized in the middle of a DMA transfer. Even though the INIT bit is set, the internal DMA controller will wait until after completing the data transfer in progress before executing the initialization.

5.3.3.3 Command Vector Register (CVR)

The host processor uses the CVR to cause the DSP to execute a vectored interrupt. The host command feature is independent of the data transfer mechanisms in the HI. It can be used to cause any of the 128 possible interrupt routines in the DSP CPU to be executed. The command vector register is shown in Figure 5-14.

5.3.3.3.1 CVR Host Vector (HV) Bits 0–5

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

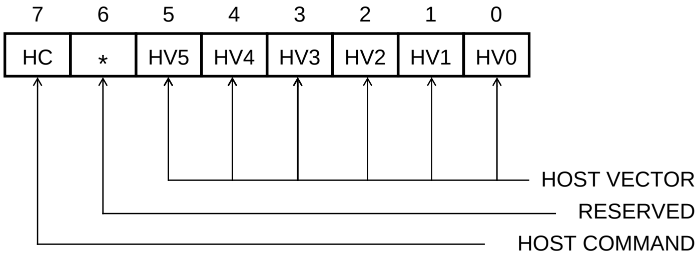


Figure 5-14 Command Vector Register

The host processor can select any of the 128 possible exception routine starting addresses in the DSP by writing the exception routine starting address divided by 2 into HV. This means that the host processor can force any of the existing exception handlers (SSI, SCI, IRQA, IRQB, etc.) and can use any of the reserved or otherwise unused starting addresses provided they have been preprogrammed in the DSP. HV is set to $12 (vector location $0024) by hardware, software, individual, and STOP resets.

CAUTION

The HV should not be used with a value of zero because the reset location is normally programmed with a JMP instruction. Doing so will cause an improper fast interrupt.

5.3.3.3.2 CVR Reserved Bit 6

This bit is unused and read by the host processor as zero.

5.3.3.3.3 CVR Host Command Bit (HC) Bit 7

The HC bit is used by the host processor to handshake the execution of host command exceptions. Normally, the host processor sets HC=1 to request the host command exception from the DSP. When the host command exception is acknowledged by the DSP, the HC bit is cleared by the HI hardware. The host processor can read the state of HC to determine when the host command has been accepted. The host processor may elect to clear the HC bit, canceling the host command exception request at any time before it is accepted by the DSP CPU.

CAUTION

The command exception might be recognized by the DSP and executed before it can be canceled by the host, even if the host clears the HC bit.

Setting HC causes host command pending (HCP) to be set in the HSR. The host can write HC and HV in the same write cycle if desired. Hardware, software, individual, and STOP resets clear HC.

5.3.3.4 Interrupt Status Register (ISR)

The ISR is an 8-bit read-only status register used by the host processor to interrogate the status and flags of the HI. The host processor can write this address without affecting the internal state of the HI, which is useful if the user desires to access all of the HI registers by stepping through the HI addresses. The ISR can not be accessed by the DSP. The status bits are described in the following paragraphs.

5.3.3.4.1 ISR Receive Data Register Full (RXDF) Bit 0

The RXDF bit indicates that the receive byte registers (RXH, RXM, and RXL) contain data from the DSP CPU and may be read by the host processor. RXDF is set when the HTX is transferred to the receive byte registers. RXDF is cleared when the receive data low (RXL) register is read by the host processor. RXL is normally the last byte of the receive byte registers to be read by the host processor. RXDF can be cleared by the host processor using the initialize function. RXDF may be used to assert the external $\overline{\text{HREQ}}$ pin if the RREQ bit is set. Regardless of whether the RXDF interrupt is enabled, RXDF provides valid status so that polling techniques may be used by the host processor. Hardware, software, individual, and STOP resets clear RXDF.

5.3.3.4.2 ISR Transmit Data Register Empty (TXDE) Bit 1

The TXDE bit indicates that the transmit byte registers (TXH, TXM, and TXL) are empty and can be written by the host processor. TXDE is set when the transmit byte registers are transferred to the HRX register. TXDE is cleared when the transmit byte low (TXL) register is written by the host processor. TXL is normally the last byte of the transmit byte registers to be written by the host processor. TXDE can be set by the host processor using the initialize feature. TXDE may be used to assert the external $\overline{\text{HREQ}}$ pin if the TREQ bit is set. Regardless of whether the TXDE interrupt is enabled, TXDE provides valid status so that polling techniques may be used by the host processor. Hardware, software, individual, and STOP resets set TXDE.

5.3.3.4.3 ISR Transmitter Ready (TRDY) Bit 2

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

TRDY=TXDE • $\overline{\text{HRDF}}$

When TRDY is set to one, the data that the host processor writes to TXH,TXM, and TXL will be immediately transferred to the DSP CPU side of the HI. This has many applications. For example, if the host processor issues a host command which causes the DSP CPU to read the HRX, the host processor can be guaranteed that the data it just transferred to the HI is what is being received by the DSP CPU.

Hardware, software, individual, and STOP resets set TRDY.

5.3.3.4.4 ISR Host Flag 2 (HF2) Bit 3

The HF2 bit in the ISR indicates the state of host flag 2 in the HCR on the CPU side. HF2 can only be changed by the DSP (see Figure 5-10). HF2 is cleared by a hardware or software reset.

5.3.3.4.5 ISR Host Flag 3 (HF3) Bit 4

The HF3 bit in the ISR indicates the state of host flag 3 in the HCR on the CPU side. HF3 can only be changed by the DSP (see Figure 5-10). HF3 is cleared by a hardware or software reset.

5.3.3.4.6 ISR Reserved Bit 5

This status bit is reserved for future expansion and will read as zero during host processor read operations.

5.3.3.4.7 ISR DMA Status (DMA) Bit 6

The DMA status bit indicates that the host processor has enabled the DMA mode of the HI (HM1 or HM0=1). When the DMA status bit is clear, it indicates that the DMA mode is disabled (HM0=HM1=0) and no DMA operations are pending. When DMA is set, it indicates that the DMA mode is enabled and the host processor should not use the active DMA channel (RXH, RXM, RXL or TXH, TXM, TXL depending on DMA direction) to avoid conflicts with the DMA data transfers. The channel not in use can be used for polled operation by the host and operates in the interrupt mode for internal DSP exceptions or polling. Hardware, software, individual, and STOP resets clear the DMA status bit.

5.3.3.4.8 ISR Host Request (HREQ) Bit 7

The HREQ bit indicates the status of the external host request output pin ($\overline{\text{HREQ}}$). When the HREQ status bit is cleared, it indicates that the external $\overline{\text{HREQ}}$ pin is deasserted and no host processor interrupts or DMA transfers are being requested. When the HREQ status bit is set, it indicates that the external $\overline{\text{HREQ}}$ pin is asserted, indicating that the DSP is interrupting the host processor or that a DMA transfer request is occurring. The HREQ interrupt request may originate from either or both of two sources – the receive byte registers are full or the transmit byte registers are empty. These conditions are indicated by the ISR RXDF and TXDE status bits, respectively. If the interrupt source has been enabled by the associated request enable bit in the ICR, HREQ will be set if one or more of the two enabled interrupt sources is set. Hardware, software, individual, and STOP resets clear HREQ.

5.3.3.5 Interrupt Vector Register (IVR)

The IVR is an 8-bit read/write register which typically contains the exception vector number used with MC68000 Family processor vectored interrupts. Only the host processor can read and write this register. The contents of IVR are placed on the host data bus (H0–H7) when both the $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ pins are asserted and the DMA mode is disabled. The contents of this register are initialized to $0F by a hardware or software reset, which corresponds to the uninitialized exception vector in the MC68000 Family.

5.3.3.6 Receive Byte Registers (RXH, RXM, RXL)

The receive byte registers are viewed as three 8-bit read-only registers by the host processor. These registers are called receive high (RXH), receive middle (RXM), and receive low (RXL). These three registers receive data from the high byte, middle byte, and low byte, respectively, of the HTX register and are selected by three external host address inputs (HA2, HA1, and HA0) during a host processor read operation or by an on-chip address counter in DMA operations. The receive byte registers (at least RXL) contain valid data when the receive data register full (RXDF) bit is set. The host processor may program the RREQ bit to assert the external $\overline{\text{HREQ}}$ pin when RXDF is set. This informs the host processor or DMA controller that the receive byte registers are full. These registers may be read in any order to transfer 8-, 16-, or 24-bit data. However, reading RXL clears the receive data full RXDF bit. Because reading RXL clears the RXDF status bit, it is normally the last register read during a 16- or 24-bit data transfer. Reset does not affect RXH, RXM, or RXL.

5.3.3.7 Transmit Byte Registers (TXH, TXM, TXL)

The transmit byte registers are viewed as three 8-bit write-only registers by the host processor. These registers are called transmit high (TXH), transmit middle (TXM), and transmit low (TXL). These three registers send data to the high byte, middle byte and low byte, respectively, of the HRX register and are selected by three external host address inputs (HA2, HA1, and HA0) during a host processor write operation. Data may be written into the transmit byte registers when the transmit data register empty (TXDE) bit is set. The host processor may program the TREQ bit to assert the external $\overline{\text{HREQ}}$ pin when TXDE is set. This informs the host processor or DMA controller that the transmit byte registers are empty. These registers may be written in any order to transfer 8-, 16-, or 24-bit data. However, writing TXL clears the TXDE bit. Because writing the TXL register clears the TXDE status bit, TXL is normally the last register written during a 16- or 24-bit data transfer. The transmit byte registers are transferred as 24-bit data to the HRX register when both TXDE and the HRDF bit are cleared. This transfer operation sets TXDE and HRDF. Reset does not affect TXH, TXM, or TXL.

5.3.3.8 Registers After Reset

Table 5-5 shows the result of four kinds of reset on bits in each of the HI registers seen by the host processor. The hardware reset is caused by asserting the $\overline{\text{RESET}}$ pin; the software reset is caused by executing the RESET instruction; the individual reset is caused by clearing the PBC register bit 0; and the stop reset is caused by executing the STOP instruction.

5.3.4 Host Interface Pins

The 15 HI pins are described here for convenience. Additional information, including timing, is given in the *DSP56003/005 Advanced Information Data Sheet.*

5.3.4.1 Host Data Bus (H0-H7)

This bidirectional data bus transfers data between the host processor and the DSP56003/005. It acts as an input unless $\overline{\text{HEN}}$ is asserted and HR/$\overline{\text{W}}$ is high, making H0–H7 become outputs and allowing the host processor to read DSP56003/005 data. It is high impedance when $\overline{\text{HEN}}$ is deasserted. H0–H7 can be programmed as general-purpose I/O pins (PB0–PB7) when the host interface is not being used. These pins are configured as GPIO input pins during hardware reset.

5.3.4.2 Host Address (HA0–HA2)

These inputs provide the address selection for each host interface register. HA0–HA2 can be programmed as general-purpose I/O pins (PB8–PB10) when the host interface is not being used. These pins are configured as GPIO input pins during hardware reset.

Table 5-5 Host Registers after Reset (Host Side)

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	IR Reset	ST Reset
ICR	INIT	0	0	0	0
	HM (1 - 0)	0	0	0	0
	TREQ	0	0	0	0
	RREQ	0	0	0	0
	HF (1 - 0)	0	0	0	0
CVR	HC	0	0	0	0
	HV (5 - 0)	$12	$12	$12	$12
ISR	HREQ	0	0	0	0
	DMA	0	0	0	0
	HF (3 - 2)	0	0	—	—
	TRDY	1	1	1	1
	TXDE	1	1	1	1
	RXDF	0	0	0	0
IVR	IV (7 - 0)	$0F	$0F	—	—
RX	RXH (23 - 16)	—	—	—	—
	RXM (15 - 8)	—	—	—	—
	RXL (7 - 0)	—	—	—	—
TX	TXH (23 - 21)	—	—	—	—
	TXM (15 - 8)	—	—	—	—
	TXL (7 - 0)	—	—	—	—

5.3.4.3 Host Read/Write (HR/$\overline{\text{W}}$)

This input selects the direction of data transfer for each host processor access. If HR/$\overline{\text{W}}$is high and $\overline{\text{HEN}}$ is asserted, H0-H7 are outputs and DSP data is transferred to the host processor. If HR/$\overline{\text{W}}$ is low and $\overline{\text{HEN}}$ is asserted, H0-H7 are inputs and host data is transferred to the DSP. HR/$\overline{\text{W}}$ is stable when $\overline{\text{HEN}}$ is asserted. It can be programmed as a general-purpose I/O pin (PB11) when the host interface is not being used, and is configured as a GPIO input pin during hardware reset.

5.3.4.4 Host Enable ($\overline{\text{HEN}}$)

This input enables a data transfer on the host data bus. When $\overline{\text{HEN}}$is asserted and HR/$\overline{\text{W}}$ is high, H0–H7 become outputs and the host processor may read DSP56003/005 data. When $\overline{\text{HEN}}$ is asserted and HR/$\overline{\text{W}}$ is low, H0–H7 become inputs. When $\overline{\text{HEN}}$ is deasserted, host data is latched inside the DSP. Normally, a chip select signal derived from host address decoding and an enable clock are used to generate $\overline{\text{HEN}}$. $\overline{\text{HEN}}$ can be programmed as a general-purpose I/O pin (PB12) when the host interface is not being used, and is configured as a GPIO input pin during hardware reset.

5.3.4.5 Host Request ($\overline{\text{HREQ}}$)

This open-drain output signal is used by the DSP56003/005 HI to request service from the host processor, DMA controller, or a simple external controller. $\overline{\text{HREQ}}$may be connected to an interrupt request pin of a host processor, a transfer request of a DMA controller or a control input of external circuitry. $\overline{\text{HREQ}}$is asserted when an enabled request occurs in the host interface. $\overline{\text{HREQ}}$ is deasserted when the enabled request is cleared or masked, DMA $\overline{\text{HACK}}$is asserted, or the DSP is reset. $\overline{\text{HREQ}}$ may be programmed as a general purpose I/O pin (not open-drain) called PB13 when the HI is not being used.

Table 5-6 Port B Pin Definitions

BC0	BC1	Function
0	0	Parallel I/O (Reset Condition)
0	1	Host Interface
1	0	Host Interface ($\overline{\text{HACK}}$ is defined as general purpose I/O)
1	1	Reserved

5.3.4.6 Host Acknowledge (HACK)

The Port B Control register allows the user to program this input independently of the other Host Interface pins. When the port is defined for general purpose I/O, this input acts as a general purpose I/O pin called PB14. When the port is defined as the host interface, the user may manipulate the Port B Control register to program this input as either PB14, or as the HACK pin. The table below shows the Port B Control register bit configurations.

HACK may act as a data strobe for HI DMA data transfers (See Figure 5-18). Or, if HACKis used as an MC68000 host interrupt acknowledge, it enables the HI interrupt vector register (IVR) on the host data bus H0-H7 if HREQis asserted (See Figure 5-16). In this case, all other HI control pins are ignored and the state of the HI is not affected.

Note: HACK should always be pulled high when it is not in use.

5.3.5 Servicing the Host Interface

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

1. Polling
2. Interrupts, which can be either
   a. non-DMA
   b. DMA

From the host processor viewpoint, the service consists of making a data transfer since this is the only way to reset the appropriate status bits.

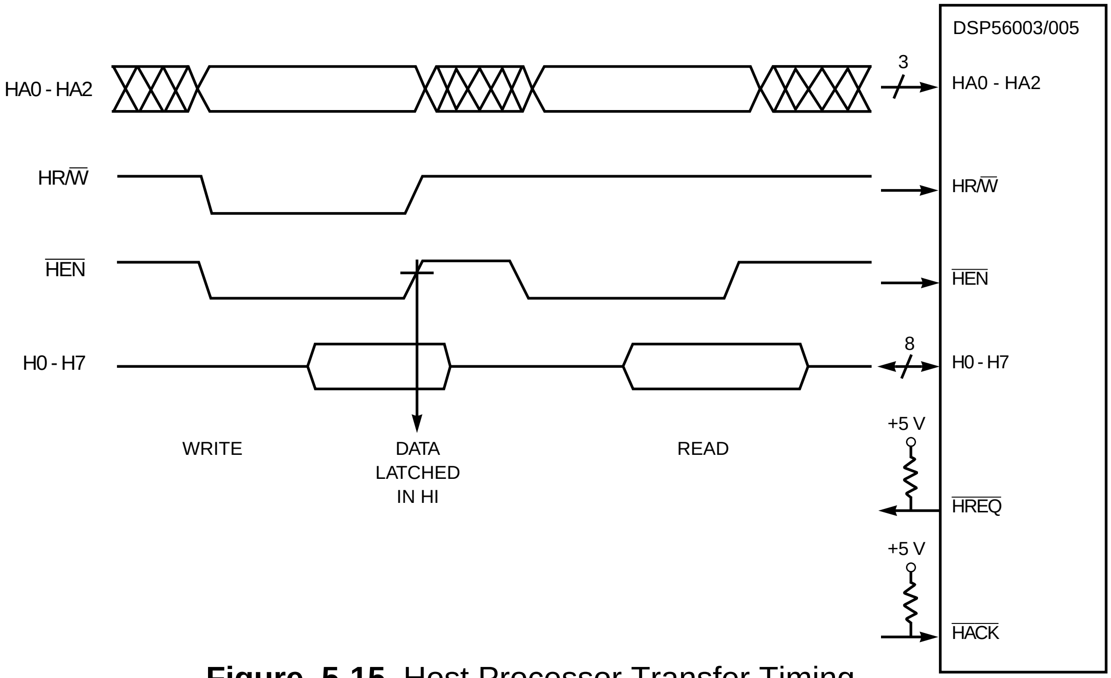


Figure 5-15 Host Processor Transfer Timing

5.3.5.1 HI Host Processor Data Transfer

The HI looks like static RAM to the host processor. Accordingly, in order to transfer data with the HI, the host processor:

1. asserts the HI address (HA0, HA1, HA2) to select the register to be read or written
2. asserts HR/$\overline{\text{W}}$ to select the direction of the data transfer
3. strobes the data transfer using $\overline{\text{HEN}}$. When data is being written to the HI by the host processor, the positive-going edge of $\overline{\text{HEN}}$ latches the data in the HI register selected. When data is being read by the host processor, the negative-going edge of $\overline{\text{HEN}}$ strobes the data onto the data bus H0-H7

Figure 5-15 illustrates this process. The specified timing relationships are given in the *DSP56003/005 Advanced Information Data Sheet*.

5.3.5.2 HI Interrupts Host Request ($\overline{\text{HREQ}}$)

The host processor interrupts are external and use the $\overline{\text{HREQ}}$ pin. $\overline{\text{HREQ}}$ is normally connected to the host processor maskable interrupt (IPL0, IPL1 or IPL2 in Figure 5-16) input. The host processor acknowledges host interrupts by executing an interrupt service routine.

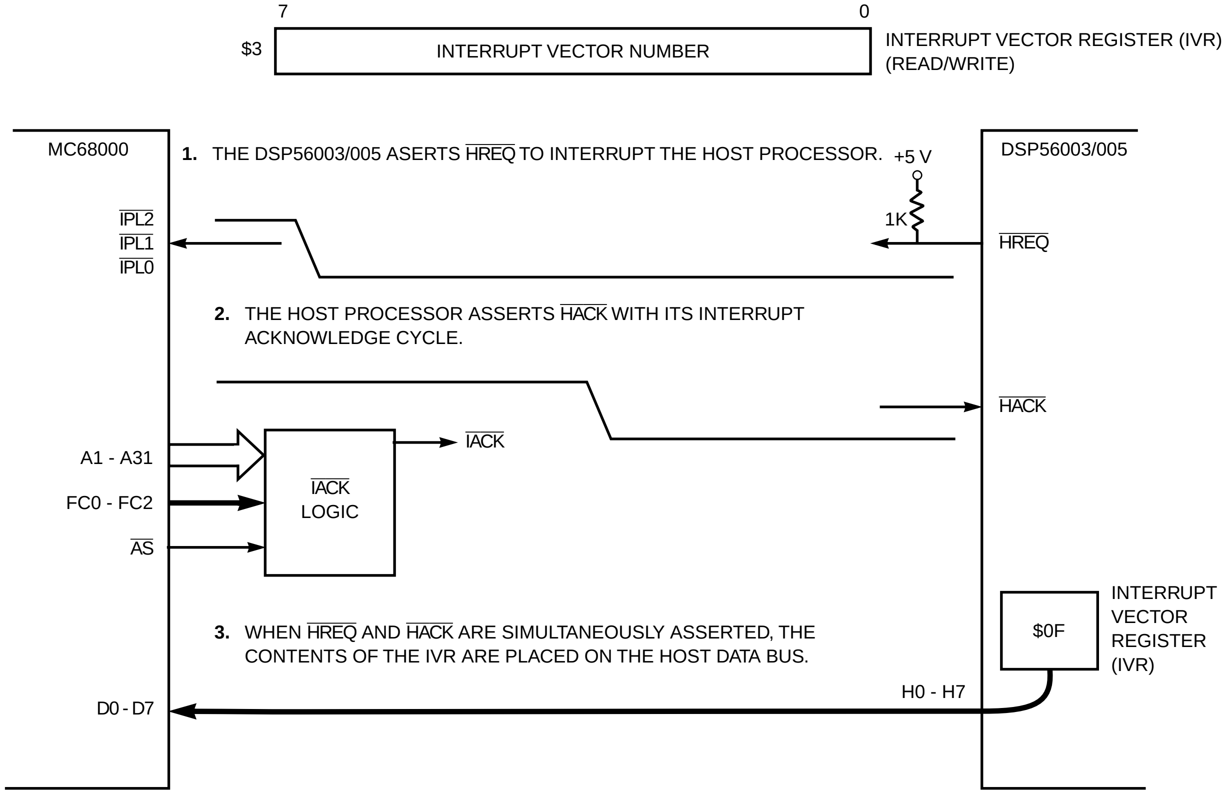

7
0
$3
INTERRUPT VECTOR NUMBER
INTERRUPT VECTOR REGISTER (IVR)
(READ/WRITE)
MC68000
DSP56003/005
1. THE DSP56003/005 ASERTS HREQ TO INTERRUPT THE HOST PROCESSOR.
+5 V
1K
IPL2
IPL1
IPL0
HREQ
2. THE HOST PROCESSOR ASSERTS HACK WITH ITS INTERRUPT ACKNOWLEDGE CYCLE.
HACK
IACK
A1 - A31
FC0 - FC2
AS
IACK LOGIC
3. WHEN HREQ AND HACK ARE SIMULTANEOUSLY ASSERTED, THE CONTENTS OF THE IVR ARE PLACED ON THE HOST DATA BUS.
$0F
INTERRUPT VECTOR REGISTER (IVR)
H0 - H7
D0 - D7

Figure 5-16 Interrupt Vector Register Read Timing

The most significant bit (HREQ) of the ISR may be tested by the host processor to determine if the DSP is the interrupting device and the two least significant bits (RXDF and TXDE) may be tested to determine the interrupt source (see Figure 5-17). The host processor interrupt service routine must read or write the appropriate HI register to clear the interrupt. $\overline{\text{HREQ}}$ is deasserted when1) the enabled request is cleared or masked, 2) DMA $\overline{\text{HACK}}$ is asserted, or 3) the DSP is reset.

5.3.5.3 Polling

In the polling mode of operation, the $\overline{\text{HREQ}}$ pin is not connected to the host processor and $\overline{\text{HACK}}$ must be deasserted to insure DMA data or IVR data is not being output on H0-H7 when other registers are being polled.

The host processor first performs a data read transfer to read the ISR (see Figure 5-17) to determine, whether:

1. RXDF=1, signifying the receive data register is full and therefore a data read should be performed
2. TXDE=1, signifying the transmit data register is empty so that a data write can be performed
3. TRDY=1, signifying the transmit data register is empty and that the receive data register on the DSP CPU side is also empty so that the data written by the host processor will be transferred directly to the DSP side
4. HF2 • HF3 ≠ 0, signifying an application-specific state within the DSP CPU has been reached, which requires action on the part of the host processor
5. DMA=1, signifying the HI is currently being used for DMA transfers. If DMA transfers are possible in the system, deactivate $\overline{\text{HACK}}$ prior to reading the ISR so both DMA data and the contents of ISR are not simultaneously output on H0- H7
6. If HREQ=1, the $\overline{\text{HREQ}}$ pin has been asserted, and one of the previous five conditions exists

Generally, after the appropriate data transfer has been made, the corresponding status bit will toggle.

If the host processor has issued a command to the DSP by writing the CVR and setting the HC bit, it can read the HC bit in the CVR to determine when the command has been accepted by the interrupt controller in the DSP's central processing module. When the command has been accepted for execution, the interrupt controller will reset the HC bit.

5.3.5.4 Servicing Non-DMA Interrupts

When HM0=HM1=0 (non-DMA) and $\overline{\text{HREQ}}$ is connected to the host processor interrupt input, the HI can request service from the host processor by asserting HREQ. In the non-DMA mode, $\overline{\text{HREQ}}$ will be asserted when TXDE=1 and/or RXDF=1 and the corresponding mask bit (TREQ or RREQ, respectively) is set. This is depicted in Figure 5-17.

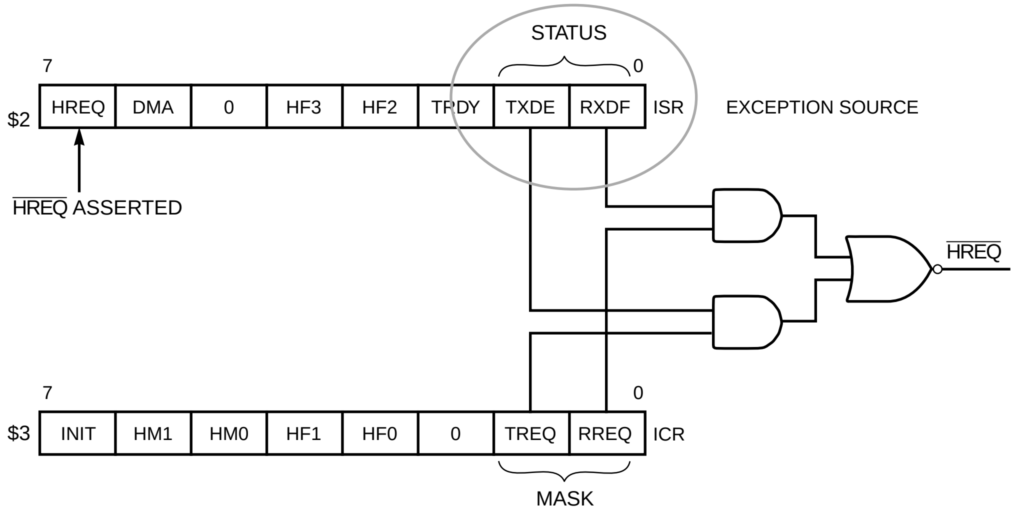


Figure 5-17 HI Interrupt Structure

Generally, servicing the interrupt starts with reading the ISR, as described in the previous section on polling, to determine which DSP has generated the interrupt and why. When multiple DSPs occur in a system, the HREQ bit in the ISR will normally be read first to determine the interrupting device. The host processor interrupt service routine must read or write the appropriate HI register to clear the interrupt. $\overline{\text{HREQ}}$ is deasserted when the enabled request is cleared or masked.

In the case where the host processor is a member of the MC680XX Family, servicing the interrupt will start by asserting $\overline{\text{HREQ}}$ to interrupt the processor (see Figure 5-17). The host processor then acknowledges the interrupt by asserting $\overline{\text{HACK}}$. While $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ are simultaneously asserted, the contents of the IVR are placed on the host data bus. This vector will tell the host processor which routine to use to service the $\overline{\text{HREQ}}$ interrupt.

The $\overline{\text{HREQ}}$ pin is an open-drain output pin so that it can be wire-ORed with the $\overline{\text{HREQ}}$ pins from other DSP56003/005 processors in the system. When the DSP56003/005 generates an interrupt request, the host processor can poll the HREQ bit in each of the ISRs to determine which device generated the interrupt.

5.3.5.5 Servicing DMA Interrupts

When HM0≠0 and/or HM1≠0, $\overline{\text{HREQ}}$ will be asserted to request a DMA transfer. Generally the $\overline{\text{HREQ}}$ pin will be connected to the $\overline{\text{REQ}}$ input of a DMA controller.

The HA0-2, $\overline{\text{HEN}}$, and HR/$\overline{\text{W}}$ pins are not used during DMA transfers; DMA transfers only use the $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ pins after the DMA channel has been initialized. $\overline{\text{HACK}}$ is used to strobe the data transfer as shown in Figure 5-18 where an MC68440 is used as the DMA controller. DMA transfers to and from the HI are considered in more detail in **Section 5.3.6 HI Application Examples**.

5.3.6 HI Application Examples

The following paragraphs describe examples of initializing the HI, transferring data with the HI, bootstrapping via the HI, and performing DMA transfers through the HI.

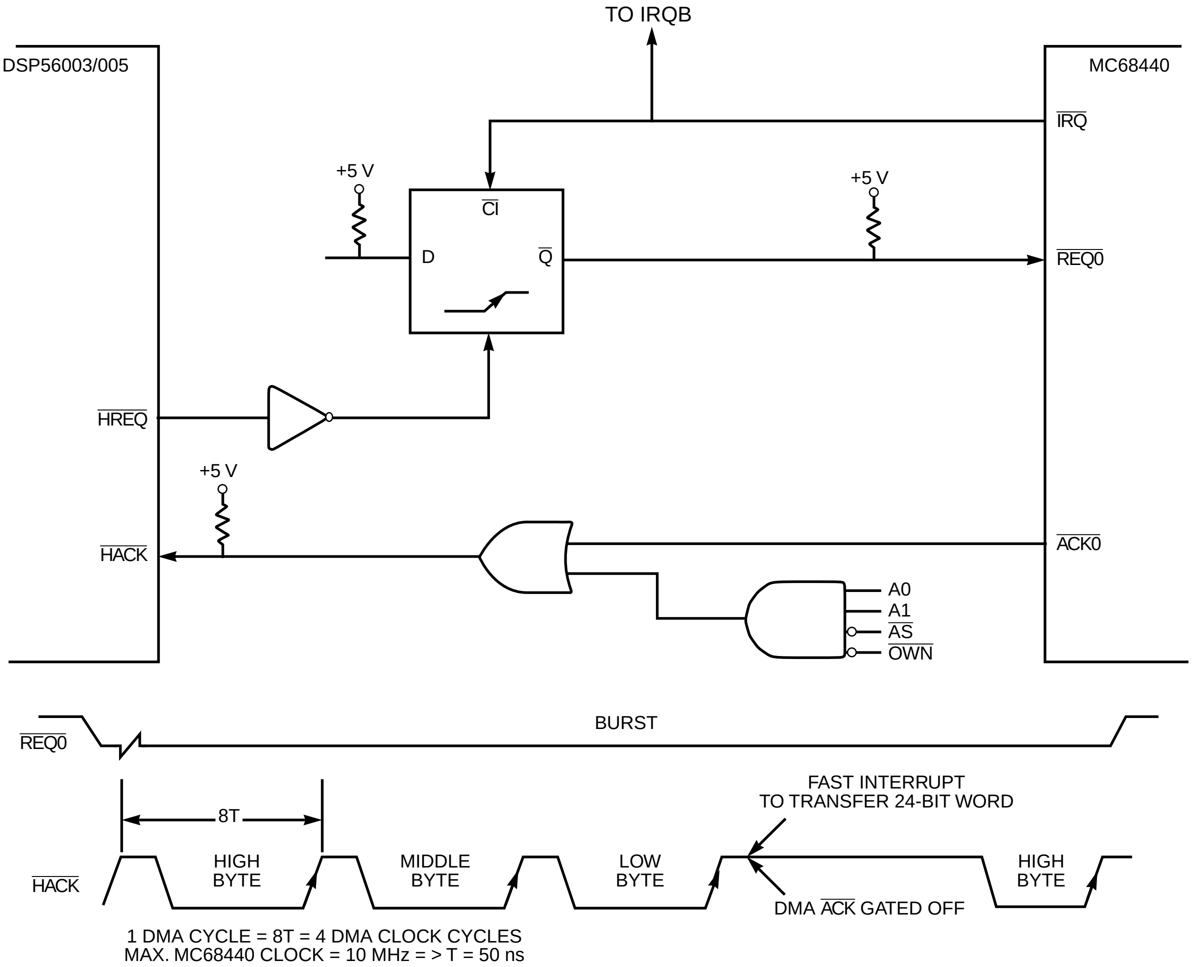
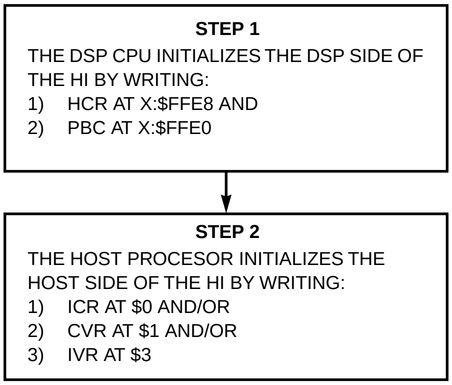


Figure 5-18 DMA Transfer Logic and Timing



STEP 1
THE DSP CPU INITIALIZES THE DSP SIDE OF THE HI BY WRITING:
1) HCR AT X:$FFE8 AND
2) PBC AT X:$FFE0

STEP 2
THE HOST PROCESOR INITIALIZES THE HOST SIDE OF THE HI BY WRITING:
1) ICR AT $0 AND/OR
2) CVR AT $1 AND/OR
3) IVR AT $3

Figure 5-19 HI Initialization Flowchart

5.3.6.1 HI Initialization

Initializing the HI takes two steps (see Figure 5-19). The first step is to initialize the DSP side of the HI, which requires that the options for interrupts and flags be selected and then the HI be selected (see Figure 5-20). The second step is for the host processor to clear the HC bit by writing the CVR, select the data transfer method - polling, interrupts, or DMA (see Figure 5-21 (d) and Figure 5-23), and write the IVR in the case of a MC680XX Family host processor. Figure 5-19 through Figure 5-22 provide a general description of how to initialize the HI. Later paragraphs in this section provide more detailed descriptions for specific examples. These subsections include some code fragments illustrating how to initialize and transfer data using the HI.

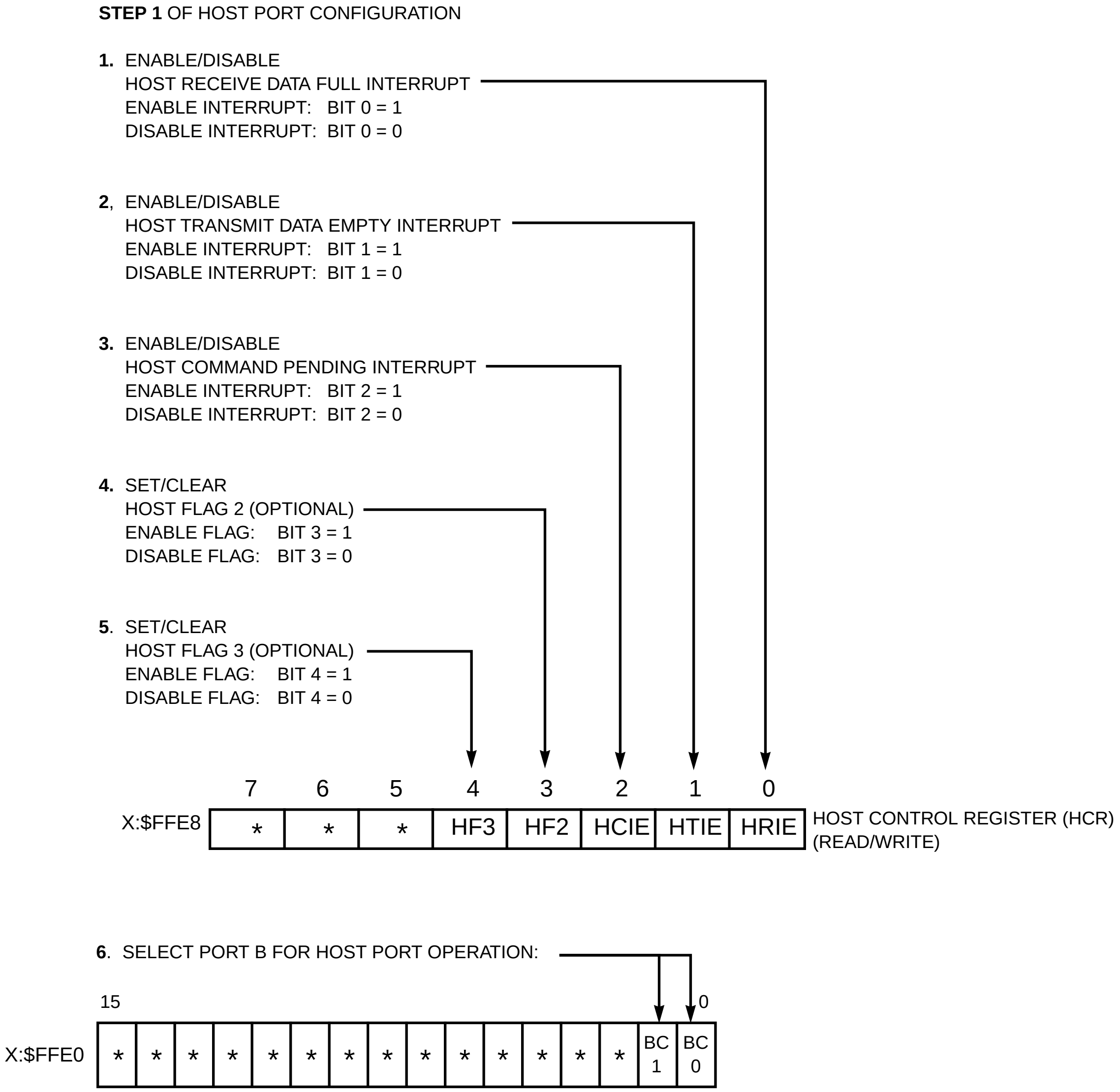
5.3.6.2 Polling/Interrupt Controlled Data Transfer

Handshake flags are provided for polled or interrupt-driven data transfers. Because the DSP interrupt response is sufficiently fast, most host microprocessors can load or store data at their maximum programmed I/O (non-DMA) instruction rate without testing the handshake flags for each transfer. If the full handshake is not needed, the host processor can treat the DSP as fast memory, and data can be transferred between the host and DSP at the fastest host processor rate. DMA hardware may be used with the external host request and host acknowledge pins to transfer data at the maximum DSP interrupt rate.

The basic data transfer process from the host processor's view (see Figure 5-15) is for the host to:

1. Assert $\overline{\text{HREQ}}$ when the HI is ready to transfer data
2. Assert $\overline{\text{HACK}}$ If the interface is using $\overline{\text{HACK}}$
3. Assert HR/$\overline{\text{W}}$ to select whether this operation will read or write a register
4. Assert the HI address (HA2, HA1, and HA0) to select the register to be read or written
5. Assert $\overline{\text{HEN}}$ to enable the HI
6. When $\overline{\text{HEN}}$ is deasserted, the data can be latched or read as appropriate if the timing requirements have been observed
7. $\overline{\text{HREQ}}$ will be deasserted if the operation is complete

The previous transfer description is an overview. Specific and exact information for the HI data transfers and their timing can be found in **Section 5.3.6.3 DMA Data Transfer** and in the *DSP56003/005 Data Sheet*.


STEP 1 OF HOST PORT CONFIGURATION
1. ENABLE/DISABLE
HOST RECEIVE DATA FULL INTERRUPT
ENABLE INTERRUPT: BIT 0 = 1
DISABLE INTERRUPT: BIT 0 = 0
2, ENABLE/DISABLE
HOST TRANSMIT DATA EMPTY INTERRUPT
ENABLE INTERRUPT: BIT 1 = 1
DISABLE INTERRUPT: BIT 1 = 0
3. ENABLE/DISABLE
HOST COMMAND PENDING INTERRUPT
ENABLE INTERRUPT: BIT 2 = 1
DISABLE INTERRUPT: BIT 2 = 0
4. SET/CLEAR
HOST FLAG 2 (OPTIONAL)
ENABLE FLAG: BIT 3 = 1
DISABLE FLAG: BIT 3 = 0
5. SET/CLEAR
HOST FLAG 3 (OPTIONAL)
ENABLE FLAG: BIT 4 = 1
DISABLE FLAG: BIT 4 = 0
7
6
5
4
3
2
1
0
X:$FFE8
*
*
*
HF3
HF2
HCIE
HTIE
HRIE
HOST CONTROL REGISTER (HCR)
(READ/WRITE)
6. SELECT PORT B FOR HOST PORT OPERATION:
15
0
X:$FFE0
* * * * * * * * * * * * * *
BC 1
BC 0

* Reserved; write as zero.

NOTE: The host flags are general-purpose semaphores. They are not required for host port operation but may be used in some applications.

Figure 5-20 HI Initialization — DSP Side

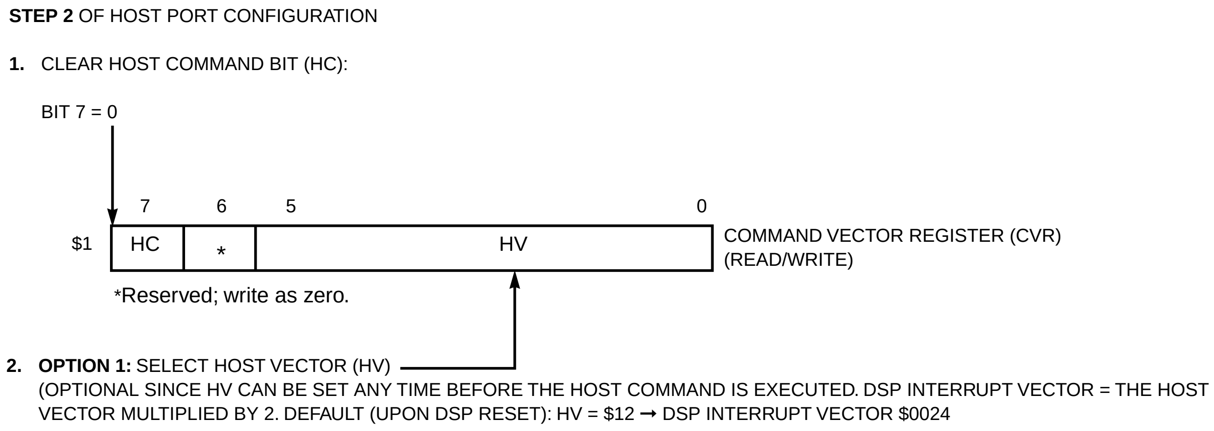

STEP 2 OF HOST PORT CONFIGURATION
1. CLEAR HOST COMMAND BIT (HC):
BIT 7 = 0
7
6
5
0
$1
HC
*
HV
COMMAND VECTOR REGISTER (CVR)
(READ/WRITE)
*Reserved; write as zero.
2. OPTION 1: SELECT HOST VECTOR (HV)
(OPTIONAL SINCE HV CAN BE SET ANY TIME BEFORE THE HOST COMMAND IS EXECUTED. DSP INTERRUPT VECTOR = THE HOST VECTOR MULTIPLIED BY 2. DEFAULT (UPON DSP RESET): HV = $12 → DSP INTERRUPT VECTOR $0024

Figure 5-21 (a) HI Configuration — Host Side

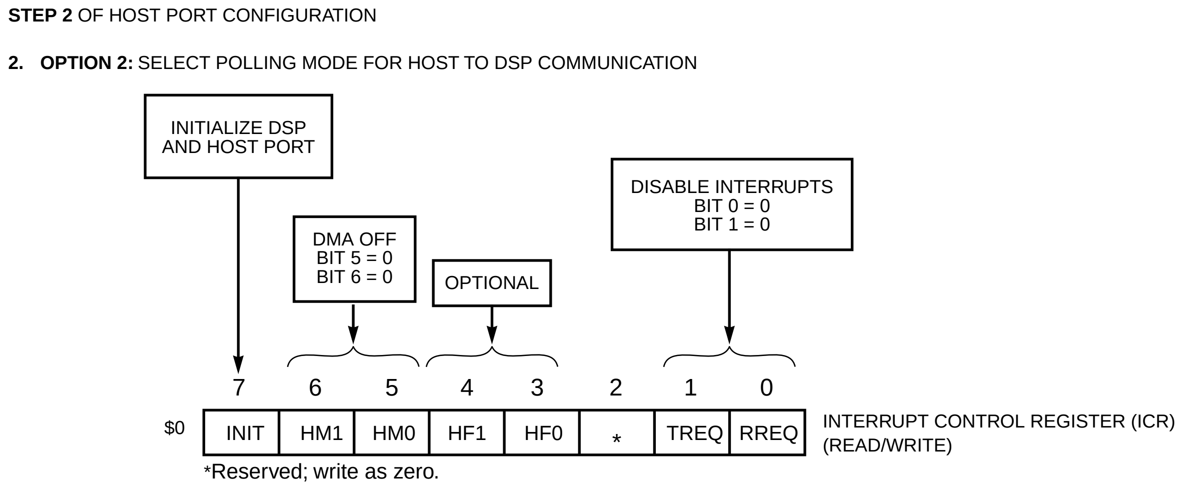
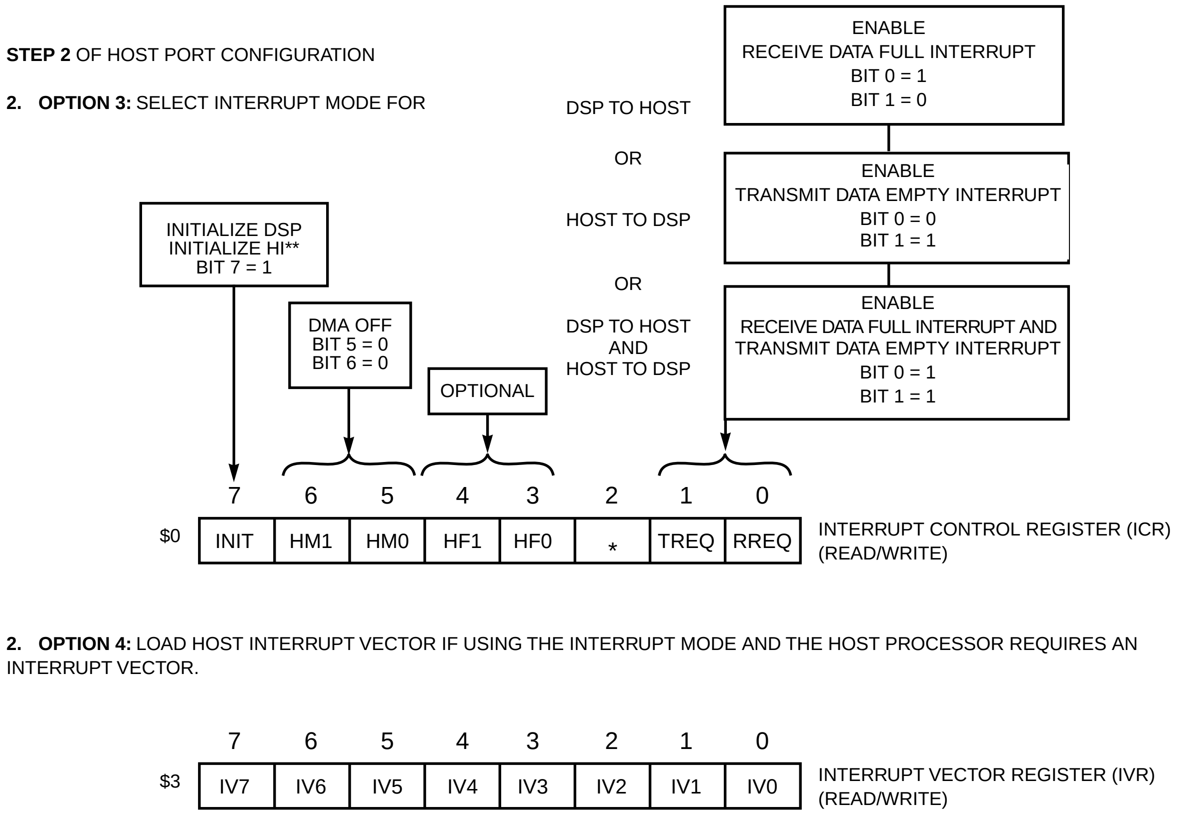

STEP 2 OF HOST PORT CONFIGURATION
2. OPTION 2: SELECT POLLING MODE FOR HOST TO DSP COMMUNICATION
INITIALIZE DSP AND HOST PORT
DMA OFF
BIT 5 = 0
BIT 6 = 0
OPTIONAL
DISABLE INTERRUPTS
BIT 0 = 0
BIT 1 = 0
7
6
5
4
3
2
1
0
$0
INIT
HM1
HM0
HF1
HF0
*
TREQ
RREQ
INTERRUPT CONTROL REGISTER (ICR)
(READ/WRITE)
*Reserved; write as zero.

Figure 5-21 (b) HI Initialization — Host Side, Polling Mode

5.3.6.2.1 Host to DSP — Data Transfer

Figure 5-23 shows the bits in the ISR and ICR registers used by the host processor and the bits in the HSR and HCR registers used by the DSP to transfer data from the host processor to the DSP. The registers shown are the status register and control register as they are seen by the host processor, and the status register and control register as they are seen by the DSP. Only the registers used to transmit data from the host processor to the DSP are described. Figure 5-24 illustrates the process of that data transfer. The steps in Figure 5-24 can be summarized as follows:


STEP 2 OF HOST PORT CONFIGURATION
2. OPTION 3: SELECT INTERRUPT MODE FOR
DSP TO HOST
OR
HOST TO DSP
OR
DSP TO HOST AND HOST TO DSP
ENABLE RECEIVE DATA FULL INTERRUPT BIT 0 = 1 BIT 1 = 0
ENABLE TRANSMIT DATA EMPTY INTERRUPT BIT 0 = 0 BIT 1 = 1
ENABLE RECEIVE DATA FULL INTERRUPT AND TRANSMIT DATA EMPTY INTERRUPT BIT 0 = 1 BIT 1 = 1
INITIALIZE DSP INITIALIZE HI** BIT 7 = 1
DMA OFF BIT 5 = 0 BIT 6 = 0
OPTIONAL
$0
INTERRUPT CONTROL REGISTER (ICR) (READ/WRITE)
2. OPTION 4: LOAD HOST INTERRUPT VECTOR IF USING THE INTERRUPT MODE AND THE HOST PROCESSOR REQUIRES AN INTERRUPT VECTOR.
$3
INTERRUPT VECTOR REGISTER (IVR) (READ/WRITE)

7	6	5	4	3	2	1	0
INIT	HM1	HM0	HF1	HF0	*	TREQ	RREQ

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

*Reserved; write as zero.

**See Figure 10 - 23.

Figure 5-21 (c) HI Initialization — Host Side, Interrupt Mode

1. When the TXDE bit in the ISR is set, it indicates that the HI is ready to receive a data byte from the host processor because the transmit byte registers (TXH, TXM, TXL) are empty.
2. The host processor can poll as shown in this step.
3. Alternatively, the host processor can use interrupts to determine the status of this bit. Setting the TREQ bit in the ICR causes the $\overline{\text{HREQ}}$ pin to interrupt the host processor when TXDE is set.
4. Once the TXDE bit is set, the host can write data to the HI. It does this by writing three bytes to TXH, TXM, and TXL, respectively, or two bytes to TXM and TXL, respectively, or one byte to TXL.
5. Writing data to TXL clears TXDE in the ISR.
6. From the DSP's viewpoint, the HRDF bit (when set) in the HSR indicates that data is waiting in the HI for the DSP.
7. When the DSP reads the HRX, the HRDF bit is automatically cleared and TXDE in the ISR is set.

8. When TXDE=0 and HRDF=0, data is automatically transferred from TBR to HRX which sets HRDF.
9. The DSP can poll HRDF to see when data has arrived, or it can use interrupts.
10. If HRIE (in the HCR) and HRDF are set, exception processing is started using interrupt vector P:$0020.

The MAIN PROGRAM initializes the HI and then hangs in a wait loop while it allows interrupts to transfer data from the host processor to the DSP. The first three MOVEP instructions enable the HI and configure the interrupts. The following MOVE enables the interrupts (this should always be done after the interrupt programs and hardware are completely initialized) and prepares the DSP CPU to look for the host flag, HF0=1. The JCLR instruction is a polling loop that looks for HF0=1, which indicates that the host processor is ready. When the host processor is ready to transfer data to the DSP, the DSP enables HRIE in the HCR, which allows the interrupt routine to receive data from the host processor. The jump-to-self instruction that follows is for test purposes only, it can be replaced by any other code in normal operation.

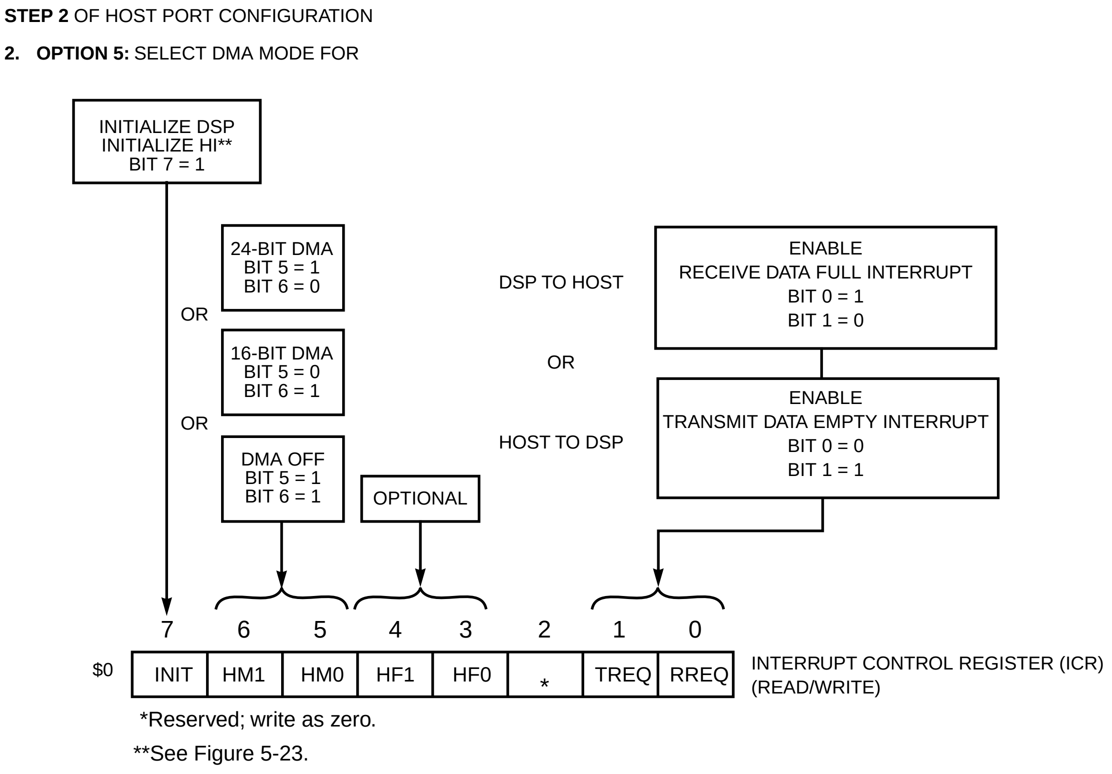
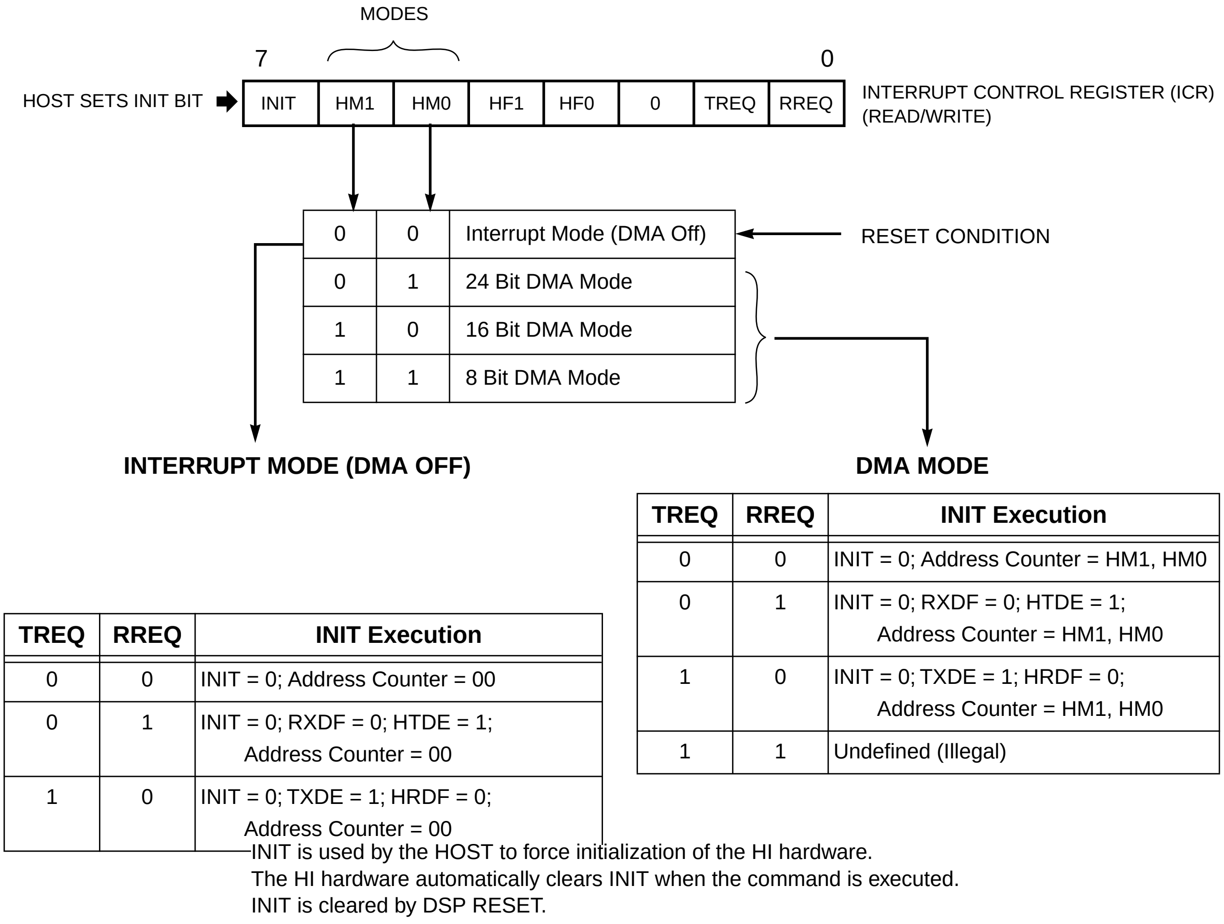


Figure 5-21 (d) HI Initialization — Host Side, DMA Mode



HM1	HM0	
0	0	Interrupt Mode (DMA Off)
0	1	24 Bit DMA Mode
1	0	16 Bit DMA Mode
1	1	8 Bit DMA Mode

INTERRUPT MODE (DMA OFF)

TREQ	RREQ	INIT Execution
0	0	INIT = 0; Address Counter = 00
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = 00
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = 00

DMA MODE

TREQ	RREQ	INIT Execution
0	0	INIT = 0; Address Counter = HM1, HM0
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = HM1, HM0
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = HM1, HM0
1	1	Undefined (Illegal)

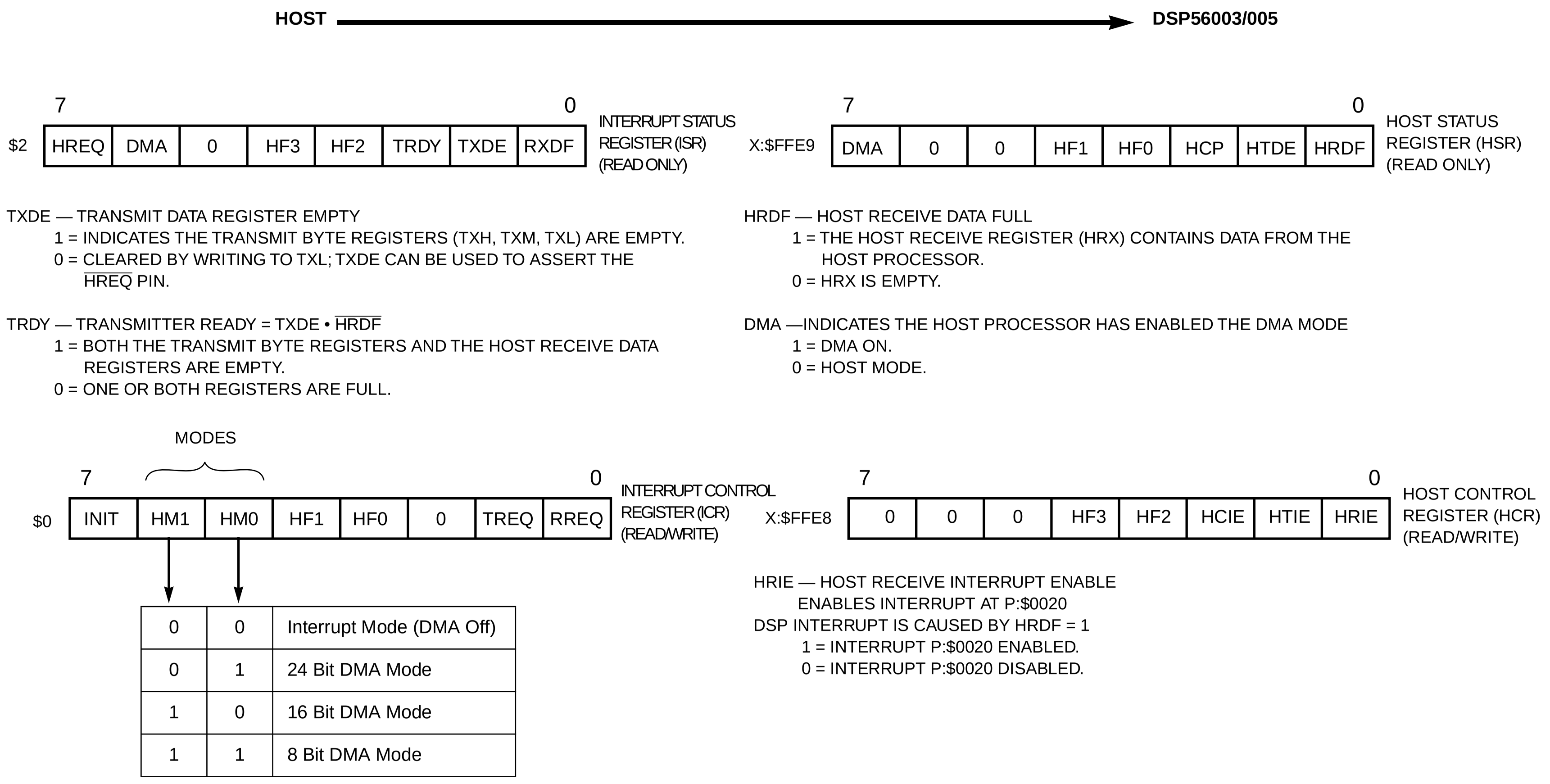
INIT is used by the HOST to force initialization of the HI hardware.
The HI hardware automatically clears INIT when the command is executed.
INIT is cleared by DSP RESET.

Figure 5-22 Host Mode and INIT Bits

The receive routine in Figure 5-26 was implemented as a long interrupt (the instruction at the interrupt vector location, which is not shown, is a JSR). Since there is only one instruction, this could have been implemented as a fast interrupt. The MOVEP instruction moves data from the HI to a buffer area in memory and increments the buffer pointer so that the next word received will be put in the next sequential location.

5.3.6.2.2 Host to DSP — Command Vector

The host processor can cause three types of interrupts in the DSP (see Figure 5-27). These are host receive data (P:$0020), host transmit data (P:$0022), and host command (P:$0024 - P:$007E). The host command (HC) can be used to control the DSP by forcing it to execute any of 45 subroutines that can be used to run tests, transfer data, process data, etc. In addition, the HC can cause any of the other 19 interrupt routines in the DSP to be executed.



TXDE — TRANSMIT DATA REGISTER EMPTY
1 = INDICATES THE TRANSMIT BYTE REGISTERS (TXH, TXM, TXL) ARE EMPTY.
0 = CLEARED BY WRITING TO TXL; TXDE CAN BE USED TO ASSERT THE $\overline{\text{HREQ}}$ PIN.

TRDY — TRANSMITTER READY = TXDE • $\overline{\text{HRDF}}$
1 = BOTH THE TRANSMIT BYTE REGISTERS AND THE HOST RECEIVE DATA REGISTERS ARE EMPTY.
0 = ONE OR BOTH REGISTERS ARE FULL.

HRDF — HOST RECEIVE DATA FULL
1 = THE HOST RECEIVE REGISTER (HRX) CONTAINS DATA FROM THE HOST PROCESSOR.
0 = HRX IS EMPTY.

DMA —INDICATES THE HOST PROCESSOR HAS ENABLED THE DMA MODE
1 = DMA ON.
0 = HOST MODE.

HM1	HM0	
0	0	Interrupt Mode (DMA Off)
0	1	24 Bit DMA Mode
1	0	16 Bit DMA Mode
1	1	8 Bit DMA Mode

HRIE — HOST RECEIVE INTERRUPT ENABLE
ENABLES INTERRUPT AT P:$0020
DSP INTERRUPT IS CAUSED BY HRDF = 1
1 = INTERRUPT P:$0020 ENABLED.
0 = INTERRUPT P:$0020 DISABLED.

TREQ — TRANSMIT REQUEST ENABLE
USED TO ENABLE INTERRUPTS THAT COME FROM TXDE TO THE HOST VIA THE $\overline{\text{HREQ}}$ PIN.
1 = TXDE INTERRUPTS PASS TO $\overline{\text{HREQ}}$.
0 = TXDE INTERRUPTS ARE MASKED.

Figure 5-23 Bits Used for Host-to-DSP Transfer

The process to execute a HC (see Figure 5-28) is as follows:

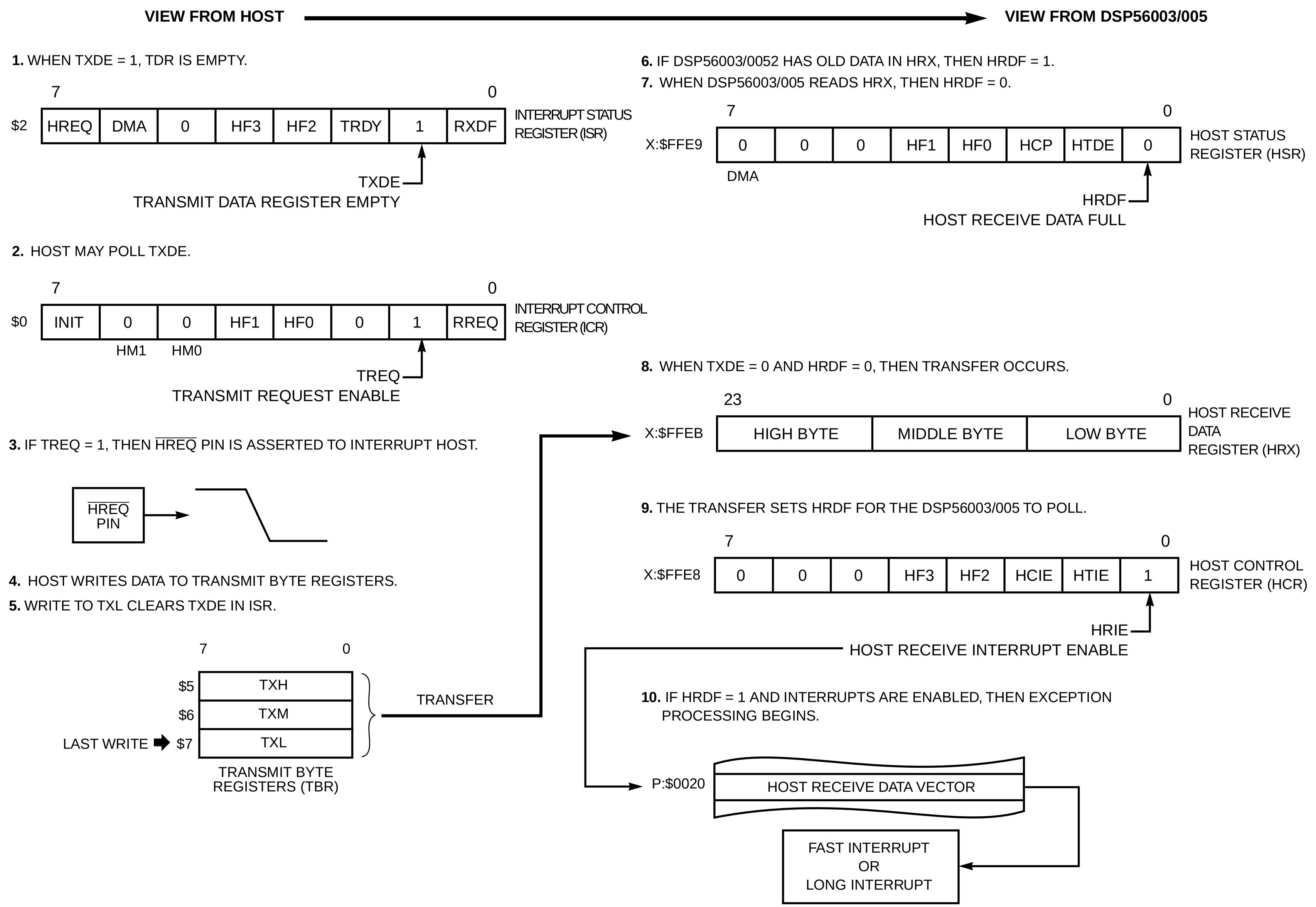
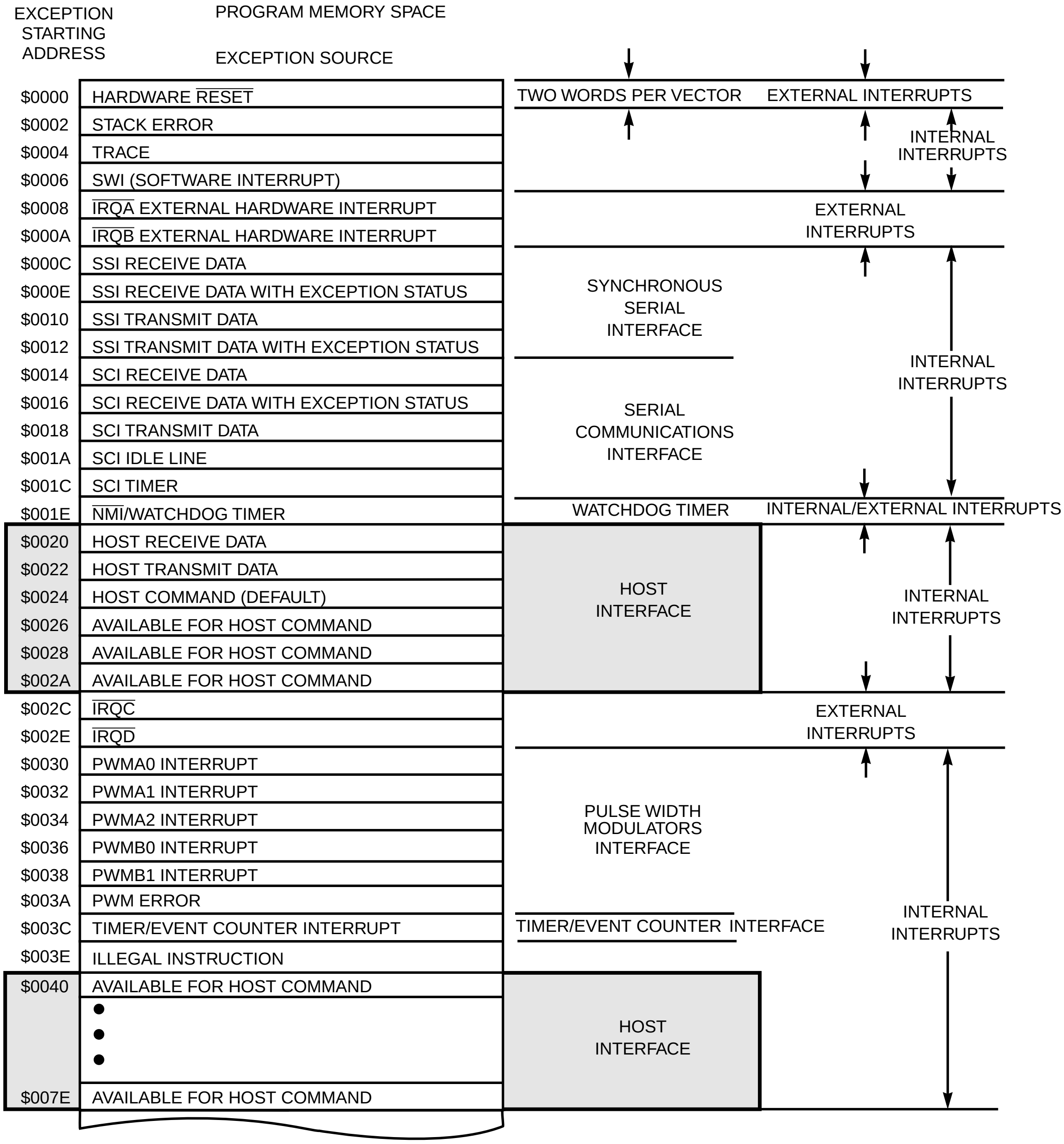

VIEW FROM HOST
VIEW FROM DSP56003/005
1. WHEN TXDE = 1, TDR IS EMPTY.
7
0
$2
HREQ
DMA
0
HF3
HF2
TRDY
1
RXDF
INTERRUPT STATUS REGISTER (ISR)
TXDE
TRANSMIT DATA REGISTER EMPTY
2. HOST MAY POLL TXDE.
7
0
$0
INIT
0
0
HF1
HF0
0
1
RREQ
INTERRUPT CONTROL REGISTER (ICR)
HM1
HM0
TREQ
TRANSMIT REQUEST ENABLE
3. IF TREQ = 1, THEN HREQ PIN IS ASSERTED TO INTERRUPT HOST.
HREQ PIN
4. HOST WRITES DATA TO TRANSMIT BYTE REGISTERS.
5. WRITE TO TXL CLEARS TXDE IN ISR.
7
0
$5
TXH
$6
TXM
LAST WRITE
$7
TXL
TRANSMIT BYTE REGISTERS (TBR)
TRANSFER
6. IF DSP56003/0052 HAS OLD DATA IN HRX, THEN HRDF = 1.
7. WHEN DSP56003/005 READS HRX, THEN HRDF = 0.
7
0
X:$FFE9
0
0
0
HF1
HF0
HCP
HTDE
0
HOST STATUS REGISTER (HSR)
DMA
HRDF
HOST RECEIVE DATA FULL
8. WHEN TXDE = 0 AND HRDF = 0, THEN TRANSFER OCCURS.
23
0
X:$FFEB
HIGH BYTE
MIDDLE BYTE
LOW BYTE
HOST RECEIVE DATA REGISTER (HRX)
9. THE TRANSFER SETS HRDF FOR THE DSP56003/005 TO POLL.
7
0
X:$FFE8
0
0
0
HF3
HF2
HCIE
HTIE
1
HOST CONTROL REGISTER (HCR)
HRIE
HOST RECEIVE INTERRUPT ENABLE
10. IF HRDF = 1 AND INTERRUPTS ARE ENABLED, THEN EXCEPTION PROCESSING BEGINS.
P:$0020
HOST RECEIVE DATA VECTOR
FAST INTERRUPT
OR
LONG INTERRUPT

Figure 5-24 Data Transfer from Host to DSP



EXCEPTION STARTING ADDRESS	PROGRAM MEMORY SPACE EXCEPTION SOURCE
$0000	HARDWARE $\overline{\text{RESET}}$
$0002	STACK ERROR
$0004	TRACE
$0006	SWI (SOFTWARE INTERRUPT)
$0008	$\overline{\text{IRQA}}$ EXTERNAL HARDWARE INTERRUPT
$000A	$\overline{\text{IRQB}}$ EXTERNAL HARDWARE INTERRUPT
$000C	SSI RECEIVE DATA
$000E	SSI RECEIVE DATA WITH EXCEPTION STATUS
$0010	SSI TRANSMIT DATA
$0012	SSI TRANSMIT DATA WITH EXCEPTION STATUS
$0014	SCI RECEIVE DATA
$0016	SCI RECEIVE DATA WITH EXCEPTION STATUS
$0018	SCI TRANSMIT DATA
$001A	SCI IDLE LINE
$001C	SCI TIMER
$001E	$\overline{\text{NMI}}$/WATCHDOG TIMER
$0020	HOST RECEIVE DATA
$0022	HOST TRANSMIT DATA
$0024	HOST COMMAND (DEFAULT)
$0026	AVAILABLE FOR HOST COMMAND
$0028	AVAILABLE FOR HOST COMMAND
$002A	AVAILABLE FOR HOST COMMAND
$002C	$\overline{\text{IRQC}}$
$002E	$\overline{\text{IRQD}}$
$0030	PWMA0 INTERRUPT
$0032	PWMA1 INTERRUPT
$0034	PWMA2 INTERRUPT
$0036	PWMB0 INTERRUPT
$0038	PWMB1 INTERRUPT
$003A	PWM ERROR
$003C	TIMER/EVENT COUNTER INTERRUPT
$003E	ILLEGAL INSTRUCTION
$0040	AVAILABLE FOR HOST COMMAND
● ● ●	
$007E	AVAILABLE FOR HOST COMMAND

Figure 5-27 HI Exception Vector Locations

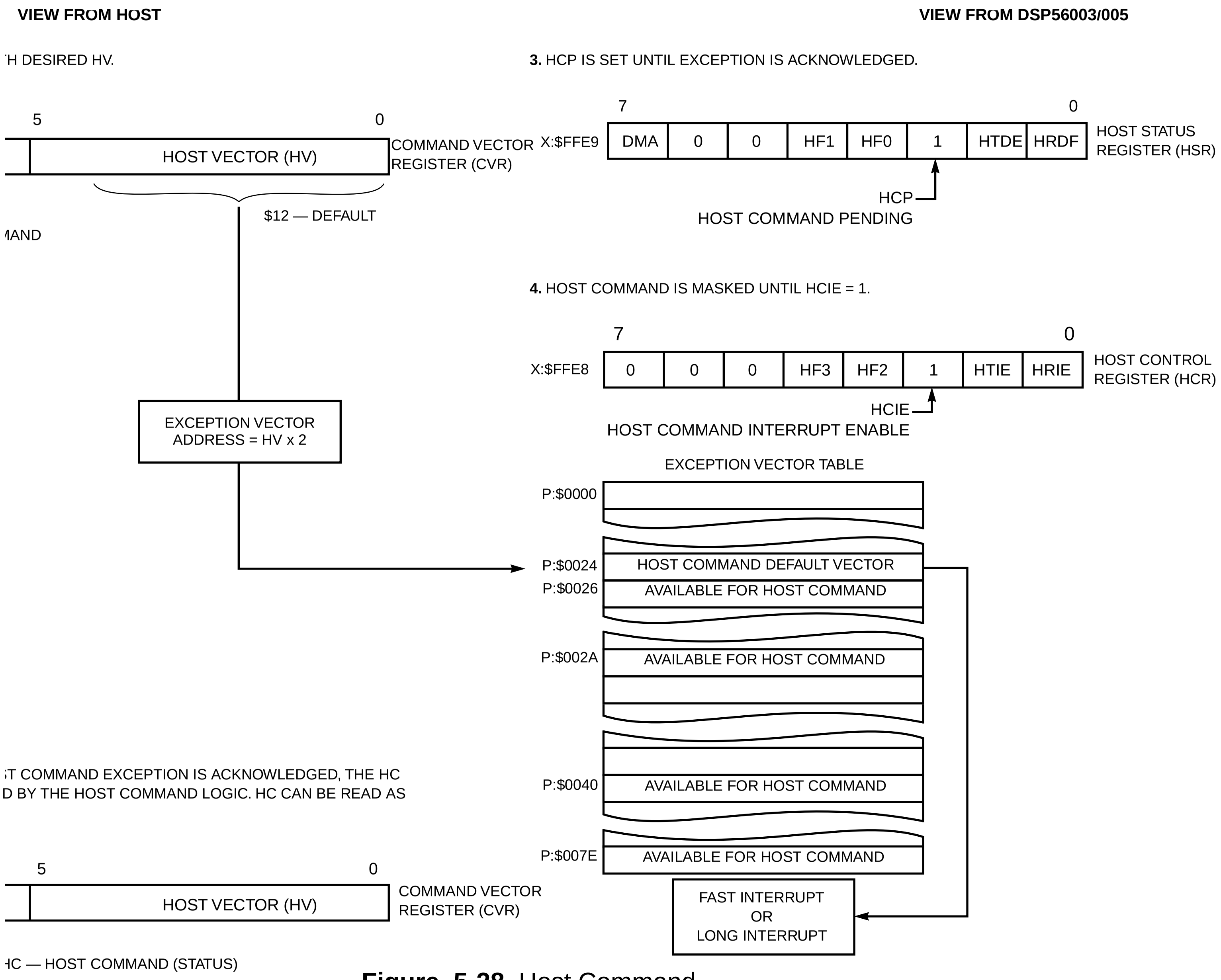


Figure 5-28 Host Command

1. The host processor writes the CVR with the desired HV (the HV is the DSP's interrupt vector (IV) location divided by two - i.e. if HV=$12, IV=$24).
2. The HC is then set.
3. The HCP bit in the HSR is set when HC is set.
4. If the HCIE bit in the HCR has been set by the DSP, the HC exception processing will start. The HV is multiplied by 2 and the result is used by the DSP as the interrupt vector.
5. When the HC exception is acknowledged, the HC bit (and therefore the HCP bit) is cleared by the HC logic. HC can be read by the host processor as a status bit to determine when the command is accepted. Similarly, the HCP bit can be read by the DSP CPU to determine if an HC is pending.

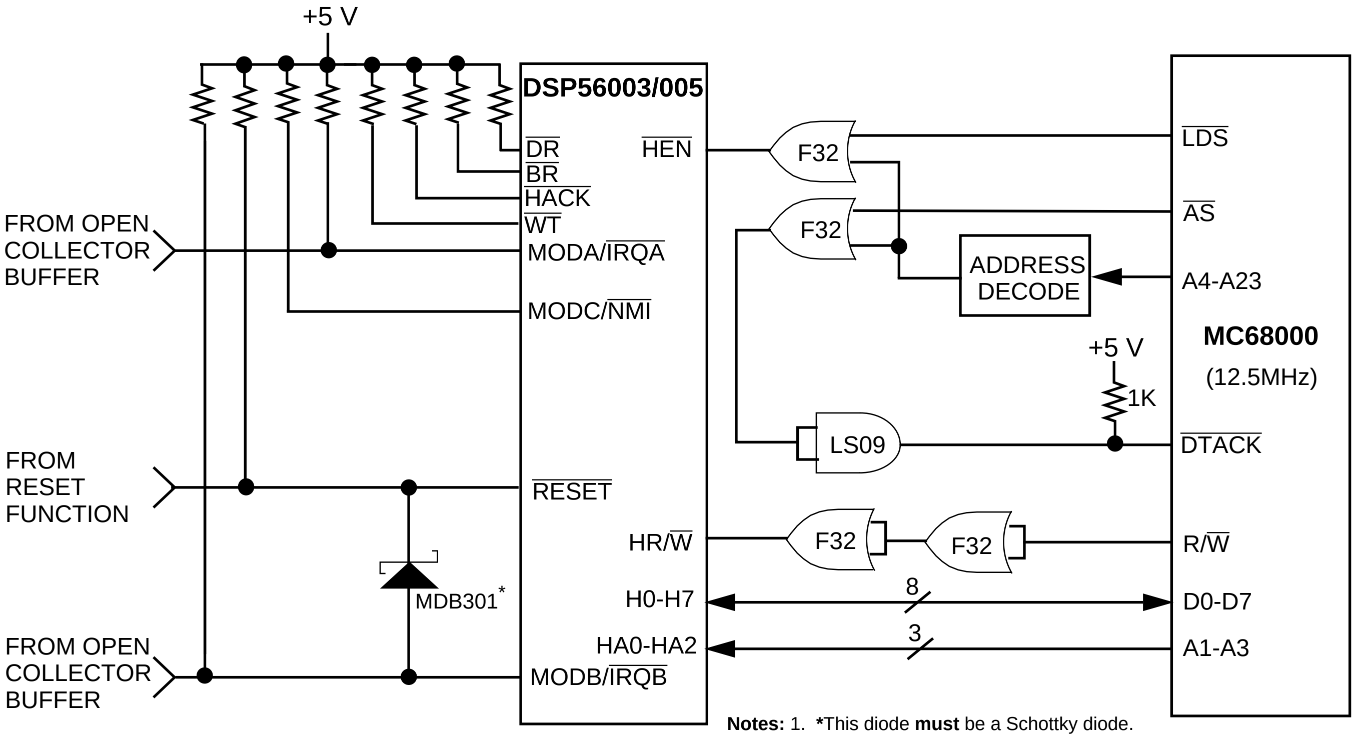
To guarantee a stable interrupt vector, write HV only when HC is clear. The HC bit and HV can be written simultaneously. The host processor can clear the HC bit to cancel a host command at any time before the DSP exception is accepted. Although the HV can be programmed to any exception vector, it is **not** recommended that HV=0 (RESET) be used because it does not reset the DSP hardware. DMA must be disabled to use the host exception.

```
;****************************************
; MAIN PROGRAM... receive data from host
;****************************************
     ORG     P:$40
     MOVE    #0,R0
     MOVE    #3,M0
     MOVEP   #1,X:PBC          ;Turn on Host Port
     MOVEP   #0,X:HCR          ;Turn off XMT and RCV interrupts
     MOVEP   #$0C00,X:IPR      ;Turn on host interrupt
     MOVE    #0,SR             ;Unmask interrupts
     JCLR    #3,X:HSR,*        ;Wait for HF0 (from host) set to 1
     MOVEP   #$1,X:HGR         ;Enable host receive interrupt
     JMP     *                 ;Now wait for interrupt
```

Figure 5-25 Receive Data from Host — Main Program

```
;************************************
; Receive from Host Interrupt Routine
;************************************
     RCV     MOVEP             X:HRX,X:(R0)+;Receive data.
     RTI
     END
```

Figure 5-26 Receive Data from Host Interrupt Routine



Notes: 1. ***This diode must** be a Schottky diode.
2. All resistors are 15KΩ unless noted otherwise.
3. When in RESET, $\overline{IRQA}$, $\overline{IRQB}$ and $\overline{NMI}$ must be deasserted by external peripherals.

	7			HOST				0
$0	INIT	HM1	HM0	HF1	HF0	0	TREQ	RREQ

INTERRUPT CONTROL REGISTER (ICR)
(READ/WRITE)

SETTING HF0 TERMINATES BOOTSTRAP LOADING AND STARTS EXECUTION AT LOCATION P:$0000.

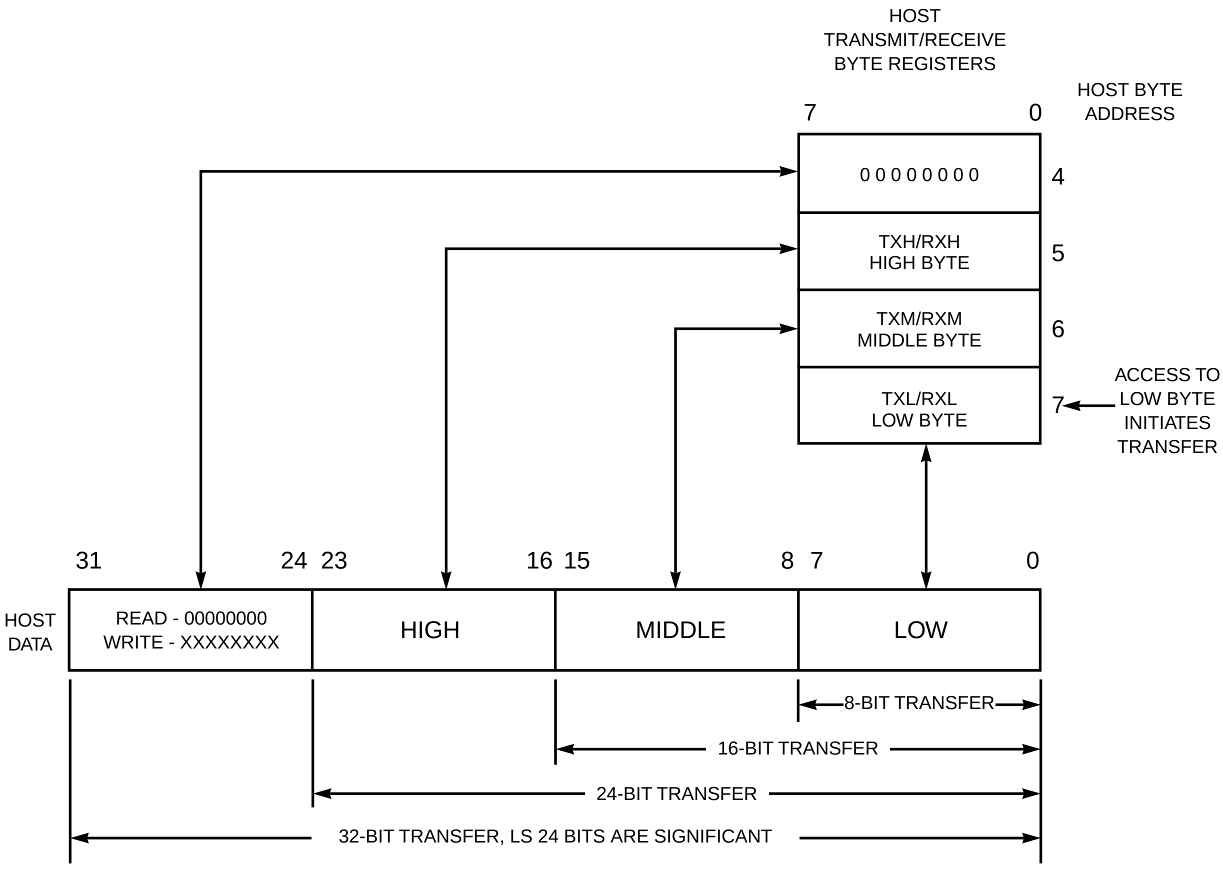
	HOST ADDRESS WRITTEN	CONTENTS LOADED TO INTERNAL P: RAM AT:
	4 (DUMMY)	
	5	P:$0000 HIGH BYTE
	6	P:$0000 MID BYTE
	7	P:$0000 LOW BYTE
	•	•
	•	•
SET HF0 FOR EARLY TERMINATION →	•	•
	4 (DUMMY)	•
	5	P:$01FF HIGH BYTE
	6	P:$01FF MID BYTE
	7	P:$01FF LOW BYTE

* Because the DSP56003/005 is so fast, host handshaking is generally not required.

Figure 5-29 Bootstrap Using the HI

5.3.6.2.3 Host to DSP — Bootstrap Loading Using the HI

The circuit shown in Figure 5-29 will cause the DSP to boot through the HI on power up. During the bootstrap program, the DSP looks at the MODC, MODB, and MODA bits. If the bits are set at 101 respectively, the DSP will load from the HI. Data is written by the host processor in a pattern of four bytes, with the high byte being a dummy and the low byte being the low byte of the DSP word (see Figure 5-29 and Figure 5-30). Figure 5-30 shows how an 8-,16-, 24-, or 32-bit word in the host processor maps into the HI registers. The HI register at address $4 is not used and will read as zero. It is not necessary to use address $4, but since many host processors are 16- or 32-bit processors, address $4 will often be used as part of the 16- or 32-bit word. The low order byte (at $7) should always be written last since writing to it causes the HI to initiate the transfer of the word to the HRX. Data is then transferred from the HRX to the DSP program memory. If the host processor needs to terminate the bootstrap loading before 4608 words have been down loaded, it can set the HF0 bit in the ICR. The DSP will then terminate the down load and start executing at location P:$0000. Since the DSP56003/005 is typically faster than the host processor, hand shaking during the data transfer is normally not required.



NOTE: Access low byte last

Figure 5-30 Transmit/Receive Byte Registers

```
; This is the routine that loads from the Host Interface.
; MC:MB:MA=100 - reserved
; MC:MB:MA=101 - Host

HOSTLD    BSET #0,X:PBC         ; Configure Port B as Host
          DO B1,_LOOP3          ; Load P_SIZE instruction words
_LBLA     JCLR #3,X:HSR,_LBLB   ; if HF0=1, stop loading data.
          ENDDO                 ; Must terminate the do loop
          JMP <_LOOP3

_LBLB     JCLR #0,X:HSR,_LBLA   ; Wait for HRDF to go high
                                ;(meaning data is present).
          MOVEP X:HRX,P:(R0)+   ; Store 24-bit data in P memory
_LOOP3                          ; and go get another 24-bit word.
                                ; finish bootstrap
FINISH    MOVE #<0,R1
```

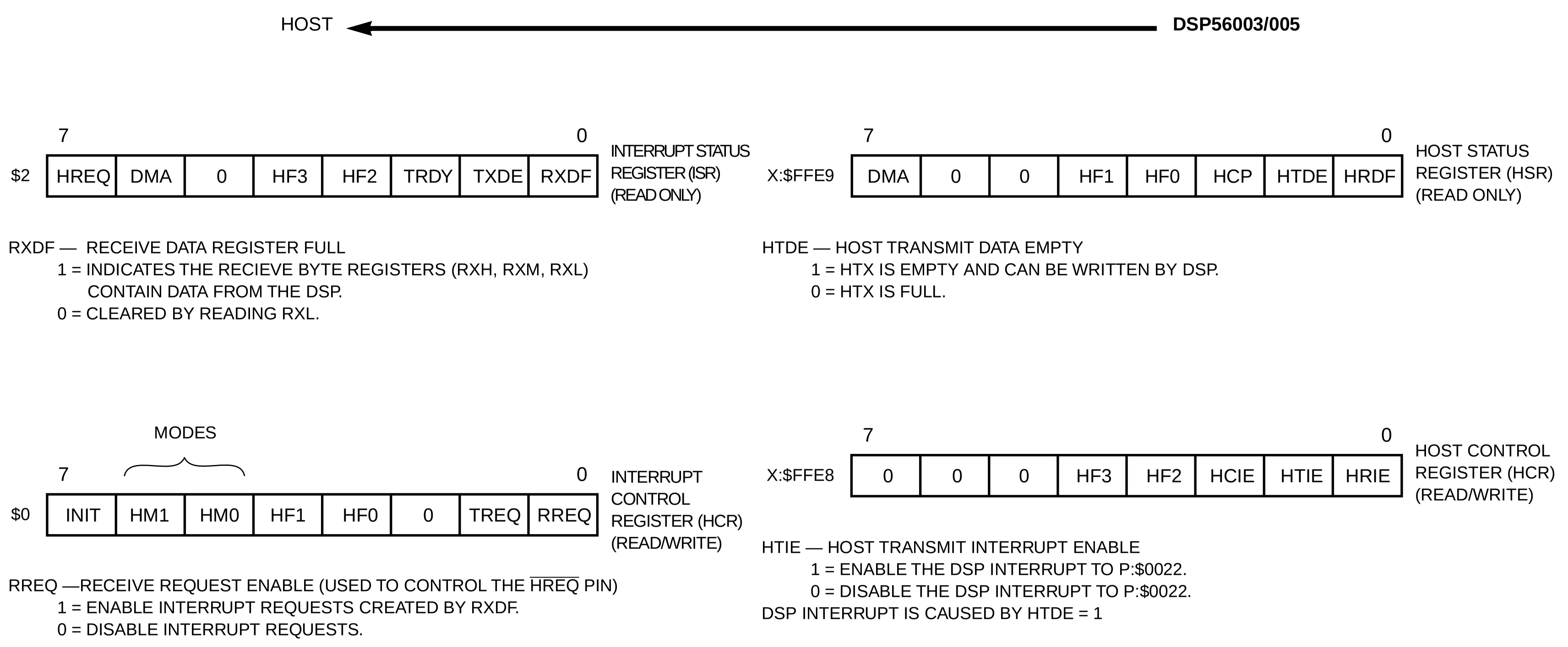
Figure 5-31 Bootstrap Code Fragment

The actual code used in the bootstrap program is given in **APPENDIX A**. The portion of the code that loads from the HI is shown in Figure 5-31. The BSET instruction configures Port B as the HI and the first JCLR looks for a flag (HF0) to indicate an early termination of the download. The second JCLR instruction causes the DSP to wait for a complete word to be received and then a MOVEP moves the data from the HI to memory.

5.3.6.2.4 DSP to Host Data Transfer

Data is transferred from the DSP to the host processor in a similar manner as from the host processor to the DSP. Figure 5-32 shows the bits in the status registers (ISR and HSR) and control registers (ICR and HCR) used by the host processor and DSP CPU, respectively. The DSP CPU (see Figure 5-33) can poll the HTDE bit in the HSR (1) to see when it can send data to the host, or it can use interrupts enabled by the HTIE bit in the HCR (2). If HTIE=1 and interrupts are enabled, exception processing begins at interrupt vector P:$0022 (3). The interrupt routine should write data to the HTX (4), which will clear HTDE in the HSR. From the host's viewpoint, (5) reading the RXL clears RXDF in the ISR. When RXDF=0 and HTDE=0 (6) the contents of the HTX will be transferred to the receive byte registers (RXH:RXM:RXL). This transfer sets RXDF in the ISR (7), which the host processor can poll to see if data is available or, if the RREQ bit in the ICR is set, the HI will interrupt the host processor with $\overline{\text{HREQ}}$ (8).

The code shown in Figure 5-34 is essentially the same as the MAIN PROGRAM in Figure 5-25 except that, since this code will transmit instead of receive data, the HTIE bit is set in the HCR instead of the HRIE bit.



RXDF — RECEIVE DATA REGISTER FULL
1 = INDICATES THE RECIEVE BYTE REGISTERS (RXH, RXM, RXL) CONTAIN DATA FROM THE DSP.
0 = CLEARED BY READING RXL.

HTDE — HOST TRANSMIT DATA EMPTY
1 = HTX IS EMPTY AND CAN BE WRITTEN BY DSP.
0 = HTX IS FULL.

RREQ —RECEIVE REQUEST ENABLE (USED TO CONTROL THE $\overline{\text{HREQ}}$ PIN)
1 = ENABLE INTERRUPT REQUESTS CREATED BY RXDF.
0 = DISABLE INTERRUPT REQUESTS.

HTIE — HOST TRANSMIT INTERRUPT ENABLE
1 = ENABLE THE DSP INTERRUPT TO P:$0022.
0 = DISABLE THE DSP INTERRUPT TO P:$0022.
DSP INTERRUPT IS CAUSED BY HTDE = 1

Figure 5-32 Bits Used for DSP to Host Transfer

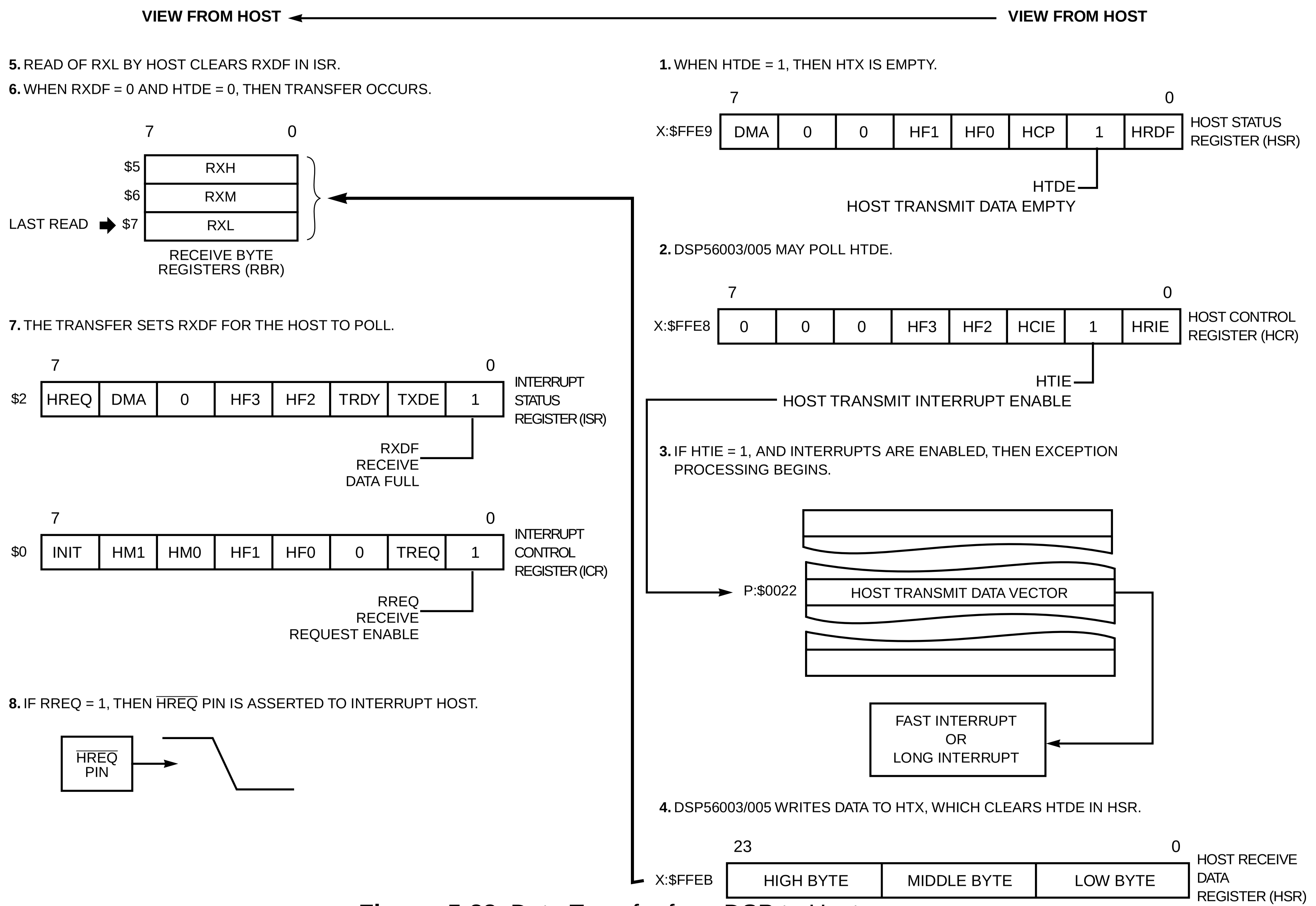


Figure 5-33 Data Transfer from DSP to Host

```
;***************************************
; MAIN PROGRAM... transmit 24-bit data to host
;***************************************
ORG      P:$40
MOVEP    #1,X:PBC      ;Turn on Host Port
MOVEP    #$0C00,X:IPR  ;Turn on host interrupt
MOVEP    #0,X:HCR      ;Turn off XMT and RCV interrupts
MOVE     #0,SR         ;Unmask interrupts
JCLR     #3,X:HSR,*    ;Wait for HF0 (from host) set to 1
AND      X0,A
JEQ      LOOP
MOVEP    #$2,X:HCR     ;Enable host transmit interrupt
JMP      *             ;Now wait for interrupt
```

Figure 5-34 Main Program — Transmit 24-Bit Data to Host

```
;**********************************
;TRANSMIT to Host Interrupt Routine
;***********************************
XMT      MOVEP         #$123456,X:HTX;Test value to transmit
         MOVEP         #0,X:HCR;Turn off XMT Interrupt
RTI
END
```

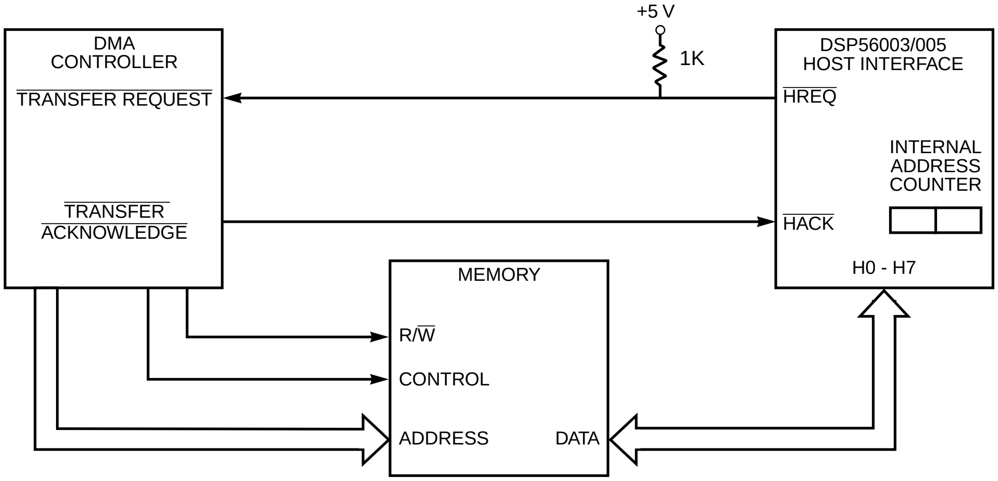
Figure 5-35 Transmit to HI Routine

The transmit routine used by the code in Figure 5-34 is shown in Figure 5-35. The interrupt vector contains a JSR, which makes it a long interrupt. The code sends a fixed test pattern ($123456) and then resets the HI for the next interrupt.

5.3.6.3 DMA Data Transfer

The DMA mode allows the transfer of 8-, 16- or 24-bit data through the DSP HI under the control of an external DMA controller. The HI provides the pipeline data registers and the synchronization logic between the two asynchronous processor systems. The DSP host exceptions provide cycle-stealing data transfers with the DSP internal or external memory. This technique allows the DSP memory address to be generated using any of the DSP addressing modes and modifiers. Queues and circular sample buffers are easily created for DMA transfer regions. The host exceptions can be programmed as high priority fast or long exception service routines. The external DMA controller provides the transfers between the DSP HI registers and the external DMA memory. The external DMA controller must provide the address to the external DMA memory; however, the address of the selected HI register is provided by a DMA address counter in the HI.

DMA transfers can only be in one direction at a time; however, the host processor can access any of the registers not in use during the DMA transfer by deasserting $\overline{\text{HACK}}$and using $\overline{\text{HEN}}$ and HA0-HA2 to transfer data. The host can therefore transfer data in the other direction during the DMA operation using polling techniques.



Characteristics of Host DMA Mode

- The $\overline{\text{HREQ}}$ pin is **NOT** available for host processor interrupts.

- TREQ and RREQ select the direction of DMA transfer.
  - —DMA to DSP56003/005
  - —DSP56003/005 to DMA
  - —Simultaneous bidirectional DMA transfers are not permitted.

- Host processor software polled transfers are permitted in the opposite direction of the DMA transfer.

- 8-, 16-, or 24-bit transfers are supported.
- 16-, or 24-bit transfers reduce the DSP interrupt rate by a factor of 2 or 3, respectively.

Figure 5-36 HI Hardware — DMA Mode

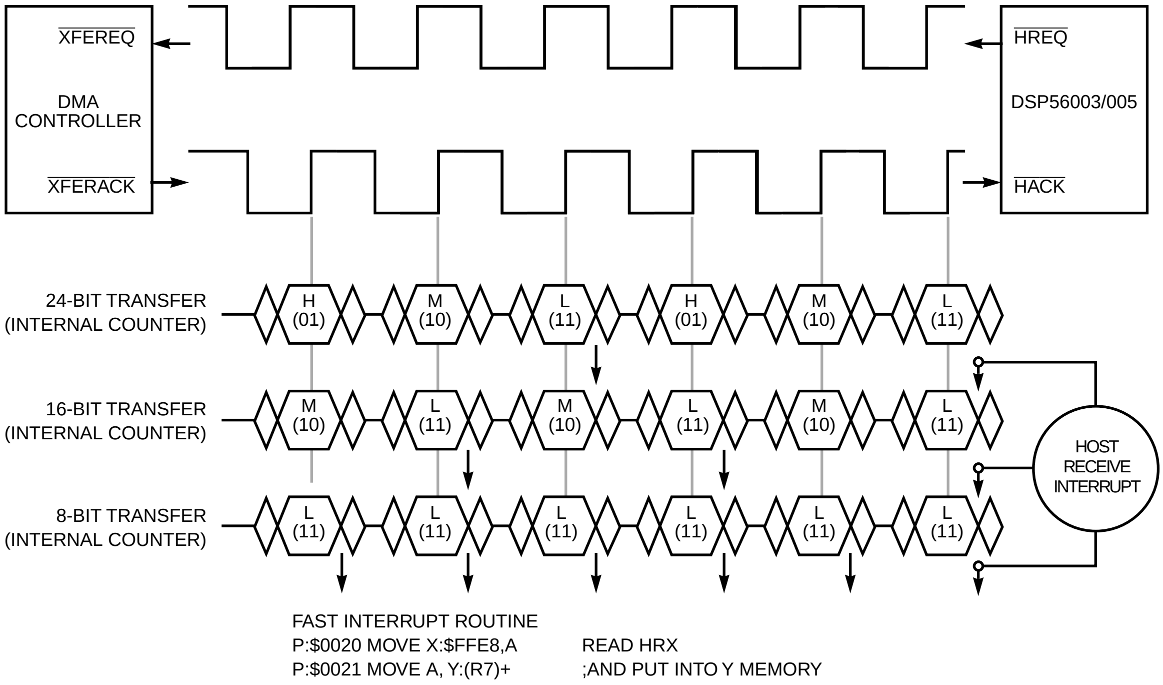
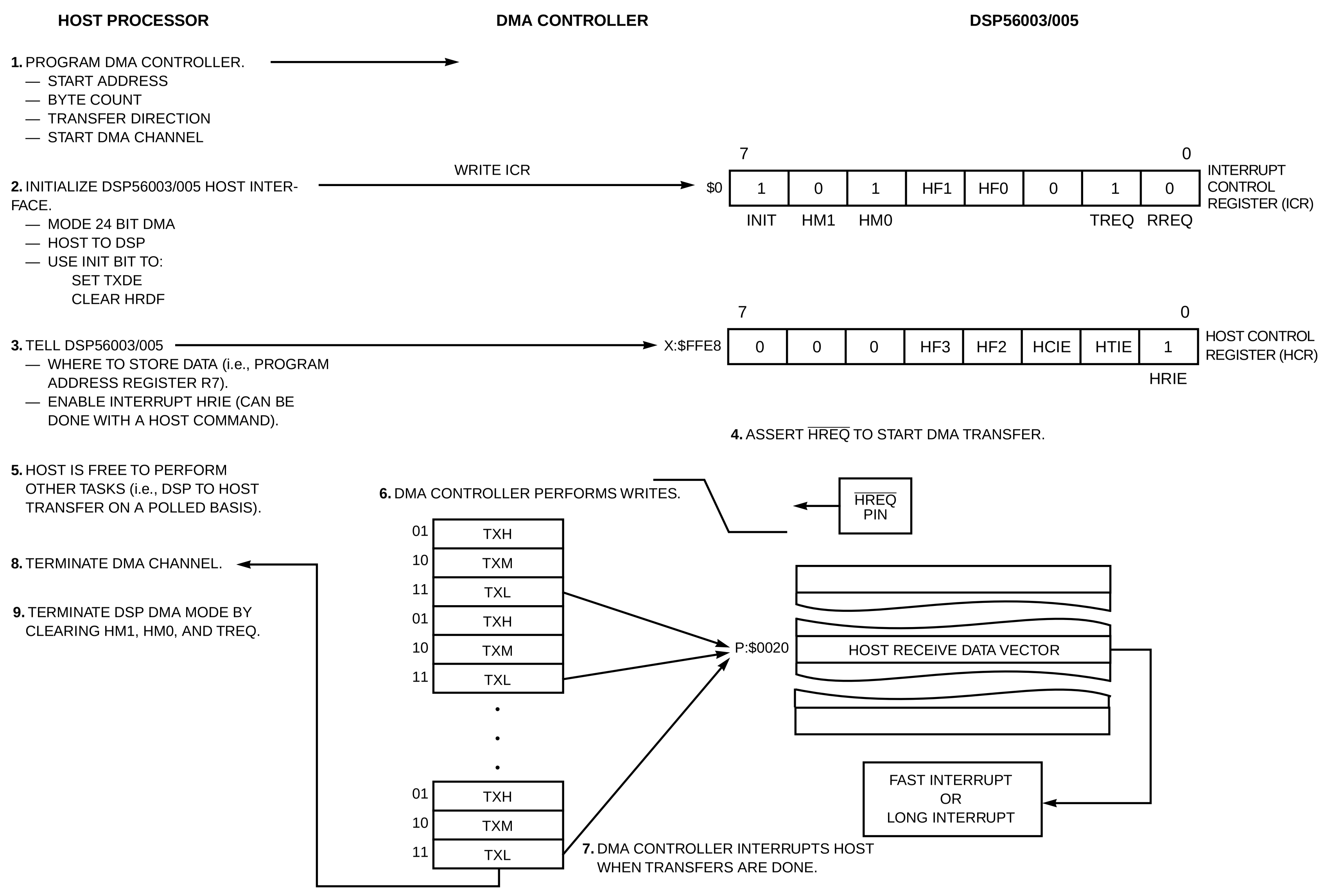


Figure 5-37 DMA Transfer and Host Interrupts

5.3.6.3.1 Host To DSP Internal Processing

The following procedure outlines the steps that the HI hardware takes to transfer DMA data from the host data bus to DSP memory (see Figure 5-36 and Figure 5-37).

1. HI asserts the $\overline{\text{HREQ}}$ pin when TXDE=1.
2. DMA controller enables data on H0-H7 and asserts $\overline{\text{HACK}}$.
3. When $\overline{\text{HACK}}$ is asserted, the HI deasserts $\overline{\text{HREQ}}$.
4. When the DMA controller deasserts $\overline{\text{HACK}}$, the data on H0-H7 is latched into the TXH, TXM, TXL registers.
5. If the byte register written was not TXL (i.e., not $7) the DMA address counter internal to the HI increments and $\overline{\text{HREQ}}$ is again asserted. Steps 2-5 are then repeated.
6. If TXL ($7) was written, TXDE will be set to zero and the address counter in the HI will be loaded with the contents of HM1 and HM0. When TXDE=0, the contents of TXH:TXM:TXL are transferred to HRX provided HRDF=0. After the transfer to HRX, TXDE will be set to one, and $\overline{\text{HREQ}}$ will be asserted to start the transfer of another word from external memory to the HI.
7. When the transfer to HRX occurs within the HI, HRDF is set to one. Assuming HRIE=1, a host receive exception will be generated. The exception routine must read the HRX to clear HRDF.



1. PROGRAM DMA CONTROLLER.
- — START ADDRESS
- — BYTE COUNT
- — TRANSFER DIRECTION
- — START DMA CHANNEL

2. INITIALIZE DSP56003/005 HOST INTERFACE.
- — MODE 24 BIT DMA
- — HOST TO DSP
- — USE INIT BIT TO:
  SET TXDE
  CLEAR HRDF

3. TELL DSP56003/005
- — WHERE TO STORE DATA (i.e., PROGRAM ADDRESS REGISTER R7).
- — ENABLE INTERRUPT HRIE (CAN BE DONE WITH A HOST COMMAND).

4. ASSERT $\overline{\text{HREQ}}$ TO START DMA TRANSFER.

5. HOST IS FREE TO PERFORM OTHER TASKS (i.e., DSP TO HOST TRANSFER ON A POLLED BASIS).

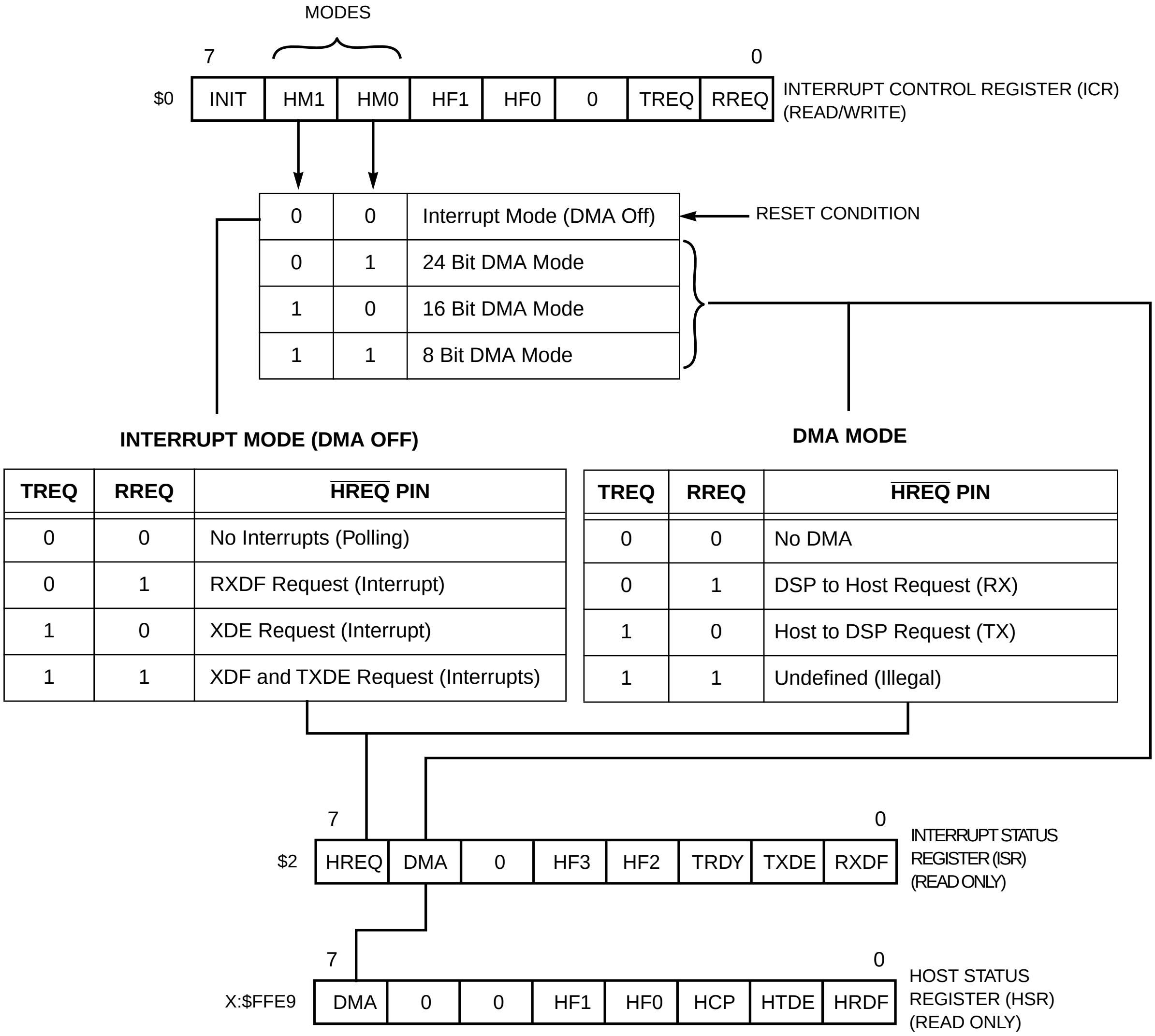
6. DMA CONTROLLER PERFORMS WRITES.

7. DMA CONTROLLER INTERRUPTS HOST WHEN TRANSFERS ARE DONE.

8. TERMINATE DMA CHANNEL.

9. TERMINATE DSP DMA MODE BY CLEARING HM1, HM0, AND TREQ.

Figure 5-38 Host-to-DSP DMA Procedure



INIT	HM1	HM0	HF1	HF0	0	TREQ	RREQ

HM1	HM0	
0	0	Interrupt Mode (DMA Off)
0	1	24 Bit DMA Mode
1	0	16 Bit DMA Mode
1	1	8 Bit DMA Mode

INTERRUPT MODE (DMA OFF)

TREQ	RREQ	HREQ PIN
0	0	No Interrupts (Polling)
0	1	RXDF Request (Interrupt)
1	0	XDE Request (Interrupt)
1	1	XDF and TXDE Request (Interrupts)

DMA MODE

TREQ	RREQ	HREQ PIN
0	0	No DMA
0	1	DSP to Host Request (RX)
1	0	Host to DSP Request (TX)
1	1	Undefined (Illegal)

HREQ	DMA	0	HF3	HF2	TRDY	TXDE	RXDF

DMA	0	0	HF1	HF0	HCP	HTDE	HRDF

Figure 5-39 Host Bits with TREQ and RREQ

Note: The transfer of data from the TXH, TXM, TXL registers to the HRX register automatically loads the DMA address counter from the HM1 and HM0 bits in the DMA host to DSP mode. This DMA address is used with the HI to place the received byte in the correct register (TXH, TXM, or TXL).

Figure 5-37 shows the differences between 24-, 16-, and 8-bit DMA data transfers. The interrupt rate is three times faster for 8-bit data transfers than for 24-bit transfers. TXL is always loaded last.

5.3.6.3.2 Host to DSP DMA Procedure

The following procedure outlines the typical steps that the host processor must take to setup and terminate a host-to-DSP DMA transfer (see Figure 5-38).

1. Set up the external DMA controller (1) source address, byte count, direction, and other control registers. Enable the DMA controller channel.
2. Initialize the HI (2) by writing the ICR to select the word size (HM0 and HM1), to select the direction (TREQ=1, RREQ=0), and to initialize the channel setting INIT=1 (see Figure 5-39).
3. Initialize the DSP's destination pointer (3) used in the DMA exception handler (an address register, for example) and set HRIE to enable the HRDF interrupt to the DSP CPU. This procedure can be done with a separate host command exception routine in the DSP. $\overline{\text{HREQ}}$ will be asserted (4) immediately by the HI to begin the DMA transfer.
4. Perform other tasks (5) while the DMA controller transfers data (6) until interrupted by the DMA controller DMA transfer complete interrupt (7). The DSP interrupt control register (ICR), the interrupt status register (ISR), and RXH, RXM, and RXL registers may be accessed at any time by the host processor but the TXH, TXM and TXL registers may not be accessed until the DMA mode is disabled.
5. Terminate the DMA controller channel (8) to disable DMA transfers.
6. Terminate the DSP HI DMA mode (9) in the ICR by clearing the HM1 and HM0 bits and clearing TREQ.

The $\overline{\text{HREQ}}$ will be active immediately after initialization is completed (depending on hardware) because the data direction is host to DSP and TXH, TXM, and TXL registers are empty. When the host writes data to TXH, TXM, and TXL, this data will be immediately transferred to HRX. If the DSP is due to work in interrupt mode, HRIE must be enabled.

5.3.6.3.3 DSP to Host Internal Processing

The following procedure outlines the steps that the HI hardware takes to transfer DMA data from DSP memory to the host data bus.

1. On the DSP side of the HI, a host transmit exception will be generated when HTDE=1 and HTIE=1. The exception routine must write HTX, thereby setting HTDE=0.
2. If RXDF=0 and HTDE=0, the contents of HTX will be automatically transferred to RXH:RXM:RXL, thereby setting RXDF=1 and HTDE=1. Since HTDE=1 again on the initial transfer, a second host transmit exception will be generated immediately, and HTX will be written, which will clear HTDE again.
3. When RXDF is set to one, the HI's internal DMA address counter is loaded (from HM1 and HM0) and $\overline{\text{HREQ}}$ is asserted.
4. The DMA controller enables the data from the appropriate byte register onto H0-H7 by asserting $\overline{\text{HACK}}$. When $\overline{\text{HACK}}$ is asserted, $\overline{\text{HREQ}}$ is deasserted by the HI.

5. The DMA controller latches the data presented on H0-H7 and deasserts $\overline{\text{HACK}}$. If the byte register read was not RXL (i.e., not $7), the HI's internal DMA counter increments, and $\overline{\text{HREQ}}$ is again asserted. Steps 3, 4, and 5 are repeated until RXL is read.
6. If RXL was read, RXDF will be set to zero and, since HTDE=0, the contents of HTX will be automatically transferred to RXH:RXM:RXL, and RXFD will be set to one. Steps 3, 4, and 5 are repeated until RXL is read again.

Note: The transfer of data from the HTX register to the RXH:RXM:RXL registers automatically loads the DMA address counter from the HM1 and HM0 bits when in the DMA DSP–HOST mode. This DMA address is used within the HI to place the appropriate byte on H0-H7.

5.3.6.3.4 DSP to Host DMA Procedure

The following procedure outlines the typical steps that the host processor must take to setup and terminate a DSP-to-host DMA transfer (see Figure 5-40).

1. Set up the DMA controller (1) destination address, byte count, direction, and other control registers. Enable the DMA controller channel.
2. Initialize the HI (2) by writing the ICR to select the word size (HM0 and HM1), the direction (TREQ=0, RREQ=1), and setting INIT=1 (see Figure 5-40 for additional information on these bits).
3. Initialize the DSP's source pointer (3) used in the DMA exception handler (an address register, for example), and set HTIE to enable the DSP host transmit interrupt. This could be done by the host processor with a host command exception routine.

   The DSP host transmit exception will be activated immediately after HTIE is set. The DSP CPU will move data to HTX. The HI circuitry will transfer the contents of HTX to RXH:RXM:RXL, setting RXDF which asserts $\overline{\text{HREQ}}$. Asserting $\overline{\text{HREQ}}$ (4) starts the DMA transfer from RXH, RXM, and RXL to the host processor.
4. Perform other tasks (5) while the DMA controller transfers data (6) until interrupted by the DMA controller DMA complete interrupt (7). The DSP interrupt control register (ICR), the interrupt status register (ISR), and TXH, TXM, and TXL may be accessed at any time by the host processor but the RXH, RXM and RXL registers may not be accessed until the DMA mode is disabled.
5. Terminate the DMA controller channel (8) to disable DMA transfers.
6. Terminate the DSP HI DMA mode (9) in the Interrupt Control Register (ICR) by clearing the HM1 and HM0 bits and clearing RREQ.

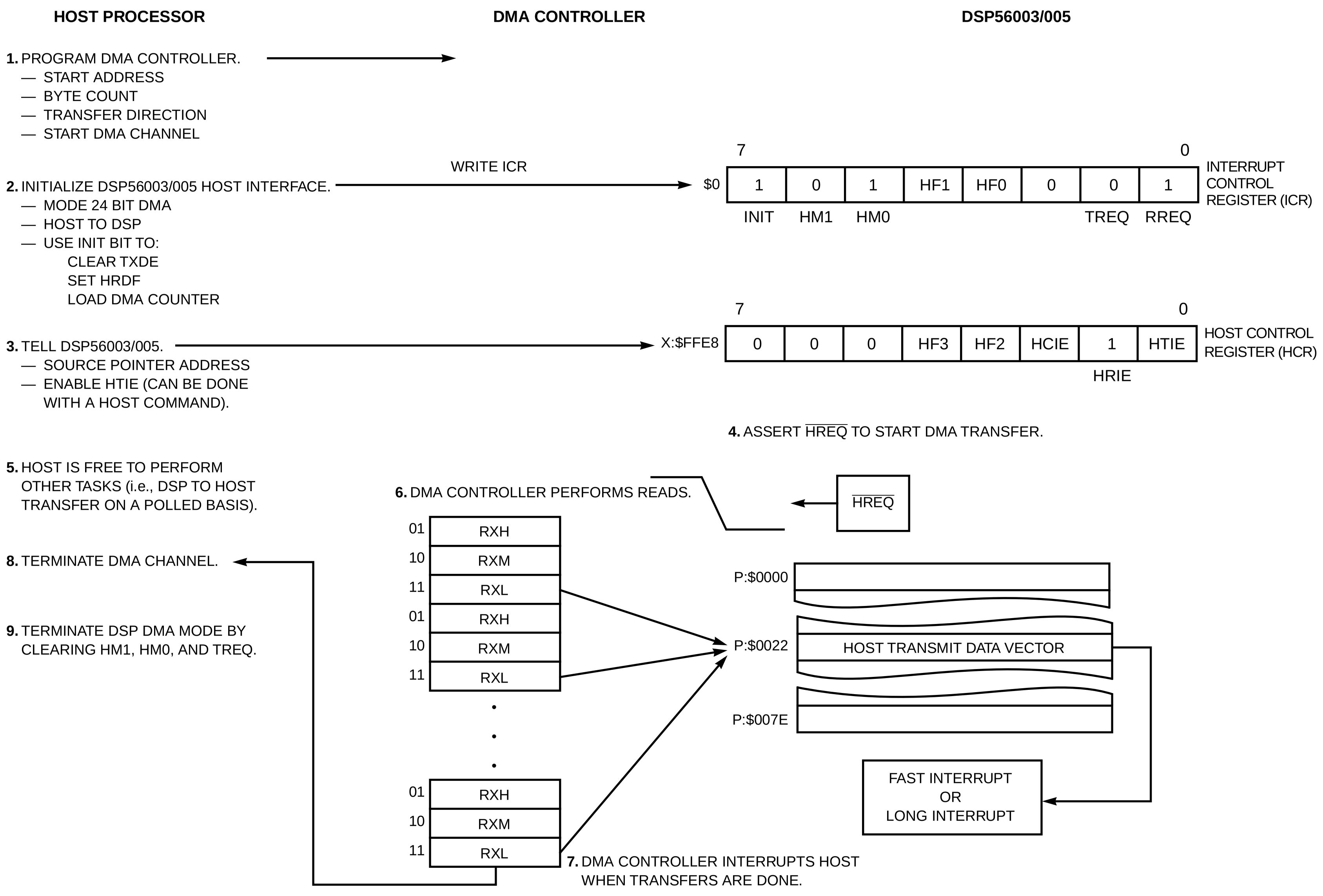

HOST PROCESSOR
DMA CONTROLLER
DSP56003/005
1. PROGRAM DMA CONTROLLER.
— START ADDRESS
— BYTE COUNT
— TRANSFER DIRECTION
— START DMA CHANNEL
WRITE ICR
2. INITIALIZE DSP56003/005 HOST INTERFACE.
— MODE 24 BIT DMA
— HOST TO DSP
— USE INIT BIT TO:
CLEAR TXDE
SET HRDF
LOAD DMA COUNTER
7
0
$0
1
0
1
HF1
HF0
0
0
1
INIT
HM1
HM0
TREQ
RREQ
INTERRUPT CONTROL REGISTER (ICR)
3. TELL DSP56003/005.
— SOURCE POINTER ADDRESS
— ENABLE HTIE (CAN BE DONE WITH A HOST COMMAND).
7
0
X:$FFE8
0
0
0
HF3
HF2
HCIE
1
HTIE
HRIE
HOST CONTROL REGISTER (HCR)
4. ASSERT HREQ TO START DMA TRANSFER.
5. HOST IS FREE TO PERFORM OTHER TASKS (i.e., DSP TO HOST TRANSFER ON A POLLED BASIS).
6. DMA CONTROLLER PERFORMS READS.
HREQ
01
RXH
10
RXM
11
RXL
01
RXH
10
RXM
11
RXL
01
RXH
10
RXM
11
RXL
P:$0000
P:$0022
HOST TRANSMIT DATA VECTOR
P:$007E
FAST INTERRUPT
OR
LONG INTERRUPT
8. TERMINATE DMA CHANNEL.
9. TERMINATE DSP DMA MODE BY CLEARING HM1, HM0, AND TREQ.
7. DMA CONTROLLER INTERRUPTS HOST WHEN TRANSFERS ARE DONE.

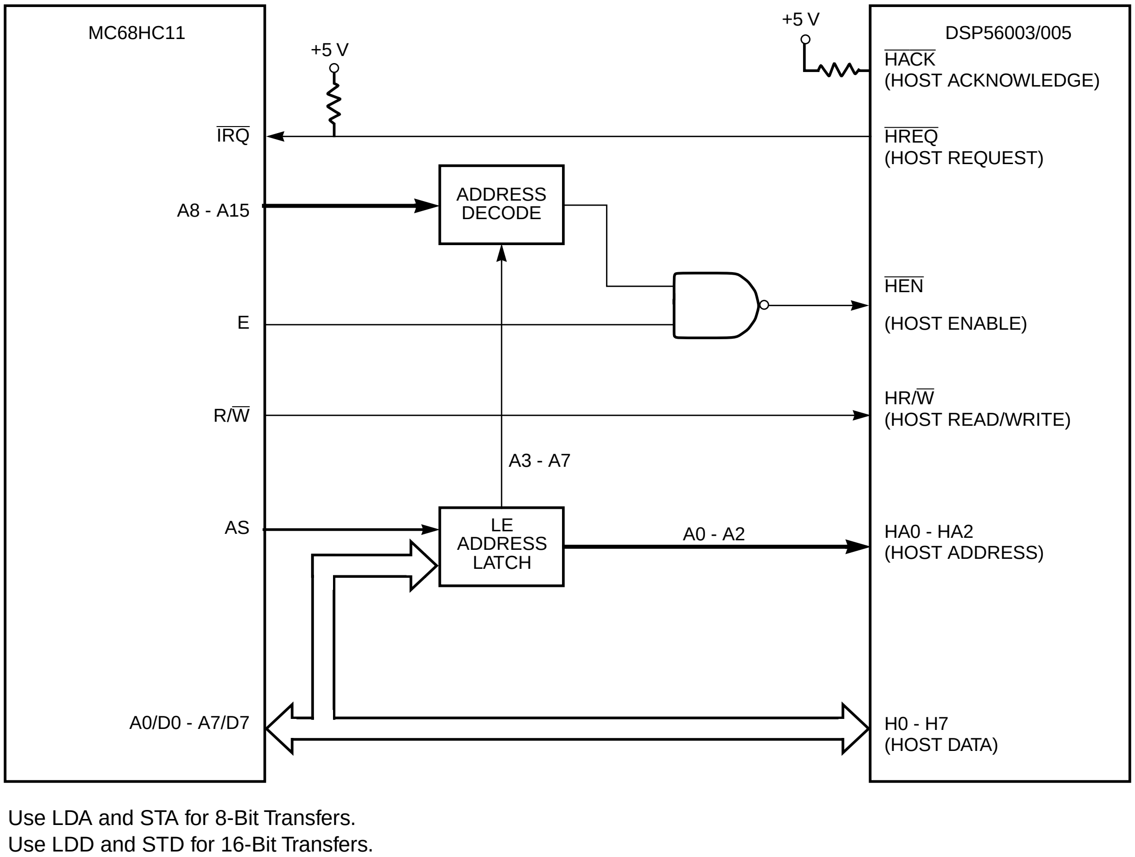
Figure 5-40 DSP to Host DMA Procedure

5.3.6.4 Example Circuits

Figure 5-41, Figure 5-42, and Figure 5-43 illustrate the simplicity of the HI. The MC68HC11 in Figure 5-42 has a multiplexed address and data bus which requires that the address be latched. Although the $\overline{\text{HACK}}$ is not used in this circuit, it is pulled up. All unused input pins should be terminated to prevent erroneous signals. When determining whether a pin is an input, keep in mind that it may change during reset or while changing Port B between general purpose I/O and HI functions.

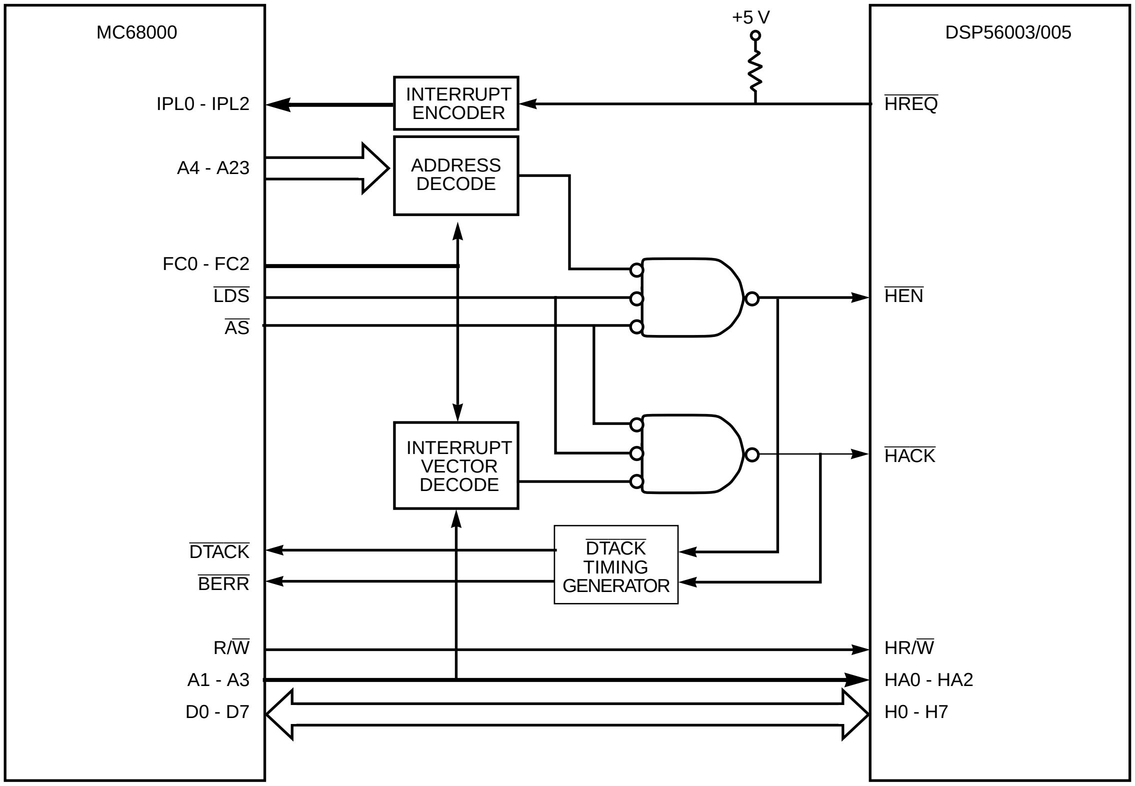
The MC68000 (see Figure 5-42) can use a MOVEP instruction with word and long-word data size to transfer multiple bytes. If an MC68020 or MC68030 is used, dynamic bus sizing can be used to transfer multiple bytes with any instruction.

Figure 5-43 is a high level block diagram of a system using a single host to control multiple DSPs. In addition, the DSPs use the SSI to network together the DSPs and multiple codecs. This system, as shown with four DSPs, can process 80 million instructions per second at 40 MHz and can be easily expanded if more processing power is needed.



Use LDA and STA for 8-Bit Transfers.
Use LDD and STD for 16-Bit Transfers.

Figure 5-41 MC68HC11 to DSP56003/005 Host Interface



MC68000 — USE MOVEP for multiple byte transfers.
MC68020 or MC68030 — Any Memory references will work due to dynamic bus sizing.

Figure 5-42 MC68000 to DSP56003/005 Host Interface

5.3.6.5 Host Port Use Considerations — Host Side

Careful synchronization is required when reading multibit registers that are written by another asynchronous system. This is a common problem when two asynchronous systems are connected. The situation exists in the Host port. The considerations for proper operation are discussed below.

1. Unsynchronized Reading of Receive Byte Registers
   When reading receive byte registers, RXH, RXM, or RXL, the Host programmer should use interrupts or poll the RXDF flag which indicates that data is available. This assures that the data in the receive byte registers will be stable.

2. Overwriting Transmit Byte Registers
   The Host programmer should not write to the transmit byte registers, TXH, TXM, or TXL, unless the TXDE bit is set indicating that the transmit byte registers are empty. This guarantees that the transmit byte registers will transfer valid data to the HRX register.

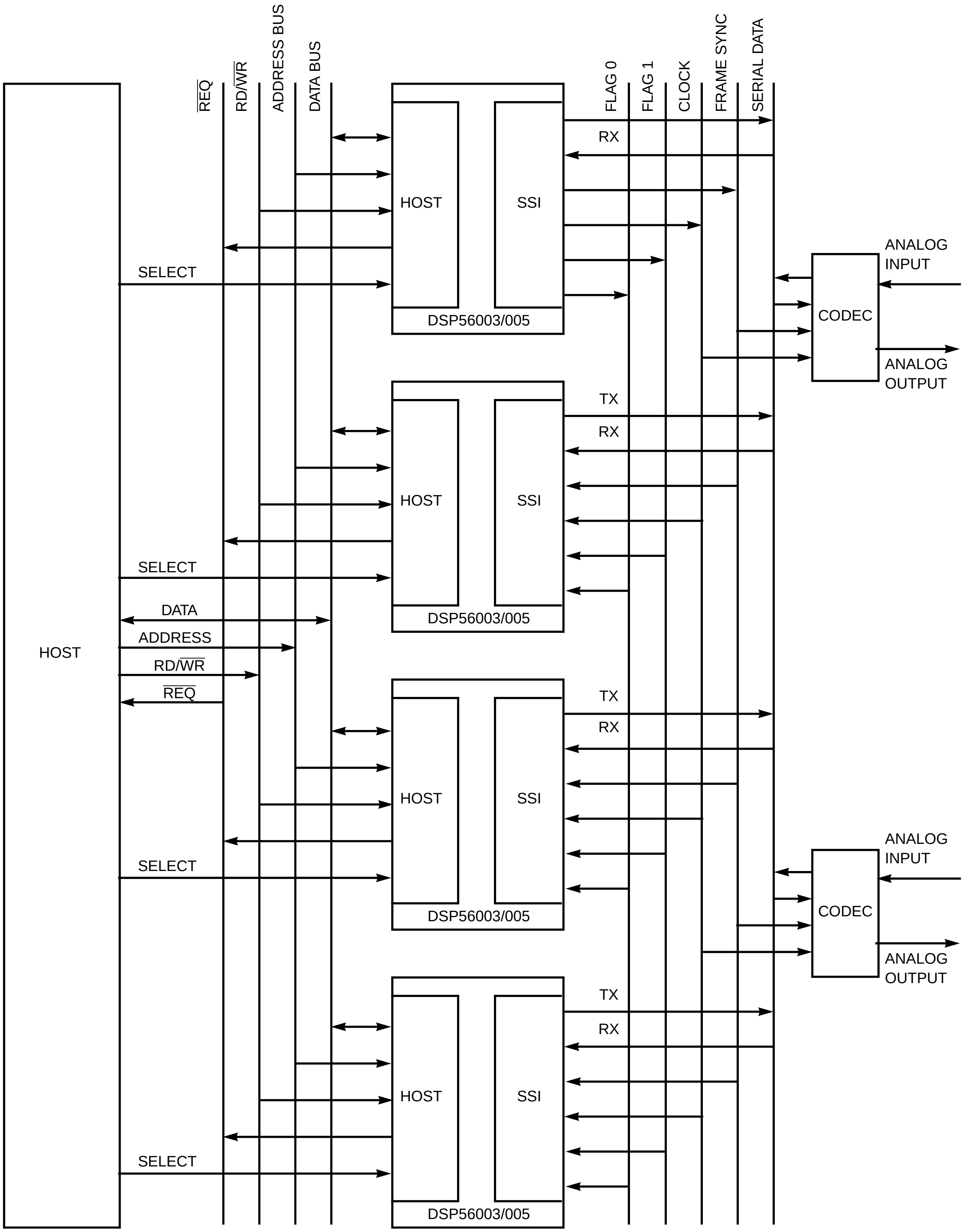


Figure 5-43 Multi-DSP Network Example

3. Synchronization of Status Bits from DSP to Host

   HC, HREQ, DMA, HF3, HF2, TRDY, TXDE, and RXDF status bits (refer to *DSP56000/001 User's Manual*, I/O Interface section, Host/DMA Interface Programming Model for descriptions of these status bits) are set or cleared from inside the DSP and read by the Host processor. The Host can read these status bits very quickly without regard to the clock rate used by the DSP, but the possibility exists that the state of the bit could be changing during the read operation. This is generally not a system problem, since the bit will be read correctly in the next pass of any Host polling routine.

   However, if the Host asserts the $\overline{\text{HEN}}$ for T31_a ns, the status data is guaranteed to be stable. A Minimum $\overline{\text{HEN}}$ deassertion time of T32 ns is required to enable internal updates of the Host status bits. This minimum time applies only if the Host processor is reading the status bits. This places a limit on the maximum polling rate for these bits.

   A potential problem exists when reading status bits HF3 and HF2 as an encoded pair. If the DSP changes HF3 and HF2 from 00 to 11, there is a small probability that the Host could read the bits during the transition and receive 01 or 10 instead of 11. If the combination of HF3 and HF2 has significance, the Host could read the wrong combination.

   Solution:

   A. Read the bits twice and check for consensus.

   B. Assert HEN access for T31 ns so that status bit transitions are stabilized.

4. Overwriting the Host Vector

   The Host programmer should change the Host Vector register only when the Host Command bit (HC) is clear. This change will guarantee that the DSP interrupt control logic will receive a stable vector.

5. Cancelling a Pending Host Command Exception

   The Host processor may elect to clear the HC bit to cancel the Host Command Exception request at any time before it is recognized by the DSP. Because the Host does not know exactly when the exception will be recognized (due to exception processing synchronization and pipeline delays), the DSP may execute the Host exception after the HC bit is cleared. For these reasons, the HV bits must not be changed at the same time the HC bit is cleared.

6. When using the $\overline{\text{HREQ}}$ pin for handshaking, wait until $\overline{\text{HREQ}}$ is asserted and then start writing/reading data using the $\overline{\text{HEN}}$ pin or the $\overline{\text{HACK}}$ pin.

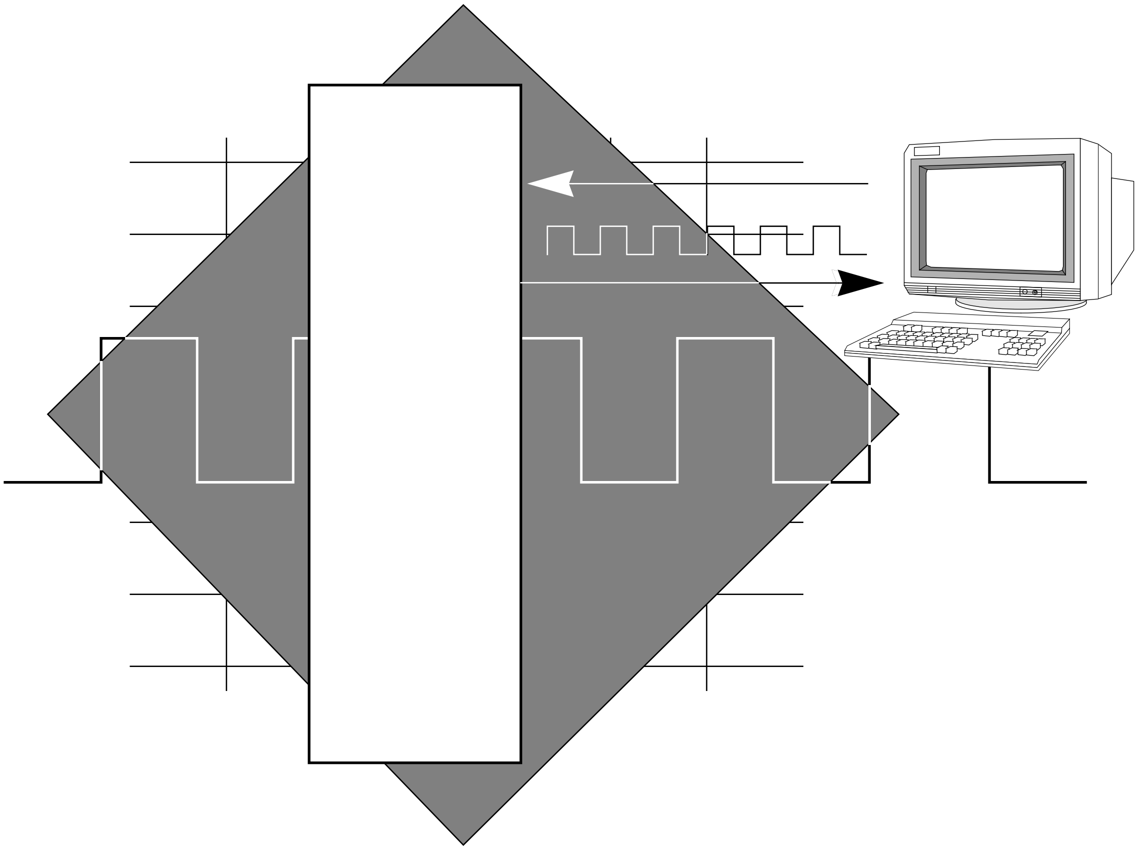
When not using $\overline{\text{HREQ}}$ for handshaking, poll the INIT bit in the ICR to make sure it is cleared by the hardware (which means the INIT execution is completed). Then, start writing/reading data.

If using neither $\overline{\text{HREQ}}$ for handshaking, nor polling the INIT bit, wait at least 6T after negation of $\overline{\text{HEN}}$ that wrote ICR, before writing/reading data. This wait ensures that the INIT is completed, because it needs 3T for synchronization (worst case) plus 3T for executing the INIT.

7. All unused input pins should be terminated. Also, any pin that is temporarily not driven by an output during reset, when reprogramming a port or pin, when a bus is not driven, or at any other time, should be pulled up or down with a resistor. For example, the $\overline{\text{HEN}}$ is capable of reacting to 2 ns noise spikes when it is not terminated. Allowing $\overline{\text{HACK}}$ to float may cause problems even though it is not needed in the circuit.

SECTION 6

SERIAL COMMUNICATIONS INTERFACE



SECTION CONTENTS

Paragraph Number	Section	Page Number

6.1 INTRODUCTION

Port C is a triple-function I/O port with nine pins (see Figure 6-1). Three of the nine pins can be configured as general-purpose I/O or as the serial communication interface (SCI) pins. The other six pins can also be configured as GPIO, or they can be configured as the synchronous serial interface (SSI) pins.

When configured as general-purpose I/O, port C can be used for device control. When the pins are configured as SCI, port C provides a convenient connection to other DSPs, processors, codecs, digital-to-analog and analog-to-digital converters, and any of several transducers. This Port C (GPIO and SCI) is identical to the one on the DSP56001 and DSP56002.

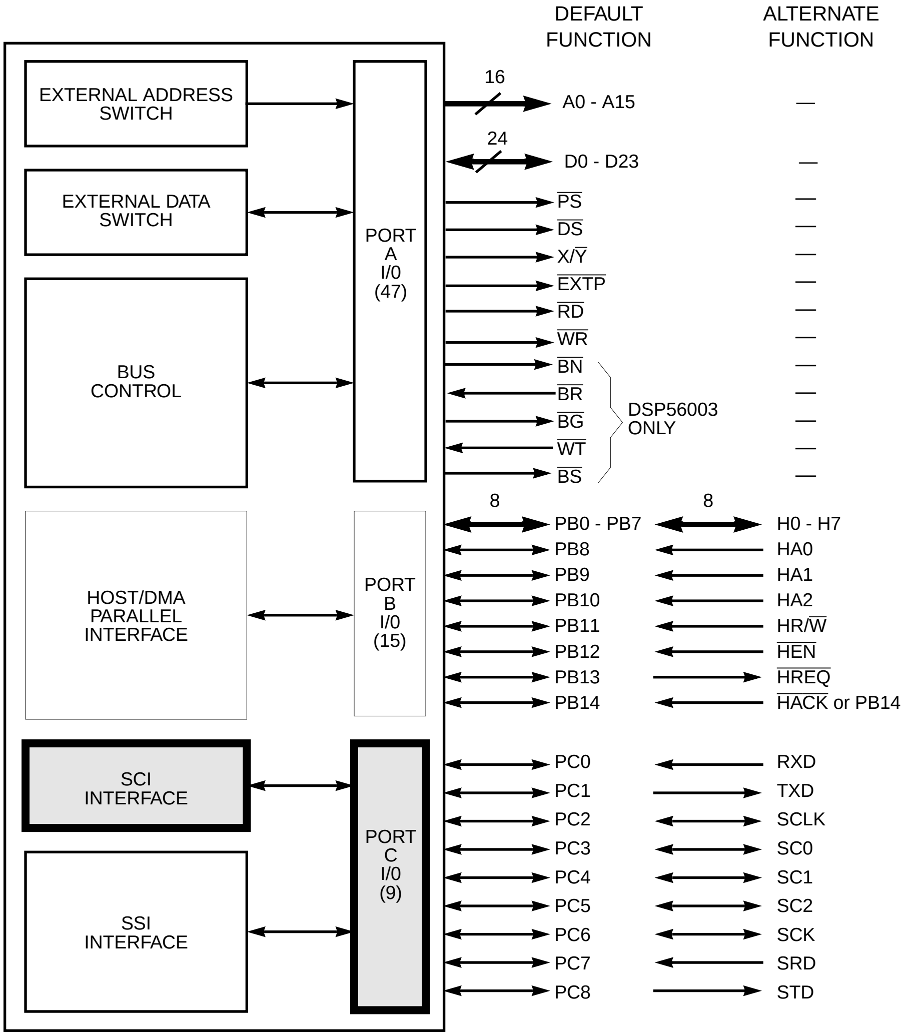


Figure 6-1 Port C Interface

6.2 GENERAL-PURPOSE I/O (PORT C)

When it is configured as GPIO, Port C can be viewed as nine I/O pins (see Figure 6-2), which are controlled by three memory-mapped registers. These registers are the Port C control register (PCC), Port C data direction register (PCDDR), and Port C data register (PCD) (see Figure 6-3).

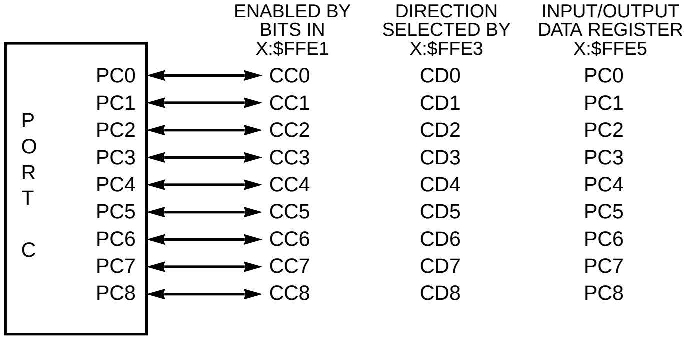


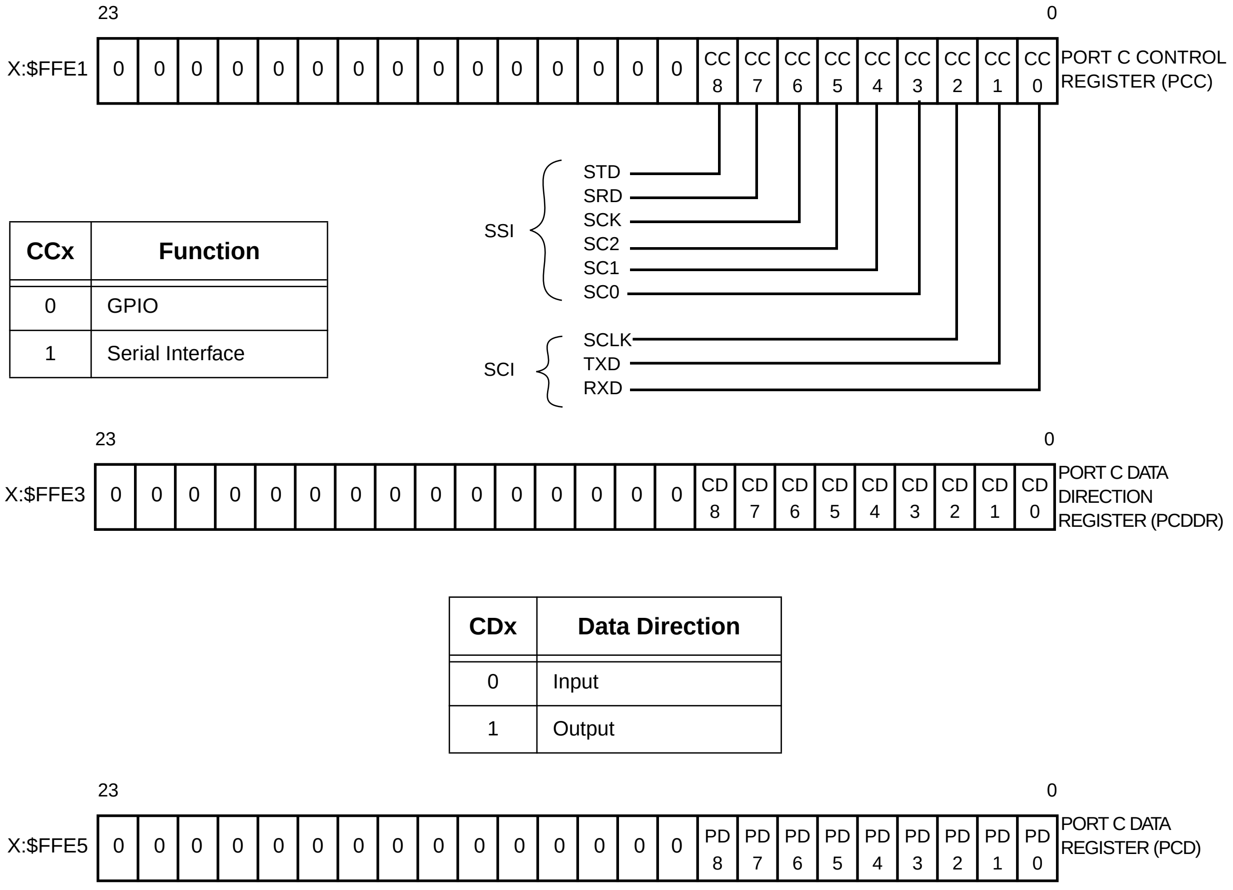
Figure 6-2 Port C GPIO Control

Reset configures Port C as general-purpose I/O with all 9 pins as inputs by clearing both the control (PCC), and data direction (PCDDR) registers (external circuitry connected to these pins may need pullups until the pins are configured for operation). There are three registers associated with each external pin. Each Port C pin may be individually programmed as a general-purpose I/O pin or as a dedicated on-chip peripheral pin under software control. Pin selection between general-purpose I/O and SCI or SSI is made by setting the appropriate PCC bit (memory location X:$FFE1) to zero for general-purpose I/O or to one for serial interface.

The PCDDR (memory location X:$FFE3) programs each pin corresponding to a bit in the PCD (memory location X:$FFE5) as an input pin (if PCDDR=0) or as an output pin (if PCDDR=1).

If a pin is configured as a GPIO **input** (as shown in Figure 6-4) and the processor reads the PCD, the processor sees the logic level on the pin. If the processor writes to the PCD, the data is latched there, but does not appear on the pin because the buffer is in the high-impedance state.

If a pin is configured as a GPIO **output** and the processor reads the PCD, the processor sees the contents of the PCD rather the logic level on the pin, which allows the PCD to be used as a general purpose 15-bit register. If the processor writes to the PCD, the data is latched there and appears on the pin during the following instruction cycle (see **Section 6.2.2**).



CCx	Function
0	GPIO
1	Serial Interface

CDx	Data Direction
0	Input
1	Output

NOTE: Hardware and software reset clears PCC and PCDDR.

Figure 6-3 Port C GPIO Registers

If a pin is configured as a **serial interface** (SCI or SSI) pin, the Port C GPIO registers can be used to help in debugging the serial interface. If the PCDDR bit for a given pin is cleared (configured as an input), the PCD will show the logic level on the pin, regardless of whether the serial interface function is using the pin as an input or an output. If the PCDDR is set (configured as an output) for a given serial interface pin, when the processor reads the PCD, it sees the contents of the PCD rather than the logic level on the pin — another case which allows the PCD to act as a general purpose register.

6.2.1 Programming General Purpose I/O

Port C and all the DSP56003/005 peripherals are memory mapped (see Figure 6-5). The standard MOVE instruction transfers data between Port C and a register; as a result, performing a memory-to-memory data transfer takes two MOVE instructions and a register. The MOVEP instruction is specifically designed for I/O data transfer as shown in Figure 6-6.

Port Control Register Bit	Data Direction Register Bit	Pin Function
0	0	Port Input Pin

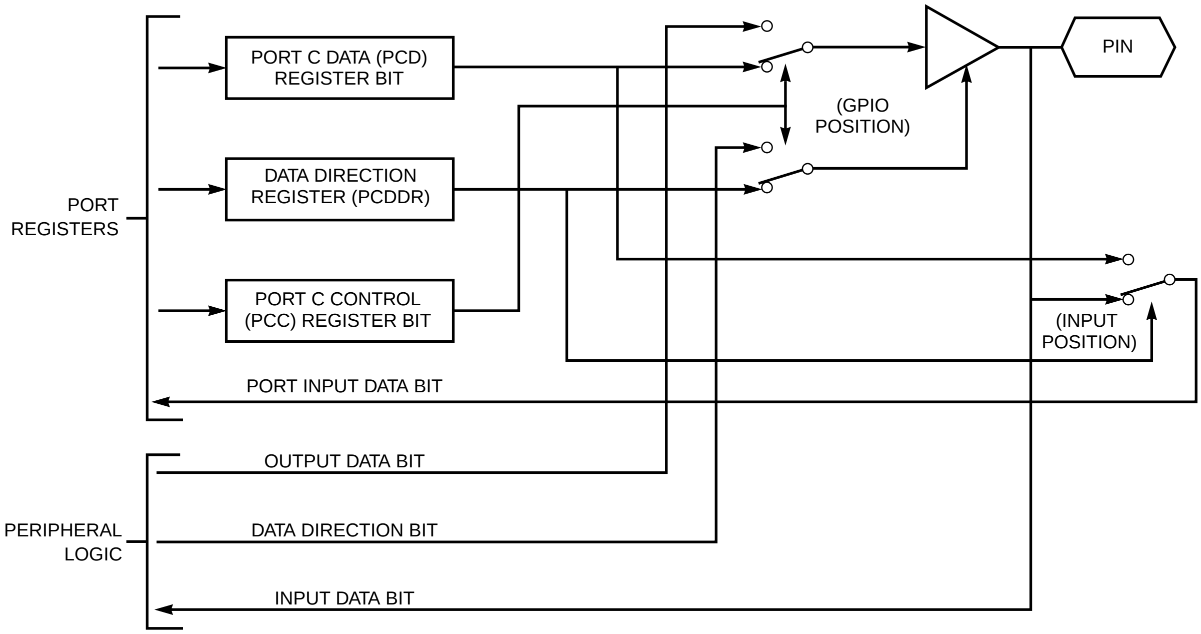
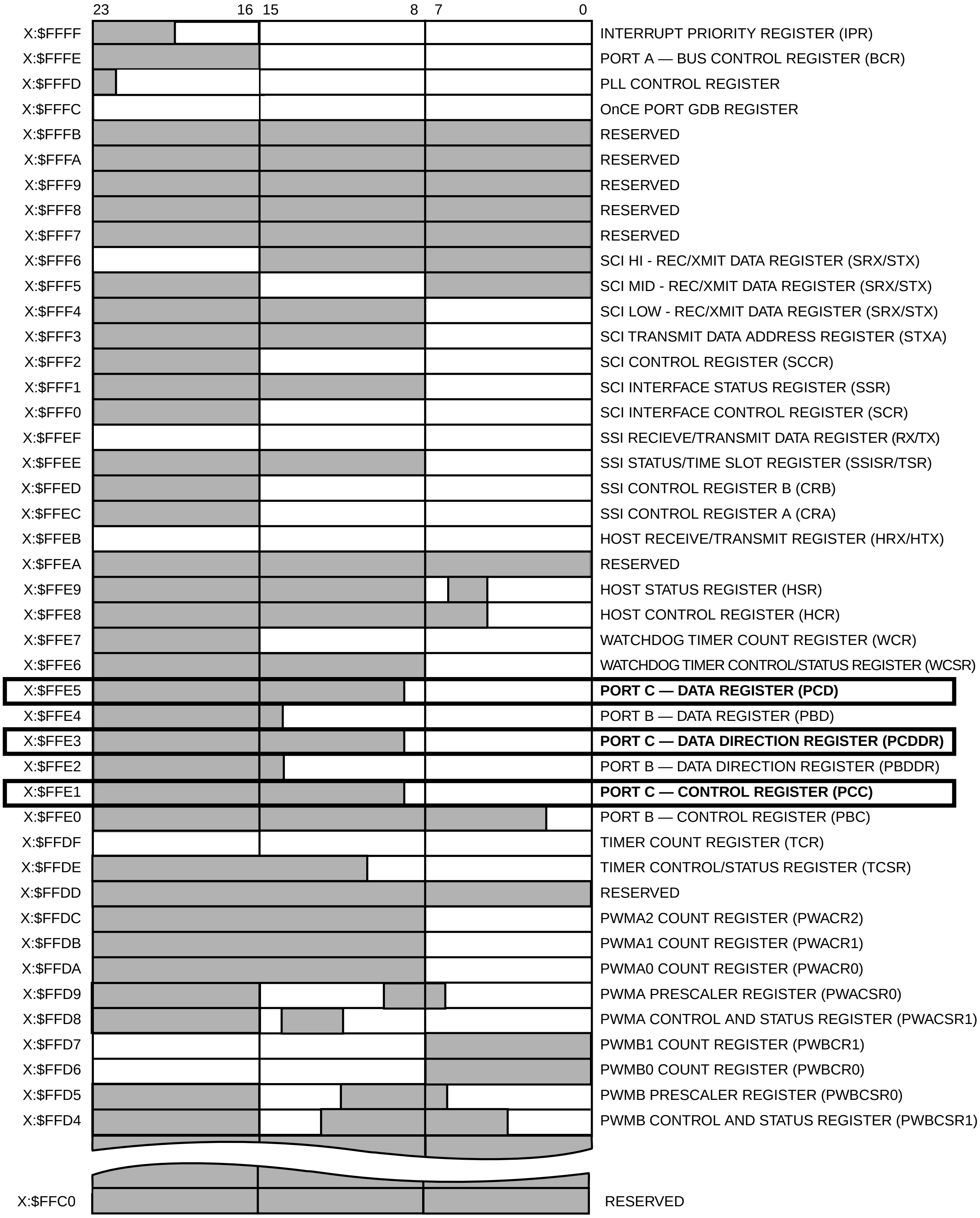


Figure 6-4 Port C I/O Pin Control Logic

Although the MOVEP instruction may take twice as long to execute as a MOVE instruction, only one MOVEP is required for a memory-to-memory data transfer, and MOVEP does not use a temporary register. Using the MOVEP instruction allows a fast interrupt to move data to/from a peripheral to memory and execute one other instruction or to move the data to an absolute address. MOVEP is the only memory-to-memory move instruction; however, one of the operands must be in the top 64 locations of either X: or Y: memory. The bit-oriented instructions which use I/O short addressing (BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, and JSSET) can also be used to address individual bits for faster I/O processing.

The DSP does not have a hardware data strobe to strobe data out of the GPIO port. If a data strobe is needed, it can be implemented using software to toggle one of the GPIO pins.



X:$FFFF	INTERRUPT PRIORITY REGISTER (IPR)
X:$FFFE	PORT A — BUS CONTROL REGISTER (BCR)
X:$FFFD	PLL CONTROL REGISTER
X:$FFFC	OnCE PORT GDB REGISTER
X:$FFFB	RESERVED
X:$FFFA	RESERVED
X:$FFF9	RESERVED
X:$FFF8	RESERVED
X:$FFF7	RESERVED
X:$FFF6	SCI HI - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF5	SCI MID - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF4	SCI LOW - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF3	SCI TRANSMIT DATA ADDRESS REGISTER (STXA)
X:$FFF2	SCI CONTROL REGISTER (SCCR)
X:$FFF1	SCI INTERFACE STATUS REGISTER (SSR)
X:$FFF0	SCI INTERFACE CONTROL REGISTER (SCR)
X:$FFEF	SSI RECIEVE/TRANSMIT DATA REGISTER (RX/TX)
X:$FFEE	SSI STATUS/TIME SLOT REGISTER (SSISR/TSR)
X:$FFED	SSI CONTROL REGISTER B (CRB)
X:$FFEC	SSI CONTROL REGISTER A (CRA)
X:$FFEB	HOST RECEIVE/TRANSMIT REGISTER (HRX/HTX)
X:$FFEA	RESERVED
X:$FFE9	HOST STATUS REGISTER (HSR)
X:$FFE8	HOST CONTROL REGISTER (HCR)
X:$FFE7	WATCHDOG TIMER COUNT REGISTER (WCR)
X:$FFE6	WATCHDOG TIMER CONTROL/STATUS REGISTER (WCSR)
X:$FFE5	**PORT C — DATA REGISTER (PCD)**
X:$FFE4	PORT B — DATA REGISTER (PBD)
X:$FFE3	**PORT C — DATA DIRECTION REGISTER (PCDDR)**
X:$FFE2	PORT B — DATA DIRECTION REGISTER (PBDDR)
X:$FFE1	**PORT C — CONTROL REGISTER (PCC)**
X:$FFE0	PORT B — CONTROL REGISTER (PBC)
X:$FFDF	TIMER COUNT REGISTER (TCR)
X:$FFDE	TIMER CONTROL/STATUS REGISTER (TCSR)
X:$FFDD	RESERVED
X:$FFDC	PWMA2 COUNT REGISTER (PWACR2)
X:$FFDB	PWMA1 COUNT REGISTER (PWACR1)
X:$FFDA	PWMA0 COUNT REGISTER (PWACR0)
X:$FFD9	PWMA PRESCALER REGISTER (PWACSR0)
X:$FFD8	PWMA CONTROL AND STATUS REGISTER (PWACSR1)
X:$FFD7	PWMB1 COUNT REGISTER (PWBCR1)
X:$FFD6	PWMB0 COUNT REGISTER (PWBCR0)
X:$FFD5	PWMB PRESCALER REGISTER (PWBCSR0)
X:$FFD4	PWMB CONTROL AND STATUS REGISTER (PWBCSR1)
X:$FFC0	RESERVED

▨ = Read as random number; write as don't care.

Figure 6-5 On-Chip Peripheral Memory Map

```
:
:
MOVEP #$0,X:$FFE1     ;Select Port C to be general-purpose I/O
MOVEP #$01F0,X:$FFE3  ;Select pins PC0–PC3 to be inputs
:                     ;and pins PC4–PC8 to be outputs
:
MOVEP #data_out,X:$FFE5 ;Put bits 4–8 of “data_out” on pins
                        ;PB4–PB8 bits 0–3 are ignored.
MOVEP X:$FFE0,#data_in  ;Put PB0–PB3 in bits 0–3 of “data_in”
```

Figure 6-6 Write/Read Parallel Data with Port C

Figure 6-7 shows the process of programming Port C as general-purpose I/O. Normally, it is not good programming practice to activate a peripheral before programming it. However, reset activates the Port C general-purpose I/O as all inputs, and the alternative is to configure the port as an SCI and/or SSI, which may not be desirable. In this case, it is probably better to insure that Port C is initially configured for general-purpose I/O and then configure the data direction and data registers. It may be better in some situations to program the data direction or the data registers first to prevent two devices from driving one signal. The order of steps 1, 2, and 3 in Figure 6-7 is optional and can be changed as needed.

6.2.2 Port C General Purpose I/O Timing

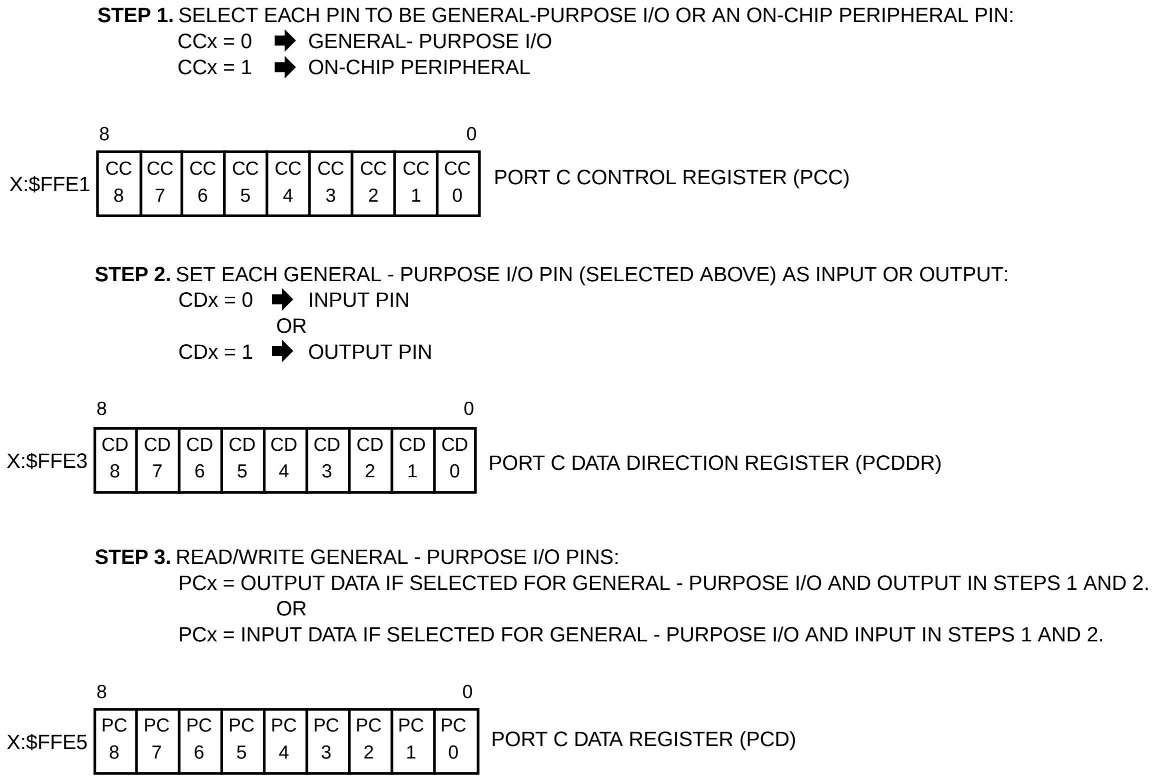
Parallel data written to Port C is delayed by one instruction cycle. For example, the following instruction:

```
MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN
```

1. writes nine bits of data to the Port C register, but the output pins do not change until the following instruction cycle
2. writes 24 bits of data to the external Y memory, which appears on Port A during T2 and T3 of the current instruction

As a result, if it is necessary to synchronize the Port A and Port C outputs, two instructions must be used:

```
MOVE    DATA9,X:PORTC
NOP                      DATA24,Y:EXTERN
```



STEP 1. SELECT EACH PIN TO BE GENERAL-PURPOSE I/O OR AN ON-CHIP PERIPHERAL PIN:
CCx = 0 ➡ GENERAL- PURPOSE I/O
CCx = 1 ➡ ON-CHIP PERIPHERAL

	8								0	
X:$FFE1	CC 8	CC 7	CC 6	CC 5	CC 4	CC 3	CC 2	CC 1	CC 0	PORT C CONTROL REGISTER (PCC)

STEP 2. SET EACH GENERAL - PURPOSE I/O PIN (SELECTED ABOVE) AS INPUT OR OUTPUT:
CDx = 0 ➡ INPUT PIN
OR
CDx = 1 ➡ OUTPUT PIN

	8								0	
X:$FFE3	CD 8	CD 7	CD 6	CD 5	CD 4	CD 3	CD 2	CD 1	CD 0	PORT C DATA DIRECTION REGISTER (PCDDR)

STEP 3. READ/WRITE GENERAL - PURPOSE I/O PINS:
PCx = OUTPUT DATA IF SELECTED FOR GENERAL - PURPOSE I/O AND OUTPUT IN STEPS 1 AND 2.
OR
PCx = INPUT DATA IF SELECTED FOR GENERAL - PURPOSE I/O AND INPUT IN STEPS 1 AND 2.

	8								0	
X:$FFE5	PC 8	PC 7	PC 6	PC 5	PC 4	PC 3	PC 2	PC 1	PC 0	PORT C DATA REGISTER (PCD)

Figure 6-7 I/O Port C Configuration

The NOP can be replaced by any instruction that allows parallel moves. Inserting one or more “MOVE DATA15,X:PORTC DATA24,Y:EXTERN” instructions between the first and second instruction produces an external 33-bit write each instruction cycle with only one instruction cycle lost in setup time:

```
MOVE    DATA9,X:PORTC

MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN

MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN

   :

   :

MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN

NOP                      DATA24,Y:EXTERN
```

One application of this technique is to create an extended address for Port A by concatenating the Port A address bits (instead of data bits) to the Port C general-purpose output bits. The Port C general-purpose I/O register would then work as a base address register, allowing the address space to be extended from 64K words (16 bits) to 33.5 million words (16 bits+ 9 bits=25 bits).

Port C uses the DSP central processing unit (CPU) four-phase clock for its operation. Therefore, if wait states are inserted in the DSP CPU timing, they also affect Port C timing. As a result, Port A and Port C in the previous synchronization example will always stay synchronized, regardless of how many wait states are used.

6.3 SERIAL COMMUNICATION INTERFACE (SCI)

The SCI provides a full-duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems. The communication can be TTL-level signals or, with additional logic, RS232C, RS422, etc.

This interface uses three dedicated pins: transmit data (TXD), receive data (RXD), and SCI serial clock (SCLK). It supports industry-standard asynchronous bit rates and protocols as well as high-speed (up to 5 Mbps for a 40-MHz clock) synchronous data transmission. The asynchronous protocols include a multidrop mode for master/slave operation with wakeup on idle line and wakeup on address bit capability.

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

- Three-Pin Interface:
  TXD – Transmit Data
  RXD – Receive Data
  SCLK – Serial Clock
- 781.25 Kbps NRZ Asynchronous Communications Interface (50-MHz System Clock)
- 6.25 Mbps Synchronous Serial Mode (50-MHz System Clock)
- Multidrop Mode for Multiprocessor Systems:
  Two Wakeup Modes: Idle Line and Address Bit
  Wired-OR Mode
- On-Chip or External Baud Rate Generation/Interrupt Timer
- Four Interrupt Priority Levels
- Fast or Long Interrupts

6.3.1 SCI I/O Pins

The three SCI pins can be configured as either general-purpose I/O or as a specific SCI pin. Each pin is independent of the other two, so that if only TXD is needed, RXD and SCLK can be programmed for general-purpose I/O. However, at least one of the three pins must be selected as an SCI pin to release the SCI from reset.

SCI interrupts may be enabled by programming the SCI control registers before any of the SCI pins 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 will operate as they do when one or more of the SCI pins is programmed as an SCI function.

6.3.1.1 Receive Data (RXD)

This input receives byte-oriented serial data and transfers the data to the SCI receive shift register. Asynchronous input data is sampled on the positive edge of the receive clock (1 × SCLK) if SCKP equals zero. See the *DSP56003/005 Data Sheet* for detailed timing information. RXD may be programmed as a general-purpose I/O pin (PC0) when the SCI RXD function is not being used.

6.3.1.2 Transmit Data (TXD)

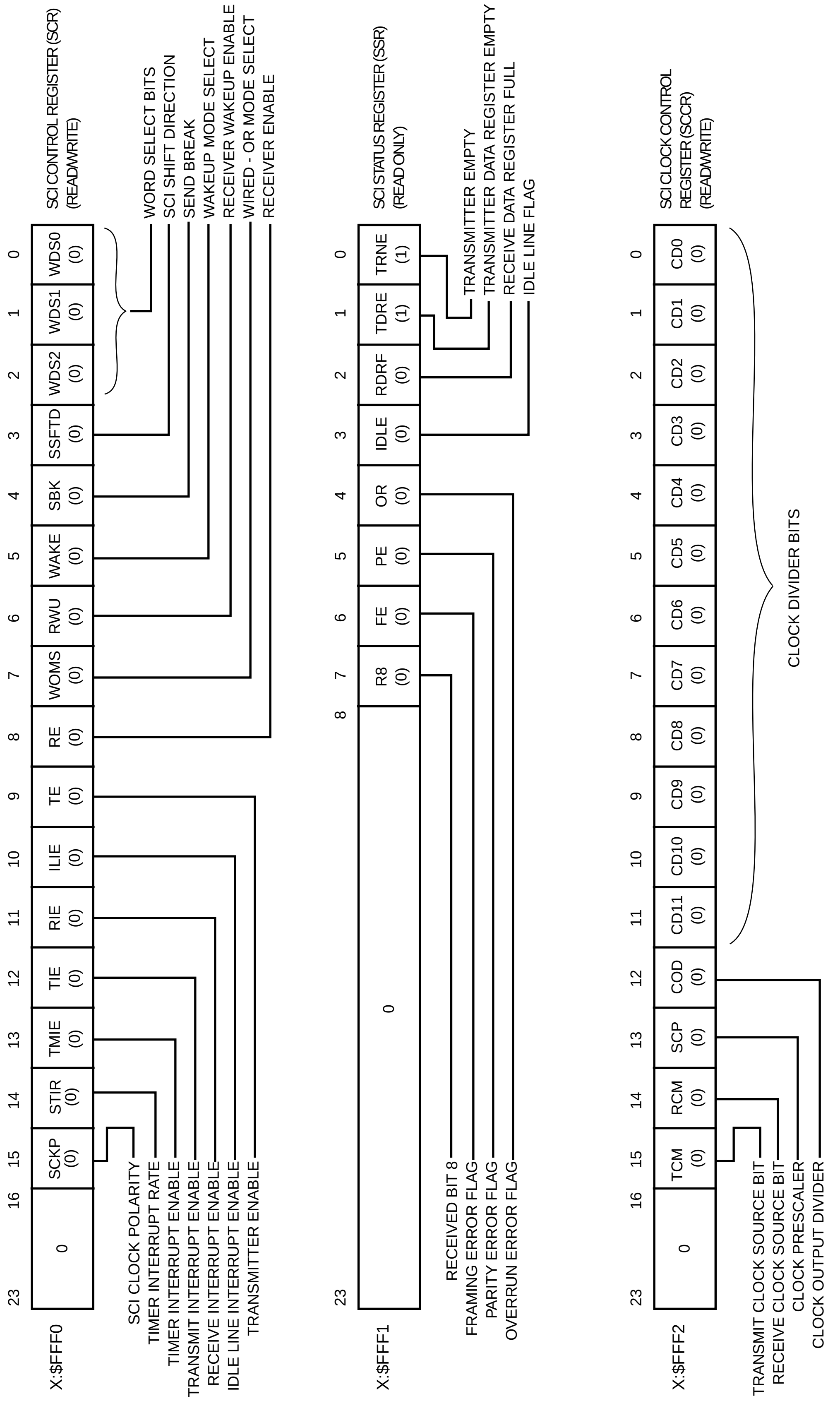
This output transmits serial data from the SCI transmit shift register. Data changes on the negative edge of the asynchronous transmit clock (SCLK) if SCKP equals zero. This output is stable on the positive edge of the transmit clock. See the *DSP56003/005 Data Sheet* for detailed timing information. TXD may be programmed as a general-purpose I/O pin (PC1) when the SCI TXD function is not being used.

6.3.1.3 SCI Serial Clock (SCLK)

This bidirectional pin provides an input or output clock from which the transmit and/or receive baud rate is derived in the asynchronous mode and from which data is transferred in the synchronous mode. SCLK may be programmed as a general-purpose I/O pin (PC2) when the SCI SCLK function is not being used. This pin may be programmed as PC2 when data is being transmitted on TXD since, in the asynchronous mode, the clock need not be transmitted. There is no connection between programming the PC2 pin as SCLK and data coming out the TXD pin because SCLK is independent of SCI data I/O.

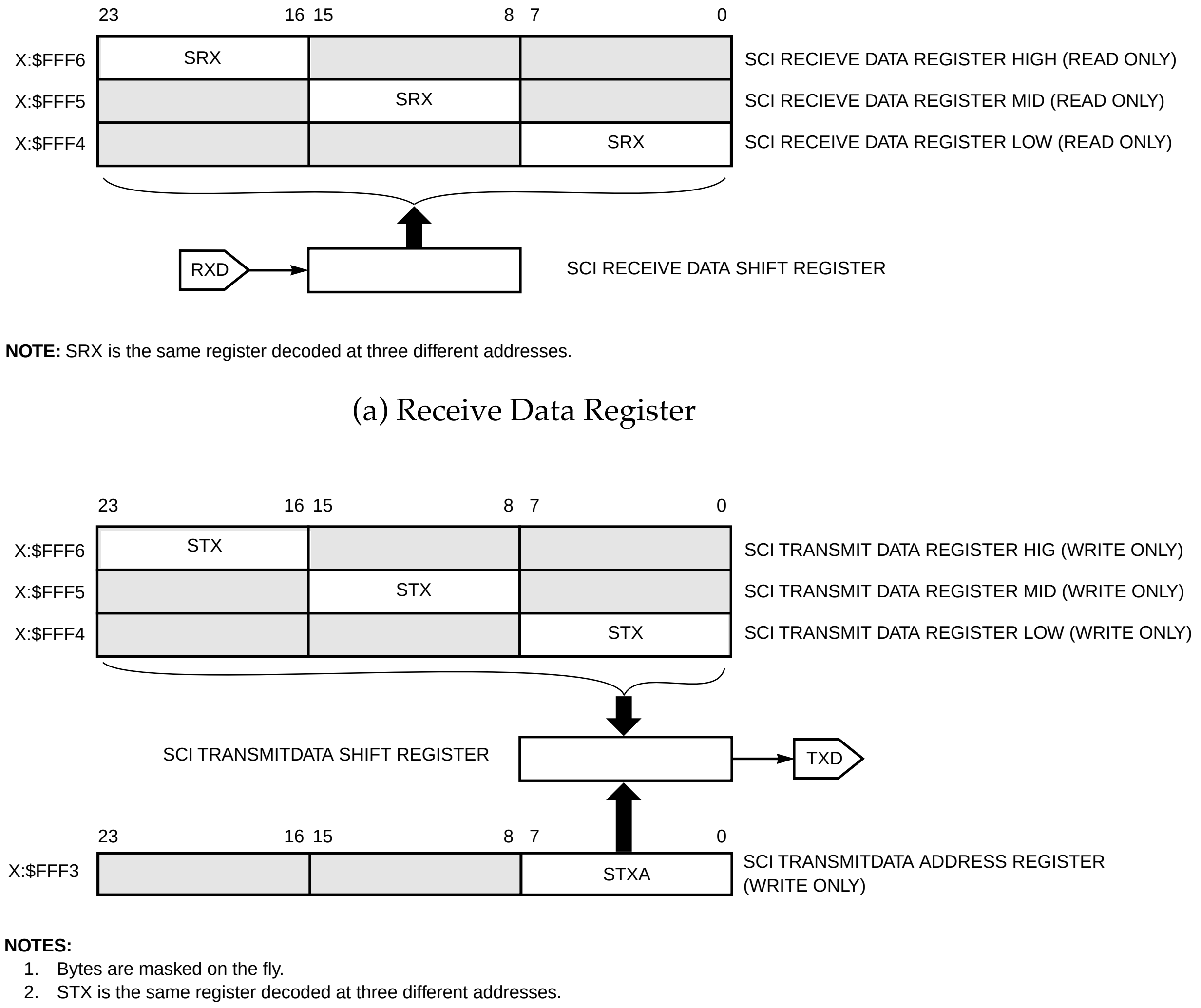
6.3.2 SCI Programming Model

The resources available in the SCI are described before discussing specific examples of how the SCI is used. The registers comprising the SCI are shown in Figure 6-8 and Figure 6-9. These registers are the SCI control register (SCR), SCI status register (SSR), SCI clock control register (SCCR), SCI receive data registers (SRX), SCI transmit data registers (STX), and the SCI transmit data address register (STXA). The SCI programming model can be viewed as three types of registers: 1) control – SCR and SCCR in Figure 6-8; 2) status – SSR in Figure 6-8; and 3) data transfer – SRX, STX, and STXA in Figure 6-9. The following paragraphs describe each bit in the programming model.



NOTE: The number in parentheses is the condition of the bit after hardware reset.

Figure 6-8 SCI Programming Model – Control and Status Registers


23
16
15
8
7
0
X:$FFF6
X:$FFF5
X:$FFF4
SRX
SRX
SRX
SCI RECIEVE DATA REGISTER HIGH (READ ONLY)
SCI RECIEVE DATA REGISTER MID (READ ONLY)
SCI RECEIVE DATA REGISTER LOW (READ ONLY)
RXD
SCI RECEIVE DATA SHIFT REGISTER
23
16
15
8
7
0
X:$FFF6
X:$FFF5
X:$FFF4
STX
STX
STXA
SCI TRANSMIT DATA REGISTER HIG (WRITE ONLY)
SCI TRANSMIT DATA REGISTER MID (WRITE ONLY)
SCI TRANSMIT DATA REGISTER LOW (WRITE ONLY)
SCI TRANSMITDATA SHIFT REGISTER
TXD
23
16
15
8
7
0
X:$FFF3
STXA
SCI TRANSMITDATA 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 three different addresses.

(b) Transmit Data Register

Figure 6-9 SCI Programming Model

6.3.2.1 SCI Control Register (SCR)

The SCR is a 16-bit read/write register that controls the serial interface operation. Each bit is described in the following paragraphs.

6.3.2.1.1 SCR Word Select (WDS0, WDS1, WDS2) Bits 0, 1, and 2

The three word-select bits (WDS0, WDS1, WDS2) select the format of the transmit and receive data. The formats include three asynchronous, one multidrop asynchronous mode, and an 8-bit synchronous (shift register) mode. The asynchronous modes are compatible with most UART-type serial devices and support standard RS232C communication links.

The multidrop asynchronous modes are compatible with the MC68681 DUART, the M68HC11 SCI interface, and the Intel 8051 serial interface.

The synchronous data mode is essentially a high-speed shift register used for I/O expansion and stream-mode channel interfaces. A gated transmit and receive clock that is compatible with the Intel 8051 serial interface mode 0 accomplishes data synchronization. The word formats are shown in Table 6-1 (also see Figure 6-10 (a) and (b)).

Table 6-1 Word Formats

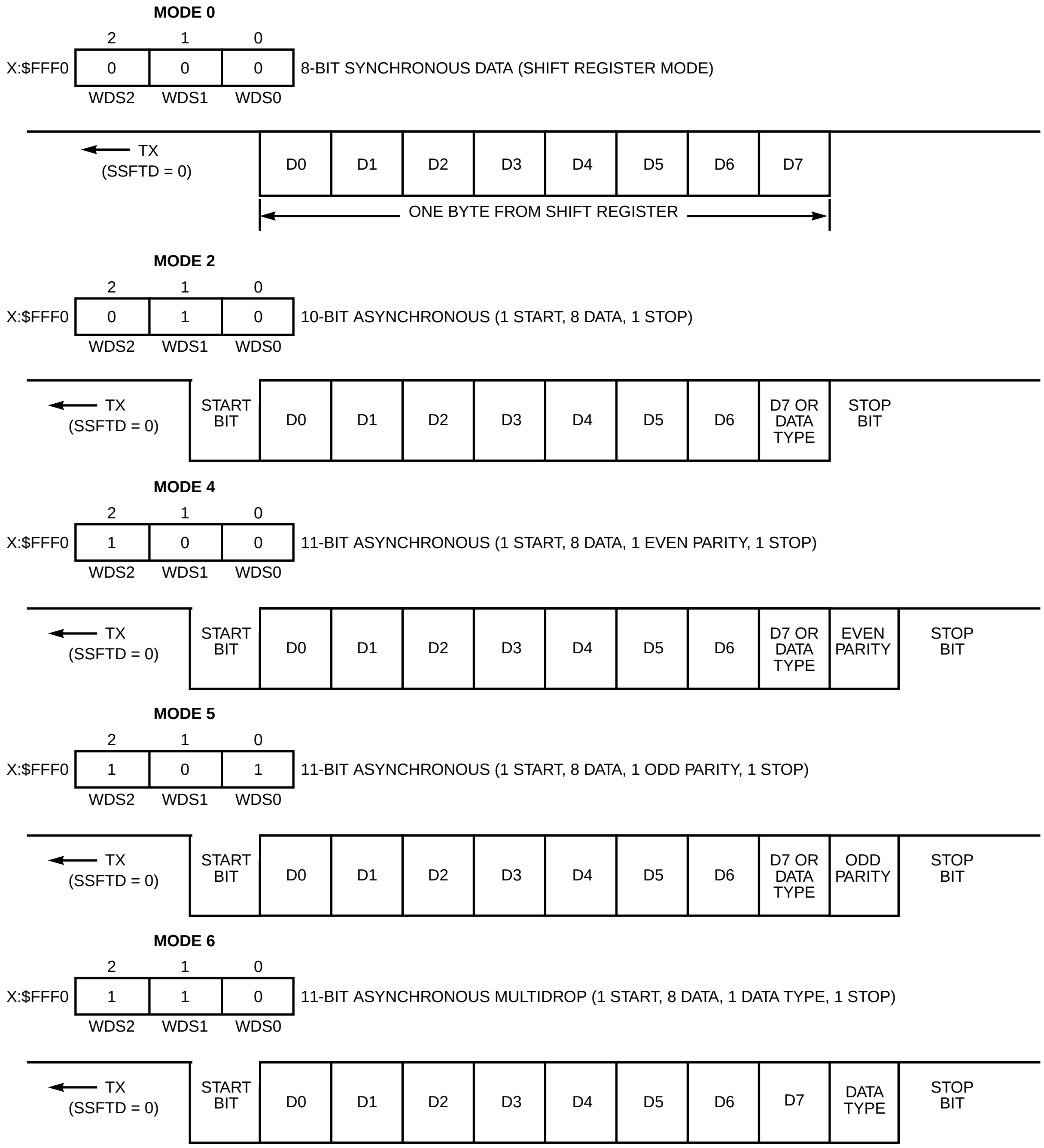
WDS2	WDS1	WDS0	Word Formats
0	0	0	8-Bit Synchronous Data (shift register mode)
0	0	1	Reserved
0	1	0	10-Bit Asynchronous (1 start, 8 data, 1 stop)
0	1	1	Reserved
1	0	0	11-Bit Asynchronous (1 start, 8 data, 1 even parity, 1 stop)
1	0	1	11-Bit Asynchronous (1 start, 8 data, 1 odd parity, 1 stop)
1	1	0	11-Bit Multidrop (1 start, 8 data, 1 data type, 1 stop)
1	1	1	Reserved

When odd parity is selected, the transmitter will count the number of bits in the data word. If the total is not an odd number, the parity bit is made equal to one and thus produces 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.

The word-select bits are cleared by hardware and software reset.

6.3.2.1.2 SCR SCI Shift Direction (SSFTD) Bit 3

The SCI data shift registers can be programmed to shift data in/out either LSB first if SSFTD equals zero, or MSB first if SSFTD equals one. The parity and data type bits do not change position and remain adjacent to the stop bit. SSFTD is cleared by hardware and software reset.



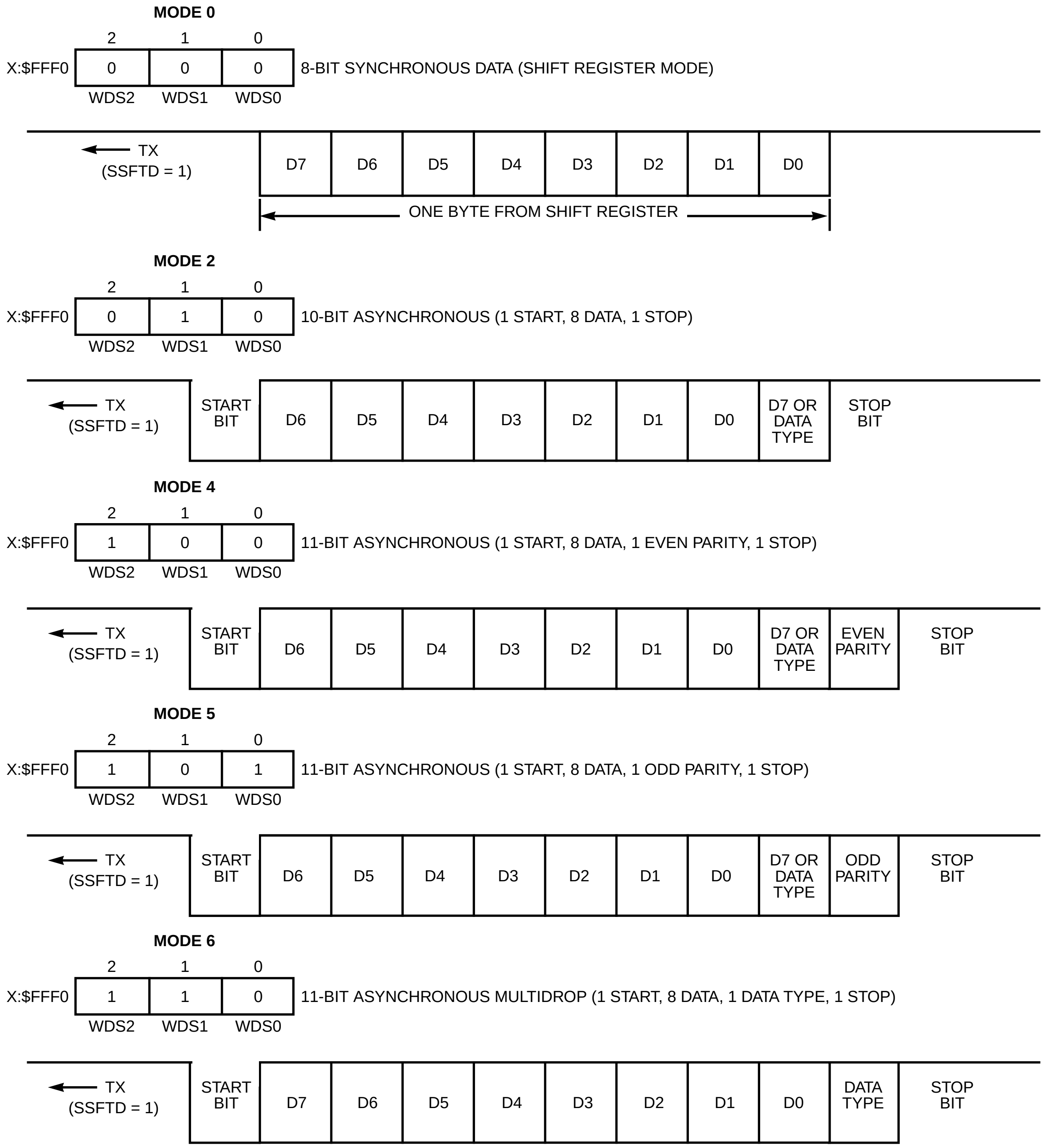
Data Type: 1 = Address Byte
0 = Data Byte

NOTES:

1. Modes1, 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.

(a) SSFTD = 0

Figure 6-10 Serial Formats (Sheet 1 of 2)



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.

(b) SSFTD = 1

Figure 6-10 Serial Formats (Sheet 2 of 2)

6.3.2.1.3 SCR Send Break (SBK) Bit 4

A break is an all-zero word frame – a start bit zero, a character of all zeros (including any parity), and a stop bit zero: i.e., 10 or 11 zeros depending on the WDS mode selected. If SBK is set and then cleared, the transmitter completes transmission of any data, sends 10 or 11 zeros, and reverts to idle or sending data. If SBK remains set, the transmitter will continually send whole frames of zeros (10 or 11 bits with no stop bit). At the completion of the break code, the transmitter sends at least one high bit before transmitting any data to guarantee recognition of a valid start bit. Break can be used to signal an unusual condition, message, etc. by forcing a frame error, which is caused by a missing stop bit. Hardware and software reset clear SBK.

6.3.2.1.4 SCR Wakeup Mode Select (WAKE) Bit 5

When WAKE equals zero, an idle line wakeup is selected. In the idle line wakeup mode, the SCI receiver is re-enabled by an idle string of at least 10 or 11 (depending on WDS mode) consecutive ones. The transmitter's software must provide this idle string between consecutive messages. The idle string cannot occur within a valid message because each word frame contains a start bit that is a zero.

When WAKE equals one, an address bit wakeup is selected. In the address bit wakeup mode, the SCI receiver is re-enabled when the last (eighth or ninth) data bit received in a character (frame) is one. The ninth data bit is the address bit (R8) in the 11-bit multidrop mode; the eighth data bit is the address bit in the 10-bit asynchronous and 11-bit asynchronous with parity modes. Thus, the received character is an address that has to be processed by all sleeping processors – i.e., each processor has to compare the received character with its own address and decide whether to receive or ignore all following characters. WAKE is cleared by hardware and software reset.

6.3.2.1.5 SCR Receiver Wakeup Enable (RWU) Bit 6

When RWU equals one and the SCI is in an asynchronous mode, the wakeup function is enabled – i.e., the SCI is put to sleep waiting for a reason (defined by the WAKE bit) to wakeup. In the sleeping state, all receive flags, except IDLE, and interrupts are disabled. When the receiver wakes up, this bit is cleared by the wakeup hardware. The programmer may also clear the RWU bit to wake up the receiver.

RWU can be used by the programmer to ignore messages that are for other devices on a multidrop serial network. Wakeup on idle line (WAKE=0) or wakeup on address bit (WAKE=1) must be chosen.

1. When WAKE equals zero and RWU equals one, the receiver will not respond to data on the data line until an idle line is detected.
2. When WAKE equals one and RWU equals one, the receiver will not respond to data on the data line until a data byte with bit 9 equal to one is detected.

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

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

RWU is cleared by hardware and software reset. RWU is a don't care in the synchronous mode.

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

When the WOMS bit is set, the SCI TXD driver is programmed to function as an open-drain output and may be wired together with other TXD pins in an appropriate bus configuration such as a master-slave multidrop configuration. An external pullup resistor is required on the bus. When the WOMS is cleared, the TXD pin uses an active internal pullup. This bit is cleared by hardware and software reset.

6.3.2.1.7 SCR Receiver Enable (RE) Bit 8

When RE is set, the receiver is enabled. When RE is cleared, the receiver is disabled, and data transfer is inhibited to the receive data register (SRX) from the receive shift register. If RE is cleared while a character is being received, the reception of the character will be completed before the receiver is disabled. RE does not inhibit RDRF or receive interrupts. RE is cleared by a hardware and software reset.

6.3.2.1.8 SCR Transmitter Enable (TE) Bit 9

When TE is set, the transmitter is enabled. When TE is cleared, the transmitter will complete transmission of data in the SCI transmit data shift register; then the serial output is forced high (idle). Data present in the SCI transmit data register (STX) will not be transmitted. STX may be written and TDRE will be cleared, but the data will not be transferred into the shift register. TE does not inhibit TDRE or transmit interrupts. TE is cleared by a hardware and software reset.

Setting TE will cause the transmitter to send a preamble of 10 or 11 consecutive ones (depending on WDS). This procedure gives the programmer a convenient way to ensure that the line goes idle before starting a new message. To force this separation of messages by the minimum idle line time, the following sequence is recommended:

1. Write the last byte of the first message to STX
2. Wait for TDRE to go high, indicating the last byte has been transferred to the transmit shift register

3. Clear TE and set TE back to one. This queues an idle line preamble to immediately follow 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 the STX prior to the finish of the preamble transmission, then the transmit data line will simply mark idle until STX is finally written.

6.3.2.1.9 SCR Idle Line Interrupt Enable (ILIE) Bit 10

When ILIE is set, the SCI interrupt occurs when IDLE is set. When ILIE is clear, the IDLE interrupt is disabled. ILIE is cleared by hardware and software reset.

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

When a valid start bit has been received, an idle interrupt will be generated if both IDLE (SCI Status Register bit 3) and ILIE equals one. The idle interrupt acknowledge from the interrupt controller clears this interrupt request. The idle interrupt will not be asserted again until at least one character has been received. The result is as follows:

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

6.3.2.1.10 SCR SCI Receive Interrupt Enable (RIE) Bit 11

The RIE bit is used to enable the SCI receive data interrupt. If RIE is cleared, receive interrupts are disabled, and the RDRF bit in the SCI status register must be polled to determine if the receive data register is full. If both RIE and RDRF are set, the SCI will request an SCI receive data interrupt from the interrupt controller.

One of two possible receive data interrupts will be requested:

1. Receive without exception will be requested if PE, FE, and OR are all clear (i.e., a normal received character).
2. Receive with exception will be requested if PE, FE, and OR are not all clear (i.e., a received character with an error condition).

RIE is cleared by hardware and software reset.

6.3.2.1.11 SCR SCI Transmit Interrupt Enable (TIE) Bit 12

The TIE bit is used to enable the SCI transmit data interrupt. If TIE is cleared, transmit data interrupts are disabled, and the transmit data register empty (TDRE) bit in the SCI status register must be polled to determine if the transmit data register is empty. If both TIE and TDRE are set, the SCI will request an SCI transmit data interrupt from the interrupt controller. TIE is cleared by hardware and software reset.

6.3.2.1.12 SCR Timer Interrupt Enable (TMIE) Bit 13

The TMIE bit is used to enable the SCI timer interrupt. If TMIE is set (enabled), the timer interrupt requests will be made 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 clock 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 pins be configured for SCI use to operate. TMIE is cleared by hardware and software reset.

6.3.2.1.13 SCR SCI Timer Interrupt Rate (STIR) Bit 14

This bit controls a divide by 32 in the SCI Timer interrupt generator. When this bit is cleared, the divide by 32 is inserted in the chain. When the bit is set, the divide by 32 is bypassed, thereby increasing the timer resolution by 32 times. This bit is cleared by hardware and software reset.

6.3.2.1.14 SCR SCI Clock Polarity (SCKP) Bit 15

The clock polarity, sourced or received on the clock pin (SCLK), can be inverted using this bit, eliminating the need for an external inverter. When bit 15 equals zero, the clock polarity is positive; when bit 15 equals one, the clock polarity is negative. In the synchronous mode, positive polarity means that the clock is normally positive and transitions negative during data valid; whereas, negative polarity means that the clock is normally negative and transitions positive during valid data. In the 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. SCKP is cleared on hardware and software reset.

6.3.2.2 SCI Status Register (SSR)

The SSR is an 8-bit read-only register used by the DSP CPU to determine the status of the SCI. When the SSR is read onto the internal data bus, the register contents occupy the low-order byte of the data bus and all high-order portions are zero filled. The status bits are described in the following paragraphs.

6.3.2.2.1 SSR Transmitter Empty (TRNE) Bit 0

The TRNE flag is set when both the transmit shift register and data register are empty to indicate that there is no data in the transmitter. When TRNE is set, data written to one of the three STX locations or to the STXA will be transferred to the transmit shift register and be the first data transmitted. TRNE is cleared when TDRE is cleared by writing data into the transmit data register (STX) or the transmit data address register (STXA), or when an idle, preamble, or break is transmitted. The purpose of this bit is to indicate that the transmitter is empty; therefore, the data written to STX or STXA will be transmitted next – i.e., there is not a word in the transmit shift register presently being transmitted. This procedure is useful when initiating the transfer of a message (i.e., a string of characters). TRNE is set by the hardware, software, SCI individual, and stop reset.

6.3.2.2.2 SSR Transmit Data Register Empty (TDRE) Bit 1

The TDRE bit is set when the SCI transmit data register is empty. When TDRE is set, new data may be written to one of the SCI transmit data registers (STX) or transmit data address register (STXA). TDRE is cleared when the SCI transmit data register is written. TDRE is set by the hardware, software, SCI individual, and stop reset.

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

In the SCI asynchronous mode, the TDRE flag is not set immediately after a word is transferred from the STX or STXA to the transmit shift register nor when the word first begins to be shifted out. TDRE is set two cycles of the 16× clock after the start bit – i.e., two 16× clock cycles into to transmission time of the first data bit.

6.3.2.2.3 SSR Receive Data Register Full (RDRF) Bit 2

The RDRF bit is set when a valid character is transferred to the SCI receive data register from the SCI receive shift register. RDRF is cleared when the SCI receive data register is read or by the hardware, software, SCI individual, and stop reset.

6.3.2.2.4 SSR Idle Line Flag (IDLE) Bit 3

IDLE is 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 zero to one can cause an IDLE interrupt (ILIE). IDLE is cleared by the hardware, software, SCI individual, and stop reset.

6.3.2.2.5 SSR Overrun Error Flag (OR) Bit 4

The OR flag is set when a byte is ready to be transferred from the receive shift register to the receive data register (SRX) that is already full (RDRF=1). The receive shift register data is not transferred to the SRX. The OR flag indicates that character(s) in the receive 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 – i.e., overrun error has higher priority than FE or PE. OR is cleared by the hardware, software, SCI individual, and stop reset.

6.3.2.2.6 SSR Parity Error (PE) Bit 5

In the 11-bit asynchronous modes, the PE bit is set when an incorrect parity bit has been detected in the received character. It is set simultaneously with RDRF for the byte which contains the parity error – i.e., when the received word is transferred to the SRX. If PE is set, it does not inhibit further data transfer into the SRX. PE is cleared when the SCI status register is read, followed by a read of SRX. PE is also cleared by the hardware, software, SCI individual, or stop reset. In the 10-bit asynchronous mode, the 11-bit multidrop mode, and the 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 will only recognize the overrun error.

6.3.2.2.7 SSR Framing Error Flag (FE) Bit 6

The FE bit is set in the asynchronous modes when no stop bit is detected in the data string received. FE and RDRE are set simultaneously – i.e., when the received word is transferred to the SRX. However, the FE flag inhibits further transfer of data into the SRX until it is cleared. FE is cleared when the SCI status register is read followed by reading the SRX. The hardware, software, SCI individual, and stop reset also clear FE. In the 8-bit synchronous mode, FE is always cleared. If the byte received causes both framing and overrun errors, the SCI receiver will only recognize the overrun error.

6.3.2.2.8 SSR Received Bit 8 Address (R8) Bit 7

In the 11-bit asynchronous multidrop mode, the R8 bit is used to indicate whether the received byte is an address or data. R8 is not affected by reading the SRX or status register. The hardware, software, SCI individual, and stop reset clear R8.

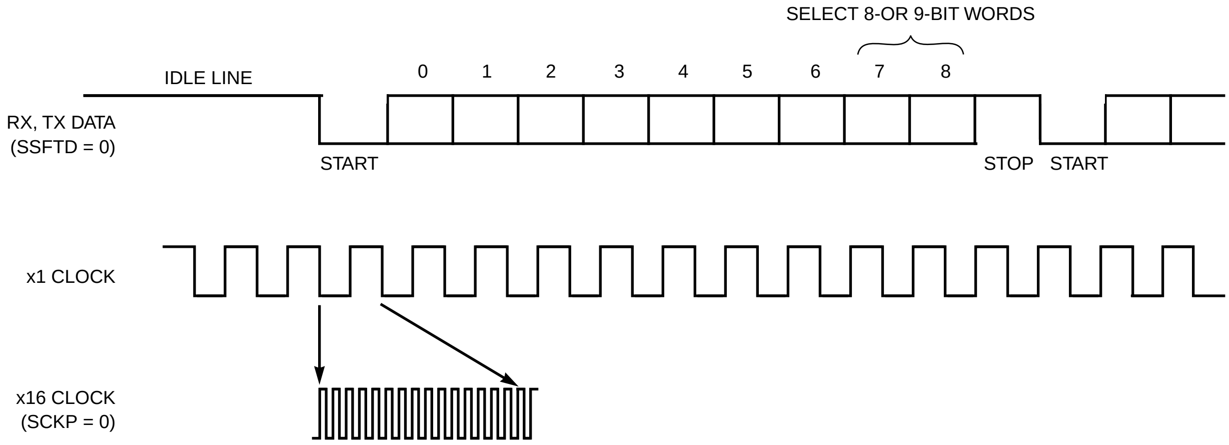


Figure 6-11 16x Serial Clock

6.3.2.3 SCI Clock Control Register (SCCR)

The SCCR is a 16-bit read/write register which controls the selection of the clock modes and baud rates for the transmit and receive sections of the SCI interface. The control bits are described in the following paragraphs. The SCCR is cleared by hardware reset.

The basic points of the clock generator are as follows:

1. The SCI core always uses a 16 × internal clock in the asynchronous modes and always uses a 2 × internal clock in the synchronous mode. The maximum internal clock available to the SCI peripheral block is the oscillator frequency divided by 4. With a 40-MHz crystal, this gives a maximum data rate of 625 Kbps for asynchonous data and 5 Mbps for synchronous data. These maximum rates are the same for internally or externally supplied clocks.

2. The 16 × clock is necessary for the asynchronous modes to synchronize the SCI to the incoming data (see Figure 6-11).

3. For the asynchronous modes, the user must provide a 16 × clock if he wishes to use an external baud rate generator (i.e., SCLK input).

4. For the asynchronous modes, the user may select either 1 × or 16 × for the output clock when using internal TX and RX clocks (TCM=0 and RCM=0).

5. The transmit data on the TXD pin changes on the negative edge of the 1 × serial clock and is stable on the positive edge (SCKP=0). For SCKP equals one, the data changes on the positive edge and is stable on the negative edge.

6. The receive data on the RXD pin is sampled on the positive edge (if SCKP=0) or on the negative edge (if SCKP=1) of the 1 × serial clock.

7. For the asynchronous mode, the output clock is continuous.
8. For the synchronous mode, a 1 × clock is used for the output or input baud rate. The maximum 1 × clock is the crystal frequency divided by 8.
9. For the synchronous mode, the clock is gated.
10. For both the asynchronous and synchronous modes, the transmitter and receiver are synchronous with each other.

6.3.2.3.1 SCCR Clock Divider (CD11–CD0) Bits 11–0

The clock divider bits (CD11–CD0) are used to preset a 12-bit counter, which is decremented at the I_{cyc} rate (crystal frequency divided by 2). The counter is not accessible to the user. When the counter reaches zero, it is reloaded from the clock divider bits. Thus, a value of 0000 0000 0000 in CD11–CD0 produces the maximum rate of I_{cyc}, and a value of 0000 0000 0001 produces a rate of $I_{cyc}/2$. The lowest rate available is $I_{cyc}/4096$. Figure 6-12 and Figure 6-35 show the clock dividers. Bits CD11–CD0 are cleared by hardware and software reset.

6.3.2.3.2 SCCR Clock Out Divider (COD) Bit 12

Figure 6-12 and Figure 6-35 show the clock divider circuit. The 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, and

1. If COD equals zero and SCLK is an output (i.e., TCM and RCM=0), the SCI clock is divided by 16 before being output to the SCLK pin; thus, the SCLK output is a 1 × clock
2. If COD equals one and SCLK is an output, the SCI clock is fed directly out to the SCLK pin; thus, the SCLK output is a 16 × baud clock

The COD bit is cleared by hardware and software reset.

6.3.2.3.3 SCCR SCI Clock Prescaler (SCP) Bit 13

The SCI SCP bit selects a divide by 1 (SCP=0) or divide by 8 (SCP=1) prescaler for the clock divider. The output of the prescaler is further divided by 2 to form the SCI clock. Hardware and software reset clear SCP. Figure 6-12 and Figure 6-35 show the clock divider diagram.

6.3.2.3.4 SCCR Receive Clock Mode Source (RCM) Bit 14

RCM selects internal or external clock for the receiver (see Figure 6-35). RCM equals zero selects the internal clock; RCM equals one selects the external clock from the SCLK pin. Hardware and software reset clear RCM.

6.3.2.3.5 SCCR Transmit Clock Source (TCM) Bit 15

The TCM bit selects internal or external clock for the transmitter (see Figure 6-35). TCM equals zero selects the internal clock; TCM equals one selects the external clock from the SCLK pin. Hardware and software reset clear TCM.

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

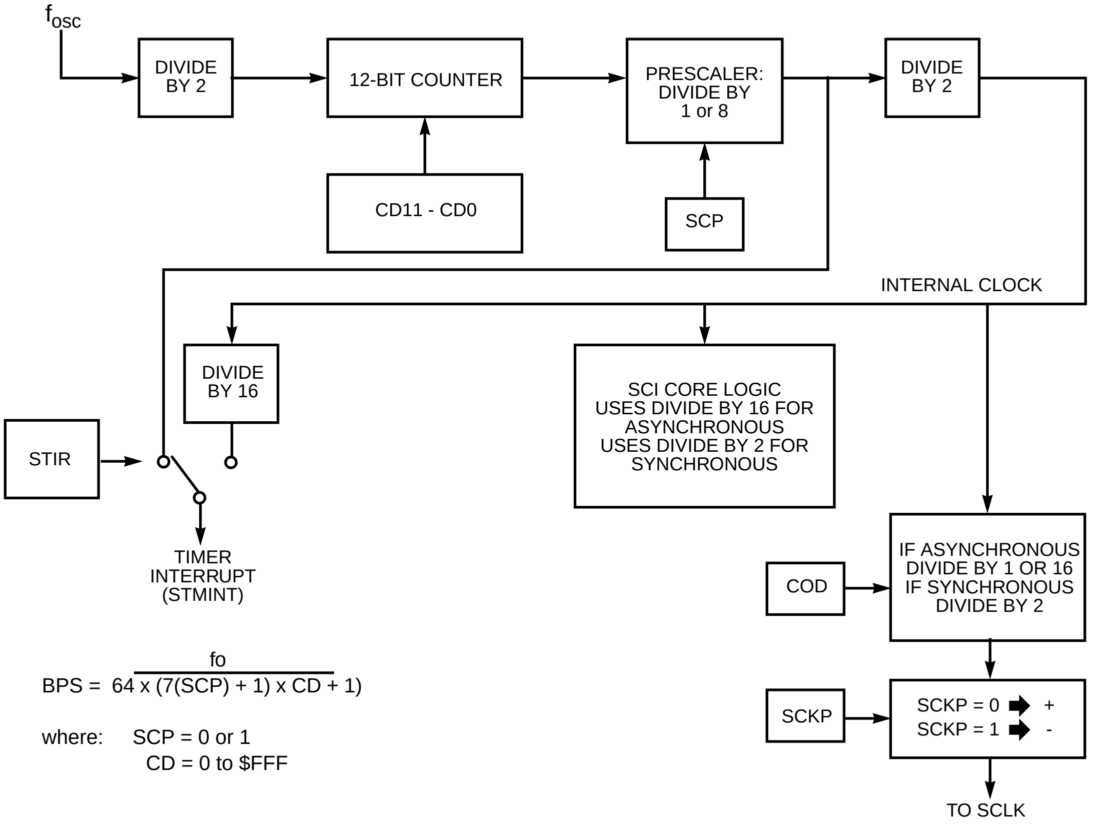

f_{OSC}
DIVIDE BY 2
12-BIT COUNTER
PRESCALER: DIVIDE BY 1 or 8
DIVIDE BY 2
CD11 - CD0
SCP
INTERNAL CLOCK
DIVIDE BY 16
SCI CORE LOGIC USES DIVIDE BY 16 FOR ASYNCHRONOUS USES DIVIDE BY 2 FOR SYNCHRONOUS
STIR
TIMER INTERRUPT (STMINT)
COD
IF ASYNCHRONOUS DIVIDE BY 1 OR 16 IF SYNCHRONOUS DIVIDE BY 2
$BPS = \frac{fo}{64 \times (7(SCP) + 1) \times CD + 1)}$
where: SCP = 0 or 1
CD = 0 to $FFF
SCKP
SCKP = 0 ➡ +
SCKP = 1 ➡ -
TO SCLK

Figure 6-12 SCI Baud Rate Generator

6.3.2.4 SCI Data Registers

The SCI data registers are divided into two groups: receive and transmit. 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.

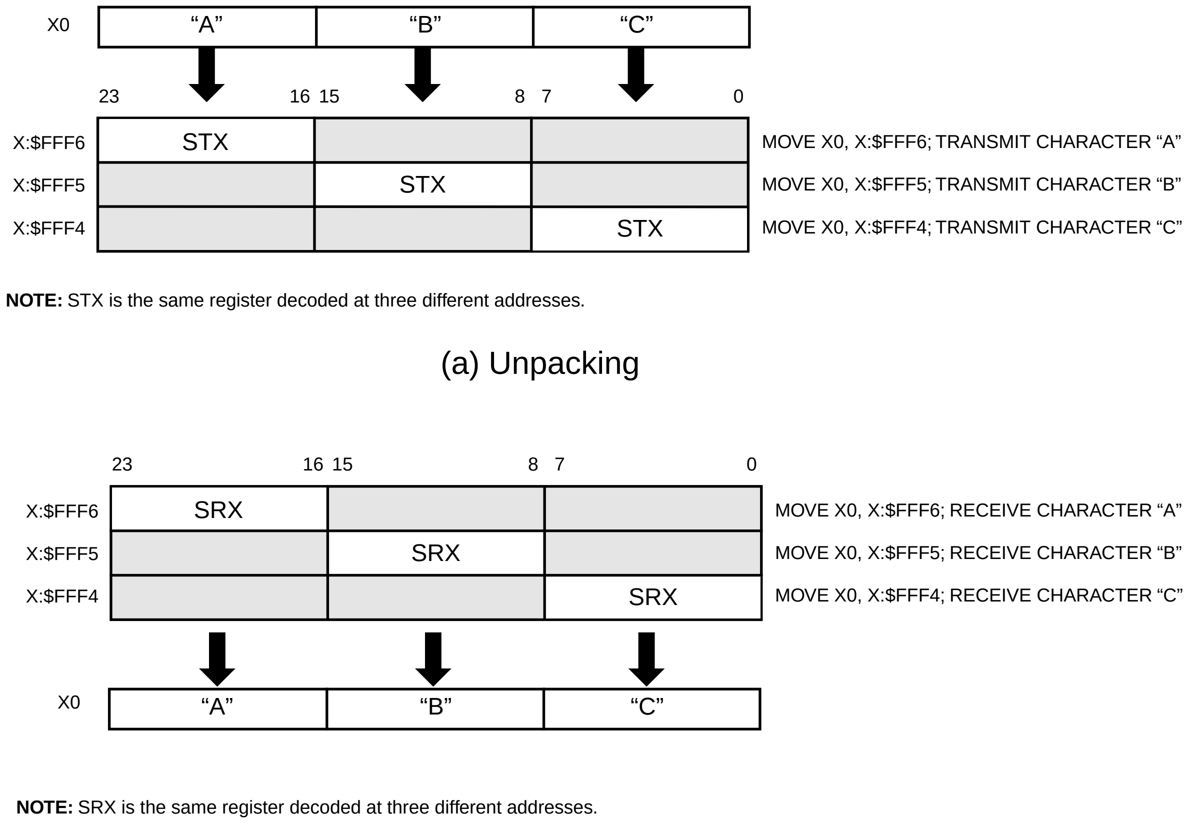
6.3.2.4.1 SCI Receive Register

Data words received on the RXD pin are shifted into the SCI receive shift register. When the complete word has been received, the data portion of the word is transferred to the byte-wide SRX. This process converts the serial data to parallel data and provides double buffering. Double buffering provides flexibility and increased throughput since the programmer can save the previous word while the current word is being received.

The SRX can be read at three locations: X:$FFF4, X:$FFF5, and X:$FFF6 (see Figure 6-13). When location X:$FFF4 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 written as zeros. Similarly, when X:$FFF5 is read, the contents of SRX are placed in the middle byte of the bus, and when X:$FFF6 is read, the contents of SRX are placed in the high byte with the remaining bits zeroed. Mapping SRX as described allows three bytes to be efficiently packed into one 24-bit word by “OR”-ing three data bytes read from the three addresses. The following code fragment requires that R0 initially points to X:$FFF4, register A is initially cleared, and R3 points to a data buffer. The only programming trick is using BCLR to test bit 1 of the packing pointer to see if it is pointing to X:$FFF6 and clearing bit 1 to point to X:$FFF4 if it had been pointing to X:$FFF6. This procedure resets the packing pointer after receiving three bytes.

```
      MOVE   X:(R0),X0   ;Copy received data to temporary register
      BCLR   #$1,R0      ;Test for last byte
                         ;reset pointer if it is the last byte
      OR     X0,A        ;Pack the data into register A
      MOVE   (R0)+       ;and increment the packing pointer
      JCS    FLAG        ;Jump to clean up routine if last byte
      RTI                ;Else return until next byte is received
FLAG  MOVE   A,(R3)+     ;Move the packed data to memory
      CLR    A           ;Prepare A for packing next three bytes
      RTI                ;Return until the next byte is received
```

The length and format of the serial word is defined by the WDS0, WDS1, and WDS2 control bits in the SCI control register. In the synchronous modes, the start bit, the eight data bits with LSB first, the address/data indicator bit and/or the parity bit, and the stop bit are received in that order for SSFTD equals zero (see Figure 6-10 (a)). For SSFTD equals one, the data bits are transmitted MSB first (see Figure 6-10(b)). The clock source is defined by the receive clock mode (RCM) select bit in the SCR. In the synchronous mode, the synchronization is provided by gating the clock. In either mode, when a complete word has been clocked in, the contents of the shift register can be transferred to the SRX and the flags; RDRF, FE, PE, and OR are changed appropriately. Because the operation of the SCI receive shift register is transparent to the DSP, the contents of this register are not directly accessible to the programmer.


X0
“A”
“B”
“C”
23
16
15
8
7
0
X:$FFF6
STX
MOVE X0, X:$FFF6; TRANSMIT CHARACTER “A”
X:$FFF5
STX
MOVE X0, X:$FFF5; TRANSMIT CHARACTER “B”
X:$FFF4
STX
MOVE X0, X:$FFF4; TRANSMIT CHARACTER “C”
23
16
15
8
7
0
X:$FFF6
SRX
MOVE X0, X:$FFF6; RECEIVE CHARACTER “A”
X:$FFF5
SRX
MOVE X0, X:$FFF5; RECEIVE CHARACTER “B”
X:$FFF4
SRX
MOVE X0, X:$FFF4; RECEIVE CHARACTER “C”
X0
“A”
“B”
“C”

NOTE: STX is the same register decoded at three different addresses.

(a) Unpacking

NOTE: SRX is the same register decoded at three different addresses.

(b) Packing

Figure 6-13 Data Packing and Unpacking

6.3.2.4.2 SCI Transmit Registers

The transmit data register is one byte-wide register mapped into four addresses: X:$FFF3, X:$FFF4, X:$FFF5, and X:$FFF6. In the asynchronous mode, when data is to be transmitted, X:$FFF4, X:$FFF5, and X:$FFF6 are used, and the register is called STX. When X:$FFF4 is written, the low byte on the data bus is transferred to the STX; when X:$FFF5 is written, the middle byte is transferred to the STX; and when X:$FFF6 is written, the high byte is transferred to the STX. This structure (see Figure 6-9) makes it easy for the programmer to unpack the bytes in a 24-bit word for transmission. Location X:$FFF3 should be written in the 11-bit asynchronous multidrop mode when the data is an address and it is desired that the ninth bit (the address bit) be set. When X:$FFF3 is written, the transmit data register is called STXA, and data from the low byte on the data bus is stored in STXA. The address data bit will be cleared in the 11-bit asynchronous multidrop mode when any of X:$FFF4, X:$FFF5, or X:$FFF6 is written. When either STX or STXA is written, TDRE is cleared.

The transfer from either STX or STXA to the transmit shift register occurs automatically, but not immediately, when the last bit from the previous word has been shifted out – i.e., the transmit shift register is empty. Like the receiver, the transmitter is double buffered. However, there will be a two to four serial clock cycle delay 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 pin. (A serial clock cycle is the time required to transmit one data bit). The transmit shift register is not directly addressable, and a dedicated flag for this register does not exist. Because of this fact and the two to four cycle delay, two bytes cannot be written consecutively to STX or STXA without polling. The second byte will overwrite the first byte. The TDRE flag should always be polled prior to writing STX or STXA to prevent overruns unless transmit interrupts have been enabled. Either STX or STXA is usually written as part of the interrupt service routine. Of course, the interrupt will only be generated if TDRE equals one. The transmit shift register is indirectly visible via the TRNE bit in the SSR.

In the synchronous modes, data is synchronized with the transmit clock, which may 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 the asynchronous modes, the start bit, the eight data bits (with the LSB first if SSFTD=0 and the MSB first if SSFTD=1), the address/data indicator bit or parity bit, and the stop bit are transmitted in that order (see Figure 6-10).

The data to be transmitted can be written to any one of the three STX addresses. If SCKP equals one and SSHTD equals one, the SCI synchronous mode is equivalent to the SSI operation in the 8-bit data on-demand mode.

6.3.2.5 Preamble, Break, and Data Transmission Priority

It is possible that two or three transmission commands are set simultaneously:

1. A preamble (TE was toggled)
2. A break (SBK was set or was toggled)
3. There is data for transmission (TDRE=0)

After the current character transmission, if two or more of these commands are set, the transmitter will execute them in the following priority:

1. Preamble
2. Break
3. Data

6.3.3 Register Contents After Reset

There are four methods to reset the SCI. Hardware or software reset clears the port control register bits, which configure all I/O as general-purpose input. The SCI will remain in the reset state while all SCI pins are programmed as general-purpose I/O (CC2, CC1, and CC0=0); the SCI will become active only when at least one of the SCI I/O pins is programmed as not general-purpose I/O.

During program execution, the CC2, CC1, and CC0 bits may be cleared (individual reset), which will cause the SCI to stop serial activity and enter the reset state. All SCI status bits will be set to their reset state; however, the contents of the interface control register are not affected, allowing the DSP program to reset the SCI separately from the other internal peripherals.

The STOP instruction halts operation of the SCI until the DSP is restarted, causing the SSR to be reset. No other SCI registers are affected by the STOP instruction. Table 6-2 illustrates how each type of reset affects each register in the SCI.

6.3.4 SCI Initialization

The correct way to initialize the SCI is as follows:

1. Hardware or software reset
2. Program SCI control registers
3. Configure SCI pins (at least one) as not general-purpose I/O

Table 6-2 SCI Registers after Reset

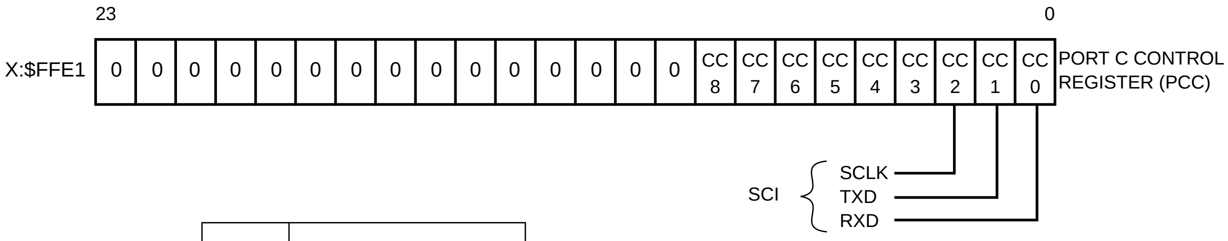
Register Bit	Bit Mnemonic	Bit Number	Reset Type			
			HW Reset	SW Reset	IR Reset	ST Reset
SCR	SCKP	15	0	0	–	–
	STIR	14	0	0	–	–
	TMIE	13	0	0	–	–
	TIE	12	0	0	–	–
	RIE	11	0	0	–	–
	ILIE	10	0	0	–	–
	TE	9	0	0	–	–
	RE	8	0	0	–	–
	WOMS	7	0	0	–	–
	RWU	6	0	0	–	–
	WAKE	5	0	0	–	–
	SBK	4	0	0	–	–
	SSFTD	3	0	0	–	–
	WDS (2–0)	2–0	0	0	–	–
SSR	R8	7	0	0	0	0
	FE	6	0	0	0	0
	PE	5	0	0	0	0
	OR	4	0	0	0	0
	IDLE	3	0	0	0	0
	RDRF	2	0	0	0	0
	TDRE	1	1	1	1	1
	TRNE	0	1	1	1	1
SCCR	TCM	15	0	0	–	–
	RCM	14	0	0	–	–
	SCP	13	0	0	–	–
	COD	12	0	0	–	–
	CD (11–0)	11–0	0	0	–	–
SRX	SRX (23–0)	23–16, 15–8, 7–0	–	–	–	–
STX	STX (23–0)	23–0	–	–	–	–
SRSH	SRS (8–0)	8–0	–	–	–	–
STSH	STS (8–0)	8–0	–	–	–	–

NOTES:
SRSH – SCI receive shift register, STSH – SCI transmit shift register
HW – Hardware reset is caused by asserting the external RESET pin.
SW – Software reset is caused by executing the RESET instruction.
IR – Individual reset is caused by clearing PCC (bits 0–2) (configured for general-purpose I/O).
ST – Stop reset is caused by executing the STOP instruction.
1 – The bit is set during the xx reset.
0 – The bit is cleared during the xx reset.
– – The bit is not changed during the xx reset.

1. PERFORM HARDWARE OR SOFTWARE RESET.

2. PROGRAM SCI CONTROL REGISTERS:
   a) SCI INTERFACE CONTROL REGISTER — X:$FFF0
   b) SCI CLOCK CONTROL REGISTER — X:$FFF2

3. CONFIGURE AT LEAST ONE PORT C CONTROL BIT AS SCI.



CCx	Function
0	GPIO
1	Serial Interface

4. SCI IS NOW ACTIVE.

Figure 6-14 SCI Initialization Procedure

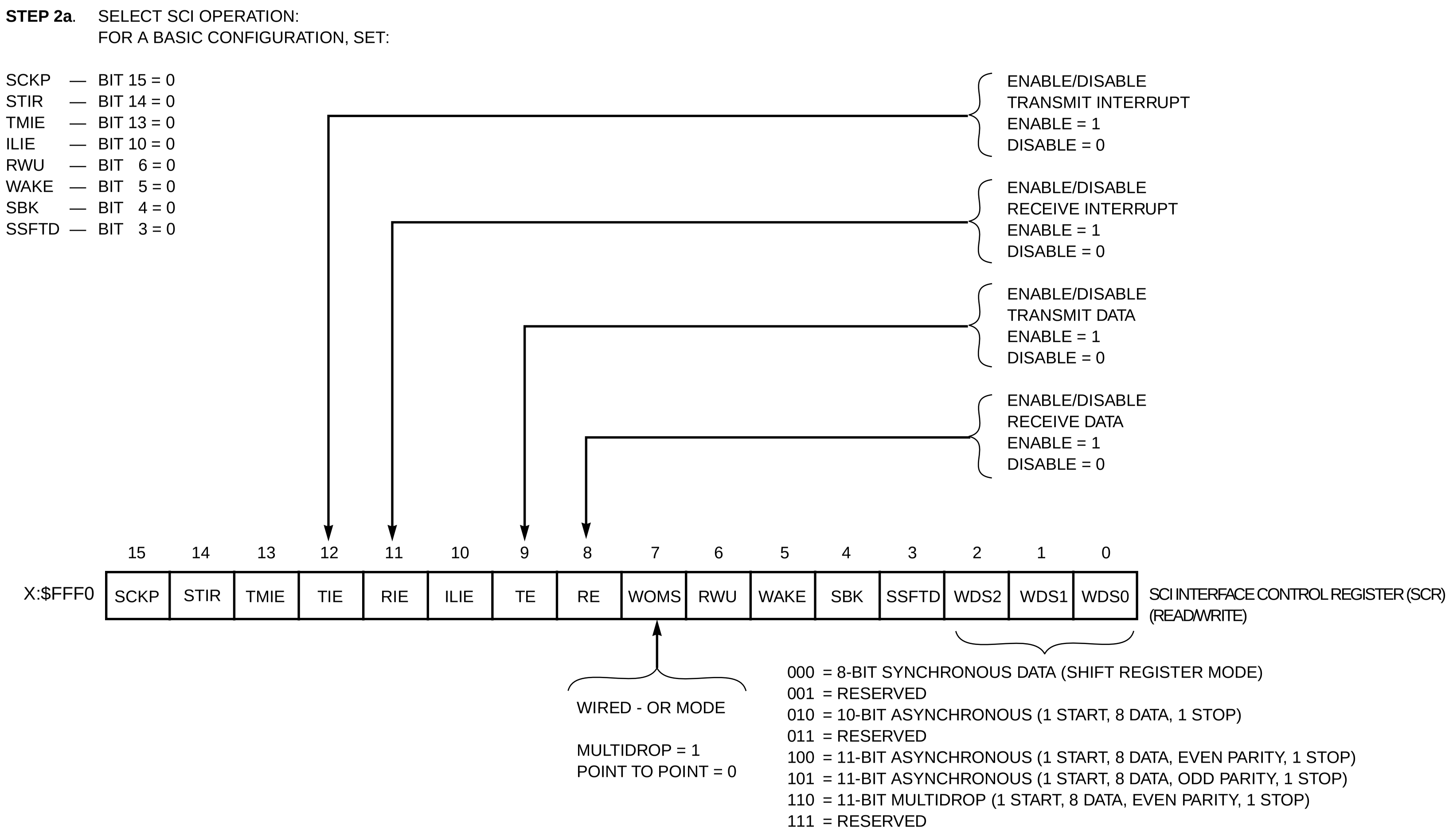
Figure 6-14 and Figure 6-15 show how to configure the bits in the SCI registers. Figure 6-14 is the basic initialization procedure showing which registers must be configured.

1. A hardware or software reset should be used to reset the SCI and prevent it from doing anything unexpected while it is being programmed
2. Both the SCI interface control register and the clock control register must be configured for any operation using the SCI
3. The pins to be used must then be selected to release the SCI from reset
4. Begin operation

If interrupts are to be used, the pins must be selected, and interrupts must be enabled and unmasked before the SCI will operate. The order does not matter; any one of these three requirements for interrupts can be used to finally enable the SCI.

Figure 6-15 shows the meaning of the individual bits in the SCR and SCCR. The figures below do not assume that interrupts will be used; they recommend selecting the appropriate pins to enable the SCI. Programs shown in Figures Figure 6-20, Figure 6-21, Figure 6-28, Figure 6-34, and Figure 6-36 control the SCI by enabling and disabling interrupts. Either method is acceptable.

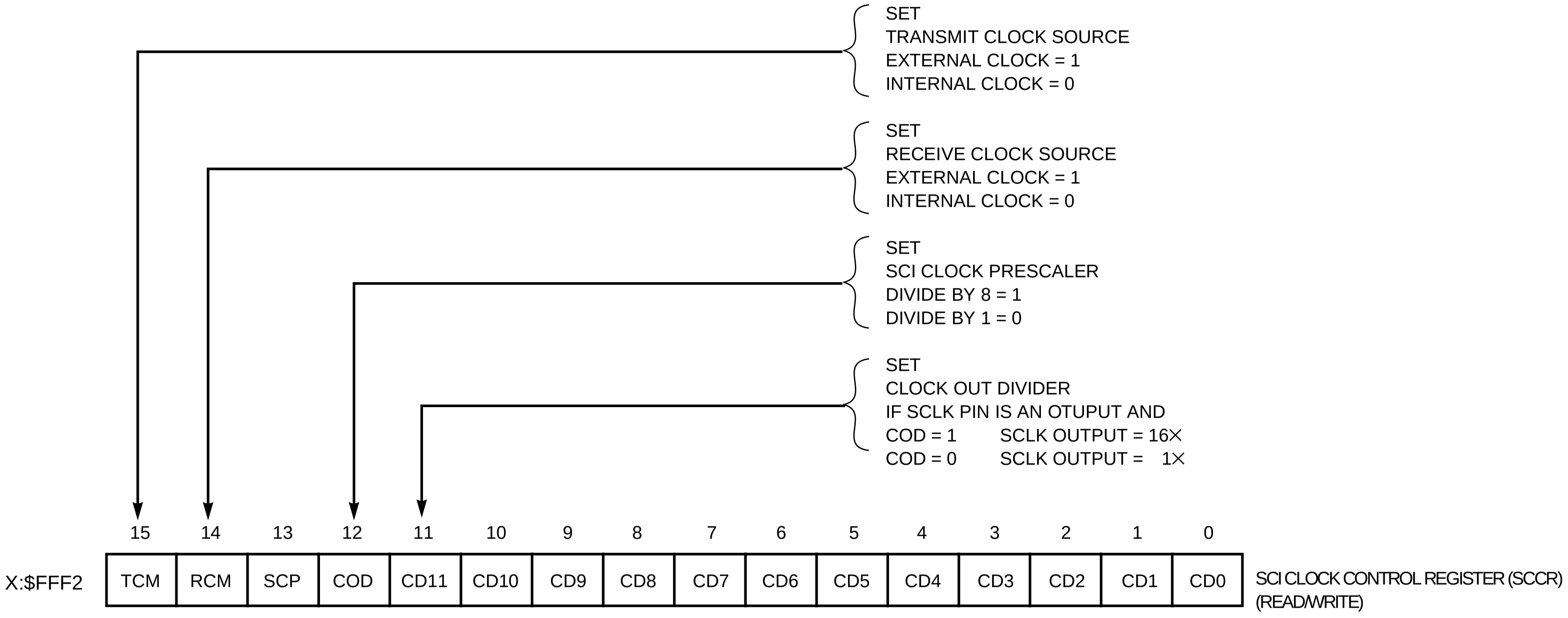
Table 6-3 (a) through Table 6-4 (b) provide the settings for common baud rates for the SCI. The asynchronous SCI baud rates show a baud rate error for the fixed oscillator frequency (see Table 6-3 (a)). These small-percentage baud rate errors should allow most UARTs to synchronize. The synchronous applications usually require exact frequencies, which require that the crystal frequency be chosen carefully (see Table 6-4 (a) and Table 6-4 (b)).


STEP 2a. SELECT SCI OPERATION:
FOR A BASIC CONFIGURATION, SET:
SCKP — BIT 15 = 0
STIR — BIT 14 = 0
TMIE — BIT 13 = 0
ILIE — BIT 10 = 0
RWU — BIT 6 = 0
WAKE — BIT 5 = 0
SBK — BIT 4 = 0
SSFTD — BIT 3 = 0
ENABLE/DISABLE TRANSMIT INTERRUPT ENABLE = 1 DISABLE = 0
ENABLE/DISABLE RECEIVE INTERRUPT ENABLE = 1 DISABLE = 0
ENABLE/DISABLE TRANSMIT DATA ENABLE = 1 DISABLE = 0
ENABLE/DISABLE RECEIVE DATA ENABLE = 1 DISABLE = 0
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
X:$FFF0
SCKP STIR TMIE TIE RIE ILIE TE RE WOMS RWU WAKE SBK SSFTD WDS2 WDS1 WDS0
SCI INTERFACE CONTROL REGISTER (SCR) (READ/WRITE)
WIRED - OR MODE
MULTIDROP = 1
POINT TO POINT = 0
000 = 8-BIT SYNCHRONOUS DATA (SHIFT REGISTER MODE)
001 = RESERVED
010 = 10-BIT ASYNCHRONOUS (1 START, 8 DATA, 1 STOP)
011 = RESERVED
100 = 11-BIT ASYNCHRONOUS (1 START, 8 DATA, EVEN PARITY, 1 STOP)
101 = 11-BIT ASYNCHRONOUS (1 START, 8 DATA, ODD PARITY, 1 STOP)
110 = 11-BIT MULTIDROP (1 START, 8 DATA, EVEN PARITY, 1 STOP)
111 = RESERVED

Step 2a

Figure 6-15 SCI General Initialization Detail – Step 2 (Sheet 1 of 2)

STEP 2b. SELECT CLOCK AND DATA RATE:
SET THE CLOCK DIVIDER BITS (CD0 - CD11) ACCORDING TO TABLES 11 - 2 OR 11 - 3.
SET THE SCI CLOCK PRESCALER BIT (SCP, BIT 13) ACCORDING TO TABLES 11 - 2 OR 11 - 3.



Step 2b

Figure 6-15 SCI General Initialization Detail – Step 2 (Sheet 2 of 2)

Table 6-3 (a) Asynchronous SCI Bit Rates for a 40-MHz Crystal

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Bit Rate Error, Percent
625.0K	0	$000	0
56.0K	0	$00A	+1.46
38.4K	0	$00F	+1.72
19.2K	0	$020	-1.36
9600	0	$040	+0.16
8000	0	$04D	+0.15
4800	0	$081	+0.15
2400	1	$020	-1.38
1200	1	$040	+0.08
600	1	$081	0
300	1	$103	0

BPS = f_0 ÷ (64 X (7 X (SCP) + 1) X (CD + 1)); f_0 = 40 MHz
SCP = 0 or 1
CD = 0 to $FFF

Table 6-3 (b) Frequencies for Exact Asynchronous SCI Bit Rates

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Crystal Frequency
9600	0	$040	39,936,000
4800	0	$081	39,936,000
2400	0	$103	39,936,000
1200	0	$207	39,936,000
300	0	$822	39,993,000
9600	1	$007	39,321,600
4800	1	$00F	39,321,600
2400	1	$01F	39,321,600
1200	1	$040	39,360,000
300	1	$103	39,936,000

f0 = BPS X 64 X (7 X (SCP) + 1) X (CD + 1))
SCP = 0 or 1
CD = 0 to $FFF

Table 6-4 (a) Synchronous SCI Bit Rates for a 32.768-MHz Crystal

Baud Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Baud Rate Error, Percent
4.096M	0	$000	0
128K	0	$01F	0
64K	0	$03F	0
56K	0	$048	-0.195
32K	0	$07F	0
16K	0	$0FF	0
8000	0	$1FF	0
4000	0	$3FF	0
2000	0	$7FF	0
1000	0	$FFF	0

BPS = $f_0 \div (8 \times (7 \text{ X } (SCP) + 1) \times (CD + 1))$; f_0 = 32.768 MHz
SCP = 0 or 1
CD = 0 to $FFF

Table 6-4 (b) Frequencies for Exact Synchronous SCI Bit Rates

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Crystal Frequency
2.048M	0	$001	32.768 MHz
1.544M	0	$002	37.056 MHz
1.536M	0	$002	36.864 MHz

f_0 = BPS $\times$ 8 $\times$ (7 X (SCP) + 1) $\times$ (CD + 1)
SCP = 0 or 1
CD = 0 to $FFF

An alternative to selecting the system clock to accommodate the SCI requirements is to provide an external clock to the SCI. For example, a 2.048 MHz bit rate requires a CPU clock of 32.768 MHz. An application may need a 40 MHz CPU clock and an external clock for the SCI.

6.3.5 SCI Exceptions

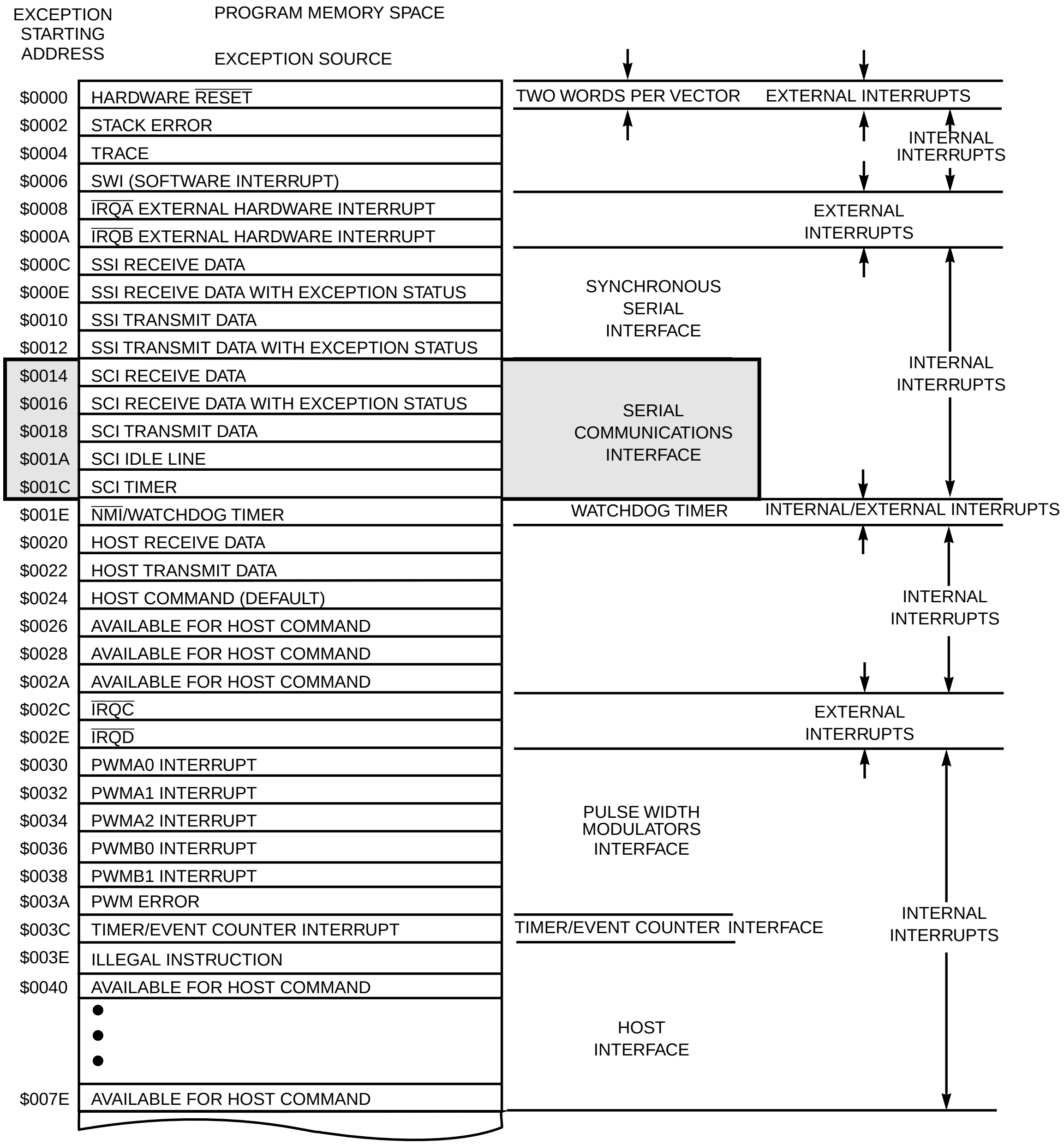
The SCI can cause five different exceptions in the DSP (see Figure 6-16). These exceptions are as follows:

1. SCI Receive Data – caused by receive data register full with no receive error conditions existing. This error-free interrupt may use a fast interrupt service routine for minimum overhead. This interrupt is enabled by SCR bit 11 (RIE).
2. SCI Receive Data with Exception Status – caused by receive data register full with a receiver error (parity, framing, or overrun error). The SCI status register must be read to clear the receiver error flag. A long interrupt service routine should be used to handle the error condition. This interrupt is enabled by SCR bit 11 (RIE).
3. SCI Transmit Data – caused by transmit data register empty. This error-free interrupt may use a fast interrupt service routine for minimum overhead. This interrupt is enabled by SCR bit 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 bit 10 (ILIE).
5. SCI Timer – caused by the baud rate counter underflowing. This interrupt is automatically reset when the interrupt is accepted. This interrupt is enabled by SCR bit 13 (TMIE).

6.3.6 Synchronous Data Mode

The synchronous mode (WDS=0, shift register mode) is designed to implement serial-to-parallel and parallel-to-serial conversions. This mode will directly interface to 8051/8096 synchronous (mode 0) buses as both a controller (master) or a peripheral (slave) and is compatible with the SSI mode if SCKP equals one. In synchronous mode, the clock is always common to the transmit and receive shift registers.

As a controller (synchronous master) shown in Figure 6-17, the DSP puts out a clock on the SCLK pin when data is present in the transmit shift register (a gated clock mode). The master mode is selected by choosing internal transmit and receive clocks (setting TCM and RCM=0). The example shows a 74HC165 parallel-to-serial shift register and 74HC164 serial-to-parallel shift register being used to convert eight bits of serial I/O to eight bits of parallel I/O. The load pulse latches eight bits into the 74HC165 and then SCLK shifts the RXD data into the SCI (these data bits are sample bits 0-7 in the timing diagram). At the same time, TXD shifts data out (B0-B7) to the 74HC164. When using the internal clock, data is transmitted when the transmit shift register is full. Data is valid on both edges of the output clock, which is compatible with an 8051 microprocessor. Received data is sampled in the middle of the clock low time if SCKP equals zero or in the middle of the clock high time if SCKP equals one.



EXCEPTION STARTING ADDRESS	PROGRAM MEMORY SPACE EXCEPTION SOURCE
$0000	HARDWARE $\overline{\text{RESET}}$
$0002	STACK ERROR
$0004	TRACE
$0006	SWI (SOFTWARE INTERRUPT)
$0008	$\overline{\text{IRQA}}$ EXTERNAL HARDWARE INTERRUPT
$000A	$\overline{\text{IRQB}}$ EXTERNAL HARDWARE INTERRUPT
$000C	SSI RECEIVE DATA
$000E	SSI RECEIVE DATA WITH EXCEPTION STATUS
$0010	SSI TRANSMIT DATA
$0012	SSI TRANSMIT DATA WITH EXCEPTION STATUS
$0014	SCI RECEIVE DATA
$0016	SCI RECEIVE DATA WITH EXCEPTION STATUS
$0018	SCI TRANSMIT DATA
$001A	SCI IDLE LINE
$001C	SCI TIMER
$001E	$\overline{\text{NMI}}$/WATCHDOG TIMER
$0020	HOST RECEIVE DATA
$0022	HOST TRANSMIT DATA
$0024	HOST COMMAND (DEFAULT)
$0026	AVAILABLE FOR HOST COMMAND
$0028	AVAILABLE FOR HOST COMMAND
$002A	AVAILABLE FOR HOST COMMAND
$002C	$\overline{\text{IRQC}}$
$002E	$\overline{\text{IRQD}}$
$0030	PWMA0 INTERRUPT
$0032	PWMA1 INTERRUPT
$0034	PWMA2 INTERRUPT
$0036	PWMB0 INTERRUPT
$0038	PWMB1 INTERRUPT
$003A	PWM ERROR
$003C	TIMER/EVENT COUNTER INTERRUPT
$003E	ILLEGAL INSTRUCTION
$0040	AVAILABLE FOR HOST COMMAND
	● ● ●
$007E	AVAILABLE FOR HOST COMMAND

Figure 6-16 HI Exception Vector Locations

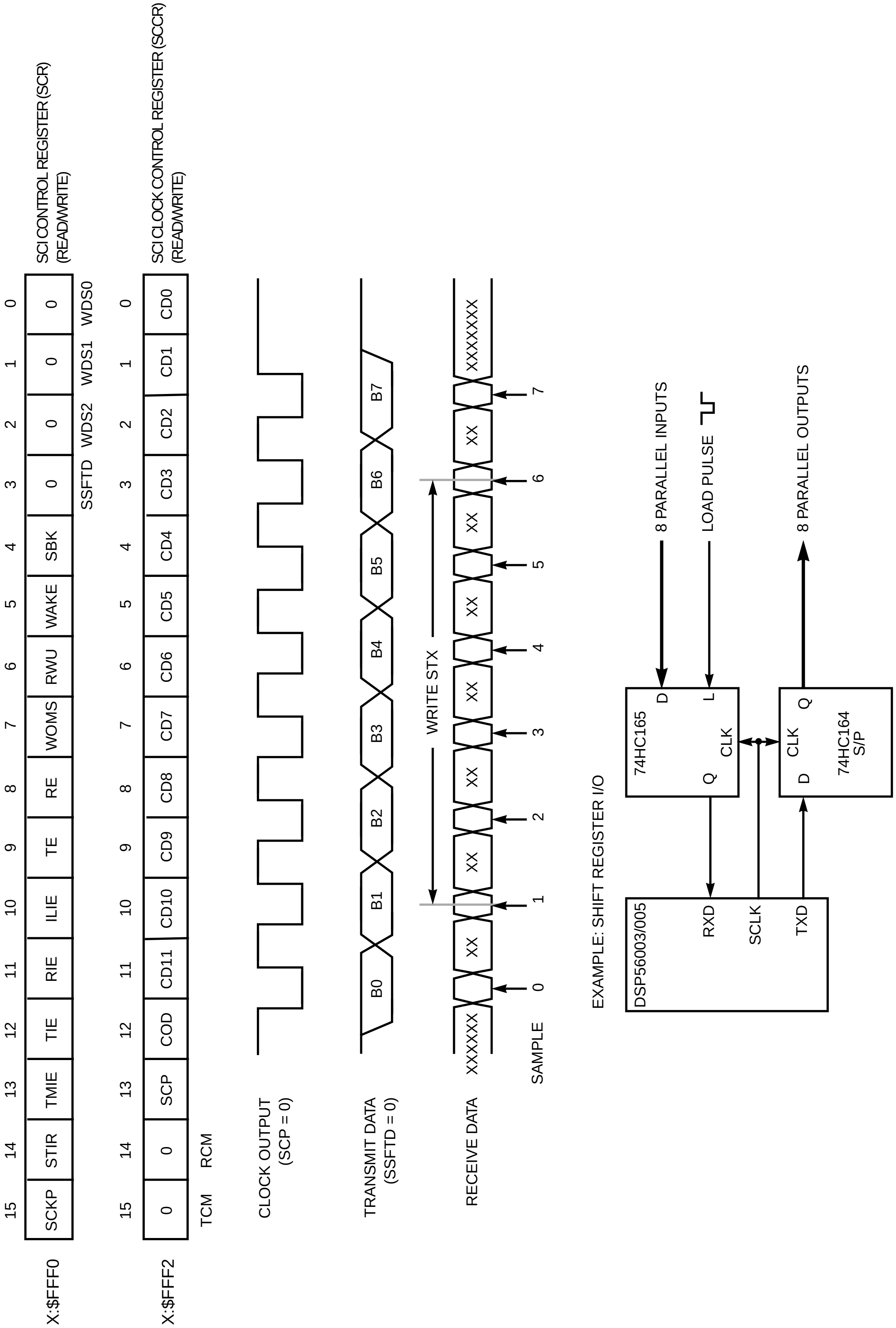


Figure 6-17 Synchronous Master

There is a window during which STX must be written with the next byte to be transmitted to prevent a gap between words. This window is from the time TDRE goes high halfway into transmission of bit 1 until the middle of bit 6 (see Figure 6-19(a)).

As a peripheral (synchronous slave) shown in Figure 6-18, the DSP accepts an input clock from the SCLK pin. If SCKP equals zero, data is clocked in on the rising edge of SCLK, and data is clocked out on the falling edge of SCLK. If SCKP equals one, data is clocked in on the falling edge of SCLK, and data is clocked out on the rising edge of SCLK. The slave mode is selected by choosing external transmit and receive clocks (TCM and RCM=1). Since there is no frame signal, if a clock is missed due to noise or any other reason, the receiver will lose synchronization with the data without any error signal being generated. Detecting an error of this type can be done 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.

The timing diagram in Figure 6-18 shows transmit data in the normal driven mode. Bit B7 is essentially one-half SCI clock long ($T_{SCI}/2 + 1.5\ T_{EXTAL}$) The last data bit is truncated so that the pin is guaranteed to go to its reset state before the start of the next data word, thereby delimiting data words. The 1.5 crystal clock cycles provide sufficient hold time to satisfy most external logic requirements. The example diagram requires that the WOMS bit be set in the SCR to wired-OR RXD and TXD, which causes TXD to be three-stated when not transmitting. Collisions (two devices transmitting simultaneously) must be avoided with this circuit by using a protocol such as alternating transmit and receive periods. In the example, the 8051 is the master device because it controls the clock. There is a window during which STX must be written with the next byte to be transmitted to prevent the current word from being retransmitted. This window is from the time TDRE goes high, which is halfway into the transmission of bit 1, until the middle of bit 6 (see Figure 6-19(b)). Of course, this assumes the clock remains continuous – i.e., there is a second word. If the clock stops, the SCI stops.

The DSP is initially configured according to the protocol to either receive data or transmit data. If the protocol determines that the next data transfer will be a DSP transmit, the DSP will configure the SCI for transmit and load STX (or STXA). When the master starts SCLK, data will be ready and waiting. If the protocol determines that the next data transfer will be a DSP receive, the DSP will configure the SCI for receive and will either poll the SCI or enable interrupts. This methodology allows multiple slave processors to use the same data line. Selection of individual slave processors can be under protocol control or by multiplexing SCLK.

Note: TCM=0, RCM=1 and TCM=1,RCM=0 are not allowed in the synchronous mode. The results are undefined.

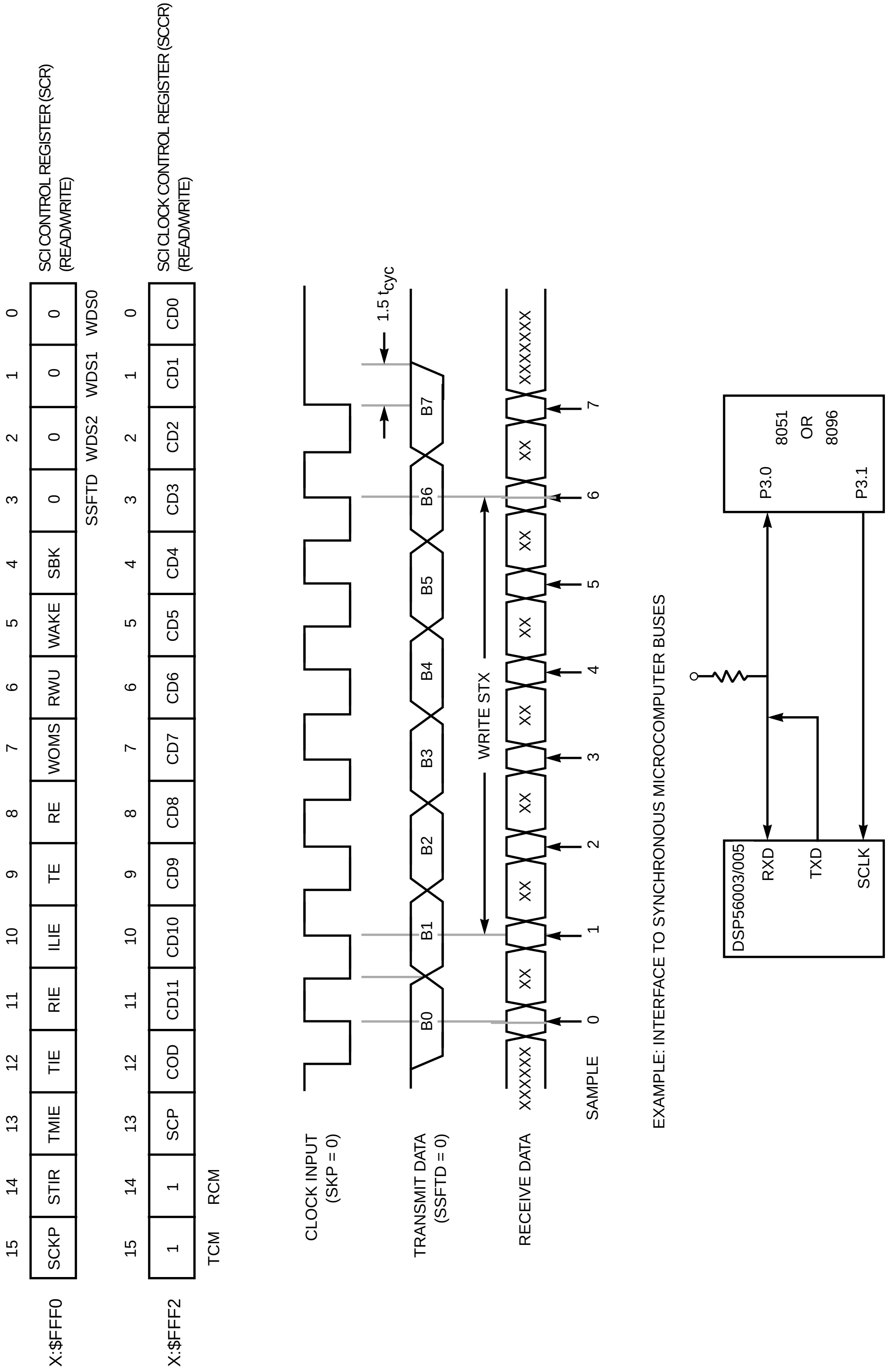
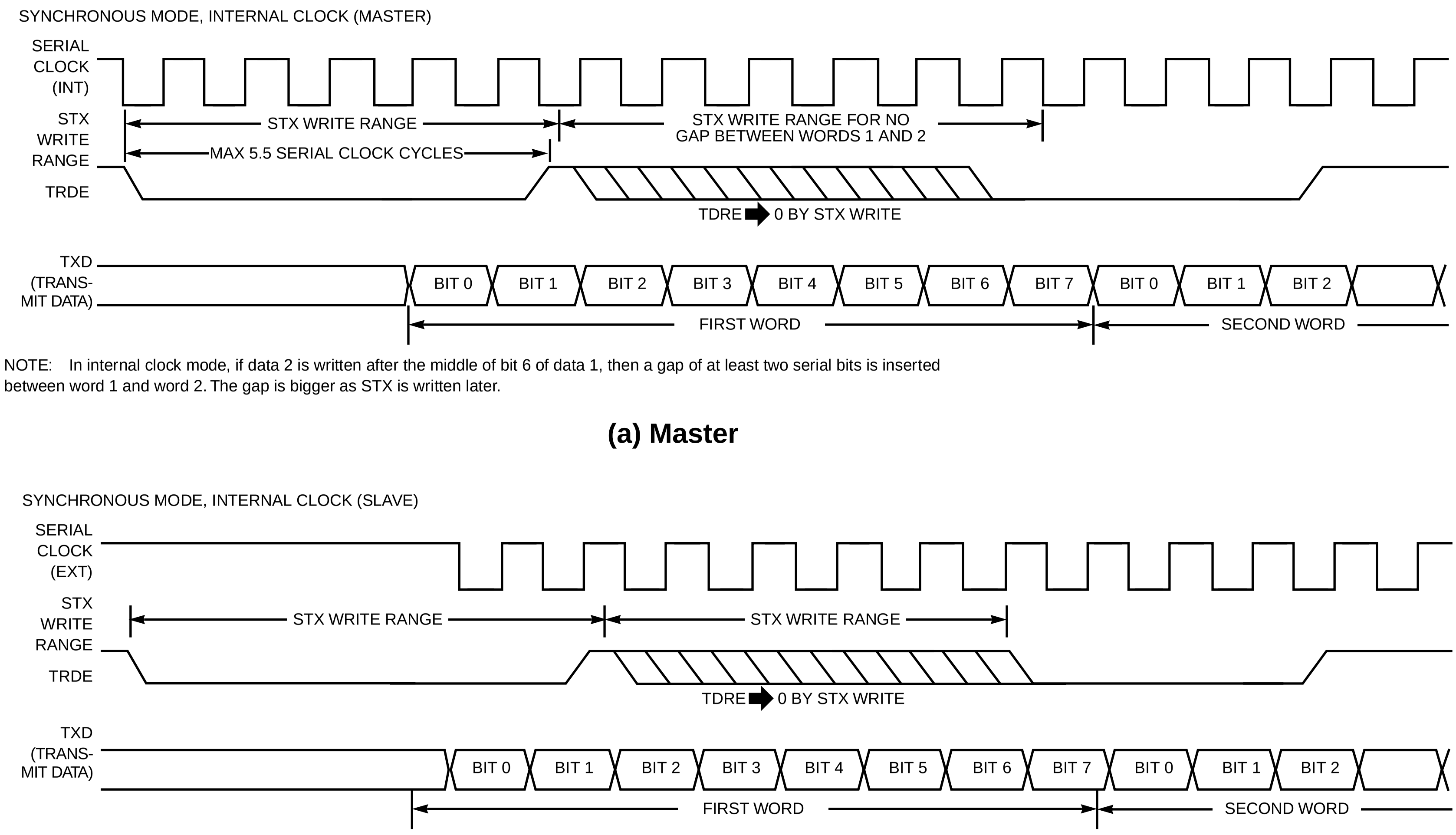


Figure 6-18 Synchronous Slave



NOTE: In internal clock mode, if data 2 is written after the middle of bit 6 of data 1, then a gap of at least two serial bits is inserted between word 1 and word 2. The gap is bigger as STX is written later.

(a) Master

NOTE: In external clock mode, if data 2 is written after the middle of bit 6 of data 1, then the previous data is retransmitted and data 2 is transmitted after the retransmission of data 1.

(b) Slave

Figure 6-19 Synchronous Timing

```
     ORG     P:0             ;Reset vector
     JMP     $40             ;
     ORG     P:$18           ;SCI transmit interrupt vector
     MOVEP   Y:(R0)+,X:$FFF4 ;Transmit low byte of data
     ORG     P:$40
     MOVEP   #0,X:$FFFE ;Clear BCR
     MOVE    #$100,R0   ;Data ROM start address
     MOVE    #$FF,M0    ;Size of data ROM - Wraps around at $200
     MOVEC   #6,OMR     ;Change operating mode to enable data ROM
     MOVEP   #$C000,X:$FFFF  ;Interrupt priority register
     MOVEP   #$1200,X:$FFF0  ;8-bit synchronous mode
     MOVEP   #7,X:$FFE1    ;Port C control register – enable SCI
     MOVEC   #0,SR           ;Unmask interrupts
LAB0 JMP     LAB0            ;Wait in loop for interrupts
```

Figure 6-20 SCI Synchronous Transmit

The assembly program shown in Figure 6-20 uses the SCI synchronous mode to transmit only the low byte of the Y data ROM contents. The program sets the reset vector to run the program after a hardware reset, puts the MOVEP instruction at the SCI transmit interrupt vector location, sets the memory wait states to zero, and configures the memory pointers, operating mode register, and the IPR.

The SCI is then configured and the interrupts are unmasked, which starts the data transfer. The jump-to-self instruction (LAB0 JMP LAB0) is used to wait while interrupts transfer the data.

The program shown in Figure 6-21 is the program for receiving data from the program presented in Figure 6-20. The program sets the reset vector to run the program after hardware reset, puts the MOVEP instruction to store the data in a circular buffer starting at $100 at the SCI receive interrupt vector location, puts another MOVEP instruction at the SCI receive interrupt vector location, sets the memory wait states to zero, and configures the memory pointers and IPR. The SCI is then configured and the interrupts are unmasked, which starts the data transfer. The jump-to-self instruction (LAB0 JMP LAB0) is used to wait while interrupts transfer the data.

```
     ORG    P:0              ;Reset vector
     JMP    $40              ;
     ORG    P:$14            ;SCI receive data vector
     MOVEP  X:$FFF4,Y:(R0)+  ;Receive low byte of data
     NOP                     ;Fast interrupt response
     MOVEP  X:$FFF1,X0       ;Receive with exception.
                             ;Read status register
     MOVEP  X:$FFF4,Y:(R0)+  ;Receive low byte of data
     ORG    P:$40
     MOVEP  #0,X:$FFFE       ;Clear BCR
     MOVE   #$100,R0         ;Data ROM start address
     MOVE   #$FF,M0       ; Size of data ROM – wraps around at $200
     MOVEP  #$C000,X:$FFFF   ;Interrupt priority register
     MOVEP  #$900,X:$FFF0    ;8-bit synchronous mode receive only
     MOVEP  #$C000,X:$FFF2   ;Clock control register external clock
     MOVEP  #7,X:$FFE1       ;Port C control register – enable SCI
     MOVEC  #0,SR            ;Unmask interrupts
LAB0 JMP    LAB0             ;Wait in loop for interrupts
```

Figure 6-21 SCI Synchronous Receive

6.3.7 Asynchronous Data

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× 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. In addition, two of the word formats have a parity bit. The multidrop 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. The SCI transmitter and receiver can use either the internal clock (TCM=0 and/or RCM=0) or an external clock (TCM=1 and/or RCM=1) or a combination. If a combination is used, the transmitter and receiver can run at different data rates.

6.3.7.1 Asynchronous Data Reception

Figure 6-22 illustrates initializing the SCI data receiver for asynchronous data. The first step (1) resets the SCI to prevent the SCI from transmitting or receiving data. Step two (2) selects the desired operation by programming the SCR. As a minimum, the word format (WDS2, WDS1, and WDS0) must be selected, and (3) the receiver must be enabled (RE=1). If (4) interrupts are to be used, set RIE equals one. Use Table 6-3 (a) through Table 6-4 (b) to set (5) the baud rate (SCP and CD0–CD11 in the SCCR). Once the SCI is completely configured, it is enabled by (6) setting the RXD bit in the PCC.

The receiver is continually sampling RDX at the 16 × clock rate to find the idle-start-bit transition edge. When that edge is detected (1) the following eight or nine bits, depending on the mode, are clocked into the receive shift register (see Figure 6-23). Once a complete byte is received, (2) the character is latched into the SRX, and RDRF is set as well as the error flags, OR, PE, and FE. If (3) interrupts are enabled, an interrupt is generated. The interrupt service routine, which can be a fast interrupt or a long interrupt, (4) reads the received character. Reading the SRX (5) automatically clears RDFR in the SSR and makes the SRX ready to receive another byte.

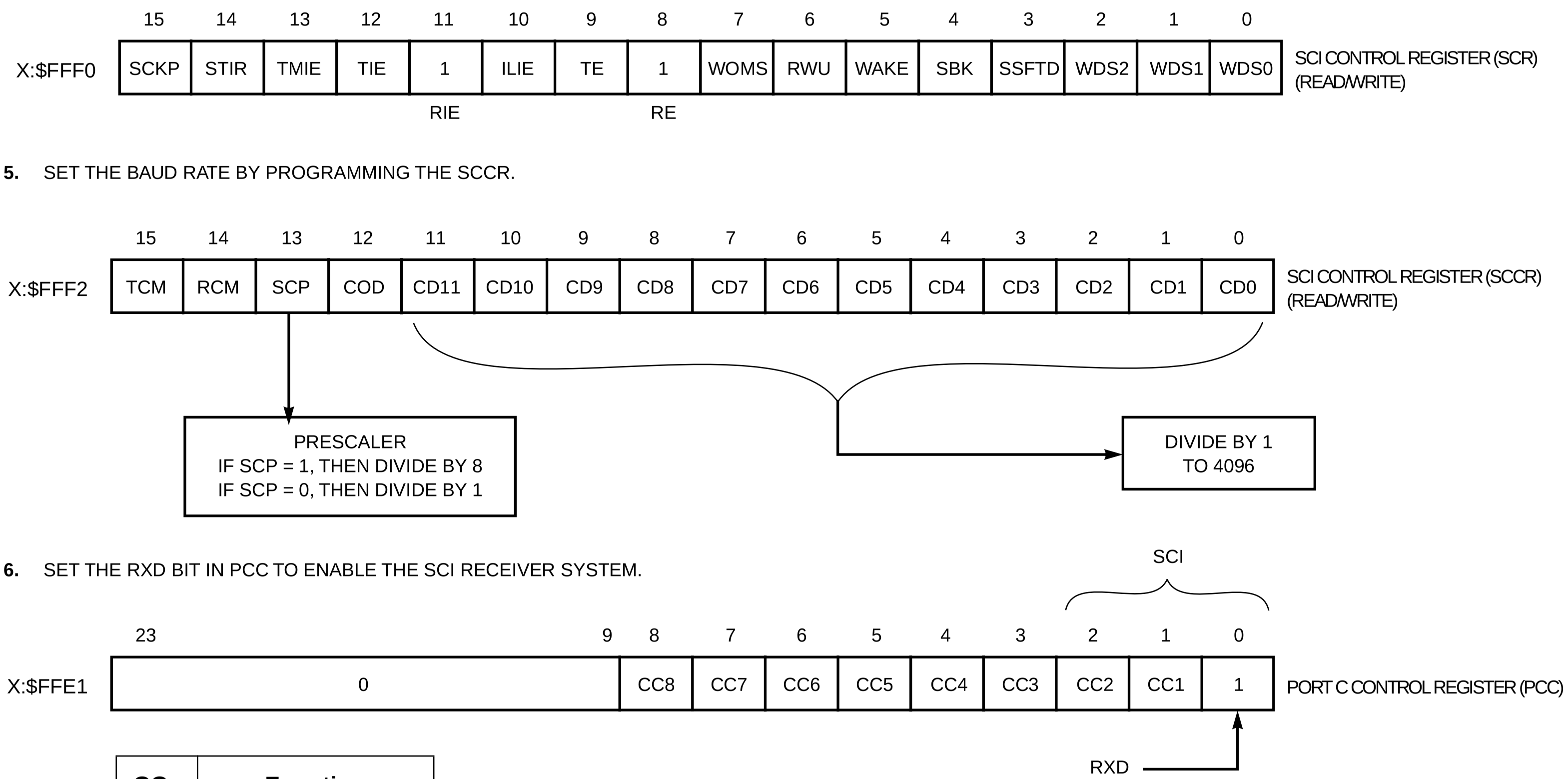
If (1) an FE, PE, or OR occurs while receiving data (see Figure 6-24), (2) RDRF is set because a character has been received; FE, PE, or OR is set in the SSR to indicate that an error was detected. Either (3) the SSR can be polled by software to look for errors, or (4) interrupts can be used to execute an interrupt service routine. This interrupt is different from the normal receive interrupt and is caused only by receive errors. The long interrupt service routine should (5) read the SSR to determine what error was detected and then (6) read the SRX to clear RDRF and all three error flags.

6.3.7.2 Asynchronous Data Transmission

Figure 6-25 illustrates initializing the SCI data transmitter for asynchronous data. The first step (1) resets the SCI to prevent the SCI from transmitting or receiving data. Step two (2) selects the desired operation by programming the SCR. As a minimum, the word format (WDS2, WDS1, and WDS0) must be selected, and (3) the transmitter must be enabled (TE=1). If (4) interrupts are to be used, set TIE equals one. Use Table 6-3 (a) through Table 6-4 (b) to set (5) the baud rate (SCP and CD0–CD11 in the SCCR). Once the SCI is completely configured, it can be enabled by (6) setting the TXD bit in the PCC. Transmission begins with (7) a preamble of ones.

If polling is used to transmit data (see Figure 6-26), the polling routine can look at either TDRE or TRNE to determine when to load another byte into STX. If TDRE is used (1), one byte may be loaded into STX. If TRNE is used (2), two bytes may be loaded into STX if enough time is allowed for the first byte to begin transmission (see Section **6.3.2.4.2**). If interrupts are used (3), then an interrupt is generated when STX is empty. The interrupt routine, which can be a fast interrupt or a long interrupt, writes (4) one byte into STX.

1. HARDWARE OR SOFTWARE RESET
2. PROGRAM SCR WITH DESIRED MODE AND FEATURES.
3. TURN ON RECEIVER (RE = 1).
4. OPTIONALLY ENABLE RECEIVER INTERRUPTS (RIE = 1).



5. SET THE BAUD RATE BY PROGRAMMING THE SCCR.

6. SET THE RXD BIT IN PCC TO ENABLE THE SCI RECEIVER SYSTEM.

CCx	Function
0	GPIO
1	Serial Interface

NOTE: If RE is cleared while a valid character is being received, the reception of the character will be completed before the receiver is disabled.

Figure 6-22 Asynchronous SCI Receiver Initialization

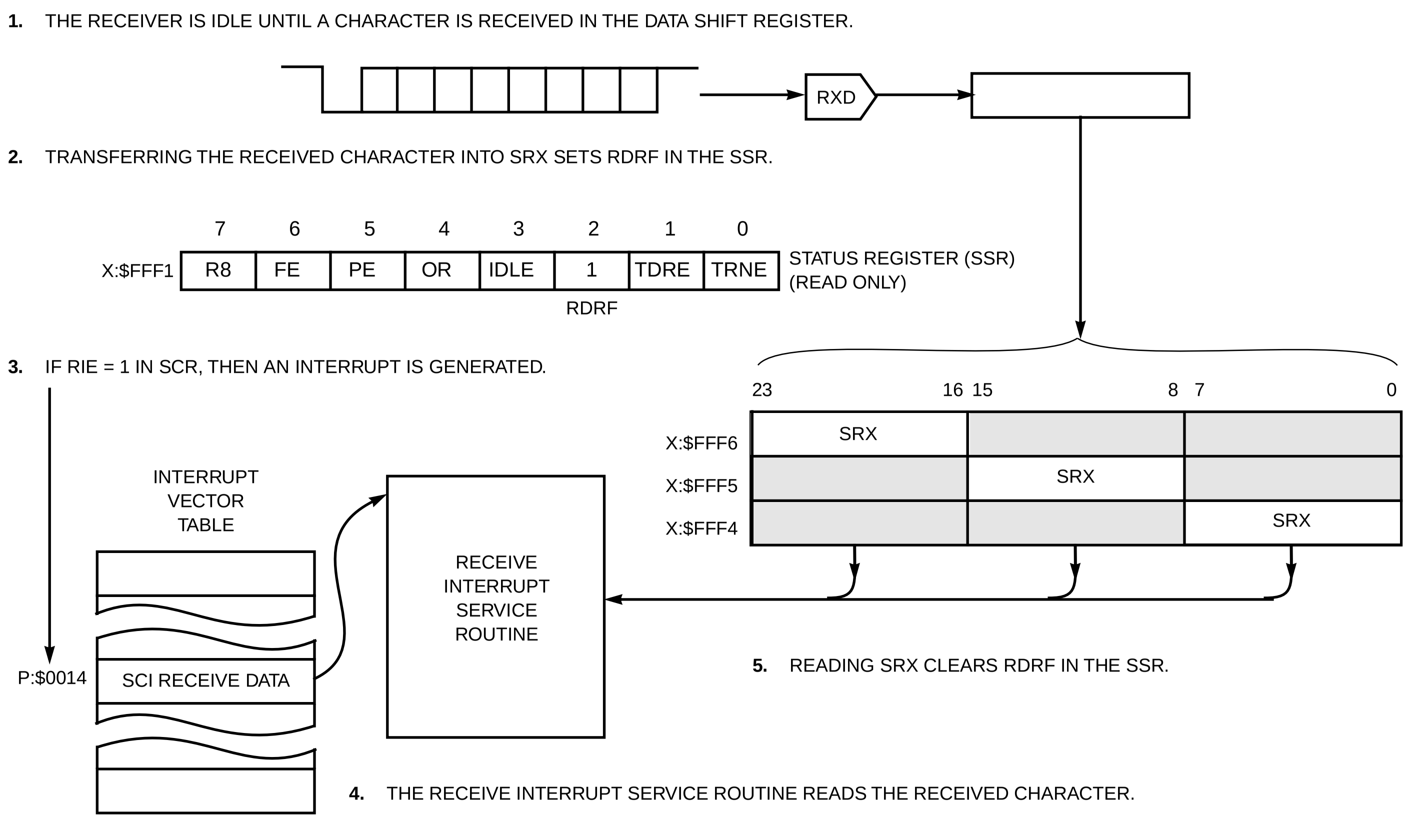

1. THE RECEIVER IS IDLE UNTIL A CHARACTER IS RECEIVED IN THE DATA SHIFT REGISTER.
RXD
2. TRANSFERRING THE RECEIVED CHARACTER INTO SRX SETS RDRF IN THE SSR.
7
6
5
4
3
2
1
0
X:$FFF1
R8
FE
PE
OR
IDLE
1
TDRE
TRNE
STATUS REGISTER (SSR)
(READ ONLY)
RDRF
3. IF RIE = 1 IN SCR, THEN AN INTERRUPT IS GENERATED.
23
16 15
8 7
0
X:$FFF6
X:$FFF5
X:$FFF4
SRX
SRX
SRX
INTERRUPT
VECTOR
TABLE
P:$0014
SCI RECEIVE DATA
RECEIVE
INTERRUPT
SERVICE
ROUTINE
5. READING SRX CLEARS RDRF IN THE SSR.
4. THE RECEIVE INTERRUPT SERVICE ROUTINE READS THE RECEIVED CHARACTER.

Figure 6-23 SCI Character Reception

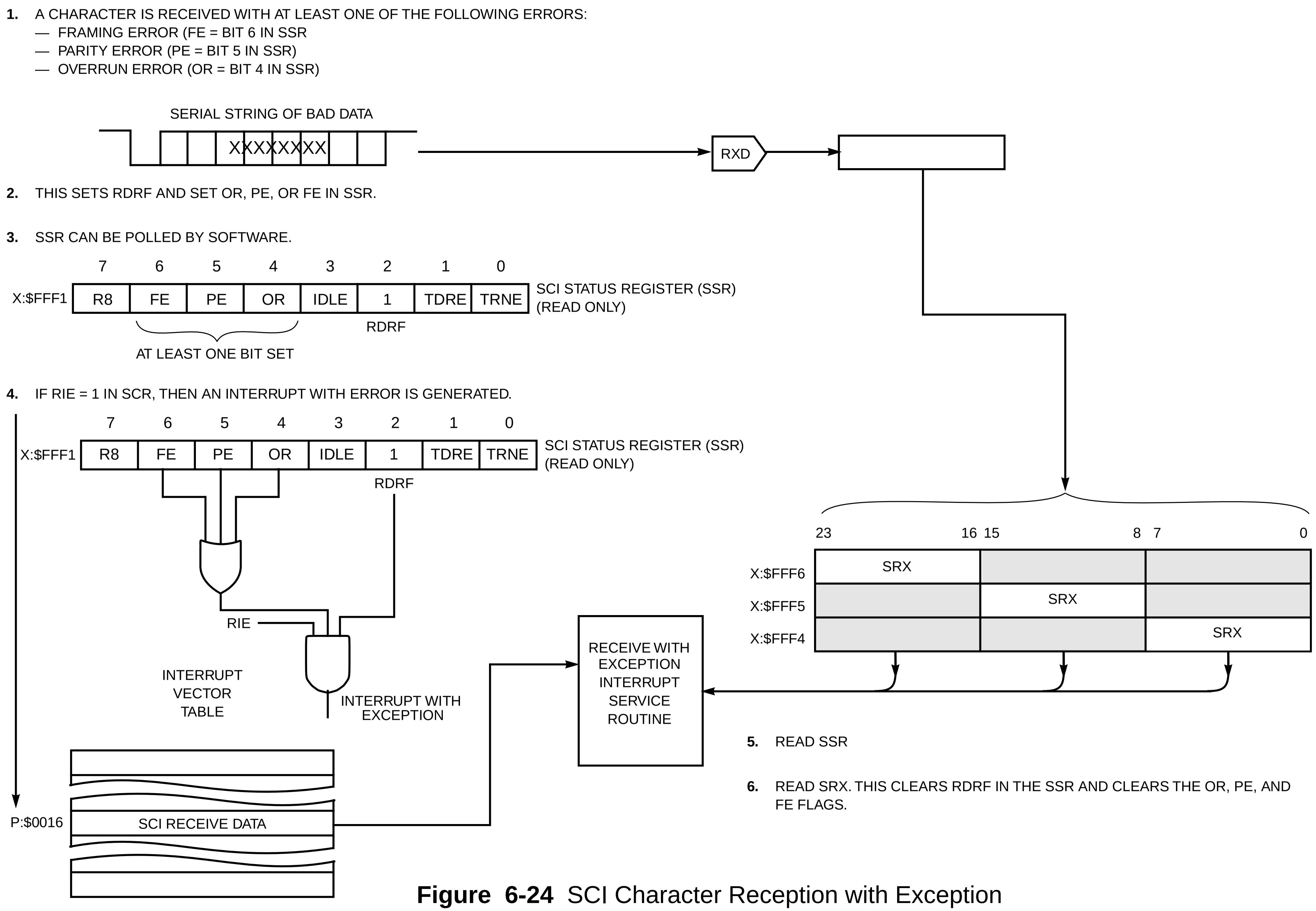

1. A CHARACTER IS RECEIVED WITH AT LEAST ONE OF THE FOLLOWING ERRORS:
— FRAMING ERROR (FE = BIT 6 IN SSR
— PARITY ERROR (PE = BIT 5 IN SSR)
— OVERRUN ERROR (OR = BIT 4 IN SSR)
SERIAL STRING OF BAD DATA
XXXXXXXX
RXD
2. THIS SETS RDRF AND SET OR, PE, OR FE IN SSR.
3. SSR CAN BE POLLED BY SOFTWARE.
7 6 5 4 3 2 1 0
X:$FFF1
R8 FE PE OR IDLE 1 TDRE TRNE
SCI STATUS REGISTER (SSR)
(READ ONLY)
RDRF
AT LEAST ONE BIT SET
4. IF RIE = 1 IN SCR, THEN AN INTERRUPT WITH ERROR IS GENERATED.
7 6 5 4 3 2 1 0
X:$FFF1
R8 FE PE OR IDLE 1 TDRE TRNE
SCI STATUS REGISTER (SSR)
(READ ONLY)
RDRF
RIE
INTERRUPT VECTOR TABLE
INTERRUPT WITH EXCEPTION
P:$0016
SCI RECEIVE DATA
RECEIVE WITH EXCEPTION INTERRUPT SERVICE ROUTINE
23
16 15
8 7
0
X:$FFF6
X:$FFF5
X:$FFF4
SRX
SRX
SRX
5. READ SSR
6. READ SRX. THIS CLEARS RDRF IN THE SSR AND CLEARS THE OR, PE, AND FE FLAGS.

Figure 6-24 SCI Character Reception with Exception

1. HARDWARE OR SOFTWARE RESET
2. PROGRAM SCR WITH DESIRED MODE AND FEATURES.
3. TURN ON TRANSMITTER (TE = 1).
4. OPTIONALLY ENABLE TRANSMITTER INTERRUPTS (TIE = 1).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
X:$FFF0	SCKP	STIR	TMIE	1	RIE	ILIE	1	RE	WOMS	RWU	WAKE	SBK	SSFTD	WDS2	WDS1	WDS0	SCI CONTROL REGISTER (SCR) (READ/WRITE)
				TIE			TE										

5. SET THE SCI CLOCK PRESCALER BIT AND THE CLOCK DIVIDER BITS IN THE SCCR.
6. SET THE TXD BIT IN PCC TO ENABLE THE SCI TRANSMITTER SYSTEM.

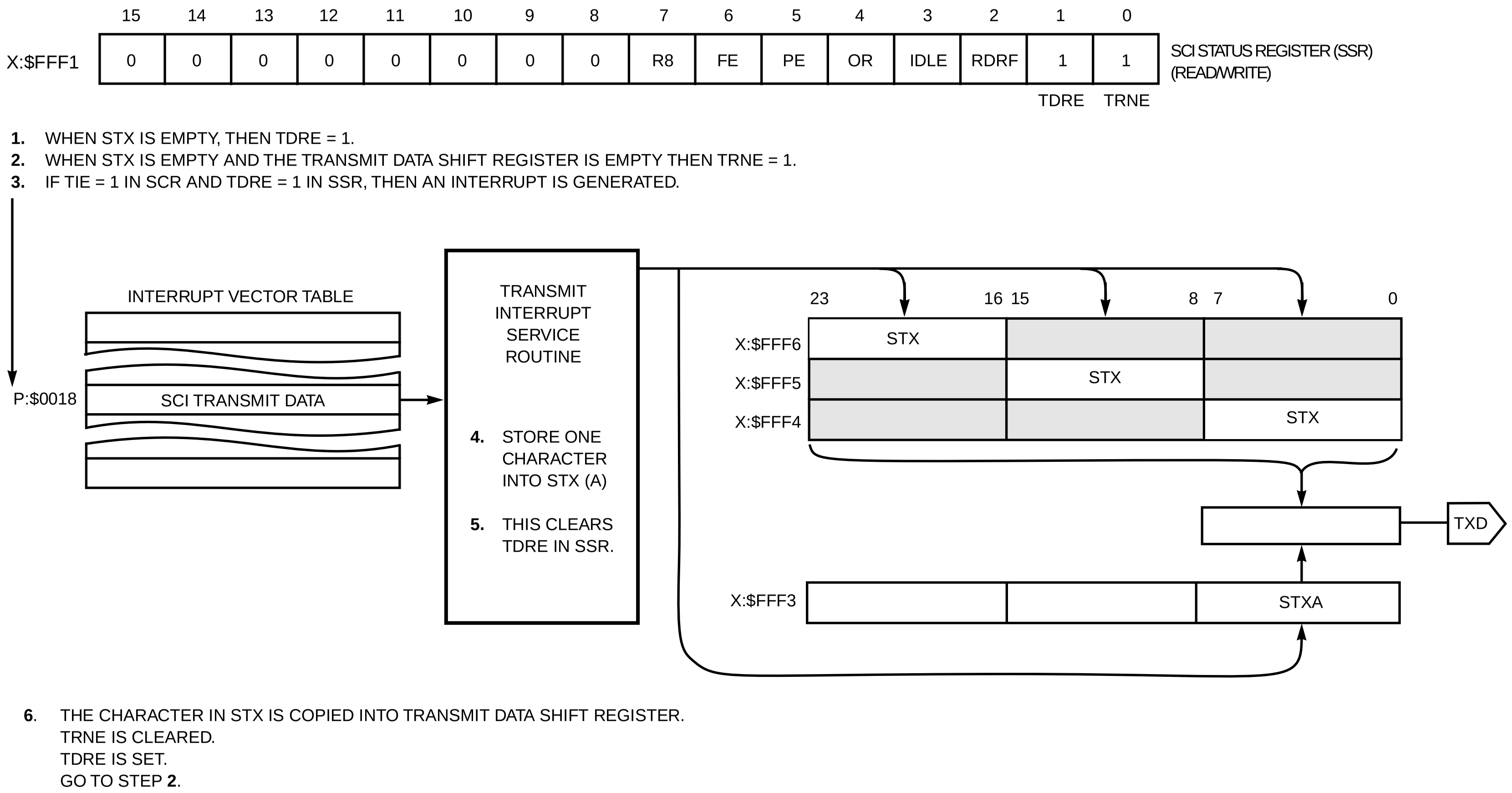
	23	9	8	7	6	5	4	3	2	1	0	
									SCI			
X:$FFE1	0		CC8	CC7	CC6	CC5	CC4	CC3	CC2	1	CC0	PORT C CONTROL REGISTER (PCC)
										TXD		

CCx	Function
0	GPIO
1	Serial Interface

7. THE TRANSMITTER WILL FIRST BROADCAST A PREAMBLE OF ONES BEFORE BEGINNING DATA TRANSMISSION:
 10 ONES WILL BE TRANSMITTED FOR THE 10-BIT ASYNCHRONOUS MODE.
 11 ONES WILL BE TRANSMITTED FOR THE 11-BIT ASYNCHRONOUS MODE.

NOTE: If TE is cleared while transmitting a character, the transmission of the character will be completed before the transmitter is disabled.

Figure 6-25 Asynchronous SCI Transmitter Initialization



1. WHEN STX IS EMPTY, THEN TDRE = 1.
2. WHEN STX IS EMPTY AND THE TRANSMIT DATA SHIFT REGISTER IS EMPTY THEN TRNE = 1.
3. IF TIE = 1 IN SCR AND TDRE = 1 IN SSR, THEN AN INTERRUPT IS GENERATED.

4. STORE ONE CHARACTER INTO STX (A)

5. THIS CLEARS TDRE IN SSR.

6. THE CHARACTER IN STX IS COPIED INTO TRANSMIT DATA SHIFT REGISTER.
TRNE IS CLEARED.
TDRE IS SET.
GO TO STEP **2**.

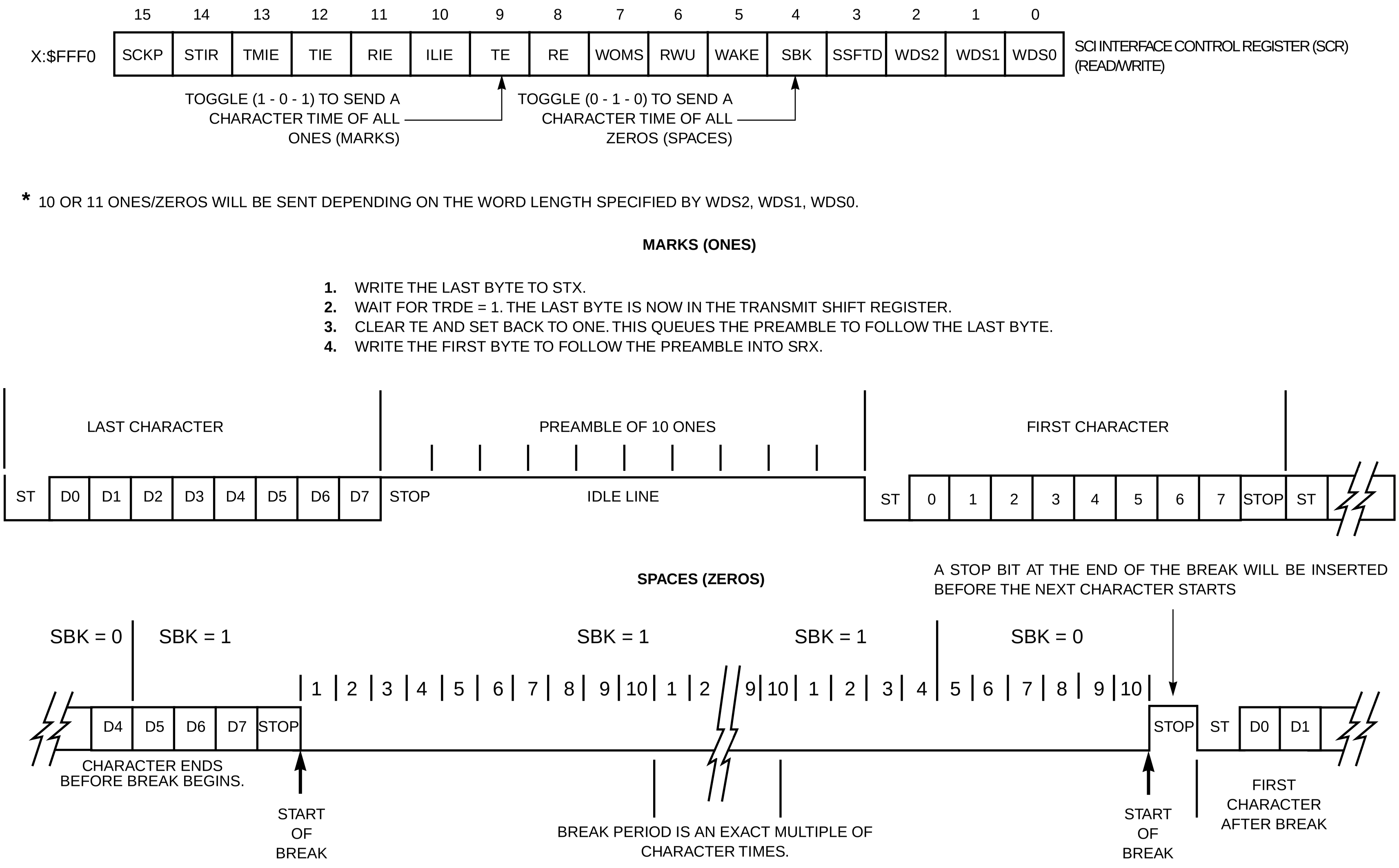
Figure 6-26 Asynchronous SCI Character Transmission

If multidrop mode is being used and this byte is an address, STXA should be used instead of STX. Writing STX or STXA (5) clears TDRE in the SSR. When the transmit data shift register is empty (6), the byte in STX (or STXA) is latched into the transmit data shift register, TRNE is cleared, and TDRE is set.

There is a provision to send a break or preamble. A break (space) consists of a period of zeros with no start or stop bits that is as long or longer than a character frame. A preamble (mark) is an inverted break. A preamble of 10 or 11 ones (depending on the word length selected by WDS2, WDS1, and WDS0) can be sent with the following procedure (see Figure 6-27). (1) Write the last byte to STX and (2) wait for TDRE equals one. This is the byte that will be transmitted immediately before the preamble. (3) Clear TE and then again set it to one. Momentarily clearing TE causes the output to go high for one character frame. If TE remains cleared for a longer period, the output will remain high for an even number of character frames until TE is set. (4) Write the first byte to follow the preamble into SRX before the preamble is complete and resume normal transmission. Sending a break follows the same procedure except that instead of clearing TE, SBK is set in the SCR to send breaks and then reset to resume normal data transmission.

The example presented in Figure 6-28 uses the SCI in the asynchronous mode to transfer data into buffers. Interrupts are used, allowing the DSP to perform other tasks while the data transfer is occurring. This program can be tested by connecting the SCI transmit and receive pins. Equates are used for convenience and readability.

The program sets the reset vector to run the program after reset, puts a MOVEP instruction at the SCI receive interrupt vector location, and puts a MOVEP and BCLR at the SCI transmit interrupt vector location so that, after transmitting a byte, the transmitter is disabled until another byte is ready for transmission. The SCI is initialized by setting the interrupt level, which configures the SCR and SCCR, and then is enabled by writing the PCC. The main program begins by enabling interrupts, which allows data to be received. Data is transmitted by moving a byte of data to the transmit register and by enabling interrupts. The jump-to-self instruction (SEND JMP SEND) is used to wait while interrupts transfer the data.


15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
X:$FFF0
SCKP
STIR
TMIE
TIE
RIE
ILIE
TE
RE
WOMS
RWU
WAKE
SBK
SSFTD
WDS2
WDS1
WDS0
SCI INTERFACE CONTROL REGISTER (SCR)
(READ/WRITE)
TOGGLE (1 - 0 - 1) TO SEND A CHARACTER TIME OF ALL ONES (MARKS)
TOGGLE (0 - 1 - 0) TO SEND A CHARACTER TIME OF ALL ZEROS (SPACES)
LAST CHARACTER
PREAMBLE OF 10 ONES
FIRST CHARACTER
ST
D0
D1
D2
D3
D4
D5
D6
D7
STOP
IDLE LINE
ST
0
1
2
3
4
5
6
7
STOP
ST
SBK = 0
SBK = 1
SBK = 1
SBK = 1
SBK = 0
1
2
3
4
5
6
7
8
9
10
1
2
9
10
1
2
3
4
5
6
7
8
9
10
D4
D5
D6
D7
STOP
STOP
ST
D0
D1
CHARACTER ENDS BEFORE BREAK BEGINS.
START OF BREAK
BREAK PERIOD IS AN EXACT MULTIPLE OF CHARACTER TIMES.
START OF BREAK
FIRST CHARACTER AFTER BREAK
A STOP BIT AT THE END OF THE BREAK WILL BE INSERTED BEFORE THE NEXT CHARACTER STARTS

* 10 OR 11 ONES/ZEROS WILL BE SENT DEPENDING ON THE WORD LENGTH SPECIFIED BY WDS2, WDS1, WDS0.

MARKS (ONES)

1. WRITE THE LAST BYTE TO STX.
2. WAIT FOR TRDE = 1. THE LAST BYTE IS NOW IN THE TRANSMIT SHIFT REGISTER.
3. CLEAR TE AND SET BACK TO ONE. THIS QUEUES THE PREAMBLE TO FOLLOW THE LAST BYTE.
4. WRITE THE FIRST BYTE TO FOLLOW THE PREAMBLE INTO SRX.

SPACES (ZEROS)

Figure 6-27 Transmitting Marks and Spaces

```
;*****************************************************************
;       SCI ASYNC WITH INTERRUPTS AND SINGLE BYTE BUFFERS*
;*****************************************************************
;**************************************************
;       SCI and other EQUATES*
;**************************************************
START   EQU     $0040           ;Start of program
PCC     EQU     $FFE1           ;Port C control register
SCR     EQU     $FFF0           ;SCI interface control register
SCCR    EQU     $FFF2           ;SCI clock control register
SRX     EQU     $FFF4           ;SCI receive register
STX     EQU     $FFF4           ;SCI transmit register
BCR     EQU     $FFFE           ;Bus control register
IPR     EQU     $FFFF           ;Interrupt priority register
RXBUF   EQU     $100            ;Receive buffer
TXBUF   EQU     $200            ;Transmit buffer
;**************************************************
;       RESET VECTOR *
;**************************************************
        ORG     P:$0000
        JMP     START
;**************************************************
;       SCI RECEIVE INTERRUPT VECTOR*
;**************************************************
        ORG     P:$0014         ;Load the SCI RX interrupt vectors
        MOVEP   X:SRX,Y:(R0)+   ;Put the received byte in the receive
                                ;buffer. This receive routine is
                                ;implemented as a fast interrupt.
;**************************************************
;       SCI TRANSMIT INTERRUPT VECTOR*
;**************************************************
        ORG     P:$0018         ;Load the SCI TX interrupt vectors
        MOVEP   X:(R3)+,X:STX   ;Transmit a byte and
                                ;increment the pointer in the
                                ;transmit buffer.
        BCLR    #12,X:SCR       ;Disable transmit interrupts
```

Figure 6-28 SCI Asynchronous Transmit/Receive Example (Sheet 1 of 2)

```
;*****************************************************************
;       INITIALIZE THE SCI PORT AND RX, TX BUFFER POINTERS*
;*****************************************************************
        ORG     P:START         ;Start the program at location $40
        ORI     #$03,MR         ;Mask interrupts temporarily
        MOVEP   #$C000,X:IPR    ;Set interrupt priority to 2
        MOVEP   #$0B02,X:SCR    ;Disable TX, enable RX interrupts
                                ;Enable transmitter, receiver
                                ;Point to point
                                ;10-bit asynchronous
                                ;(1 start, 8 data, 1 stop)
        MOVEP   #$0022,X:SCCR   ;Use internal TX, RX clocks
                                ;9600 BPS
        MOVEP   #>$03,X:PCC     ;Select pins TXD and RXD for SCI
        MOVE    RXBUF,R0        ;Initialize the receive buffer
        MOVE    TXBUF,R3        ;Initialize the transmit buffer
;*************************************************
;       MAIN PROGRAM *
;*************************************************
        ANDI    #$FC,MR         ;Re-enable interrupts
        MOVE    #>$41,X:(R3)    ;Move a byte to the transmit buffer
        MOVE    R0,X:(R3)
        BSET    #12,X:SCR       ;and enable interrupts so it
                                ;will be transmitted
SEND    JMP     SEND            ;Normally something more useful
                                ;would be put here.
        END                     ;End of example.
```

Figure 6-28 SCI Asynchronous Transmit/Receive Example (Sheet 2 of 2)

6.3.8 Multidrop

Multidrop is a special case of asynchronous data transfer. The key difference is that a protocol is used to allow networking transmitters and receivers on a single data-transmission line. Interprocessor messages in a multidrop 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 wakeup 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.

These message formats include point-to-point, bus, token-ring, and custom configurations. The SCI multidrop network is compatible with other leading microprocessors.

Figure 6-29 shows a multidrop system with one master and N slaves. The multidrop mode is selected by setting WDS2 equals one, WDS1 equals one, and WDS0 equals zero. One possible protocol is to have a preamble or idle line between messages, followed by an address and then a message. The idle line causes the slaves to wake up and compare the address with their own address. If the addresses match, the slave receives the message. If the addresses do not match, the slave ignores the message and goes back to sleep. It is also possible to generate an interrupt when an address is received, eliminating the need for idle time between consecutive messages and addresses. It is also possible for each slave to look for more than one address, which allows each slave to respond to individual messages as well as broadcast messages (e.g., a global reset).

6.3.8.1 Transmitting Data and Address Characters

Transmitting data and address when the multidrop mode is selected is shown in Figure 6-30. The output sequence shown is idle line, data/address, and the next character. In both cases, an “A” is being transmitted. To send data, TE must be toggled to send the idle line, and then “A” must be sent to STX. Sending the “A” to the STX sets the ninth bit in the frame to zero, which indicates that this frame contains data. If the “A” is sent to STXA instead, the ninth bit in the frame is set to a one, which indicates that this frame contains an address.

6.3.8.2 Wired-OR Mode

Building a multidrop 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 multidrop word format provides an address field to support this protocol. Figure 6-31 shows a multidrop configuration using wired-OR (set bit 7 of the SCR). The protocol shown consists of an idle line between messages; each message begins with an address character. The message can be any length, depending on the protocol. Each processor in this system has one address that it responds to although each processor can be programmed to respond to more than one address.

6.3.8.3 Idle Line Wakeup

A wakeup 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 DSPs address. If the addresses do not match, the SCI again suspends reception until the next address. If the address matches, the DSP will read and process the message and then suspend reception until the next address.

The idle line wakeup mode wakes up the SCI to read a message before the first character arrives. This mode allows the message to be in any format.

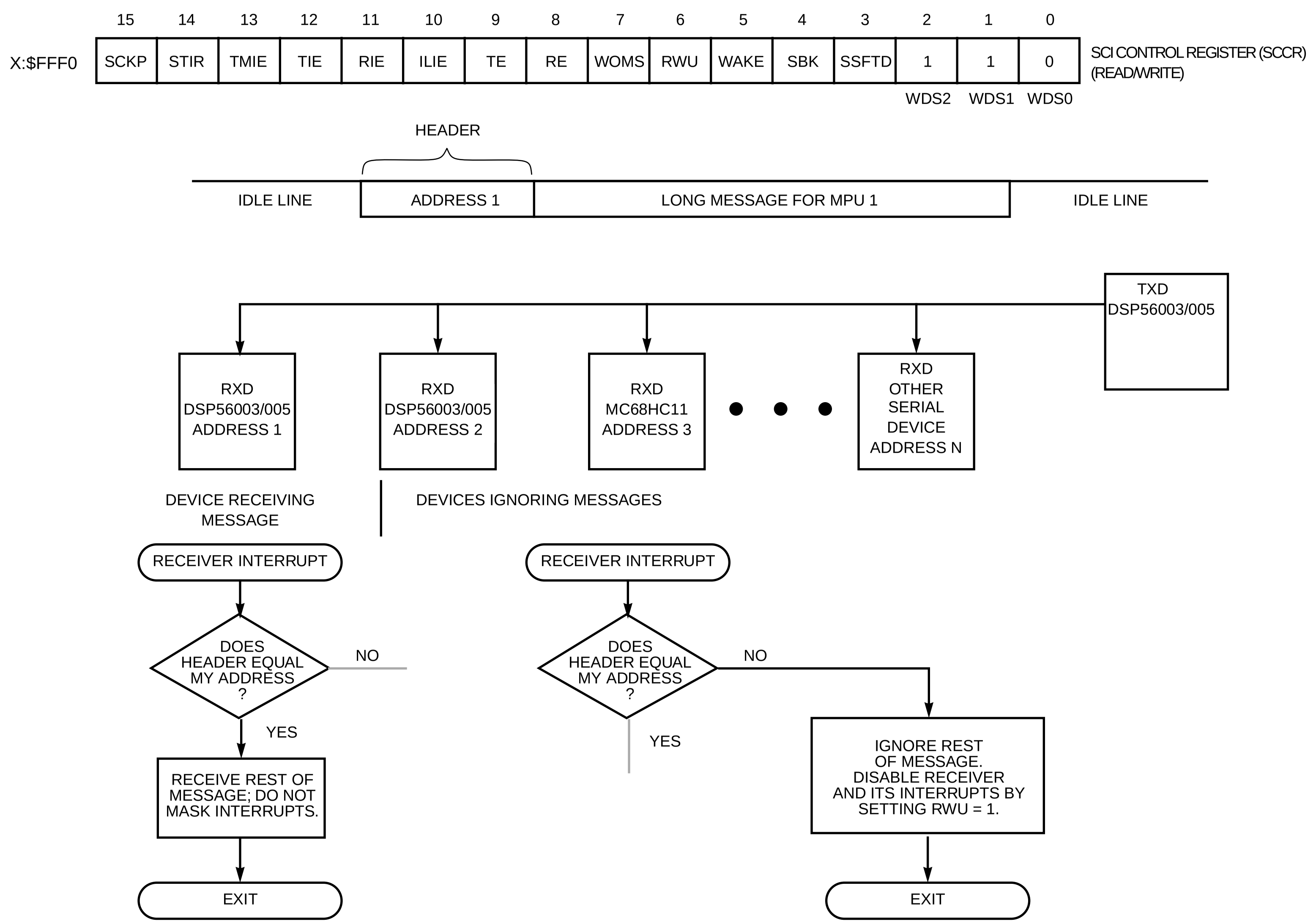


Figure 6-29 11-Bit Multidrop Mode

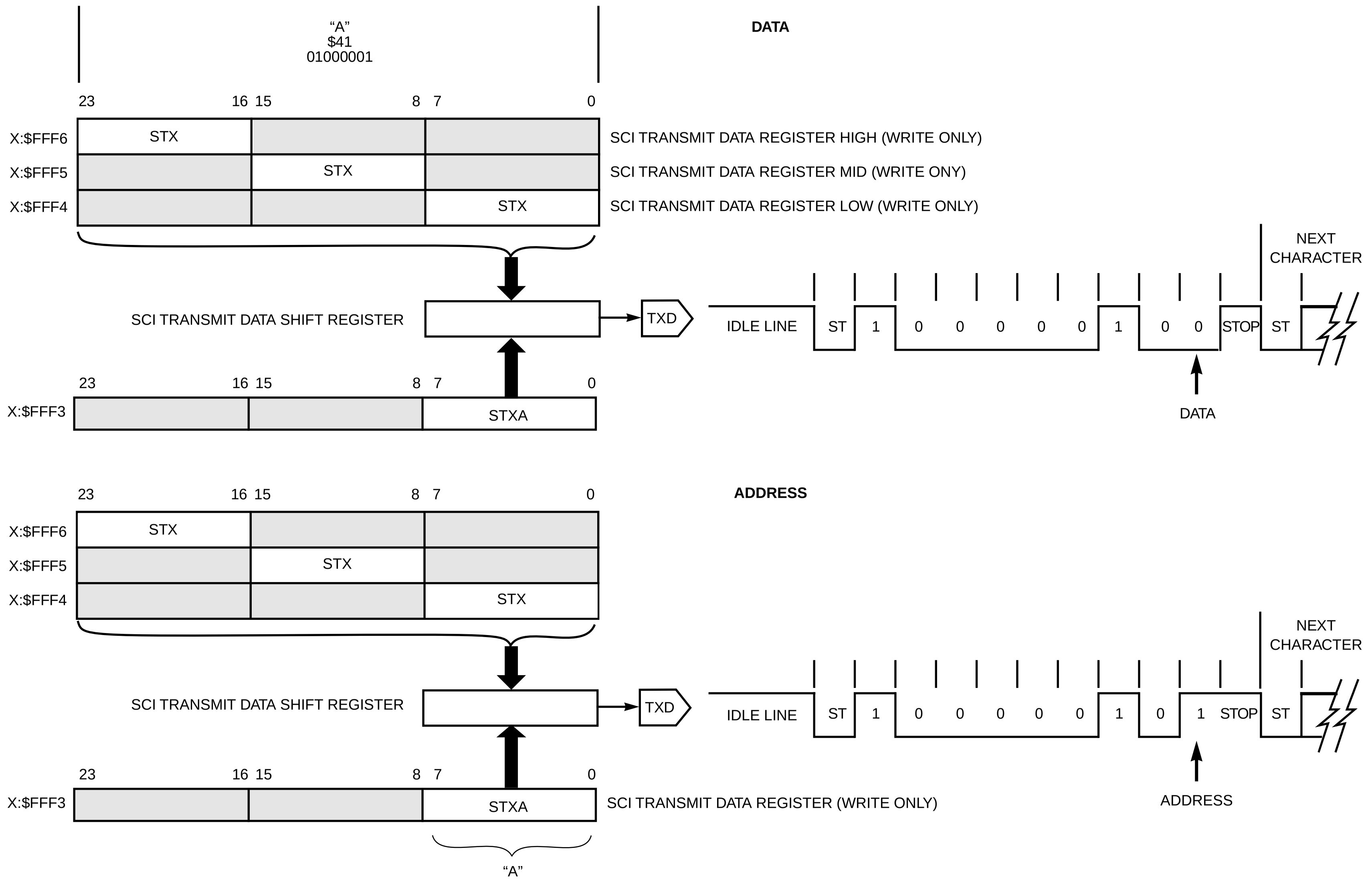

"A"
$41
01000001
DATA
23
16
15
8
7
0
X:$FFF6
X:$FFF5
X:$FFF4
STX
STX
STX
SCI TRANSMIT DATA REGISTER HIGH (WRITE ONLY)
SCI TRANSMIT DATA REGISTER MID (WRITE ONY)
SCI TRANSMIT DATA REGISTER LOW (WRITE ONLY)
SCI TRANSMIT DATA SHIFT REGISTER
TXD
IDLE LINE
ST
1
0
0
0
0
0
1
0
0
STOP
ST
NEXT CHARACTER
X:$FFF3
STXA
DATA
ADDRESS
X:$FFF6
X:$FFF5
X:$FFF4
STX
STX
STX
SCI TRANSMIT DATA SHIFT REGISTER
TXD
IDLE LINE
ST
1
0
0
0
0
0
1
0
1
STOP
ST
NEXT CHARACTER
X:$FFF3
STXA
SCI TRANSMIT DATA REGISTER (WRITE ONLY)
ADDRESS
"A"

Figure 6-30 Transmitting Data and Address Characters

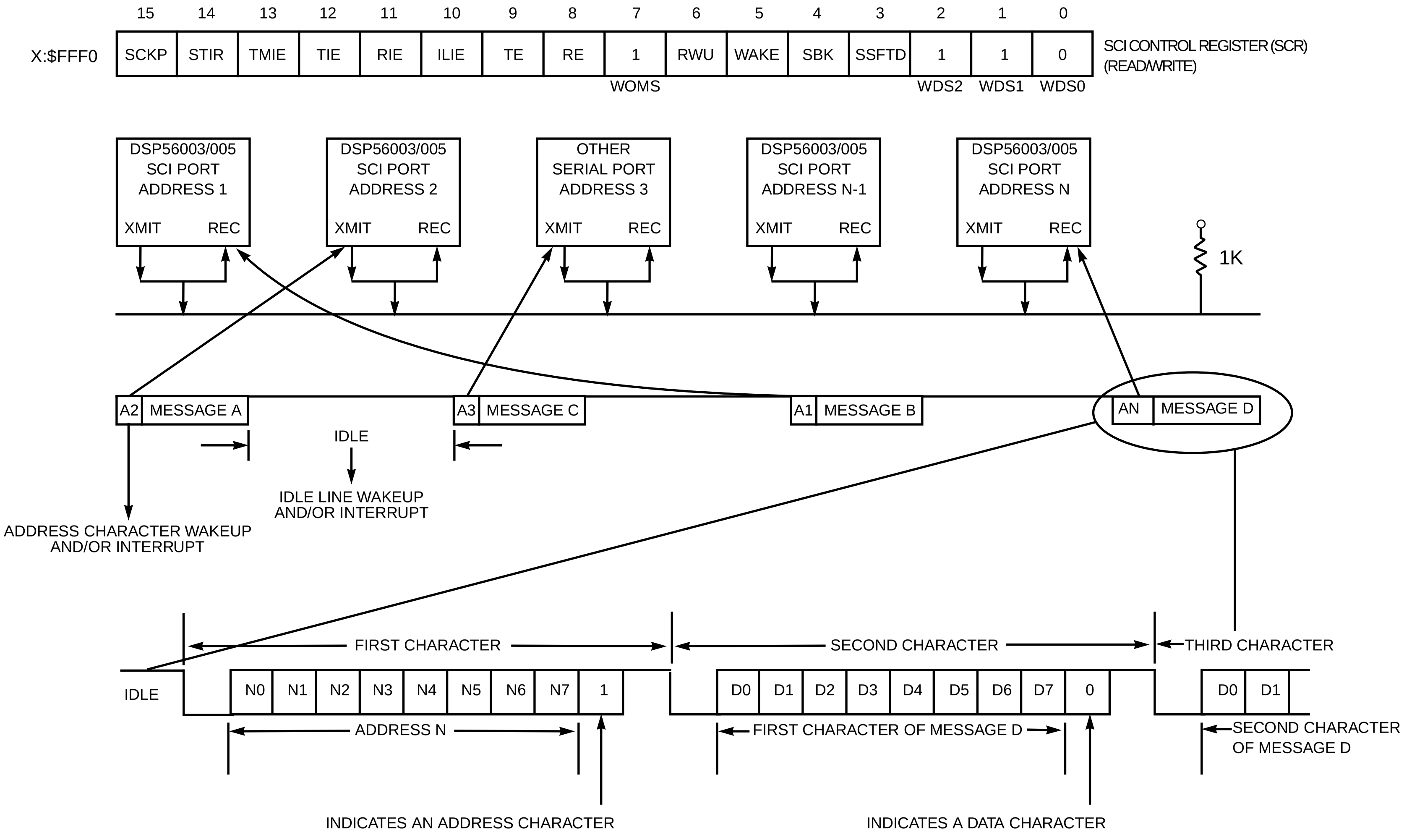

15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
X:$FFF0
SCKP
STIR
TMIE
TIE
RIE
ILIE
TE
RE
1
RWU
WAKE
SBK
SSFTD
1
1
0
SCI CONTROL REGISTER (SCR)
(READ/WRITE)
WOMS
WDS2
WDS1
WDS0
DSP56003/005 SCI PORT ADDRESS 1
XMIT
REC
DSP56003/005 SCI PORT ADDRESS 2
XMIT
REC
OTHER SERIAL PORT ADDRESS 3
XMIT
REC
DSP56003/005 SCI PORT ADDRESS N-1
XMIT
REC
DSP56003/005 SCI PORT ADDRESS N
XMIT
REC
1K
A2
MESSAGE A
A3
MESSAGE C
A1
MESSAGE B
AN
MESSAGE D
IDLE
IDLE LINE WAKEUP AND/OR INTERRUPT
ADDRESS CHARACTER WAKEUP AND/OR INTERRUPT
FIRST CHARACTER
SECOND CHARACTER
THIRD CHARACTER
IDLE
N0
N1
N2
N3
N4
N5
N6
N7
1
D0
D1
D2
D3
D4
D5
D6
D7
0
D0
D1
ADDRESS N
FIRST CHARACTER OF MESSAGE D
SECOND CHARACTER OF MESSAGE D
INDICATES AN ADDRESS CHARACTER
INDICATES A DATA CHARACTER

Figure 6-31 Wired-OR Mode

Figure 6-32 shows how to configure the SCI to detect and respond to an idle line. The word format chosen (WDS2, WDS1, and WDS0 in the SCR) must be asynchronous. The WAKE bit must be clear to select idle line wakeup, and RWU must be set to put the SCI to "sleep" and enable the wakeup function. RIE should be set if interrupts are to be used to receive data. If processing must occur when the idle line is first detected, ILIE should be set. The current message is followed by one or more data frames of ones (10 or 11 bits each, depending on which word format is used), which are detected as an idle line. If the word format is multidrop (an 11-bit code), after the 11 ones, the receiver determines the line is idle and (1) clears the RWU, enabling the receiver. The IDLE bit (2) and an internal flag SRIINT (3) are set, indicating the line is idle. The SCI is now ready to receive messages; however, nothing more will happen until the next start bit unless (4) ILIE is set.

If ILIE is set, an SCI idle line interrupt will be recognized as pending. When the idle line interrupt is recognized (5), SRIINT is automatically cleared, and the SCI waits for the first start bit of the next character. Since RIE was set, when the first character is received, an SCI receive data interrupt (or SCI receive data with exception status interrupt if an error is detected) will be recognized as pending. When the receiver has processed the message and is ready to wait for another idle line, RWU must be set to one again.

6.3.8.4 Address Mode Wakeup

The purpose and basic operational procedure for address mode wakeup is the same as idle line wakeup. The difference is that address mode wakeup 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. If the protocol is such that the address byte is not needed or is not wanted in the first byte of the message, a data byte can be written to STXA at the beginning of each message. It is not essential that the first byte of the message contain an address; it is essential that the start of a new message is indicated by setting the ninth bit to one using STXA.

Figure 6-33 shows how to configure the SCI to detect and respond to an address character. The word format chosen (WDS2, WDS1, and WDS0 in the SCR) must be an asynchronous word format. The WAKE bit must be set to select address mode wakeup and RWU must be set to put the SCI to "sleep" and enable the wakeup function. RIE should be set if interrupts are to be used to receive data. (1) When an address character (ninth bit=1) is received, then R8 is set to one in the SSR, and RWU is cleared. Clearing RWU re-enables the SCI receiver. Since (2) RIE was set in this example, when the first character is received, an SCI receive data interrupt (or SCI receive data with exception status interrupt if an error is detected) will be recognized as pending. When the receiver is ready to wait for another address character, RWU must be set to one again.

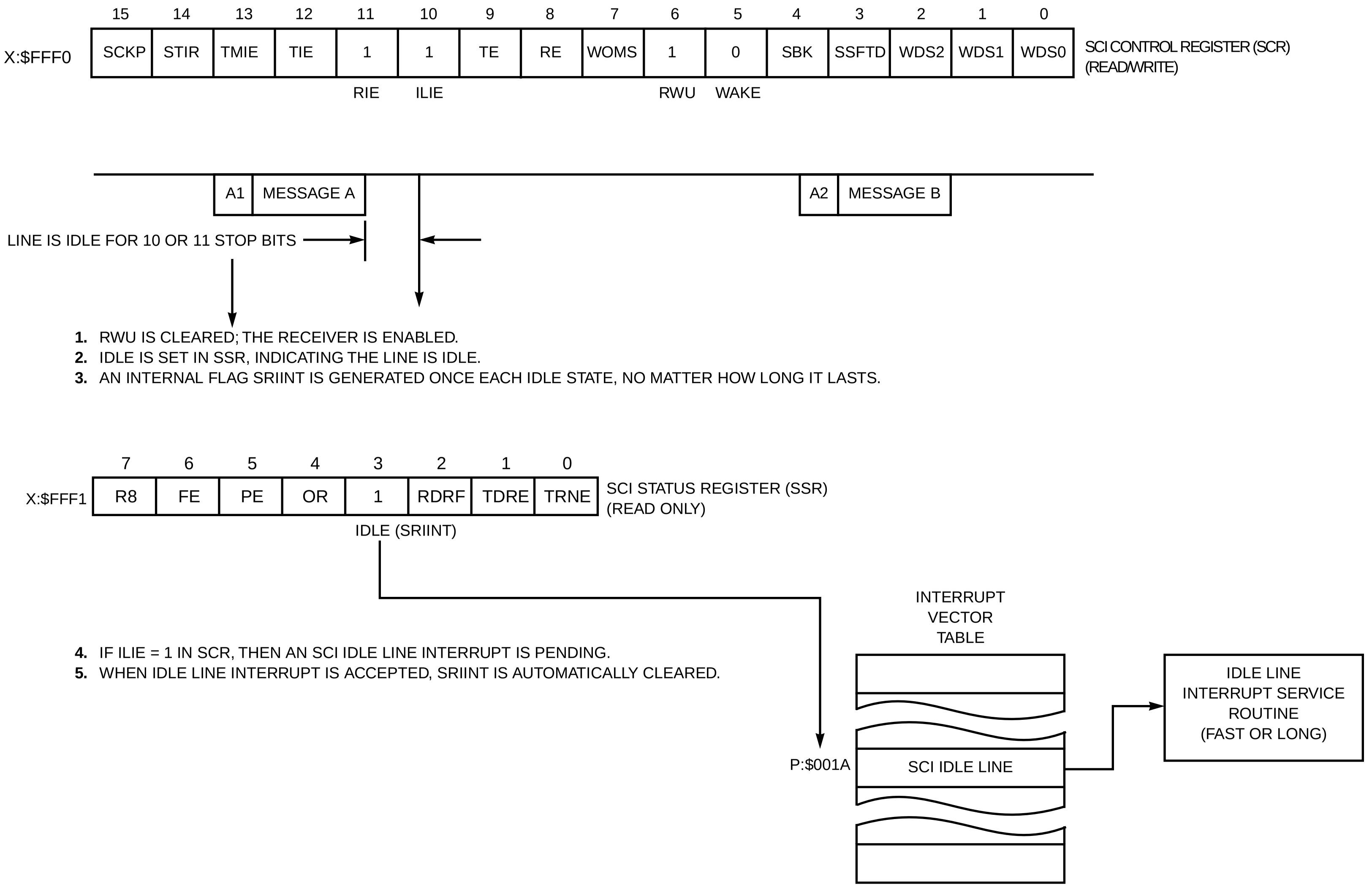
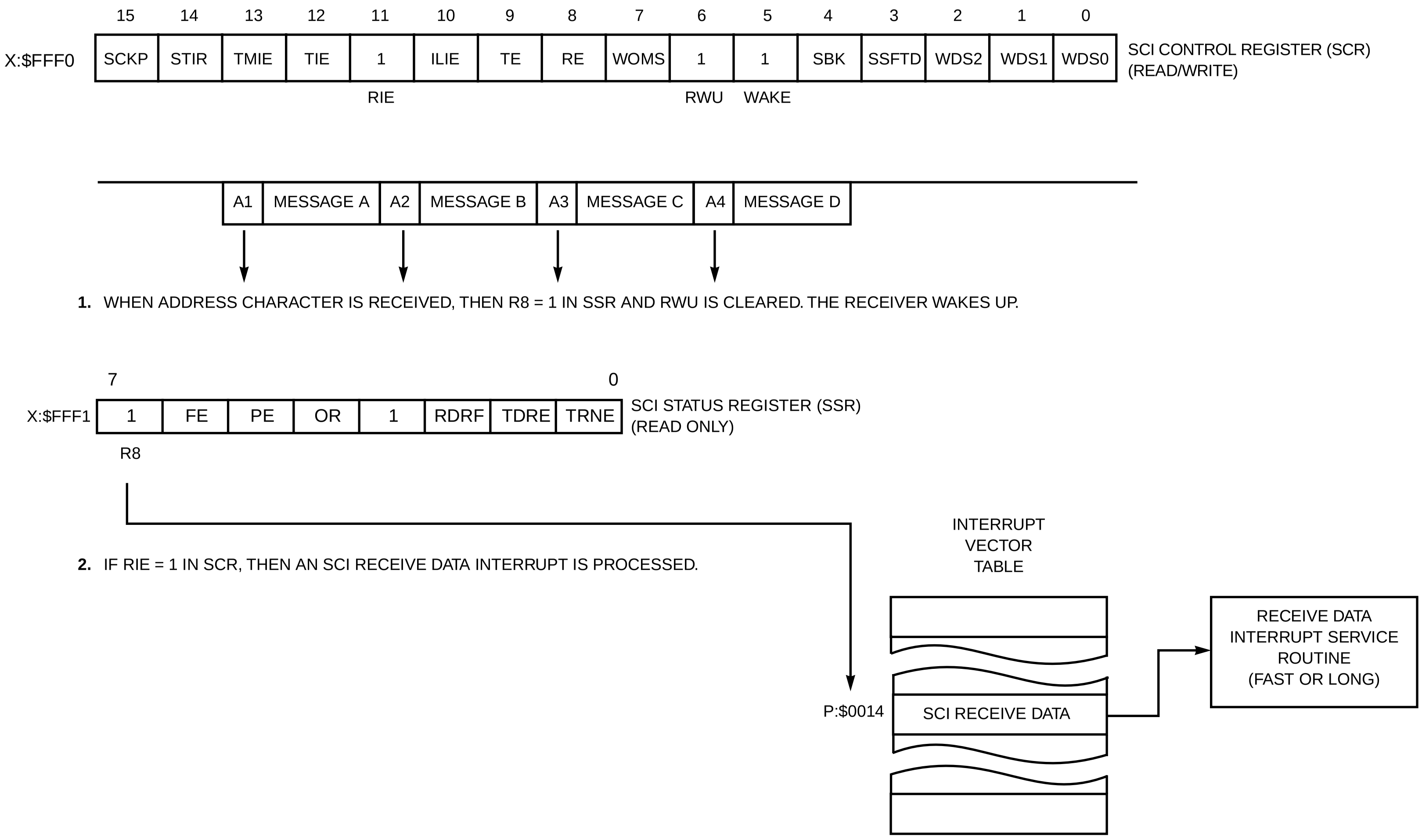

15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
X:$FFF0
SCKP
STIR
TMIE
TIE
1
1
TE
RE
WOMS
1
0
SBK
SSFTD
WDS2
WDS1
WDS0
SCI CONTROL REGISTER (SCR)
(READ/WRITE)
RIE
ILIE
RWU
WAKE
A1
MESSAGE A
A2
MESSAGE B
LINE IS IDLE FOR 10 OR 11 STOP BITS
1. RWU IS CLEARED; THE RECEIVER IS ENABLED.
2. IDLE IS SET IN SSR, INDICATING THE LINE IS IDLE.
3. AN INTERNAL FLAG SRIINT IS GENERATED ONCE EACH IDLE STATE, NO MATTER HOW LONG IT LASTS.
7
6
5
4
3
2
1
0
X:$FFF1
R8
FE
PE
OR
1
RDRF
TDRE
TRNE
SCI STATUS REGISTER (SSR)
(READ ONLY)
IDLE (SRIINT)
4. IF ILIE = 1 IN SCR, THEN AN SCI IDLE LINE INTERRUPT IS PENDING.
5. WHEN IDLE LINE INTERRUPT IS ACCEPTED, SRIINT IS AUTOMATICALLY CLEARED.
INTERRUPT
VECTOR
TABLE
P:$001A
SCI IDLE LINE
IDLE LINE
INTERRUPT SERVICE
ROUTINE
(FAST OR LONG)

Figure 6-32 Idle Line Wakeup



1. WHEN ADDRESS CHARACTER IS RECEIVED, THEN R8 = 1 IN SSR AND RWU IS CLEARED. THE RECEIVER WAKES UP.

2. IF RIE = 1 IN SCR, THEN AN SCI RECEIVE DATA INTERRUPT IS PROCESSED.

Figure 6-33 Address Mode Wakeup

6.3.8.5 Multidrop Example

The program shown in Figure 6-34 configures the SCI as a multidrop master transmitter and slave receiver (using wakeup on address bit) that uses interrupts to transmit data from a circular buffer and to receive data into a different circular buffer. This program can be run with the I/O pins (RXD and TXD) connected and with a pullup resistor for test purposes.

The program starts by setting equates for convenience and clarity and then points the reset vector to the start of the program. The receive and transmit interrupt vector locations have JSRs forming long interrupts because the multidrop protocol and circular buffers require more than two instructions for maintenance. Byte packing and unpacking are not used in this example. The SRX and STX registers are equated to $FFF4, causing only the LSB of the 24-bit DSP word to be used for SCI data. The SCI is then initialized as wired-OR, multidrop, and using interrupts. The SCI is enabled but the interrupts are masked, which prevents the SCI from transmitting or receiving data at this time.

The circular buffers used have two pointers. The first points to the first data byte; the second points to the last data byte. This configuration allows the transmit buffer to act as a first-in first-out (FIFO) memory. The FIFO can be loaded by a program and emptied by the SCI in real time. As long as the number of data bytes never exceeds the buffer size, there will be no overflow or underflow of the buffer. Registers M0-M3 must be loaded with the buffer size minus one to make pointer registers R0-R3 work as circular pointers. Register N2 is used as a constant to clear the receive buffer empty flag.

The main program starts by filling the transmit buffer with a data packet. When the transmit buffer is full, it calls the subroutine that transmits the slave’s address and then jumps to self (SEND jmp SEND), allowing interrupts to transmit and receive the data.

The receive subroutine first checks each byte to see if it is address or data. If it is an address, it compares the address with its own. If the addresses do not match, the SCI is put back to sleep. If the addresses match, the SCI is left awake, and control is returned to the main program. If the byte is data, it is placed in the receive buffer, and the receive buffer empty flag is cleared. Although this flag is not used in this program, it can be used by another program as a simple test to see if data is available. Using N2 as the constant $0 allows the flag to be cleared with a single-word instruction, which can be part of a fast interrupt.

The transmit subroutine transmits a byte and then checks to see if the transmit buffer is empty. If the buffer is not empty, control is returned to the main program, and interrupts are allowed to continue emptying the buffer. If the buffer is empty, the transmit buffer empty flag is set, the transmit interrupt is disabled, and control is returned to the main program.

The wakeup subroutine transmits the slave’s address by writing the address to the STXA register and by enabling the transmit interrupt to allow interrupts to empty the transmit buffer. Control is then returned to the main program.

```
;****************************************************************
;    MULTIDROP MASTER/SLAVE WITH INTERRUPTS AND CIRCULAR BUFFERS*
;***************************************************************
;*************************************************
;    SCI and other EQUATES*
;*************************************************
START    EQU   $0040          ;Start of program
TX_BUFF  EQU   $0010          ;Transmit buffer location
RX_BUFF  EQU   $0020          ;Receive buffer location
B_SIZE   EQU   $000E          ;Transmit and receive buffer size
                              ;(don't allow the TX buffer and RX
                              ;buffers to overlap).
TX_MTY   EQU   $0000          ;Transmit buffer empty
RX_MTY   EQU   $0001          ;Receive buffer empty
PCC      EQU   $FFE1          ;Port C control register
SCR      EQU   $FFF0          ;SCI interface control register
SCCR     EQU   $FFF2          ;SCI clock control register
STXA     EQU   $FFF3          ;SCI transmit address register
SRX      EQU   $FFF4          ;SCI receive register
STX      EQU   $FFF4          ;SCI transmit register
BCR      EQU   $FFFE          ;Bus control register
IPR      EQU   $FFFF          ;Interrupt priority register
;*************************************************
;    RESET VECTOR*
;*************************************************
         ORG   P:$0000
         JMP   START
;*************************************************
;    SCI RECEIVE INTERRUPT VECTOR*
;*************************************************
         ORG   P:$0014   ;Load the SCI RX interrupt vectors
         JSR   RX        ;Jump to the receive routine that puts
                         ;data packet in a circular buffer if it
                         ;is forthis address.
         NOP             ;Second word of fast interrupt not needed
```

Figure 6-34 Multidrop Transmit Receive Example (Sheet 1 of 4)

```
        ORG    P:$0016         ;This interrupt occurs when data is
                               ;received with errors. This example
        NOP                    ;does not trap errors so this
        NOP                    ;interrupt is not used.
;************************************************
;    SCI TRANSMIT INTERRUPT VECTOR*
;************************************************
        ORG    P:$0018         ;Load the SCI TX interrupt vectors
        JSR    TX              ;Transmit next byte in buffer
        NOP
;************************************************
;    INITIALIZE THE SCI PORT*
;************************************************
        ORG    P:START         ;Start the program at location $40
        ORI    #$03,MR         ;Mask interrupts temporarily
        MOVEP  #$C000,X:IPR    ;Set interrupt priority to 2
        MOVEP  #$0BE6,X:SCR    ;Disable TX, enable RX interrupts
                               ;Enable transmitter and receiver,
                               ;Wired-OR mode, Rec. wakeup
                               ;mode,11-bit multidrop (1 start,
                               ;8 data,1 data type, 1 stop)
        MOVEP  #$0000,X:SCCR   ;Use internal TX, RX clocks
                               ;625K BPS at 40 MHz
        MOVEP  #>$03,X:PCC     ;Select pins TXD and RXD for SCI
;*************************************************
;INITIALIZE INTERRUPTS, REGISTERS, ETC.*
;*************************************************
        MOVEP  #$0,X:BCR       ;No wait states
        MOVE   #TX_BUFF,R0     ;Load start pointer of transmit buffer
        MOVE   #TX_BUFF,R1     ;Load end   pointer of transmit buffer
        MOVE   #RX_BUFF,R2     ;Load start pointer of receive buffer
        MOVE   #RX_BUFF,R3     ;Load end pointer of receive buffer
        MOVE   #>$41,R5        ;Init data register... R5 contains
                               ;the data that will be sent in this
                               ;example; it is initialized to an ASCII A.
```

Figure 6-34 Multidrop Transmit Receive Example (Sheet 2 of 4)

```
          MOVE    #B_SIZE,M0     ;Load transmit buffer size
          MOVE    #B_SIZE,M1     ;Load transmit buffer size
          MOVE    #B_SIZE,M2     ;Load receive buffer size
          MOVE    #B_SIZE,M3     ;Load receive buffer size
          MOVE    #>$1,N0        ;Load receive address
          MOVE    #>$1,N1        ;Load first slave address
          MOVE    #0,N2          ;Load a constant (0) into N2
          MOVEP   X:SRX,X:(R0)   ;Clear receive register
;**************************************************
;    MAIN PROGRAM*
;**************************************************
          ANDI    #$FC,MR        ;Re-enable interrupts
          MOVE    (R1)+          ;Temporarily increment the tail pointer
                                 ;Build a packet
LOOP      MOVE    R1,A           ;Check to see if the TX buffer is full
          MOVE    (R1)-          ;(fix tail pointer now that we've used
it)
          MOVE    R0,B           ;by comparing the head and tail pointers
          CMP     A,B            ;of the circular transmit buffer.
          JEQ     SND_BUF        ;if equal, transmit completed packet
          MOVE    R5,X:(R1)+     ;if not, put next character in
                                 ;transmit buffer and
          MOVE    (R5)+          ;increment the pointers.
          MOVE    (R1)+          ;Temporarily increment the tail
                                 ;pointer to test buffer again
          JMP     LOOP
SND_BUF   JSR     WAKE_UP        ;Wake up proper slave and send packet
SEND      JMP     SEND           ;and allow interrupts to drain
                                 ;the transmit buffer.
```

Figure 6-34 Multidrop Transmit Receive Example (Sheet 3 of 4)

```
;**********************************************************************
; SUBROUTINE TO READ SCI AND STORE IN BUFFER USING A LONG INTERRUPT*
;**********************************************************************
RX        JCLR    #7,X:$FFF1,RX_DATA   ;Check if this is address or data.
          MOVEP   X:SRX,A        ;Compare the received address
          MOVE    N1,B           ;with the slave address.
          CMP     A,B
          JEQ     END_RX         ;If address OK, use interrupts to Rx
                                 ;packet
          BSET    #6,X:$FFF0     ;if not, go back to sleep
          JMP     END_RX         ;and return to previous program.
RX_DATA   MOVEP   X:SRX,X:(R3)+  ;Put data in buffer,
          MOVE    N2,X:RX_MTY    ;and clear the Rx buffer empty flag
END_RX    RTI                    ;Return to previous program
;**********************************************************************
;    SUBROUTINE TO WRITE BUFFER TO SCI USING A LONG INTERRUPT*
;**********************************************************************
TX        MOVEP   X:(R0)+,X:STX  ;Transmit a byte and increment the
                                 ;pointer
          MOVE    R0,A           ;Check to see if the TX buffer is
                                 ;empty
          MOVE    R1,B
          CMP     A,B
          JNE     END_TX         ;If not, return to main
          MOVE    #$000001,X0    ;If it is, set the TX buffer empty flag
          MOVE    X0,X:TX_MTY
          BCLR    #12,X:SCR      ;disable transmit interrupts, and
END_TX    RTI                    ;return to main
;**********************************************************************
;    SUBROUTINE TO WAKE UP THE ADDRESSED SLAVE*
;**********************************************************************
WAKE_UP   MOVEP   N1,X:STXA      ;Transmit slave address using STXA
                                 ;not STX
          BSET    #12,X:SCR      ;Enable transmit interrupts to send
                                 ;packet
AWAKE     RTI
          END                    ;End of example.
```

Figure 6-34 Multidrop Transmit/Receive Example (Sheet 4 of 4)

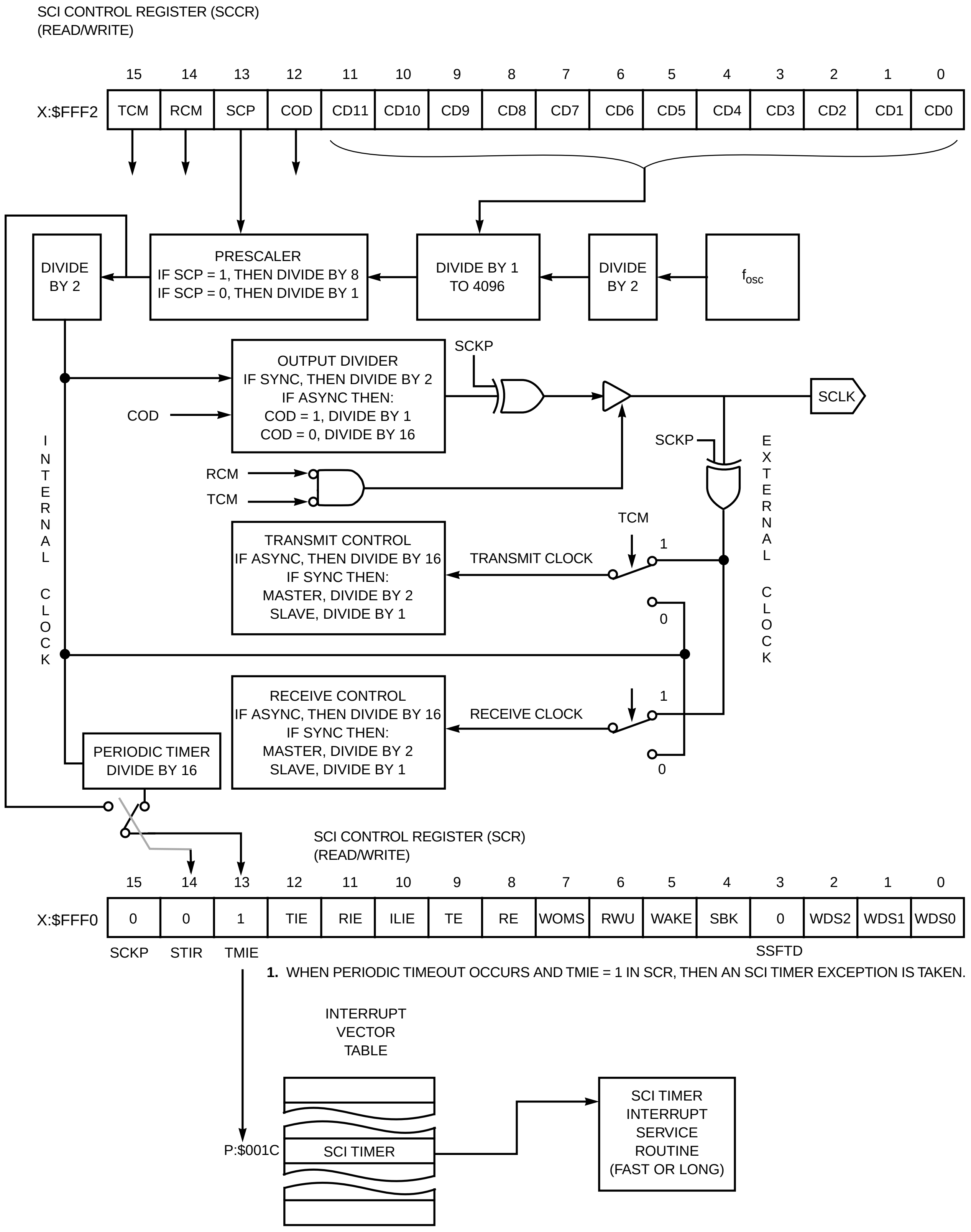
6.3.9 SCI Timer

The SCI clock determines the data transmission rate and can also be used to establish a periodic interrupt that can act as an event timer or be used in any other timing function. Figure 6-35 illustrates how the SCI timer is programmed. Bits CD11–CD0, SCP, and STIR in the SCCR work together to determine the time base. The crystal oscillator f_{osc} is first divided by 2 and then divided by the number CD11–CD0 in the SCCR. The oscillator is then divided by 1 (if SCP=0) or eight (if SCP=1). This output is used as is if STIR = 1 or, if STIR = 0, it is divided by 2 and then by 16 before being used. If TMIE in the SCR = 1 when the periodic timeout occurs, the SCI timer interrupt is recognized and pending. The SCI timer interrupt is automatically cleared when the interrupt is serviced. This interrupt will occur every time the periodic timer times out. If only the timer function is being used (i.e., PC0, PC1, and PC2 pins have been programmed as GPIO pins), the transmit interrupts should be turned off (TIE=0). Under individual reset, TDRE will remain set and the timer will continuously generate interrupts.

Figure 6-35 shows that an external clock can be used for SCI receive and/or transmit, which frees the SCI timer to be programmed for a different interrupt rate. In addition, both the SCI timer interrupt and the SCI can use the internal time base if the SCI receiver and/or transmitter require the same clock period as the SCI timer.

The program in Figure 6-36 configures the SCI to interrupt the DSP at fixed intervals. The program starts by setting equates for convenience and clarity and then points the reset vector to the start of the program. The SCI timer interrupt vector location contains “move (R0)+”, incrementing the contents of R0, which serves as an elapsed time counter.

The timer initialization consists of enabling the SCI timer interrupt, setting the SCI baud rate counters for the desired interrupt rate, setting the interrupt mask, enabling the interrupt, and then enabling the SCI state machine.



SCI CONTROL REGISTER (SCCR) (READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFF2	TCM	RCM	SCP	COD	CD11	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0

SCI CONTROL REGISTER (SCR) (READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFF0	0	0	1	TIE	RIE	ILIE	TE	RE	WOMS	RWU	WAKE	SBK	0	WDS2	WDS1	WDS0
	SCKP	STIR	TMIE										SSFTD			

1. WHEN PERIODIC TIMEOUT OCCURS AND TMIE = 1 IN SCR, THEN AN SCI TIMER EXCEPTION IS TAKEN.

2. PENDING TIMER INTERRUPT IS AUTOMATICALLY CLEARED WHEN INTERRUPT IS SERVICED.

Figure 6-35 SCI Timer Operation

; *

```
;         TIMER USING SCI TIMER INTERRUPT*
;********************************************************************
;*************************************************
;         SCI and other EQUATES*
;*************************************************
START     EQU     $0040          ;Start of program
SCR       EQU     $FFF0          ;SCI control register
SCCR      EQU     $FFF2          ;SCI clock control register
IPR       EQU     $FFFF          ;Interrupt priority register
;*************************************************
;         RESET VECTOR*
;*************************************************
          ORG     P:$0000
          JMP     START
;*************************************************
;         SCI TIMER INTERRUPT VECTOR*
;*************************************************
          ORG     P:$001C        ;Load the SCI timer interrupt vectors
          MOVE    (R0)+          ;Increment the timer interrupt counter
          NOP                    ;This timer routine is implemented
                                 ;as a fast interrupt
;*************************************************
;         INITIALIZE THE SCI PORT*
;*************************************************
          ORG     P:START        ;Start the program at location $40
          MOVE    #0,R0          ;Initialize the timer interrupt counter
          MOVEP   #$2000,X:SCR   ;Select the timer interrupt
          MOVEP   #$013F,X:SCCR  ;Set the interrupt rate at 1 ms
                                 ;(arbitrarily chosen)
                                 ;Interrupts/second =
                                 ;fosc/(64X(7(SCP)-+1)X(CD+1))
                                 ;Note that this is the same equation
                                 ;as for SCI async baud rate
```

Figure 6-36 SCI Timer Example (Sheet 1 of 2)

```
;For 1 ms, SCP=0,
                              ;CD=0001 0011 1111.
        MOVEP   #$C000,X:IPR  ;Set the interrupt priority level–
                              ;application specific.
        ANDI    #$FC,MR       ;Enable interrupts, set MR bits I1 and
                              ;I0=0
END     JMP     END           ;Normally something more useful
                              ;would be put here.
        END                   ;End of example.
```

Figure 6-36 SCI Timer Example (Sheet 2 of 2)

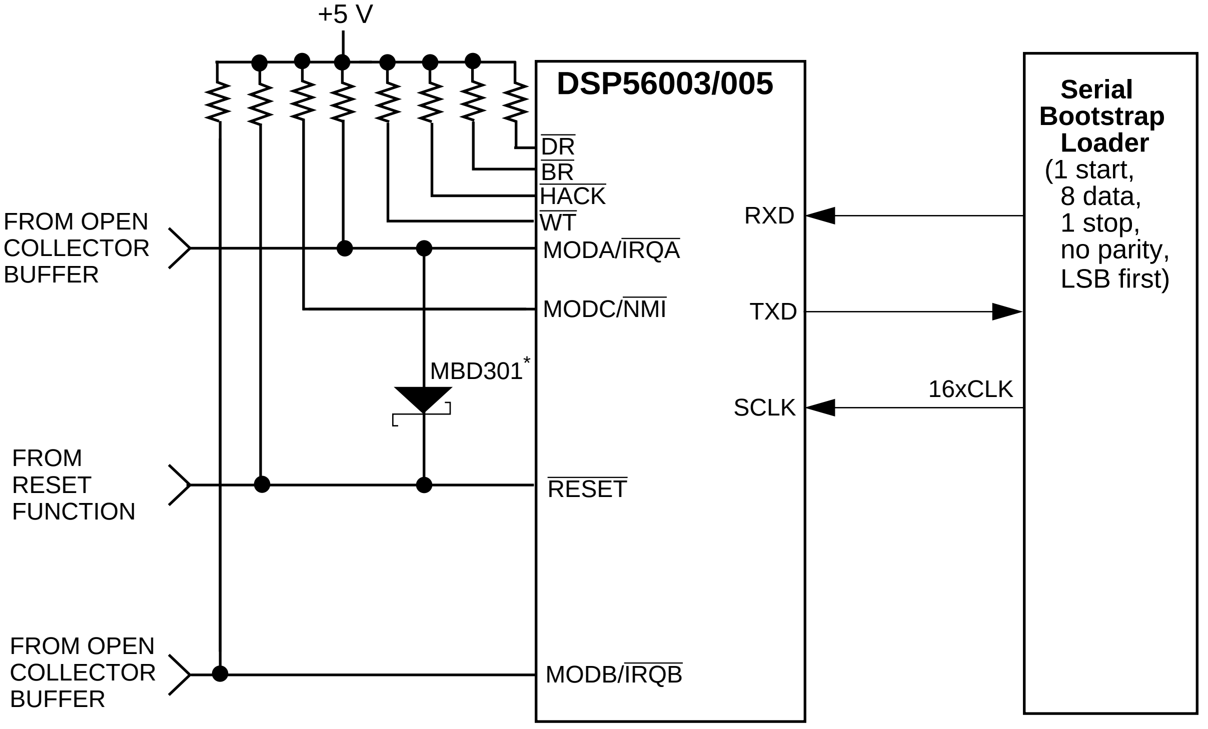
6.3.10 Bootstrap Loading Through the SCI (Operating Mode 6)

When the DSP comes out of reset, it looks at the MODC, MODB, and MODA pins and sets the corresponding mode bits in the OMR. If the mode bits are set to 110 respectively, the DSP will load the program RAM from the SCI. Figure 6-37 shows how the SCI is configured for receiving this code and Figure 6-37 shows the segment of bootstrap code that is used to load from the SCI. The complete code used in the bootstrap program is given in **APPENDIX A.** This program (1) configures the SCI, (2) loads the program size, (3) loads the location where the program will begin loading in program memory, and (4) loads the program.

First, the SCI Control Register is set to $0302 (see Figure 5-2) 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 $C000 which configures the SCI to use external receive and transmit clocks on the SCLK pin. This **clock** must be **16 times the 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 will be 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 loading the program, the operating mode is set to zero, the CCR is cleared, and the DSP begins execution with the first instruction that was loaded.



Notes: 1. ***These diodes **must** be Schottky diodes.
2. All resistors are 15KΩ unless noted otherwise.
3. When in RESET, IRQA, IRQB and NMI must be deasserted by external peripherals.

Figure 6-37 DSP56003/005 Bootstrap Example - Mode 6

6.3.11 Example Circuits

The SCI can be used in a number of configurations to connect multiple processors. The synchronous mode shown in Figure 6-39 shows the DSP acting as a slave. The 8051 provides the clock that clocks data in and out of the SCI, which is possible because the SCI shift register mode timing is compatible with the timing for 8051/8096 processors. Transmit data is changed on the negative edge of the clock, and receive data is latched on the positive edge of the clock. A protocol must be used to prevent both processors from transmitting simultaneously. The DSP is also capable of being the master device.

A multimaster system can be configured (see Figure 6-40) using a single transmit/receive line, multidrop word format, and wired-OR. The use of wired-OR requires a pullup resistor as shown. A protocol must be used to prevent collisions. This scheme is physically the simplest multiple DSP interconnection because it uses only one wire and one resistor.

```
; This is the routine that loads from the SCI.
; MC:MB:MA=110 - external SCI clock
; MC:MB:MA=111 - reserved

        ORG PL:$0D00,PL:$0D00 ; starting address of 2nd ROM
SCILD   MOVEP #$0302,X:SCR    ; Configure SCI Control Reg
        MOVEP #$C000,X:SCCR   ; Configure SCI Clock Control Reg
        MOVEP #7,X:PCC        ; Configure SCLK, TXD and RXD

_SCI1   DO #6,_LOOP6          ; get 3 bytes for number of
                              ; program words and 3 bytes
                              ; for the starting address
        JCLR #2,X:SSR,*       ; Wait for RDRF to go high
        MOVEP X:SRXL,A2       ; Put 8 bits in A2
        JCLR #1,X:SSR,*       ; Wait for TDRE to go high
        MOVEP A2,X:STXL       ; echo the received byte
        REP #8
        ASR A
_LOOP6
        MOVE A1,R0            ; starting address for load
        MOVE A1,R1            ; save starting address
        DO A0,_LOOP4          ; Receive program words

        DO #3,_LOOP5
        JCLR #2,X:SSR,*       ; Wait for RDRF to go high
        MOVEP X:SRXL,A2       ; Put 8 bits in A2
        JCLR #1,X:SSR,*       ; Wait for TDRE to go high
        MOVEP A2,X:STXL       ; echo the received byte
        REP #8
        ASR A
_LOOP5
        MOVEM A1,P:(R0)+      ; Store 24-bit result in P mem.
_LOOP4
        JMP FINISH+1          ; Boot from SCI done
```

Figure 6-38 Bootstrap Code Fragment

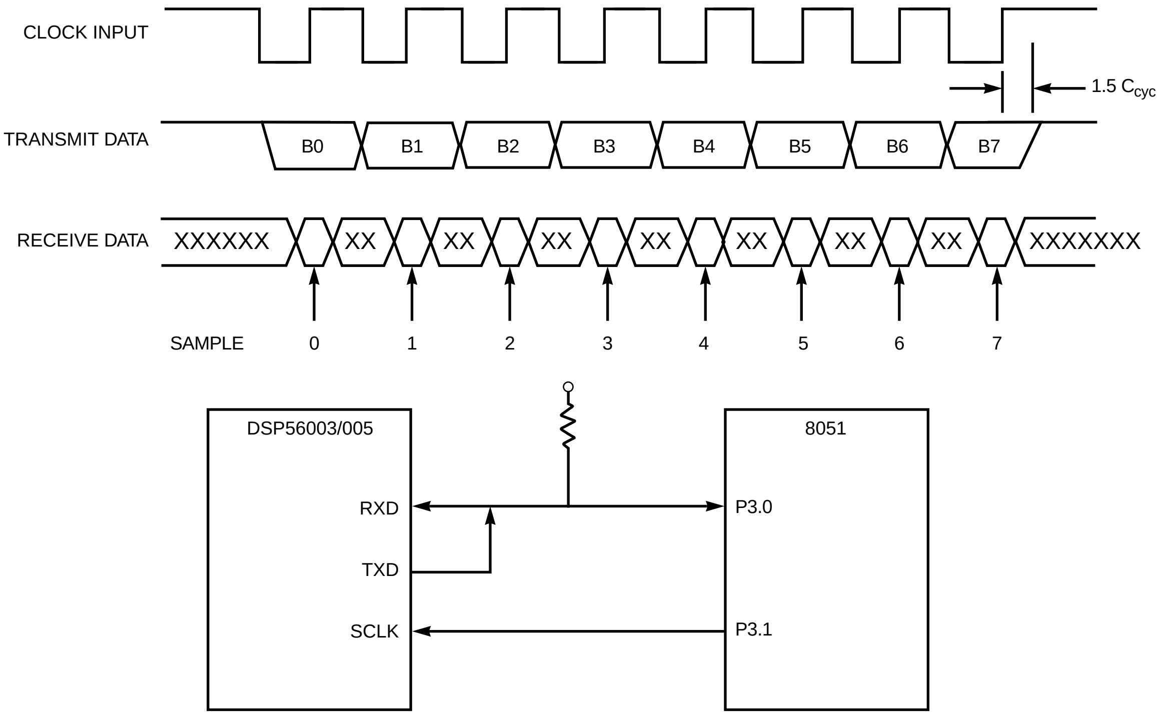


Figure 6-39 Synchronous Mode Example

The master-slave system shown in Figure 6-41 is different in that it is full duplex. The clock pin is not required; thus, it is configured as a GPIO pin. Communication is asynchronous. The slave's transmitters must be wire-ORed because more than one transmitter is on one line. The master's transmitter does not need to be wire-ORed.

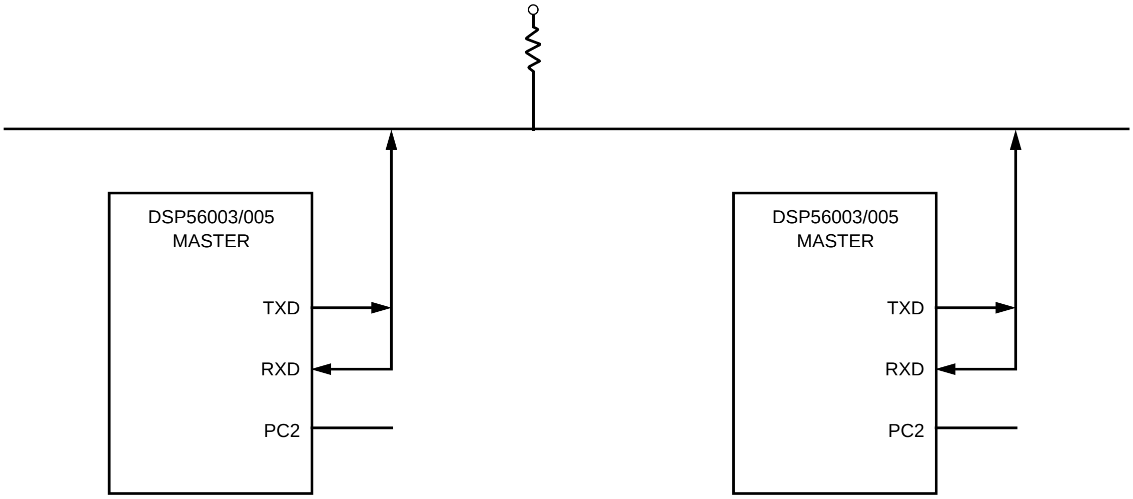


Figure 6-40 Multimaster System Example

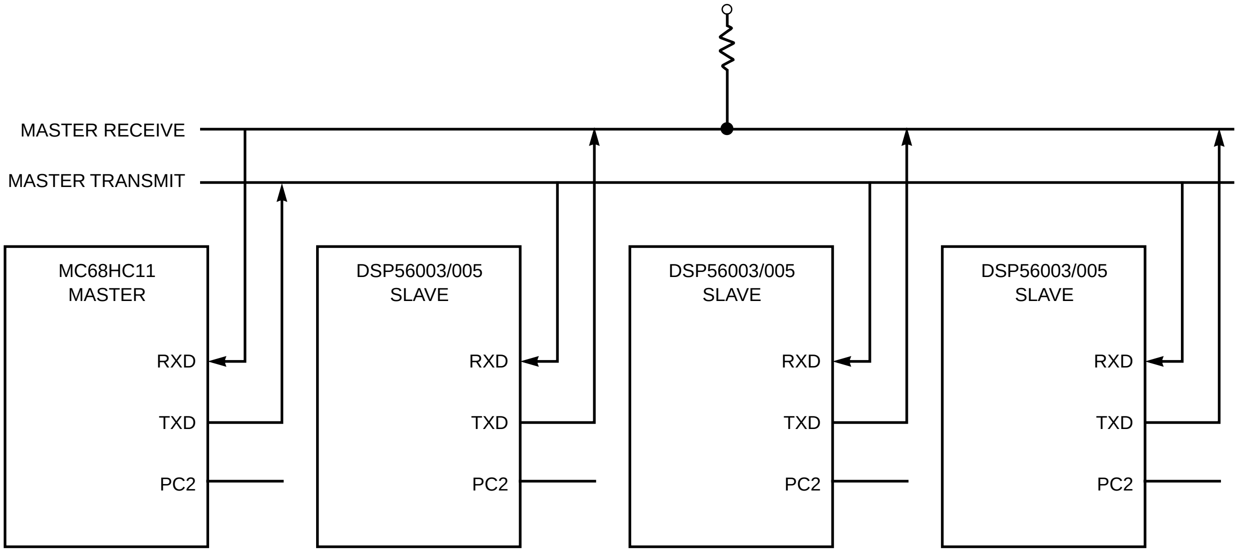
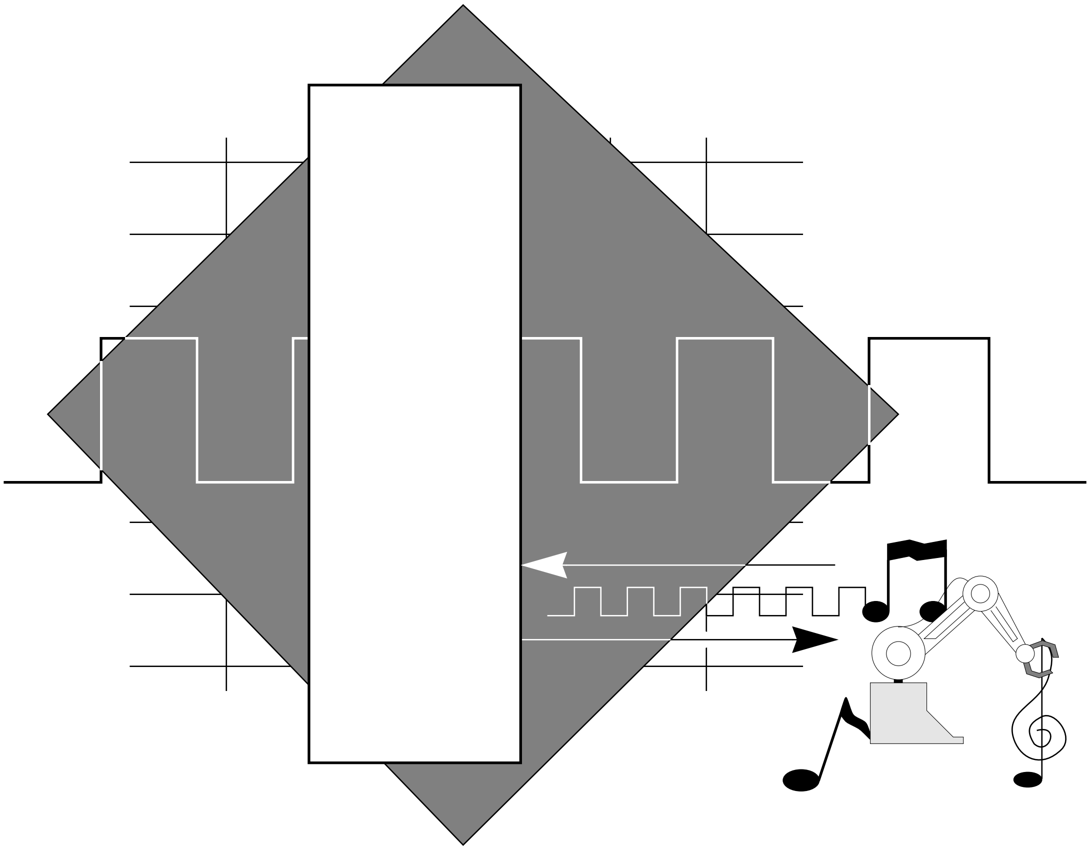


Figure 6-41 Master-Slave System Example

SECTION 7

SYNCHRONOUS SERIAL INTERFACE



SECTION CONTENTS

7.1 INTRODUCTION

Port C is a triple-function I/O port with nine pins (see Figure 7-1). Three of the nine pins can be configured as general-purpose I/O or as the serial communication interface (SCI) pins. The other six pins can also be configured as GPIO, or they can be configured as the synchronous serial interface (SSI) pins.

When configured as general-purpose I/O, port C can be used for device control. When the pins are configured as a SSI, port C provides a convenient connection to other DSPs, processors, codecs, digital-to-analog and analog-to-digital converters, and any of several transducers. This Port C (SSI and GPIO) is identical to the one on the DSP56001 and DSP56002.

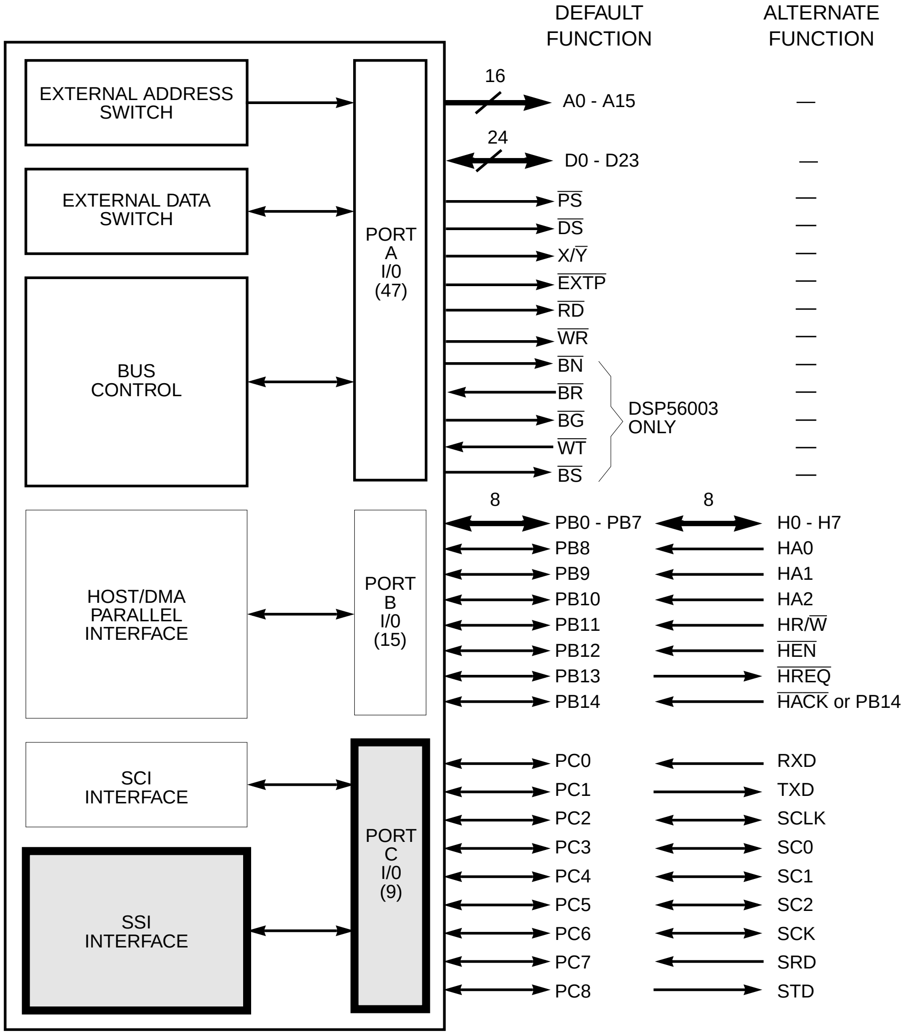
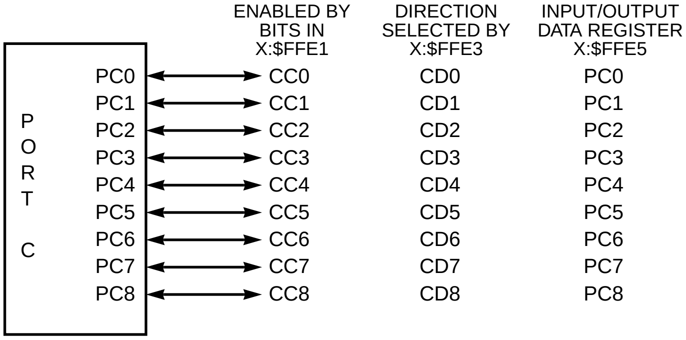


Figure 7-1 Port C Interface

7.2 GENERAL-PURPOSE I/O (PORT C)

When it is configured as GPIO, Port C can be viewed as nine I/O pins (see Figure 7-2), which are controlled by three memory-mapped registers. These registers are the Port C control register (PCC), Port C data direction register (PCDDR), and Port C data register (PCD) (see Figure 7-3).



ENABLED BY BITS IN X:$FFE1	DIRECTION SELECTED BY X:$FFE3	INPUT/OUTPUT DATA REGISTER X:$FFE5
CC0	CD0	PC0
CC1	CD1	PC1
CC2	CD2	PC2
CC3	CD3	PC3
CC4	CD4	PC4
CC5	CD5	PC5
CC6	CD6	PC6
CC7	CD7	PC7
CC8	CD8	PC8

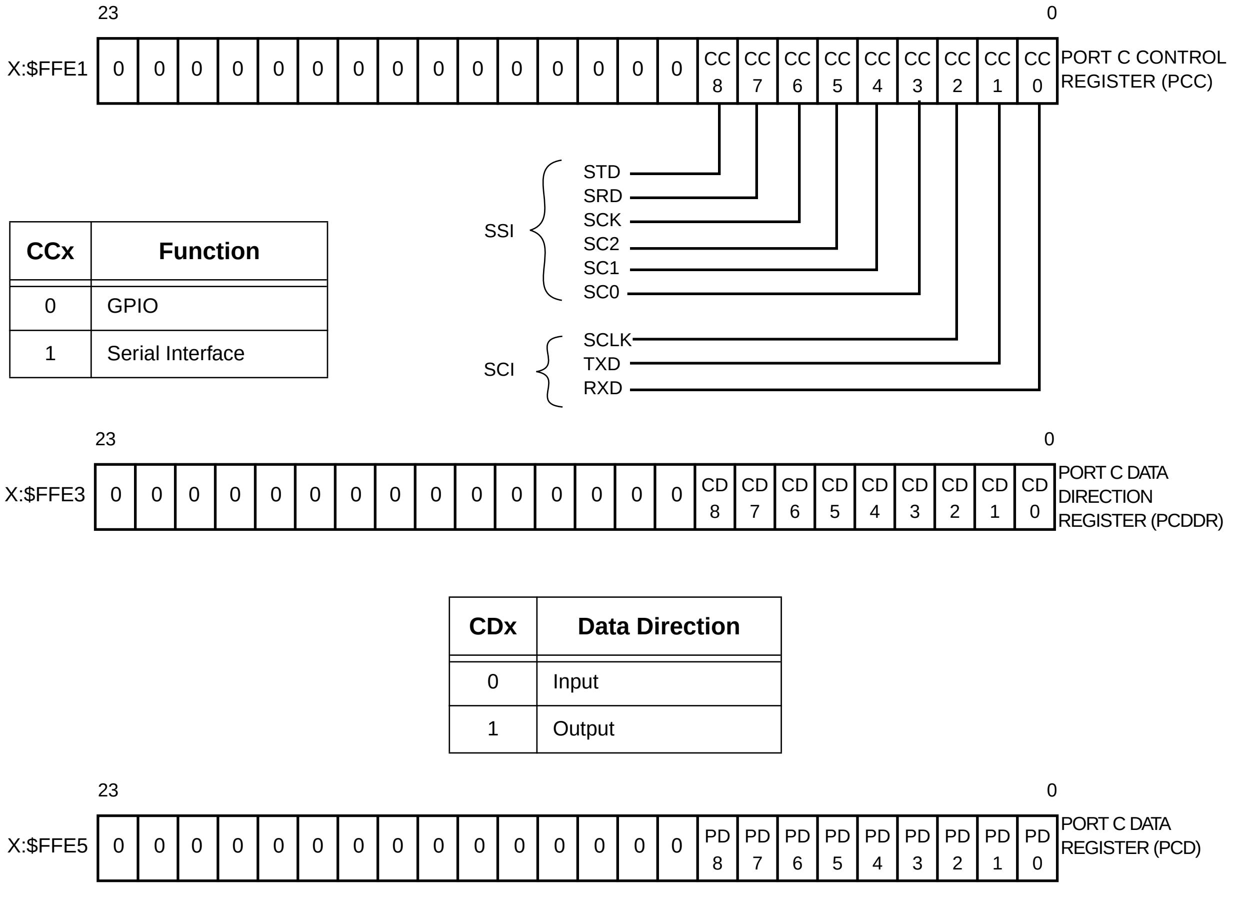
Figure 7-2 Port C GPIO Control

Reset configures Port C as general-purpose I/O with all 9 pins as inputs by clearing both the control (PCC), and data direction (PCDDR) registers (external circuitry connected to these pins may need pullups until the pins are configured for operation). There are three registers associated with each external pin. Each Port C pin may be individually programmed as a general-purpose I/O pin or as a dedicated on-chip peripheral pin under software control. Pin selection between general-purpose I/O and SCI or SSI is made by setting the appropriate PCC bit (memory location X:$FFE1) to zero for general-purpose I/O or to one for serial interface.

The PCDDR (memory location X:$FFE3) programs each pin corresponding to a bit in the PCD (memory location X:$FFE5) as an input pin (if PCDDR=0) or as an output pin (if PCDDR=1).

If a pin is configured as a GPIO **input** (as shown in Figure 7-4) and the processor reads the PCD, the processor sees the logic level on the pin. If the processor writes to the PCD, the data is latched there, but does not appear on the pin because the buffer is in the high-impedance state.

If a pin is configured as a GPIO **output** and the processor reads the PCD, the processor sees the contents of the PCD rather the logic level on the pin, which allows the PCD to be used as a general purpose 15-bit register. If the processor writes to the PCD, the data is latched there and appears on the pin during the following instruction cycle (see **Section 7.2.2**).



CCx	Function
0	GPIO
1	Serial Interface

CDx	Data Direction
0	Input
1	Output

NOTE: Hardware and software reset clears PCC and PCDDR.

Figure 7-3 Port C GPIO Registers

If a pin is configured as a **serial interface** (SCI or SSI) pin, the Port C GPIO registers can be used to help in debugging the serial interface. If the PCDDR bit for a given pin is cleared (configured as an input), the PCD will show the logic level on the pin, regardless of whether the serial interface function is using the pin as an input or an output. If the PCDDR is set (configured as an output) for a given serial interface pin, when the processor reads the PCD, it sees the contents of the PCD rather than the logic level on the pin — another case which allows the PCD to act as a general purpose register.

7.2.1 Programming General Purpose I/O

Port C and all the DSP56003/005 peripherals are memory mapped (see Figure 7-5). The standard MOVE instruction transfers data between Port C and a register; as a result, performing a memory-to-memory data transfer takes two MOVE instructions and a register. The MOVEP instruction is specifically designed for I/O data transfer as shown in Figure 7-6. Although the MOVEP instruction may take twice as long to execute as a MOVE instruction, only one MOVEP is required for a memory-to-memory data transfer, and MOVEP does not use a temporary register.

Port Control Register Bit	Data Direction Register Bit	Pin Function
0	0	Port Input Pin

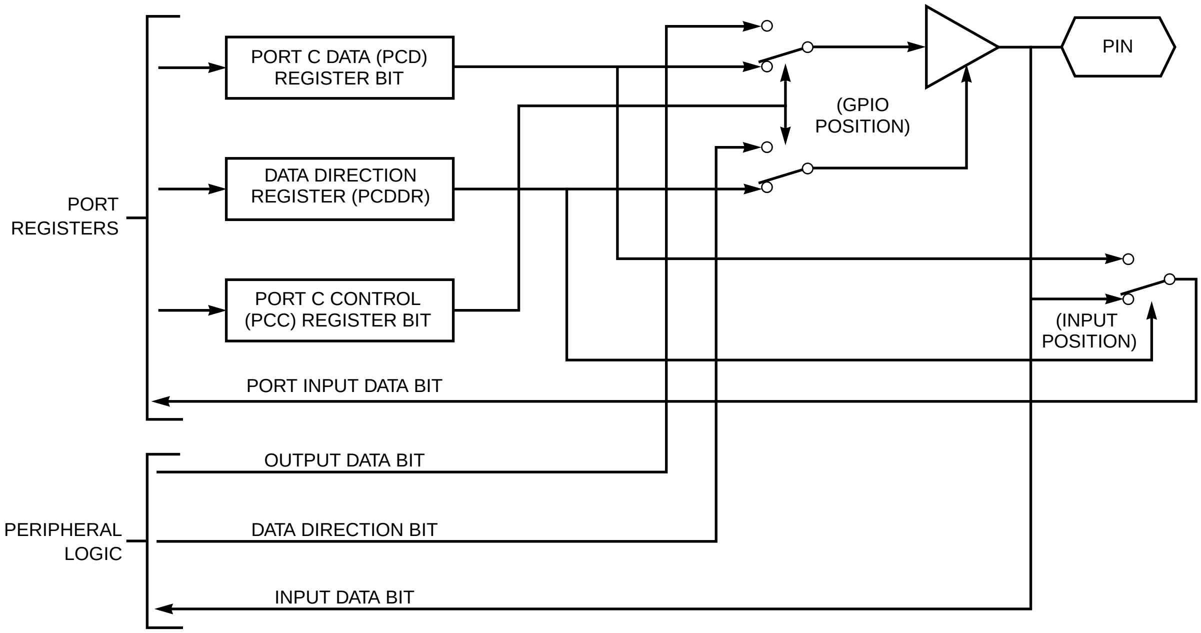
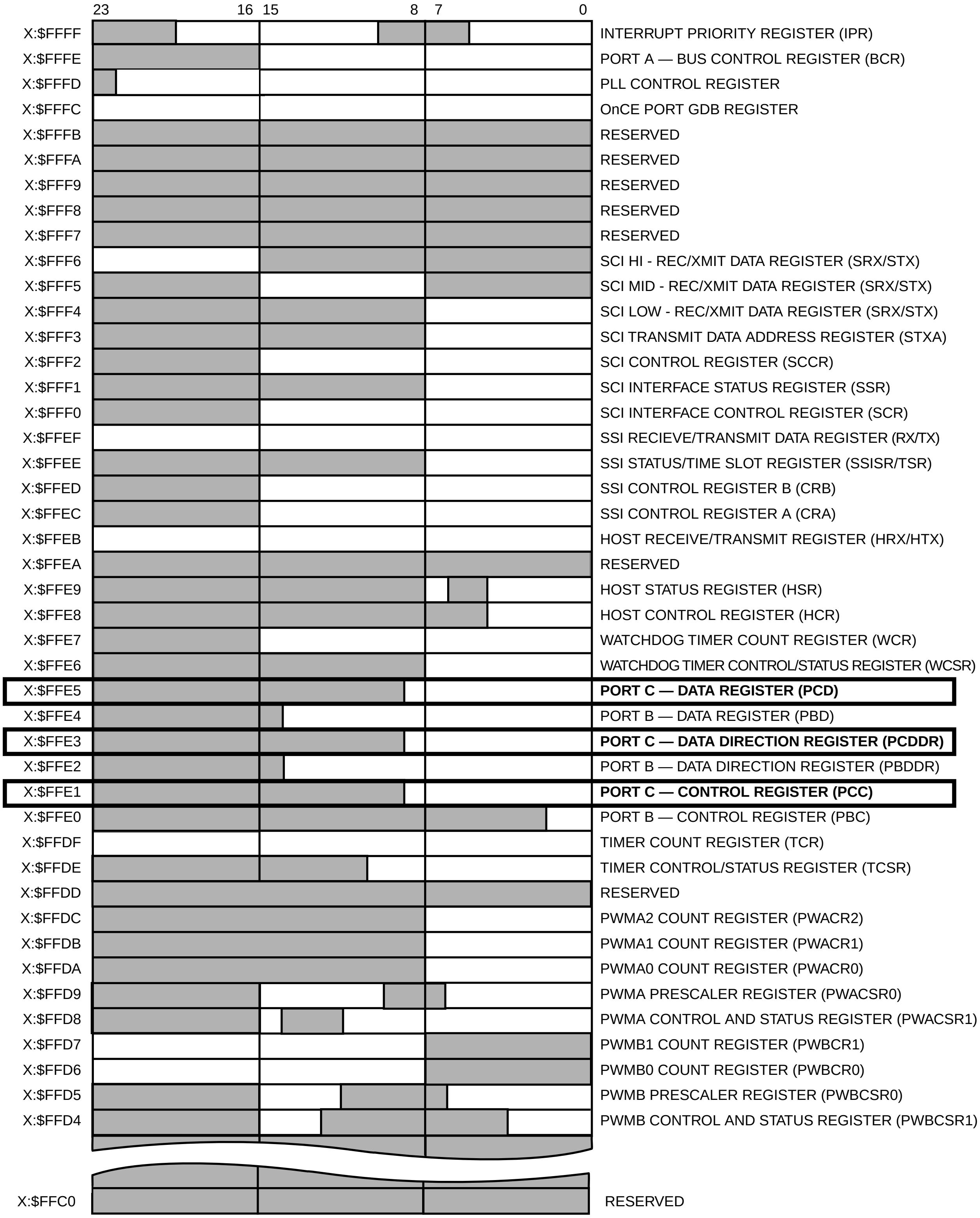


Figure 7-4 Port C I/O Pin Control Logic

Using the MOVEP instruction allows a fast interrupt to move data to/from a peripheral to memory and execute one other instruction or to move the data to an absolute address. MOVEP is the only memory-to-memory move instruction; however, one of the operands must be in the top 64 locations of either X: or Y: memory. The bit-oriented instructions which use I/O short addressing (BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, and JSSET) can also be used to address individual bits for faster I/O processing.

The DSP does not have a hardware data strobe to strobe data out of the GPIO port. If a data strobe is needed, it can be implemented using software to toggle one of the GPIO pins.

Figure 7-7 shows the process of programming Port C as general-purpose I/O. Normally, it is not good programming practice to activate a peripheral before programming it. However, reset activates the Port C general-purpose I/O as all inputs, and the alternative is to configure the port as an SCI and/or SSI, which may not be desirable. In this case, it is probably better to insure that Port C is initially configured for general-purpose I/O and then configure the data direction and data registers.



X:$FFFF	INTERRUPT PRIORITY REGISTER (IPR)
X:$FFFE	PORT A — BUS CONTROL REGISTER (BCR)
X:$FFFD	PLL CONTROL REGISTER
X:$FFFC	OnCE PORT GDB REGISTER
X:$FFFB	RESERVED
X:$FFFA	RESERVED
X:$FFF9	RESERVED
X:$FFF8	RESERVED
X:$FFF7	RESERVED
X:$FFF6	SCI HI - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF5	SCI MID - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF4	SCI LOW - REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF3	SCI TRANSMIT DATA ADDRESS REGISTER (STXA)
X:$FFF2	SCI CONTROL REGISTER (SCCR)
X:$FFF1	SCI INTERFACE STATUS REGISTER (SSR)
X:$FFF0	SCI INTERFACE CONTROL REGISTER (SCR)
X:$FFEF	SSI RECIEVE/TRANSMIT DATA REGISTER (RX/TX)
X:$FFEE	SSI STATUS/TIME SLOT REGISTER (SSISR/TSR)
X:$FFED	SSI CONTROL REGISTER B (CRB)
X:$FFEC	SSI CONTROL REGISTER A (CRA)
X:$FFEB	HOST RECEIVE/TRANSMIT REGISTER (HRX/HTX)
X:$FFEA	RESERVED
X:$FFE9	HOST STATUS REGISTER (HSR)
X:$FFE8	HOST CONTROL REGISTER (HCR)
X:$FFE7	WATCHDOG TIMER COUNT REGISTER (WCR)
X:$FFE6	WATCHDOG TIMER CONTROL/STATUS REGISTER (WCSR)
X:$FFE5	**PORT C — DATA REGISTER (PCD)**
X:$FFE4	PORT B — DATA REGISTER (PBD)
X:$FFE3	**PORT C — DATA DIRECTION REGISTER (PCDDR)**
X:$FFE2	PORT B — DATA DIRECTION REGISTER (PBDDR)
X:$FFE1	**PORT C — CONTROL REGISTER (PCC)**
X:$FFE0	PORT B — CONTROL REGISTER (PBC)
X:$FFDF	TIMER COUNT REGISTER (TCR)
X:$FFDE	TIMER CONTROL/STATUS REGISTER (TCSR)
X:$FFDD	RESERVED
X:$FFDC	PWMA2 COUNT REGISTER (PWACR2)
X:$FFDB	PWMA1 COUNT REGISTER (PWACR1)
X:$FFDA	PWMA0 COUNT REGISTER (PWACR0)
X:$FFD9	PWMA PRESCALER REGISTER (PWACSR0)
X:$FFD8	PWMA CONTROL AND STATUS REGISTER (PWACSR1)
X:$FFD7	PWMB1 COUNT REGISTER (PWBCR1)
X:$FFD6	PWMB0 COUNT REGISTER (PWBCR0)
X:$FFD5	PWMB PRESCALER REGISTER (PWBCSR0)
X:$FFD4	PWMB CONTROL AND STATUS REGISTER (PWBCSR1)
X:$FFC0	RESERVED

▨ = Read as random number; write as don't care.

Figure 7-5 On-Chip Peripheral Memory Map

It may be better in some situations to program the data direction or the data regis-

```
:
:
MOVEP   #$0,X:$FFE1     ;Select Port C to be general-purpose I/O
MOVEP   #$01F0,X:$FFE3 ;Select pins PC0–PC3 to be inputs
:                       ;and pins PC4–PC8 to be outputs
:
MOVEP   #data_out,X:$FFE5 ;Put bits 4–8 of “data_out” on pins
                          ;PB4–PB8 bits 0–3 are ignored.
MOVEP   X:$FFE0,#data_in  ;Put PB0–PB3 in bits 0–3 of “data_in”
```

Figure 7-6 Write/Read Parallel Data with Port C

ters first to prevent two devices from driving one signal. The order of steps 1, 2, and 3 in Figure 7-7 is optional and can be changed as needed.

7.2.2 Port C General Purpose I/O Timing

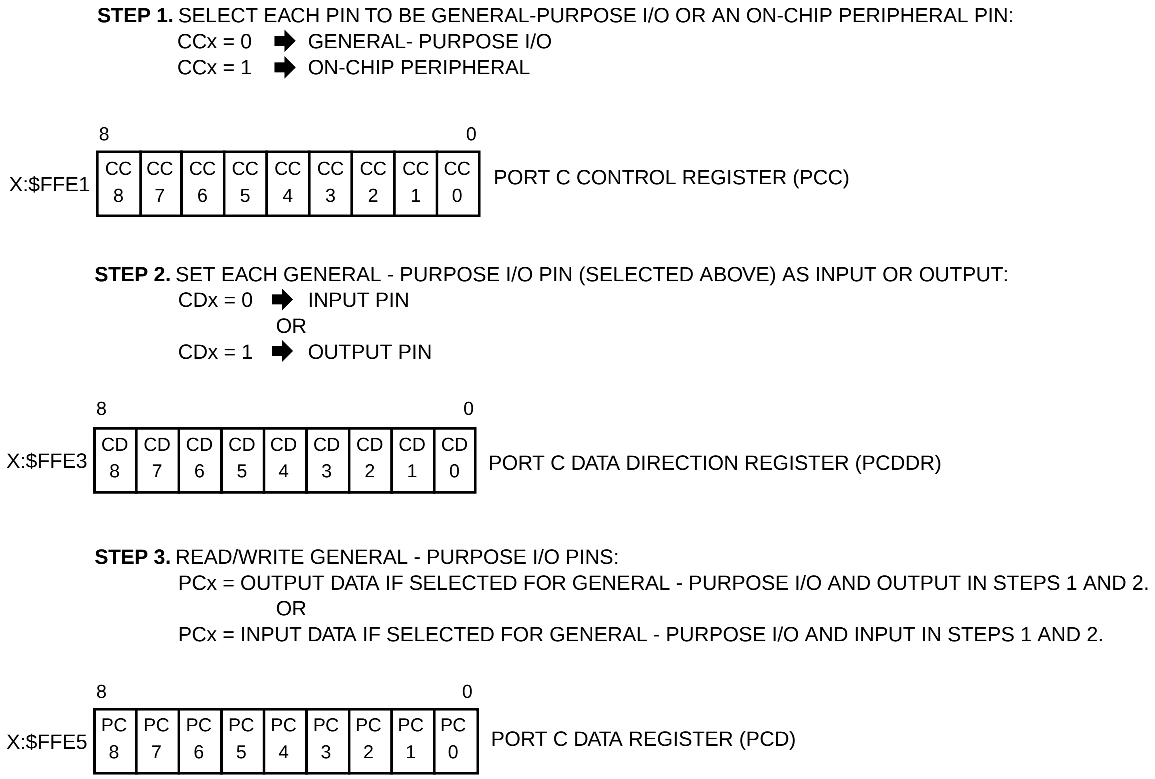
Parallel data written to Port C is delayed by one instruction cycle. For example, the following instruction:

```
MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN
```

1. writes nine bits of data to the Port C register, but the output pins do not change until the following instruction cycle
2. writes 24 bits of data to the external Y memory, which appears on Port A during T2 and T3 of the current instruction

As a result, if it is necessary to synchronize the Port A and Port C outputs, two instructions must be used:

```
MOVE    DATA9,X:PORTC
NOP                      DATA24,Y:EXTERN
```



STEP 1. SELECT EACH PIN TO BE GENERAL-PURPOSE I/O OR AN ON-CHIP PERIPHERAL PIN:
CCx = 0 ➡ GENERAL- PURPOSE I/O
CCx = 1 ➡ ON-CHIP PERIPHERAL

	8								0	
X:$FFE1	CC 8	CC 7	CC 6	CC 5	CC 4	CC 3	CC 2	CC 1	CC 0	PORT C CONTROL REGISTER (PCC)

STEP 2. SET EACH GENERAL - PURPOSE I/O PIN (SELECTED ABOVE) AS INPUT OR OUTPUT:
CDx = 0 ➡ INPUT PIN
OR
CDx = 1 ➡ OUTPUT PIN

	8								0	
X:$FFE3	CD 8	CD 7	CD 6	CD 5	CD 4	CD 3	CD 2	CD 1	CD 0	PORT C DATA DIRECTION REGISTER (PCDDR)

STEP 3. READ/WRITE GENERAL - PURPOSE I/O PINS:
PCx = OUTPUT DATA IF SELECTED FOR GENERAL - PURPOSE I/O AND OUTPUT IN STEPS 1 AND 2.
OR
PCx = INPUT DATA IF SELECTED FOR GENERAL - PURPOSE I/O AND INPUT IN STEPS 1 AND 2.

	8								0	
X:$FFE5	PC 8	PC 7	PC 6	PC 5	PC 4	PC 3	PC 2	PC 1	PC 0	PORT C DATA REGISTER (PCD)

Figure 7-7 I/O Port C Configuration

The NOP can be replaced by any instruction that allows parallel moves. Inserting one or more “MOVE DATA15,X:PORTC DATA24,Y:EXTERN” instructions between the first and second instruction produces an external 33-bit write each instruction cycle with only one instruction cycle lost in setup time:

```
MOVE    DATA9,X:PORTC
MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN
MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN
:
:
MOVE    DATA9,X:PORTC    DATA24,Y:EXTERN
NOP                      DATA24,Y:EXTERN
```

One application of this technique is to create an extended address for Port A by concatenating the Port A address bits (instead of data bits) to the Port C general-purpose output bits. The Port C general-purpose I/O register would then work as a base address register, allowing the address space to be extended from 64K words (16 bits) to 33.5 million words (16 bits+ 9 bits=25 bits).

Port C uses the DSP central processing unit (CPU) four-phase clock for its operation. Therefore, if wait states are inserted in the DSP CPU timing, they also affect Port C timing. As a result, Port A and Port C in the previous synchronization example will always stay synchronized, regardless of how many wait states are used.

7.3 SYNCHRONOUS SERIAL INTERFACE (SSI)

The synchronous serial interface (SSI) 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 which implement the Motorola SPI.

The user can independently define the following characteristics of the SSI: the number of bits per word, the protocol, the clock, and the transmit/receive synchronization.

The user can select among three modes: normal, on-demand, and network. The normal mode is typically used to interface with devices on a regular or periodic basis. The data-driven on-demand mode is intended to be used to communicate with devices on a non-periodic basis. The network mode provides time slots in addition to a bit clock and frame synchronization pulse.

The SSI functions with a range of 2 to 32 words of I/O per frame in the network mode. This mode is typically used in star or ring time division multiplex networks with other DSP56K processors and/or codecs. The clock can be programmed to be continuous or gated. Since the transmitter and receiver sections of the SSI are independent, they can be programmed to be synchronous (using a common clock) or asynchronous with respect to each other.

The SSI requires up to six pins, depending on its operating mode. The most common minimum configuration is three pins: transmit data (STD), receive data (SRD) and clock (SCK).

The SSI consists of independent transmitter and receiver sections and a common SSI clock generator. Three to six pins are required for operation, depending on the operating mode selected.

The following is a short list of SSI features:

- Three-Pin Interface:
  TXD – Transmit Data
  RXD – Receive Data
  SCLK – Serial Clock
- A 10 Mbps at 40 MHz (f_{osc}/4) serial interface
- Double Buffered

- User Programmable
- Separate Transmit and Receive Sections
- Control and Status Bits
- Interface to a Variety of Serial Devices, Including:
  - Codecs (usually without additional logic)
    - MC145502
    - MC145503
    - MC145505
    - MC145402 (13-bit linear codec)
    - MC145554 Family of Codecs
    - MC145532
  - Serial Peripherals (A/D, D/A)
    - Most Industry-Standard A/D, D/A
    - DSP56ADC16 (16-bit linear A/D)
  - DSP56K to DSP56K Networks
  - Motorola SPI Peripherals and Processors
  - Shift Registers
- Interface to Time Division Multiplexed Networks without Additional Logic
- Six Pins:
  - STD SSI Transmit Data
  - SRD SSI Receive Data
  - SCK SSI Serial Clock
  - SC0 Serial Control 0 (defined by SSI mode)
  - SC1 Serial Control 1 (defined by SSI mode)
  - SC2 Serial Control 2 (defined by SSI mode)
- On-chip Programmable Functions Include:
  - Clock – Continuous, Gated, Internal, External
  - Synchronization Signals – Bit Length and Word Length
  - TX/RX Timing – Synchronous, Asynchronous
  - Operating Modes – Normal, Network, On-Demand
  - Word Length – 8, 12, 16, 24 Bits
  - Serial Clock and Frame Sync Generator
- Four Interrupt Vectors:
  - Receive
  - Receive with Exception
  - Transmit
  - Transmit with Exception

This interface is descriptively named “synchronous” because all serial transfers are synchronized to a clock. Additional synchronization signals are used to delineate the word frames. The normal mode of operation is used to transfer data at a periodic rate, but only one word per period. The network mode is similar in that it is also intended for periodic transfers; however, it will support up to 32 words (time slots) per period. This mode can be used to build time division multiplexed (TDM) networks. In contrast, the on-demand mode is intended for nonperiodic transfers of data. This mode can be used to transfer data serially at high speed when the data becomes available. This mode offers a subset of the SPI protocol.

7.3.1 SSI Data and Control Pins

The SSI has three dedicated I/O pins (see Figure 7-1), which are used for transmit data (STD), receive data (SRD), and serial clock (SCK), where SCK may be used by both the transmitter and the receiver for synchronous data transfers or by the transmitter only for asynchronous data transfers. Three other pins may also be used, depending on the mode selected; they are serial control pins SC0, SC1, and SC2. They may be programmed as SSI control pins in the Port C control register. Table 7-1 shows the definition of SC0, SC1, SC2, and SCK in the various configurations.

Table 7-1 Definition of SC0, SC1, SC2, and SCK

SSI Pin Name (Control Bit Name)	Asynchronous (SYN=0)		Synchronous (SYN=1)	
	Continuous Clock (GCK=0)	Gated Clock (GCK=1)	Continuous Clock (GCK=0)	Gated Clock (GCK=1)

SC0=0 (in)
SC0=1 (out)
(SCD0)
	RXC External
	RXC Internal
		RXC External
		RXC Internal
			Input F0
			Output F0
				Input F0
				Output F0

SC1=0 (in)
SC1=1 (out)
(SCD1)
	FSR External
	FSR Internal
		Not Used
		FSR Internal
			Input F1
			Output F1
				Input F1
				Output F1

SC2=0 (in)
SC2=1 (out)
(SCD2)
	FST External
	FST Internal
		Not Used
		FST Internal
			FS* External
			FS* Internal
				Not Used
				FS* Internal

SCK=0 (in)
SCK=1 (out
(SCKD)
	TXC External
	TXC Internal
		TXC External
		TXC Internal)
			*XC External
			*XC Internal
				*XC External
				*XC Internal

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)
F0 – Flag 0
F1 – Flag 1

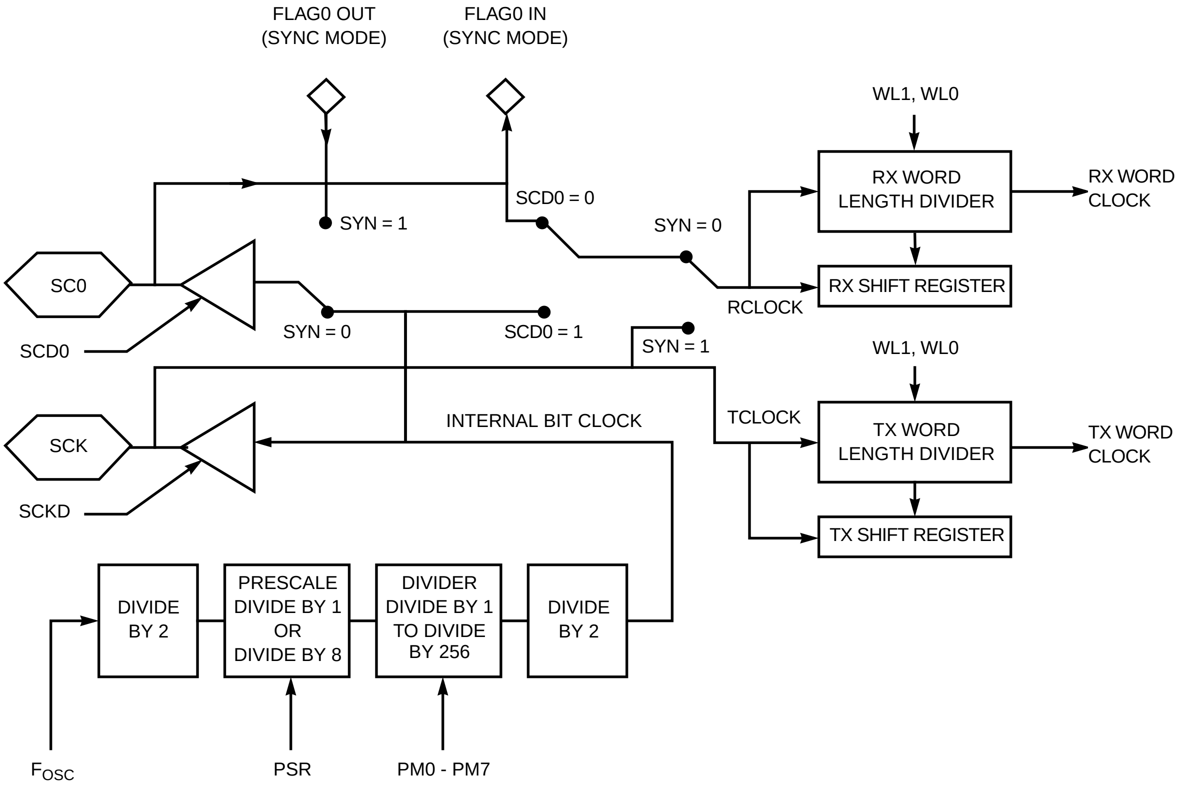


Figure 7-8 SSI Clock Generator Functional Block Diagram

The following paragraphs describe the uses of these pins for each of the SSI operating modes. Figure 7-8 and Figure 7-9 show the internal clock path connections in block diagram form. The receiver and transmitter clocks can be internal or external depending on the SYN, SCD0, and SCKD bits in CRB.

7.3.1.1 Serial Transmit Data Pin (STD)

STD is used for transmitting data from the serial transmit shift register. STD is an output when data is being transmitted. Data changes on the positive edge of the bit clock. STD goes to high impedance on the negative edge of the bit clock of the last data bit of the word (i.e., during the second half of the last data bit period) with external gated clock, regardless of the mode. With an internally generated bit clock, the STD pin becomes high impedance after the last data bit has been transmitted for a full clock period, assuming another data word does not follow immediately. If a data word follows immediately, there will not be a high-impedance interval.

Codecs label the MSB as bit 0; whereas, the DSP labels the LSB as bit 0. Therefore, when using a standard codec, the DSP MSB (or codec bit 0) is shifted out first when SHFD=0, and the DSP LSB (or codec bit 7) is shifted out first when SHFD=1. STD may be programmed as a general-purpose pin called PC8 when the SSI STD function is not being used.

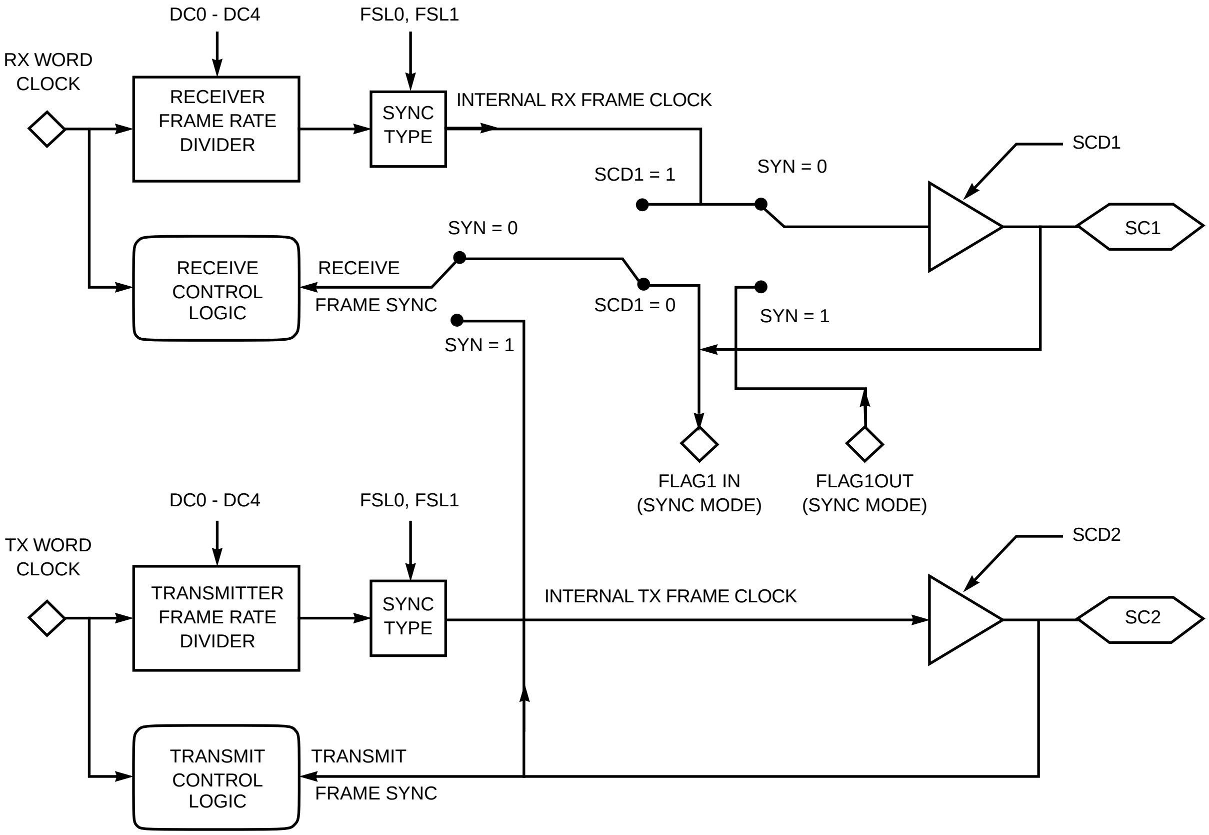


Figure 7-9 SSI Frame Sync Generator Functional Block Diagram

Table 7-2 SSI Clock Sources, Inputs, and Outputs

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

EXT – External Pin Name
INT – Internal Bit Clock

7.3.1.2 Serial Receive Data Pin (SRD)

SRD receives serial data and transfers the data to the SSI receive shift register. SRD may be programmed as a general-purpose I/O pin called PC7 when the SSI SRD function is not being used. Data is sampled on the negative edge of the bit clock.

7.3.1.3 Serial Clock (SCK)

SCK is a bidirectional pin providing the serial bit rate clock for the SSI interface. The SCK is a clock input or output used by both the transmitter and receiver in synchronous modes or by the transmitter in asynchronous modes (see Table 7-2).

Note: 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 8T (i.e., the system clock frequency must be at least four times the external SSI clock frequency). The SSI needs at least four DSP phases (DSP phase=T) inside each half of the serial clock.

7.3.1.4 Serial Control Pin (SC0)

The function of this pin is determined solely on the selection of either synchronous or asynchronous mode (see Table 7-1 and Table 7-2). For asynchronous mode, this pin will be used for the receive clock I/O. For synchronous mode, this pin is used for serial flag I/O. A typical application of flag I/O would be multiple device selection for addressing in codec systems. The direction of this pin is determined by the SCD0 bit in the CRB as described in Table 7-3. When configured as an output, this pin will be either serial output flag 0, based on control bit OF0 in CRB, or a receive shift register clock output. When configured as an input, this pin may be used either as serial input flag 0, which will control status bit IF0 in the SSISR, or as a receive shift register clock input.

Table 7-3 SSI Operation: Flag 0 and Rx Clock

SYN	GCK	SCD0	Operation
Synchronous	Continuous	Input	Flag 0 Input
Synchronous	Continuous	Output	Flag 0 Output
Synchronous	Gated	Input	Flag 0 Input
Synchronous	Gated	Output	Flag 0 Output
Asynchronous	Continuous	Input	Rx Clock – External
Asynchronous	Continuous	Output	Rx Clock – Internal
Asynchronous	Gated	Input	Rx Clock – External
Asynchronous	Gated	Output	Rx Clock – Internal

7.3.1.5 Serial Control Pin (SC1)

The function of this pin is determined solely on the selection of either synchronous or asynchronous mode (see Table 7-1 and Table 7-4). In asynchronous mode (such as a single codec with asynchronous transmit and receive), this pin is the receiver frame sync I/O. For synchronous mode with continuous clock, this pin is serial flag SC1 and operates like the previously described SC0. SC0 and SC1 are independent serial I/O flags but may be used together for multiple serial device selection. SC0 and SC1 can be used unencoded to select up to two codecs or may be decoded externally to select up to four codecs. The direction of this pin is determined by the SCD1 bit in the CRB. When configured as an output, this pin will be either a serial output flag, based on control bit OF1, or it will make the receive frame sync signal available. When configured as an input, this pin may be used as a serial input flag, which will control status bit IF1 in the SSI status register, or as a receive frame sync from an external source for continuous clock mode. In the gated clock mode, external frame sync signals are not used.

Table 7-4 SSI Operation: Flag 1 and Rx Frame Sync

SYN	GCK	SCD1	Operation
Synchronous	Continuous	Input	Flag 1 Input
Synchronous	Continuous	Output	Flag 1 Output
Synchronous	Gated	Input	Flag 1 Input
Synchronous	Gated	Output	Flag 1 Output
Asynchronous	Continuous	Input	RX Frame Sync – External
Asynchronous	Continuous	Output	RX Frame Sync – Internal
Asynchronous	Gated	Input	–
Asynchronous	Gated	Output	RX Frame Sync – Internal

7.3.1.6 Serial Control Pin (SC2)

This pin is used for frame sync I/O (see Table 7-1 and Table 7-5). SC2 is the frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode. The direction of this pin is determined by the SCD2 bit in CRB. When configured as an output, this pin is the internally generated frame sync signal. When configured as an input, this pin receives an external frame sync signal for the transmitter (and the receiver in synchronous operation). In the gated clock mode, external frame sync signals are not used.

Table 7-5 SSI Operation: Tx and Rx Frame Sync

SYN	GCK	SCD2	Operation
Synchronous	Continuous	Input	TX and RX Frame Sync
Synchronous	Continuous	Output	TX and RX Frame Sync
Synchronous	Gated	Input	–
Synchronous	Gated	Output	TX and RX Frame Sync
Asynchronous	Continuous	Input	TX Frame Sync – External
Asynchronous	Continuous	Output	TX Frame Sync – Internal
Asynchronous	Gated	Input	–
Asynchronous	Gated	Output	TX Frame Sync – Internal

7.3.2 SSI Programming Model

The SSI can be viewed as two control registers, one status register, a transmit register, a receive register, and special-purpose time slot register. These registers are illustrated in Figure 7-10 and Figure 7-11. The following paragraphs give detailed descriptions and operations of each of the bits in the SSI registers. The SSI registers are not prefaced with an "S" (for serial) as are the SCI registers.

7.3.2.1 SSI Control Register A (CRA)

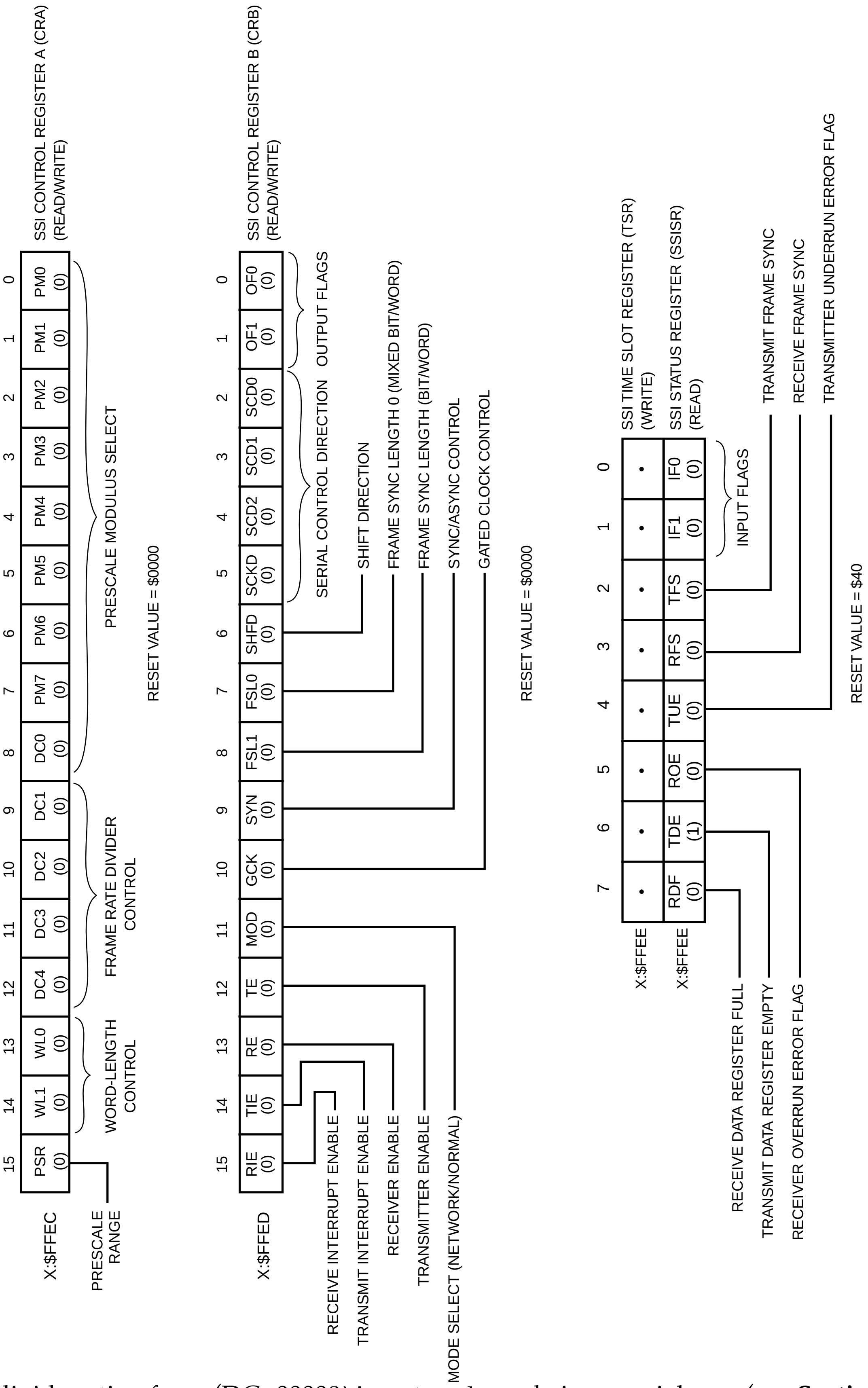
CRA is one of two 16-bit read/write control registers used to direct the operation of the SSI. The CRA controls the SSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The high-order bits of CRA are read as zeros by the DSP CPU. The CRA control bits are described in the following paragraphs.

7.3.2.1.1 CRA Prescale Modulus Select (PM7–PM0) Bits 0–7

The PM0–PM7 bits specify the divide ratio of the prescale divider in the SSI clock generator. A divide ratio from 1 to 256 (PM=0 to $FF) may be selected. The bit clock output is available at the transmit clock (SCK) and/or the receive clock (SC0) pins 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. Careful choice of the crystal oscillator frequency and the prescaler modulus will allow the industry-standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz to be generated. Hardware and software reset clear PM0–PM7.

7.3.2.1.2 CRA Frame Rate Divider Control (DC4–DC0) Bits 8–12

The DC4–DC0 bits control the divide ratio for the programmable frame rate dividers used to generate the frame clocks (see Figure 7-9). In network mode, this ratio may be interpreted as the number of words per frame minus one. In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (DC=00000 to 11111) for normal mode and 2 to 32 (DC=00001 to 11111) for network mode.



SSI CONTROL REGISTER A (CRA) (READ/WRITE) X:$FFEC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR (0)	WL1 (0)	WL0 (0)	DC4 (0)	DC3 (0)	DC2 (0)	DC1 (0)	DC0 (0)	PM7 (0)	PM6 (0)	PM5 (0)	PM4 (0)	PM3 (0)	PM2 (0)	PM1 (0)	PM0 (0)

SSI CONTROL REGISTER B (CRB) (READ/WRITE) X:$FFED

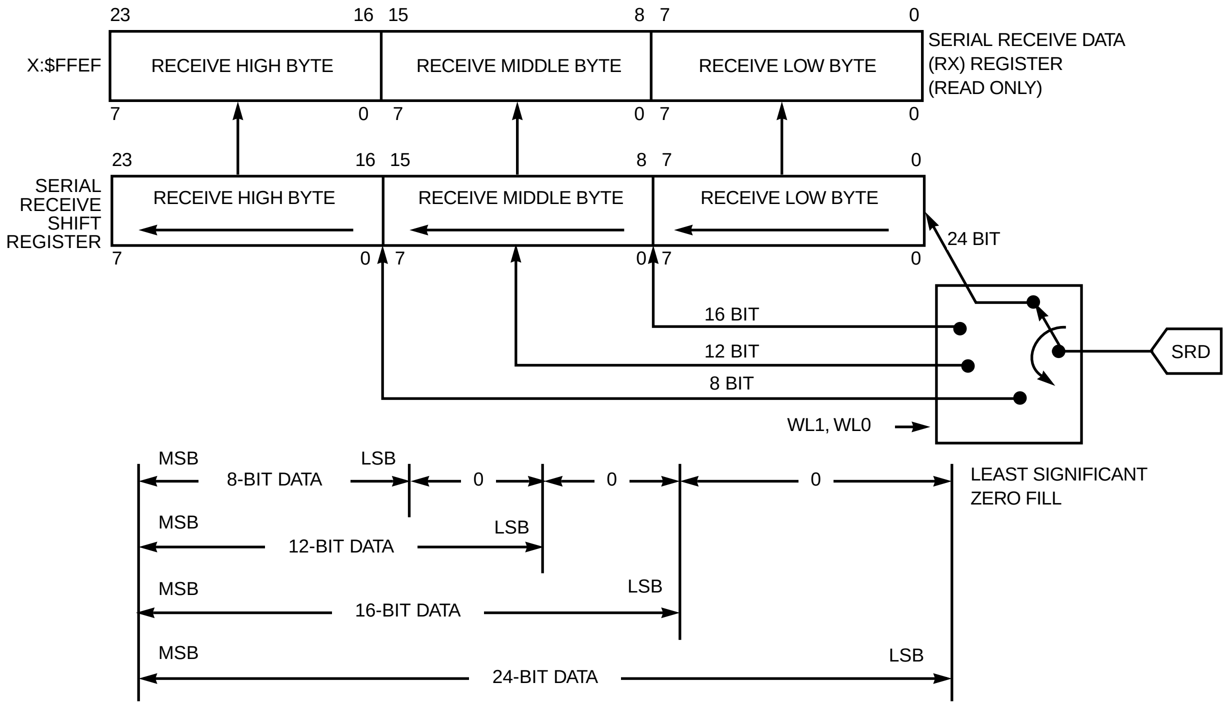
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE (0)	TIE (0)	RE (0)	TE (0)	MOD (0)	GCK (0)	SYN (0)	FSL1 (0)	FSL0 (0)	SHFD (0)	SCKD (0)	SCD2 (0)	SCD1 (0)	SCD0 (0)	OF1 (0)	OF0 (0)

SSI TIME SLOT REGISTER (TSR) (WRITE) X:$FFEE; SSI STATUS REGISTER (SSISR) (READ) X:$FFEE

	7	6	5	4	3	2	1	0
X:$FFEE	•	•	•	•	•	•	•	•
X:$FFEE	RDF (0)	TDE (1)	ROE (0)	TUE (0)	RFS (0)	TFS (0)	IF1 (0)	IF0 (0)

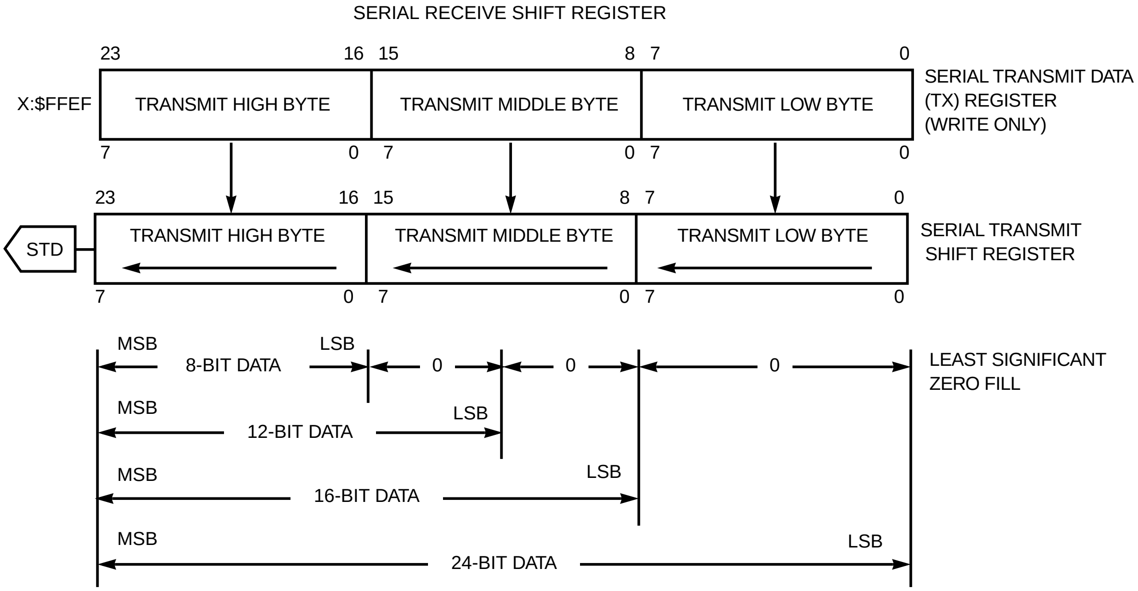
Figure 7-10 SSI Programming Model — Control and Status Registers

A divide ratio of one (DC=00000) in network mode is a special case (see **Section 7.3.7.4**).



NOTES:
1. Data is received MSB first if SHFD = 0.
2. Compatible with fractional format.

(a) Receive Registers for SHFD = 0

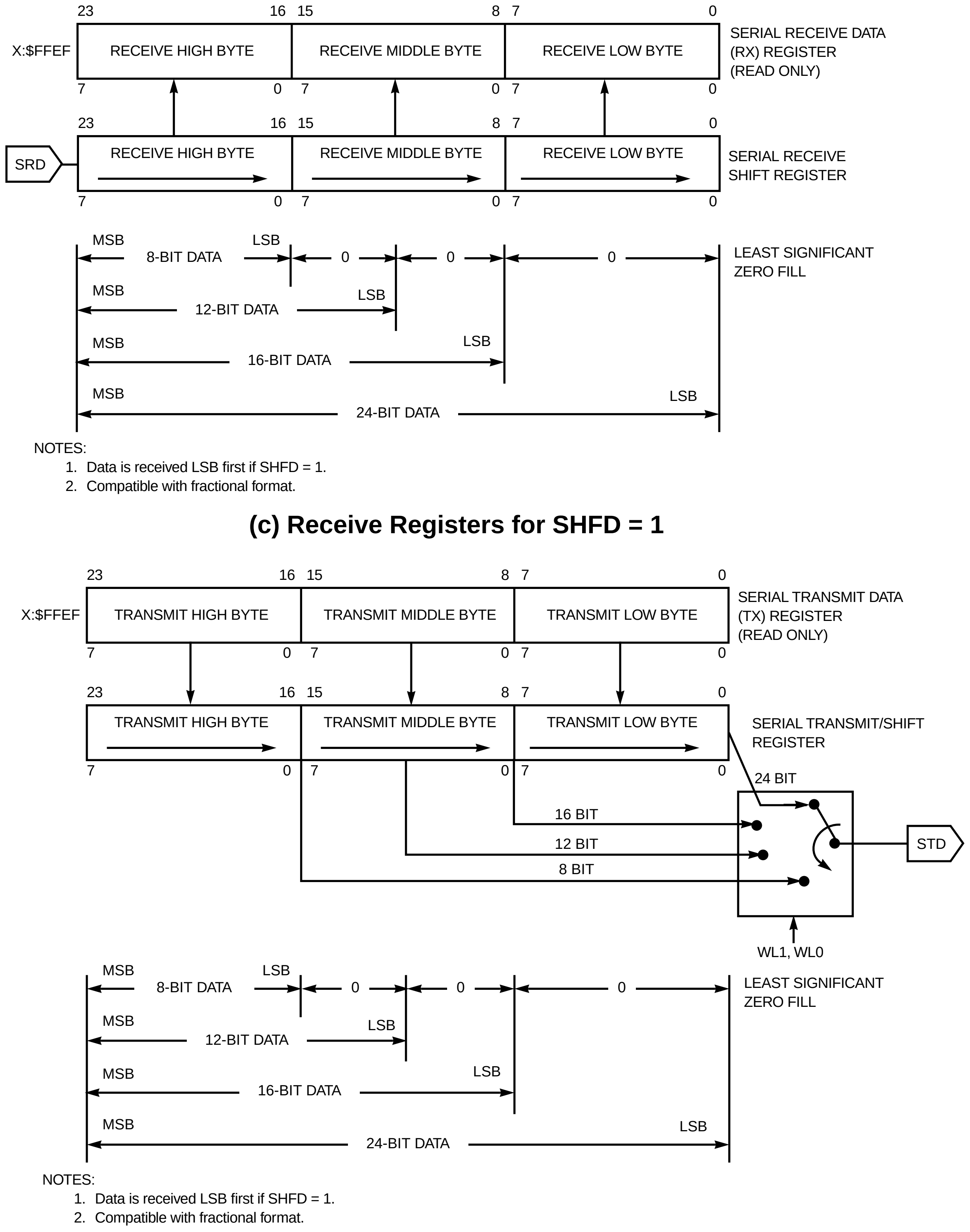


NOTES:
1. Data is sent MSB first if SHFD = 0.
2. Compatible with fractional format.

(b) Transmit Registers for SHFD = 0

Figure 7-11 SSI Programming Model (Sheet 1 of 2)

In normal mode, a divide ratio of one (DC=00000) provides continuous periodic data



NOTES:
1. Data is received LSB first if SHFD = 1.
2. Compatible with fractional format.

(c) Receive Registers for SHFD = 1

NOTES:
1. Data is received LSB first if SHFD = 1.
2. Compatible with fractional format.

(d) Transmit Registers for SHFD = 1

Figure 7-11 SSI Programming Model (Sheet 2 of 2)

word transfers. A bit-length sync (FSL1=1, FSL0=0) must be used in this case. Hardware and software reset clear DC4–DC0.

7.3.2.1.3 CRA Word Length Control (WL0, WL1) Bits 13 and 14

The WL1 and WL0 bits are used to select the length of the data words being transferred via the SSI. Word lengths of 8, 12, 16, or 24 bits may be selected according to Table 7-6.

Table 7-6 Number of Bits/Word

WL1	WL0	Number of Bits/Word
0	0	8
0	1	12
1	0	16
1	1	24

These bits control the number of active clock transitions in the gated clock modes and control the word length divider (see Figure 7-8 and Figure 7-9), which is part of the frame rate signal generator for continuous clock modes. The WL control bits also control the frame sync pulse length when FSL0 and FSL1 select a WL bit clock (see Figure 7-8). Hardware and software reset clear WL0 and WL1.

7.3.2.1.4 CRA Prescaler Range (PSR) Bit 15

The PSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired (see Figure 7-8). When PSR is cleared, the fixed prescaler is bypassed. When PSR is set, the fixed divide-by-eight prescaler is operational. This allows a 128-kHz master clock to be generated for MC14550x series codecs.

The maximum internally generated bit clock frequency is fosc/4, the minimum internally generated bit clock frequency is fosc/4/8/256=fosc/8192. Hardware and software reset clear PSR.

7.3.2.2 SSI Control Register B (CRB)

The CRB is one of two 16-bit read/write control registers used to direct the operation of the SSI. CRB controls the SSI multifunction pins, SC2, SC1, and SC0, which can be used as clock inputs or outputs, frame synchronization pins, or serial I/O flag pins. The serial output flag control bits and the direction control bits for the serial control pins are in the SSI CRB. Interrupt enable bits for each data register interrupt are provided in this control register. When read by the DSP, CRB appears on the two low-order bytes of the 24-bit word, and the high-order byte reads as zeros. Operating modes are also selected in this register. Hardware and software reset clear all the bits in the CRB. The relationships between the SSI pins (SC0, SC1, SC2, and SCK) and some of the CRB bits are summarized in Tables Table 7-1, Table 7-9, and Table 7-8. The SSI CRB bits are described in the following paragraphs.

7.3.2.2.1 CRB Serial Output Flag 0 (OF0) Bit 0

When the SSI is in the synchronous clock mode and the serial control direction zero bit (SCD0) is set, indicating that the SC0 pin is an output, then data present in OF0 will be written to SC0 at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode. Hardware and software reset clear OF0.

7.3.2.2.2 CRB Serial Output Flag 1 (OF1) Bit 1

When the SSI is in the synchronous clock mode and the serial control direction one (SCD1) bit is set, indicating that the SC1 pin is an output, then data present in OF1 will be written to the SC1 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode (see Section **7.3.7**).

The normal sequence for setting output flags when transmitting data is to poll TDE (TX empty), to first write the flags, and then write the transmit data to the TX register. OF0 and OF1 are double buffered so that the flag states appear on the pins when the TX data is transferred to the transmit shift register (i.e., the flags are synchronous with the data). Hardware and software reset clear OF1.

Note: The optional serial output pins (SC0, SC1, and SC2) are controlled by the frame timing and are not affected by TE or RE.

7.3.2.2.3 CRB Serial Control 0 Direction (SCD0) Bit 2

SCD0 controls the direction of the SC0 I/O line. When SCD0 is cleared, SC0 is an input; when SCD0 is set, SC0 is an output (see Tables Table 7-1 and Table 7-2, and Figure 7-12). Hardware and software reset clear SCD0.

7.3.2.2.4 CRB Serial Control 1 Direction (SCD1) Bit 3

SCD1 controls the direction of the SC1 I/O line. When SCD1 is cleared, SC1 is an input; when SCD1 is set, SC1 is an output (see Tables Table 7-1 and Table 7-2 and Figure 7-12). Hardware and software reset clear SCD1.

7.3.2.2.5 CRB Serial Control 2 Direction (SCD2) Bit 4

SCD2 controls the direction of the SC2 I/O line. When SCD2 is cleared, SC2 is an input; when SCD2 is set, SC2 is an output (see Tables Table 7-1 and Table 7-2, and Figure 7-12). Hardware and software reset clear SCD2.

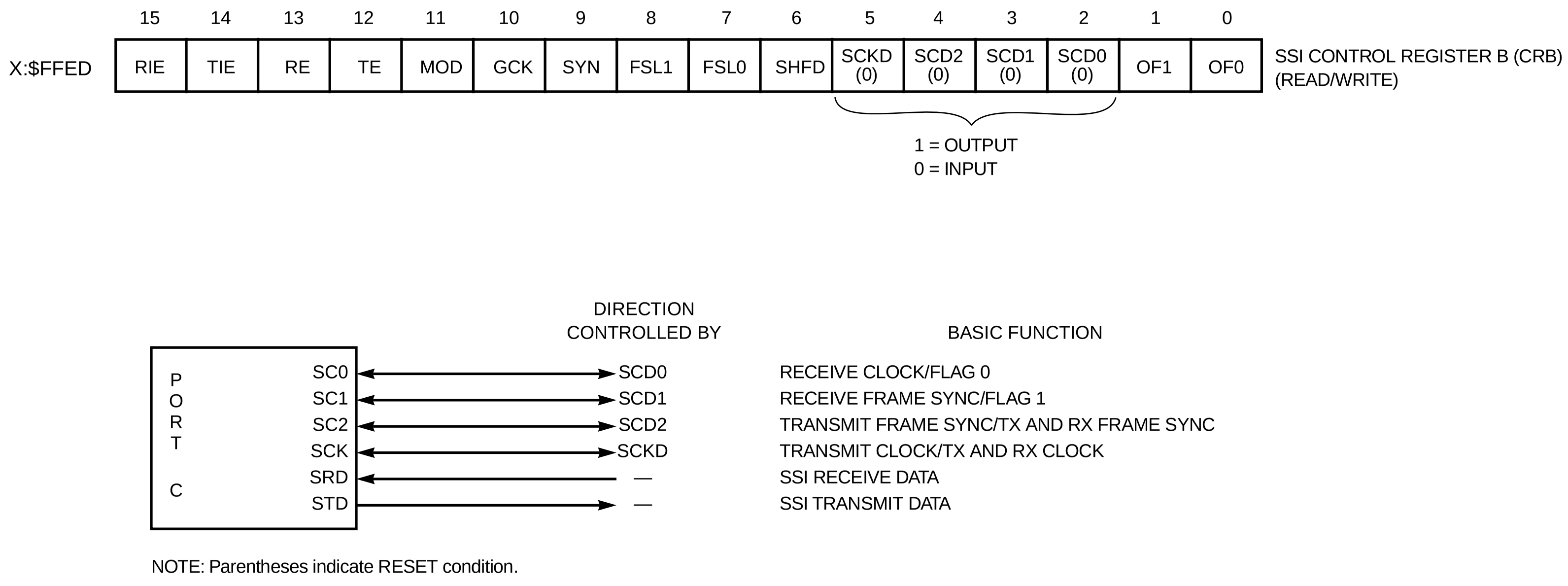


Figure 7-12 Serial Control, Direction Bits

7.3.2.2.6 CRB Clock Source Direction (SCKD) Bit 5

SCKD selects the source of the clock signal used to clock the transmit shift register in the asynchronous mode and both the transmit shift register and the receive shift register in the synchronous mode. When SCKD is set, the internal clock source becomes the bit clock for the transmit shift register and word length divider and is the output on the SCK pin. When SCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCK pin, and an external clock source may drive this pin. Hardware and software reset clear SCKD.

7.3.2.2.7 CRB Shift Direction (SHFD) Bit 6

This bit causes the transmit shift register to shift data out MSB first when SHFD equals zero or LSB first when SHFD equals one. Receive data is shifted in MSB first when SHFD equals zero or LSB first when SHFD equals one. Hardware reset and software reset clear SHFD.

7.3.2.2.8 CRB Frame Sync Length (FSL0 and FSL1) Bits 7 and 8

These bits select the type of frame sync to be generated or recognized (see Table 7-7). If FSL1 equals zero and FSL0 equals zero, a word-length frame sync is selected for both TX and RX that is the length of the data word defined by bits WL1 and WL0. If FSL1 equals one and FSL0 equals zero, a 1-bit clock period frame sync is selected for both TX and RX. When FSL0 equals one, the TX and RX frame syncs are different lengths. Hardware reset and software reset clear FSL0 and FSL1.

Table 7-7 Frame Sync Length

FSL1	FSL0	Frame Sync Length
0	0	WL bit clock for both TX/RX
0	1	One-bit clock for TX and WL bit clock for RX
1	0	One-bit clock for both TX/RX
1	1	One-bit clock for RX and WL bit clock for TX

7.3.2.2.9 CRB Sync/Async (SYN) Bit 9

SYN controls whether the receive and transmit functions of the SSI occur synchronously or asynchronously with respect to each other. When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. Hardware reset and software reset clear SYN.

7.3.2.2.10 CRB Gated Clock Control (GCK) Bit 10

GCK is used to select between a continuously running data clock or a clock that runs only when there is data to be sent in the transmit shift register. When GCK is cleared, a continuous clock is selected; when GCK is set, the clock will be gated. Hardware reset and software reset clear GCK.

Note: For gated clock mode with externally generated bit clock, internally generated frame sync is not defined.

7.3.2.2.11 CRB SSI Mode Select (MOD) Bit 11

MOD selects the operational mode of the SSI. When MOD is cleared, the normal mode is selected; when MOD is set, the network mode is selected. In the 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 is (possibly) transferred every time slot. For more details, see Section **7.3.3**. Hardware and software reset clear MOD.

7.3.2.2.12 CRB SSI Transmit Enable (TE) Bit 12

TE enables the transfer of data from TX to the transmit shift register. When TE is set and a frame sync is detected, the transmit portion of the SSI is enabled for that frame. When TE is cleared, the transmitter will be disabled after completing transmission of data currently in the SSI transmit shift register. The serial output is three-stated, and any data present in TX will not be transmitted (i.e., data can be written to TX with TE cleared; TDE will be cleared, but data will not be transferred to the transmit shift register).

The normal mode transmit enable sequence is to write data to TX or TSR before setting TE. The normal transmit disable sequence is to clear TE and TIE after TDE equals one.

In the network mode, the operation of clearing TE and setting it again will disable the transmitter after completing transmission of the current data word until the beginning of the next frame. During that time period, the STD pin will remain in the high-impedance state. Hardware reset and software reset clear TE.

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

Note: TE does not inhibit TDE or transmitter interrupts. TE does not affect the generation of frame sync or output flags.

7.3.2.2.13 CRB SSI Receive Enable (RE) Bit 13

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

RE must be set in the normal mode and on-demand mode to receive data. In network mode, the operation of clearing RE and setting it again will disable the receiver after reception of the current data word until the beginning of the next data frame. Hardware and software reset clear RE.

Note: RE does not inhibit RDF or receiver interrupts. RE does not affect the generation of a frame sync.

7.3.2.2.14 CRB SSI Transmit Interrupt Enable (TIE) Bit 14

The DSP will be interrupted when TIE and the TDE flag in the SSI status register is set. (In network mode, the interrupt takes effect in the next frame synch, not in the next time slot.) When TIE is cleared, this interrupt is disabled. However, the TDE bit will always indicate the transmit data register empty condition even when the transmitter is disabled with the TE bit. Writing data to TX or TSR will clear TDE, thus clearing the interrupt. Hardware and software reset clear RE.

There are two transmit data interrupts that have separate interrupt vectors:

1. Transmit data with exceptions – This interrupt is generated on the following condition:
   TIE=1, TDE=1, and TUE=1
2. Transmit data without exceptions – This interrupt is generated on the following condition:
   TIE=1, TDE=1, and TUE=0

See **SECTION 7 — *PROCESSING STATES*** in the *DSP56000 Family Manual* for more information on exceptions.

7.3.2.2.15 CRB SSI Receive Interrupt Enable (RIE) Bit 15

When RIE is set, the DSP will be interrupted when RDF in the SSI status register is set. (In network mode, the interrupt takes effect in the next frame synch, not in the next time slot.) When RIE is cleared, this interrupt is disabled. However, the RDF bit still indicates the receive data register full condition. Reading the receive data register will clear RDF, thus clearing the pending interrupt. Hardware and software reset clear RIE.

There are two receive data interrupts that have separate interrupt vectors:

1. Receive data with exceptions – This interrupt is generated on the following condition:
   RIE=1, RDF=1, and ROE=1
2. Receive data without exceptions – This interrupt is generated on the following condition:
   RIE=1, RDF=1, and ROE=0

See **SECTION 7 —** ***PROCESSING STATES*** in the *DSP56000 Family Manual* for more information on exceptions.

7.3.2.3 SSI Status Register (SSISR)

The SSISR is an 8-bit read-only status register used by the DSP to interrogate the status and serial input flags of the SSI. When the SSISR is read to the internal data bus, the register contents occupy the low-order byte of the data bus, and the high-order portion is zero filled. The status bits are described in the following paragraphs.

7.3.2.3.1 SSISR Serial Input Flag 0 (IF0) Bit 0

The SSI latches data present on the SC0 pin during reception of the first received bit after frame sync is detected. IF0 is updated with this data when the receive shift register is transferred into the receive data register. The IF0 bit is enabled only when SCD0 is cleared and SYN is set, indicating that SC0 is an input and the synchronous mode is selected (see Table 7-1); otherwise, IF0 reads as a zero when it is not enabled. Hardware, software, SSI individual, and STOP reset clear IF0.

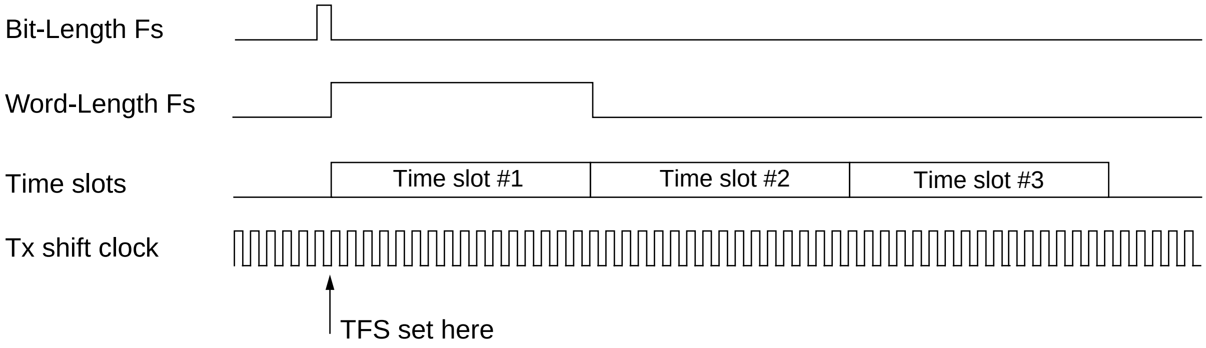
7.3.2.3.2 SSISR Serial Input Flag 1 (IF1) Bit 1

The SSI latches data present on the SC1 pin during reception of the first received bit after frame sync is detected. The IF1 flag is updated with the data when the receiver shift register is transferred into the receive data register. The IF1 bit is enabled only when SCD1 is cleared and SYN is set, indicating that SC1 is an input and the synchronous mode is selected (see Table 7-1); otherwise, IF1 reads as a zero when it is not enabled. Hardware, software, SSI individual, and STOP reset clear IF1.

7.3.2.3.3 SSISR Transmit Frame Sync Flag (TFS) Bit 2

When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. If word-wide transmit frame sync is selected (FSL0=FSL1), this indicates that the frame sync was high at least at the beginning of the time slot if external frame sync is selected, or high throughout the time slot if internal frame sync was selected.

If bit-wide transmit frame sync is selected (FSL0≠FSL1), this indicates that the frame sync (either internal or external) was high during the last Tx clock bit period prior to the current time slot, and that the frame sync falling edge corresponds to the assertion of the first output data bit, as shown below.



Data written to the transmit data register during the time slot when TFS is set will be transmitted (in network mode) during the second time slot in the frame. TFS is useful in network mode to identify the start of the frame. This is illustrated in a typical transmit interrupt handler:

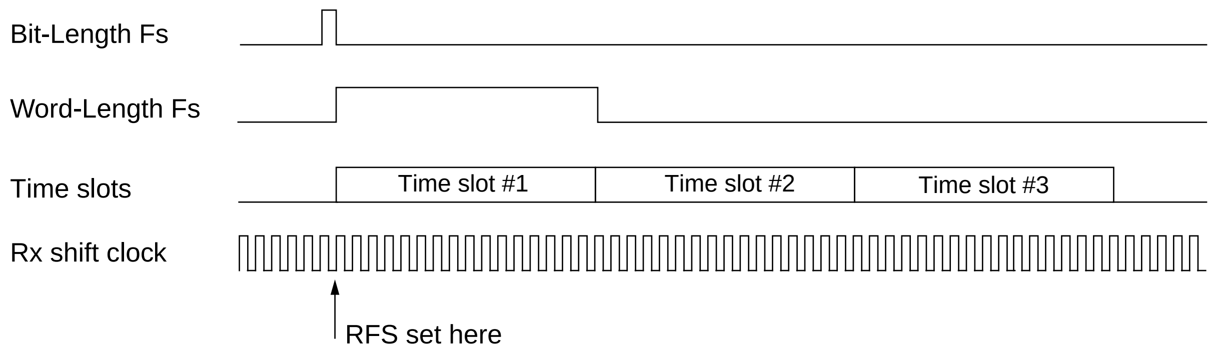
```
          MOVEP     X:(R4)+,X:SSITx
          JCLR      #2,X:SSISR,_NoTFS;1 = FIRST TIMESLOT
                    ;Do something
          JMP       _DONE
_NoTFS
                    ;Do something else
_DONE
```

Note: In normal mode, TFS will always read as a one when transmitting data because there is only one time slot per frame – the “frame sync” time slot.

TFS, which is cleared by hardware, software, SSI individual, or STOP reset, is not affected by TE.

7.3.2.3.4 SSISR Receive Frame Sync Flag (RFS) Bit 3

When set, RFS indicates that a receive frame sync occurred during reception of the word in the serial receive data register. This indicates that the data word is from the first time slot in the frame. If word-wide receive frame sync is selected (FSL1=0), this indicates that the frame sync was high at least at the beginning of the timeslot. If bit-wide receive frame sync is selected (FSL1=1), this indicates that the frame sync (either internal or external) was high during the last bit period prior to the current timeslot, and that the frame sync falling edge corresponds to the assertion of the first output data bit, as shown below.



When RFS is clear 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 useful in network mode to identify the start of the frame. This feature is illustrated in a typical receive interrupt handler:

```
          MOVEP     X:SSIRx,X:(R4)+
          JCLR      #3,X:SSISR,_NoRFS;1 = FIRST TIMESLOT
                    ;Do something
          JMP       _DONE
_NoRFS
                    ;Do something else
_DONE
```

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

RFS, which is cleared by hardware, software, SSI individual, or STOP reset, is not affected by RE.

7.3.2.3.5 SSISR Transmitter Underrun Error Flag (TUE) Bit 4

TUE is set when the serial transmit shift register is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TX) will be retransmitted.

In the normal mode, there is only one transmit time slot per frame. In the network mode, there can be up to 32 transmit time slots per frame.

TUE does not cause any interrupts; however, TUE does cause a change in the interrupt vector used for transmit interrupts so that a different interrupt handler may be used for a transmit underrun condition. If a transmit interrupt occurs with TUE set, the transmit data with exception status interrupt will be generated; if a transmit interrupt occurs with TUE clear, the transmit data without errors interrupt will be generated.

Hardware, software, SSI individual, and STOP reset clear TUE. TUE is also cleared by reading the SSISR with TUE set, followed by writing TX or TSR.

7.3.2.3.6 SSISR Receiver Overrun Error Flag (ROE) Bit 5

This flag is set when the serial receive shift register is filled and ready to transfer to the receiver data register (RX) and RX is already full (i.e., RDF=1). The receiver shift register is not transferred to RX. ROE does not cause any interrupts; however, ROE does cause a change in the interrupt vector used for receive interrupts so that a different interrupt handler may be used for a receive error condition. If a receive interrupt occurs with ROE set, the receive data with exception status interrupt will be generated; if a receive interrupt occurs with ROE clear, the receive data without errors interrupt will be generated.

Hardware, software, SSI individual, and STOP reset clear ROE. ROE is also cleared by reading the SSISR with ROE set, followed by reading the RX. Clearing RE does not affect ROE.

7.3.2.3.7 SSISR SSI Transmit Data Register Empty (TDE) Bit 6

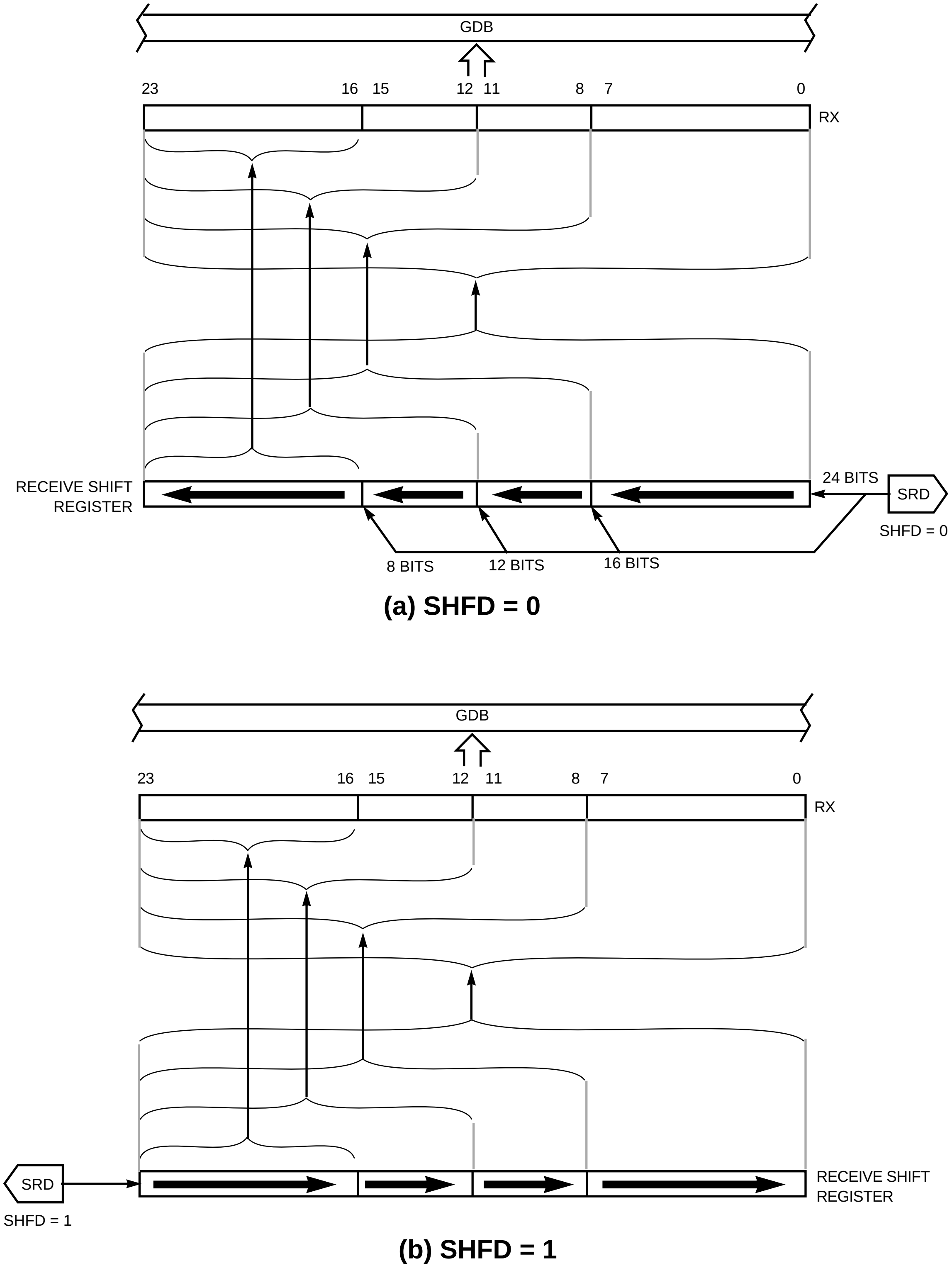
This flag is set when the contents of the transmit data register are transferred to the transmit shift register; it is also set for a disabled time slot period in network mode (as if data were being transmitted after the TSR was written). Thirdly, it can be set by the hardware, software, SSI individual, or STOP reset. When set, TDE indicates that data should be written to the TX or to the time slot register (TSR). TDE is cleared when the DSP writes to the transmit data register or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, a DSP transmit data interrupt request will be issued when TDE is set. The vector of the interrupt will depend on the state of the transmitter underrun bit.

7.3.2.3.8 SSISR SSI Receive Data Register Full (RDF) Bit 7

RDF is set when the contents of the receive shift register are transferred to the receive data register. RDF is cleared when the DSP reads the receive data register or cleared by hardware, software, SSI individual, or STOP reset. If RIE is set, a DSP receive data interrupt request will be issued when RDF is set. The vector of the interrupt request will depend on the state of the receiver overrun bit.

7.3.2.4 SSI Receive Shift Register

This 24-bit shift register receives the incoming data from the serial receive data pin. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O (or gated clock) is asserted. Data is assumed to be received MSB first if SHFD equals zero and LSB first if SHFD equals one. Data is transferred to the SSI receive data register after 8, 12, 16, or 24 bits have been shifted in, depending on the word-length control bits in the CRA (see Figure 7-13).



(a) SHFD = 0

(b) SHFD = 1

Figure 7-13 Receive Data Path

7.3.2.5 SSI Receive Data Register (RX)

RX is a 24-bit read-only register that accepts data from the receive shift register as it becomes full. The data read will occupy the most significant portion of the receive data register (see Figure 7-13). The unused bits (least significant portion) will read as zeros. The DSP is interrupted whenever RX becomes full if the associated interrupt is enabled.

7.3.2.6 SSI Transmit Shift Register

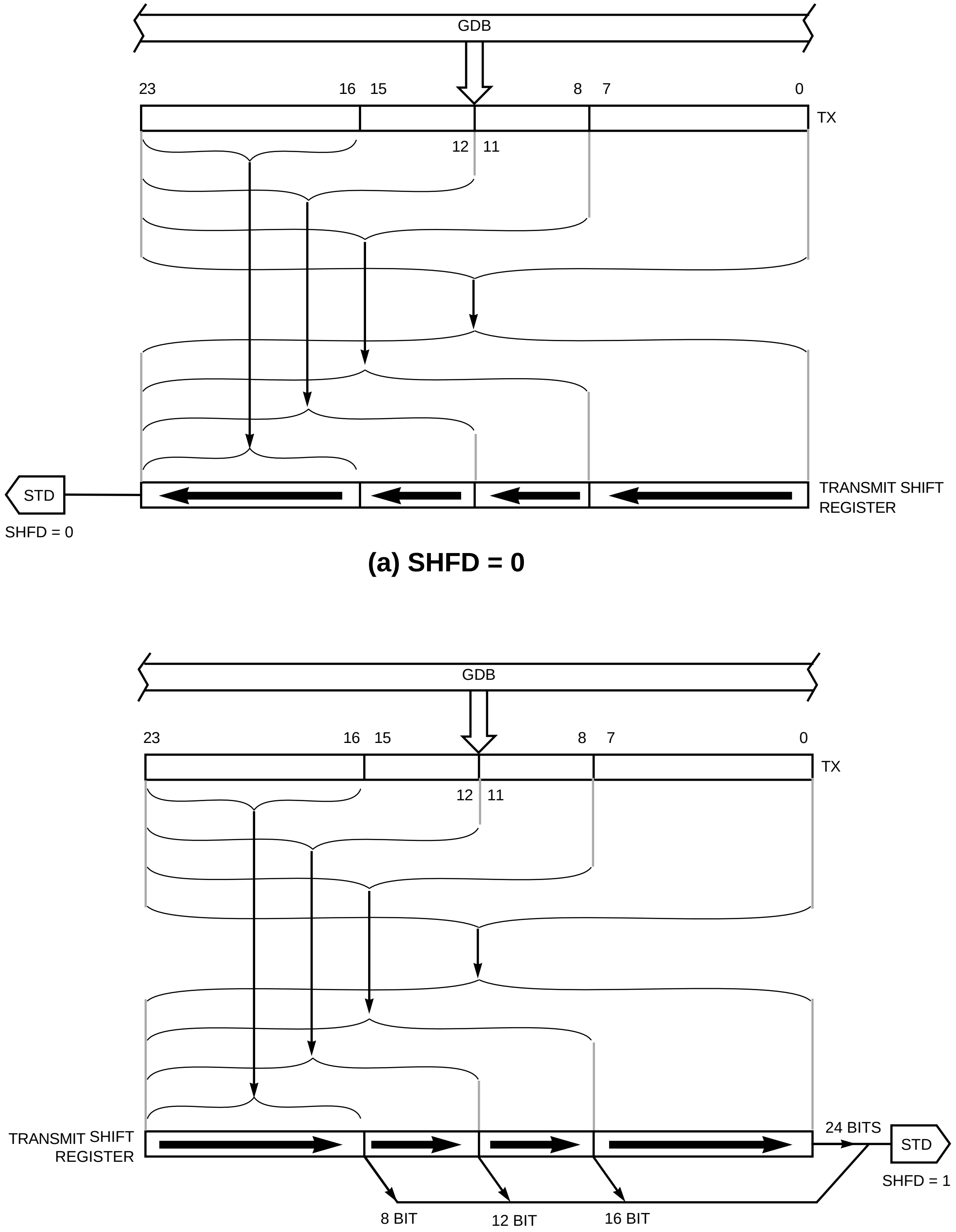
This 24-bit shift register contains the data being transmitted. Data is shifted out to the serial transmit data pin by the selected (internal/external) bit clock when the associated frame sync I/O (or gated clock) is asserted. The number of bits shifted out before the shift register is considered empty and may be written to again can be 8, 12, 16, or 24 bits (determined by the word-length control bits in CRA). The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is shifted out of this register MSB first if SHFD equals zero and LSB first if SHFD equals one (see Figure 7-14).

7.3.2.7 SSI Transmit Data Register (TX)

TX is a 24-bit write-only register. Data to be transmitted is written into this register and is automatically transferred to the transmit shift register. The data written (8, 12, 16, or 24 bits) should occupy the most significant portion of TX (see Figure 7-14). The unused bits (least significant portion) of TX are don’t care bits. The DSP is interrupted whenever TX becomes empty if the transmit data register empty interrupt has been enabled.

7.3.2.8 Time Slot Register (TSR)

TSR is effectively a null data register that is used when the data is not to be transmitted in the available transmit time slot. For the purposes of timing, TSR is a write-only register that behaves like an alternative transmit data register, except that, rather than transmitting data, the transmit data pin is in the high-impedance state for that time slot.



(a) SHFD = 0

(b) SHFD = 1

Figure 7-14 Transmit Data Path

Table 7-8 Mode and Pin Definition Table — Gated Clock

Control Bits								Mode		SC0		SC1		SC2		SCK	
MOD	GCLK	SYN	SCD2	SCD1	SCD0	SCKD	DC4-DC0	TX	RX	In	Out	In	Out	In	Out	In	Out
0	1	0	X	X	1	1	X	6	6	—	RXC	?	FSR	?	FST	—	TXC
0	1	1	X	X	X	1	X	6	6	F0	F0	F0	F1	?	FS*	—	*XC
0	1	0	X	X	1	0	X	5	6	—	RXC	?	FSR	?	?	TXC	—
0	1	0	X	X	0	0	X	5	5	RXC	—	?	?	?	?	TXC	—
0	1	1	X	X	X	0	X	5	5	F0	F0	F1	F1	?	?	*XC	—
1	1	0	X	X	1	1	0	8	7	—	RXC	?	FSR	?	FST	—	TXC
1	1	0	X	X	0	1	0	8	5	RXC	—	?	?	?	FST	—	TXC
1	1	1	X	X	X	1	0	8	9	F0	F0	F1	F1	?	FS*	—	*XC
0	1	0	X	X	0	1	X	6	5	RXC	—	?	?	?	FST	—	TXC

DC4–DC0=0 means that bits DC4=0, DC3=0, DC2=0, DC1=0, and DC0=0.
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)
F0 – Flag 0
F1 – Flag 1
? – Undefined

7.3.3 Operational Modes and Pin Definitions

Table 7-9 and Table 7-8 completely describe the SSI operational modes and pin definitions (Table 7-1 is a simplified version of these tables). The operational modes are as follows:

1. Continuous Clock
   Mode 1 – Normal with Internal Frame Sync
   Mode 2 – Network with Internal Frame Sync
   Mode 3 – Normal with External Frame Sync
   Mode 4 – Network with External Frame Sync

2. Gated Clock
   Mode 5 – External Gated Clock
   Mode 6 – Normal with Internal Gated Clock
   Mode 7 – Network with Internal Gated Clock

3. Special Case (Both Gated and Continuous Clock)
   Mode 8 – On-Demand Mode (Transmitter Only)
   Mode 9 – Receiver Follows Transmitter Clocking

Table 7-9 Mode and Pin Definition Table — Continuous Clock

Control Bits								Mode		SC0		SC1		SC2		SCK	
MOD	GCLK	SYN	SCD2	SCD1	SCD0	SCKD	DC4-DC0	TX	RX	In	Out	In	Out	In	Out	In	Out
0	0	0	1	1	X	X	X	1	1	RXC	RXC	—	FSR	—	FST	TXC	TXC
0	0	1	1	X	X	X	X	1	1	F0	F0	F1	F1	—	FS*	*XC	*XC
1	0	0	1	1	X	X	1	2	2	RXC	RXC	—	FSR	—	FST	TXC	TXC
1	0	1	1	X	X	X	1	2	2	F0	F0	F1	F1	—	FS*	*XC	*XC
0	0	0	0	1	X	X	X	3	1	RXC	RXC	—	FSR	FST	—	TXC	TXC
0	0	0	1	0	X	X	X	1	3	RXC	RXC	FSR	—	—	FST	TXC	TXC
0	0	0	0	0	X	X	X	3	3	RXC	RXC	FSR	—	FST	—	TXC	TXC
0	0	1	0	X	X	X	X	3	3	F0	F0	F1	F1	FS*	—	*XC	*XC
1	0	0	0	1	X	X	X	4	2	RXC	RXC	—	FSR	FST	—	TXC	TXC
1	0	0	1	0	X	X	1	2	4	RXC	RXC	FSR	—	—	FST	TXC	TXC
1	0	0	0	0	X	X	X	4	4	RXC	RXC	FSR	—	FST	—	TXC	TXC
1	0	1	0	X	X	X	X	4	4	F0	F0	F1	F1	FS*	—	*XC	*XC
1	0	0	1	1	X	X	0	8	2	RXC	RXC	—	FSR	—	FST	TXC	TXC
1	0	1	1	X	X	X	0	8	9	F0	F0	F1	F1	—	FS*	*XC	*XC
1	0	0	1	0	X	X	0	8	4	RXC	RXC	FSR	—	—	FST	TXC	TXC

DC4-DC0 = 0 means that bits DC4 = 0, DC3 = 0, DC2 = 0, DC1 = 0, and DC0 = 0
DC4-DC0 = 1 means that bits DC4-DC0≠0
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)
F0 — Flag 0
F1 — Flag 1

7.3.4 Registers After Reset

Hardware or software reset clears the port control register bits, which configure all I/O as general-purpose input. The SSI will remain in reset while all SSI pins are programmed as general-purpose I/O (CC8–CC3=0) and will become active only when at least one of the SSI I/O pins is programmed as not general-purpose I/O. Table 7-10 shows how each type of reset affects each SSI register bit.

Table 7-10 SSI Registers After Reset

Register Name	Register Data	Bit Number	Reset			
			HW Reset	SW Reset	Individual Reset	ST Reset
CRA	PSR	15	0	0	–	–
	WL(2–0)	13,14	0	0	–	–
	DC(4–0)	8–12	0	0	–	–
	PM(7–0)	0–7	0	0	–	–
CRB	RIE	15	0	0	–	–
	TIE	14	0	0	–	–
	RE	13	0	0	–	–
	TE	12	0	0	–	–
	MOD	11	0	0	–	–
	GCK	10	0	0	–	–
	SYN	9	0	0	–	–
	FSL1	8	0	0	–	–
	FSL0	7	0	0	–	–
	SHFD	6	0	0	–	–
	SCKD	5	0	0	–	–
	SCD(2–0)	2–4	0	0	–	–
	OF(1–0)	0,1	0	0	–	–
SSISR	RDF	7	0	0	0	0
	TDE	6	1	1	1	1
	ROE	5	0	0	0	0
	TUE	4	0	0	0	0
	RFS	3	0	0	0	0
	TFS	2	0	0	0	0
	IF(1–0)	0,1	0	0	0	0
RDR	RDR (23–0)	23–0	–	–	–	–
TDR	TDR (23–0)	23–0	–	–	–	–
RSR	RDR (23–0)	23–0	–	–	–	–
TSR	RDR (23–0)	23–0	–	–	–	–

NOTES:

1. RSR – SSI receive shift register
2. TSR – SSI transmit shift register
3. HW – Hardware reset is caused by asserting the external pin $\overline{\text{RESET}}$.
4. SW – Software reset is caused by executing the RESET instruction.
5. IR – Individual reset is caused by SSI peripheral pins (i.e., PCC(3–8)) being configured as general-purpose I/O.
6. ST – Stop reset is caused by executing the STOP instruction.

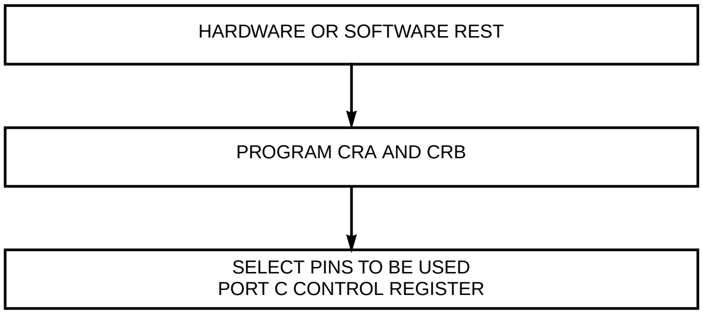


Figure 7-15 SSI Initialization Block Diagram

7.3.5 SSI Initialization

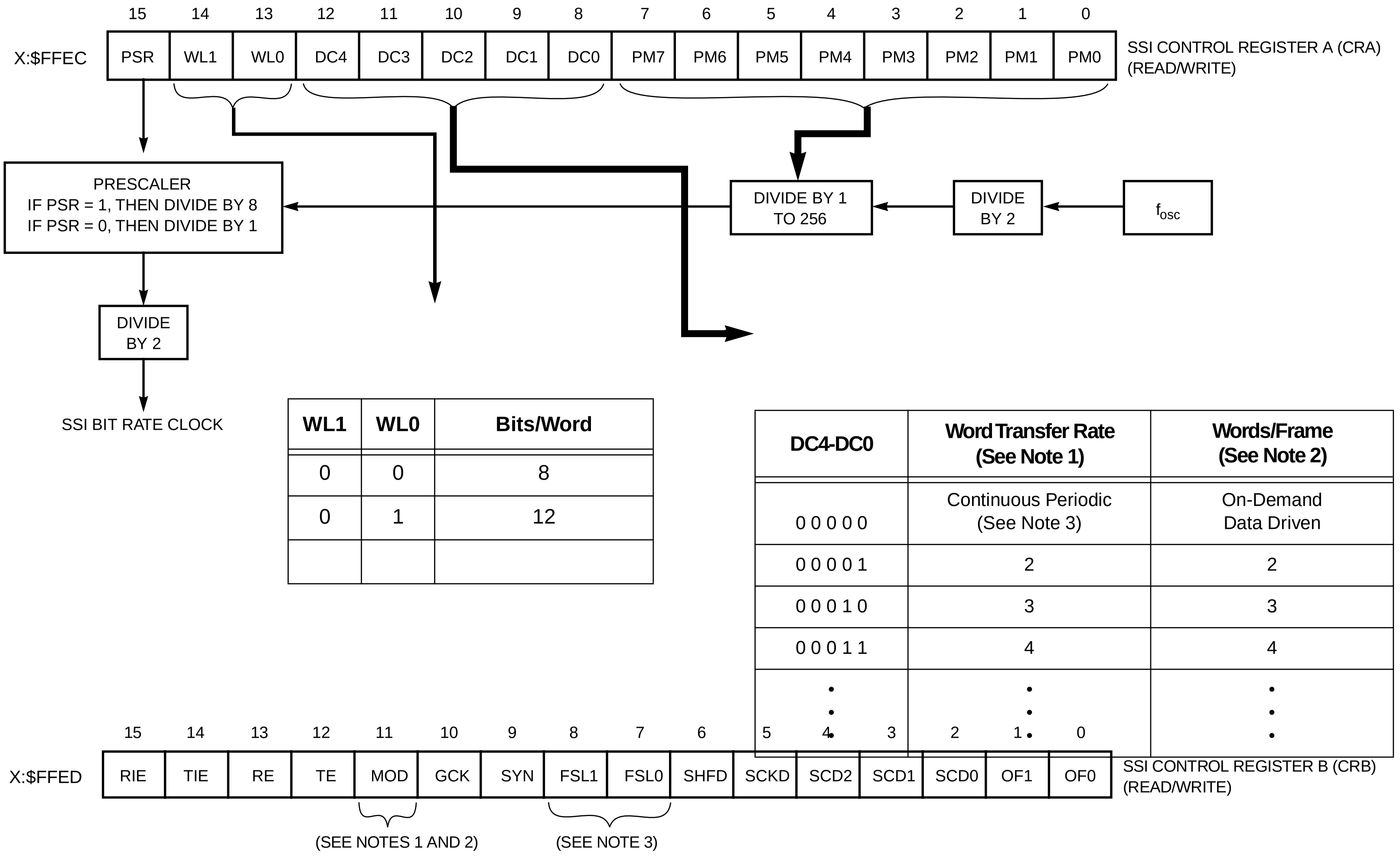
The correct way to initialize the SSI is as follows:

1. Hardware, software, SSI individual, or STOP reset
2. Program SSI control registers
3. Configure SSI pins (at least one) as not general-purpose I/O

During program execution, CC8–CC3 may be cleared, causing the SSI to stop serial activity and enter the individual reset state. All status bits of the interface will be set to their reset state; however, the contents of CRA and CRB are not affected. This procedure allows the DSP program to reset each interface separately from the other internal peripherals.

The DSP program must use an SSI reset when changing the MOD, GCK, SYN, SCKD, SCD2, SCD1, or SCD0 bits to ensure proper operation of the interface. Figure 7-15 is a flowchart illustrating the three initialization steps previously listed. Figure 7-16, Figure 7-17, and Figure 7-18 provide additional detail to the flowchart.

Figure 7-18 shows the six control bits in the PCC, which select the six SSI pins as either general-purpose I/O or as SSI pins. The STD pin can only transmit data; the SRD pin can only receive data. The other four pins can be inputs or outputs, depending on how they are programmed. This programming is accomplished by setting bits in CRA and CRB as shown in Figure 7-12. The CRA (see Figure 7-16) sets the SSI bit rate clock with PSR and PM0–PM7, sets the word length with WL1 and WL0, and sets the number of words in a frame with DC0–DC4. There is a special case where DC4–DC0 equals zero (one word per frame). Depending on whether the normal or network mode is selected (MOD=0 or MOD=1, respectively), either the continuous periodic data mode is selected, or the on-demand data driven mode is selected. The continuous periodic mode requires that FSL1 equals one and FSL0 equals zero. Figure 7-17 shows the meaning of each individual bit in the CRB. These bits should be set according to the application requirements.



WL1	WL0	Bits/Word
0	0	8
0	1	12

DC4-DC0	Word Transfer Rate (See Note 1)	Words/Frame (See Note 2)
0 0 0 0 0	Continuous Periodic (See Note 3)	On-Demand Data Driven
0 0 0 0 1	2	2
0 0 0 1 0	3	3
0 0 0 1 1	4	4
• • •	• • •	• • •

NOTES:
1. NORMAL — MOD = 0
2. NETWORK — MOD = 1
3. FSL1 = 1, FSL0 = 0

Figure 7-16 SSI CRA Initialization Procedure

Table 7-11 (a) and Table 7-11 (b) provide a convenient listing of PSR and PM0–PM7 settings

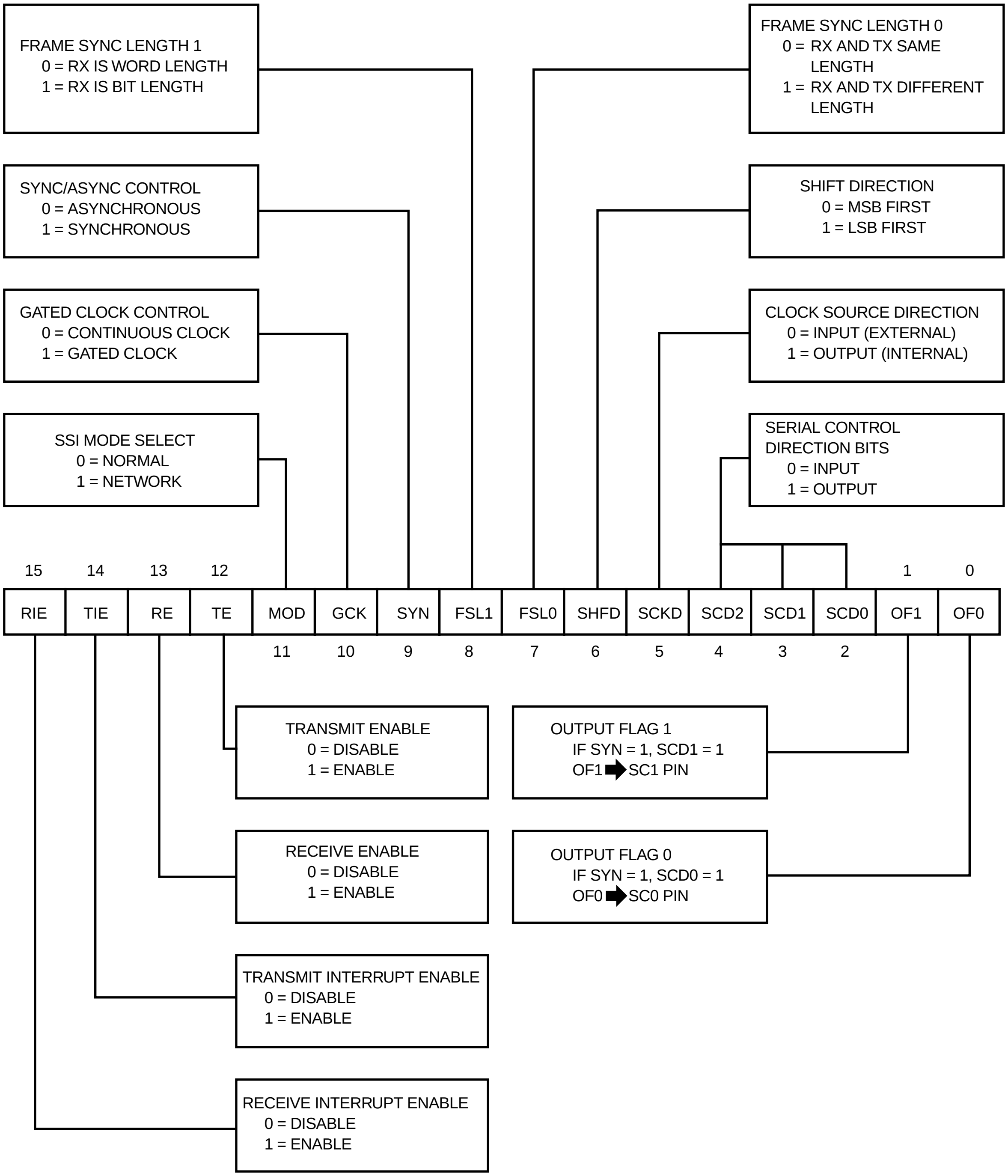
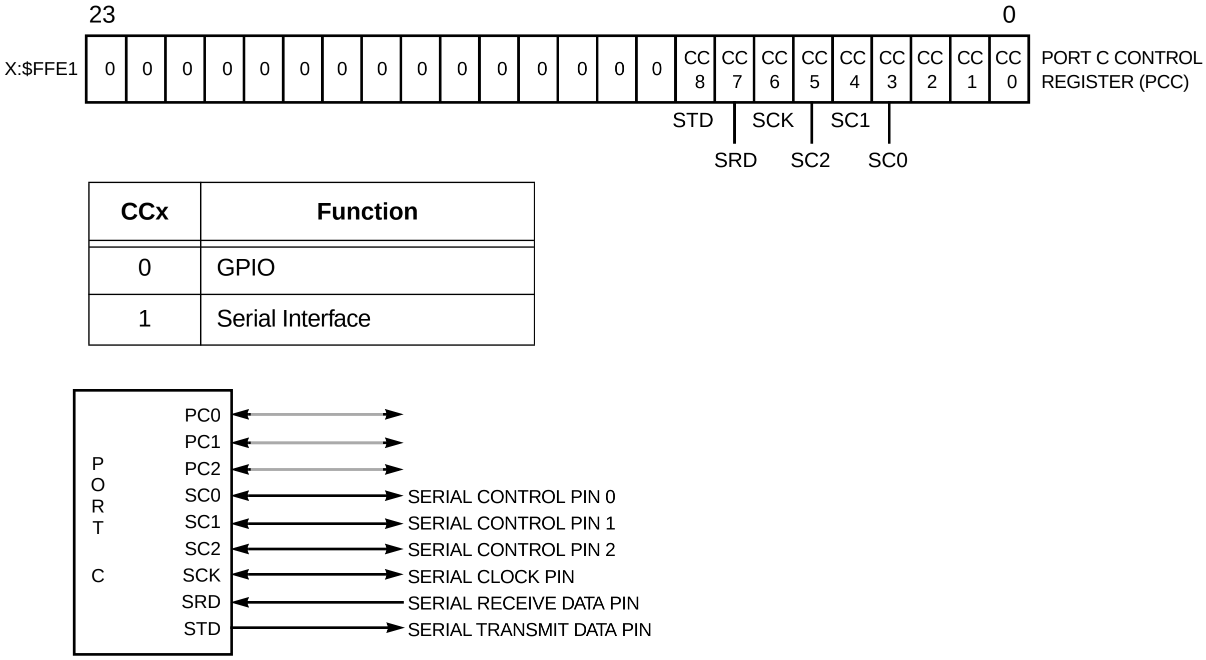


Figure 7-17 SSI CRB Initialization Procedure



CCx	Function
0	GPIO
1	Serial Interface

Figure 7-18 SSI Initialization Procedure

for the common data communication rates and the highest rate possible for the SSI for the chosen crystal frequencies. The crystal frequency selected for Table 7-11 (a) is the one used by the DSP56003/005ADS board; the one selected for Table 7-11 (b) is the closest one to 40 MHz that divides down to exactly 128 kHz. If an exact baud rate is required, the crystal frequency may have to be selected. Table 7-12 gives the PSR and PM0–PM7 settings in addition to the required crystal frequency for three common telecommunication frequencies.

7.3.6 SSI Exceptions

The SSI can generate four different exceptions (see Figure 7-19 and Figure 7-20):

1. SSI Receive Data – occurs when the receive interrupt is enabled, the receive data register is full, and no receive error conditions exist. Reading RX clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead.

2. SSI Receive Data with Exception Status – occurs when the receive interrupt is enabled, the receive data register is full, and a receiver overrun error has occurred. ROE is cleared by first reading the SSISR and then reading RX.

Table 7-11 (a) SSI Bit Rates for a 40-MHz Crystal

Bit Rate (BPS)	PSR	PM
1000	1	$4E1
2000	1	$270
4000	1	$138
8000	1	$9B
16K	1	$4D
32K	1	$26
64K	0	$9B
128K	0	$4D
10M	0	$00

BPS = f_{osc} ÷ (4 × (7 × (PSR) +1) × (PM + 1)) where
f_{osc} = 40 MHz
PSR = 0 or 1
PM = 0 to $FFF

Table 7-11 (b) SSI Bit Rates for a 39.936-MHz Crystal

Bit Rate (BPS)	PSR	PM
1000	1	$4DF
2000	1	$26F
4000	1	$137
8000	1	$9B
16K	1	$4D
32K	1	$26
64K	0	$9B
128K	0	$4D
9.984M	0	$00

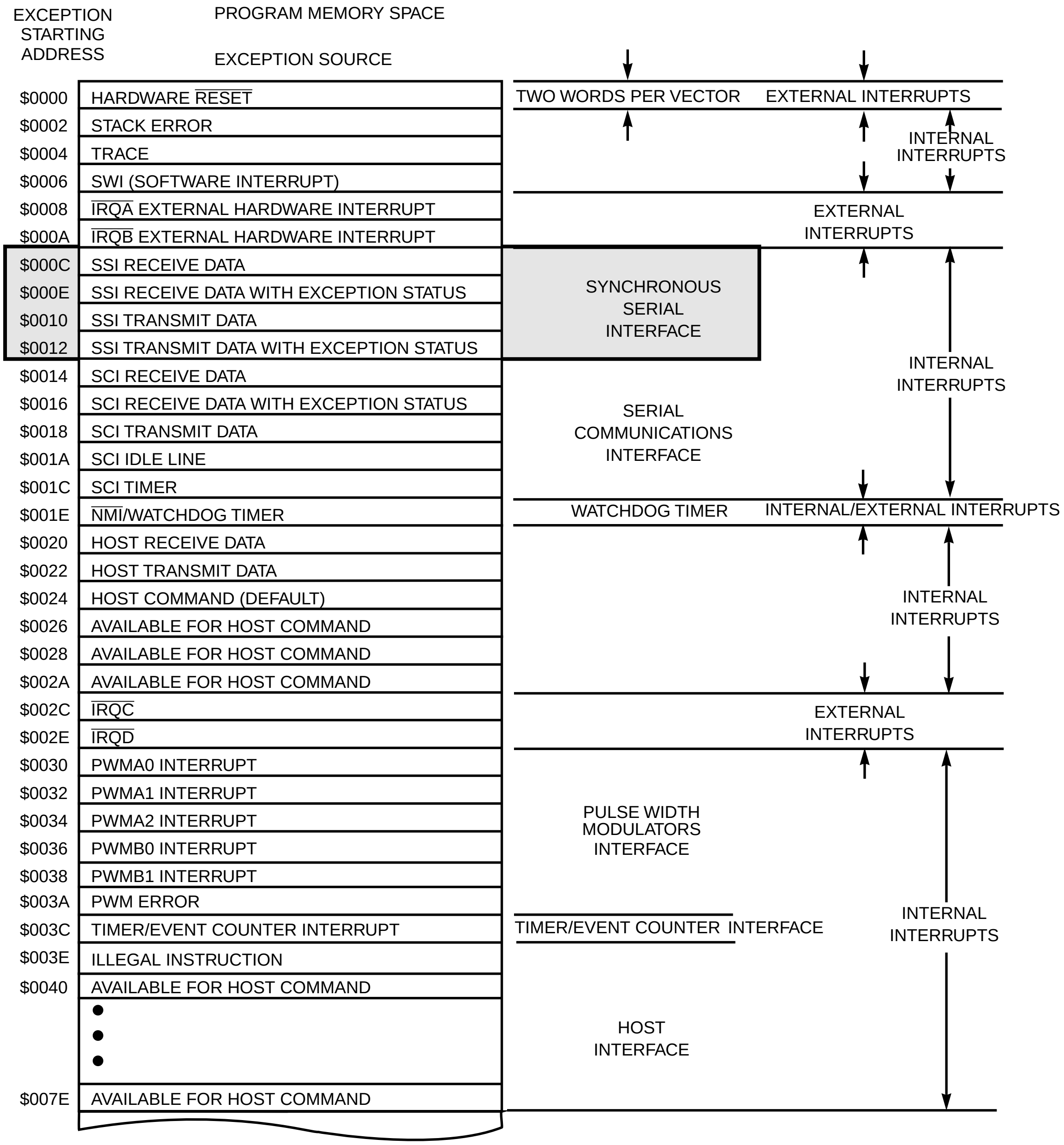
BPS = f_{osc} ÷ (4 × (7 × (PSR) +1) × (PM + 1)) where
f_{osc} = 39.936 MHz
PSR = 0 or 1
PM = 0 to $FFF

Table 7-12 Crystal Frequencies Required for Codecs

Bit Rate (BPS)	PSR	PM	Crystal Frequency
1.536M	0	$05	36.864 MHz
1.544M	0	$05	37.056 MHz
2.048M	0	$03	32.678 MHz

BPS = f_{osc} ÷ (4 × (7 × (PSR) +1) × (PM + 1))
PSR = 0 or 1
PM = 0 to $FFF

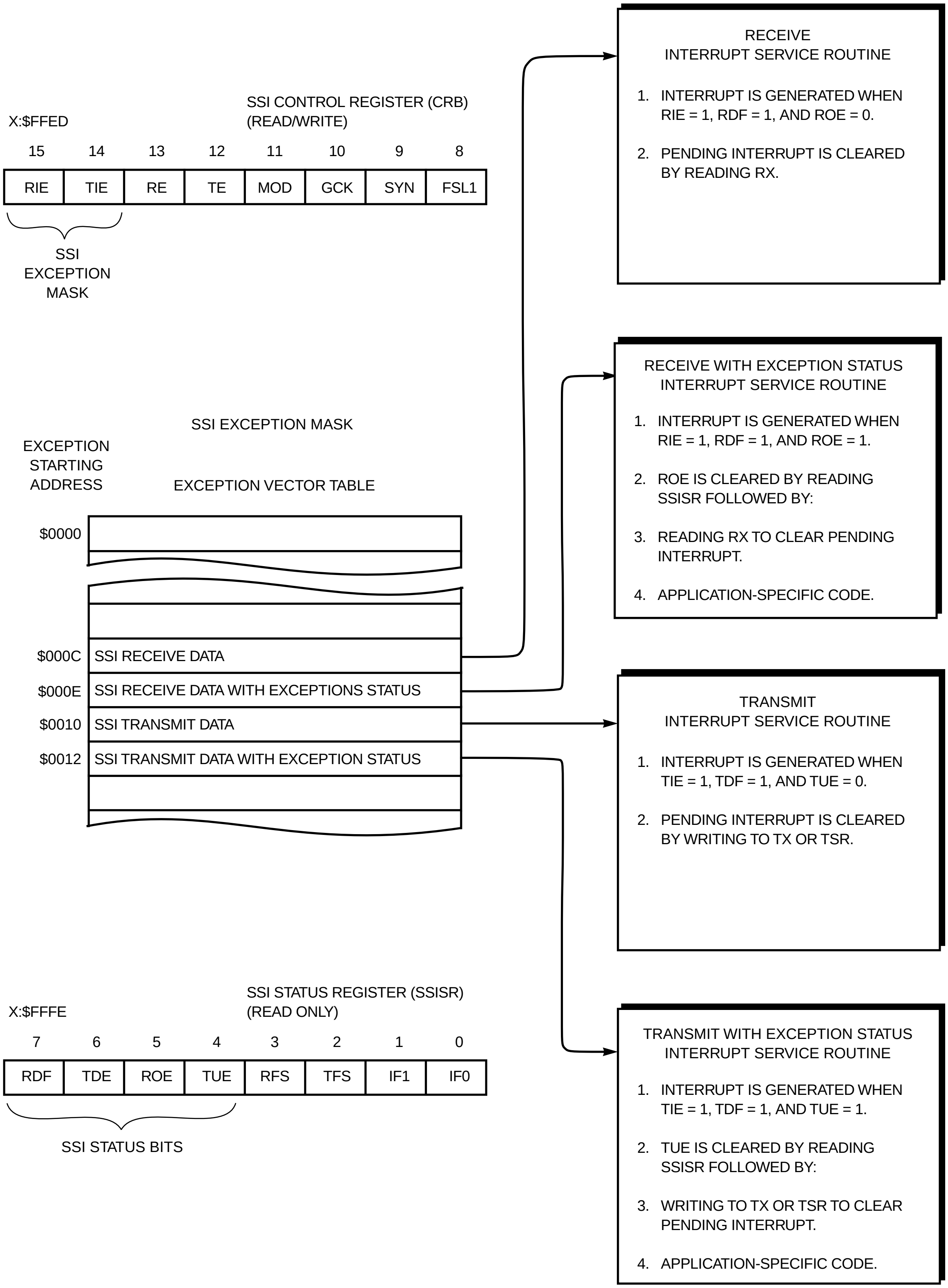
3. SSI Transmit Data – occurs when the transmit interrupt is enabled, the trans-



EXCEPTION STARTING ADDRESS	PROGRAM MEMORY SPACE EXCEPTION SOURCE
$0000	HARDWARE $\overline{\text{RESET}}$
$0002	STACK ERROR
$0004	TRACE
$0006	SWI (SOFTWARE INTERRUPT)
$0008	$\overline{\text{IRQA}}$ EXTERNAL HARDWARE INTERRUPT
$000A	$\overline{\text{IRQB}}$ EXTERNAL HARDWARE INTERRUPT
$000C	SSI RECEIVE DATA
$000E	SSI RECEIVE DATA WITH EXCEPTION STATUS
$0010	SSI TRANSMIT DATA
$0012	SSI TRANSMIT DATA WITH EXCEPTION STATUS
$0014	SCI RECEIVE DATA
$0016	SCI RECEIVE DATA WITH EXCEPTION STATUS
$0018	SCI TRANSMIT DATA
$001A	SCI IDLE LINE
$001C	SCI TIMER
$001E	$\overline{\text{NMI}}$/WATCHDOG TIMER
$0020	HOST RECEIVE DATA
$0022	HOST TRANSMIT DATA
$0024	HOST COMMAND (DEFAULT)
$0026	AVAILABLE FOR HOST COMMAND
$0028	AVAILABLE FOR HOST COMMAND
$002A	AVAILABLE FOR HOST COMMAND
$002C	$\overline{\text{IRQC}}$
$002E	$\overline{\text{IRQD}}$
$0030	PWMA0 INTERRUPT
$0032	PWMA1 INTERRUPT
$0034	PWMA2 INTERRUPT
$0036	PWMB0 INTERRUPT
$0038	PWMB1 INTERRUPT
$003A	PWM ERROR
$003C	TIMER/EVENT COUNTER INTERRUPT
$003E	ILLEGAL INSTRUCTION
$0040	AVAILABLE FOR HOST COMMAND
	● ● ●
$007E	AVAILABLE FOR HOST COMMAND

Figure 7-19 HI Exception Vector Locations

mit data register is empty, and no transmitter error conditions exist. Writing to



15	14	13	12	11	10	9	8
RIE	TIE	RE	TE	MOD	GCK	SYN	FSL1

$000C	SSI RECEIVE DATA
$000E	SSI RECEIVE DATA WITH EXCEPTIONS STATUS
$0010	SSI TRANSMIT DATA
$0012	SSI TRANSMIT DATA WITH EXCEPTION STATUS

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

RECEIVE
INTERRUPT SERVICE ROUTINE

1. INTERRUPT IS GENERATED WHEN RIE = 1, RDF = 1, AND ROE = 0.
2. PENDING INTERRUPT IS CLEARED BY READING RX.

RECEIVE WITH EXCEPTION STATUS
INTERRUPT SERVICE ROUTINE

1. INTERRUPT IS GENERATED WHEN RIE = 1, RDF = 1, AND ROE = 1.
2. ROE IS CLEARED BY READING SSISR FOLLOWED BY:
3. READING RX TO CLEAR PENDING INTERRUPT.
4. APPLICATION-SPECIFIC CODE.

TRANSMIT
INTERRUPT SERVICE ROUTINE

1. INTERRUPT IS GENERATED WHEN TIE = 1, TDF = 1, AND TUE = 0.
2. PENDING INTERRUPT IS CLEARED BY WRITING TO TX OR TSR.

TRANSMIT WITH EXCEPTION STATUS
INTERRUPT SERVICE ROUTINE

1. INTERRUPT IS GENERATED WHEN TIE = 1, TDF = 1, AND TUE = 1.
2. TUE IS CLEARED BY READING SSISR FOLLOWED BY:
3. WRITING TO TX OR TSR TO CLEAR PENDING INTERRUPT.
4. APPLICATION-SPECIFIC CODE.

Figure 7-20 SSI Exceptions

TX or the TSR will clear this interrupt. This error-free interrupt may use a fast interrupt service routine for minimum overhead.

4. SSI Transmit Data with Exception Status – occurs when the transmit interrupt is enabled, the transmit data register is empty, and a transmitter underrun error has occurred. TUE is cleared by first reading the SSISR and then writing to TX or the TSR to clear the pending interrupt.

7.3.7 Operating Modes – Normal, Network, and On-Demand

The SSI has three basic operating modes and many data/operation formats. These modes can be programmed by several bits in the SSI control registers. Table 7-13 lists the SSI operating modes and some of the typical applications in which they may be used.

The data/operation formats are selected by choosing between gated and continuous clocks, synchronization of transmitter and receiver, selection of word or bit frame sync, and whether the LSB is transferred first or last. The following paragraphs describe how to select a particular data/operation format and describe examples of normal-mode and network-mode applications. The on-demand mode is selected as a special case of the network mode.

The SSI can function as an SPI master or SPI slave, using additional logic for arbitration, which is required because the SSI interface does not perform SPI master/slave arbitration. An SPI master device always uses an internally generated clock; whereas, an SPI slave device always uses an external clock.

7.3.7.1 Data/Operation Formats

The data/operation formats available to the SSI are selected by setting or clearing control

Table 7-13 SSI Operating Modes

Operating Format	Serial Clock	TX, RX Sections	Typical Applications
Normal	Continuous	Asynchronous	Single Asynchronous Codec; Stream-Mode Channel Interface
Normal	Continuous	Synchronous	Multiple Synchronous Codecs
Normal	Gated	Asynchronous	DSP-to-DSP; Serial Peripherals (A/D,D/A)
Normal	Gated	Synchronous	SPI-Type Devices; DSP to MCU
Network	Continuous	Asynchronous	TDM Networks
Network	Continuous	Synchronous	TDM Codec Networks, TDM DSP Networks
On Demand	Gated	Asynchronous	Parallel-to-Serial and Serial-to-Parallel Conversion
On Demand	Gated	Synchronous	DSP to SPI Peripherals

bits in the CRB. These control bits are MOD, GCK, SYN, FSL1, FSL0, and SHFD.

7.3.7.1.1 Normal/Network Mode Selection

Selecting between the normal mode and network mode is accomplished by clearing or setting the MOD bit in the CRB (see Figure 7-21). For normal mode, the SSI functions with one data word of I/O per frame (see Figure 7-22). For the network mode, 2 to 32 data words of I/O may be used per frame. In either case, the transfers are periodic. The normal mode is typically used to transfer data to/from a single device. Network mode is typically used in time division multiplexed (TDM) networks of codecs or DSPs with multiple words per frame (see Figure 7-22, which shows two words in a frame with either word-length or bit-length frame sync). The frame sync shown in Figure 7-21 is the word-length frame sync. A bit-length frame sync can be chosen by setting FSL1 and FSL0 for the configuration desired.

7.3.7.1.2 Continuous/Gated Clock Selection

The TX and RX clocks may be programmed as either continuous or gated clock signals by the GCK bit in the CRB. A continuous TX and RX clock is required in applications such as communicating with some codecs where the clock is used for more than just data transfer. A gated clock, in which the clock only toggles while data is being transferred, is useful for many applications and is required for SPI compatibility. The frame sync outputs may be used as a start conversion signal by some A/D and D/A devices.

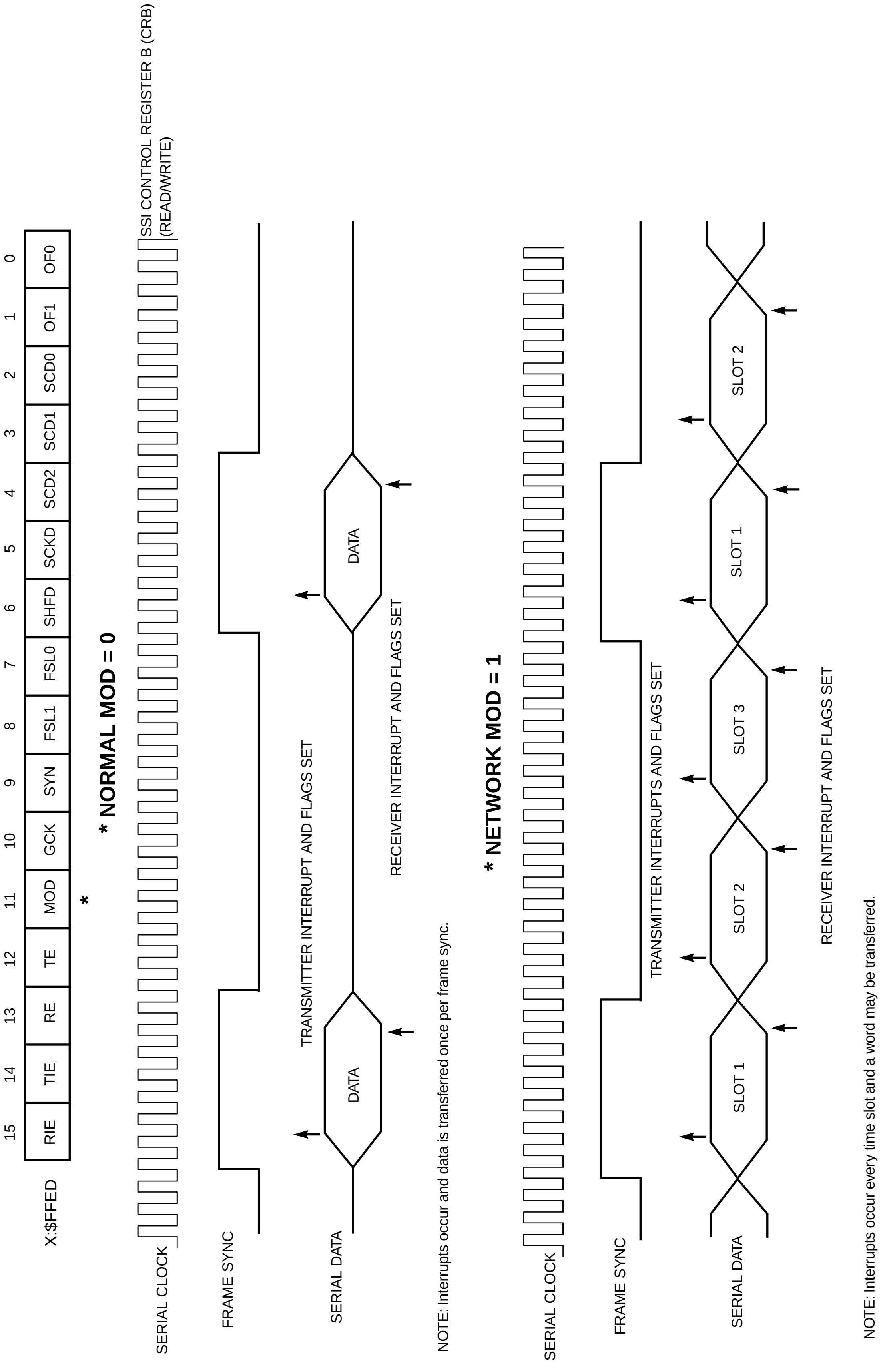
Figure 7-23 illustrates the difference between continuous clock and gated clock systems. A separate frame-sync signal is required in continuous clock systems to delimit the active clock transitions. Although the word-length frame sync is shown in Figure 7-23, a bit-length frame sync can be used (see Figure 7-24). In gated clock systems, frame synchronization is inherent in the clock signal; thus a separate sync signal is not required (see Figure 7-25 and Figure 7-26). The SSI can be programmed to generate frame sync outputs in gated clock mode but does not use frame sync inputs.

Input flags (see Figure 7-25 and Figure 7-26) are latched on the negative edge of the first data bit of a frame. Output flags are valid during the entire frame.

7.3.7.1.3 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of this interface may be synchronous or asynchronous – i.e., the transmitter and receiver may use common clock and synchronization signals (synchronous operating mode, see Figure 7-27) or they may have their own separate clock and sync signals (asynchronous operating mode). The SYN bit in CRB selects synchronous or asynchronous operation. Since the SSI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

Figure 7-28 illustrates the operation of the SYN bit in the CRB. When SYN equals zero, the SSI



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

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

Figure 7-21 CRB MOD Bit Operation

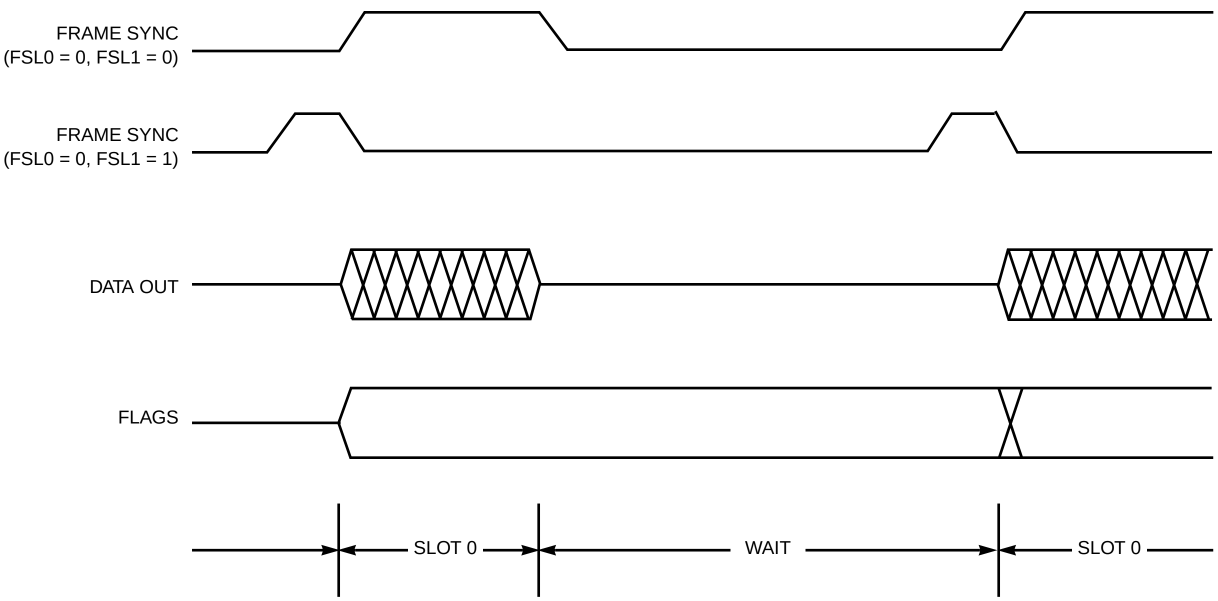


Figure 7-22 Normal Mode, External Frame Sync (8 Bit, 1 Word in Frame)

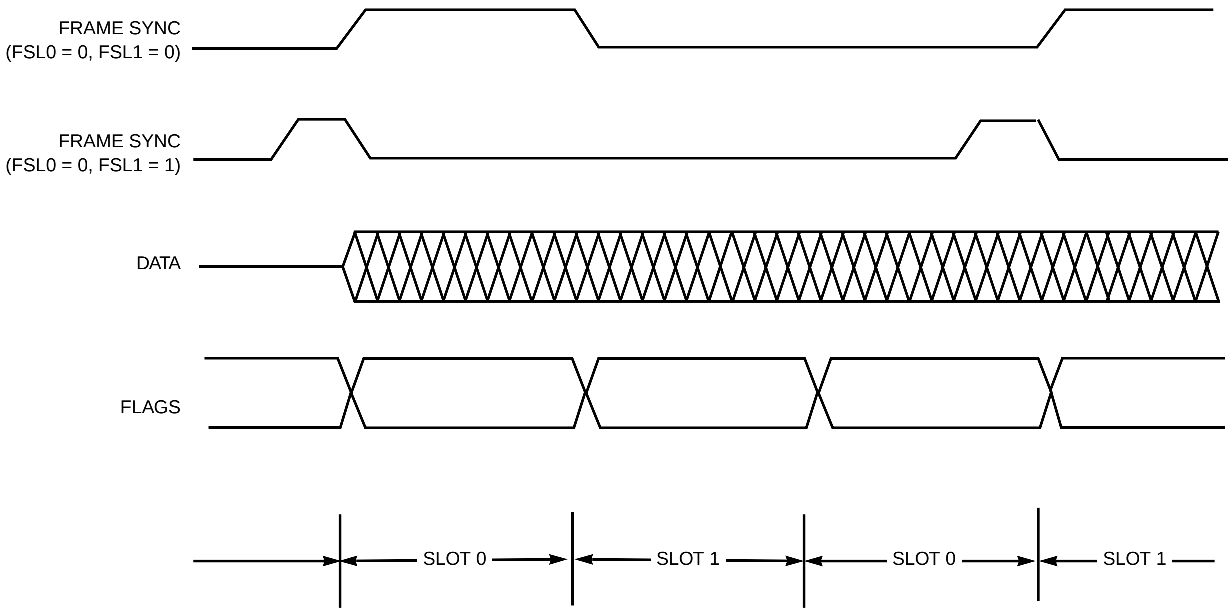
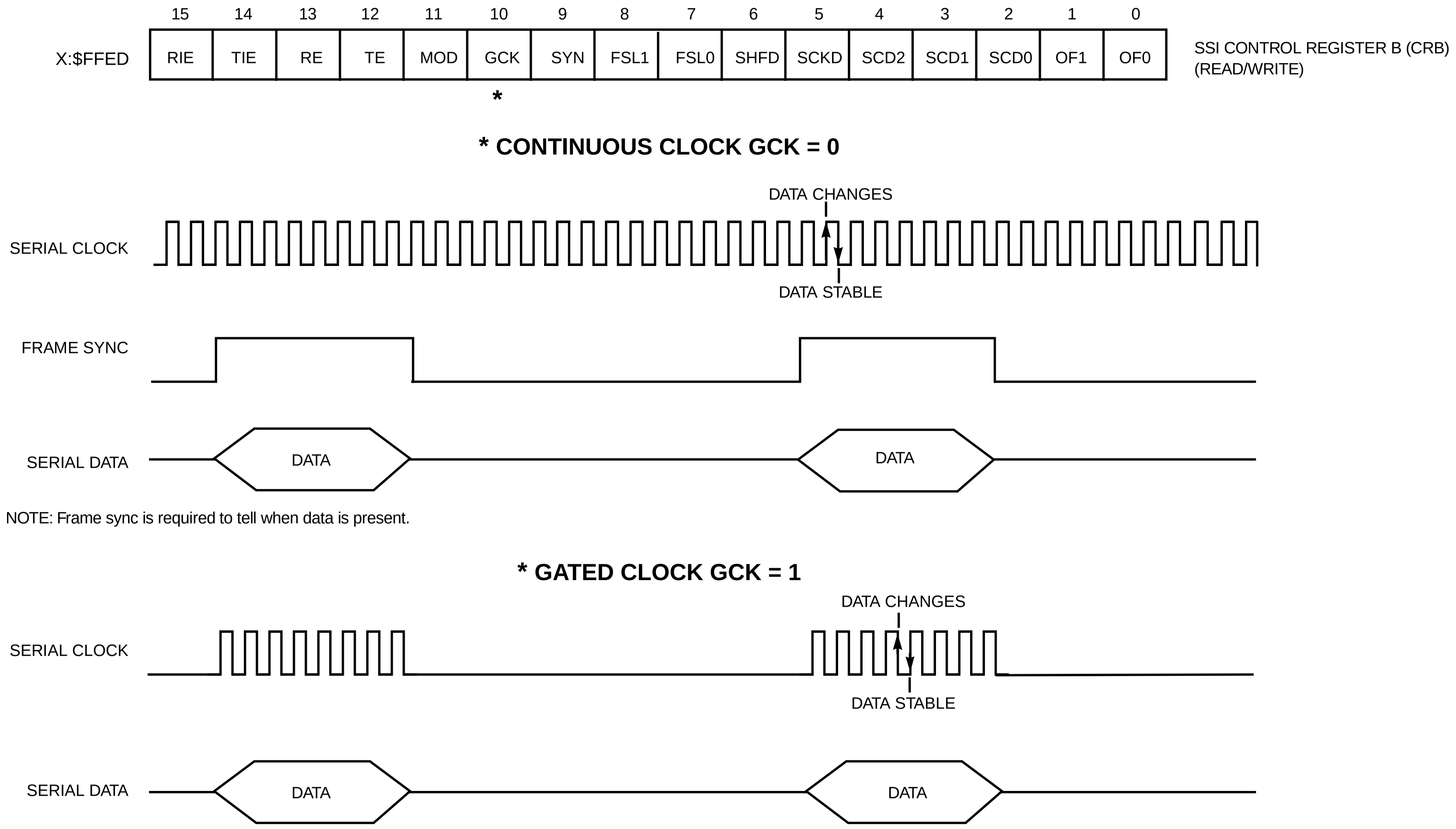


Figure 7-22 Network Mode, External Frame Sync (8 Bit, 2 Words in Frame)

TX and RX clocks and frame sync sources are independent. If SYN equals one, the SSI TX

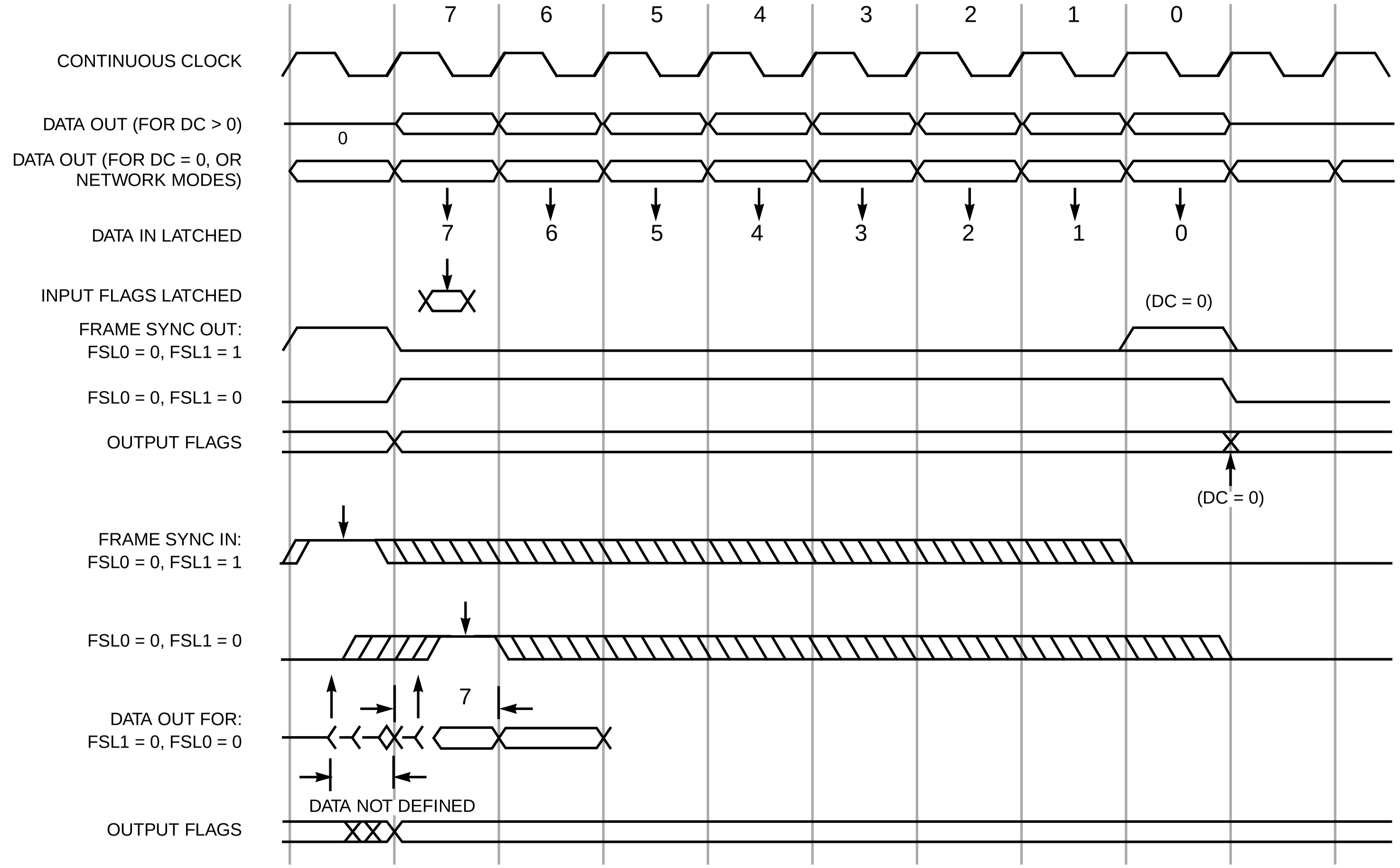


NOTE: Frame sync is required to tell when data is present.

NOTES:
1. Word synchronization is inherent in the serial clock signal.
2. Frame Sync generation is optional.

Figure 7-23 CRB GCK Bit Operation

and RX clocks and frame sync come from the same source (either external or internal).



NOTES:
1. For FSL1 = 0 the frame sync is latched and enables the STD output buffer, but data may not be valid until the rising edge of the bit clock.
2. WL bit frame sync (FSL0 = 0, FSL1 = 0) is not defined for DC = 0 in continuous clock mode.
3. Data and flags transition after external frame sync but not before the rising edge of the clock.

Figure 7-24 Continuous Clock Timing Diagram (8-Bit Example)

Data clock and frame sync signals can be generated internally by the DSP or may be ob-

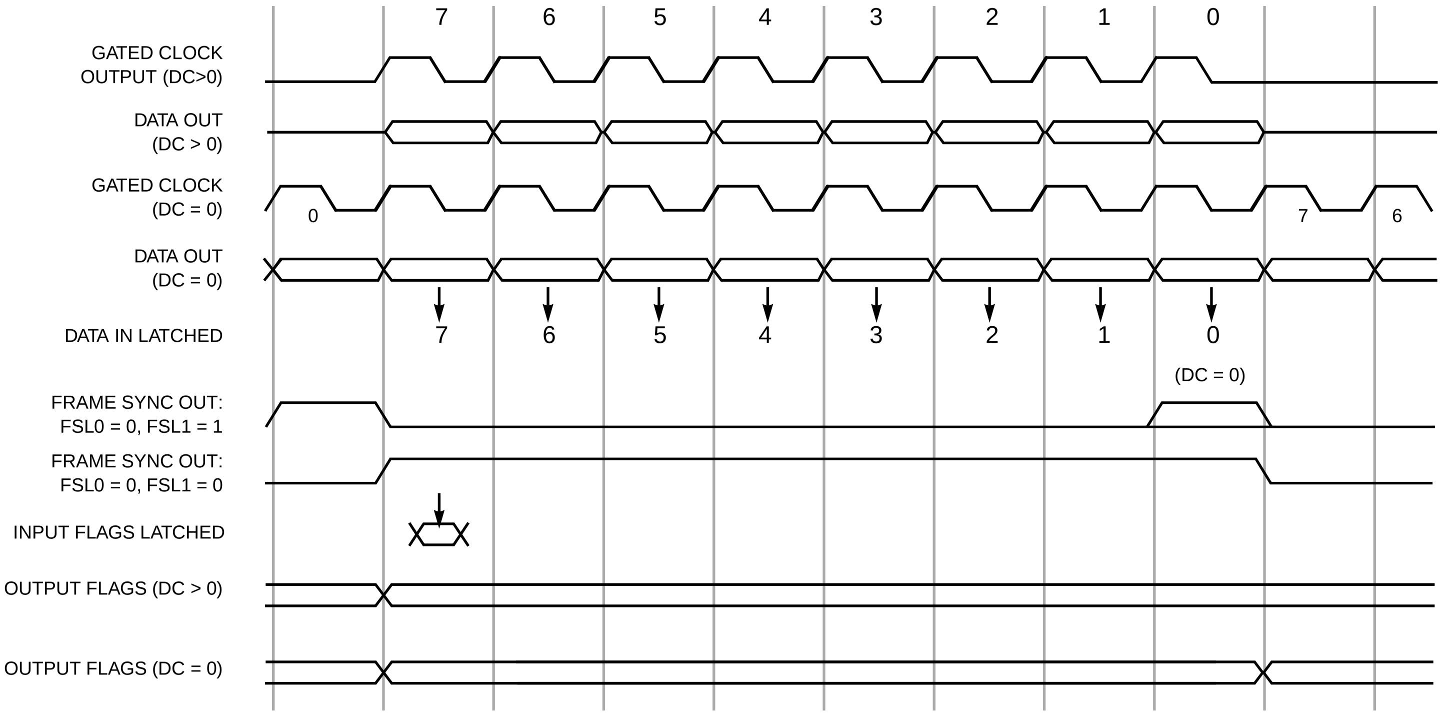
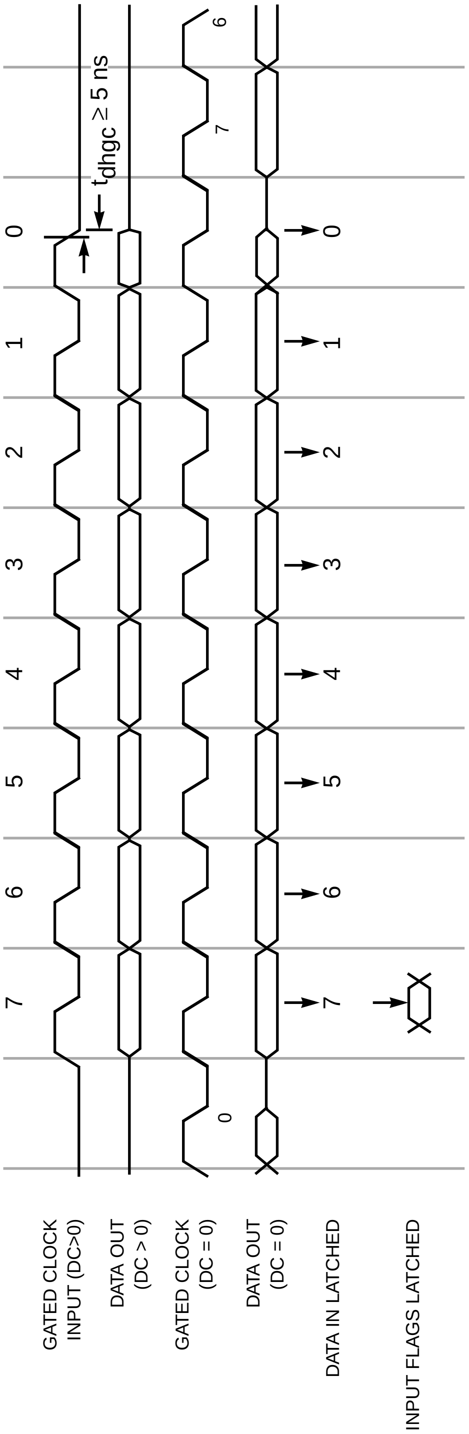


Figure 7-25 Internally Generated Clock Timing (8-Bit Example)



NOTES:
1. Output enabled on rising edge of first clock input.
2. Output disabled on falling edge of last clock pulse.
3. t_{dhgc} is guaranteed by circuit design.
4. Frame syncs (in or out) are not defined for external gated clock mode.

Figure 7-26 Externally Generated Gated Clock Timing (8-Bit Example)

tained from external sources. If internally generated, the SSI clock generator is used to de-

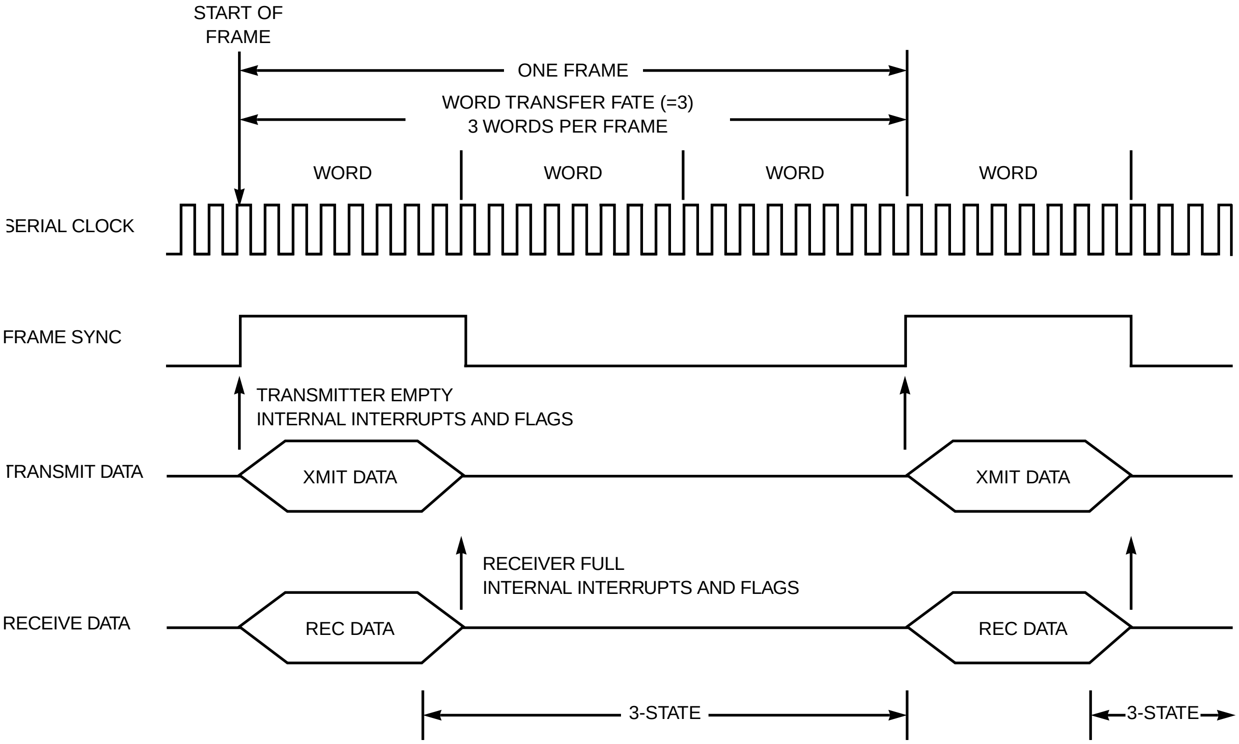


Figure 7-27 Synchronous Communication

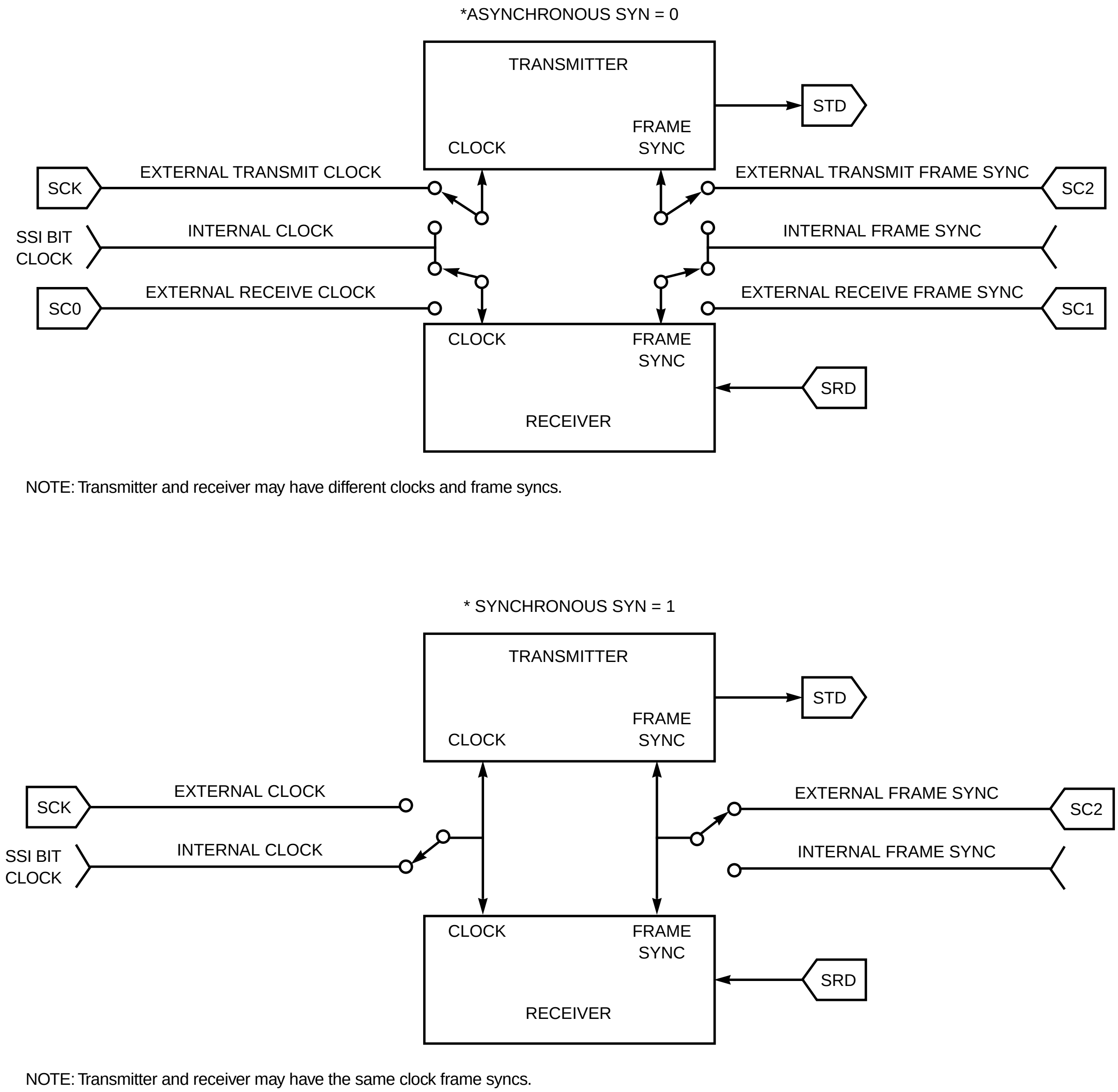
rive bit clock and frame sync signals from the DSP internal system clock. The SSI clock generator consists of a selectable fixed prescaler and a programmable prescaler for bit rate clock generation and also a programmable frame-rate divider and a word-length divider for frame-rate sync-signal generation.

Figures Figure 7-29 through Figure 7-32 show the definitions of the SSI pins during each of the four main operating modes of the SSI I/O interface. Figure 7-29 uses a gated clock (from either an external source or the internal clock), which means that frame sync is inherent in the clock. Since both the transmitter and receiver use the same clock (synchronous configuration), both use the SCK pin. SC0 and SC1 are designated as flags or can be used as general purpose-parallel I/O. SC2 is not defined if it is an input; SC2 is the transmit and receive frame sync if it is an output.

Figure 7-30 shows a gated clock (from either an external source or the internal clock), which means that frame sync is inherent in the clock. Since this configuration is asynchronous, SCK is the transmitter clock pin (input or output) and SC0 is the receiver clock pin (input or output). SC1 and SC2 are designated as receive or transmit frame sync, respectively, if they are selected to be outputs; these bits are undefined if they are selected to be inputs. SC1 and SC2 can also be used as general-purpose parallel I/O.

SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FF	RIE	TIE	RE	TE	MOD	GCK	SYN	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0
							*									



NOTE: Transmitter and receiver may have different clocks and frame syncs.

NOTE: Transmitter and receiver may have the same clock frame syncs.

Figure 7-28 CRB SYN Bit Operation

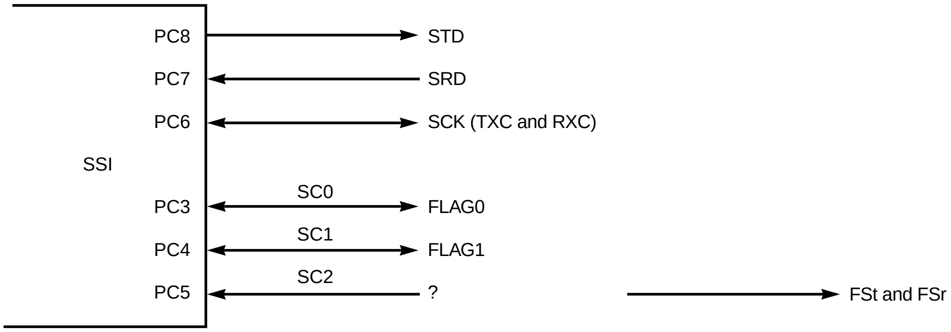


Figure 7-29 Gated Clock — Synchronous Operation

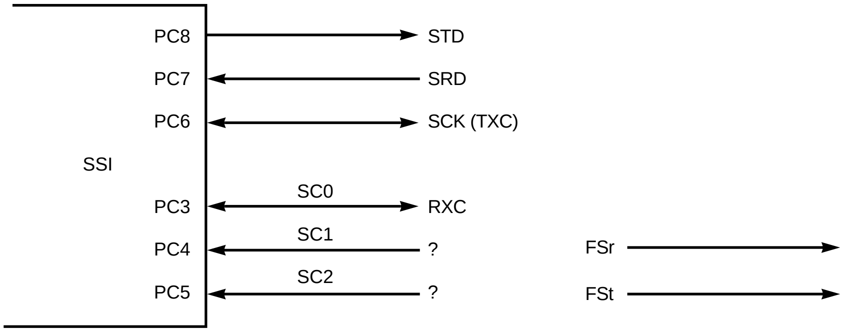


Figure 7-30 Gated Clock — Asynchronous Operation

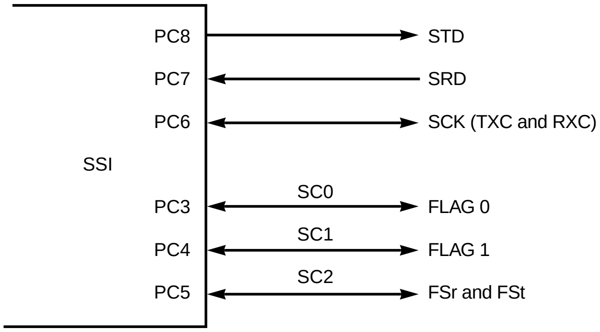


Figure 7-31 Continuous Clock — Synchronous Operation

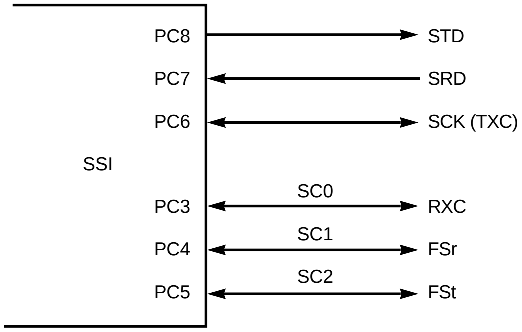


Figure 7-32 Continuous Clock — Asynchronous Operation

Figure 7-31 shows a continuous clock (from either an external source or the internal clock), which means that frame sync must be a separate signal. SC2 is used for frame sync, which can come from an internal or external source. Since both the transmitter and receiver use the same clock (synchronous configuration), both use the SCK pin. SC0 and SC1 are designated as flags or can be used as general-purpose parallel I/O.

Figure 7-32 shows a continuous clock (from either an external source or the internal clock), which means that frame sync must be a separate signal. SC1 is used for the receive frame sync, and SC2 is used for the transmit frame sync. Either frame sync can come from an internal or external source. Since the transmitter and receiver use different clocks (asynchronous configuration), SCK is used for the transmit clock, and SC0 is used for the receive clock.

7.3.7.1.4 Frame Sync Selection

The transmitter and receiver can operate totally independent of each other. The transmitter can have either a bit-long or word-long frame-sync signal format, and the receiver can have the same or opposite format. The selection is made by programming FSL0 and FSL1 in the CRB as shown in Figure 7-33.

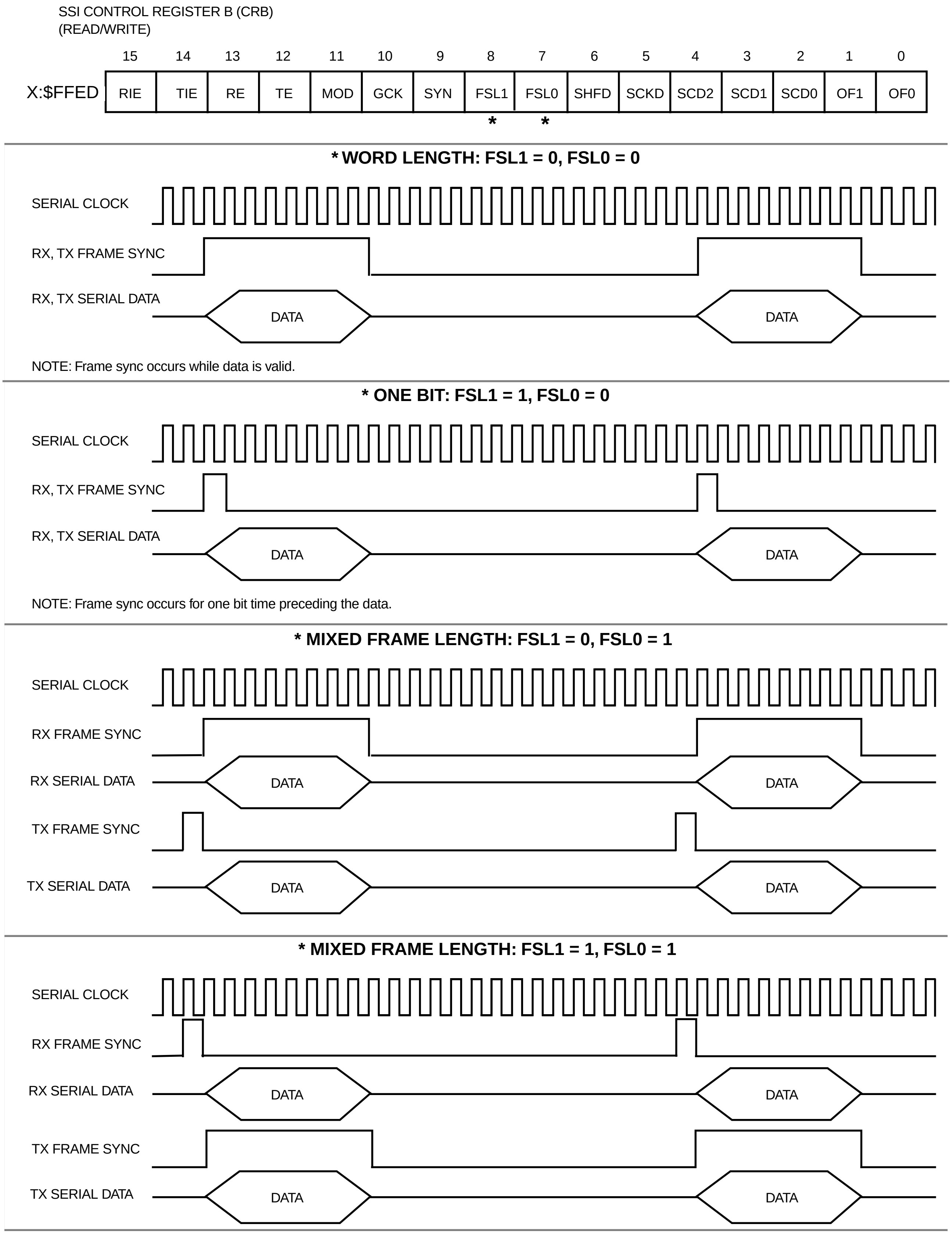
1. If FSL1 equals zero (see Figure 7-34), the RX frame sync is asserted during the entire data transfer period. This frame sync length is compatible with Motorola codecs, SPI serial peripherals, serial A/D and D/A converters, shift registers, and telecommunication PCM serial I/O.

2. If FSL1 equals one (see Figure 7-35), the RX frame sync pulses active for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs, and telecommunication PCM serial I/O.

The ability to mix frame sync lengths is useful in configuring systems in which data is received from one type device (e.g., codec) and transmitted to a different type device.

FSL0 controls whether RX and TX have the same frame sync length (see Figure 7-33). If FSL0 equals zero, RX and TX have the same frame sync length, which is selected by FSL1. If FSL0 equals one, RX and TX have different frame sync lengths, which are selected by FSL1.

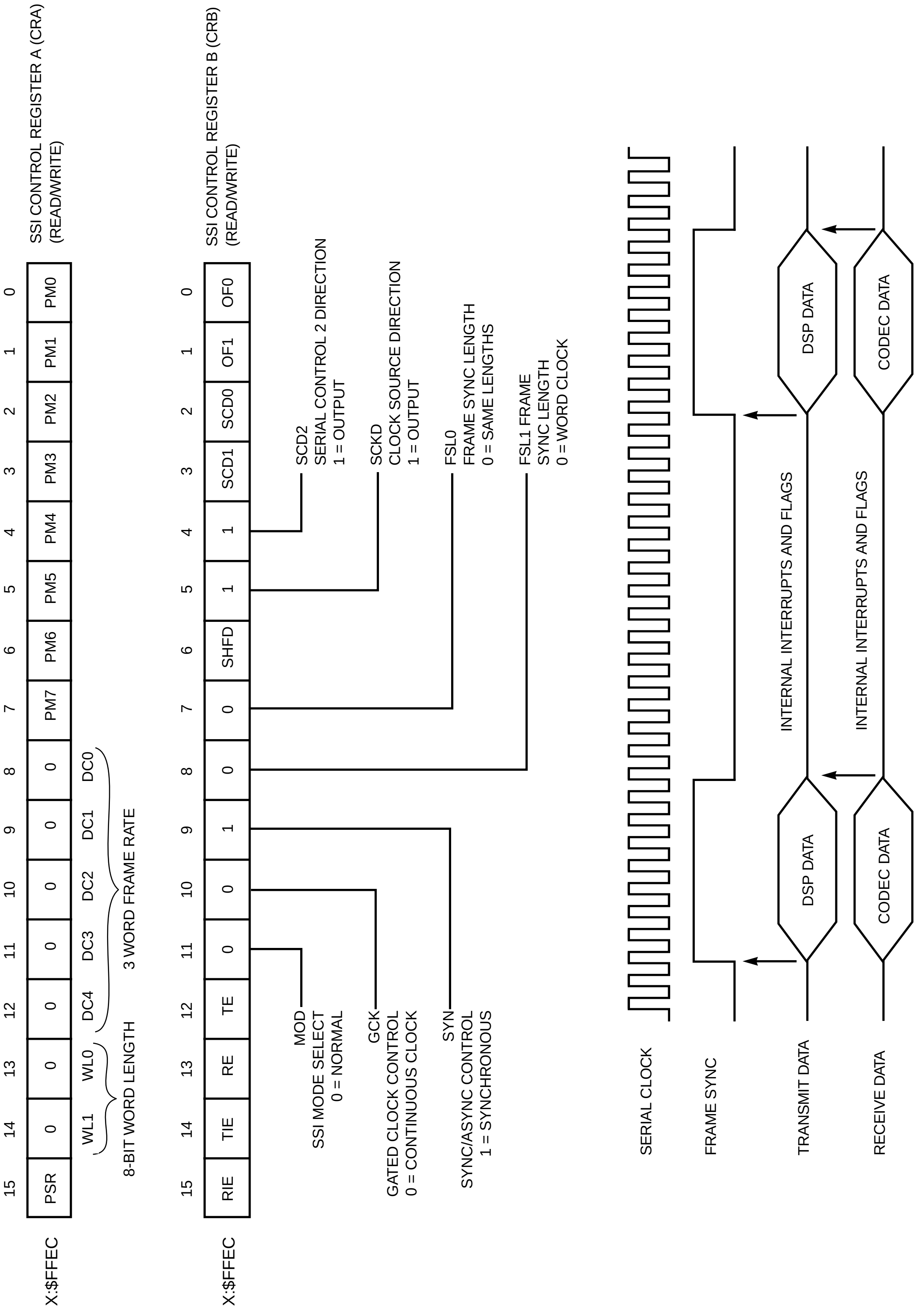
The SSI receiver looks for a receive frame sync leading edge only when the previous frame is completed. If the frame sync goes high before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync), the current frame sync will not be recognized, and the receiver will be internally disabled until the next frame sync. Frames do not have to be adjacent – i.e., a new frame sync does not have to immediately follow the previous frame. Gaps of arbitrary periods can occur between frames. The transmitter will be three-stated during these gaps.



SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFED	RIE	TIE	RE	TE	MOD	GCK	SYN	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0
								*	*							

Figure 7-33 CRB FSL0 and FSL1 Bit Operation



SSI CONTROL REGISTER A (CRA) (READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFEC	PSR	0	0	0	0	0	0	0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
		WL1	WL0	DC4	DC3	DC2	DC1	DC0								

8-BIT WORD LENGTH

3 WORD FRAME RATE

SSI CONTROL REGISTER B (CRB) (READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFEC	RIE	TIE	RE	TE	0	0	1	0	0	SHFD	1	1	SCD1	SCD0	OF1	OF0

MOD
SSI MODE SELECT
0 = NORMAL

GCK
GATED CLOCK CONTROL
0 = CONTINUOUS CLOCK

SYN
SYNC/ASYNC CONTROL
1 = SYNCHRONOUS

FSL1 FRAME
SYNC LENGTH
0 = WORD CLOCK

FSL0
FRAME SYNC LENGTH
0 = SAME LENGTHS

SCKD
CLOCK SOURCE DIRECTION
1 = OUTPUT

SCD2
SERIAL CONTROL 2 DIRECTION
1 = OUTPUT

Figure 7-34 Normal Mode Initialization for FLS1=0 and FSL0=0

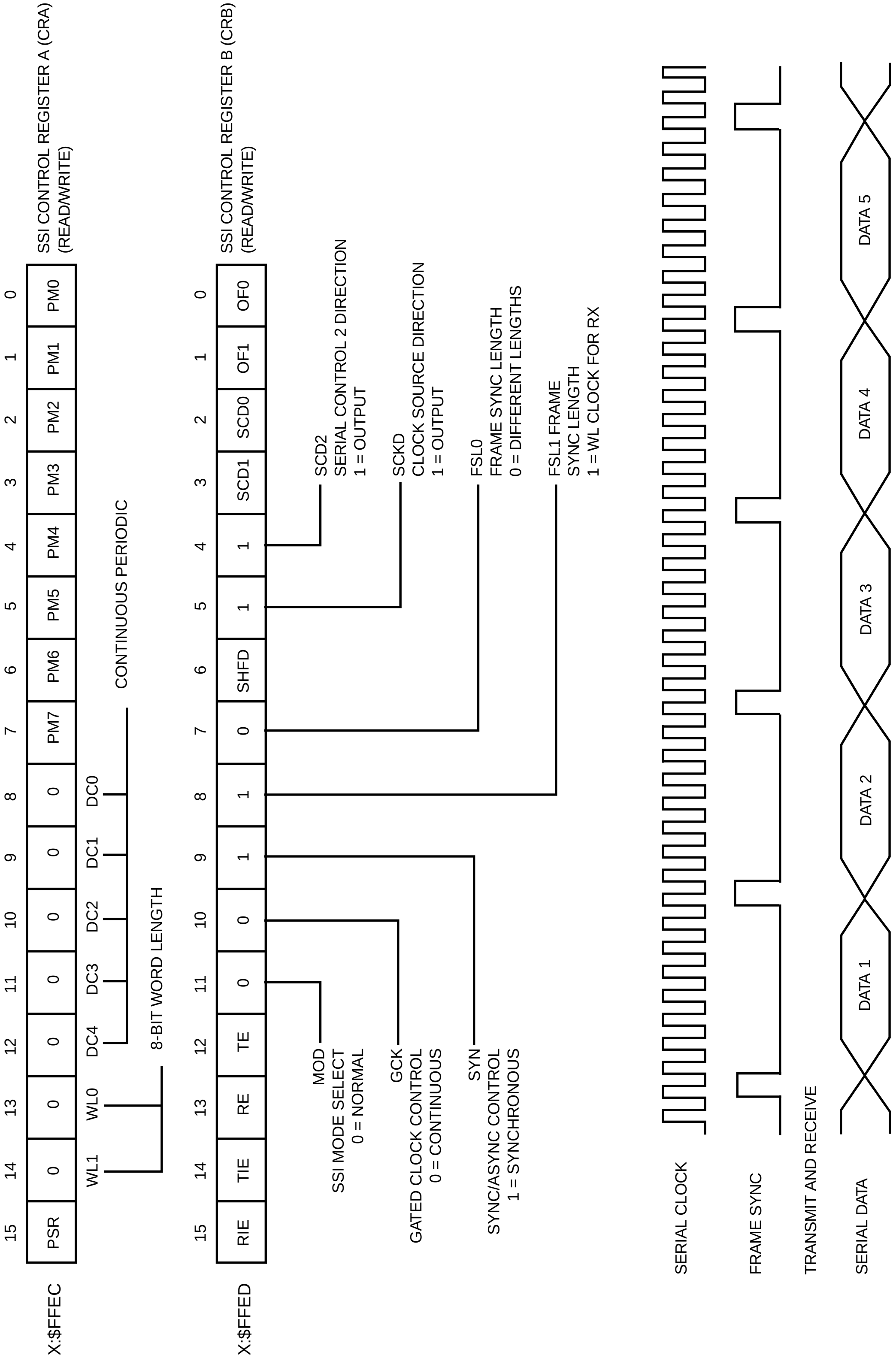


Figure 7-35 Normal Mode Initialization for FSL1=1 and FSL0=0

7.3.7.1.5 Shift Direction Selection

Some data formats, such as those used by codecs, specify MSB first other data formats, such as the AES-EBU digital audio, specify LSB first. To interface with devices from both systems, the shift registers in the SSI are bidirectional. The MSB/LSB selection is made by programming SHFD in the CRB.

Figure 7-36 illustrates the operation of the SHFD bit in the CRB. If SHFD equals zero (see Figure 7-36(a)), data is shifted into the receive shift register MSB first and shifted out of the transmit shift register MSB first. If SHFD equals one (see Figure 7-37(b)), data is shifted into the receive shift register LSB first and shifted out of the transmit shift register LSB first.

7.3.7.2 Normal Mode Examples

The normal SSI operating mode characteristically has one time slot per serial frame, and data is transferred every frame sync. When the SSI is not in the normal mode, it is in the network mode. The MSB is transmitted first (SHFD=0), with overrun and underrun errors detected by the SSI hardware. Transmit flags are set when data is transferred from the transmit data register to the transmit shift register. The receive flags are set when data is transferred from the receive shift register to the receive data register.

Figure 7-38 shows an example of using the SSI to connect an MC15500 codec with a DSP56003/005. No glue logic is needed. The serial clock, which is generated internally by the DSP, provides the transmit and receive clocks (synchronous operation) for the codec. SC2 provides all the necessary handshaking. Data transfer begins when the frame sync is asserted. Transmit data is clocked out and receive data is clocked in with the serial clock while the frame sync is asserted (word-length frame sync). At the end of the data transfer, DSP internal interrupts programmed to transfer data to/from will occur, and the SSISR will be updated.

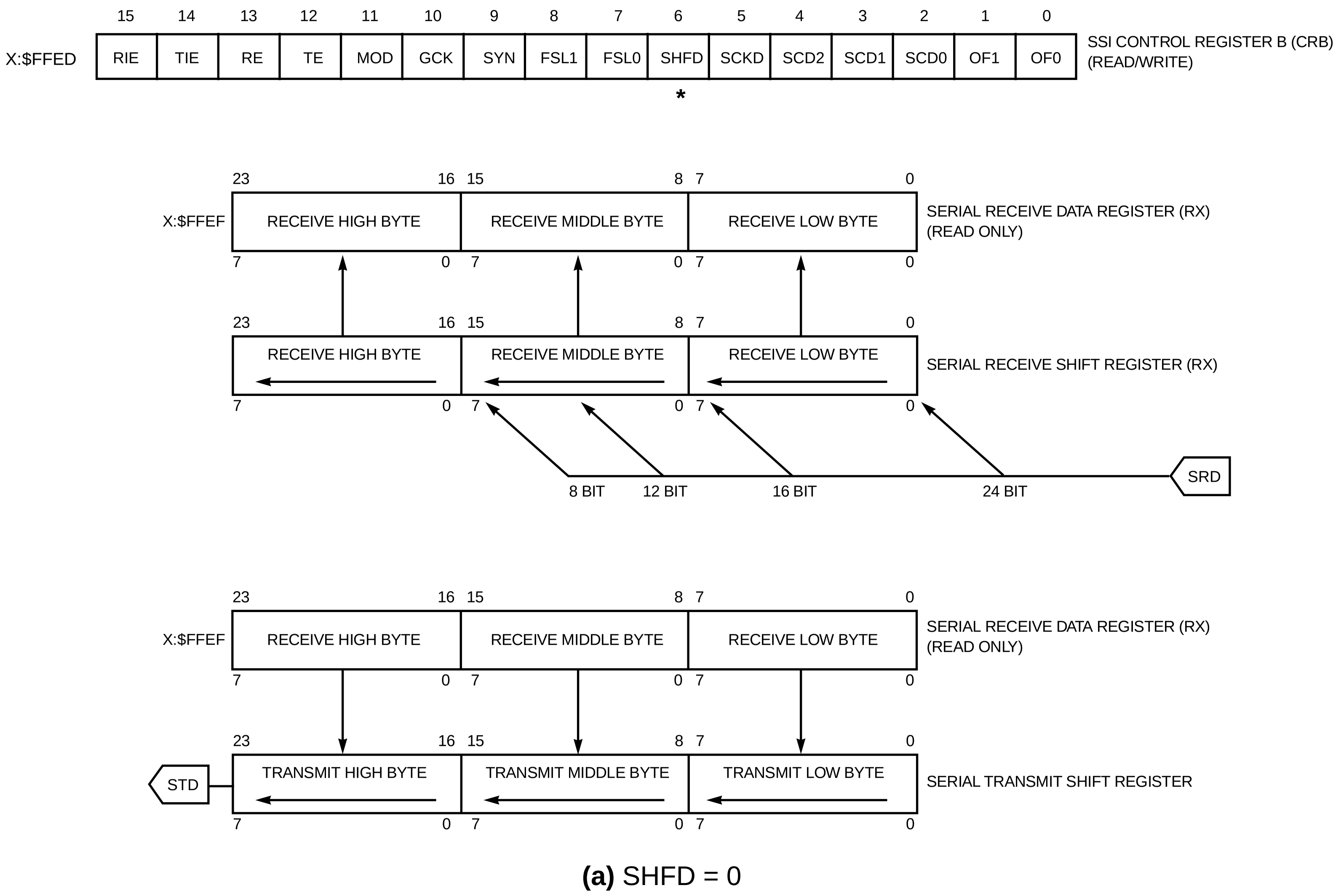
7.3.7.2.1 Normal Mode Transmit

The conditions for data transmission from the SSI are as follows:

1. Transmitter is Enabled (TE=1)
2. Frame sync (or clock in gated clock mode) is active

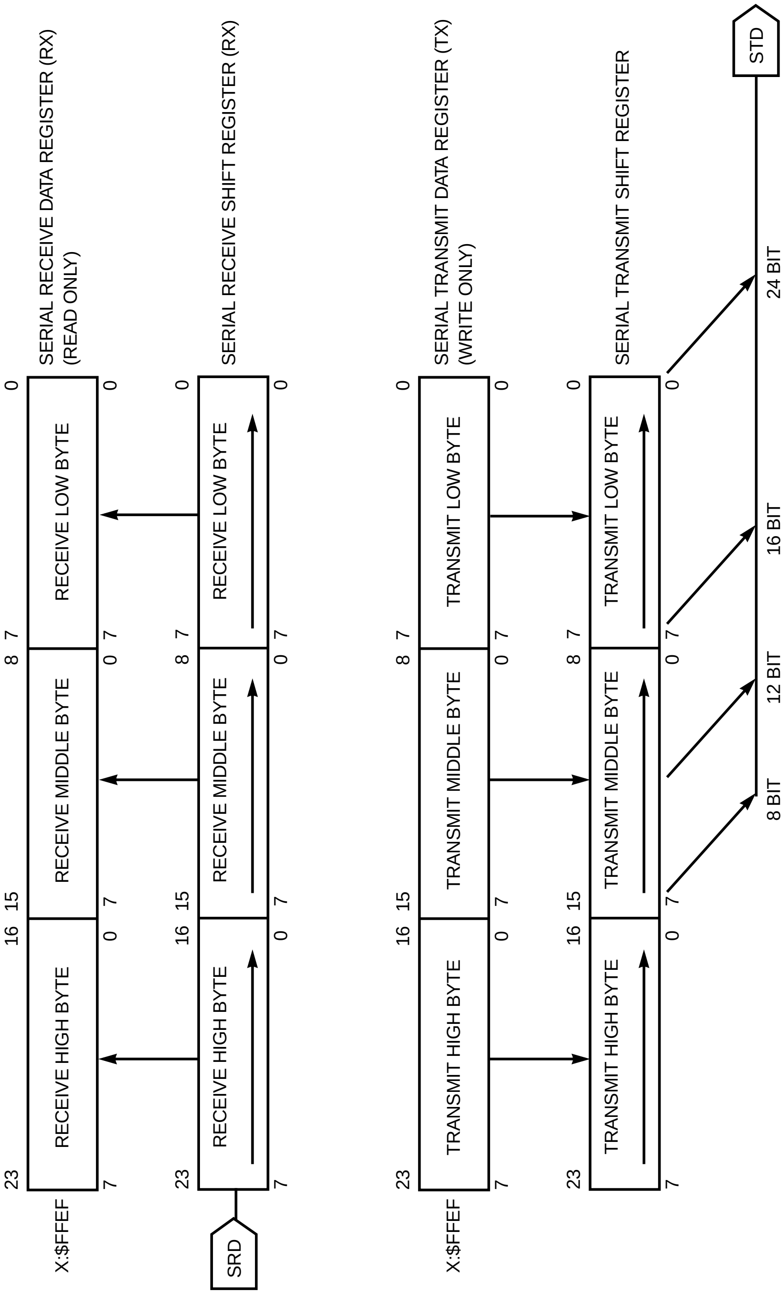
When these conditions occur in normal mode, the next data word will be transferred from TX to the transmit shift register, the TDE flag will be set (transmitter empty), and the transmit interrupt will occur if TIE equals one (transmit interrupt enabled.) The new data word will be transmitted immediately.

The transmit data output (STD) is three-stated, except during the data transmission period. The optional frame sync output, flag outputs, and clock outputs are not three-stated even if both receiver and transmitter are disabled.



(a) SHFD = 0

Figure 7-36 CRB SHFD Bit Operation (Sheet 1 of 2)



(b) SHFD=1

Figure 7-37 CRB SHFD Bit Operation (Sheet 2 of 2)

The optional output flags are always updated at the beginning of the frame, regardless of

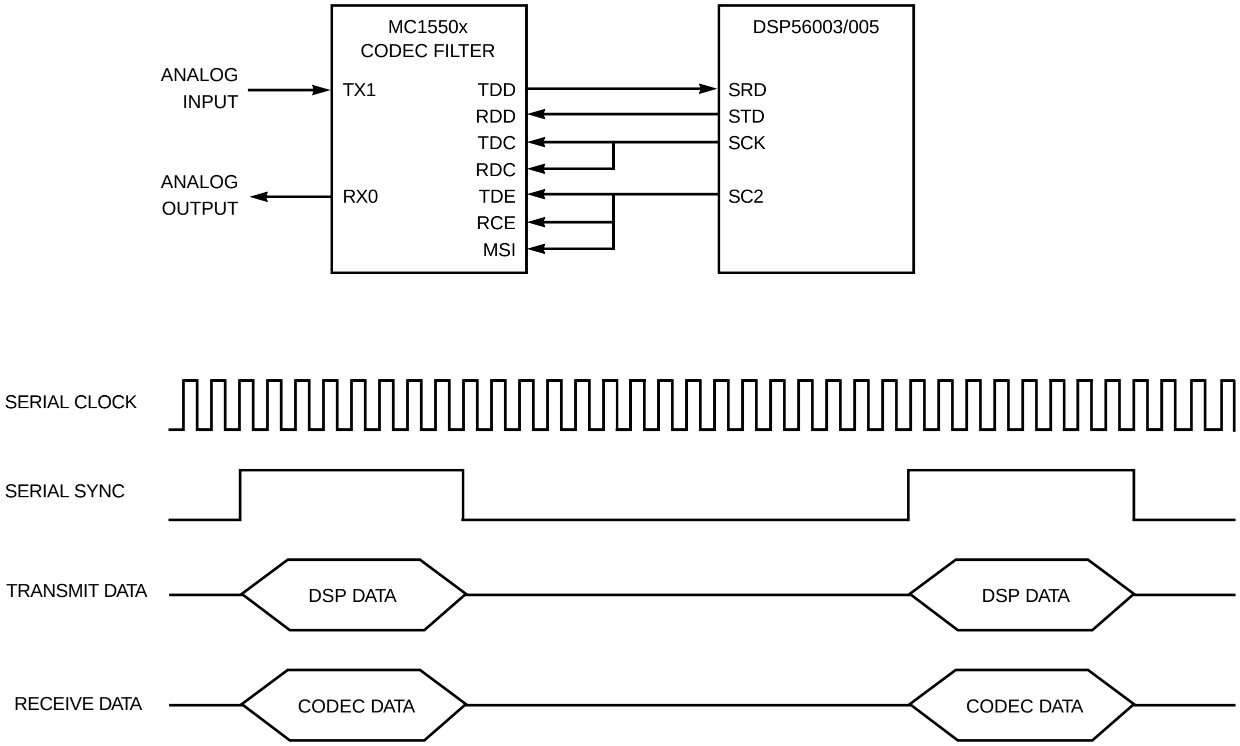


Figure 7-38 Normal Mode Example

TE. The state of the flag does not change for the entire frame.

Figure 6-39 is an example of transmitting data using the SSI in the normal mode with a continuous clock, a bit-length frame sync, and 16-bit data words. The purpose of the program is to interleave and transmit right and left channels in a compact disk player. Four SSI pins are used:

1. SC0 is used as an output flag to indicate right-channel data (OF0=1) or left-channel data (OF0=0)
2. SC2 is TX and RX frame sync out
3. STD is transmit data out
4. SCK clocks the transmit data out

Equates are set for convenience and readability. Test data is then put in the low X: memory locations. The transmit interrupt vector contains a JSR instruction (which forms a long interrupt). The data pointer and channel flag are initialized before initializing CRA and CRB. It is assumed that the DSP CPU and SSI have been previously reset.

At this point, the SSI is ready to transmit except that the interrupt is masked because the MR was cleared on reset and Port C is still configured as general-purpose I/O. Unmasking the interrupt and enabling the SSI pins allows transmission to begin. A “jump to self” instruction causes the DSP to hang and wait for interrupts to transmit the data. When an interrupt occurs, a JSR instruction at the interrupt vector location causes the XMT routine to be executed. Data is then moved to the TX register, and the data pointer is incremented. The flag is tested by the JSET instruction and, if it is set, a jump to left occurs, and the code for the left channel is executed. If the flag is not set, the code for the right channel is executed. In either case, the channel flag in X0 and then the output flag are set to reflect the channel being transmitted. Control is then returned to the main program, which will wait for the next interrupt.

```
;*************************************************
;    SSI and other I/O EQUATES*
;*************************************************
IPR     EQU     $FFFF
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
TX      EQU     $FFEF
FLG     EQU     $0010
        ORG     X:0
        DC      $AAAA00;Data to transmit.
        DC      $333300
        DC      $CCCC00
        DC      $F0F000
;*************************************************
;    INTERRUPT VECTOR*
;*************************************************
        ORG     P:$0010
        JSR     XMT
;*************************************************
;    MAIN PROGRAM*
;*************************************************
        ORG       P:$40
        MOVE      #0,R0          ;Pointer to data buffer.
        MOVE      #3,M0          ;Set modulus to 4.
        MOVE      #0,X0          ;Initialize channel flag for SSI flag.
        MOVE      X0,X:FLG       ;Start with right channel first.
;*************************************************
;    Initialize SSI Port*
;*************************************************
        MOVEP     #$3000,X:IPR ;Set interrupt priority register for SSI.
        MOVEP     #$401F,X:CRA   ;Set continuous clock=5.12/32 MHz
                                 ;word length=16.
        MOVEP     #$5334,X:CRB   ;Enable TIE and TE; make clock and
                                 ;frame sync outputs; frame
                                 ;sync=bit mode; synchronous mode;
                                 ;make SC0 an output.
```

Figure 7-39 Normal Mode Transmit Example (Sheet 1 of 2)

```
;************************************************
;    Init SSI Interrupt*
;************************************************
        ANDI    #$FC,MR        ;Unmask interrupts.
        MOVEP   #$01F8,X:PCC   ;Turn on SSI port.
        JMP     *              ;Wait for interrupt.
;************************************************
;    MAIN INTERRUPT ROUTINE*
;************************************************
XMT     MOVEP   X:(R0);pl,X:TX ;Move data to TX register.
        JSET    #0,X:FLG,LEFT  ;Check channel flag.
RIGHT   BCLR    #0,X:CRB  ;Clear SC0 indicating right channel data
        MOVE    #>$01,X0       ;Set channel flag to 1 for next data.
        MOVE    X0,X:FLG
        RTI
LEFT    BSET    #0,X:CRB  ;Set SC0 indicating left channel data.
        MOVE    #>$00,X0       ;Clear channel flag for next data.
        MOVE    X0,X:FLG
        RTI
        END
```

Figure 6-39 Normal Mode Transmit Example (Sheet 2 of 2)

7.3.7.2.2 Normal Mode Receive

If the receiver is enabled, a data word will be clocked in each time the frame sync signal is generated (internal) or detected (external). After receiving the data word, it will be transferred from the SSI receive shift register to the receive data register (RX), RDF will be set (receiver full), and the receive interrupt will occur if it is enabled (RIE=1).

The DSP program has to read the data from RX before a new data word is transferred from the receive shift register; otherwise, the receiver overrun error will be set (ROE=1).

Figure 7-40 illustrates the program that receives the data transmitted by the program shown in Figure 6-39. Using the flag to identify the channel, the receive program receives the right- and left-channel data and separates the data into a right data buffer and a left data buffer. The program shown in Figure 7-40 begins by setting equates and then using a JSR instruction at the receive interrupt vector location to form a long interrupt. The main program starts by initializing pointers to the right and left data buffers. The IPR, CRA, and CRB are then initialized. The clock divider bits in the CRA do not have to be set since an external receive clock is specified (SCKD=0). Pin SC0 is specified as an input flag (SYN=1, SCD0=0); pin SC2 is specified as TX and RX frame sync (SYN=1, SCD2=0).

The SSI port is then enabled and interrupts are unmasked, which allows the SSI port to begin data reception. A jump-to-self instruction is then used to hang the processor and allow interrupts to receive the data. Normally, the processor would execute useful instructions while waiting for the receive interrupts. When an interrupt occurs, the JSR instruction at the interrupt vector location transfers control to the RCV subroutine. The input flag is tested, and data is put in the left or right data buffer depending on the results of the test. The RTI instruction then returns control to the main program, which will wait for the next interrupt.

```
;**************************************************
;    SSI and other I/O EQUATES*
;**************************************************
IPR     EQU     $FFFF
SSISR   EQU     $FFEE
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
RX      EQU     $FFEF
FLG     EQU     $0010
;**************************************************
;    INTERRUPT VECTOR*
;**************************************************
        ORG     P:$000C
        JSR     RCV
;**************************************************
;    MAIN PROGRAM*
;**************************************************
        ORG     P:$40
        MOVE    #0,R0           ;Pointer to memory buffer for
        MOVE    #$08,R1         ;received data. Note data will be
        MOVE    #1,M0           ;split between two buffers which are
        MOVE    #1,M1           ;modulus 2.
```

Figure 7-40 Normal Mode Receive Example (Sheet 1 of 2)

```
;************************************************
;    Initialize SSI Port*
;************************************************
        MOVEP   #$3000,X:IPR    ;Set interrupt priority register
                                ;for SSI.
        MOVEP   #$4000,X:CRA    ;Set word length = 16 bits.
        MOVEP   #$A300,X:CRB    ;Enable RIE and RE; synchronous
                                ;mode with bit frame sync;
                                ;clock and frame sync are
                                ;external; SC0 is an output.
;************************************************
;    Init SSI Interrupt*
;************************************************
        ANDI    #$FC,MR              ;Unmask interrupts.
        MOVEP   #$01F8,X:PCC         ;Turn on SSI port.
        JMP     *                    ;Wait for interrupt.
;************************************************
;    MAIN INTERRUPT ROUTINE*
;************************************************
RCV     JSET    #0,X:SSISR, RIGHT    ;Test SCO flag.
LEFT    MOVEP   X:RX,X:(RO)+    ;If SCO clear, receive data
        RTI                     ;into left buffer (R0).
RIGHT   MOVEP   X:RX,X:(R1)+    ;If SCO set, receive data
        RTI                     ;into right buffer (R1).
        END
```

Figure 7-40 Normal Mode Receive Example (Sheet 2 of 2)

7.3.7.3 Network Mode Examples

The network mode, the typical mode in which the DSP would interface to a TDM codec network or a network of DSPs, is compatible with Bell and CCITT PCM data/operation formats. The DSP may be a master device (see Figure 7-41) that controls its own private network or a slave device that is connected to an existing TDM network, occupying one or more time slots. The key characteristic of the network mode is that each time slot (data word time) is identified by an interrupt or by polling status bits, which allows the option of ignoring the time slot or transmitting data during the time slot. The receiver operates in the same manner except that data is always being shifted into the receive shift register and transferred to the RX. The DSP reads the receive data register and uses or discards the contents. Overrun and underrun errors are detected.

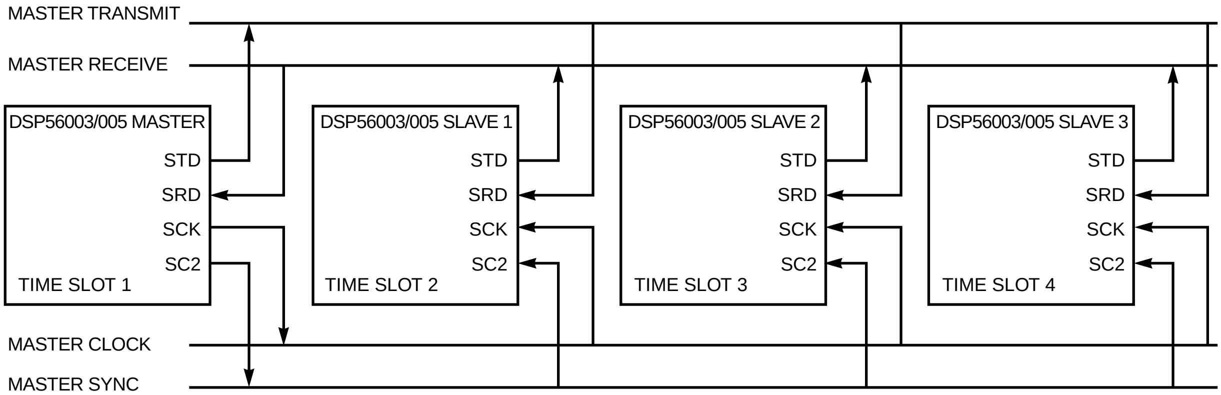


Figure 7-41 Network Mode Example

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots; transmission or reception can occur in each time slot (rather than in just the frame sync time slot as in normal mode). The frame rate dividers (controlled by DC4, DC3, DC2, DC1, and DC0) control the number of time slots per frame from 2 to 32. Time-slot assignment is totally under software control. Devices can transmit on multiple time slots, receive multiple time slots, and the time-slot assignment can be changed dynamically.

A simplified flowchart showing operation of the network mode is shown in Figure 7-42. Two counters are used to track the current transmit and receive time slots. Slot counter one (SLOTCT1) is used to track the transmit time slot; slot counter two (SLOTCT2) is used for receive. When the transmitter is empty, it generates an interrupt; a test is then made to see if it is the beginning of a frame. If it is the beginning of a frame, SLOTCT1 is cleared to start counting the time slots. If it is not the beginning of a frame, SLOTCT1 is incremented. The next test checks to see if the SSI should transmit during this time slot. If it is time to transmit, data is written to the TX; otherwise, dummy data is written to the TSR, which prevents a transmit underrun error from occurring and three-states the STD pin. The DSP can then return to what it was doing before the interrupt and wait for the next interrupt to occur. SLOTCT1 should reflect the data in the shift registers to coincide with TFS. Software must recognize that the data being written to TX will be transmitted in time slot SLOTCT1 plus one.

The receiver operates in a similar manner. When the receiver is full, an interrupt is generated, and a test is made to see if this is the beginning of a frame. If it is the beginning of a frame, SLOTCT2 is cleared to start counting the time slots. If it is not the beginning of a frame, SLOTCT2 is incremented. The next test checks to see if the data received is intended for this DSP. If the current time slot is the one assigned to the DSP receiver, the data is kept; otherwise, the data is discarded, and the DSP can then return to what it was doing before the interrupt.

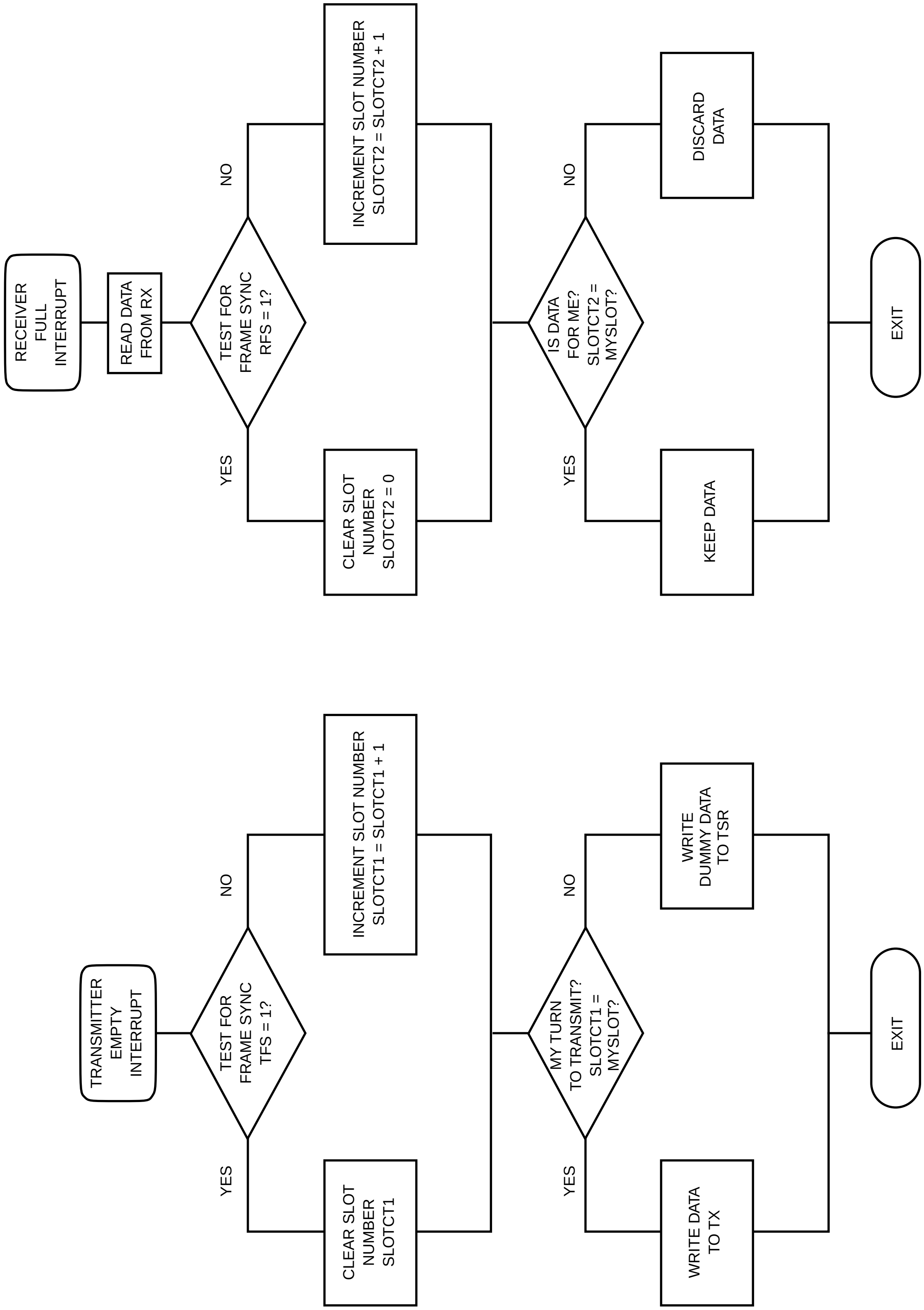

RECEIVER FULL INTERRUPT
READ DATA FROM RX
TEST FOR FRAME SYNC RFS = 1?
NO
YES
INCREMENT SLOT NUMBER SLOTCT2 = SLOTCT2 + 1
CLEAR SLOT NUMBER SLOTCT2 = 0
IS DATA FOR ME? SLOTCT2 = MYSLOT?
NO
YES
DISCARD DATA
KEEP DATA
EXIT
TRANSMITTER EMPTY INTERRUPT
TEST FOR FRAME SYNC TFS = 1?
NO
YES
INCREMENT SLOT NUMBER SLOTCT1 = SLOTCT1 + 1
CLEAR SLOT NUMBER SLOTCT1
MY TURN TO TRANSMIT? SLOTCT1 = MYSLOT?
NO
YES
WRITE DUMMY DATA TO TSR
WRITE DATA TO TX
EXIT

Figure 7-42 TDM Network Software Flowchart

SLOTCT2 should reflect the data in the receive shift register to coincide with the RFS flag. Software must recognize that the data being read from RX is for time slot SLOTCT2 minus two.

Initializing the network mode is accomplished by setting the bits in CRA and CRB as follows (see Figure 7-43):

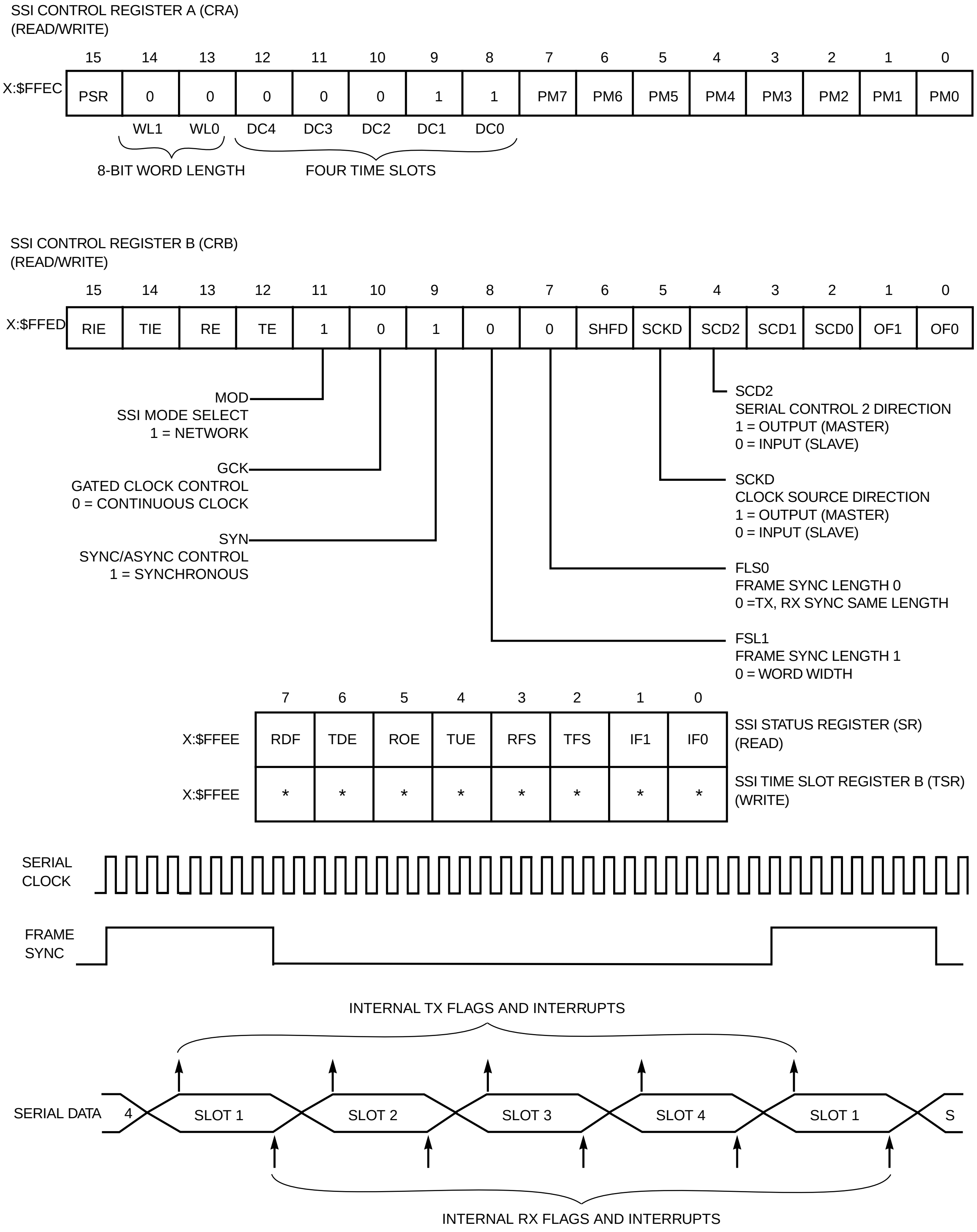
1. The word length must be selected by setting WL1 and WL0. In this example, an 8-bit word length was chosen (WL1=0 and WL0=0).
2. The number of time slots is selected by setting DC4–DC0. Four time slots were chosen for this example (DC4–DC0=$03).
3. The serial clock rate must be selected by setting PSR and PM7–PM0 (see Table 7-11 (a), Table 7-11 (b), and Table 7-12).
4. RE and TE must be set to activate the transmitter and receiver. If interrupts are to be used, RIE and TIE should be set. RIE and TIE are usually set after everything else is configured and the DSP is ready to receive interrupts.
5. The network mode must be selected (MOD=1).
6. A continuous clock is selected in this example by setting GCK=0.
7. Although it is not required for the network mode, synchronous clock control was selected (SYN=1).
8. The frame sync length was chosen in this example as word length (FSL1=0) for both transmit and receive frame sync (FSL0=0). Any other combinations could have been selected, depending on the application.
9. Control bits SHFD, SCKD, SCD2, SCD1, SCD0, and the flag bits (OF1 and OF0) should be set as needed for the application.

7.3.7.3.1 Network Mode Transmit

When TE is set, the transmitter will be enabled only after detection of a new data frame sync. This procedure allows the SSI to synchronize to the network timing.

Normal startup sequence for transmission in the first time slot is to write the data to be transmitted to TX, which clears the TDE flag. Then set TE and TIE to enable the transmitter on the next frame sync and to enable transmit interrupts.

Alternatively, the DSP programmer may decide not to transmit in the first time slot by writing any data to the time slot register (TSR). This will clear the TDE flag just as if data were going to be transmitted, but the STD pin will remain in the high-impedance state for the first time slot. The programmer then sets TE and TIE.



SSI CONTROL REGISTER A (CRA)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFEC	PSR	0	0	0	0	0	1	1	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
		WL1	WL0	DC4	DC3	DC2	DC1	DC0								

SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFED	RIE	TIE	RE	TE	1	0	1	0	0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0

	7	6	5	4	3	2	1	0	
X:$FFEE	RDF	TDE	ROE	TUE	RFS	TFS	IF1	IF0	SSI STATUS REGISTER (SR) (READ)
X:$FFEE	*	*	*	*	*	*	*	*	SSI TIME SLOT REGISTER B (TSR) (WRITE)

SERIAL
CLOCK
FRAME
SYNC
INTERNAL TX FLAGS AND INTERRUPTS
SERIAL DATA
4
SLOT 1
SLOT 2
SLOT 3
SLOT 4
SLOT 1
S
INTERNAL RX FLAGS AND INTERRUPTS

Figure 7-43 Network Mode Initialization

When the frame sync is detected (or generated), the first data word will be transferred from TX to the transmit shift register and will be shifted out (transmitted). TX being empty will cause TDE to be set, which will cause a transmitter interrupt. Software can poll TDE or use interrupts to reload the TX register with new data for the next time slot. Software can also write to TSR to prevent transmitting in the next time slot. Failing to reload TX (or writing to the TSR) before the transmit shift register is finished shifting (empty) will cause a transmitter underrun. The TUE error bit will be set, causing the previous data to be retransmitted.

The operation of clearing TE and setting it again will disable the transmitter after completion of transmission of the current data word until the beginning of the next frame sync period. During that time, the STD pin will be three-stated. When it is time to disable the transmitter, TE should be cleared after TDE is set to ensure that all pending data is transmitted.

The optional output flags are updated every time slot regardless of TE.

To summarize, the network mode transmitter generates interrupts every time slot and requires the DSP program to respond to each time slot. These responses can be:

1. Write data register with data to enable transmission in the next time slot
2. Write the time slot register to disable transmission in the next time slot
3. Do nothing – transmit underrun will occur the at beginning of the next time slot, and the previous data will be transmitted

Figure 7-44 differs from the program shown in Figure 6-39 only in that it uses the network mode to transmit only right-channel data. A time slot is assigned for the left-channel data, which could be inserted by another DSP using the network mode. In the “Initialize SSI Port” section of the program, two words per frame are selected using CRA, and the network mode is selected by setting MOD to one in the CRB. The main interrupt routine, which waits to move the data to TX, only transmits data if the current time slot is for the right channel. If the current time slot is for the left channel, the TSR is written, which three-states the output to allow another DSP to transmit the left channel during the time slot.

```
;*************************************************
;    SSI and other I/O EQUATES*
;*************************************************
IPR     EQU     $FFFF
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
TX      EQU     $FFEF
TSR     EQU     $FFEE
FLG     EQU     $0010
        ORG     X:0
        DC      $AAAA00        ;Data to transmit.
        DC      $333300
        DC      $CCCC00
        DC      $F0F000
;*************************************************
;    INTERRUPT VECTOR*
;*************************************************
        ORG     P:$0010
        JSR     XMT
;*************************************************
;    MAIN PROGRAM*
;*************************************************
        ORG     P:$40
        MOVE    #0,R0          ;Pointer to data buffer.
        MOVE    #3,M0          ;Set modulus to 4.
        MOVE    #0,X0          ;Initialize user flag for SSI flag.
        MOVE    X0,X:FLG       ;Start with the right channel.
```

Figure 7-44 Network Mode Transmit Example Program (Sheet 1 of 2)

```
;*************************************************
;    Initialize SSI Port*
;*************************************************
        MOVEP   #$3000,X:IPR   ;Set interrupt priority register
                               ;for SSI.
        MOVEP   #$411F,X:CRA   ;Set continuous clock=5.12/32 MHz
                               ;word length=16.
        MOVEP   #$5B34,X:CRB   ;Enable TIE and TE; make clock and
                               ;frame sync outputs; frame
                               ;sync=bit mode; synchronous mode;
                               ;make SC0 an output.
;*************************************************
;    Init SSI Interrupt*
;*************************************************
        ANDI    #$FC,MR        ;Unmask interrupts.
        MOVEP   #$01F8,X:PCC   ;Turn on SSI port.
        JMP     *              ;Wait for interrupt.
;*************************************************
;    MAIN INTERRUPT ROUTINE*
;*************************************************
XMT
        JSET    #0,X:FLG,LEFT  ;Check user flag.
RIGHT   BCLR    #0,X:CRB  ;Clear SC0 indicating right channel data
        MOVEP   X:(R0)+,X:TX   Move data to TX register.
        MOVE    #>$01,X0       ;Set user flag to 1
        MOVE    X0,X:FLG       ;for next data.
        RTI
LEFT    BSET    #0,X:CRB    ;Set SC0 indicating left channel data.
        MOVEP   X0,X:TSR       ;Write to TSR register.
        MOVE    #>$00,X0       ;Clear user flag
        MOVE    X0,X:FLG          ;for next data.
        RTI
        END
```

Figure 7-44 Network Mode Transmit Example Program (Sheet 2 of 2)

```
;*************************************************
;    SSI and other I/O EQUATES*
;*************************************************
IPR    EQU     $FFFF
SSISR  EQU     $FFEE
CRA    EQU     $FFEC
CRB    EQU     $FFED
PCC    EQU     $FFE1
RX     EQU     $FFEF
;*************************************************
;    INTERRUPT VECTOR*
;*************************************************
       ORG     P:$000C
       JSR     RCV
;*************************************************
;    MAIN PROGRAM*
;*************************************************
       ORG     P:$40
       MOVE    #0,R0          ;Pointer to memory buffer for
       MOVE    #$08,R1        ;received data. Note data will be
       MOVE    #3,M0          ;split between two buffers which are
       MOVE    #3,M1          ;modulus 4.
;*************************************************
;    Initialize SSI Port*
;*************************************************
       MOVEP   #$3000,X:IPR   ;Set interrupt priority register
                              ;for SSI.
       MOVEP   #$4100,X:CRA   ;Set word length = 16 bits.
       MOVEP   #$AB00,X:CRB   ;Enable RIE and RE; synchronous
                              ;mode with bit frame sync;
                              ;clock and frame sync are
                              ;external; SC0 is an input.
```

Figure 7-45 Network Mode Receive Example Program (Sheet 1 of 2)

```
;*************************************************
;    Init SSI Interrupt*
;*************************************************
        ANDI    #$FC,MR         ;Unmask interrupts.
        MOVEP   #$01F8,X:PCC    ;Turn on SSI port.
        JMP     *               ;Wait for interrupt.
;*************************************************
;    MAIN INTERRUPT ROUTINE*
;*************************************************
RCV     JSET    #0,X:SSISR, RIGHT;Test SCO flag.
LEFT    MOVEP   X:RX,X:(RO)+    ;If SCO clear, receive data
        RTI                     ;into left buffer (R0).
RIGHT   MOVEP   X:RX,X:(R1)+    ;If SCO set, receive data
        RTI                     ;into right buffer (R1).
        END
```

Figure 7-45 Network Mode Receive Example Program (Sheet 2 of 2)

7.3.7.3.2 Network Mode Receive

The receive enable will occur only after detection of a new data frame with RE set. The first data word is shifted into the receive shift register and is transferred to the RX, which sets RDF if a frame sync was received (i.e., this is the start of a new frame). Setting RDF will cause a receive interrupt to occur if the receiver interrupt is enabled (RIE=1).

The second data word (second time slot in the frame) begins shifting in immediately after the transfer of the first data word to the RX. The DSP program has to read the data from RX (which clears RDF) before the second data word is completely received (ready to transfer to RX), or a receive overrun error will occur (ROE=1), and the data in the receiver shift register will not be transferred and will be lost.

If RE is cleared and set again by the DSP program, the receiver will be disabled after receiving the current time slot in progress until the next frame sync (first time slot). This mechanism allows the DSP programmer to ignore data in the last portion of a data frame.

Note: The optional frame sync output and clock output signals are not affected, even if the transmitter and/or receiver are disabled. TE and RE do not disable bit clock and frame sync generation.

To summarize, the network mode receiver receives every time slot data word unless the receiver is disabled. An interrupt can occur after the reception of each data word, or the programmer can poll RDF. The DSP program response can be

1. Read RX and use the data
2. Read RX and ignore the data
3. Do nothing – the receiver overrun exception will occur at the end of the current time slot
4. Toggle RE to disable the receiver until the next frame, and read RX to clear RDF

Figure 7-45 is essentially the same program shown in Figure 7-40 except that this program uses the network mode to receive only right-channel data. In the “Initialize SSI Port” section of the program, two words per frame are selected using the DC bits in the CRA, and the network mode is selected by setting MOD to one in the CRB. If the program in Figure 7-44 is used to transmit to the program in Figure 7-45, the correct data will appear in the data buffer for the right channel, but the buffer for the left channel will probably contain $000000 or $FFFFFF, depending on whether the transmitter output was high or low when TSR was written and whether the output was three-stated.

7.3.7.4 On-Demand Mode Examples

A divide ratio of one (DC=00000) in the network mode is defined as the on-demand mode of the SSI because it is the only data-driven mode of the SSI – i.e., data is transferred whenever data is present (see Figure 7-46 and Figure 7-47). STD and SCK from DSP1 are connected to DSP2 – SRD and SC0, respectively. SC0 is used as an input clock pin in this application. Receive data and receive data clock are separate from the transmit signals. On-demand data transfers are nonperiodic, and no time slots are defined. When there is a clock in the gated clock mode, data is transferred. Although they are not necessarily needed, frame sync and flags are generated when data is transferred. Transmitter underruns (TUE) are impossible in this mode and are therefore disabled. In the on-demand transmit mode, two additional SSI clock cycles are automatically inserted between each data word transmitted. This procedure guarantees that frame sync will be low between every transmitted data word or that the clock will not be continuous between two consecutive words in the gated clock mode. The on-demand mode is similar to the SCI shift register mode with SSFTD equals one and SCKP equals one. The receiver should be configured to receive the bit clock and, if continuous clock is used, to receive an external frame sync. Therefore, for all full-duplex communication in on-demand mode, the asynchronous mode should be used. The on-demand mode is SPI compatible.

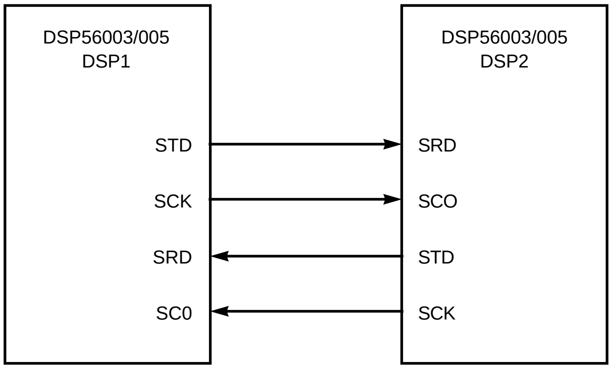
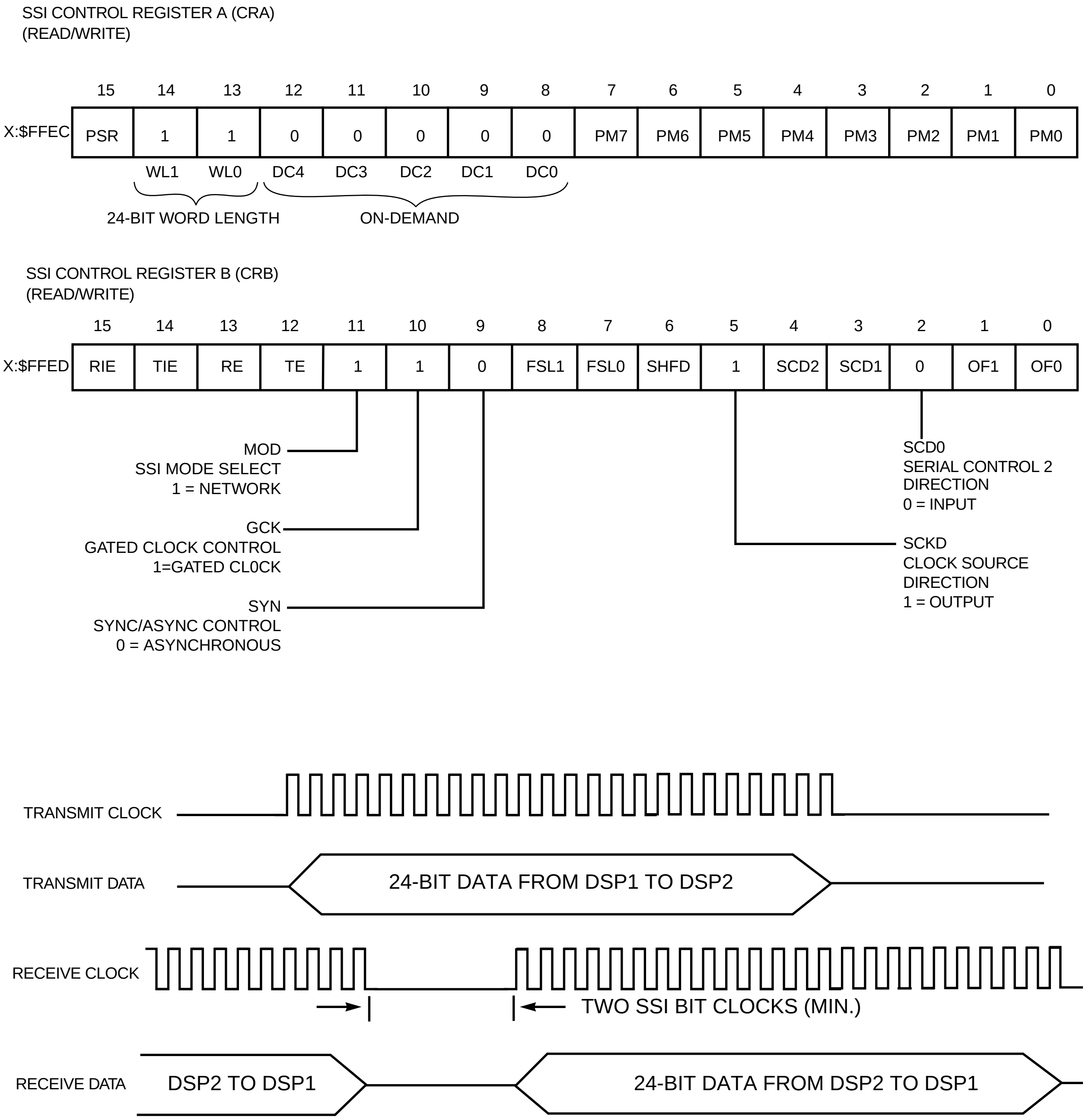


Figure 7-46 On Demand Example

Initializing the on-demand mode for the example illustrated in Figure 7-47 is accomplished by setting the bits in CRA and CRB as follows:

1. The word length must be selected by setting WL1 and WL0. In this example, a 24-bit word length was chosen (WL1=1 and WL0=1).
2. The on-demand mode is selected by clearing DC4–DC0.
3. The serial clock rate must be selected by setting PSR and PM7–PM0 (see Table 7-11 (a), Table 7-11 (b), and Table 7-12).
4. RE and TE must be set to activate the transmitter and receiver. If interrupts are to be used, RIE and TIE should be set. RIE and TIE are usually set after everything else is configured and the DSP is ready to receive interrupts.
5. The network mode must be selected (MOD=1).
6. A gated clock (GCK=1) is selected in this example. A continuous clock example is shown in Figure 7-44.
7. Asynchronous clock control was selected (SYN=0) in this example.
8. Since gated clock is used, the frame sync is not necessary. FSL1 and FSL0 can be ignored.
9. SCKD must be an output (SCKD=1).
10. SCD0 must be an input (SCD0=0).
11. Control bit SHFD should be set as needed for the application. Pins SC1 and SC2 are undefined in this mode (see Table 7-8) and should be programmed as general-purpose I/O pins.



NOTE: Two SSI bit clock times are automatically inserted between each data word. This guarantees frame sync will be low between every data word transmitted and the clock will not be continuous for two consecutive data words.

Figure 7-47 On-Demand Data-Driven Network Mode

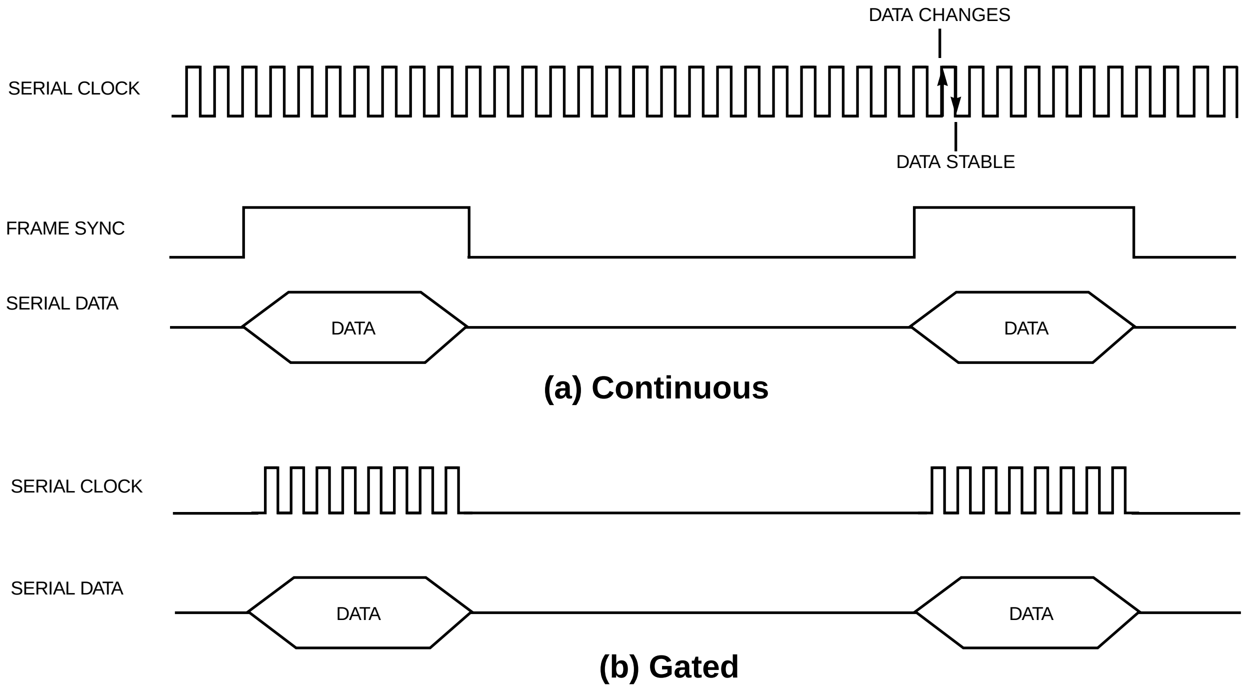


Figure 7-48 Clock Modes

7.3.7.4.1 On-Demand Mode – Continuous Clock

This special case will not generate a periodic frame sync. A frame sync pulse will be generated only when data is available to transmit (see Figure 7-48(a)). The frame sync signal indicates the first time slot in the frame. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). Therefore, for simplex operation, the synchronous mode could be used; however, for full-duplex operation, the asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into TX. Although the SSI is double buffered, only one word can be written to TX, even if the transmit shift register is empty. The receive and transmit interrupts function as usual using TDE and RDF; however, transmit and receive underruns are impossible for on-demand transmission and are disabled. This mode is useful for interfacing to codecs requiring a continuous clock.

7.3.7.4.2 On-Demand Mode – Gated Clock

Gated clock mode (see Figure 7-48(b)) is defined for on-demand mode, but the gated clock mode is considered a frame sync source; therefore, in gated clock mode, the transmit clock must be internal (output) and the receive clock must be external (input). For ondemand mode, with internal (output) synchronous gated clock, output clock is enabled for the transmitter and receiver when TX data is transferred to the transmit data shift register.

This SPI master operating mode is shown in Figure 7-49. Word sync is inherent in the clock signal, and the operation format must provide frame synchronization.

Figure 7-50 is the block diagram for the program presented in Figure 7-51. This program contains a transmit test program that was written as a scoping loop (providing a repetitive sync) using the on-demand, gated, synchronous mode with no interrupts (polling) to transmit data to the program shown in Figure 7-52. The program also demonstrates using GPIO pins as general-purpose control lines. PC3 is used as an external strobe or enable for hardware such as an A/D converter.

The transmit program sets equates for convenience and readability. Test data is then written to X: memory, and the data pointer is initialized. Setting M0 to two makes the buffer circular (modulo 3), which saves the step of resetting the pointer each loop. PC3 is configured as a general-purpose output for use as a scope sync, and CRA and CRB are then initialized. Setting the PCC bits begins SSI operation; however, no data will be transmitted until data is written to TX. PC3 is set high at the beginning of data transmission; data is then moved to TX to begin transmission. A JCLR instruction is then used to form a wait loop until TDE equals one and the SSI is ready for another data word to be transmitted. Two more data words are transmitted in this fashion (this is an arbitrary number chosen for this test loop). An additional wait is included to make sure that the frame sync has gone low before PC3 is cleared, indicating on the scope that transmission is complete. A wait of 100 NOPs is implemented by using the REP instruction before starting the loop again.

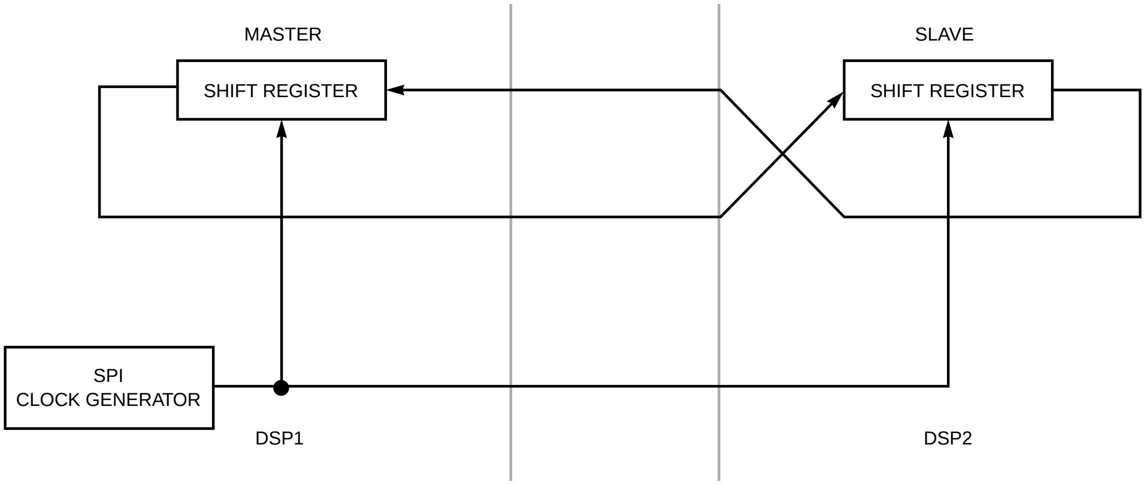


Figure 7-49 SPI Configuration

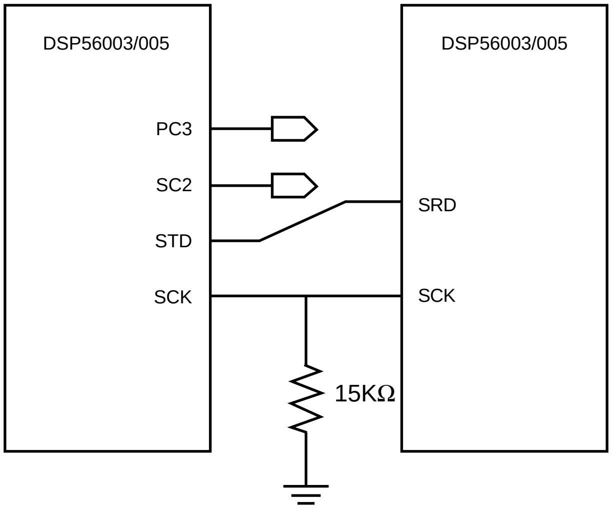


Figure 7-50 On-Demand Mode Example — Hardware Configuration

```
;*************************************************
;     SSI and other I/O EQUATES*
;*************************************************
CRA       EQU      $FFEC
CRB       EQU      $FFED
PCC       EQU      $FFE1
PCD       EQU      $FFE5
SSISR     EQU      $FFEE
TX        EQU      $FFEF
PCDDR     EQU      $FFE3
          ORG      X:0
          DC       $AA0000          ;Data to transmit.
          DC       $330000
          DC       $F00000
;*************************************************
;     MAIN PROGRAM*
;*************************************************
          ORG      P:$40
          MOVE     #0,R0            ;Pointer to data buffer
          MOVE     #2,M0            ;Length off buffer is 3
```

Figure 7-51 On-Demand Mode Transmit Example Program (Sheet 1 of 2)

```
        MOVEP  #$08,X:PCDDR    ;SC0 (PC3) as general purpose output.
        MOVEP  #$001F,X:CRA    ;Set Word Length=8, CLK=5.12/32 MHz
        MOVEP  #$1E30,X:CRB   ;Enable transmitter, Mode=On- Demand,
                               ;Gated clock on, synchronous mode,
                               ;Word frame sync selected, frame
                               ;sync and clock are internal and
                               ;output to port pins.
        MOVEP  #$1F0,X:PCC     ;Set PCC for SSI and
LOOP0   BSET   #3,X:PCD        ;Set PC3 high (this is example enable
                               ;or strobe for an external device
                               :such as an ADC).
        MOVEP  X:(R0);pl,X:TX  ;Move data to TX register
TDE1    JCLR   #6,X:SSISR,TDE1 ;Wait for TDE (transmit data register
                               ;empty) to go high.
        MOVEP  X:(R0);pl,X:TX  ;Move next data to TX.
TDE2    JCLR   #6,X:SSISR,TDE2 ;Wait for TDE to go high.
        MOVEP  X:(R0);pl,X:TX  ;Move data to TX.
TDE3    JCLR   #6,X:SSISR,TDE3 ;Wait for TDE=1.
FSC     JSET   #5,X:PCD,FSC    ;Wait for frame sync to go low. NOTE:
                               ;State of frame sync is directly
                               ;determined by reading PC5.
        BCLR   #3,X:PCD       ;Set PC3 lo (example external enable).
;anything goes here (i.e., any processing)
        REP    #100
        NOP
        JMP    LOOP0           ;Continue sequence forever.
        END
```

Figure 7-51 On-Demand Mode Transmit Example Program (Sheet 2 of 2)

Figure 7-52 is the receive program for the scoping loop program presented in Figure 7-51. The receive program also uses the on-demand, gated, synchronous mode with no interrupts (polling). Initialization for the receiver is slightly different than for the transmitter. In CRB, RE is set rather than TE, and SCKD and SCD2 are inputs rather than outputs. After initialization, a JCLR instruction is used to wait for a data word to be received (RDF=1). When a word is received, it is put into the circular buffer and loops to wait for another data word. The data in the circular buffer will be overwritten after three words are received (does not matter in this application).

```
;*************************************************
;    SSI and other I/O EQUATES*
;*************************************************
CRA     EQU    $FFEC
CRB     EQU    $FFED
PCC     EQU    $FFE1
PCD     EQU    $FFE5
SSISR   EQU    $FFEE
RX      EQU    $FFEF
PCDDR   EQU    $FFE3
;*************************************************
;    MAIN PROGRAM*
;*************************************************
        ORG    P:$40
        MOVE   #0,R0          ;Pointer to data buffer
        MOVE   #2,M0          ;Length of buffer is 3
        MOVEP  #$001F,X:CRA   ;Set Word Length=8, CLK=5.12/32 MHz
        MOVEP  #$1E30,X:CRB   ;Enable receiver, Mode=On-Demand,
                              ;gated clock on, synchronous mode,
                              ;Word frame sync selected, frame
                              ;sync and clock are external.
        MOVEP  #$1F0,X:PCC    ;Set PCC for SSI
LOOP
RDF1    JCLR   #7,X:SSISR,RDF1;Wait for RDF (receive data register
                              ;Full) go to high.
        MOVEP  X:RX,X:(R0)+   ;Read data from RX into memory.
        JMP    LOOP           ;Continue sequence forever.
        END
```

Figure 7-52 On-Demand Mode Receive Example Program

7.3.8 Flags

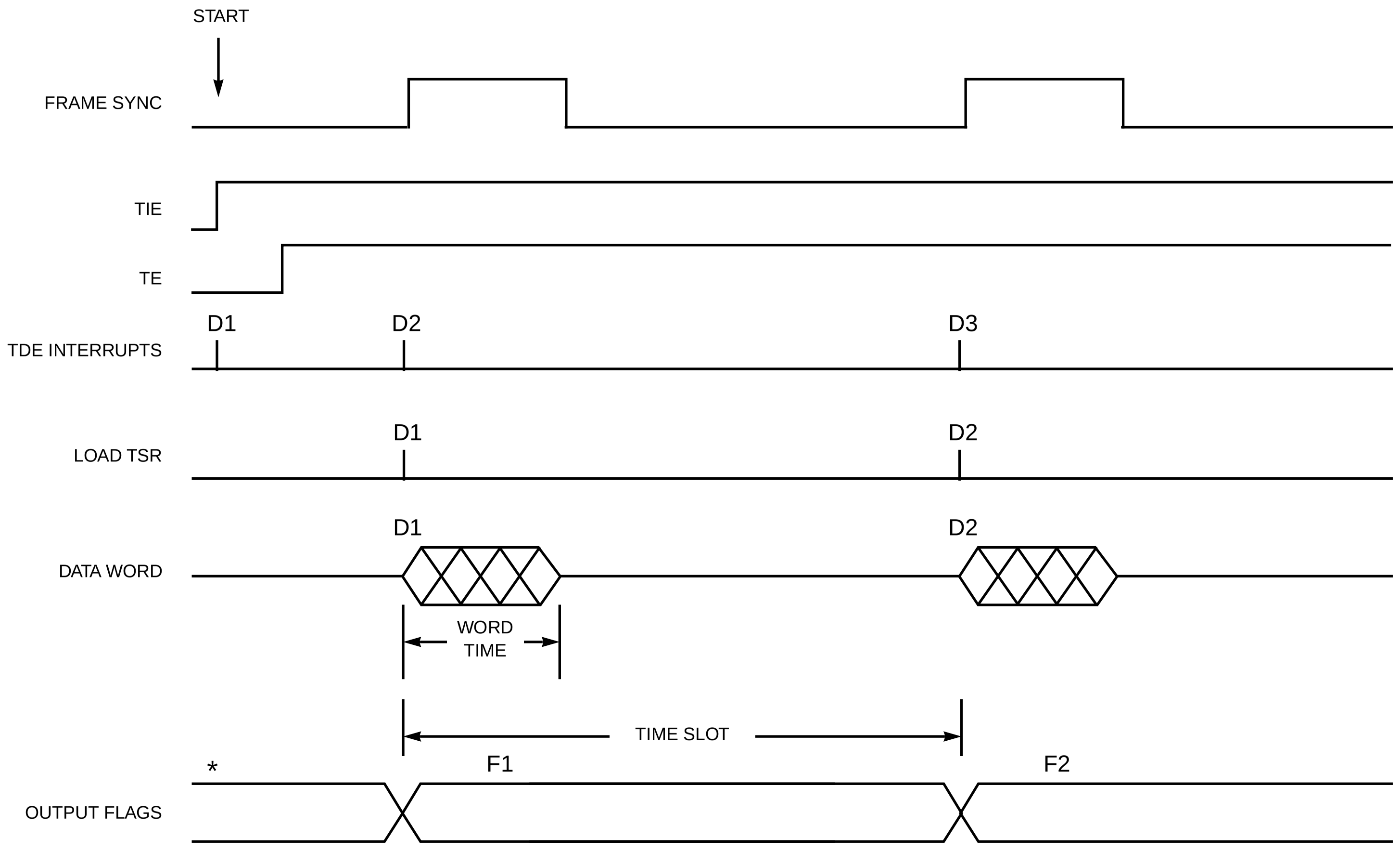
Two SSI pins (SC1 and SC0) are available in the synchronous mode for use as serial I/O flags. The control bits (OF1 and OF0) and status bits (IF1 and IF0) are double buffered to/from SC1 and SC0. Double buffering the flags keeps them in sync with TX and RX. The direction of SC1 and SC0 is controlled by SCD1 and SCD0 in CRB.

Figure 7-53 shows the flag timing for a network mode example. Initially, neither TIE nor TE is set, and the flag outputs are the last flag output value. When TIE is set, a TDE interrupt occurs (the transmitter does not have to be enabled for this interrupt to occur). Data (D1) is written to TX, which clears TDE, and the transmitter is enabled by software. When the frame sync occurs, data (D1) is transferred to the transmit shift register, setting TDE. Data (D1) is shifted out during the first word time, and the output flags are updated. These flags will remain stable until the next frame sync. The TDE interrupt is then serviced by writing data (D2) to TX, clearing TDE. After the TSR completes transmission, the transmit pin is three-stated until the next frame sync

Figure 7-54 shows a speaker phone example that uses a DSP56003/005 and two codecs. No additional logic is required to connect the codecs to the DSP. The two serial output flags in this example (OF1 and OF0) are used as chip selects to enable the appropriate codec for I/O. This procedure allows the transmit lines to be ORed together. The appropriate output flag pin changes at the same time as the first bit of the transmit word and remains stable until the next transmit word (see Figure 7-55). Applications include serial-device chip selects, implementing multidrop protocols, generating Bell PCM signaling frame syncs, and outputting status information.

Initializing the flags (see Figure 7-55) is accomplished by setting SYN, SCD1, and SCD0. No other control bits affect the flags. The synchronous control bit must be set (SYN=1) to select the SC1 and SC0 pins as flags. SCD1 and SCD0 select whether SC1 and SC0 are inputs or outputs (input=0, output=1). The other bits selected in Figure 7-55 are chosen for the speaker phone example in Figure 7-54. In this example, the codecs require that the SSI be set for normal mode (MOD=0) with a gated clock (GCK=1) out (SCKD=1).

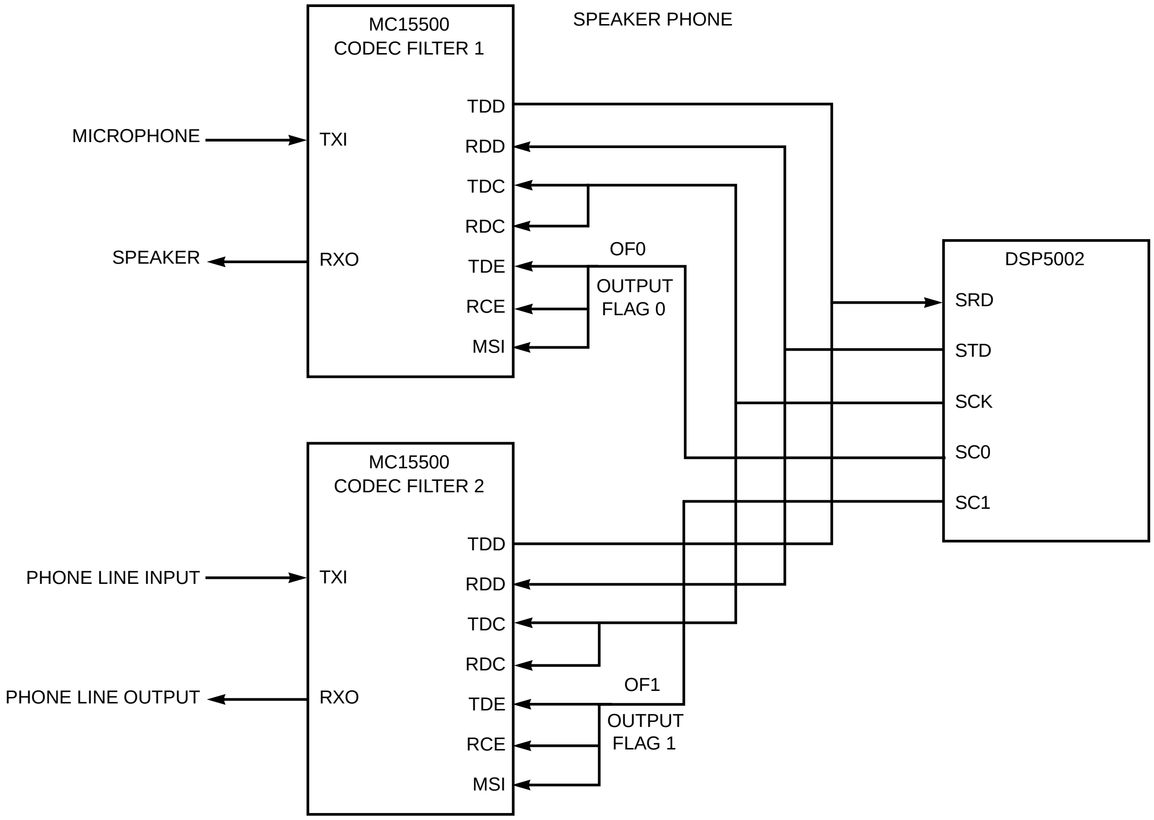
Serial input flags, IF1 and IF0, are latched at the same time as the first bit is sampled in the receive data word (see Figure 7-56). Since the input was latched, the signal on the input flag pin can change without affecting the input flag until the first bit of the next receive data word. To initialize SC1 or SC0 as input flags, the synchronous control bit in CRB must be set to one (SYN=1) and SCD1 set to zero for pin SC1, and SCD0 must be set to zero for pin SC0. The input flags are bits 1 and 0 in the SSISR (at X:$FFEE).



NOTES:

1. Fn = flags associated with Dn data.
2. Output flags are double buffered with transmit data.
3. Output flags change when data is transferred from TX to the transmit data shift register.
4. Initial flag outputs (*) = last flag output value.
5. Data and flags transition after external frame sync but not before rising edge of clock.

Figure 7-53 Output Flag Timing



NOTE: SC0 and SC1 are output flag 0 and 1 used to software select either filter 1 or 2.

Figure 7-54 Output Flag Example

7.3.9 Example Circuits

The DSP-to-DSP serial network shown in Figure 7-57 uses no additional logic chips for the network connection. All serial data is synchronized to the data source (all serial clocks and serial syncs are common). This basic configuration is useful for decimation and data reduction when more processing power is needed than one DSP can provide. Cascading DSPs in this manner is useful in several network topologies including star and ring networks.

TDM networks are useful to reduce the wiring needed for connecting multiple processors. A TDM parallel topology, such as the one shown in Figure 7-58, is useful for interpolating filters. Serial data can be received simultaneously by all DSPs, processing can occur in parallel, and the results are then multiplexed to a single serial data out line. This configuration can be cascaded and/or looped back on itself as needed to fit a particular application (see Figure 7-59). The serial and parallel configurations can be combined to form the array processor shown in Figure 7-60. A nearest neighbor array, which is applicable to matrix relaxation processing, is shown in Figure 7-61.

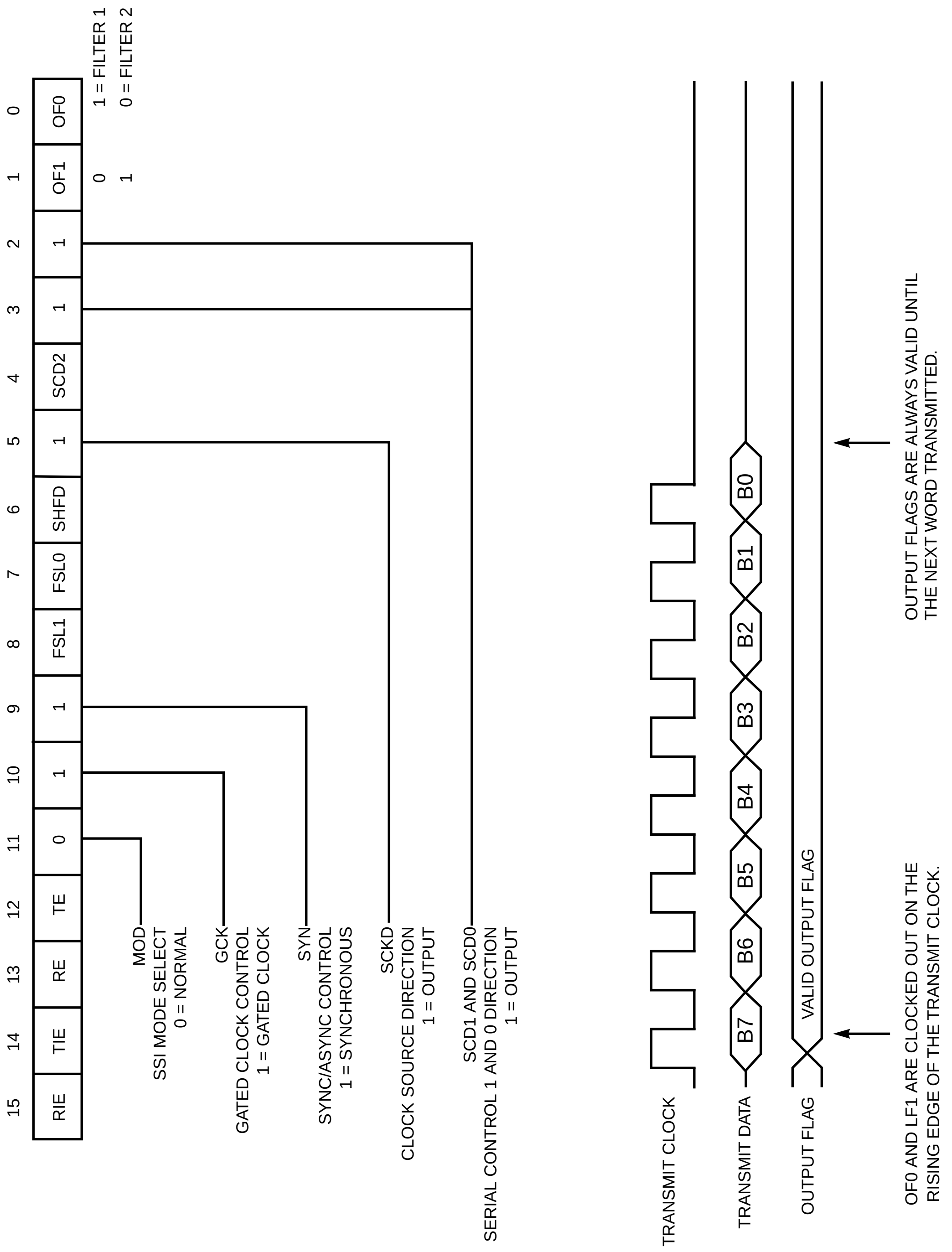


Figure 7-55 Output Flag Initialization

To simplify the drawing, only the center DSP is connected in this illustration. In use,

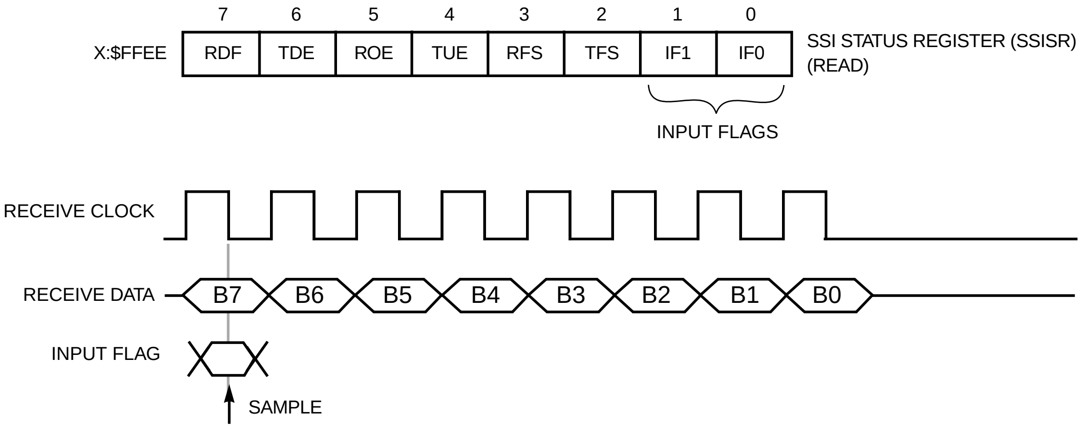


Figure 7-56 Input Flags

all DSPs would have four three-state buffers connected to their STD pin. The flags

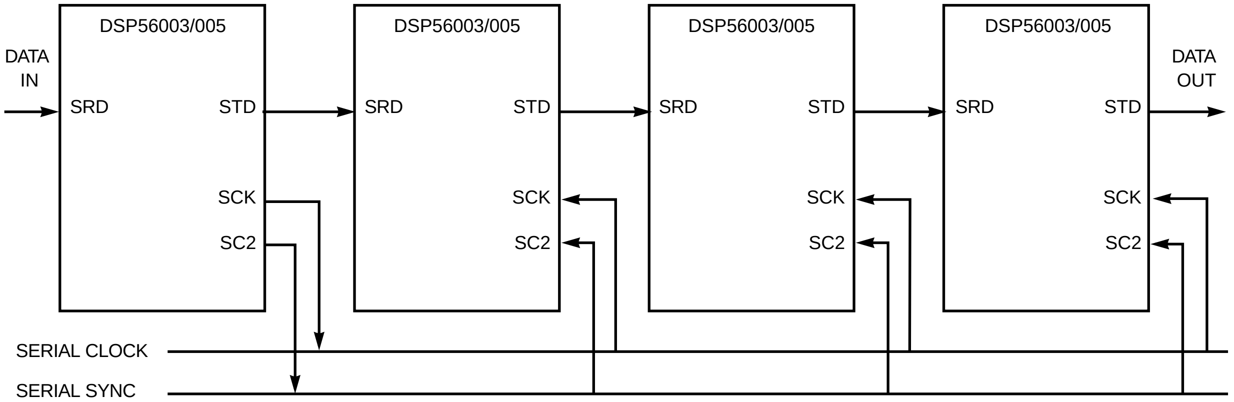


Figure 7-57 SSI Cascaded Multi-DSP System

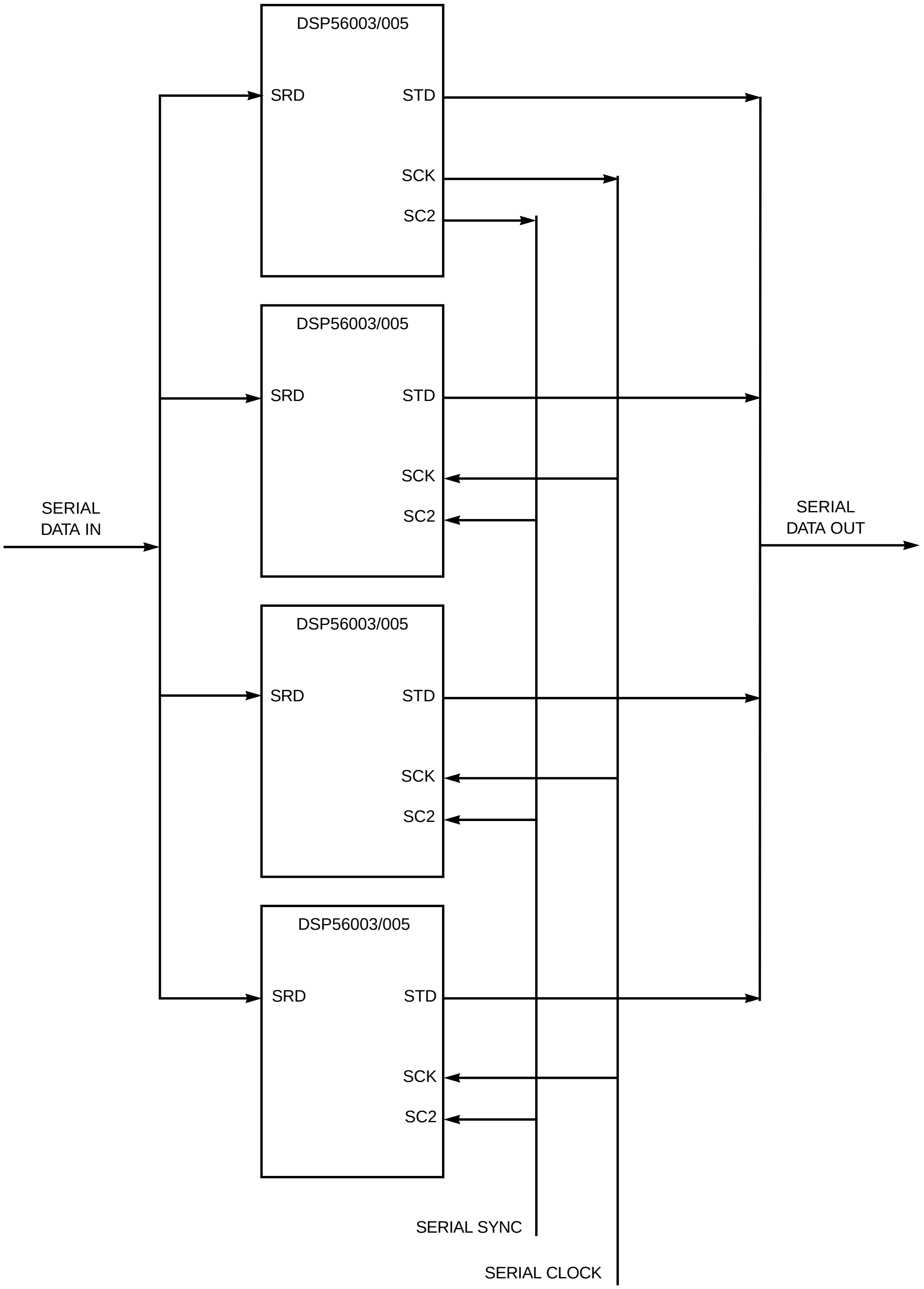


Figure 7-58 SSI TDM Parallel DSP Network

(SC0 and SC1) on the control master operate the three-state buffers, which control the

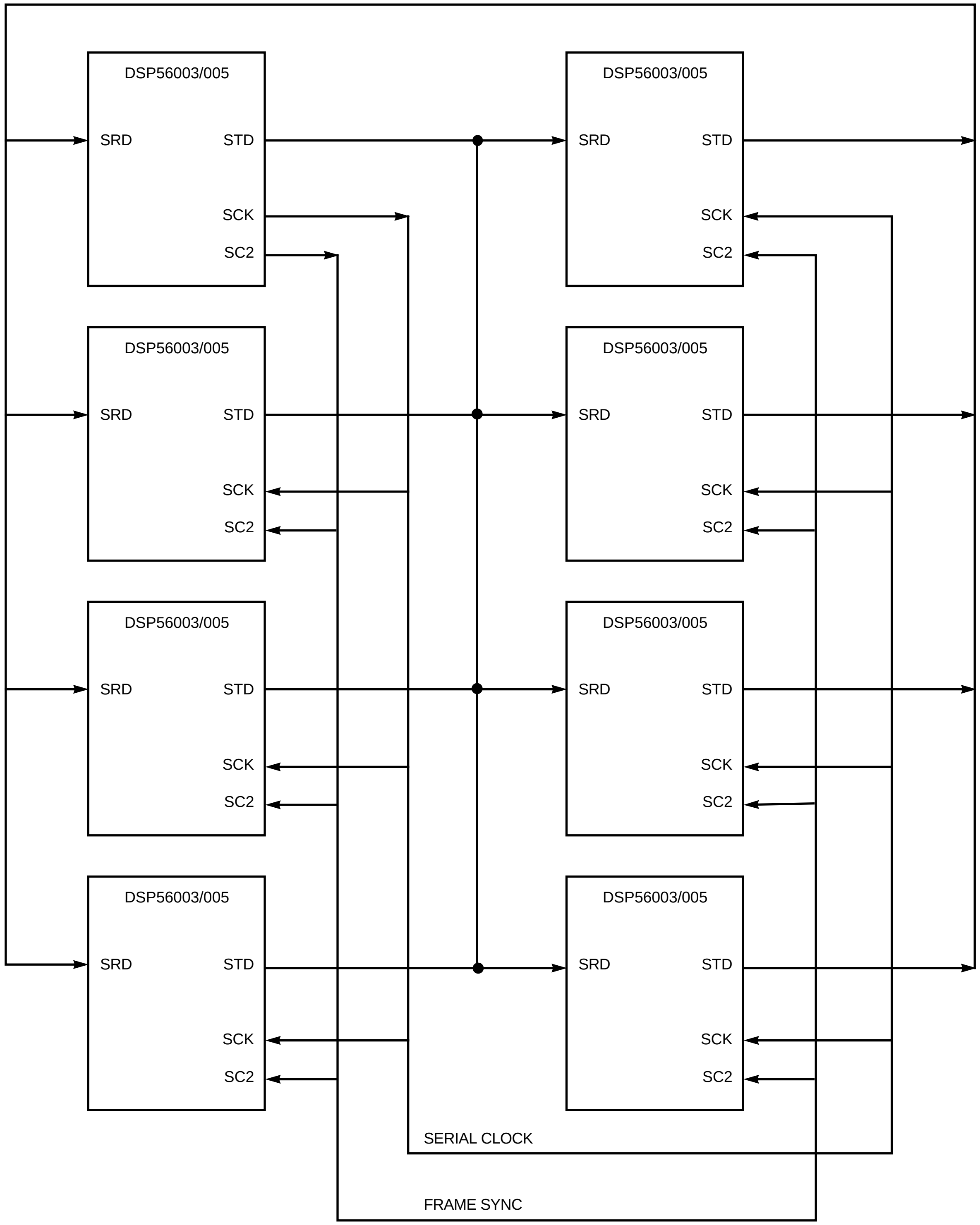


Figure 7-59 SSI TDM Connected Parallel Processing Array

direction that data is transferred in the matrix (north, south, east, or west).

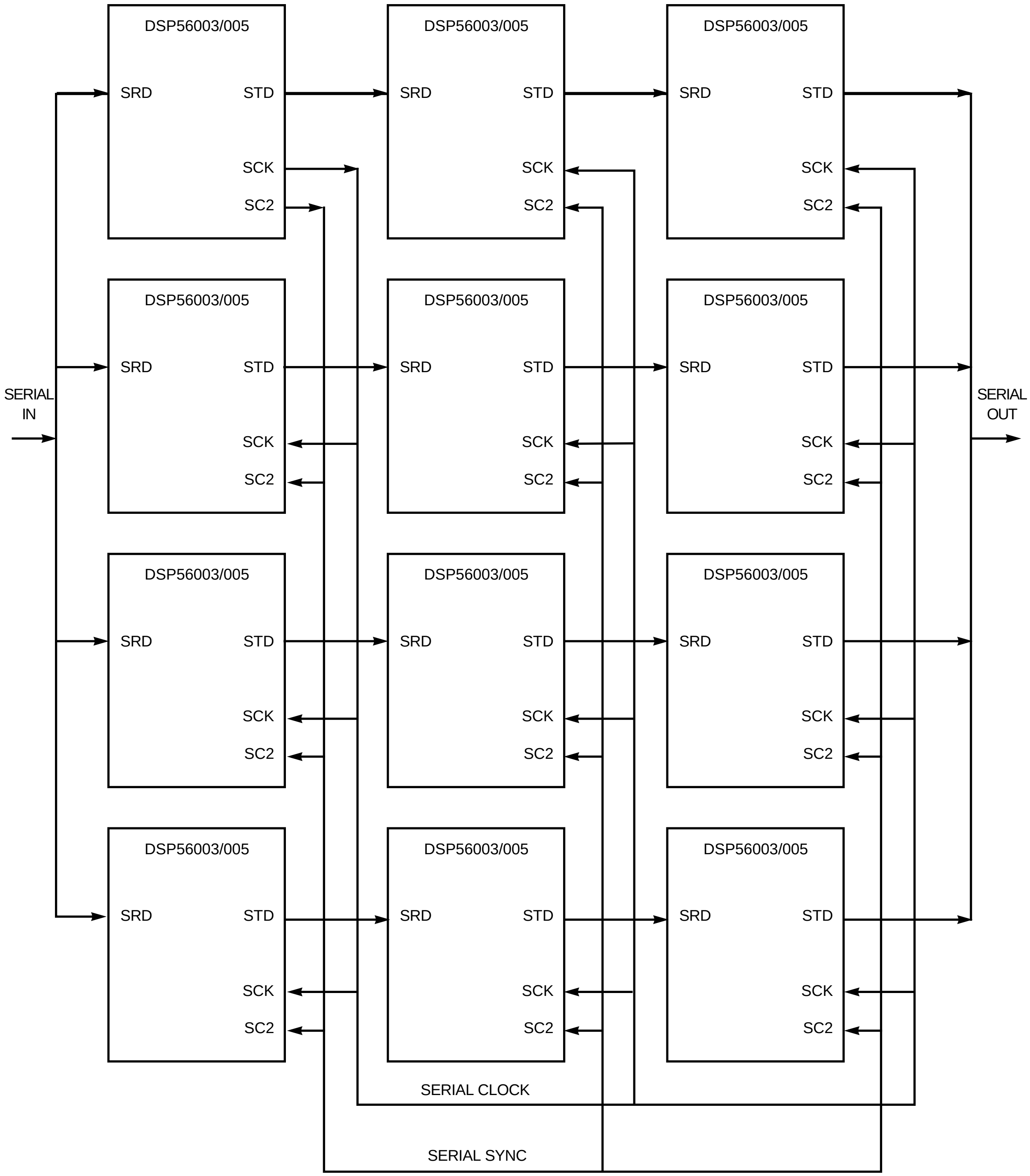


Figure 7-60 SSI TDM Serial/Parallel Processing Array

The bus architecture shown in Figure 7-62 allows data to be transferred between any two

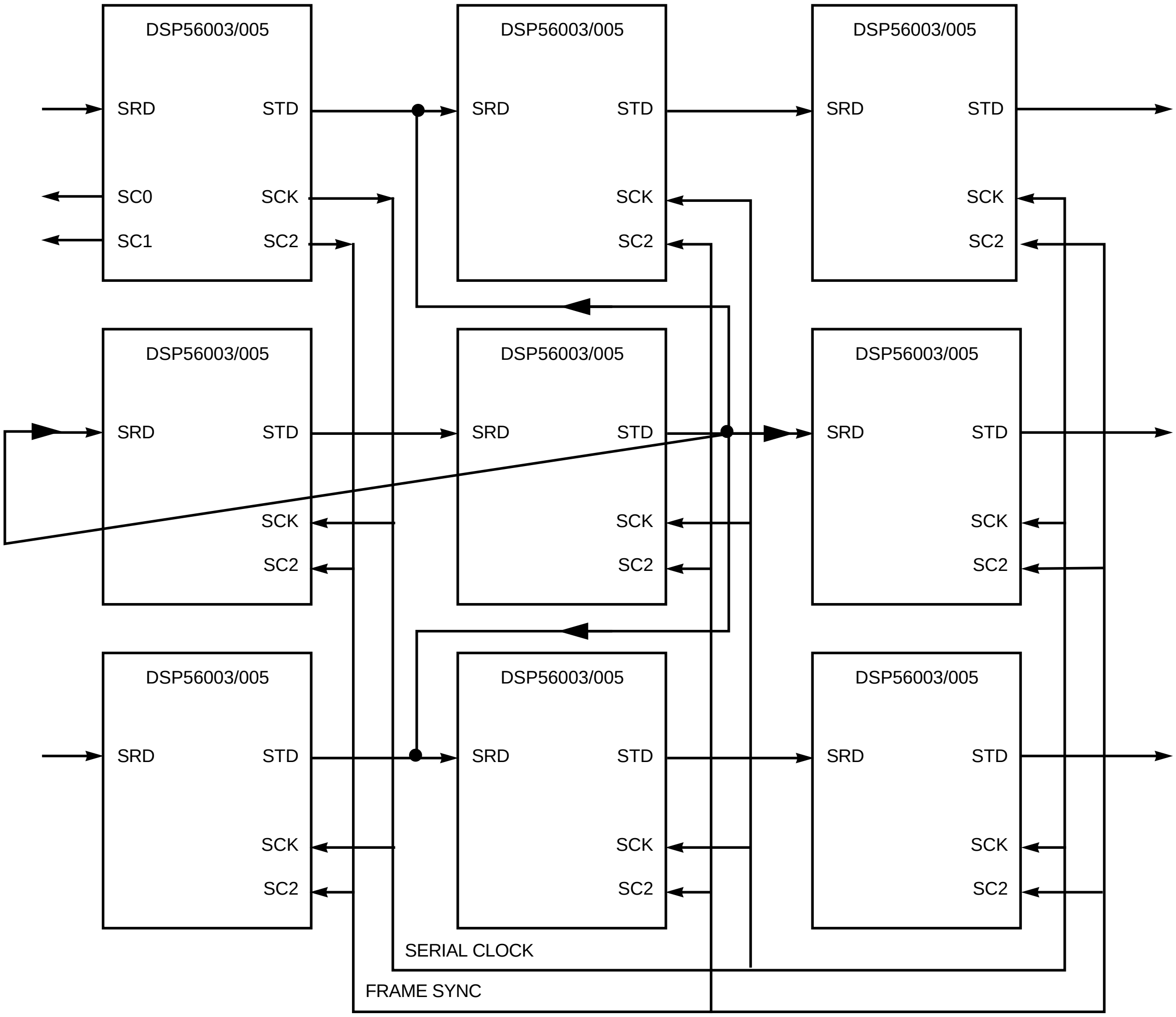


Figure 7-61 SSI Parallel Processing — Nearest Neighbor Array

DSPs. However, the bus must be arbitrated by hardware or a software protocol to prevent collisions. The master/slave configuration shown in Figure 7-63 also allows data to be transferred between any two DSPs but simplifies network control.

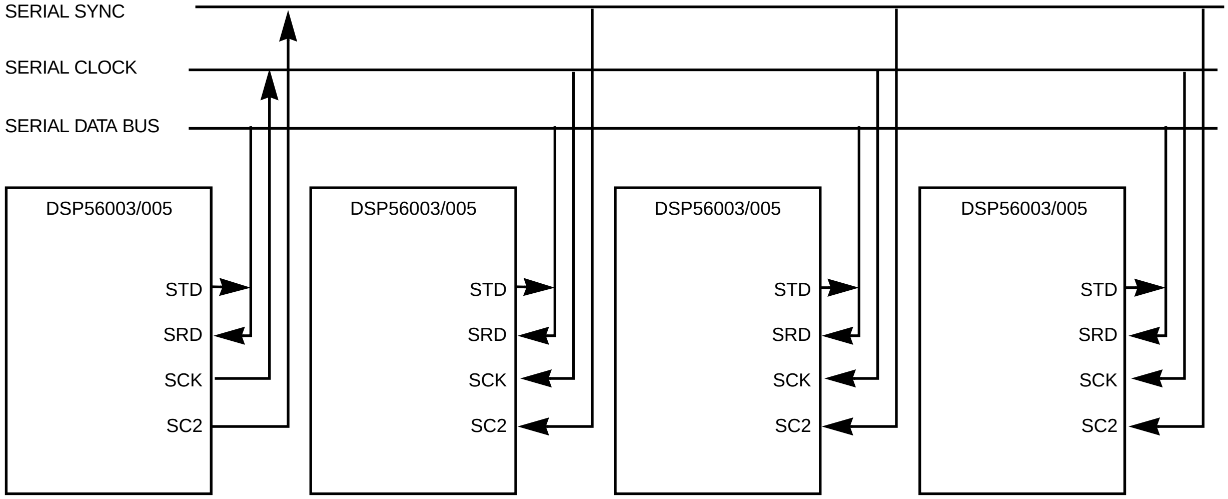
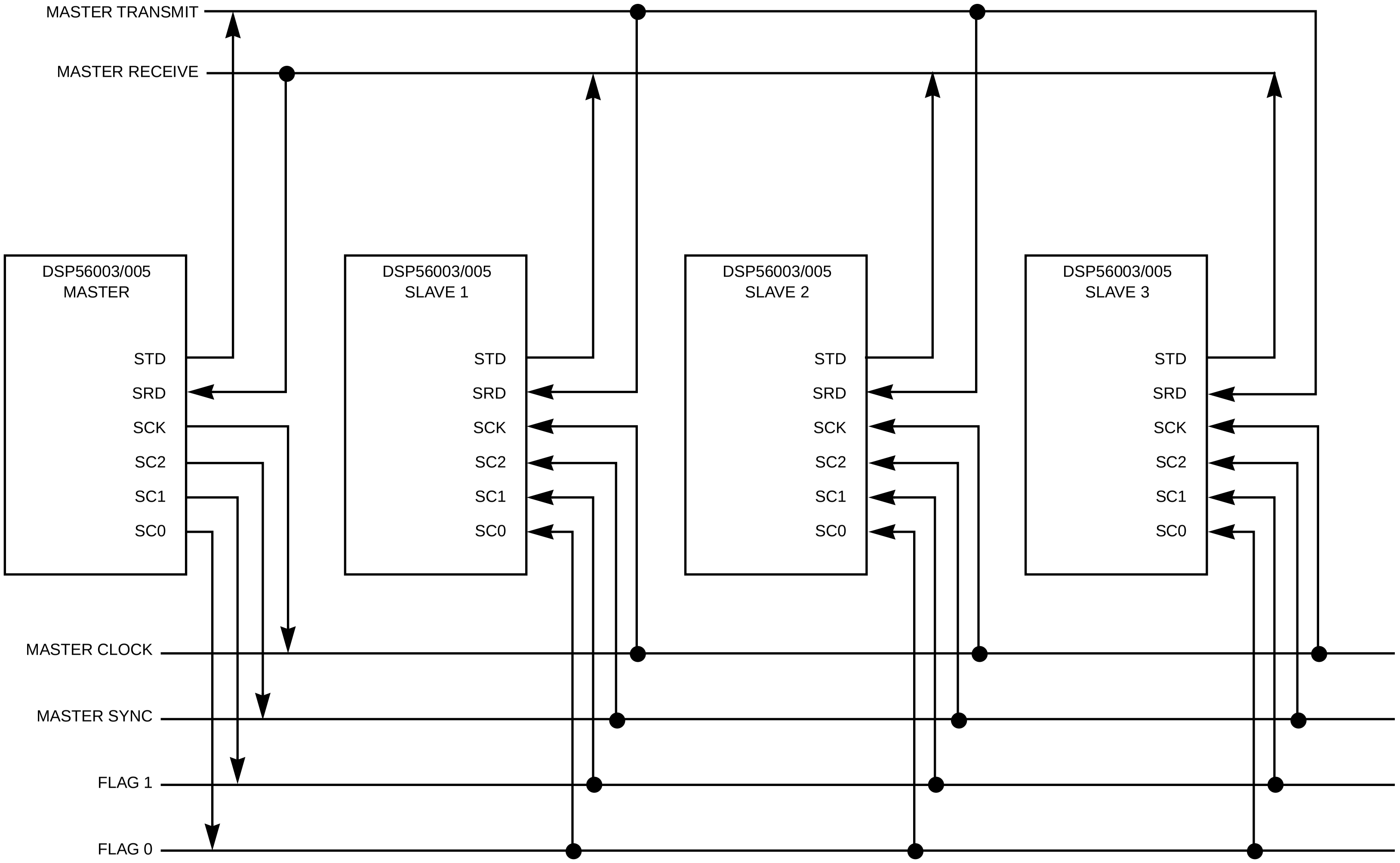


Figure 7-62 SSI TDM Bus DSP Network

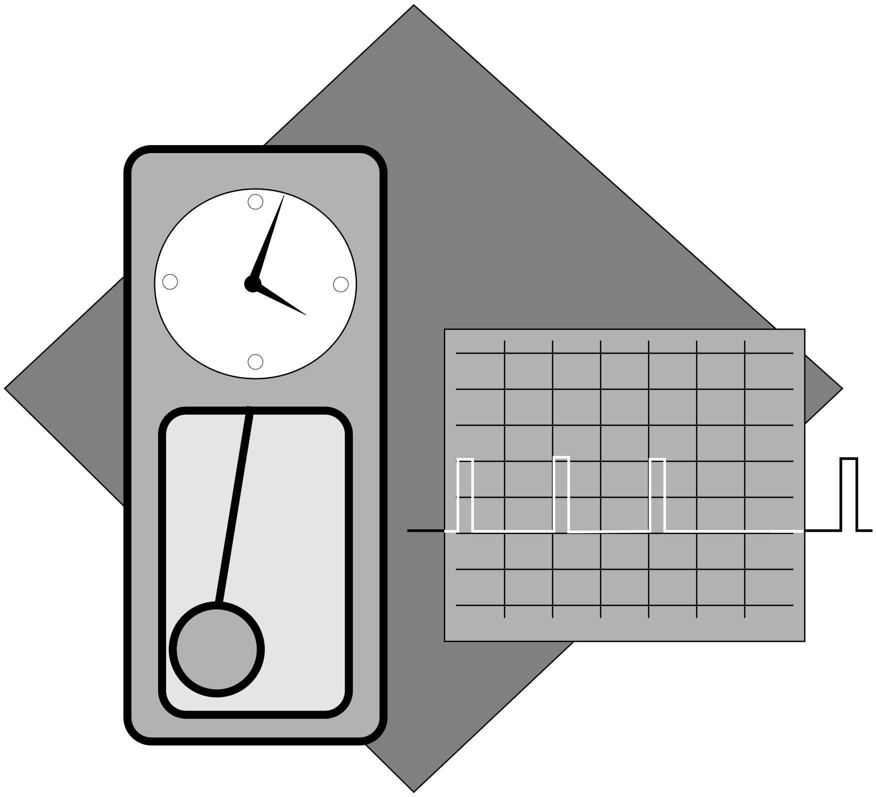


NOTE: Flags can specify data types: control, address, and data.

Figure 7-63 SSI TDM Master-Slave DSP Network

SECTION 8

TIMER/EVENT COUNTER



SECTION CONTENTS

8.1 INTRODUCTION

This section describes the Timer/Event Counter module. The timer can use internal or external clocking and can interrupt the processor after a number of events (clocks) specified by a user program, or it can signal an external device after counting internal events. This Timer/Event Counter is identical to the one on the DSP56002.

The timer connects to the external world through the bidirectional TIO pin. When TIO is used as input, the module is functioning as an external event counter or is measuring external pulse width/signal period. When TIO is used as output, the module is functioning as a timer and TIO becomes the timer pulse. When the TIO pin is not used by the timer module it can be used as a general purpose I/O (GPIO) pin.

Note: When the timer is disabled, the TIO pin becomes three-stated. The TIO pin should be pulled up or down to prevent undesired spikes from occurring when enabling it for use as a clock source when it is three-stated.

8.2 TIMER/EVENT COUNTER BLOCK DIAGRAM

Figure 8-1 shows a block diagram of the timer module. It includes a 24-bit read-write Timer Control and Status Register (TCSR), a 24-bit read-write Timer Count Register (TCR), a 24-bit counter, and logic for clock selection and interrupt generation.

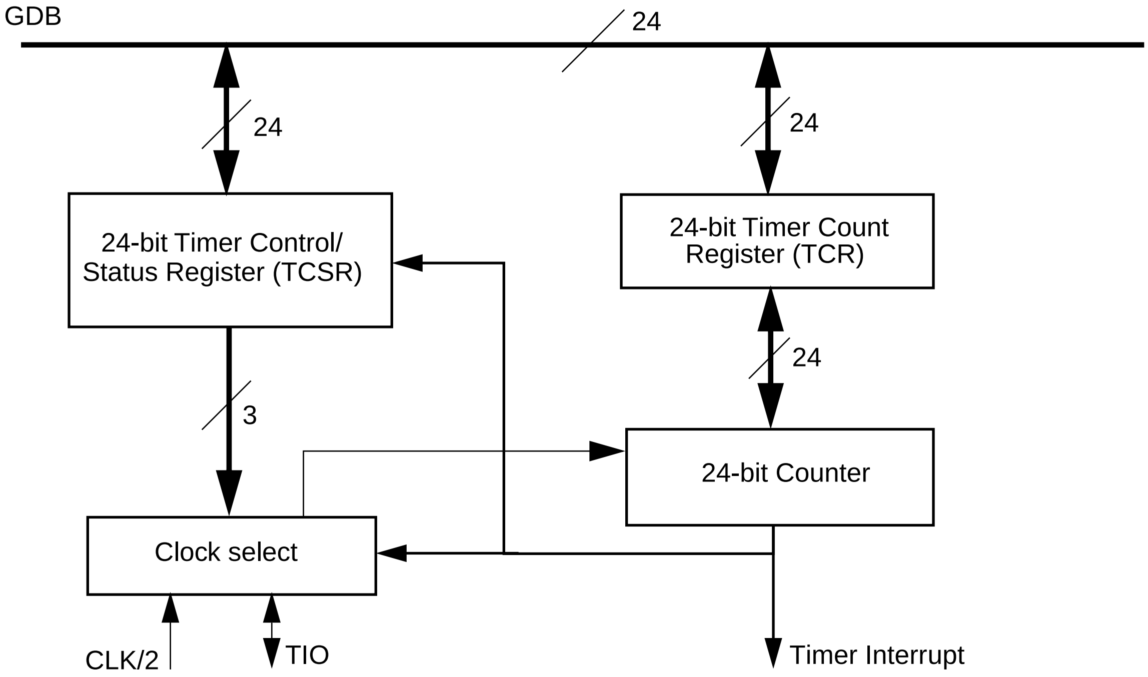
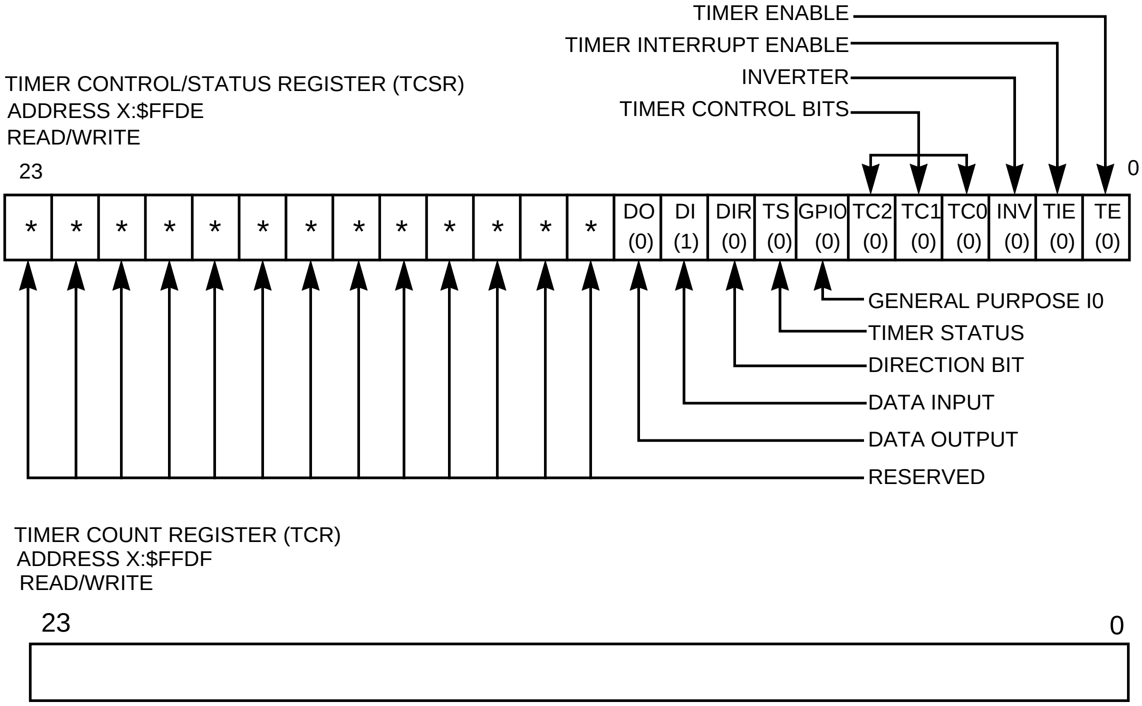


Figure 8-1 Timer/Event Counter Module Block Diagram



* - reserved, read as zero, should be written with zero for future compatibility

Figure 8-2 Timer/Event Counter Programming Model

The DSP56003/005 views the timer as a memory-mapped peripheral occupying two 24-bit words in the X data memory space, and may use it as a normal memory-mapped peripheral by using standard polled or interrupt programming techniques.The programming model is shown in Figure 8-2.

8.3 TIMER COUNT REGISTER (TCR)

The 24-bit read-write TCR contains the value (specified by the user program) to be loaded into the counter when the timer is enabled (TE=1), or when the counter has been decremented to zero and a new event occurs. If the TCR is loaded with n, the counter will be reloaded after (n+1) events.

If the timer is disabled (TE=0) and the user program writes to the TCR, the value is stored there but will not be loaded into the counter until the timer becomes enabled. When the timer is enabled (TE=1) and the user program writes to the TCR, the value is stored there and will be loaded into the counter after the counter has been decremented to zero and a new event occurs.

In Timer Modes 4 and 5, however, the TCR will be loaded with the current value of the counter on the appropriate edge of the TIO input signal (rather than with a value specified by the user program). The value loaded to the TCR represents the width or the period of the signal coming in on the TIO pin, depending on the timer mode. See **Sections 8.5.4** and **8.5.5** for detailed descriptions of Timer Modes 4 and 5.

8.4 TIMER CONTROL/STATUS REGISTER (TCSR)

The 24-bit read/write TCSR controls the timer and verifies its status. The TCSR can be accessed by normal move instructions and by bit manipulation instructions. The control and status bits are described in the following paragraphs.

8.4.1 TCSR Timer Enable (TE) Bit 0

The TE bit enables or disables the timer. Setting the TE bit (TE=1) will enable the timer, and the counter will be loaded with the value contained in the TCR and will start decrementing at each incoming event. Clearing the TE bit will disable the timer. Hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction) clear TE.

8.4.2 TCSR Timer Interrupt Enable (TIE) Bit 1

The TIE bit enables the timer interrupts after the counter reaches zero and a new event occurs. If TCR is loaded with n, an interrupt will occur after (n+1) events.

Setting TIE (TIE=1) will enable the interrupts.When the bit is cleared (TIE=0) the interrupts are disabled. Hardware and software resets clear TIE.

8.4.3 TCSR Inverter (INV) Bit 2

The INV bit affects the polarity of the external signal coming in on the TIO input and the polarity of the output pulse generated on the TIO output.

If TIO is programmed as an input and INV=0, the 0-to-1 transitions on the TIO input pin will decrement the counter. If INV=1, the 1-to-0 transitions on the TIO input pin will decrement the counter.

If TIO is programmed as output and INV=1, the pulse generated by the timer will be inverted before it goes to the TIO output pin. If INV=0, the pulse is unaffected.

In Timer Mode 4 (see **Section 8.5.4 Timer Mode 4 (Pulse Width Measurement Mode)**), the INV bit determines whether the high pulse or the low pulse is measured to determine input pulse width. In Timer Mode 5 (see **Section 8.5.5 Timer Mode 5 (Period Measurement Mode)**), the INV bit determines whether the period is measured between leading or trailing edges.

In GPIO mode, the INV bit determines whether the data read from or written to the TIO pin shall be inverted (INV=1) or not (INV=0).

INV is cleared by hardware and software resets.

Note: Because of its affect on signal polarity, and on how GPIO data is read and written, the status of the INV bit is crucial to the timer's function. Change it only when the timer is disabled (TE=0).

8.4.4 TCSR Timer Control (TC0-TC2) Bits 3-5

The three TC bits control the source of the timer clock, the behavior of the TIO pin, and the timer mode of operation. Table 8-1 summarizes the functionality of the TC bits.

The timer control bits are cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction).

Note 1: If the clock is external, the counter will be decremented by the transitions on the TIO pin. The DSP synchronizes the external clock to its own internal clock. The external clock's frequency should be lower than the maximum internal frequency divided by 4 (CLK/4).

Note 2: The TC2-TC0 bits should be changed only when TE=0 (timer disabled) to ensure proper functionality.

Table 8-1 Timer/Event Counter Control Bits

TC2	TC1	TC0	TIO	CLOCK	MODE
0	0	0	GPIO*	Internal	Timer (Mode 0)
0	0	1	Output	Internal	Timer Pulse (Mode 1)
0	1	0	Output	Internal	Timer Toggle (Mode 2)
0	1	1	—	—	Reserved - Do Not Use
1	0	0	Input	Internal	Input Width (Mode 4)
1	0	1	Input	Internal	Input Period (Mode 5)
1	1	0	Input	External	Standard Time Counter (Mode 6)
1	1	1	Input	External	Event Counter (Mode 7)

* - the GPIO function is enabled only if TC2-TC0 are all 0 (zero) and the GPIO bit is set.

8.4.5 TCSR General Purpose I/O (GPIO) Bit 6

If the GPIO bit is set (GPIO=1) and if TC2-TC0 are all zeros, the TIO pin operates as a general purpose I/O pin, whose direction is determined by the DIR bit. If GPIO=0 the general purpose I/O function is disabled. GPIO is cleared by hardware and software resets.

Note: The case where TC2-TC0 are not all zero and GPIO=1 is undefined and should not be used.

8.4.6 TCSR Timer Status (TS) Bit 7

When the TS bit is set, it indicates that the counter has been decremented to zero.

The TS bit is cleared when the TCSR is read. The bit is also cleared when the timer interrupt is serviced (timer interrupt acknowledge). TS is cleared by hardware and software resets.

8.4.7 TCSR Direction (DIR) Bit 8

The DIR bit determines the behavior of the TIO pin when TIO acts as general purpose I/O. When DIR=0, the TIO pin acts as an input. When DIR=1, the TIO pin acts as an output. DIR is cleared by hardware and software resets.

Note: The TIO pin can act as a general purpose I/O pin only when TC2-TC0 are all zero **and** the GPIO bit is set. If one of TC2, TC1 or TC0 is not 0, the GPIO function is disabled and the DIR bit has no effect.

8.4.8 TCSR Data Input (DI) Bit 9

When the TIO pin acts as a general purpose I/O input pin (TC2-TC0 are all zero and DIR=0), the contents of the DI bit will reflect the value the TIO pin. However, if the INV bit is set, the data in DI will be inverted. When GPIO mode is disabled or it is enabled in output mode (DIR=1), the DI bit reflects the value of the TIO pin, again depending on the status of the INV bit. DI is set by hardware and software resets.

8.4.9 TCSR Data Output (DO) Bit 10

When the TIO pin acts as a general purpose I/O output pin (TC2-TC0 are all zero and DIR=1), writing to the DO bit writes the data to the TIO pin. However, if the INV bit is set, the data written to the TIO pin will be inverted. When GPIO mode is disabled, writing to the DO bit will have no effect. DO is cleared by hardware and software resets.

8.4.10 TCSR Reserved Bits 11-23

These reserved bits are read as zero and should be written with zero for future compatibility.

8.5 TIMER/EVENT COUNTER MODES OF OPERATION

This section gives the details of each of the timer modes of operation. Table 8-1 on page 8-6 summarizes the items which determine the timer mode, including the configuration of the timer control bits, the function of the TIO pin, and the clock source.

8.5.1 Timer Mode 0 (Standard Timer Mode, Internal Clock, No Timer Output)

Timer Mode 0 is defined by TCSR bits TC2-TC0 equal to 000.

With the timer enabled (TE=1), the counter is loaded with the value contained by the TCR. The counter is decremented by a clock derived from the internal DSP clock, divided by two (CLK/2). During the clock cycle following the point where the counter reaches 0, the TS bit is set and, if the TIE bit is set, the timer generates an interrupt. The counter is reloaded with the value contained by the TCR, and the entire process is repeated until the timer is disabled (TE=0). Figure 8-3 illustrates Mode 0 with the timer enabled. Figure 8-4 illustrates the events with the timer disabled.

Note: It is recommended that the GPIO input function of Mode 0 only be activated with the timer disabled. If the processor attempts to read the DI bit to determine the GPIO pin direction, it must read the entire TCSR register, which would clear the TS bit and, thus, clear a pending timer interrupt.

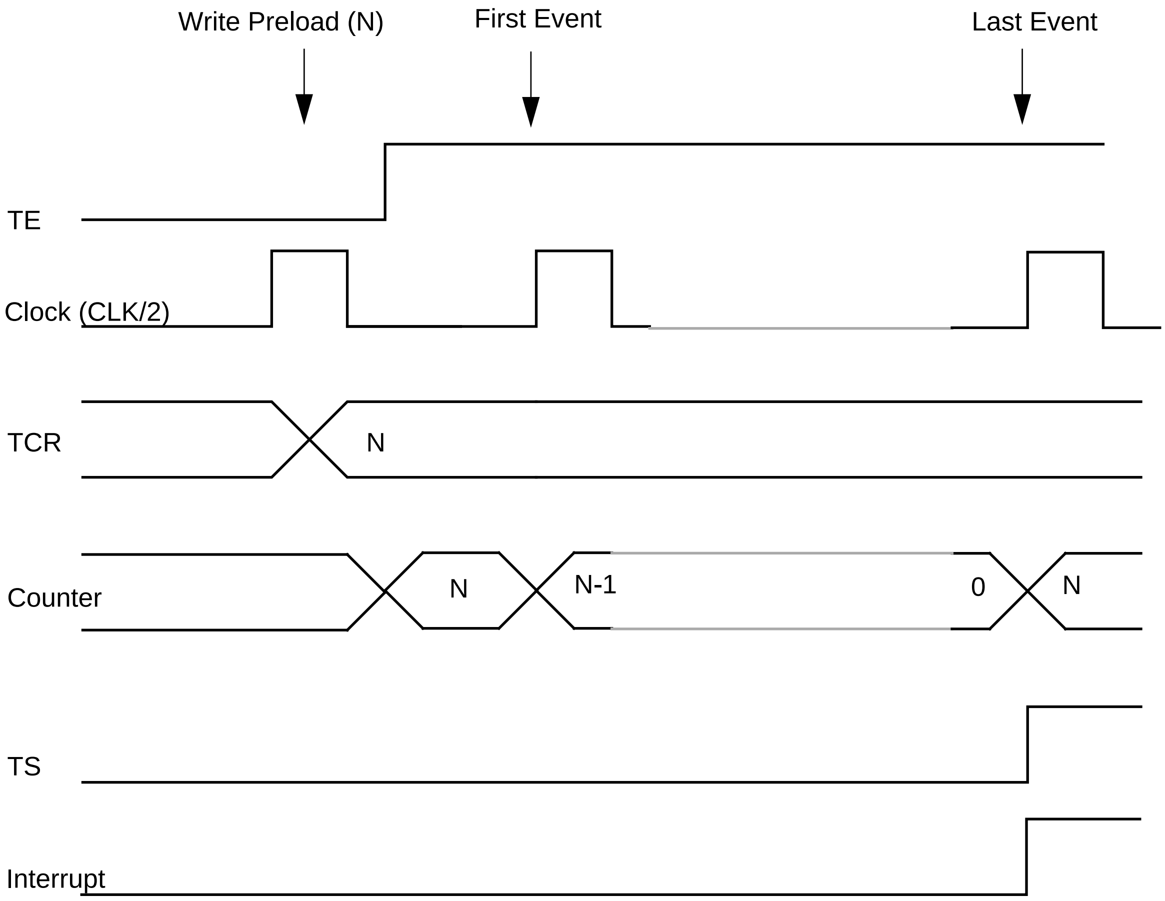


Figure 8-3 Mode 0 — Standard Timer Mode

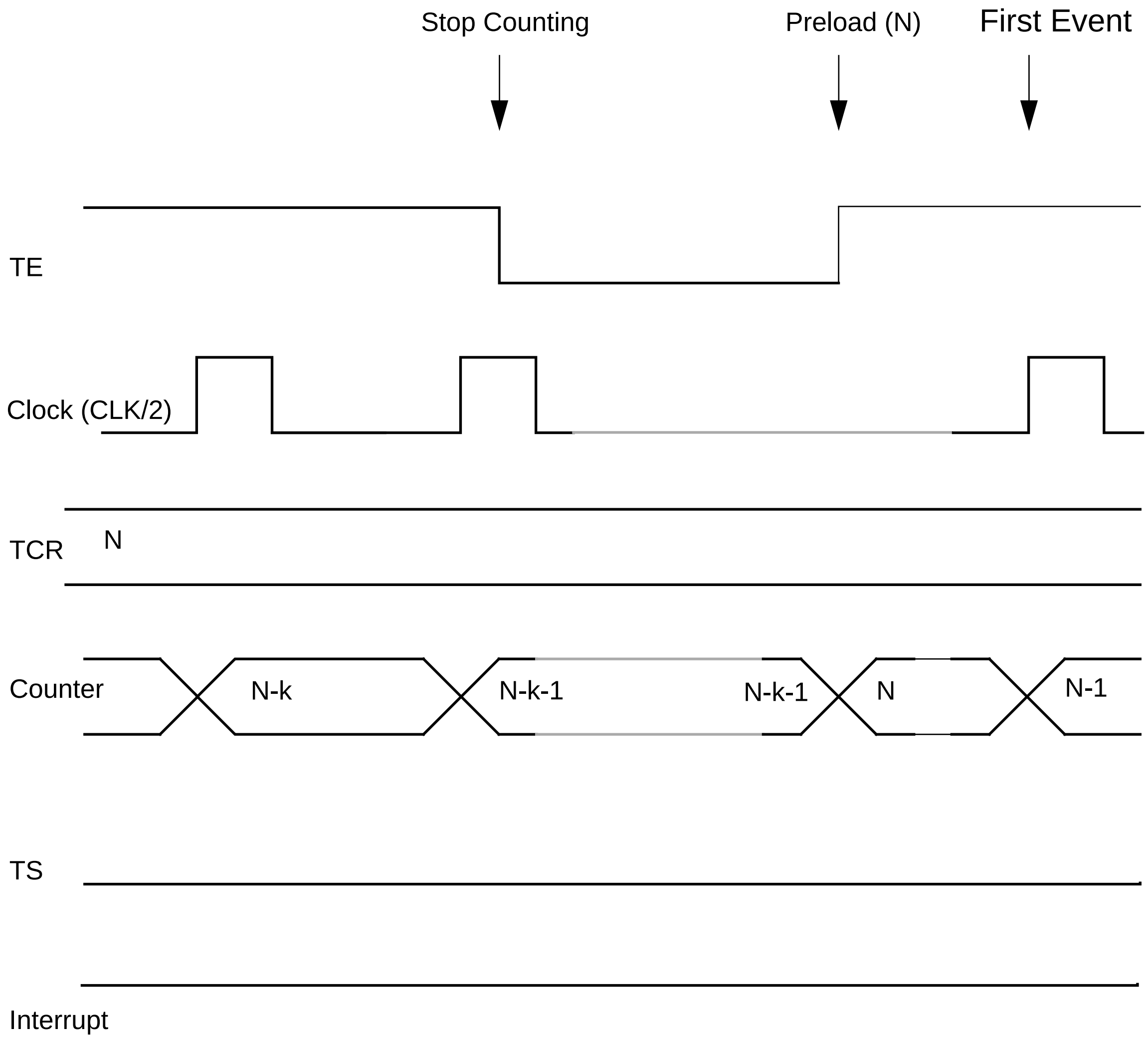


Figure 8-4 Timer/Event Counter Disable

8.5.2 Timer Mode 1 (Standard Timer Mode, Internal Clock, Output Pulse Enabled)

Timer Mode 1 is defined by TC2-TC0 equal to 001. With the timer enabled (TE=1), the counter is loaded with the value contained by the TCR. The counter is decremented by a clock derived from the DSP's internal clock, divided by two (CLK/2). During the clock cycle following the point where the counter reaches 0, the TS bit is set and, if the TIE bit is set, the timer generates an interrupt. A pulse with a two clock cycle width and whose polarity is determined by the INV bit, will be put out on the TIO pin. The counter is reloaded with the value contained by the TCR. The entire process is repeated until the timer is disabled (TE=0). Figure 8-5 illustrates Timer Mode 1 when INV=0, and Figure 8-6 illustrates Timer Mode 1 when INV=1.

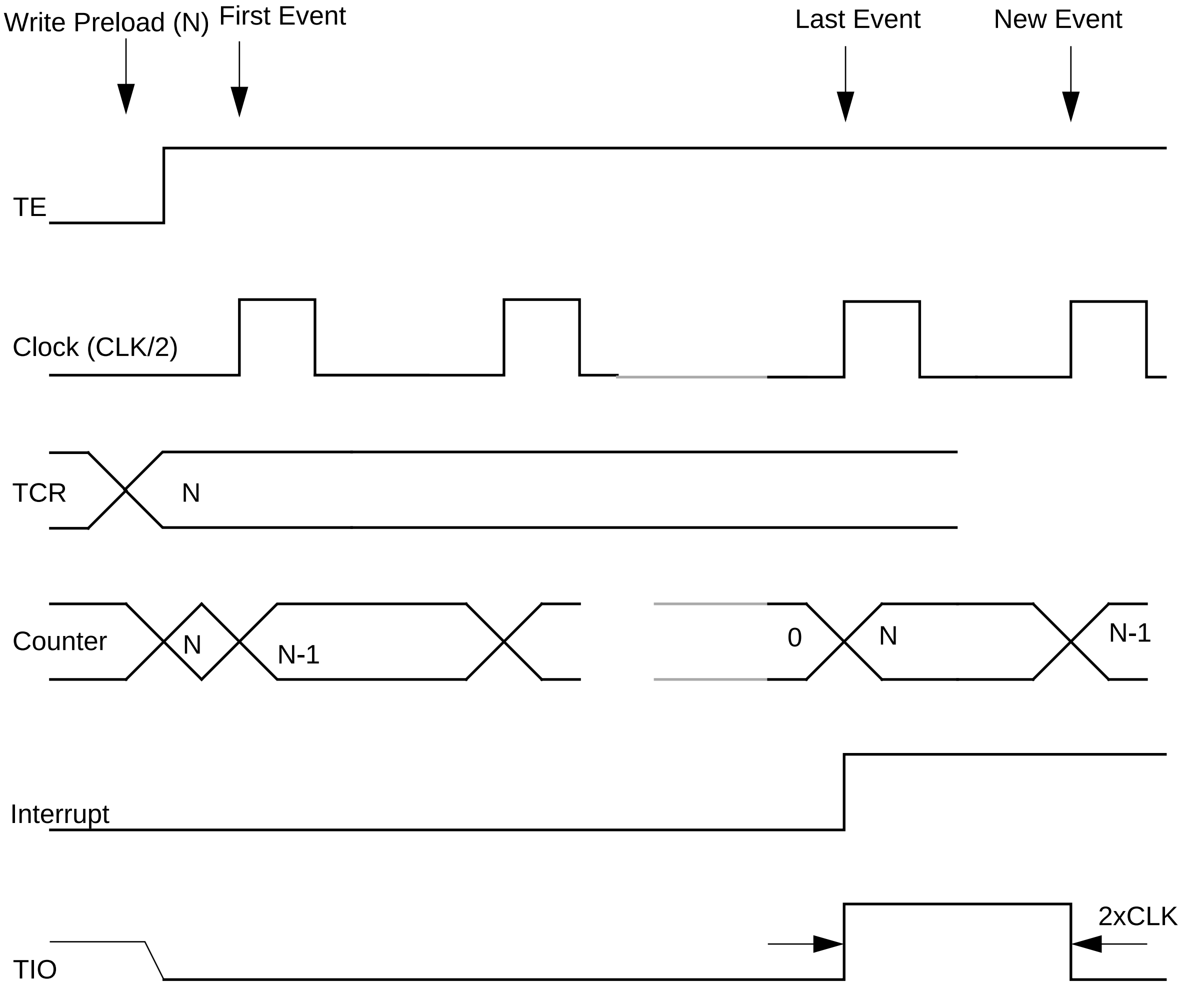


Figure 8-5 Mode 1 — Standard Timer Mode, Internal Clock, Output Pulse Enabled (INV=0)

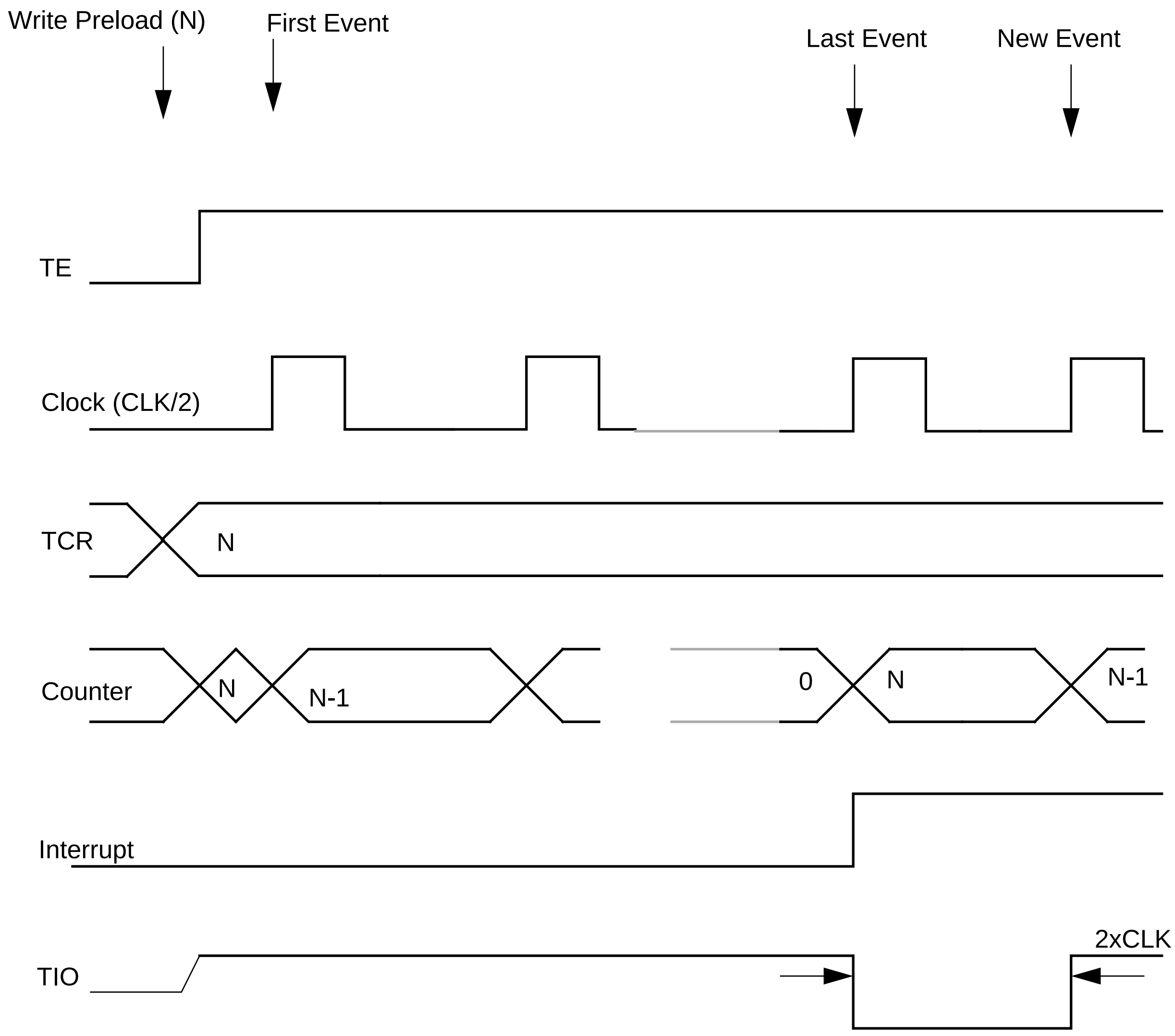


Figure 8-6 Mode 1 — Standard Timer Mode, Internal Clock, Output Pulse Enabled (INV=1)

8.5.3 Timer Mode 2 (Standard Timer Mode, Internal Clock, Output Toggle Enabled)

Timer Mode 2 is defined by TC2-TC0 equal to 010. With the timer enabled (TE=1), the counter is loaded with the value contained by the TCR. The counter is decremented by a clock derived from the DSP's internal clock, divided by two (CLK/2). During the clock cycle following the point where the counter reaches 0, the TS bit in TCSR is set and, if the TIE is set, an interrupt is generated.The counter is reloaded with the value contained by the TCR and the entire process is repeated until the timer is disabled (TE=0). Each time the counter reaches 0, the TIO output pin will be toggled. The INV bit determines the polarity of the TIO output. Figure 8-7 illustrates Timer Mode 2.

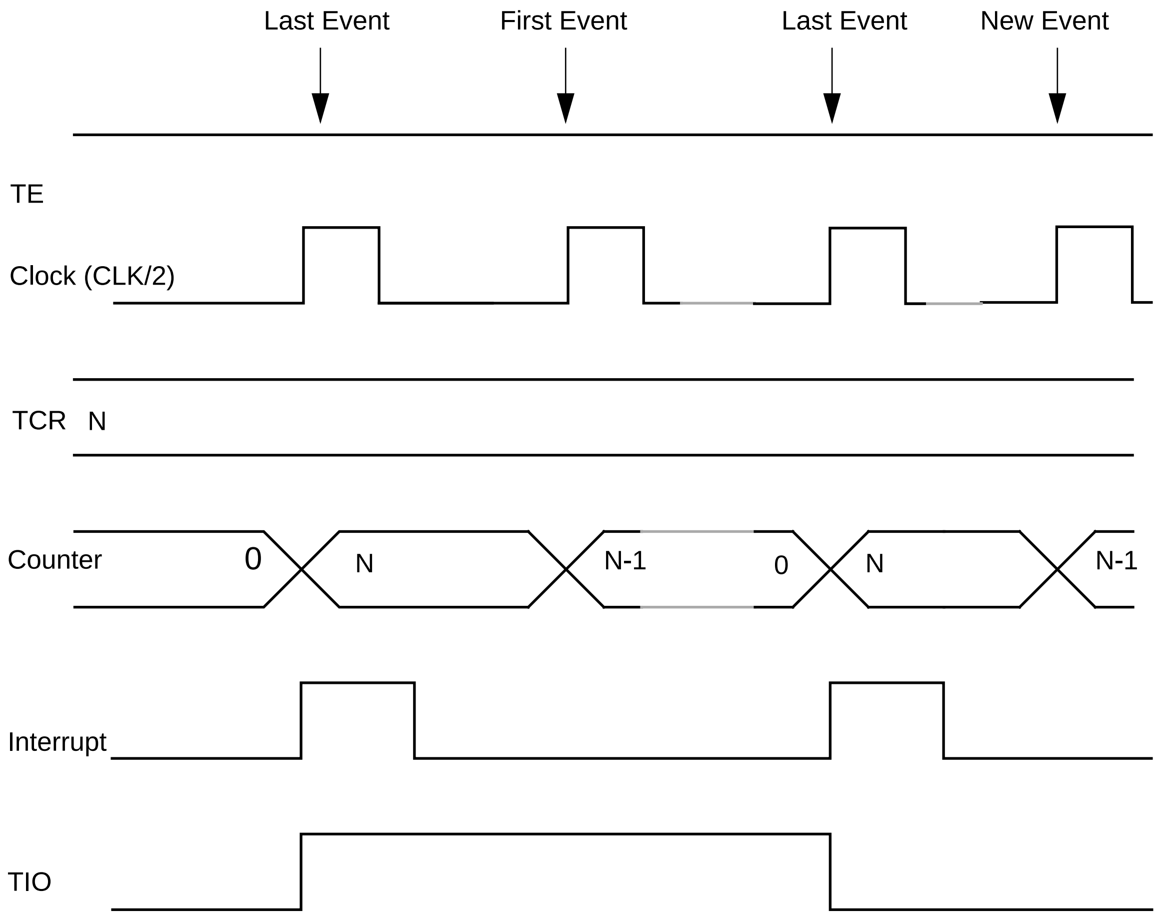


Figure 8-7 Mode 2 — Standard Timer Mode, Internal Clock, Output Toggle Enable

8.5.4 Timer Mode 4 (Pulse Width Measurement Mode)

Timer Mode 4 is defined by TC2-TC0 equal to 100. In this mode, TIO acts as a gating signal for the DSP’s internal clock (see Figure 8-9). With the timer enabled (TE=1), the counter is driven by a clock derived from the DSP’s internal clock divided by two (CLK/2). The counter is loaded with 0 by the first transition occurring on the TIO input pin and starts incrementing. When the first edge of opposite polarity occurs on TIO, the counter stops, the TS bit in TCSR is set and, if TIE is set, an interrupt is generated.

The contents of the counter is loaded into the TCR. The user’s program can read the TCR, which now represents the width of the TIO pulse. The process is repeated until the timer is disabled (TE=0).The INV bit determines whether the counting is enabled when TIO is high (INV=0) or when TIO is low (INV=1). Figure 8-8 illustrates Timer Mode 4 when INV=0 and Figure 8-10 illustrates Timer Mode 4 with INV=1.

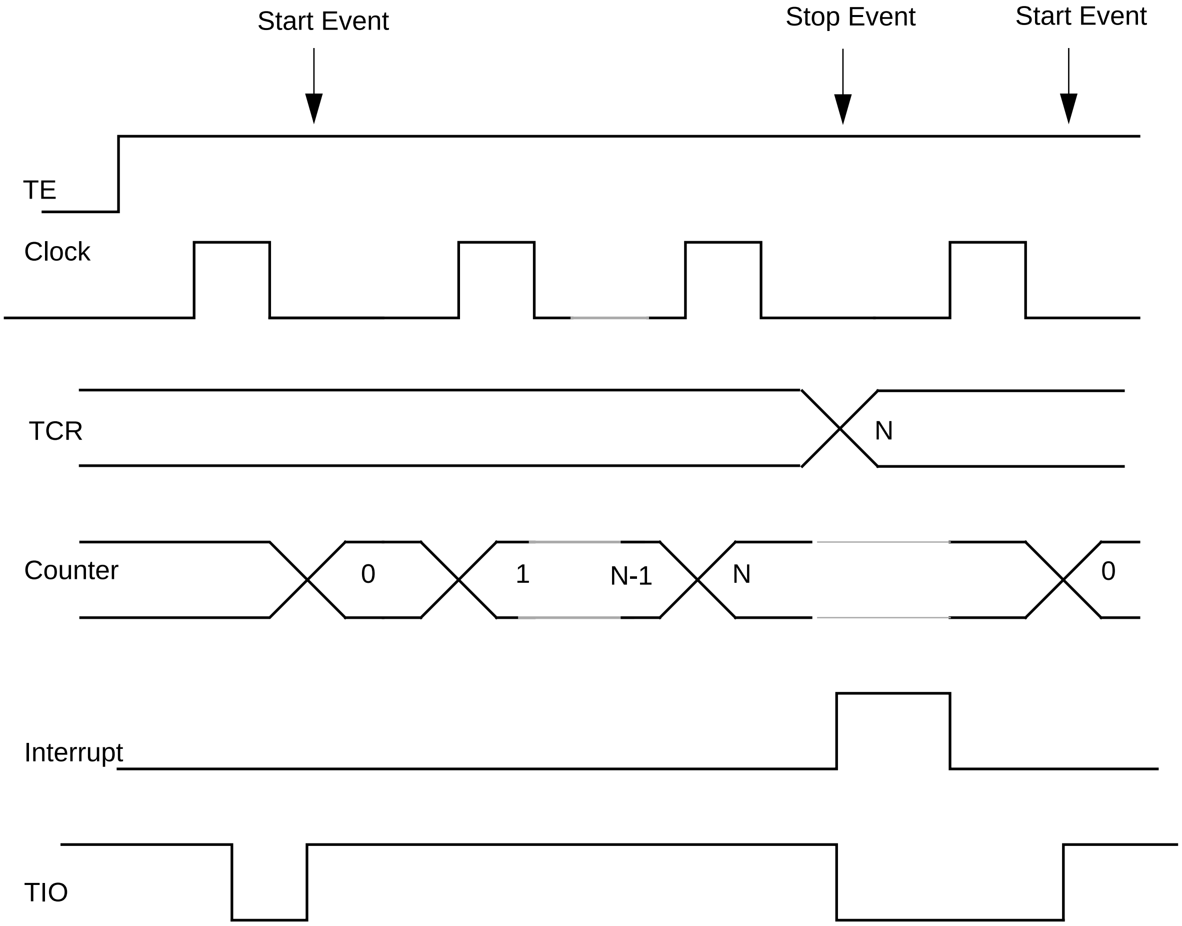


Figure 8-8 Mode 4 — Pulse Width Measurement Mode (INV=0)

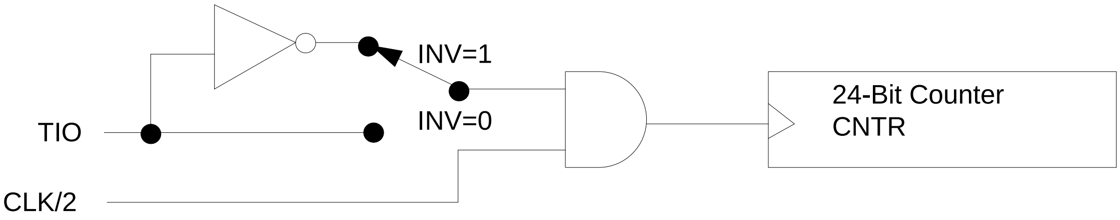


Figure 8-9 Mode 4 —TIO Gates the Internal Clock

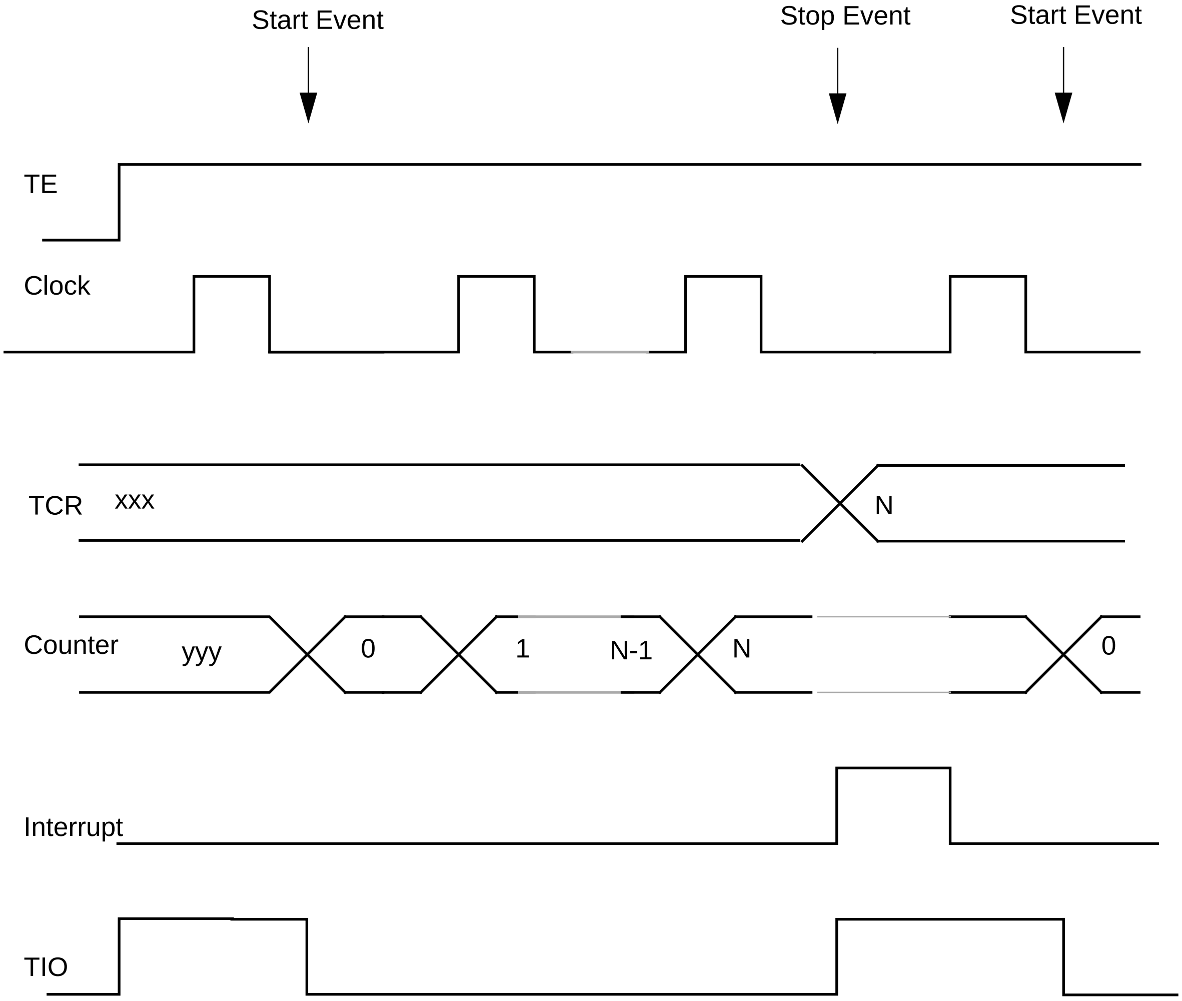


Figure 8-10 Mode 4 — Pulse Width Measurement Mode (INV=1)

8.5.5 Timer Mode 5 (Period Measurement Mode)

Timer Mode 5 is defined by TC2-TC0 equal to 101. In Timer Mode 5, the counter is driven by a clock derived from the DSP's internal clock divided by 2 (CLK/2). With the timer enabled (TE=1), the counter is loaded with the value contained by the TCR and starts incrementing. On each transition of the same polarity that occurs on TIO, the TS bit in TCSR is set and, if TIE is set, an interrupt is generated. The contents of the counter are loaded into the TCR. The user's program can read the TCR and subtract consecutive values of the counter to determine the distance between TIO edges. The counter is not stopped and it continues to increment. The INV bit determines whether the period is measured between 0-to-1 transitions of TIO (INV=0), or between 1-to-0 transitions of TIO (INV=1). Figure 8-11 illustrates Timer Mode 5 when INV=0, and Figure 8-12 illustrates this mode with INV=1.

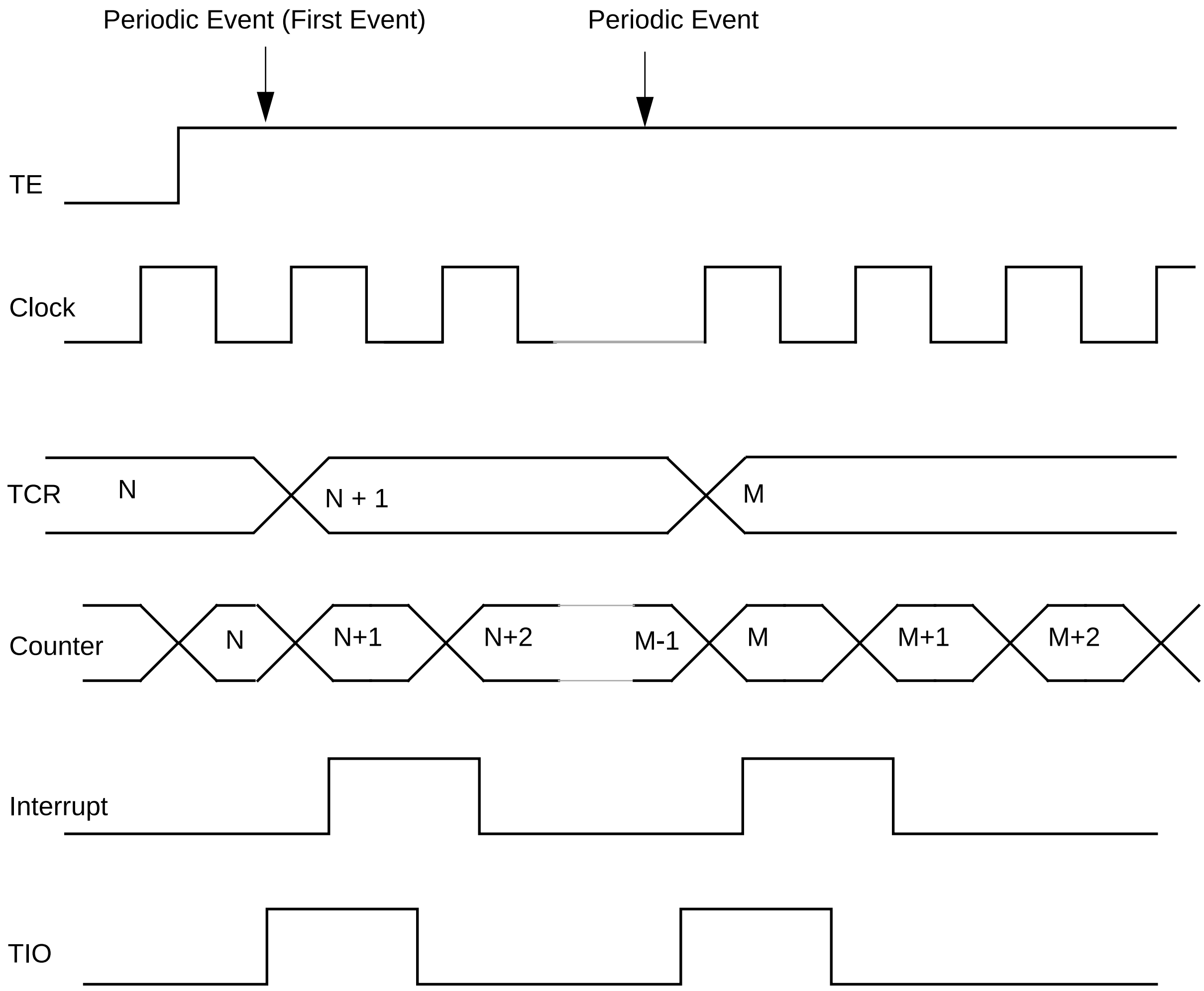


Figure 8-11 Mode 5 — Period Measurement Mode (INV=0)

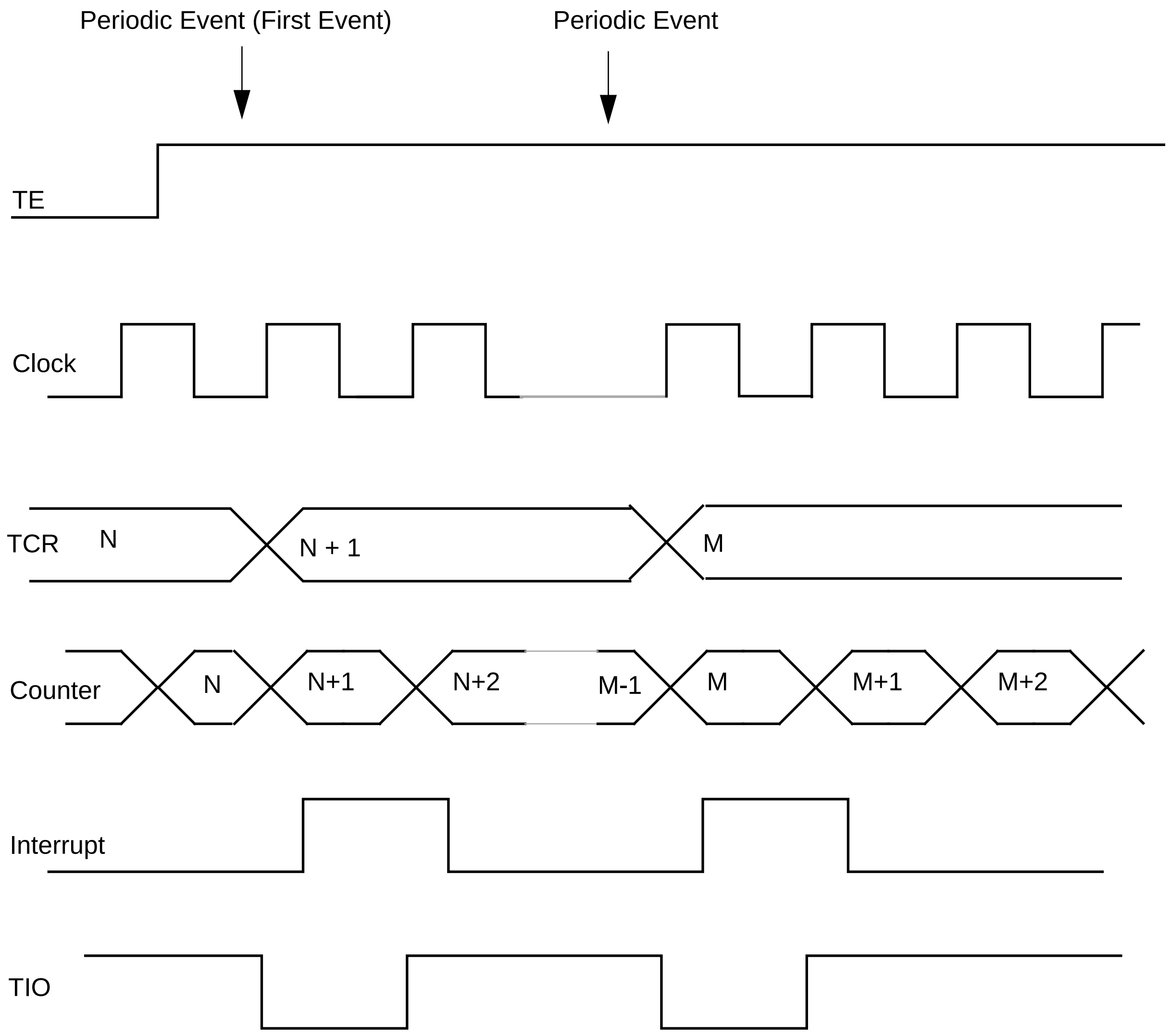


Figure 8-12 Mode 5 — Period Measurement Mode (INV=1)

8.5.6 Timer Mode 6 (Standard Time Counter Mode, External Clock)

Time Mode 6 is defined by TC2-TC0 equal to 110. With the timer enabled (TE=1) the counter is loaded with the 1's complement of the value contained by the TCR. The counter is incremented by the transitions on the incoming signal on the TIO input pin. After each increment, the counter value is loaded into the TCR. Thus, reading the TCR will give the value of the counter at any given moment. At the transition following the point where the counter reaches 0, the TS bit in TCSR is set and, if the TIE is set, an interrupt is generated.The counter will wrap around and the process is repeated until the timer is disabled (TE=0). The INV bit determines whether 0-to-1 transitions (INV=0) or 1-to-0 transitions (INV=1) will increment the counter. Figure 8-13 illustrates Timer Mode 6 when INV=0. Figure 8-14 illustrates Timer Mode 6 when INV=1.

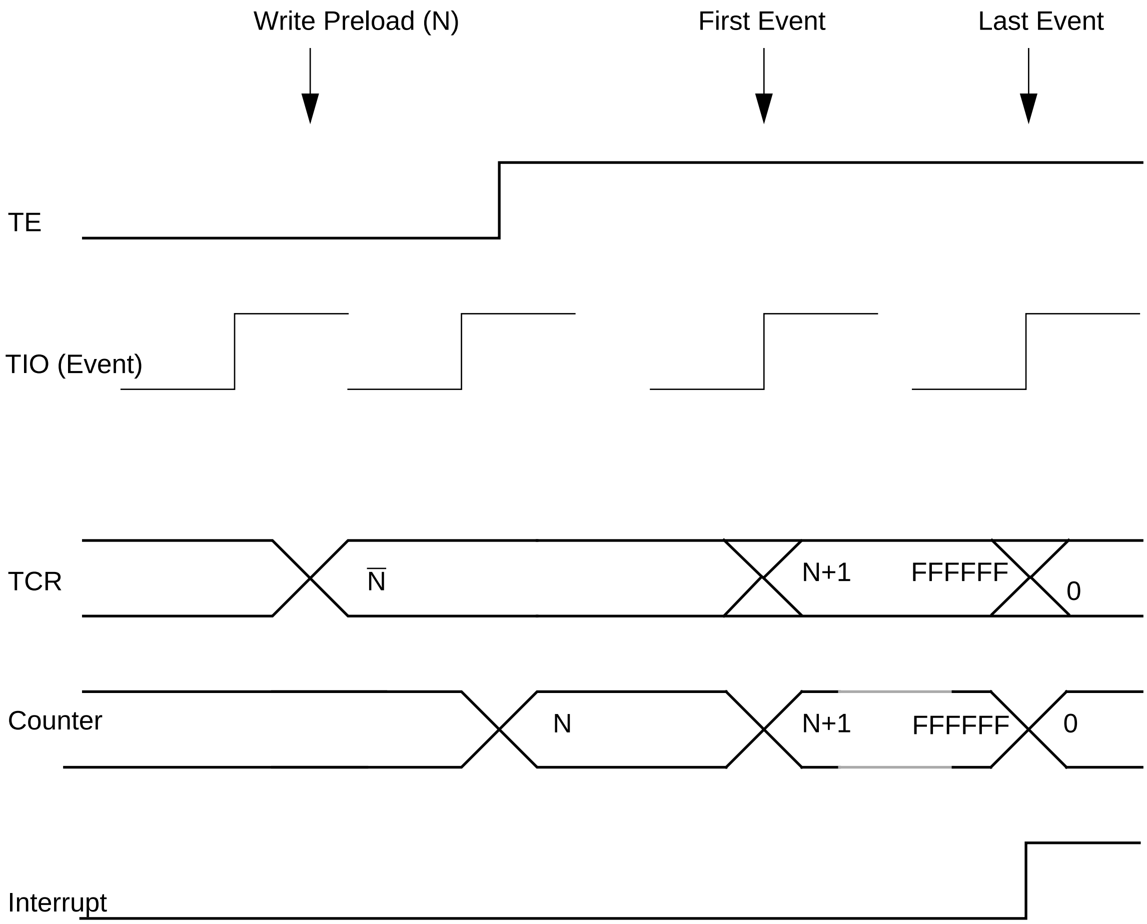


Figure 8-13 Mode 6 — Standard Time Counter Mode, External Clock (INV=0)

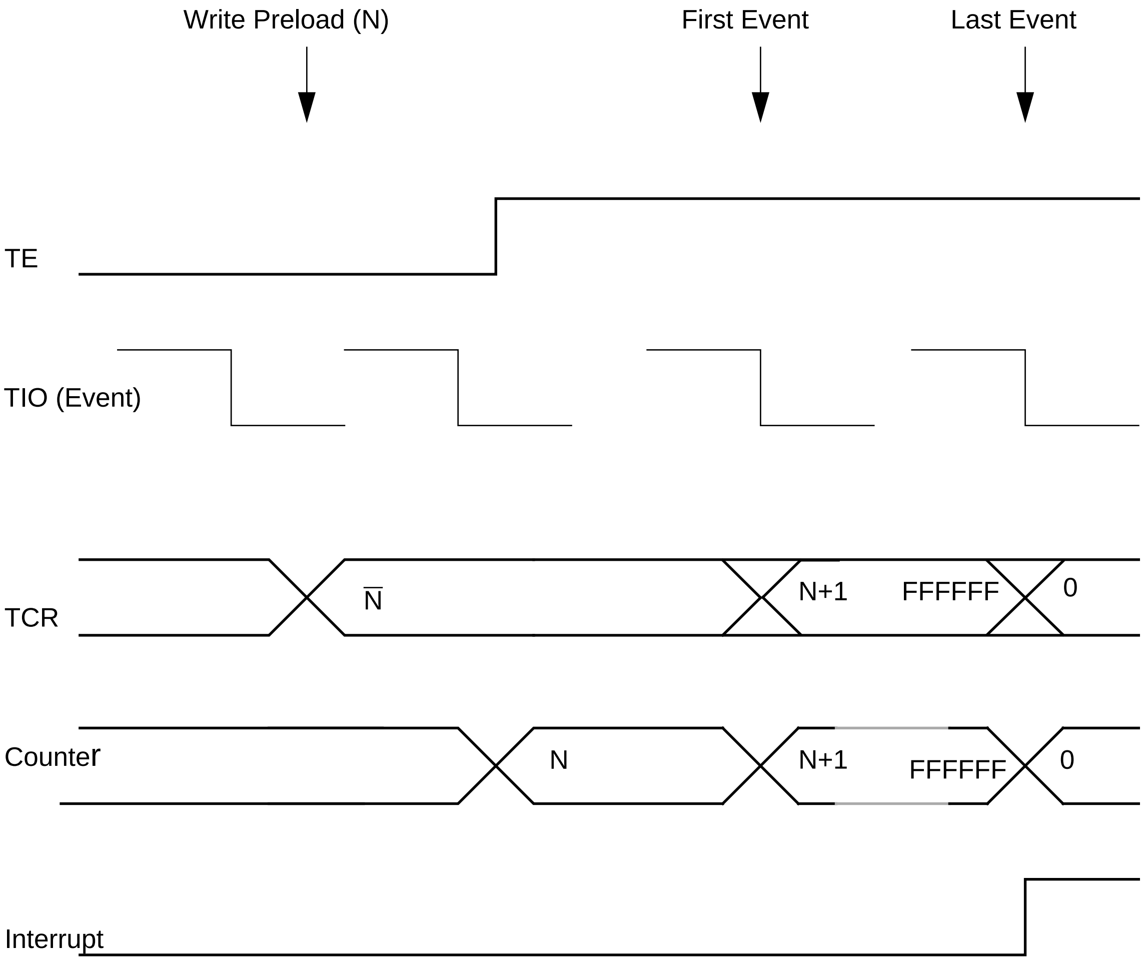


Figure 8-14 Mode 6 — Standard Timer Mode, External Clock (INV=1)

8.5.7 Timer Mode 7 (Standard Timer Mode, External Clock)

Timer Mode 7 is defined by TC2-TC0 equal to 111. With the timer enabled (TE=1), the counter is loaded with the value contained by the TCR. The counter is decremented by the transitions of the signal coming in on the TIO input pin. At the transition that occurs after the counter has reached 0, the TS bit in TCSR is set and, if the TIE is set, the timer generates an interrupt. The counter is reloaded with the value contained by the TCR, and the entire process is repeated until the timer is disabled (TE=0). The INV bit determines whether 0-to-1 transitions (INV=0) or 1-to-0 transitions (INV=1) will decrement the counter. Figure 8-15 illustrates Timer Mode 7 when INV=0, and Figure 8-16 illustrates Timer Mode 7 when INV=1.

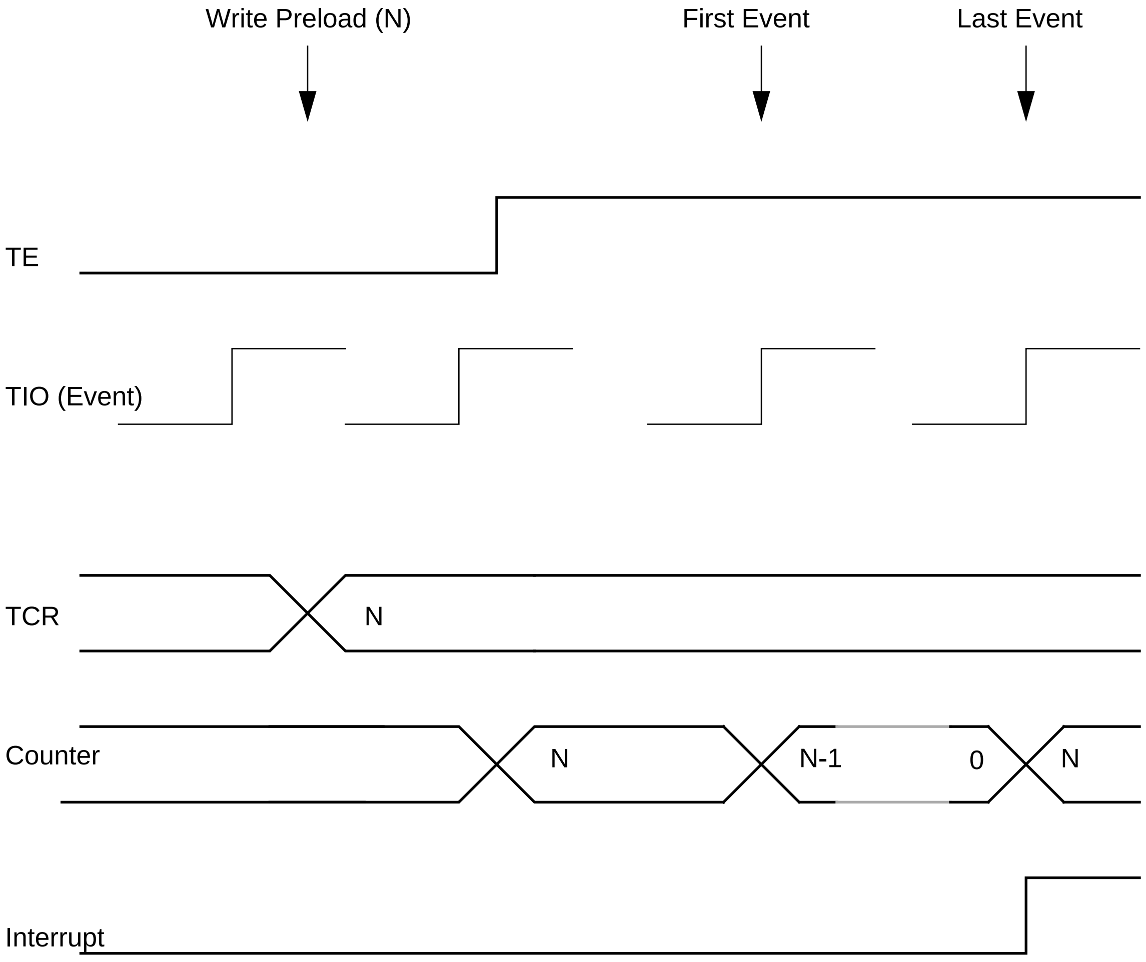


Figure 8-15 Mode 7 — Standard Timer Mode, External Clock (INV=0)

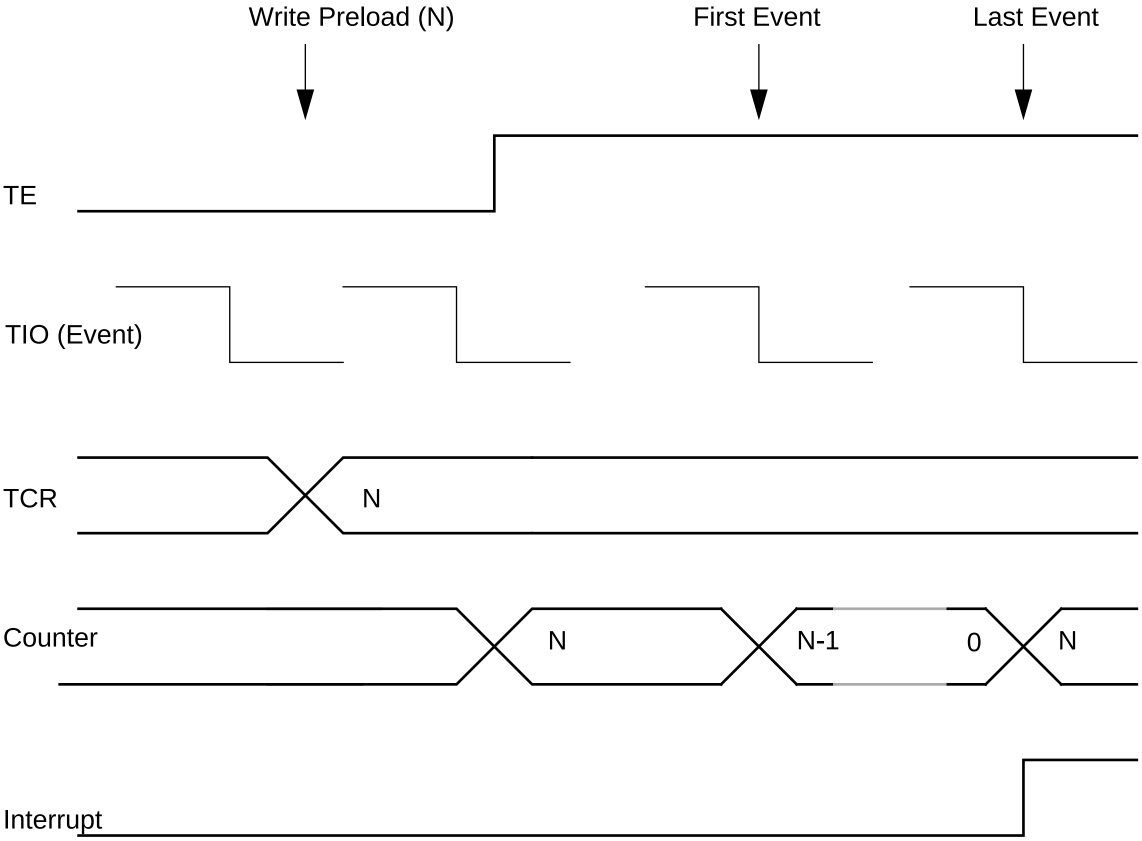


Figure 8-16 Mode 7 — Standard Timer Mode, External Clock (INV=1)

8.6 TIMER/EVENT COUNTER BEHAVIOR DURING WAIT AND STOP

During the execution of the WAIT instruction, the timer clocks are active and the timer activity continues undisturbed. If the timer interrupt is enabled when the final event occurs, an interrupt will be generated and serviced.

It is recommended that the timer be disabled before executing the STOP instruction because during the execution of the STOP instruction, the timer clocks are disabled and the timer activity will be stopped. If, for example, the TIO pin is used as input, the changes that occur while in STOP will be ignored.

8.7 OPERATING CONSIDERATIONS

The value 0 for the Timer Count Register (TCR) is considered a boundary case and affects the behavior of the timer under the following conditions:

- If the TCR is loaded with 0, and the counter contained a non-zero value before the TCR was loaded, then after the timer is enabled, it will count 2^{24} events, generate an interrupt, and then generate an interrupt for every new event.
- If the TCR is loaded with 0, and the counter contained a zero value prior to loading, then after the timer is enabled, it will generate an interrupt for every event.
- If the TCR is loaded with 0 after the timer has been enabled, the timer will be loaded with 0 when the current count is completed and then generate an interrupt for every new event.

8.8 SOFTWARE EXAMPLES

8.8.1 General Purpose I/O Input

The following routine can be used to read the TIO input pin:

```
MOVEP    #$000040,X:TCSR     ;clear TC2-TC0, set GPIO and
                             ;clear INV for GPIO input here
JSET     #DI,X:TCSR,here     ;spin here until TIO is set
```

8.8.2 General Purpose I/O Output

The following routine can be used to write the TIO output pin:

```
MOVEP   #$000140,X:TCSR      ;clear TC2-TC0, set GPIO and
                        ;set DIR for GPIO output, set TIO to 0
BSET    #DO,X:TCSR           ;set TIO to 1
NOP
NOP
BCLR     #DO,X:TCSR          ;set TIO to 0
```

This routine generates a pulse on the TIO pin with the duration equal to 8 CLK (assuming no wait states, no external bus conflict, etc.).

8.8.3 Timer Mode 0, Input Clock, GPIO Output, and No Timer Output

The following program (see Figure 8-17) illustrates the standard timer mode with simultaneous GPIO. The timer is used to activate an internal task after 65536 clocks; at the end of the task the TIO pin is toggled to signal end of task.

```
        ORG     P:$3C          ;this is timer interrupt vector address
        JSR     TASK           ;go and execute task (long interrupt)
        ORG     P:MAIN_BODY
        MOVEP   #$000042,X:TCSR    ;enable timer interrupts and
                                   ;enable GPIO (input!) and set
                                   ;DO =0 to have stable data
        BSET    #DIR,X:TCSR        ;change DIR to output
                                   ;(clean 0, no spikes)
        MOVEP   #$00FFFF,X:TCR     ;load 64k -1 into the counter
        BSET    #IPL,X:IPR         ;enable IPL for timer
        ANDI    #$CF,MR            ;remove interrupt masking
                                   ;in status register
        BSET    #TE,X:TCSR         ; timer enable
......
; application program
.....
task
.....
; task instructions
....
end_of_task
        BSET    #DO,X:TCSR         ;set TIO to signal end of task
        BCLR    #DO,X:TCSR         ;clear TIO
        RTI                        ;return to main program
```

Figure 8-17 Standard Timer Mode with Simultaneous GPIO Program

8.8.4 Pulse Width Measurement Mode (Timer Mode 4)

The following program (see Figure 8-18) illustrates the use of the timer module for input pulse width measurement. The width is measured in this example for the low active period of the input pulse on the TIO pin and is stored in a table (in multiples of the chip operating clock divided by 2).

```
              ORG  X:$100     ;define buffer in X memory internal
pulse_width   DS   $100       ;measure up to 256 pulses

              ORG  P:$3C      ;this is timer interrupt
                              ;vector address
              MOVEP X:TCR,X:(r0)+   ;store width value in table
              NOP             ;second word of the short interrupt
...
              ORG  P:MAIN_BODY
...
              MOVE #PULSE_WIDTH,r0  ;r0 points to start of table
              MOVE #$FF,M0          ;modulo 100 to wrap around on
                                    ;end of table
              MOVEP #$000026,X:TCSR ;enable timer interrupts,
                                    ;mode 4 and set INV to
                                    ;measure the low active pulse
              BSET #IPL,X:IPR       ;enable IPL for timer
              ANDI  #$CF,MR         ;remove interrupt masking in
                                    ;status register
              BSET #TE,X:TCSR       ;timer enable
...
; do other tasks
...
```

Figure 8-18 Input Pulse Width Measurement Program

8.8.5 Period Measurement Mode (Timer Mode 5)

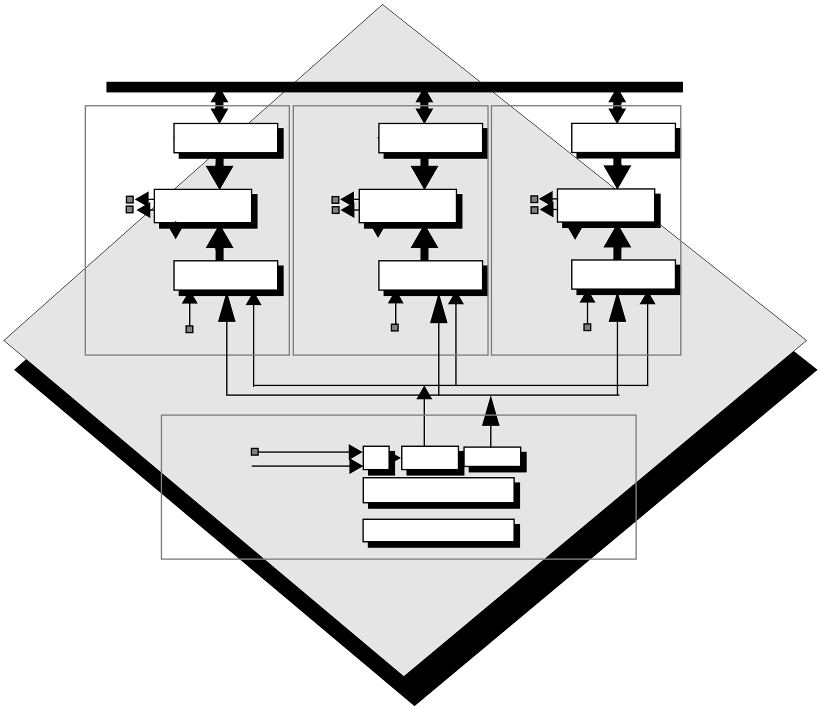
The following program (see Figure 8-19) illustrates the use of the timer module for input period measurement. The period is measured in this example between 0 to 1 transitions of the input signal on TIO and is stored in a table (in multiples of the chip operating clock divided by 2).

```
         ORG     X:$100         ;define buffer in X memory internal
period   DS     $100            ;measure up to 256 pulses
temp     DS     $1              ;temporary storage
         ORG    P:$3C     ;this is timer interrupt vector address
         JSR    MEASURE         ;long interrupt to measure period
....
         ORG    P:MAIN_BODY
.....
         MOVE    #0,X:TEMP      ;clear temporary storage
         MOVE    #PERIOD,r0     ;r0 points to start of table
         MOVE   #$FF,M0   ;modulo 100 to wrap around on end of table
         MOVEP  #$00002A,X:TCSR   ;enable timer interrupts, mode 5
         BSET   #IPL,X:IPR      ;enable IPL for timer
         ANDI   #$CF,MR         ;remove interrupt masking in
                                ;status register
         BSET   #TE,X:TCSR      ;timer enable
......
; do other tasks
.....
measure
         MOVEP  X:TCR,A         ;read new counter value
         MOVE   X:TEMP,X0       ;retrieve former read value
                                ;(initially zero)
         SUB    X0,A    A,X:TEMP  ;compute delta (i.e. new -old)
                                  ;and store the
                                  ;new read value in temp
         MOVE   A,X:(R0)+       ;store period value in table
         RTI
```

Figure 8-19 Input Period Measurement Program

SECTION 9

PULSE WIDTH MODULATORS



SECTION CONTENTS

9.1 INTRODUCTION

The Pulse Width Modulator (PWM) module uses two different blocks:

- The PWMA block, which is a 16-bit signed data pulse width modulator
- The PWMB block, which is a 16-bit positive fractional data pulse width modulator.

The Pulse Width Modulator module consists of three PWMA blocks, two PWMB blocks, as well as their associated pins and clock prescaler blocks.

The following is a list of the PWMA features:

- Programmable width from 9-bit to 16-bit signed two's complement fractional data
- Internal or external clock
- Internal or external carrier
- Maximum clock rate equal to 1/2 of the DSP core clock rate
- Four Interrupt Vectors

The following is list of the PWMB features:

- Programmable width from 9-bit to 16-bit positive fractional data
- Internal or external clock
- Internal or external carrier
- Maximum clock rate equal to 1/2 of the DSP core clock rate
- Three Interrupt Vectors

9.2 PULSE WIDTH MODULATOR INTERNAL ARCHITECTURE

The Pulse Width Modulator module includes three PWMA blocks and two PWMB blocks. The 56kCORE views each block as a memory mapped peripheral occupying one 16-bit word in the X data memory space and four additional 16-bit words (two of them shared by all of the PWMA blocks and the other two shared by all of the PWMB blocks). The 56kCORE may use the pulse width modulator as a normal memory mapped peripheral using standard polled or interrupt programming techniques.

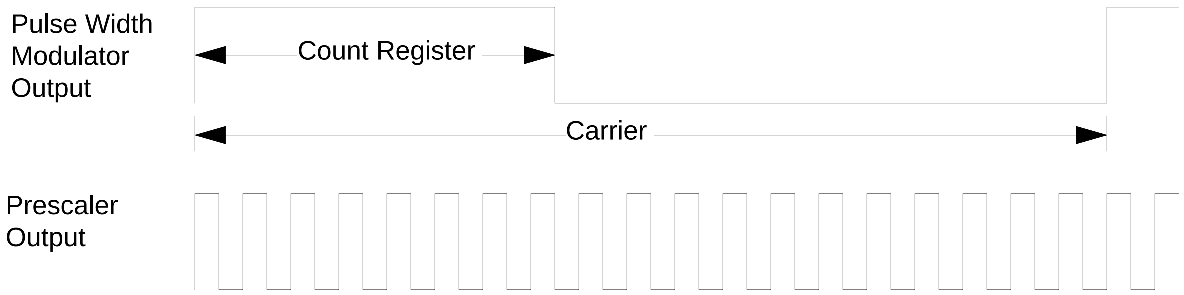


Figure 9-1 Pulse Width Modulator Waveform Controls

Pulses from the PWMn blocks are generated in the following way (see Figure 9-1):

1. either an external or internal carrier controls the period of the PWMn output i.e. from rising edge to rising edge
2. the count register (PWACRn or PWBCRn) is loaded with a number that will determine the pulse width i.e. from rising edge to falling edge
3. selection of the clock source and the number loaded into the prescaler determine the resolution of the pulse

9.2.1 Pulse Width Modulator A (PWMA) Overview

Figure 9-3 shows the internal architecture of PWMA.

9.2.1.1 PWMA Count Registers PWMA0, PWMA1, and PWMA2

Each one of the PWMA0, PWMA1, and PWMA2 blocks consists of:

- one 16-bit Count Register (PWACRn)
- one 16-bit Buffer Register (PWABUFn)
- one 15-bit Counter (PWACNn)
- one Comparator
- Control Logic which is responsible for generating the output pulses on the PWM pins, the interrupts, and the status bits

If the PWMAn count register (PWACRn) is loaded with two's complement fractional data from the 56KCORE through the Global Data Bus, then beginning at the rising edge of the carrier signal:

- this data will be transferred to the register buffer PWABUFn
- the 15-bit counter PWACNn will start incrementing
- the PWAPn or PWANn signal (according to the sign of the data — PWABUFn(23)) will be asserted

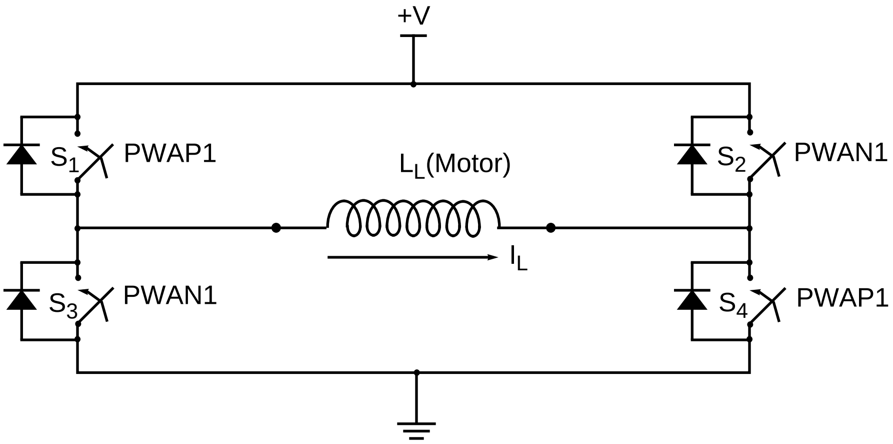


Figure 9-2 DC Motor Control Example Using Pulse Width Modulator A

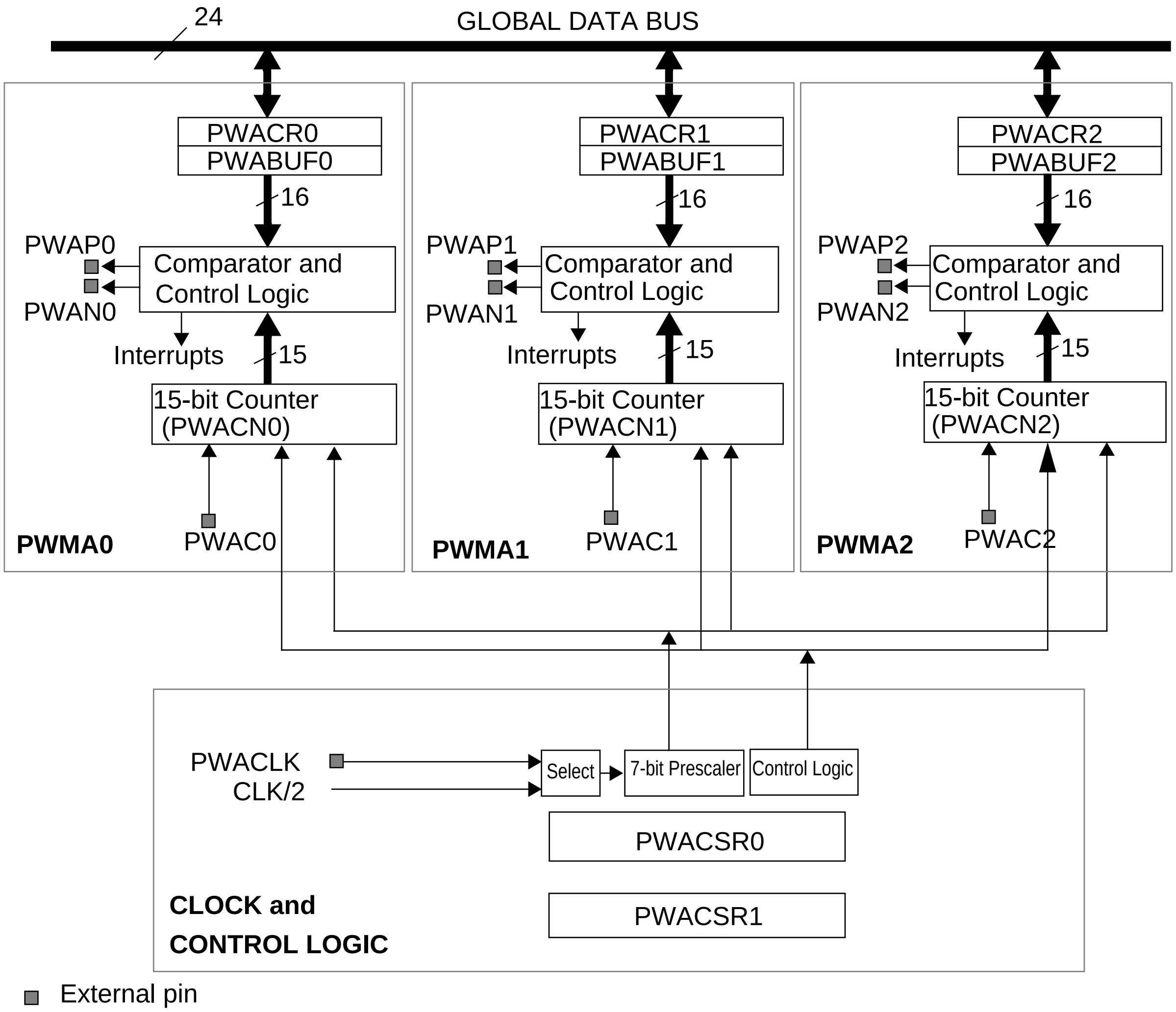


Figure 9-3 PWMA Block Diagram

When the comparator detects equality between the PWABUFn and the PWACNn value, the output signal (PWAPn or PWANn) is deasserted (see Figure 9-6 through Figure 9-6 for relative timing of the above events). Figure 9-1 shows a motor controlled by the PWAP1 and PWAN1 pins. When a positive number is loaded into PWACN1, the PWAP1 pin is driven closing switches S_1 and S_4 and creating a positive load current I_1. As the number in PWACN1 decreases to zero, the driving force decreases to zero. When a negative number is loaded into PWACN1, the PWA output switches to PWAN1 turning on switches S_2 and S_3, creating a negative I_L and thus driving the motor in the opposite direction.

Since fractional signed data representation is used, if less than 16-bit data is used, this data will be loaded as left-aligned in the PWMA Count Register (PWACRn). If, for example, the data width is 15-bit (i.e. 14-bit plus sign bit), then the bits WAW2:WAW1:WAW0 in the PWACSR0 should be written by the programmer with the value 0:0:1 and the Comparator will compare only the bits 22 through 9 of PWABUFn with the bits 0 through 13 of the PWACNn.

9.2.1.2 PWMA Clock and Control Logic

The clock used to increment the PWMA0, PWMA1, and PWMA2 counters may be:

- external, received through the PWACLK pin; in this case, the external clock is internally synchronized to the internal clock and enters the prescaler. Its frequency must be lower than the internal 56KCORE clock frequency divided by 2 (CLK/2). The maximum external clock frequency is given in the *DSP56003/005 Data Sheet.*
- internal, derived from the 56KCORE clock, after prescaling; the maximum clock rate for the counters is one half of the 56KCORE clock (CLK/2)

If the carrier signal is programmed as internal, then the internal signal which is equivalent to the “carrier signal rising edge” occurs in the following cases:

- when the counter wraps around (e.g. when PWACNn increments from $7FFF to 0)
- when this PWMAn module is enabled (WAEn=1) after having been previously cleared (WAEn=0)

If less than 16-bit fractional data is used, the counter wraps around according to the data width; e.g. if the data width is 15 (i.e. 14-bits plus sign bit), then the counter wraps around after it reaches $3FFF). The “width” of the counter is programmable allowing a width between 9 and 16 bits (i.e. the counter may wrap around when reaching a value from $FF to $7FFF, according to the value of the bits WAW(2:0) in PWACSR0).

9.2.2 Pulse Width Modulator B (PWMB) Overview

Figure 9-4 shows the internal architecture of PWMB

9.2.2.1 PWMB Count Registers PWMB0 and PWMB1

Each one of the PWMB0 and PWMB1 blocks consists of:

- one 16-bit Count Register (PWBCRn)
- one 16-bit Buffer Register (PWBBUFn)
- one Comparator
- control logic which is responsible for generating the output pulses on the PWM pins, the interrupts and the status bits

If the PWMBn count register (PWBCRn) is loaded with positive fractional data from the 56KCORE through the Global Data Bus, then beginning at the rising edge of the carrier signal:

- this data will be transferred to the register buffer PWBBUFn
- the PWBn signal will be asserted
- the 15-bit counter PWBCN will start incrementing

When the Comparator detects equality between the PWBBUFn and the PWBCN value, the output signal (PWBn) is deasserted (see Figure 9-3).

Since a fractional positive data representation is used, if less than 16-bit data is used, this data will be loaded as left-aligned in the PWMB Count Register (PWBCRn). If, for example, the data width is 15-bit (i.e. 14-bit plus sign bit), then the bits WBW2:WBW1:WBW0 in PWBCSR0 should be written by the programmer with the value 0:0:1 and the Comparator will compare only the bits 22 through 9 of PWBBUFn with the bits 0 through 13 of the PWBCN.

9.2.2.2 PWMB Clock and Control Logic

The clock which increments the counters of PWMB0 and PWMB1 (see Figure 9-4) may be:

- external, received through the PWBCLK pin; in this case, the external clock is internally synchronized to the internal clock and enters the prescaler. Its frequency must be lower than the internal 56KCORE clock frequency divided by 2 (CLK/2). The maximum external clock frequency is given in the *DSP56003/005 Data Sheet.*
- internal, derived from the 56KCORE clock after prescaling; the maximum clock rate for the counters is one half of the 56KCORE clock (CLK/2)

If the carrier signal is programmed as internal, then the internal signal which is equivalent to the “carrier signal rising edge” occurs in the following cases:

- when the counter wraps around (e.g. when PWBCN increments from $7FFF to 0)
- when this PWMBn module is enabled (WBEn=1) after having been previously cleared (WBEn=0) while the second PWMBk module is disabled; if the second PWMBk module is enabled, then the next “carrier signal rising edge” occurs when the counter wraps around (e.g. when PWBCN increments from $7FFF to 0 — see Figure 9-4)

If less than 16-bit fractional data is used, the Counter should wrap around according to the data width; e.g. if the data width is 15 (i.e. 14-bit plus sign bit), then the Counter should wrap around after it reaches $3FFF). The “width” of the Counter is programmable allowing a width between 9 and 16 (i.e. the Counters may wrap around when reaching a value from $FF to $7FFF, according to the value of the bits WBW(2:0) in PWBCSR0).

The sign bit of the 16-bit fractional data word loaded in the PWMB count registers is ignored and PWMB operates assuming that this word is positive.

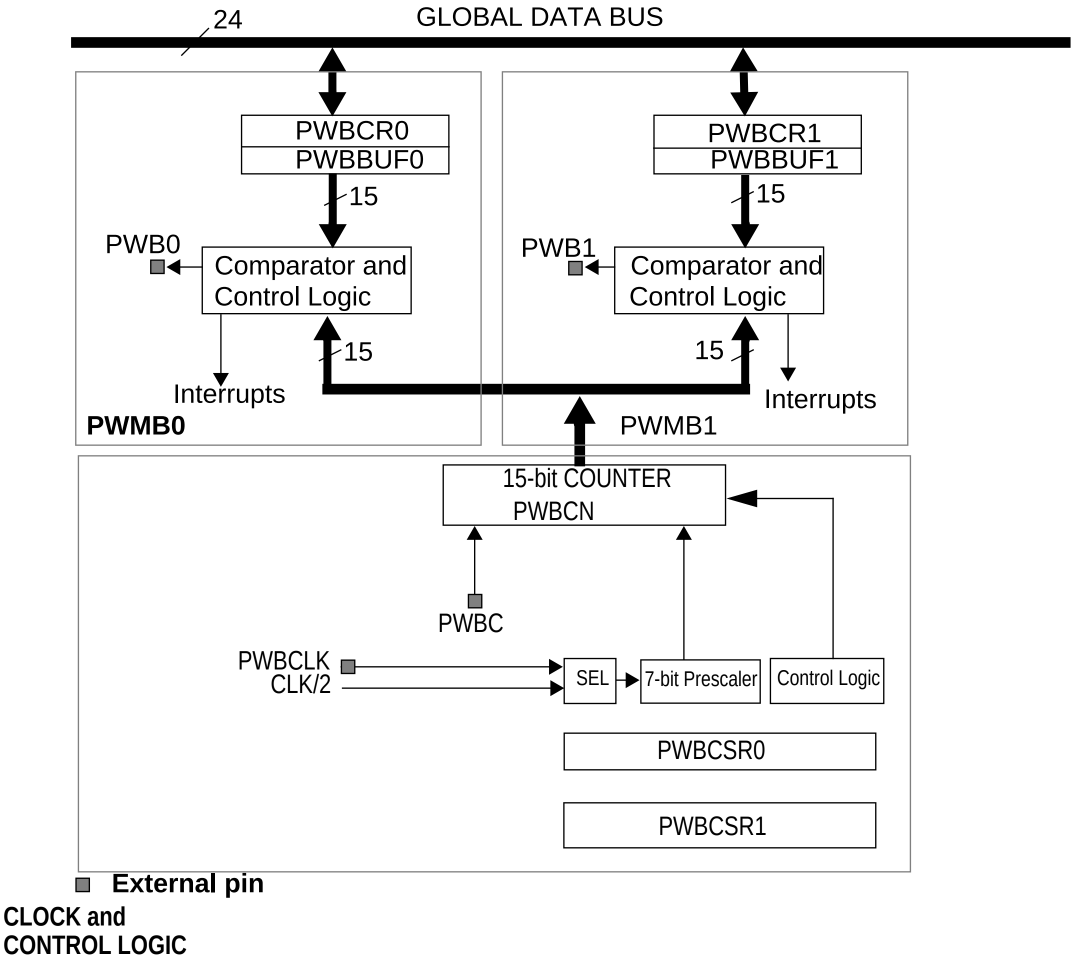


Figure 9-4 PWMB Block Diagram

9.3 PULSE WIDTH MODULATOR PROGRAMMING MODEL

The pulse width modulator registers which are available to the programmer are shown in Figure 9-5. These registers are described in the following paragraphs.

9.3.1 PWMAn Count Registers — PWACR0, PWACR1, and PWACR2

The PWACRn (n=0…2) count registers are 16-bit read / write registers. Data written to the PWACRn register is automatically transferred to the associated register buffer PWABUFn after the leading edge of the carrier signal PWACn or (when using an internal carrier) after the wrap around of the PWACNn counter.

9.3.2 PWMAn Control/Status Register 0 — PWACSR0

PWACSR0 is a 16-bit read / write register controlling the prescale rates of the PWM clocks, their sources and the PWM data width.The PWACSR0 status bits allow the DSP programmer to interrogate the PWMA status.

PWMA0 COUNT REGISTER (PWACR0)
X: $FFDA

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

PWMA1 COUNT REGISTER (PWACR1)
X: $FFDB

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

PWMA2 COUNT REGISTER (PWACR2)
X: $FFDC

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

PWMA CONTROL AND STATUS REGISTER 0 (PWACSR0)
X: $FFD9

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAR2	WAR1	WAR0	WAS2	WAS1	WAS0	▨	▨	▨	WAW2	WAW1	WAW0	WACK	WAP2	WAP1	WAP0

PWMA CONTROL AND STATUS REGISTER (PWACSR1)
X: $FFD8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAEI	▨	▨	▨	WAL2	WAL1	WAL0	WAC2	WAC1	WAC0	WAI2	WAI1	WAI0	WAE2	WAE1	WAE0

PWMB0 COUNT REGISTER (PWBCR0)
X: $FFD6

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

PWMB1 COUNT REGISTER (PWBCR1)
X: $FFD7

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

PWMB PRESCALER REGISTER (PWBCSR0)
X: $FFD5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WBR1	WBR0	WBS1	WBS0	▨	▨	▨	▨	▨	WBW2	WBW1	WBW0	WBCK	WBP2	WBP1	WBP0

PWMB CONTROL AND STATUS REGISTER (PWBCSR1)
X: $FFD4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WBEI	WBO	WBC	▨	▨	▨	▨	▨	▨	▨	▨	▨	WBI1	WBI0	WBE1	WBE0

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

Figure 9-5 PWM Programming Model

9.3.2.1 PWMAn Prescale (WAP0-WAP2) Bits 0-2

The read/write WAP0-WAP2 bits specify the PWMA prescale divide ratio. These bits specify any power of two prescale factor in the range from 2^0 to 2^7. The clock derived from the 56KCORE clock (CLK/2) or driven from the PWACLK pin is divided according to this prescale factor.Table 9-1 shows the programming of the WAP0-WAP2 bits. These bits are cleared (prescale by one) after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WAP0-WAP2 bits should be changed only when all of the PWMA blocks are disabled to ensure proper operation.

Table 9-1 Prescale Factor Bits WAP0-WAP2

WAP2-WAP0	Prescale Factor
$0	2^0
$1	2^1
$2	2^2
$3	2^3
$4	2^4
$5	2^5
$6	2^6
$7	2^7

9.3.2.2 PWMAn Clock Source (WACK) Bit 3

The read/write WACK bit specifies the clock source for the 7-bit clock prescaler. When WACK is set, the prescaler clock is driven from the internal 56KCORE CLK/2. When WACK is cleared, the prescaler clock is driven from the external clock fed through the PWACLK pin. This bit is cleared (external clock) after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: WACK should be changed only when all of the PWMA blocks are disabled to ensure proper operation.

9.3.2.3 PWMAn Data Width (WAW0-WAW2) Bits 4-6

The read/write WAW0-WAW2 bits specify the PWMA data width. These bits specify data widths from 9 to 16 bits in length.

Note: The data representation remains left-aligned, as a fractional number regardless of the value of WAW0-WAW2.

Table 9-1 shows the programming of the WAW0-WAW2 bits. These bits are cleared (16-bit data width) after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WAW0-WAW2 bits should be changed only when all of the PWMA blocks are disabled to ensure proper operation.

Table 9-2 Data Width Bits WAW0-WAW2

WAW2-WAW0	Data Width
$0	16
$1	15
$2	14
$3	13
$4	12
$5	11
$6	10
$7	9

9.3.2.4 PWMAn PWACSR0 Reserved Bits 7-9

Bits 7-9 in the PWACSR0 are reserved and unused. They read as zero and should be written with zero for future compatibility.

9.3.2.5 PWMAn Status (WAS0-WAS2) Bits 10-12

The read-only status bit WASn (n=0…2) is set when the data from PWMAn count register (PWACRn) is transferred to the PWMAn buffer register (PWABUFn). The WASn status bit is cleared when the PWMAn Count Register (PWACRn) is written with new data. The WASn bit is set after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction). The user program may test this bit to see if the count register (PWACRn) may be loaded with new data.

9.3.2.6 PWMAn Error (WAR0-WAR2) Bits 13-15

The read-only status bit WARn (n=0…2) is set when an error condition occurs in PWMAn, e.g. when a carrier signal rising edge occurs before the PWMAn comparator detected equality between the PWACRn and PWACNn registers. The WARn status bit is cleared when PWMAn is disabled (WAEn cleared). The WARn bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

9.3.3 PWMA Control/Status Register 1 (PWACSR1)

The PWACSR1 is a 16-bit read/write control/status register used to direct operation of the PWMA. The PWACSR1 control bits enable/disable the PWMA0, PWMA1, and PWMA2:

- registers
- interrupts
- carrier signal source
- PWMA output pin polarity.

The PWACSR1 bits are described in the following paragraphs.

9.3.3.1 PWACSR1 PWMAn Enable (WAEn) Bits 0-2

The read/write control bit WAEn (n=0…2) enables/disables the operation of PWMAn. When WAEn is set, PWMAn is enabled. When WAEn is cleared, PWMAn is disabled and in the personal reset state. This bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

9.3.3.2 PWACSR1 PWMAn Interrupt Enable (WAIn) Bits 3-5

The read/write control bit WAIn (n=0…2) enables/disables the interrupts from PWMAn. When WAIn is set, an interrupt (PWMAn interrupt) is generated after the data is transferred from the PWMAn Count Register (PWACRn) to the PWMAn Buffer Register (PWABUFn), i.e after the occurrence of a new carrier signal edge. When WAIn is cleared, this interrupt is disabled. The WAIn bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: After being serviced, a PWMAn interrupt will be cleared only if the respective status bit (WASn) has been cleared. WASn is cleared by a write to PWACRn or reset. A PWMAn interrupt will not be cleared unless there has been a write to PWACRn or a reset.

9.3.3.3 PWACSR1 PWMAn Carrier Select (WACn) Bits 6-8

The read/write control bit WACn (n=0…2) selects between the external and internal carrier for PWMAn. When WACn is set, PWMAn carrier is driven internally. The internal carrier signal is asserted every PWACNn wrap around. This wrap around may occur at different count values, according to the data width programed in the bits WAW0-WAW2 of PWACSR0. Note that since the internal carrier can be software controlled, the period of the PWM signal (rising edge to rising edge) can be controlled or modulated independently from the pulse width controlled by the count register (rising edge to falling edge). When WACn is cleared, the PWMAn carrier is driven from the PWACn pin. This bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WACn bit should be changed only when the WAEn bit is cleared (i.e., the PWMAn block is disabled) to ensure proper operation.

9.3.3.4 PWACSR1 PWMAn Output Polarity (WALn) Bits 9-11

The read/write control bit WALn (n=0…2) selects the polarity of the PWAPn and PWANn pins. When WALn is cleared, PWAPn and PWANn are active-low outputs. When WALn is set, PWAPn and PWANn are active-high outputs. This bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WALn bit should be changed only when the WAEn bit is cleared (i.e., the PWMAn block is disabled) to ensure proper operation.

9.3.3.5 PWACSR1 Reserved Bits 12-14

Bits 12-14 in PWACSR1 are reserved and unused. They are read as zero and should be written with zero for future compatibility.

9.3.3.6 PWACSR1 PWMA Error Interrupt Enable (WAEI) Bit 15

The read/write control bit WAEI enables/disables the error interrupt from PWMA. When WAEI is set and an error condition occurs, the PWMA error interrupt is generated. When WAEI is cleared, this interrupt is disabled. When an error interrupt occurs, the user's program should test all of the PWMAn Error bits (WAR0, WAR1 and WAR2) and the PWMBn Error bits (WBR0 and WBR1) in order to find out whether the PWMAn or the PWMBn block generated the error. The WAEI bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

9.3.4 PWMB Count Registers — PWBCR0, PWBCR1

The PWBCR0 and PWBCR1 count registers are 16-bit read/write registers. Data written to these registers is automatically transferred to the associated register buffer after the leading edge of the carrier signal or (when using internal carrier) after the PWBCN counter wraps around.

9.3.5 PWMB Control/Status Register 0 — PWBCSR0

The PWBCSR0 is a 16-bit read/write register controlling the prescale rates of the PWMB clock, its source and the PWMB data width.The PWBCSR0 status bits allow the DSP programmer to interrogate the PWMB status.

9.3.5.1 PWBCSR0 PWMB Prescale (WBP0- WBP2) Bits 0-2

The read/write WBP0-WBP2 bits specify the divide ratio of the PWMB prescale divider. These bits specify any power of two prescale factor in the range from 2^0 to 2^7. The clock derived from the 56KCORE clock (CLK/2) or driven from the PWBCLK pin is divided according to this prescale factor. Table 9-1 shows the programming of the WBP0-WBP2 bits. These bits are cleared (prescale by one) after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WBP0-WBP2 bits should be changed only when all of the PWMB blocks are disabled to ensure proper operation.

Table 9-3 Prescale Factor Bits WBP0-WBP2

WBP2-WBP0	Prescale Factor
\$0	2^0
\$1	2^1
\$2	2^2
\$3	2^3
\$4	2^4
\$5	2^5
\$6	2^6
\$7	2^7

9.3.5.2 PWBCSR0 PWMB Clock Source (WBCK) Bit 3

The read/write WBCK bit specifies the clock source for the 7-bit clock prescaler. When WBCK is set, the prescaler clock is driven from the internal 56KCORE CLK/2. When WBCK is cleared, the prescaler clock is driven from the external clock fed through the PWBCLK pin. This bit is cleared (external clock) after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WBCK should be changed only when all of the PWMB blocks are disabled to ensure proper operation.

9.3.5.3 PWBCSR0 PWMB Data Width (WBW0-WBW2) Bits 4-6

The read/write WBW0-WBW2 bits specify the PWMB data width. These bits specify any data width in the range from 9 bits to 16 bits. The data representation remains left-aligned, as a fractional number regardless of the values of these bits.

Table 9-1 shows the programming of the three WBW0-WBW2 bits. These bits are cleared (16-bit data width) after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WBW0-WBW2 bits should be changed only when all of the PWMB blocks are disabled to ensure proper operation.

Table 9-4 Data Width Bits WBW2-WBW0

WBW2-WBW0	Data Width
$0	16
$1	15
$2	14
$3	13
$4	12
$5	11
$6	10
$7	9

9.3.5.4 PWBCSR0 Reserved Bits 7-11

Bits 7-11 in PWBCSR0 are reserved and unused. They read as zero and should be written with zero for future compatibility.

9.3.5.5 PWBCSR0 PWMBn Status (WBSn) Bits 12-13

The read-only status bit WBSn (n=0…1) is set when data from the PWMBn count register (PWBCRn) is transferred to the PWMBn buffer register (PWBBUFn). The WBSn status bit is cleared when the PWMBn Count Register (PWBCRn) is written with new data. This bit is set after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction). The user program may test this bit in order to tell if the count register (PWACRn) may be loaded with new data.

9.3.5.6 PWBCSR0 PWMBn Error (WBRn) Bit 14-15

The read-only status bit WBRn (n=0…1) is set when an error condition occurs in the PWMBn, e.g. when a new rising edge of the carrier signal occurs before the PWMBn comparator detects equality between the PWBCRn and PWBCNn. The WBRn status bit is cleared when PWMBn is disabled (WBEn cleared). The WBRn bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

9.3.6 PWMB Control/Status Register 1 — PWBCSR1

The PWBCSR1 is a 16-bit read/write control/status register used to direct the PWMB operation. The PWBCSR1 control bits enable/disable the PWMB:

- registers
- interrupts
- carrier signal source

The PWBCSR1 bits are described in the following paragraphs.

9.3.6.1 PWBCSR1 PWMBn Enable (WBEn) Bits 0-1

The read/write control bit WBEn (n=0…1) enables/disables operation of the PWMBn. When WBEn is set, PWMBn is enabled. When WBEn is cleared, PWMBn is disabled and in the personal reset state. This bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

9.3.6.2 PWBCSR1 PWMBn Interrupt Enable (WBIn) Bits 2-3

The read/write control bit WBIn (n=0…1) enables/disables interrupts from PWMBn. When WBIn is set, an interrupt (PWMBn interrupt) is generated after data is transferred from the PWMBn Count Register (PWBCRn) to the PWMBn Buffer Register (PWBBUFn). When WBIn is cleared, this interrupt is disabled. The WBIn bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: After being serviced, a PWMBn interrupt will be cleared only if the respective status bit (WBSn) has been cleared. WBSn is cleared by a write to PWBCRn or reset. A PWMBn interrupt will not be cleared unless there has been a write to PWBCRn or a reset.

9.3.6.3 PWBCSR1 Reserved Bits 4-12

Bits 4-12 in PWBCSR1 are reserved and unused. They read as zero and should be written with zero for future compatibility.

9.3.6.4 PWBCSR1 PWMB Carrier Select (WBC) Bit 13

The read/write control bit WBC selects between the external and internal carrier for PWMB. When WBC is set, PWMB carrier is driven internally. The internal carrier signal is asserted every PWBCN wrap around. This wrap around may occur at different count values, according to the data width programed in the bits WBW0-WBW2 of PWBCSR0. Note that since the internal carrier can be software controlled, the period of the PWM signal (rising edge to rising edge) can be controlled or modulated independently from the pulse width controlled by the count register (rising edge to falling edge). When WBC is cleared, PWMB carrier is driven from the PWBC pin. This bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

Note: The WBCn bit should be changed only when the WBEn bit is cleared (i.e., the PWMBn block is disabled) to ensure proper operation.

9.3.6.5 PWBCSR1 PWMB Open Drain Output (WBO) Bit 14

This read/write control bit configures the PWMB output pins ($\overline{\text{PWB0}}$ and $\overline{\text{PWB1}}$) as either open-drain pins or TTL level pins. When WBO is cleared, the open-drain configuration is forced on the PWMB output pins ($\overline{\text{PWB0}}$ and $\overline{\text{PWB1}}$). When WBO is set, these pins are TTL outputs. The WBO bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

9.3.6.6 PWBCSR1 PWMB Error Interrupt Enable (WBEI) Bit 15

The read/write control bit WBEI enables/disables the error interrupt from PWMB. When WBEI is set and an error condition occurs, a PWMB error interrupt is generated. When WBEI is cleared, this interrupt is disabled. When an error interrupt occurs, the user's program should test all the PWMAn Error bits (WAR0, WAR1 and WAR2) and the PWMBn Error bits (WBR0 and WBR1) in order to find out whether the PWMAn or the PWMBn block generated the error. The WBEI bit is cleared after hardware $\overline{\text{RESET}}$ or after a software reset (RESET instruction).

9.4 PULSE WIDTH MODULATOR FUNCTIONAL DESCRIPTION

This section shows timing diagram which illustrate the operation of the PWM blocks.

9.4.1 Timing Diagrams

Note that in Figure 9-6, the first assertion width for PWAPn is N, and the second is M (both are in units of PWACLK). PWMAn's output is active low and is sent to pin PWAPn because the PWABUFn sign bit is low.

Again, in Figure 9-7, N and M are the first and second PWM cycle values, respectively, and PWAPn is active low. However, in this instance the second edge of the carrier signal occurs before the end of the first PWM pulse. Hence, an error is flagged on WAEn.

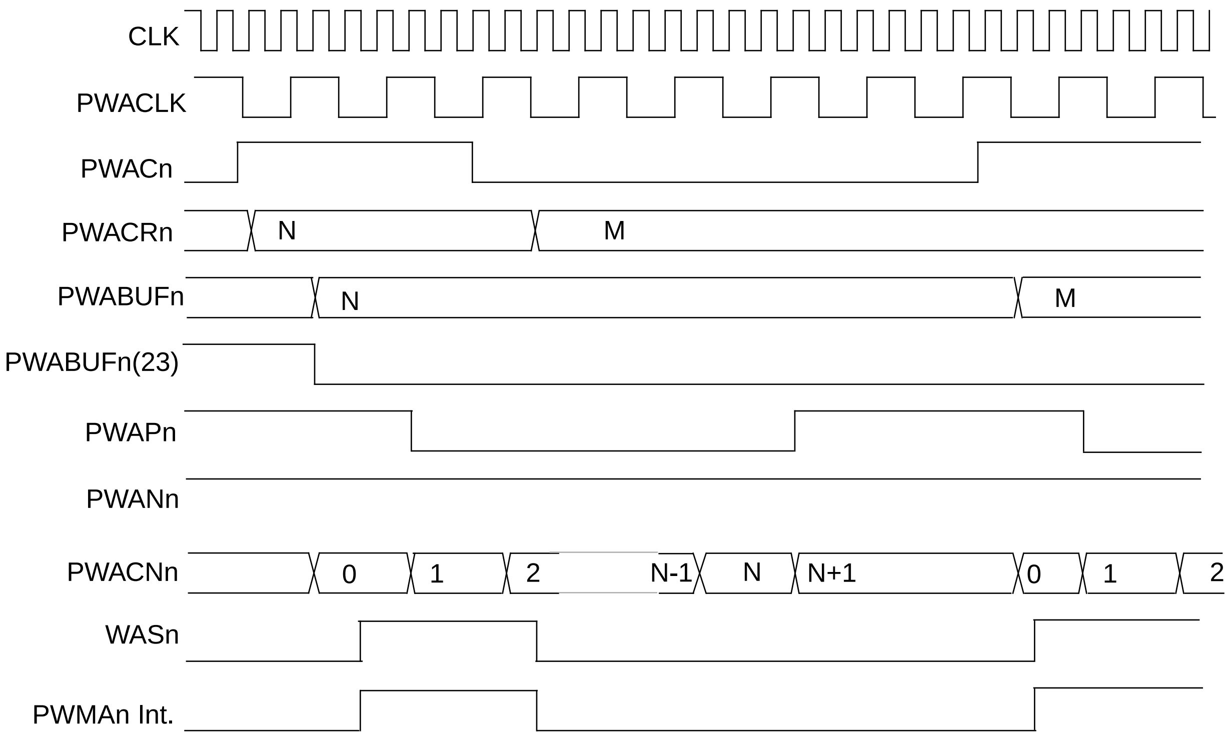


Figure 9-6 PWMA Timing — External Clock, External Carrier, Positive Data

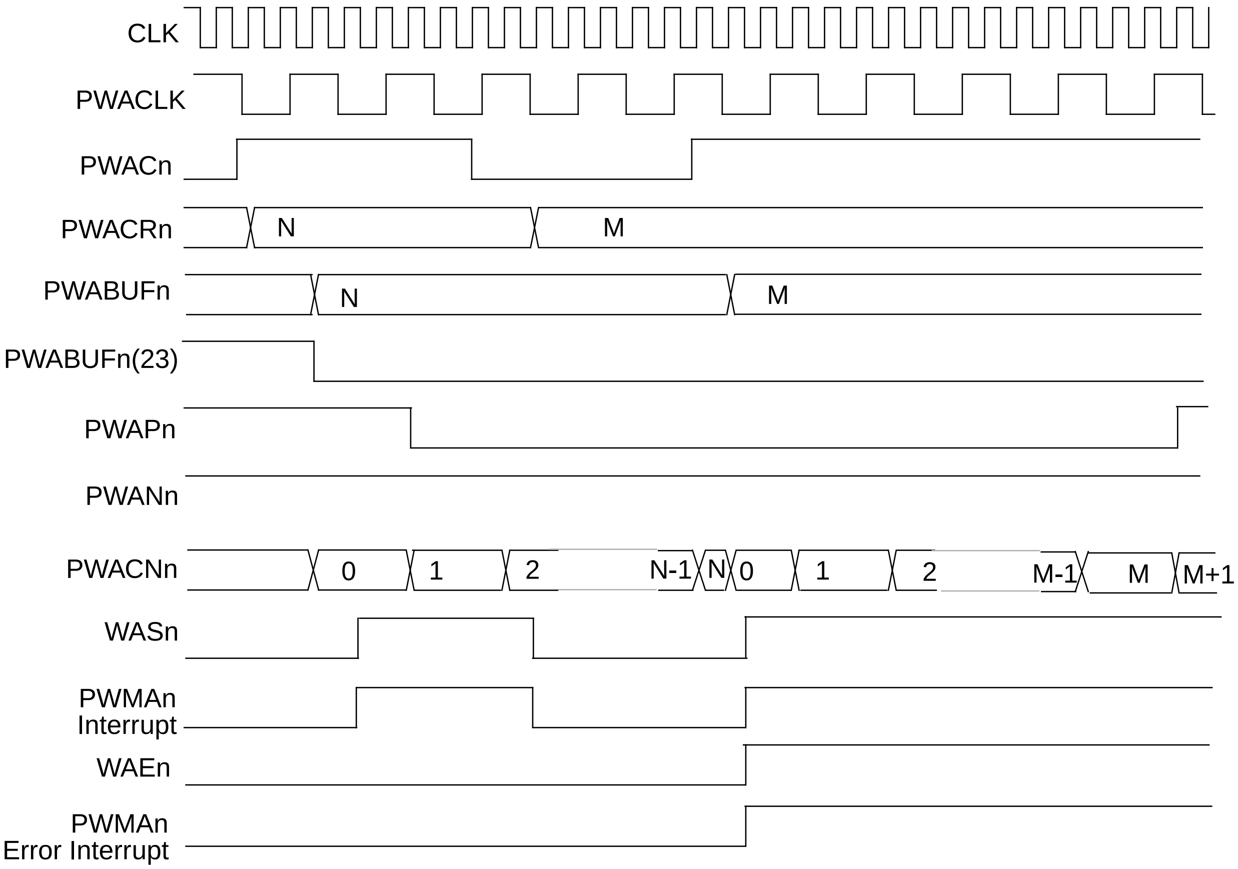


Figure 9-7 PWMA Timing — External Clock, External Carrier, Error

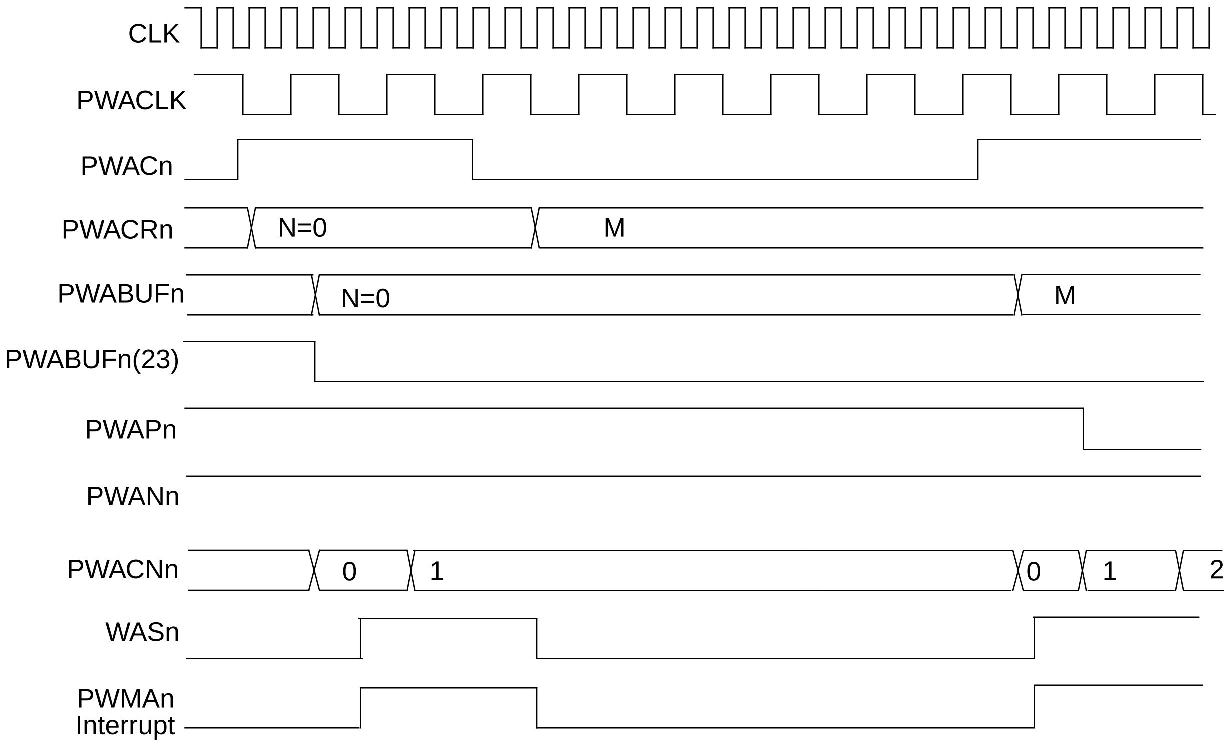


Figure 9-8 PWMA Timing — External Clock, External Carrier, N=0

Figure 9-8 shows the case where the width of the first PWM cycle is zero (N=0), and therefore the output is never driven low (active) for that cycle. The next width (M) is non-zero.

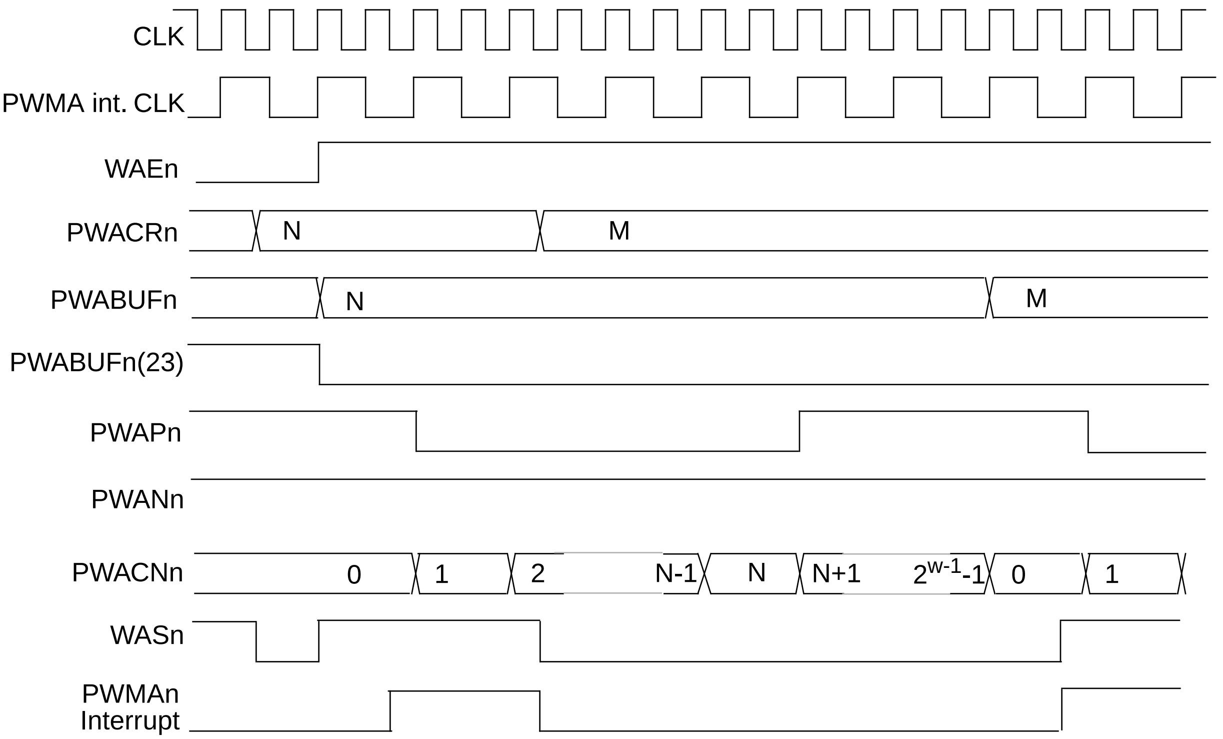


Figure 9-9 PWMA Timing — Internal Clock, Internal Carrier Width=w

In Figure 9-9, the PWM uses an internal carrier that has a rising edge whenever the counter (PWACNn) wraps around at the 2^wth count, where w is the width of the data as specified by WAW0-WAW2.

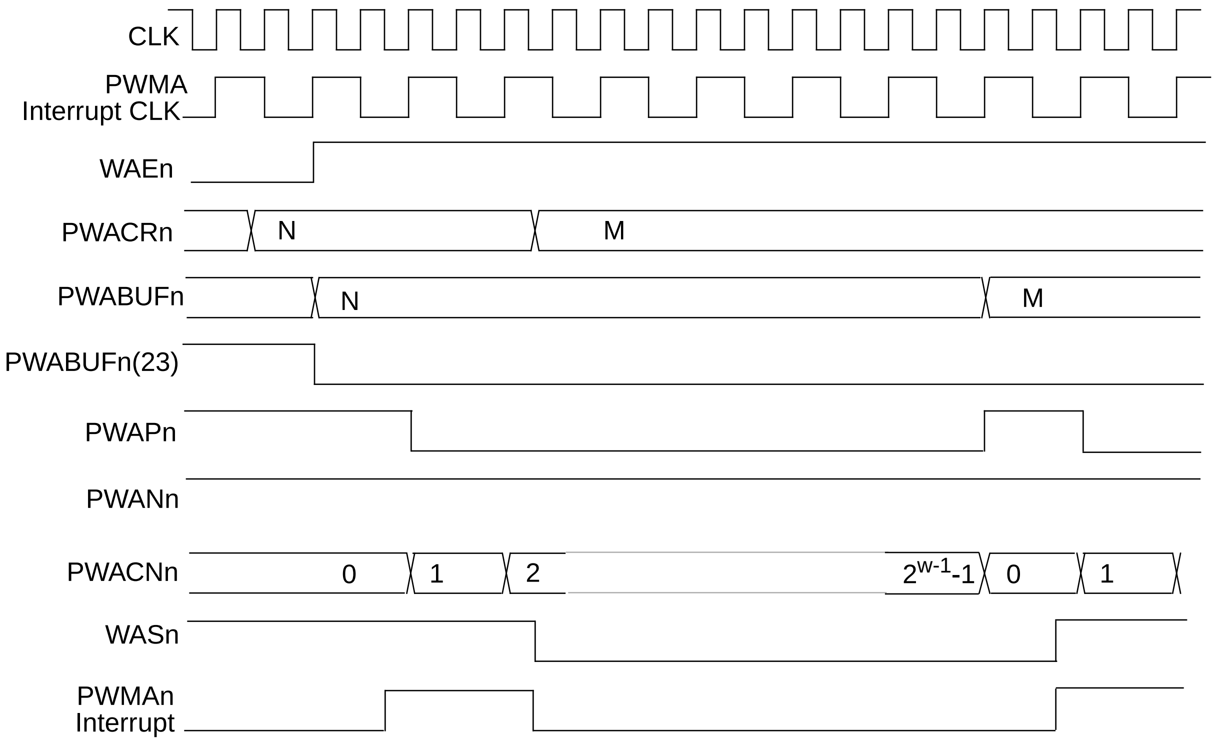


Figure 9-10 PWMA Timing — Internal Clock, Internal Carrier, N=$7FFF, w=16

Figure 9-10 shows the maximum pulse width that can be used for 16-bit positive two's complement data, $7FFF. Note that this value does allow the PWAPn pin to be deasserted for one PWMA clock cycle.

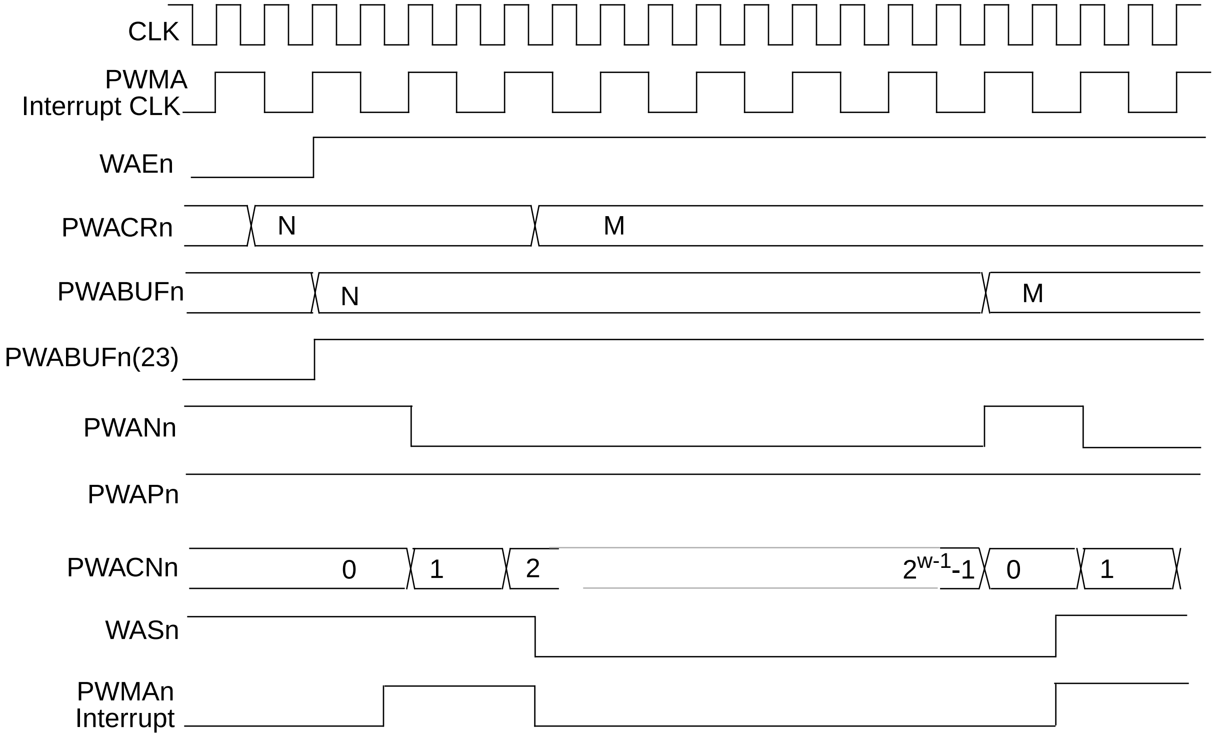


Figure 9-11 PWMA Timing — Internal Clock, Internal Carrier, N=$8001, w=16

Figure 9-11 shows the results of negating the count register value shown in Figure 9-10. The value becomes a 16-bit negative two's complement number, $8001. Note that the output is seen on the PWANn pin and is again deasserted for one PWMA clock cycle.

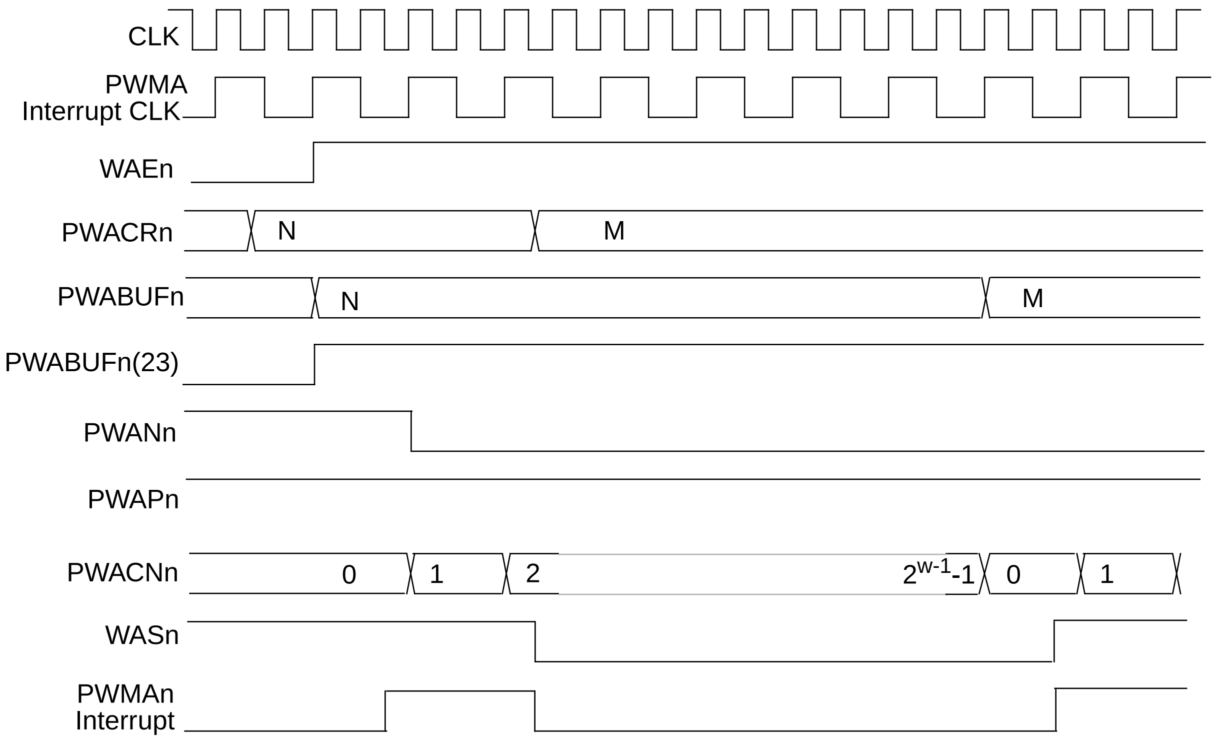


Figure 9-12 PWMA Timing — Internal Clock, Internal Carrier, N=$8000, w=16

The maximum pulse width that can be used for 16-bit negative two's complement data is obtained by writing $8000 to the counter register. Figure 9-12 shows the resulting signals. Note that once driven active (low in this case), the PWANn pin remains active yet avoids an error signal.

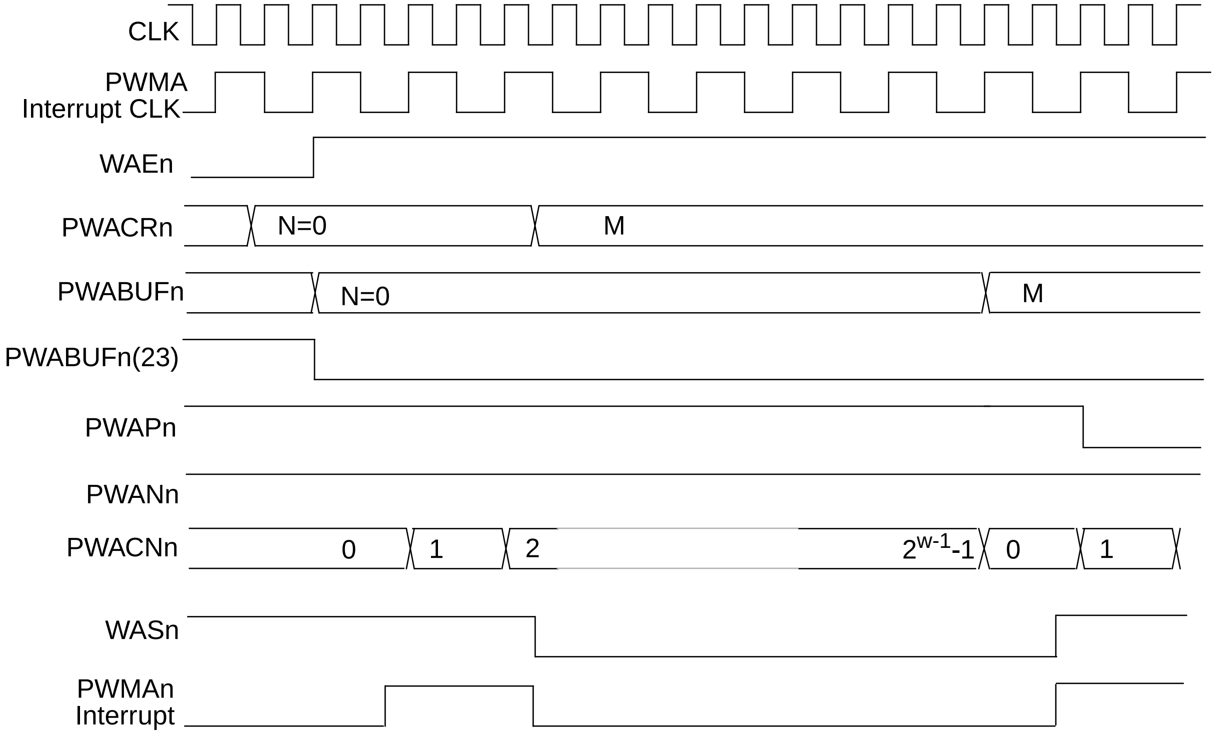
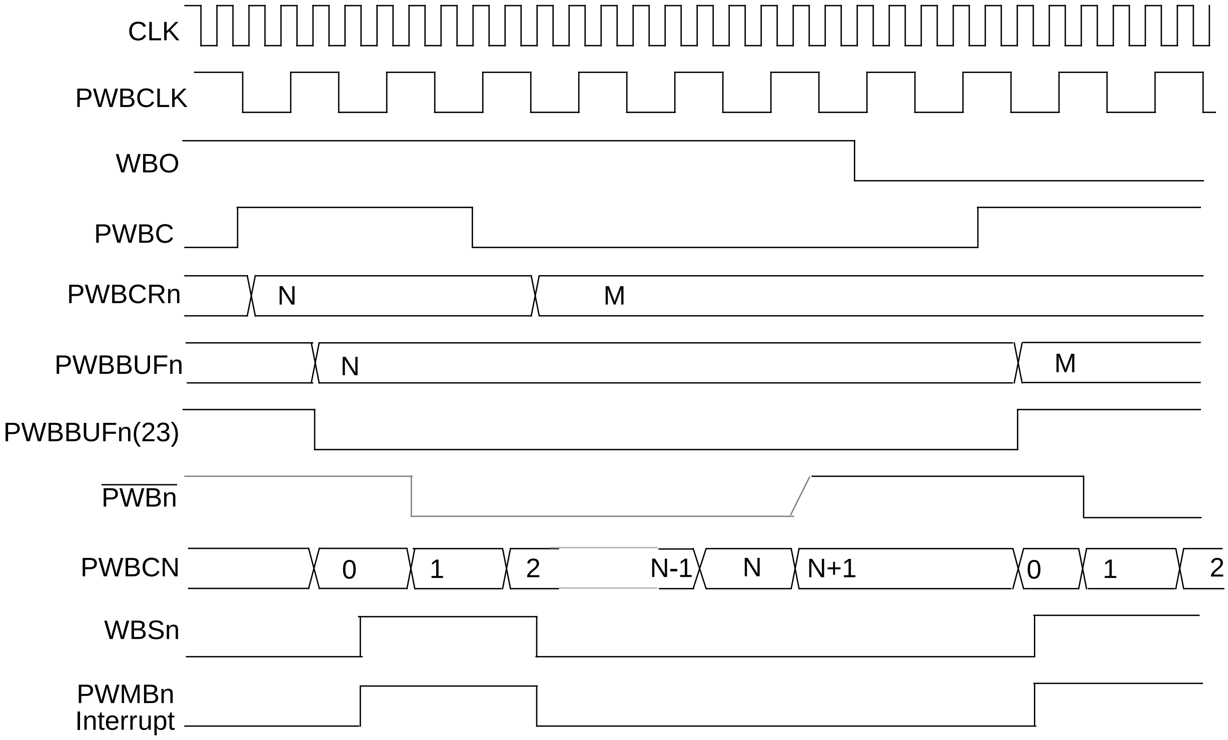


Figure 9-13 PWMA Timing — Internal Clock, Internal Carrier, N=0, w=16

Figure 9-13 again shows a PWMA channel configured for a zero pulse width, followed by a non-zero pulse width M. In Figure 9-13, however, the carrier and clock are both internal with a data width (w) of sixteen. Therefore a single PWM cycle lasts for 2^{16} PWM clock cycles, as opposed to that of Figure 9-8 which lasts until the next rising edge of the external carrier.



Denotes an Open-drain output; a pull-up resistor should be connected to this pin

Figure 9-14 PWMB Timing — External Clock, External Carrier

Figure 9-14 shows the timings for PWMB with a pulse width of N. Each block of PWMB has only a single output pin, $\overline{\text{PWBn}}$.

9.4.2 Boundary conditions

Due to synchronization between the external signals (Carrier, Clock) and the internal clock, there may be some uncertainty in the:

- delay between the external carrier assertion and the PWM output assertion
- delay between the external clock edges and the PWM output

For the same reasons, there might be synchronization delays between two PWMs even if they use the same external clock and the same external carrier. The maximum delay values are given in the *DSP56003/005 Data Sheet.*

There is no error condition when the PWM clock is internal.

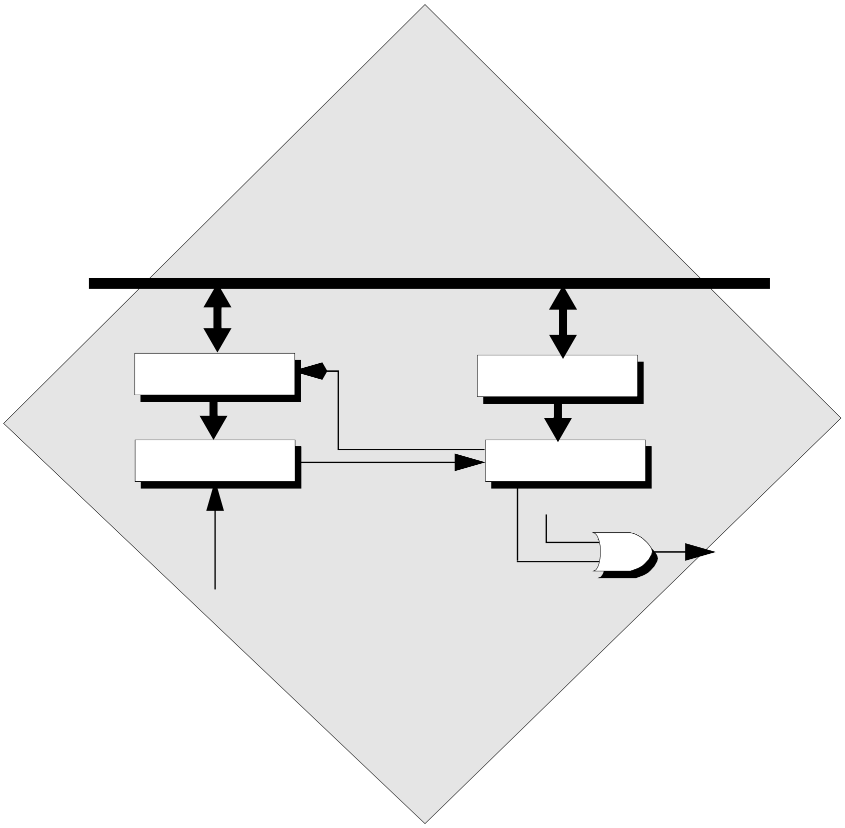
If the external carrier signal is asserted after deassertion of the output pin, then it guarantees no error.

If an error condition occurs in a PWM module due to premature assertion of the carrier signal of the module, then the output pin will remain asserted for a period determined by the data value in the respective count register. The respective status bit will be set due to this premature assertion of the carrier signal.

The minimum assertion and deassertion duration of the carrier signal are given in the *DSP56003/005 Data Sheet.*

SECTION 10

WATCHDOG TIMER



SECTION CONTENTS

10.1 INTRODUCTION

This section describes the Watchdog Timer module of the DSP56003/005. The Watchdog Timer can interrupt the DSP56003/005 after a specified number of clocks. It generates a Non-Maskable Interrupt (NMI) to the 56KCORE which has the same vector address as the NMI exception vector (P:$001E).

10.2 WATCHDOG TIMER ARCHITECTURE

Figure 10-1 shows a block diagram of the Watchdog Timer. It includes:

- 16-bit read-write Watchdog Timer Control/Status Register (WCSR)
- 16-bit read-write Watchdog Timer Count Register (WCR)
- 16-bit counter
- 7-bit clock prescaler
- logic for interrupt generation.

The DSP56003/005 views the Watchdog Timer as a memory mapped peripheral occupying two 16-bit words in the X data memory space. The programming model is shown in Figure 10-2.

10.2.1 Watchdog Timer Count Register (WCR)

The Count Register is a 16-bit read-write register which contains the value to be loaded into the counter. This counter is loaded with the value contained in the Count Register on three occasions:

- when the Watchdog Timer Enable bit is set (WE=1) after being previously cleared (WE=0)
- when the Watchdog Timer Load (WLD) bit is set while the Watchdog is enabled (WE=1)
- when the counter has been decremented to zero and a new watchdog clock occurs (WE=1)

In the last case, if the WCR is loaded with N, the counter will be reloaded after (N+1) watchdog clocks. The term “watchdog clock” refers to the output of the clock prescaler.

If the Watchdog Timer is disabled (WE=0) and the WCR is written by the user program, the value is stored in the WCR and will be loaded into the counter when the WE bit is set.

If the Watchdog Timer is enabled (WE=1) and the WCR is written by the user program, the value is stored in the WCR and will be loaded into the counter after the counter has been decremented to zero and a new watchdog clock occurs.

If the Watchdog Timer is enabled (WE=1) and the WLD bit is written with “one”, the WCR contents will be loaded into the counter regardless of the counter value at the moment.

10.2.2 Watchdog Timer Control/status Register (WCSR)

The Watchdog Timer Control/Status Register is a 16-bit read/write register that controls the Watchdog Timer and verifies its status. The WCSR can be accessed both by normal move instructions as well as by bit manipulation instructions. The control and status bits are described in the following paragraphs.

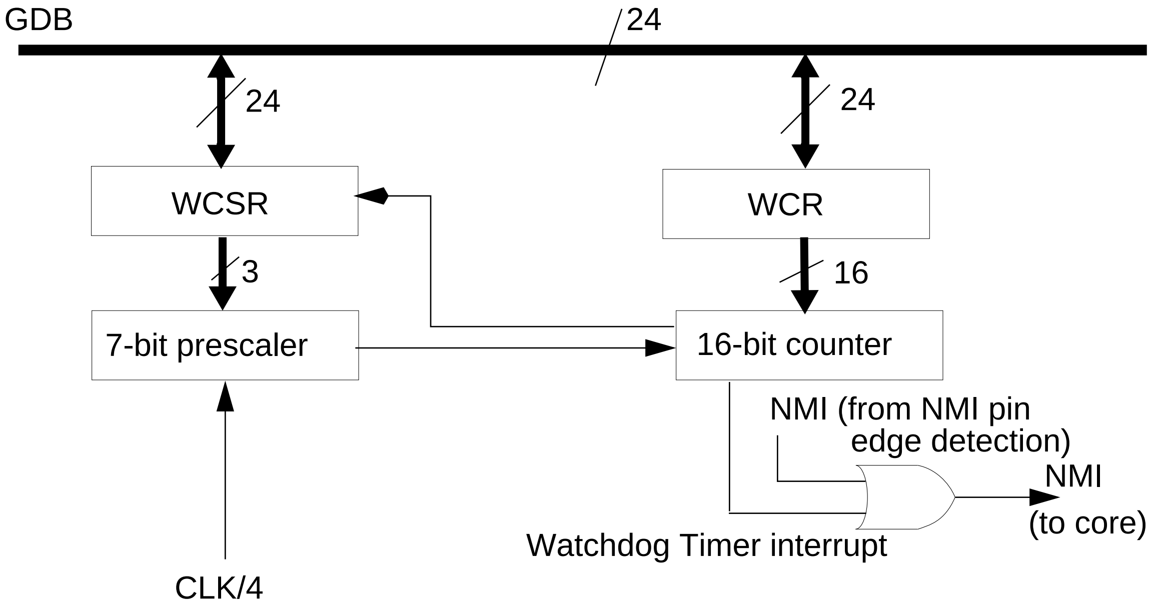


Figure 10-1 16-bit Timer Module Block Diagram

WATCHDOG TIMER COUNT REGISTER (WCR)
X: $FFE7

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	WD15	WD14	WD13	WD12	WD11	WD10	WD9	WD8	WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0

WATCHDOG TIMER CONTROL/STATUS REGISTER (WCSR)
X: $FFE6

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	WDB	WLD	WE	WIE	WS	WP2	WP1	WP0

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

Figure 10-2 Watchdog Timer Module Programming Model

10.2.2.1 WCSR Watchdog Timer Prescale (WP0-WP2) Bits 0-2

The Watchdog Timer Prescale bits (WP2-WP0) define the divide ratio of the prescale divider. These bits specify any power of two prescale factor in the range from 2^0 to 2^7. Table 10-1 shows the programming of the WP0-WP2 bits. These bits are cleared (prescale by one) after hardware $\overline{\text{RESET}}$ or after software reset (RESET instruction).

Table 10-1 Prescale Factor Bits WP0-WP2

WP2-WP0	Prescale Factor
$0	2^0
$1	2^1
$2	2^2
$3	2^3
$4	2^4
$5	2^5
$6	2^6
$7	2^7

Note: The WP0-WP2 bits may be changed at any time, but the 7-bit Prescaler will be loaded according to their value only in the following three cases:

- when the Watchdog Timer Enable bit is set (WE=1) after being previously cleared (WE=0)
- when the WLD bit is set, while the Watchdog Timer is enabled (WE=1)
- after the counter has been decremented to zero and a new watchdog clock occurs (WE=1)

10.2.2.2 WCSR Watchdog Timer status (WS) Bit 3

The Watchdog Timer status bit, when set, indicates that the counter has been decremented to zero. The Watchdog Timer status bit is cleared when the WCSR is read. WS is also cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction).

10.2.2.3 WCSR Watchdog Timer Interrupt Enable (WIE) Bit 4

The Watchdog Timer Interrupt Enable bit is used to enable the Watchdog Timer interrupts after the counter reaches zero and a new watchdog clock occurs. If WCR is loaded with N, a non-maskable interrupt will occur after (N+1) watchdog clocks. Setting WIE (WIE=1) will enable the interrupts. The interrupts are disabled when WIE is cleared (WIE=0).

WIE is cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction).

10.2.2.4 WCSR Watchdog Timer Enable (WE) Bit 5

The Watchdog Timer Enable is used to enable or disable the timer. Setting the WE bit (WE=1) will:

- enable the Watchdog Timer
- load the value specified by WP0-WP2 (according to Table 10-1) into the 7-bit prescaler which has clk/4 as an input
- load the counter with the value contained in WCR and begin decrementing at each watchdog clock

Clearing the WE bit will disable the Watchdog Timer and freeze the prescaler and counter.

WE is cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction).

10.2.2.5 WCSR Watchdog Timer Load (WLD) Bit 6

The Watchdog Timer Load is used to reload the Watchdog Timer 16-bit counter and the 7-bit prescaler respectively with the values specified by the WCR and WCSR. Setting the WLD bit (WLD=1) will load the prescaler and the counter. The WLD bit will be immediately cleared by the internal hardware. Clearing the WLD bit will have no effect on the Watchdog Timer activity.

WLD is cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction).

Note: Due to delays in the internal pipeline, the user should allow a two-instruction delay between setting the WLD bit and attempting to write the WCR or WCSR registers with new values.

10.2.2.6 WCSR Watchdog Timer Debug (WDB) Bit 7

The Watchdog Timer Debug is used to freeze the Watchdog Timer 16-bit counter and the 7-bit prescaler during debug mode. Setting the WDB bit (WDB=1) will freeze the Watchdog Timer 16-bit counter and the 7-bit prescaler during Debug mode. Clearing the WDB bit (WDB=0) will allow the Watchdog Timer 16-bit counter and the 7-bit prescaler to continue their operation during the debug mode.

WDB is cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction).

Note: The WDB bit should be changed only when the Watchdog Timer is disabled to ensure proper functionality.

10.2.2.7 WCSR Reserved Bits 8-15

These reserved bits are read as zero and should be written with zero for future compatibility.

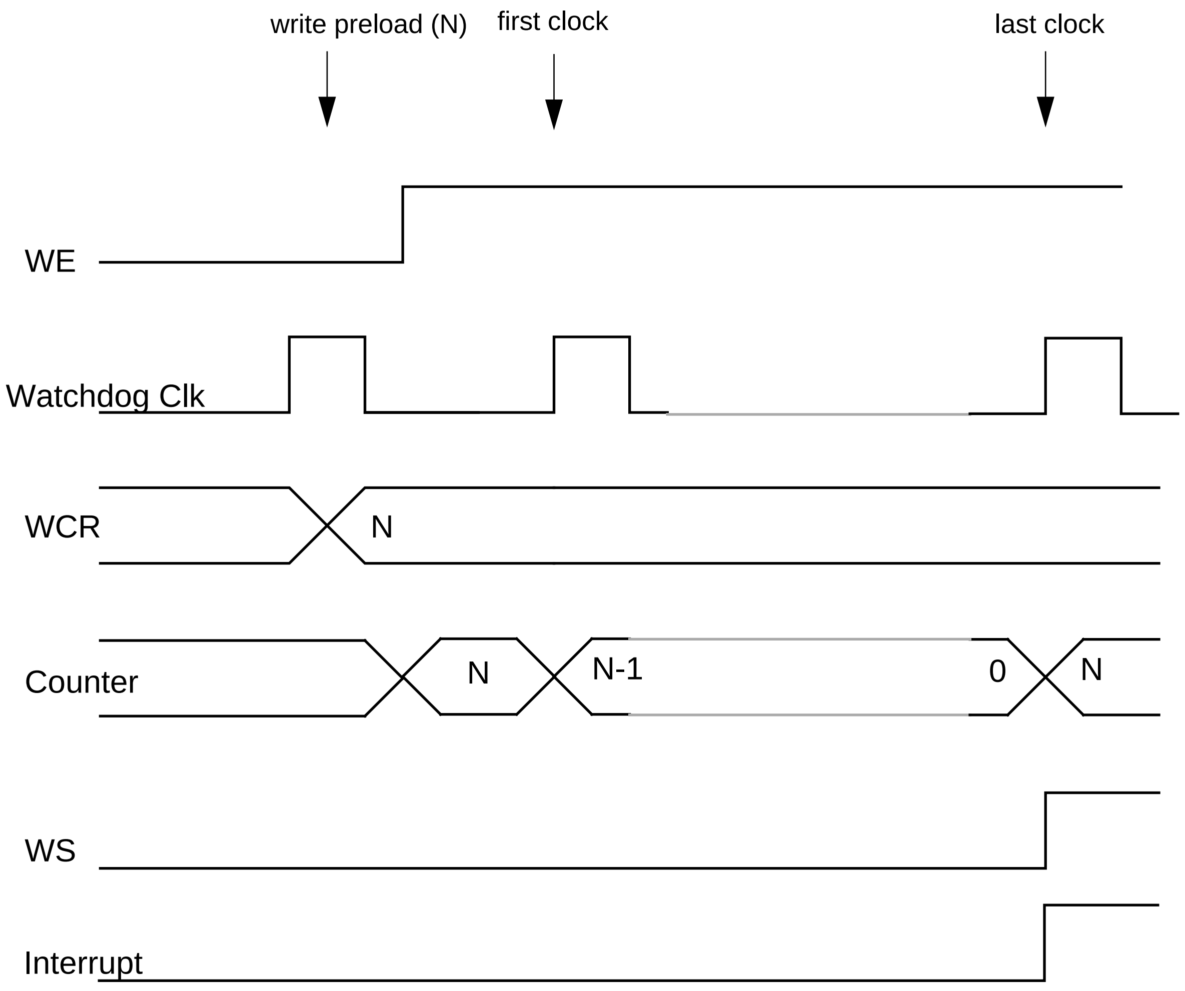


Figure 10-3 Watchdog Timer Interrupt

10.3 WATCHDOG TIMER FUNCTIONAL DESCRIPTION

The counter is loaded with the value contained by the WCR when WE=1 and the counter is decremented by the watchdog clock, which is one fourth the 56KCORE clock (CLK/4), after prescaling according to the prescale factor. At the next watchdog clock after the counter reaches zero, the WS bit in WCSR is set and, if the WIE is set, a non-maskable interrupt (NMI) is generated (see Figure 10-3). The interrupt signal generated by the Watchdog Timer is internally OR-ed with the NMI signal (generated through the MODC pin). The counter is reloaded with the value contained by the WCR and the entire process is repeated until the timer is disabled (WE=0). Figure 10-3 illustrates this mode. Figure 10-4 describes the Watchdog Timer disable.

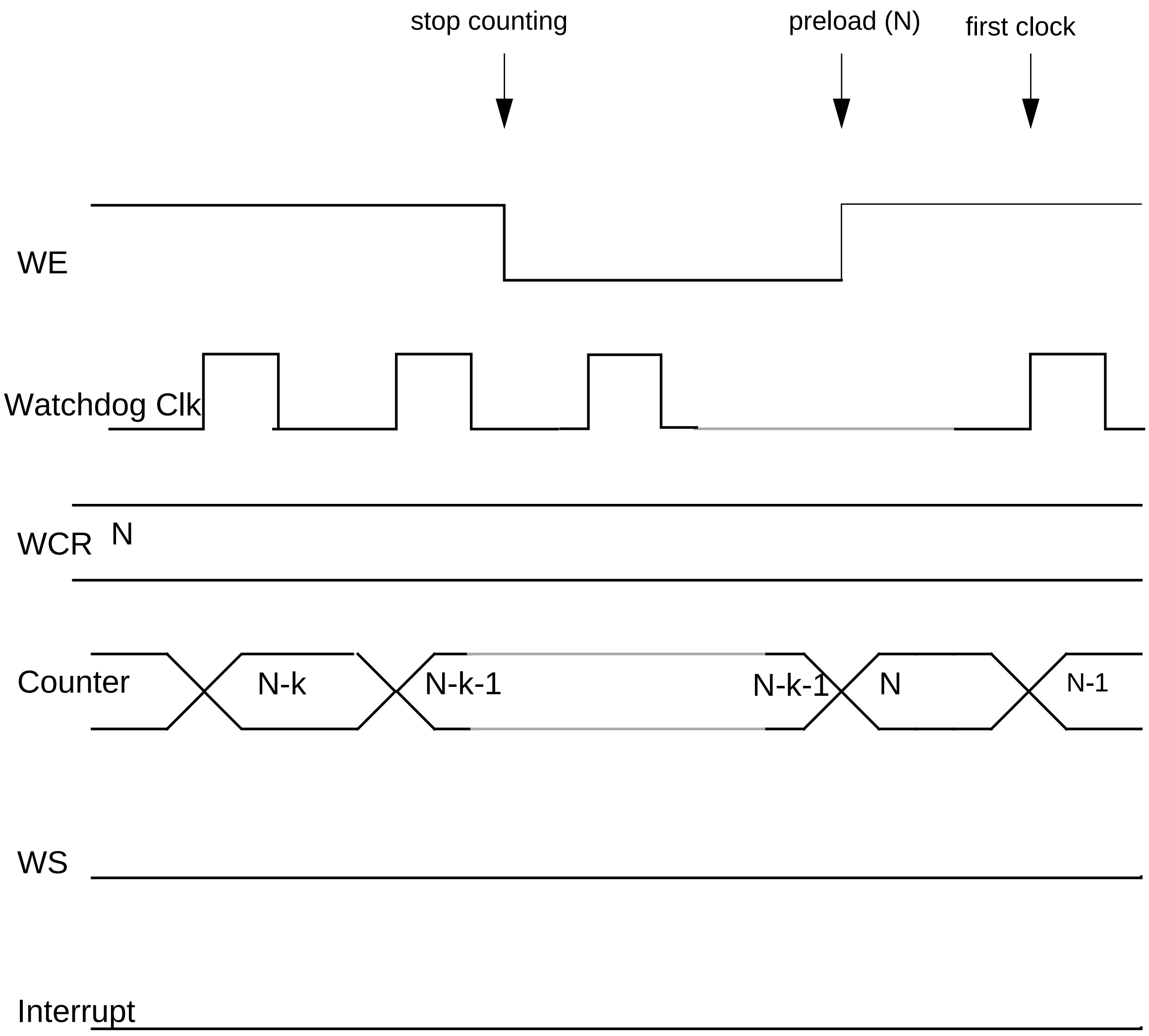


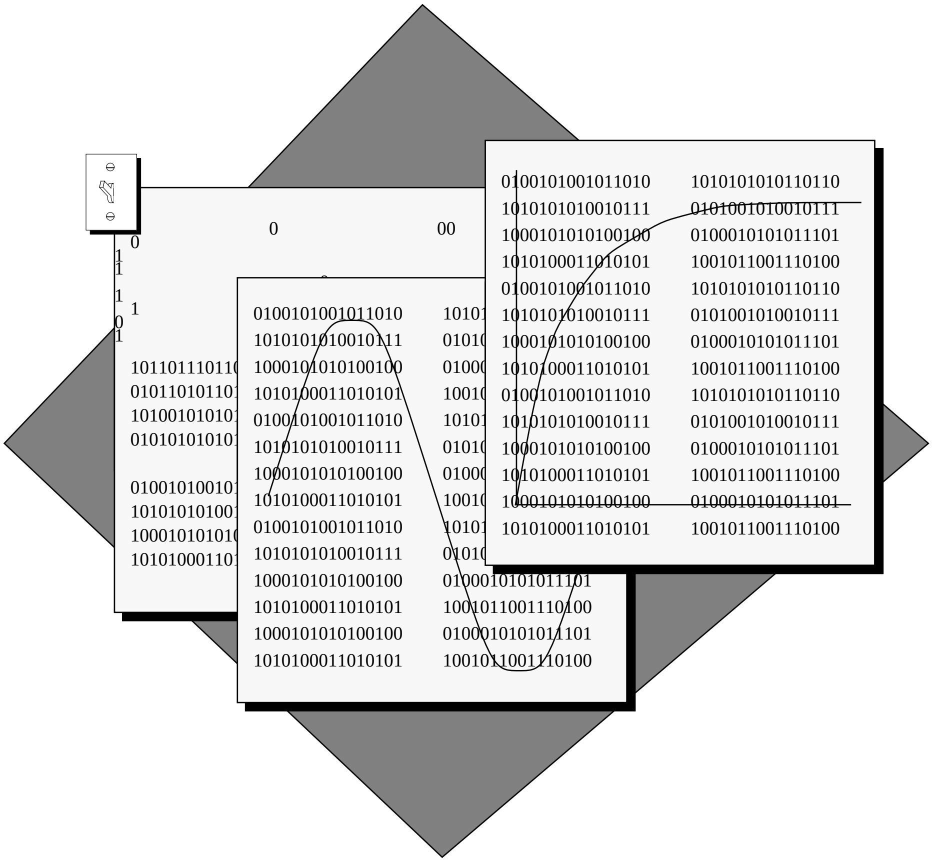
Figure 10-4 Watchdog Timer Disable

10.4 PROGRAMMING CONSIDERATIONS

The Watchdog Timer interrupt and the NMI are serviced through the same exception vector. It is the user's responsibility to identify the source of this interrupt. A typical scenario consists in using a long-interrupt routine for the NMI exception, in which a test of the WS (Watchdog Timer Status bit) from WCSR (Watchdog Timer Control/Status Register); if this bit is cleared, then the interrupt was generated through the MODC pin; if this bit is set, then the interrupt was generated through the MODC pin or by the Watchdog Timer.

APPENDIX A

BOOTSTRAP PROGRAM AND DATA ROM LISTINGS



SECTION CONTENTS

A.1 INTRODUCTION

This section presents the Bootstrap program contained in the DSP56003/005 96-word Boot ROM. This program can load the internal program RAM starting at P:$0 from an external EPROM or the Host Interface, and may load any program RAM segment from the SCI serial interface.

If MC:MB:MA=001, the program loads the internal program RAM from 13,824 consecutive byte-wide P memory locations, starting at P:$C000 (bits 7-0). These will be packed into 4608 24-bit words and stored in contiguous program RAM memory locations starting at P:$0. After assembling one 24-bit word, the bootstrap program stores the result in internal program RAM memory. Note that the routine loads data starting with the least significant byte of P:$0.

If MC:MB:MA=111, the program loads the internal program RAM from 13,824 consecutive byte-wide P memory locations, starting at P:$8000 (bits 7-0). These will be packed into 4608 24-bit words and stored in contiguous program RAM memory locations starting at P:$0. After assembling one 24-bit word, the bootstrap program stores the result in internal program RAM memory. Note that the routine loads data starting with the least significant byte of P:$0.

If MC:MB:MA=10x, the program loads internal program RAM from the Host Interface, starting at P:$0. If only a portion of the P memory is to be loaded, the Host Interface bootstrap load program may be stopped by setting Host Flag 0 (HF0). This will terminate the bootstrap loading operation and start executing the loaded program at location P:$0 of the internal program RAM.

If MC:MB:MA=110, the program loads 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 in internal program RAM to start loading the program words and then three 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 at 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.

The bootstrap program listing is shown in Figure A-1.

A.2 BOOTSTRAP PROGRAM LISTING

```
 BOOT EQU $C000                        ; this is the location in P memory
                                       ; on the external memory bus
                                       ; where the external byte-wide
                                       ; EPROM would be located (option 1)
BOOT1 EQU $8000                        ; this is the location in P memory
                                       ; on the external memory bus
                                       ; where the external byte-wide
                                       ; EPROM would be located (option 2)

PBC EQU $FFE0                          ; Port B Control Register
HSR EQU $FFE9                          ; Host Status Register
HRX EQU $FFEB                          ; Host Receive Register
PCC EQU $FFE1                          ; Port C Control Register
SCR EQU $FFF0                          ; SCI Control Register
SSR EQU $FFF1                          ; SCI Status Register
SCCR EQU $FFF2                         ; SCI Clock Control Register
SRXL EQU $FFF4                         ; SCI Receive Register Low
STXL EQU $FFF4                         ; SCI Transmit Register Low
P_SIZE EQU $1200                       ; Internal P_RAM size
            ORG PL:$0,PL:$0            ; bootstrap code starts at $0

START       MOVE #<0,R0                ; default P address where prog
                                       ; will begin loading
            MOVE #P_SIZE,B1            ; B1 will keep the number of
                                       ; words to be loaded through Host
            JCLR #4,OMR,EPROMLD        ; If MC:MB:MA=0xx, go load from
                                       ; EPROM
            JCLR #1,OMR,HOSTLD         ; If MC:MB:MA=10x, go load from HOST
            JCLR #0,OMR,SCILD          ; If MC:MB:MA=110, go load from SCI
            MOVE #BOOT1,R1             ; R1 = Ext address of EPROM
            JMP EPROMLD1

; This is the routine that loads from the Host Interface.
; MC:MB:MA=100 - reserved
; MC:MB:MA=101 - Host

HOSTLD      BSET #0,X:PBC              ; Configure Port B as Host
            DO B1,_LOOP3               ; Load P_SIZE instruction words
_LBLA       JCLR #3,X:HSR,_LBLB        ; if HF0=1, stop loading data.
            ENDDO                      ; Must terminate the do loop
            JMP <_LOOP3

_LBLB       JCLR #0,X:HSR,_LBLA        ; Wait for HRDF to go high
                                       ;(meaning data is present).
            MOVEP X:HRX,P:(R0)+        ; Store 24-bit data in P mem.
_LOOP3                                 ; and go get another 24-bit word.
                                       ; finish bootstrap
FINISH      MOVE #<0,R1
```

Figure A-1 DSP56003/005 Bootstrap Program Listing (Sheet 1 of 3)

```
; This is the exit handler that returns execution to normal
; expanded mode and jumps to the RESET vector.

BOOTEND      ANDI #$EC,OMR            ; Set operating mode to 0
                                      ; (and trigger an exit from
                                      ; bootstrap mode).
             ANDI #$0,CCR             ; Clear CCR as if RESET to 0.
                                      ; Delay needed for Op. Mode change
             JMP (R1)                 ; Then go to starting Prog addr.

; This is the routine that loads from the SCI.
; MC:MB:MA=110 - external SCI clock
; MC:MB:MA=111 - reserved

             ORG PL:$0D00,PL:$0D00    ; starting address of 2nd ROM
SCILD        MOVEP #$0302,X:SCR       ; Configure SCI Control Reg
             MOVEP #$C000,X:SCCR      ; Configure SCI Clock Control Reg
             MOVEP #7,X:PCC           ; Configure SCLK, TXD and RXD

_SCI1        DO #6,_LOOP6             ; get 3 bytes for number of
                                      ; program words and 3 bytes
                                      ; for the starting address
             JCLR #2,X:SSR,*          ; Wait for RDRF to go high
             MOVEP X:SRXL,A2          ; Put 8 bits in A2
             JCLR #1,X:SSR,*          ; Wait for TDRE to go high
             MOVEP A2,X:STXL          ; echo the received byte
             REP #8
             ASR A
_LOOP6
             MOVE A1,R0               ; starting address for load
             MOVE A1,R1               ; save starting address
             DO A0,_LOOP4             ; Receive program words

             DO #3,_LOOP5
             JCLR #2,X:SSR,*          ; Wait for RDRF to go high
             MOVEP X:SRXL,A2          ; Put 8 bits in A2
             JCLR #1,X:SSR,*          ; Wait for TDRE to go high
             MOVEP A2,X:STXL          ; echo the received byte
             REP #8
             ASR A
_LOOP5
             MOVEM A1,P:(R0)+         ; Store 24-bit result in P mem.
_LOOP4
             JMP FINISH+1             ; Boot from SCI done
```

Figure A-1 DSP56003/005 Bootstrap Program Listing (Sheet 2 of 3)

```
            ORG PL:$1100,PL:$1100   ; starting address of 3rd ROM
; This is the routine that loads from external EPROM.
; MC:MB:MA=001

EPROMLD     MOVE #BOOT,R1           ; R1 = Ext address of EPROM
EPROMLD1    DO B1,_LOOP1            ; Load P_SIZE instruction words
            DO #3,_LOOP2            ; Each instruction has 3 bytes
            MOVEM P:(R1)+,A2        ; Get the 8 LSB from ext. P mem.
            REP #8                  ; Shift 8 bit data into A1
            ASR A
_LOOP2                              ; Get another byte.
            MOVEM A1,P:(R0)+        ; Store 24-bit result in P mem.
_LOOP1                              ; and go get another 24-bit word.
            JMP FINISH              ; Boot from EPROM done
; End of bootstrap code. Number of program words: 70
```

Figure A-1 DSP56003/005 Bootstrap Program Listing (Sheet 3 of 3)

A.3 ARCTANGENT TABLE CONTENTS

This arctangent table (see Figure A-2) which is located in X memory ROM contains 256 unsigned 24-bit values for the arctangent function with an argument range of 0-4/9π.

```
        ORG X:$100
;
T_00 DC $000000
T_01 DC $03AA60
T_02 DC $0753CD
T_03 DC $0AFB57
T_04 DC $0EA00D
T_05 DC $124107
T_06 DC $15DD60
T_07 DC $19743B
T_08 DC $1D04C1
T_09 DC $208E27
T_0A DC $240FAB
T_0B DC $278894
T_0C DC $2AF837
T_0D DC $2E5DF4
T_0E DC $31B938
T_0F DC $35097B
T_10 DC $384E43
T_11 DC $3B8723
T_12 DC $3EB3BA
T_13 DC $41D3B3
T_14 DC $44E6C6
T_15 DC $47ECB6
T_16 DC $4AE552
T_17 DC $4DD073
T_18 DC $50ADFC
T_19 DC $537DDC
T_1A DC $564007
T_1B DC $58F47D
T_1C DC $5B9B44
T_1D DC $5E3469
T_1E DC $60C001
T_1F DC $633E26
T_20 DC $65AEF6
T_21 DC $681298
T_22 DC $6A6931
T_23 DC $6CB2F1
T_24 DC $6EF005
T_25 DC $7120A0
T_26 DC $7344F8
T_27 DC $755D43
T_28 DC $7769BA
T_29 DC $796A97
T_2A DC $7B6015
T_2B DC $7D4A70
T_2C DC $7F29E5
T_2D DC $80FEAF
T_2E DC $82C90C
T_2F DC $848939
T_30 DC $863F71
T_31 DC $87EBF2
T_32 DC $898EF6
T_33 DC $8B28B7
T_34 DC $8CB972
T_35 DC $8E415E
T_36 DC $8FC0B3
T_37 DC $9137AA
T_38 DC $92A679
T_39 DC $940D55
T_3A DC $956C73
T_3B DC $96C405
T_3C DC $98143D
T_3D DC $995D4D
T_3E DC $9A9F64
T_3F DC $9BDAB2
T_40 DC $9D0F62
T_41 DC $9E3DA2
T_42 DC $9F659D
T_43 DC $A0877D
T_44 DC $A1A36A
T_45 DC $A2B98D
T_46 DC $A3CA0C
T_47 DC $A4D50E
T_48 DC $A5DAB6
T_49 DC $A6DB28
T_4A DC $A7D687
T_4B DC $A8CCF5
T_4C DC $A9BE92
T_4D DC $AAAB7F
T_4E DC $AB93D9
T_4F DC $AC77BE
T_50 DC $AD574D
T_51 DC $AE32A1
T_52 DC $AF09D6
T_53 DC $AFDD05
T_54 DC $B0AC4A
T_55 DC $B177BD
T_56 DC $B23F76
T_57 DC $B3038E
T_58 DC $B3C41B
T_59 DC $B48133
T_5A DC $B53AED
T_5B DC $B5F15C
T_5C DC $B6A496
T_5D DC $B754AF
T_5E DC $B801BA
T_5F DC $B8ABC8
T_60 DC $B952EE
T_61 DC $B9F73B
T_62 DC $BA98C2
T_63 DC $BB3793
T_64 DC $BBD3BE
T_65 DC $BC6D53
T_66 DC $BD0461
T_67 DC $BD98F7
T_68 DC $BE2B24
T_69 DC $BEBAF5
```

Figure A-2 Arc-tangent Table Contents Listing (Part 1 of 3)

```
T_6A DC $BF4878
T_6B DC $BFD3BA
T_6C DC $C05CC9
T_6D DC $C0E3B0
T_6E DC $C1687D
T_6F DC $C1EB3A
T_70 DC $C26BF3
T_71 DC $C2EAB4
T_72 DC $C36787
T_73 DC $C3E278
T_74 DC $C45B90
T_75 DC $C4D2D9
T_76 DC $C5485D
T_77 DC $C5BC27
T_78 DC $C62E3E
T_79 DC $C69EAC
T_7A DC $C70D7B
T_7B DC $C77AB1
T_7C DC $C7E659
T_7D DC $C85079
T_7E DC $C8B91A
T_7F DC $C92044
T_80 DC $C985FE
T_81 DC $C9EA4F
T_82 DC $CA4D3F
T_83 DC $CAAED4
T_84 DC $CB0F16
T_85 DC $CB6E0A
T_86 DC $CBCBB8
T_87 DC $CC2826
T_88 DC $CC835A
T_89 DC $CCDD59
T_8A DC $CD362A
T_8B DC $CD8DD2
T_8C DC $CDE458
T_8D DC $CE39C0
T_8E DC $CE8E0F
T_8F DC $CEE14C
T_90 DC $CF337A
T_91 DC $CF84A0
T_92 DC $CFD4C1
T_93 DC $D023E2
T_94 DC $D07209
T_95 DC $D0BF39
T_96 DC $D10B77
T_97 DC $D156C7
T_98 DC $D1A12E
T_99 DC $D1EAAF
T_9A DC $D2334F
T_9B DC $D27B11
T_9C DC $D2C1FA
T_9D DC $D3080C
T_9E DC $D34D4C
T_9F DC $D391BE
```

```
T_A0 DC $D3D564
T_A1 DC $D41842
T_A2 DC $D45A5C
T_A3 DC $D49BB5
T_A4 DC $D4DC50
T_A5 DC $D51C30
T_A6 DC $D55B58
T_A7 DC $D599CC
T_A8 DC $D5D78D
T_A9 DC $D614A0
T_AA DC $D65107
T_AB DC $D68CC4
T_AC DC $D6C7DB
T_AD DC $D7024E
T_AE DC $D73C1F
T_AF DC $D77551
T_B0 DC $D7ADE7
T_B1 DC $D7E5E4
T_B2 DC $D81D48
T_B3 DC $D85418
T_B4 DC $D88A54
T_B5 DC $D8C000
T_B6 DC $D8F51E
T_B7 DC $D929AF
T_B8 DC $D95DB6
T_B9 DC $D99135
T_BA DC $D9C42E
T_BB DC $D9F6A3
T_BC DC $DA2895
T_BD DC $DA5A08
T_BE DC $DA8AFC
T_BF DC $DABB74
T_C0 DC $DAEB71
T_C1 DC $DB1AF6
T_C2 DC $DB4A03
T_C3 DC $DB789B
T_C4 DC $DBA6BF
T_C5 DC $DBD471
T_C6 DC $DC01B2
T_C7 DC $DC2E85
T_C8 DC $DC5AEA
T_C9 DC $DC86E4
T_CA DC $DCB273
T_CB DC $DCDD99
T_CC DC $DD0858
T_CD DC $DD32B1
T_CE DC $DD5CA6
T_CF DC $DD8637
T_D0 DC $DDAF67
T_D1 DC $DDD836
T_D2 DC $DE00A6
T_D3 DC $DE28B8
T_D4 DC $DE506D
T_D5 DC $DE77C7
```

Figure A-2 Arc-tangent Table Contents Listing (Part 2 of 3)

```
T_D6 DC $DE9EC6
T_D7 DC $DEC56D
T_D8 DC $DEEBBC
T_D9 DC $DF11B3
T_DA DC $DF3756
T_DB DC $DF5CA4
T_DC DC $DF819E
T_DD DC $DFA646
T_DE DC $DFCA9D
T_DF DC $DFEEA3
T_E0 DC $E0125B
T_E1 DC $E035C4
T_E2 DC $E058E0
T_E3 DC $E07BB0
T_E4 DC $E09E34
T_E5 DC $E0C06E
T_E6 DC $E0E25F
T_E7 DC $E10407
T_E8 DC $E12567
T_E9 DC $E14681
T_EA DC $E16755
```

```
T_EB DC $E187E4
T_EC DC $E1A82E
T_ED DC $E1C835
T_EE DC $E1E7FA
T_EF DC $E2077D
T_F0 DC $E226BF
T_F1 DC $E245C0
T_F2 DC $E26482
T_F3 DC $E28306
T_F4 DC $E2A14C
T_F5 DC $E2BF54
T_F6 DC $E2DD20
T_F7 DC $E2FAB0
T_F8 DC $E31804
T_F9 DC $E3351F
T_FA DC $E351FF
T_FB DC $E36EA7
T_FC DC $E38B16
T_FD DC $E3A74D
T_FE DC $E3C34D
T_FF DC $E3DF17
```

Figure A-2 Arc-tangent Table ContentsListing (Part 3 of 3)

The data values for Figure A-2 were calculated using the following formula:

$$\text{Data} = \frac{16777216}{\pi} \times \operatorname{atan}\left(\frac{\text{Address}}{256 \times \cos\frac{4\pi}{9}}\right)$$

A.4 SINE TABLE CONTENTS

This sine table (Figure A-3) which is located in Y memory ROM is normally used by FFT routines which use bit reversed address pointers. This table can be used as it is for up to 512 point FFTs; however, for larger FFTs, the table must be copied to a different memory location to allow the reverse-carry addressing mode to be used (see **REVERSE-CARRY MODIFIER (Mn=$0000)** in the **DSP56000 Family Manual** for additional information).

```
        ORG   Y:$100
;
S_00    DC    $000000  ; +0.0000000000
S_01    DC    $03242B  ; +0.0245412998
S_02    DC    $0647D9  ; +0.0490676016
S_03    DC    $096A90  ; +0.0735644996
S_04    DC    $0C8BD3  ; +0.0980170965
S_05    DC    $0FAB27  ; +0.1224106997
S_06    DC    $12C810  ; +0.1467303932
S_07    DC    $15E214  ; +0.1709619015
S_08    DC    $18F8B8  ; +0.1950902939
S_09    DC    $1C0B82  ; +0.2191012055
S_0A    DC    $1F19F9  ; +0.2429800928
S_0B    DC    $2223A5  ; +0.2667128146
S_0C    DC    $25280C  ; +0.2902846038
S_0D    DC    $2826B9  ; +0.3136816919
S_0E    DC    $2B1F35  ; +0.3368898928
S_0F    DC    $2E110A  ; +0.3598949909
S_10    DC    $30FBC5  ; +0.3826833963
S_11    DC    $33DEF3  ; +0.4052414000
S_12    DC    $36BA20  ; +0.4275551140
S_13    DC    $398CDD  ; +0.4496113062
S_14    DC    $3C56BA  ; +0.4713967144
S_15    DC    $3F174A  ; +0.4928981960
S_16    DC    $41CE1E  ; +0.5141026974
S_17    DC    $447ACD  ; +0.5349975824
S_18    DC    $471CED  ; +0.5555701852
S_19    DC    $49B415  ; +0.5758082271
S_1A    DC    $4C3FE0  ; +0.5956993103
S_1B    DC    $4EBFE9  ; +0.6152315736
S_1C    DC    $5133CD  ; +0.6343932748
S_1D    DC    $539B2B  ; +0.6531729102
S_1E    DC    $55F5A5  ; +0.6715589762
S_1F    DC    $5842DD  ; +0.6895405054
S_20    DC    $5A827A  ; +0.7071068287
S_21    DC    $5CB421  ; +0.7242470980
S_22    DC    $5ED77D  ; +0.7409511805
S_23    DC    $60EC38  ; +0.7572088242
S_24    DC    $62F202  ; +0.7730104923
S_25    DC    $64E889  ; +0.7883464098
S_26    DC    $66CF81  ; +0.8032075167
S_27    DC    $68A69F  ; +0.8175848722
S_28    DC    $6A6D99  ; +0.8314697146
S_29    DC    $6C2429  ; +0.8448535204
S_2A    DC    $6DCA0D  ; +0.8577286005
S_2B    DC    $6F5F03  ; +0.8700870275
S_2C    DC    $70E2CC  ; +0.8819212914
S_2D    DC    $72552D  ; +0.8932244182
S_2E    DC    $73B5EC  ; +0.9039893150
S_2F    DC    $7504D3  ; +0.9142097235
S_30    DC    $7641AF  ; +0.9238795042
S_31    DC    $776C4F  ; +0.9329928160
S_32    DC    $788484  ; +0.9415441155
S_33    DC    $798A24  ; +0.9495282173
S_34    DC    $7A7D05  ; +0.9569402933
S_35    DC    $7B5D04  ; +0.9637761116
S_36    DC    $7C29FC  ; +0.9700313210
S_37    DC    $7CE3CF  ; +0.9757022262
S_38    DC    $7D8A5F  ; +0.9807853103
S_39    DC    $7E1D94  ; +0.9852777123
S_3A    DC    $7E9D56  ; +0.9891765118
S_3B    DC    $7F0992  ; +0.9924796224
S_3C    DC    $7F6237  ; +0.9951847792
S_3D    DC    $7FA737  ; +0.9972904921
S_3E    DC    $7FD888  ; +0.9987955093
S_3F    DC    $7FF622  ; +0.9996988773
S_40    DC    $7FFFFF  ; +0.9999998808
S_41    DC    $7FF622  ; +0.9996988773
S_42    DC    $7FD888  ; +0.9987955093
S_43    DC    $7FA737  ; +0.9972904921
S_44    DC    $7F6237  ; +0.9951847792
S_45    DC    $7F0992  ; +0.9924796224
S_46    DC    $7E9D56  ; +0.9891765118
S_47    DC    $7E1D94  ; +0.9852777123
S_48    DC    $7D8A5F  ; +0.9807853103
S_49    DC    $7CE3CF  ; +0.9757022262
S_4A    DC    $7C29FC  ; +0.9700313210
S_4B    DC    $7B5D04  ; +0.9637761116
S_4C    DC    $7A7D05  ; +0.9569402933
S_4D    DC    $798A24  ; +0.9495282173
S_4E    DC    $788484  ; +0.9415441155
S_4F    DC    $776C4F  ; +0.9329928160
S_50    DC    $7641AF  ; +0.9238795042
S_51    DC    $7504D3  ; +0.9142097235
S_52    DC    $73B5EC  ; +0.9039893150
S_53    DC    $72552D  ; +0.8932244182
S_54    DC    $70E2CC  ; +0.8819212914
S_55    DC    $6F5F03  ; +0.8700870275
S_56    DC    $6DCA0D  ; +0.8577286005
S_57    DC    $6C2429  ; +0.8448535204
S_58    DC    $6A6D99  ; +0.8314697146
S_59    DC    $68A69F  ; +0.8175848722
S_5A    DC    $66CF81  ; +0.8032075167
S_5B    DC    $64E889  ; +0.7883464098
S_5C    DC    $62F202  ; +0.7730104923
S_5D    DC    $60EC38  ; +0.7572088242
S_5E    DC    $5ED77D  ; +0.7409511805
S_5F    DC    $5CB421  ; +0.7242470980
S_60    DC    $5A827A  ; +0.7071068287
S_61    DC    $5842DD  ; +0.6895405054
S_62    DC    $55F5A5  ; +0.6715589762
S_63    DC    $539B2B  ; +0.6531729102
S_64    DC    $5133CD  ; +0.6343932748
S_65    DC    $4EBFE9  ; +0.6152315736
S_66    DC    $4C3FE0  ; +0.5956993103
S_67    DC    $49B415  ; +0.5758082271
S_68    DC    $471CED  ; +0.5555701852
```

Figure A-3 Sine Table Contents (Part 1 of 3)

```
S_6A    DC    $41CE1E   ; +0.5141026974
S_6B    DC    $3F174A   ; +0.4928981960
S_6C    DC    $3C56BA   ; +0.4713967144
S_6D    DC    $398CDD   ; +0.4496113062
S_6E    DC    $36BA20   ; +0.4275551140
S_6F    DC    $33DEF3   ; +0.4052414000
S_70    DC    $30FBC5   ; +0.3826833963
S_71    DC    $2E110A   ; +0.3598949909
S_72    DC    $2B1F35   ; +0.3368898928
S_73    DC    $2826B9   ; +0.3136816919
S_74    DC    $25280C   ; +0.2902846038
S_75    DC    $2223A5   ; +0.2667128146
S_76    DC    $1F19F9   ; +0.2429800928
S_77    DC    $1C0B82   ; +0.2191012055
S_78    DC    $18F8B8   ; +0.1950902939
S_79    DC    $15E214   ; +0.1709619015
S_7A    DC    $12C810   ; +0.1467303932
S_7B    DC    $0FAB27   ; +0.1224106997
S_7C    DC    $0C8BD3   ; +0.0980170965
S_7D    DC    $096A90   ; +0.0735644996
S_7E    DC    $0647D9   ; +0.0490676016
S_7F    DC    $03242B   ; +0.0245412998
S_80    DC    $000000   ; +0.0000000000
S_81    DC    $FCDBD5   ; -0.0245412998
S_82    DC    $F9B827   ; -0.0490676016
S_83    DC    $F69570   ; -0.0735644996
S_84    DC    $F3742D   ; -0.0980170965
S_85    DC    $F054D9   ; -0.1224106997
S_86    DC    $ED37F0   ; -0.1467303932
S_87    DC    $EA1DEC   ; -0.1709619015
S_88    DC    $E70748   ; -0.1950902939
S_89    DC    $E3F47E   ; -0.2191012055
S_8A    DC    $E0E607   ; -0.2429800928
S_8B    DC    $DDDC5B   ; -0.2667128146
S_8C    DC    $DAD7F4   ; -0.2902846038
S_8D    DC    $D7D947   ; -0.3136816919
S_8E    DC    $D4E0CB   ; -0.3368898928
S_8F    DC    $D1EEF6   ; -0.3598949909
S_90    DC    $CF043B   ; -0.3826833963
S_91    DC    $CC210D   ; -0.4052414000
S_92    DC    $C945E0   ; -0.4275551140
S_93    DC    $C67323   ; -0.4496113062
S_94    DC    $C3A946   ; -0.4713967144
S_95    DC    $C0E8B6   ; -0.4928981960
S_96    DC    $BE31E2   ; -0.5141026974
S_97    DC    $BB8533   ; -0.5349975824
S_98    DC    $B8E313   ; -0.5555701852
S_99    DC    $B64BEB   ; -0.5758082271
S_9A    DC    $B3C020   ; -0.5956993103
S_9B    DC    $B14017   ; -0.6152315736
S_9C    DC    $AECC33   ; -0.6343932748
S_9D    DC    $AC64D5   ; -0.6531729102
S_9E    DC    $AA0A5B   ; -0.6715589762
S_9F    DC    $A7BD23   ; -0.6895405054
```

```
S_A0    DC    $A57D86   ; -0.7071068287
S_A1    DC    $A34BDF   ; -0.7242470980
S_A2    DC    $A12883   ; -0.7409511805
S_A3    DC    $9F13C8   ; -0.7572088242
S_A4    DC    $9D0DFE   ; -0.7730104923
S_A5    DC    $9B1777   ; -0.7883464098
S_A6    DC    $99307F   ; -0.8032075167
S_A7    DC    $975961   ; -0.8175848722
S_A8    DC    $959267   ; -0.8314697146
S_A9    DC    $93DBD7   ; -0.8448535204
S_AA    DC    $9235F3   ; -0.8577286005
S_AB    DC    $90A0FD   ; -0.8700870275
S_AC    DC    $8F1D34   ; -0.8819212914
S_AD    DC    $8DAAD3   ; -0.8932244182
S_AE    DC    $8C4A14   ; -0.9039893150
S_AF    DC    $8AFB2D   ; -0.9142097235
S_B0    DC    $89BE51   ; -0.9238795042
S_B1    DC    $8893B1   ; -0.9329928160
S_B2    DC    $877B7C   ; -0.9415441155
S_B3    DC    $8675DC   ; -0.9495282173
S_B4    DC    $8582FB   ; -0.9569402933
S_B5    DC    $84A2FC   ; -0.9637761116
S_B6    DC    $83D604   ; -0.9700313210
S_B7    DC    $831C31   ; -0.9757022262
S_B8    DC    $8275A1   ; -0.9807853103
S_B9    DC    $81E26C   ; -0.9852777123
S_BA    DC    $8162AA   ; -0.9891765118
S_BB    DC    $80F66E   ; -0.9924796224
S_BC    DC    $809DC9   ; -0.9951847792
S_BD    DC    $8058C9   ; -0.9972904921
S_BE    DC    $802778   ; -0.9987955093
S_BF    DC    $8009DE   ; -0.9996988773
S_C0    DC    $800000   ; -1.0000000000
S_C1    DC    $8009DE   ; -0.9996988773
S_C2    DC    $802778   ; -0.9987955093
S_C3    DC    $8058C9   ; -0.9972904921
S_C4    DC    $809DC9   ; -0.9951847792
S_C5    DC    $80F66E   ; -0.9924796224
S_C6    DC    $8162AA   ; -0.9891765118
S_C7    DC    $81E26C   ; -0.9852777123
S_C8    DC    $8275A1   ; -0.9807853103
S_C9    DC    $831C31   ; -0.9757022262
S_CA    DC    $83D604   ; -0.9700313210
S_CB    DC    $84A2FC   ; -0.9637761116
S_CC    DC    $8582FB   ; -0.9569402933
S_CD    DC    $8675DC   ; -0.9495282173
S_CE    DC    $877B7C   ; -0.9415441155
S_CF    DC    $8893B1   ; -0.9329928160
S_D0    DC    $89BE51   ; -0.9238795042
S_D1    DC    $8AFB2D   ; -0.9142097235
S_D2    DC    $8C4A14   ; -0.9039893150
S_D3    DC    $8DAAD3   ; -0.8932244182
S_D4    DC    $8F1D34   ; -0.8819212914
S_D5    DC    $90A0FD   ; -0.8700870275
```

Figure A-3 Sine Table Contents (Part 2 of 3)

```
S_D6    DC    $9235F3  ; -0.8577286005
S_D7    DC    $93DBD7  ; -0.8448535204
S_D8    DC    $959267  ; -0.8314697146
S_D9    DC    $975961  ; -0.8175848722
S_DA    DC    $99307F  ; -0.8032075167
S_DB    DC    $9B1777  ; -0.7883464098
S_DC    DC    $9D0DFE  ; -0.7730104923
S_DD    DC    $9F13C8  ; -0.7572088242
S_DE    DC    $A12883  ; -0.7409511805
S_DF    DC    $A34BDF  ; -0.7242470980
S_E0    DC    $A57D86  ; -0.7071068287
S_E1    DC    $A7BD23  ; -0.6895405054
S_E2    DC    $AA0A5B  ; -0.6715589762
S_E3    DC    $AC64D5  ; -0.6531729102
S_E4    DC    $AECC33  ; -0.6343932748
S_E5    DC    $B14017  ; -0.6152315736
S_E6    DC    $B3C020  ; -0.5956993103
S_E7    DC    $B64BEB  ; -0.5758082271
S_E8    DC    $B8E313  ; -0.5555701852
S_E9    DC    $BB8533  ; -0.5349975824
S_EA    DC    $BE31E2  ; -0.5141026974
```

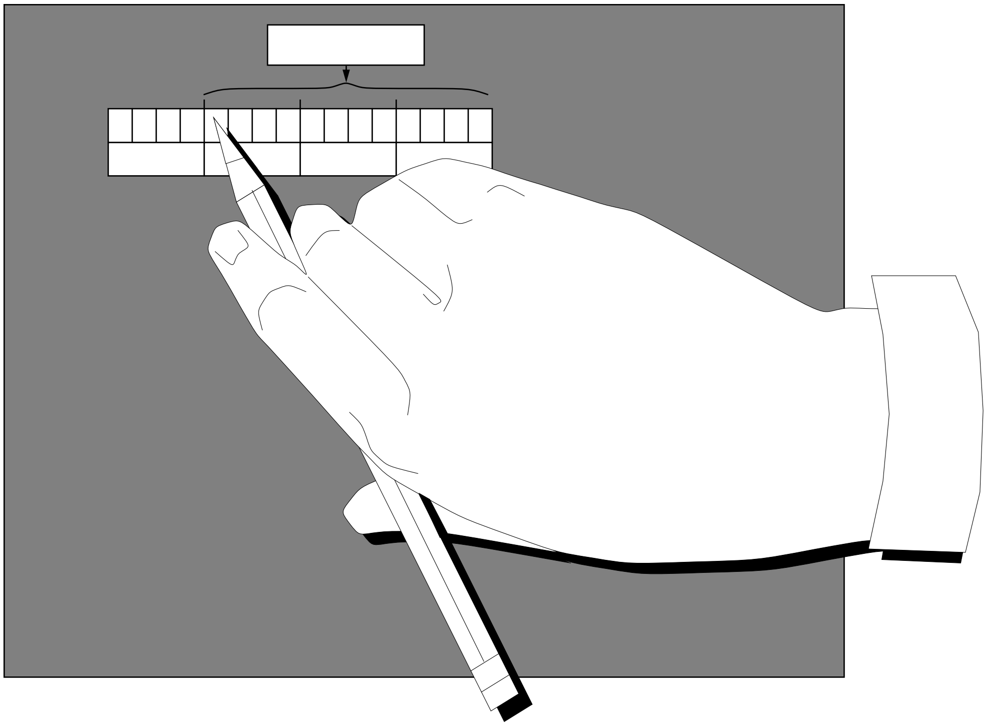
```
S_EB    DC    $C0E8B6  ; -0.4928981960
S_EC    DC    $C3A946  ; -0.4713967144
S_ED    DC    $C67323  ; -0.4496113062
S_EE    DC    $C945E0  ; -0.4275551140
S_EF    DC    $CC210D  ; -0.4052414000
S_F0    DC    $CF043B  ; -0.3826833963
S_F1    DC    $D1EEF6  ; -0.3598949909
S_F2    DC    $D4E0CB  ; -0.3368898928
S_F3    DC    $D7D947  ; -0.3136816919
S_F4    DC    $DAD7F4  ; -0.2902846038
S_F5    DC    $DDDC5B  ; -0.2667128146
S_F6    DC    $E0E607  ; -0.2429800928
S_F7    DC    $E3F47E  ; -0.2191012055
S_F8    DC    $E70748  ; -0.1950902939
S_F9    DC    $EA1DEC  ; -0.1709619015
S_FA    DC    $ED37F0  ; -0.1467303932
S_FB    DC    $F054D9  ; -0.1224106997
S_FC    DC    $F3742D  ; -0.0980170965
S_FD    DC    $F69570  ; -0.0735644996
S_FE    DC    $F9B827  ; -0.0490676016
S_FF    DC    $FCDBD5  ; -0.0245412998
```

Figure A-3 Sine Table Contents (Part 3 of 3)

APPENDIX B

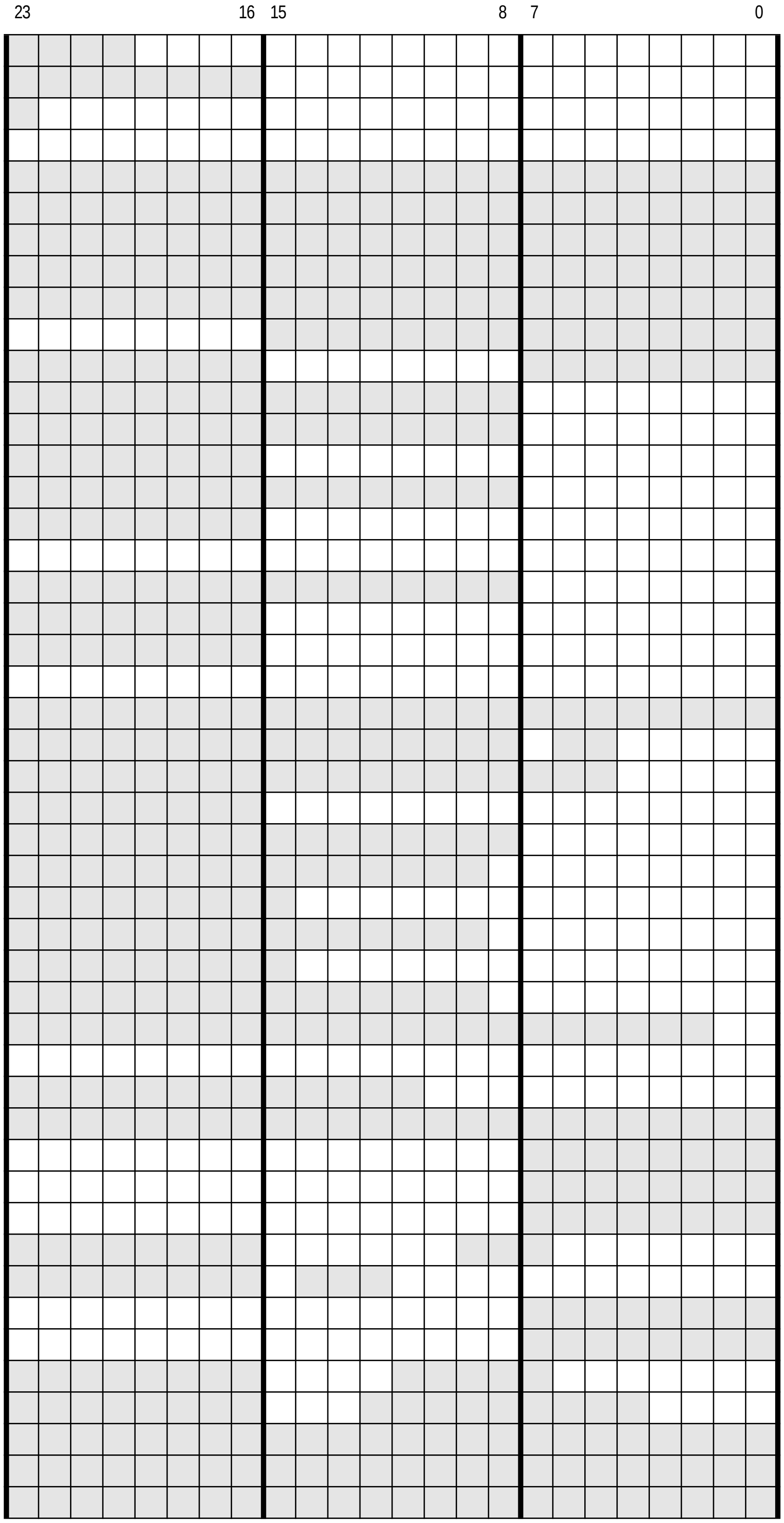
PROGRAMMING SHEETS

The following pages are a set of programming sheets intended to simplify programming the various DSP56003/005 programmable registers. The registers are grouped between the central processing module and each peripheral. Each register includes the name, address, reset value, and meaning of each bit. The sheets provide room to write the value for each bit and the hexadecimal equivalent for each register.



SECTION CONTENTS

PERIPHERAL ADDRESSES



X:$FFFF	INTERRUPT PRIORITY REGISTER (IPR)
X:$FFFE	PORT A — BUS CONTROL REGISTER (BCR)
X:$FFFD	PLL CONTROL REGISTER (PCTL)
X:$FFFC	ONCE PORT GDB REGISTER
X:$FFFB	RESERVED
X:$FFFA	RESERVED
X:$FFF9	RESERVED
X:$FFF8	RESERVED
X:$FFF7	RESERVED
X:$FFF6	SCI HI-REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF5	SCI MID-REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF4	SCI LOW-REC/XMIT DATA REGISTER (SRX/STX)
X:$FFF3	SCI TRANSMIT DATA ADDRESS REGISTER (STXA)
X:$FFF2	SCI CONTROL REGISTER (SCCR)
X:$FFF1	SCI INTERFACE STATUS REGISTER (SSR)
X:$FFF0	SCI INTERFACE CONTROL REGISTER (SCR)
X:$FFEF	SSI RECEIVE/TRANSMIT DATA REGISTER (RX/TX)
X:$FFEE	SSI STATUS/TIME SLOT REGISTER (SSISR/TSR)
X:$FFED	SSI CONTROL REGISTER B (CRB)
X:$FFEC	SSI CONTROL REGISTER A (CRA)
X:$FFEB	HOST RECEIVE/TRANSMIT REGISTER (HRX/HTX)
X:$FFEA	RESERVED
X:$FFE9	HOST STATUS REGISTER (HSR)
X:$FFE8	HOST CONTROL REGISTER (HCR)
X:$FFE7	WATCHDOG TIMER COUNT REGISTER (WCR)
X:$FFE6	WATCHDOG TIMER CONTROL/STATUS REGISTER (WCSR)
X:$FFE5	PORT C - GPIO DATA REGISTER (PCD)
X:$FFE4	PORT B - GPIO DATA REGISTER (PBD)
X:$FFE3	PORT C - GPIO DATA DIRECTION REGISTER (PCDDR)
X:$FFE2	PORT B - GPIO DATA DIRECTION REGISTER (PBDDR)
X:$FFE1	PORT C - GPIO CONTROL REGISTER (PCC)
X:$FFE0	PORT B - GPIO CONTROL REGISTER (PBC)
X:$FFDF	TIMER COUNT REGISTER (TCR)
X:$FFDE	TIMER CONTROL/STATUS REGISTER (TCSR)
X:$FFDD	RESERVED
X:$FFDC	PWMA2 COUNT REGISTER (PWACR2)
X:$FFDB	PWMA1 COUNT REGISTER (PWACR1)
X:$FFDA	PWMA0 COUNT REGISTER (PWACR0)
X:$FFD9	PWMA CONTROL AND STATUS REGISTER 0 (PWACSR0)
X:$FFD8	PWMA CONTROL AND STATUS REGISTER 1 (PWACSR1)
X:$FFD7	PWMB1 COUNT REGISTER (PWBCR1)
X:$FFD6	PWMB0 COUNT REGISTER (PWBCR0)
X:$FFD5	PWMB CONTROL AND STATUS REGISTER 0 (PWBCSR0)
X:$FFD4	PWMB CONTROL AND STATUS REGISTER 1 (PWBCSR1)
X:$FFD3	RESERVED
X:$FFC0	RESERVED

▨ = Read as a random number; write as don't care.

Figure B-1 On-chip Peripheral Memory Map

INTERRUPT VECTOR ADDRESSES

Table B-1 Interrupt Starting Addresses and Sources

Interrupt Starting Address	IPL	Interrupt Source
P:$0000	3	Hardware $\overline{\text{RESET}}$
P:$0002	3	Stack Error
P:$0004	3	Trace
P:$0006	3	SWI
P:$0008	0 - 2	$\overline{\text{IRQA}}$
P:$000A	0 - 2	$\overline{\text{IRQB}}$
P:$000C	0 - 2	SSI Receive Data
P:$000E	0 - 2	SSI Receive Data With Exception Status
P:$0010	0 - 2	SSI Transmit Data
P:$0012	0 - 2	SSI Transmit Data with Exception Status
P:$0014	0 - 2	SCI Receive Data
P:$0016	0 - 2	SCI Receive Data with Exception Status
P:$0018	0 - 2	SCI Transmit Data
P:$001A	0 - 2	SCI Idle Line
P:$001C	0 - 2	SCI Timer
P:$001E	3	NMI/Watchdog Timer
P:$0020	0 - 2	Host Receive Data
P:$0022	0 - 2	Host Transmit Data
P:$0024	0 - 2	Host Command (Default)
P:$0026	0 - 2	Available for Host Command
P:$0028	0 - 2	Available for Host Command
P:$002A	0 - 2	Available for Host Command
P:$002C	0 - 2	$\overline{\text{IRQC}}$
P:$002E	0 - 2	$\overline{\text{IRQD}}$
P:$0030	0 - 2	PWMA0
P:$0032	0 - 2	PWMA1
P:$0034	0 - 2	PWMA2
P:$0036	0 - 2	PWMB0
P:$0038	0 - 2	PWMB
P:$003A	0 - 2	PWM Error
P:$003C	0 - 2	Timer/Event Counter
P:$003E	3	Illegal Instruction
P:$0040	0 - 2	Available for Host Command
P:$007E	0 - 2	Available for Host Command

EXCEPTION PRIORITIES

Table B-2 Exception Priorities Within an IPL

Priority	Exception
Level 3 (Nonmaskable)	
Highest	Hardware $\overline{\text{RESET}}$
	Illegal Instruction
	$\overline{\text{NMI}}$
	Stack Error
	Trace
Lowest	SWI
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)
	Host Command Interrupt
	Host Receive Data Interrupt
	Host Transmit Data Interrupt
	SSI RX Data with Exception Interrupt
	SSI RX Data Interrupt
	SSI TX Data with Exception Interrupt
	SSI TX Data Interrupt
	SCI RX Data with Exception Interrupt
	SCI RX Data Interrupt
	SCI TX Data with Exception Interrupt
	SCI TX Data Interrupt
	SCI Idle Line Interrupt
	SCI Timer Interrupt
	Timer/Event Counter Interrupt
	PWM Error
	PWMA0 Ready
	PWMA1 Ready
	PWMA2 Ready
	PWMB0 Ready
Lowest	PWMB1 Ready

INSTRUCTIONS

Table B-3 Instruction Set Summary — Sheet 1 of 5

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
ABS	D	(parallel move)	1+mv	2+mv	* * * * * * * -
ADC	S,D	(parallel move)	1+mv	2+mv	* * * * * * * *
ADD	S,D	(parallel move)	1+mv	2+mv	* * * * * * * *
ADDL	S,D	(parallel move)	1+mv	2+mv	* * * * * * ?*
ADDR	S,D	(parallel move)	1+mv	2+mv	* * * * * * * *
AND	S,D	(parallel move)	1+mv	2+mv	* * - - ? ? 0-
AND(I)	#xx,D		1	2	? ? ? ? ? ? ? ?
ASL	D	(parallel move)	1+mv	2+mv	* * * * * * ??
ASR	D	(parallel move)	1+mv	2+mv	* * * * * * 0?
BCHG	#n,X:<aa> #n,X:<pp> #n,X:<ea> #n,Y:<aa> #n,Y:<pp> #n,Y:<ea> #n,D		1+ea	4+mvb	? ? ? ? ? ? ? ?
BCLR	#n,X:<aa> #n,X:<pp> #n,X:<ea> #n,Y:<aa> #n,Y:<pp> #n,Y:<ea> #n,D		1+ea	4+mvb	? ? ? ? ? ? ? ?
BSET	#n,X:<aa> #n,X:<pp> #n,X:<ea> #n,Y:<aa> #n,Y:<pp> #n,Y:<ea> #n,D		1+ea	4+mvb	? ? ? ? ? ? ? ?
BTST	#n,X:<aa> #n,X:<pp> #n,X:<ea> #n,Y:<aa> #n,Y:<pp> #n,Y:<ea> #n,D		1+ea	4+mvb	- * - - - - - ?
CLR	D	(parallel move)	1+mv	2+mv	* * ? ? ? ? ?-
CMP	S1,S2	(parallel move)	1+mv	2+mv	* * * * * * * *
CMPM	S1,S2	(parallel move)	1+mv	2+mv	* * * * * * * *
DEBUG			1	4	- - - - - - - -
DEBUGcc			1	4	- - - - - - - -
DEC	D		1	2	- * * * * * * *
DIV	S,D		1	2	- * - - - - ??

INSTRUCTIONS

Table B-3 Instruction Set Summary — Sheet 2 of 5

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
DO	X:<ea>,expr X:<aa>,expr Y:<ea>,expr Y:<aa>,expr #xxx,expr S,expr		2	6+mv	* * - - - - - -
ENDDO			1	2	- - - - - - - -
EOR	S,D	(parallel move)	1+mv	2+mv	* * - - ? ? 0 -
ILLEGAL			1	8	- - - - - - - -
INC	D		1	2	- * * * * * * *
Jcc	xxx		1+ea	4+jx	- - - - - - - -
JCLR	#n,X:<ea>,xxxx #n,X:<aa>,xxxx #n,X:<pp>,xxxx #n,Y:<ea>,xxxx #n,Y:<aa>,xxxx #n,Y:<pp>,xxxx #n,S,xxxx		2	6+jx	* * - - - - - -
JMP	xxxx ea		1+ea	4+jx	- - - - - - - -
JScc	xxxx ea		1+ea	4+jx	- - - - - - - -
JSCLR	#n,X:<ea>,xxxx #n,X:<aa>,xxxx #n,X:<pp>,xxxx #n,Y:<ea>,xxxx #n,Y:<aa>,xxxx #n,Y:<pp>,xxxx #n,S,xxxx		2	6+jx	* * - - - - - -
JSET	#n,X:<ea>,xxxx #n,X:<aa>,xxxx #n,X:<pp>,xxxx #n,Y:<ea>,xxxx #n,Y:<aa>,xxxx #n,Y:<pp>,xxxx #n,S,xxxx		2	6+jx	* * - - - - - -
JSR	xxx ea		1+ea	4+jx	- - - - - - - -
JSSET	#n,X:<ea>,xxxx #n,X:<aa>,xxxx #n,X:<pp>,xxxx #n,Y:<ea>,xxxx #n,Y:<aa>,xxxx #n,Y:<pp>,xxxx #n,S,xxxx		2	6+jx	* * - - - - - -
LSL	D	(parallel move)	1+mv	2+mv	* * - - ? ? 0 ?
LSR	D	(parallel move)	1+mv	2+mv	* * - - ? ? 0 ?
LUA	<ea>,D		1	4	- - - - - - - -
MAC	(±)S2,S1,D	(parallel move)	1+mv	2+mv	* * * * * * * -
	(±)S1,S2,D	(parallel move)			
	(±)S,#n,D	(**no** parallel move)	1	2	

INSTRUCTIONS

Table B-3 Instruction Set Summary — Sheet 3 of 5

Mnemonic	Syntax	Parallel Moves		Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
MACR	(±)S2,S1,D	(parallel move)		1+mv	2+mv	* * * * * * * -
	(±)S1,S2,D	(parallel move)				
	(±)S,#n,D	(**no** parallel move)		1	2	
MOVE	S,D			1+mv	2+mv	* * - - - - - -
No parallel data move		(.....)		mv	mv	- - - - - - - -
Immediate short data move		(.....)#xx,D		mv	mv	- - - - - - - -
Register to register data move		(.....)S,D		mv	mv	* * - - - - - -
Address register update		(.....)ea		mv	mv	- - - - - - - -
X memory data move		(.....)X:<ea>,D		mv	mv	* * - - - - - -
		(.....)X:<aa>,D				
		(.....)S,X:<ea>				
		(.....)S,X:<aa>				
		(.....)#xxxxxx,D				
X memory and register data move		(.....)X:<ea>,D1	S2,D2	mv	mv	* * - - - - - -
		(.....)S1,X:<ea>	S2,D2			
		(.....)#xxxxxx,D1	S2,D2			
		(.....)A,X:<ea>	X0,A			
		(.....)B,X:<ea>	X0,B			
Y memory data move		(.....)Y:<ea>,D		mv	mv	* * - - - - - -
		(.....)Y:<aa>,D				
		(.....)S,Y:<ea>				
		(.....)S,Y:<aa>				
		(.....)#xxxxxx,D				
Register and Y memory data move		(.....)S1,D1	Y:<ea>,D2	mv	mv	* * - - - - - -
		(.....)S1,D1	S2,Y:<ea>			
		(.....)S1,D1	#xxxxxx,D2			
		(.....)Y0,A	A,Y:<ea>			
		(.....)Y0,B	B,Y:<ea>			
Long memory data move		(.....)L:<ea>,D		mv	mv	* * - - - - - -
		(.....)L:<aa>,D				
		(.....)S,L:<ea>				
		(.....)S,L:<aa>				
XY memory data move		(.....)X:<eax>,D1	Y:<eay>,D2	mv	mv	* * - - - - - -
		(.....)X:<eax>,D1	S2,Y:<eay>			
		(.....)S1,X:<eax>	Y:<eay>,D2			
		(.....)S1,X:<eax>	S2,Y:<eay>			
MOVE(C)	X:<ea>,D1			1+ea	2+mvc	? ? ? ? ? ? ? ?
	X:<aa>,D1					
	S1,X:<ea>					
	S1,X:<aa>					
	Y:<ea>,D1					
	Y:<aa>,D1					
	S1,Y:<ea>					
	S1,Y:<aa>					
	S1,D2					
	S2,D1					
	#xxxx,D1					
	#xx,D1					

INSTRUCTIONS

Table B-3 Instruction Set Summary — Sheet 4 of 5

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
MOVE(M)	P:<ea>,D	. . . .	1+ea	2+mvm	??? ?????
	S,P:<ea>				
	S,P:<aa>				
	P:<aa>,D				
MOVE(P)	X:<pp>,D	. . . .	1+ea	2+mvp	??? ?????
	X:<pp>,X:<ea>				
	X:<pp>,Y:<ea>				
	X:<pp>,P:<ea>				
	S,X:<pp>				
	#xxxxxx,X:<pp>				
	X:<ea>,X:<pp>				
	Y:<ea>,X:<pp>				
	P:<ea>,X:<pp>				
	Y:<pp>,D				
	Y:<pp>,X:<ea>				
	Y:<pp>,Y:<ea>				
	Y:<pp>,P:<ea>				
	S,Y:<pp>				
	#xxxxxx,Y:<pp>				
	X:<ea>,Y:<pp>				
	Y:<ea>,Y:<pp>				
	P:<ea>,Y:<pp>				
MPY	(±)S2,S1,D	(parallel move)	1+mv	2+mv	* * * * * * * -
	(±)S1,S2,D	(parallel move)			
	(±)S,#n,D	(**no** parallel move).	1	2	
MPYR	(±)S2,S1,D	(parallel move)	1+mv	2+mv	* * * * * * * -
	(±)S1,S2,D	(parallel move)			
	(±)S,#n,D	(**no** parallel move).	1	2	
NEG	D	(parallel move)	1+mv	2+mv	* * * * * * * -
NOP		. . . .	1	2	- - - - - - - -
NORM	Rn,D	. . . .	1	2	- * * * * * ?-
NOT	D	(parallel move).	1+mv	2+mv	* * - - ??0-
OR	S,D	(parallel move).	1+mv	2+mv	* * - - ??0-
ORI	#xx,D	. . . .	1	2	??? ?????
REP	X:<ea>	. . . .	1	4+mv	??- - - - - -
	X:<aa>				
	Y:<ea>				
	Y:<aa>				
	S				
	#xxx				

INSTRUCTIONS

Table B-3 Instruction Set Summary — Sheet 5 of 5

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
RESET		. . . .	1	4	- - - - - - - -
RND	D	(parallel move)	1+mv	2+mv	* * * * * * * -
ROL	D	(parallel move)	1+mv	2+mv	* * - - ? ? 0 ?
ROR	D	(parallel move)	1+mv	2+mv	* * - - ? ? 0 ?
RTI		. . . .	1	4+rx	? ? ? ? ? ? ? ?
RTS		. . . .	1	4+rx	- - - - - - - -
SBC	S,D	(parallel move)	1+mv	2+mv	* * * * * * * *
STOP		. . . .	1	n/a	- - - - - - - -
SUB	S,D	(parallel move)	1+mv	2+mv	* * * * * * * *
SUBL	S,D	(parallel move)	1+mv	2+mv	* * * * * * ? *
SUBR	S,D	(parallel move)	1+mv	2+mv	* * * * * * * *
SWI		. . . .	1	8	- - - - - - - -
Tcc	S1,D1	. . . .	1	2	- - - - - - - -
	S1,D1 S2,D2				
TFR	S,D	(parallel move)	1+mv	2+mv	* * - - - - - -
TST	S	(parallel move)	1+mv	2+mv	* * * * * * 0 -
WAIT		. . . .	1	n/a	- - - - - - - -

NOTATION:
- denotes the bit is unaffected by the operation.
* denotes the bit may be set according to the definition, depending on parallel move conditions.
? denotes the bit is set according to a special definition. See the instruction descriptions in **Appendix A** of the DSP56000 Family Manual (DSP56KFAMUM/AD).
0 denotes the bit is cleared.

Application: ______________________________

____

Date: ______________

Programmer: ______________

Sheet 1 of 4

CENTRAL PROCESSOR

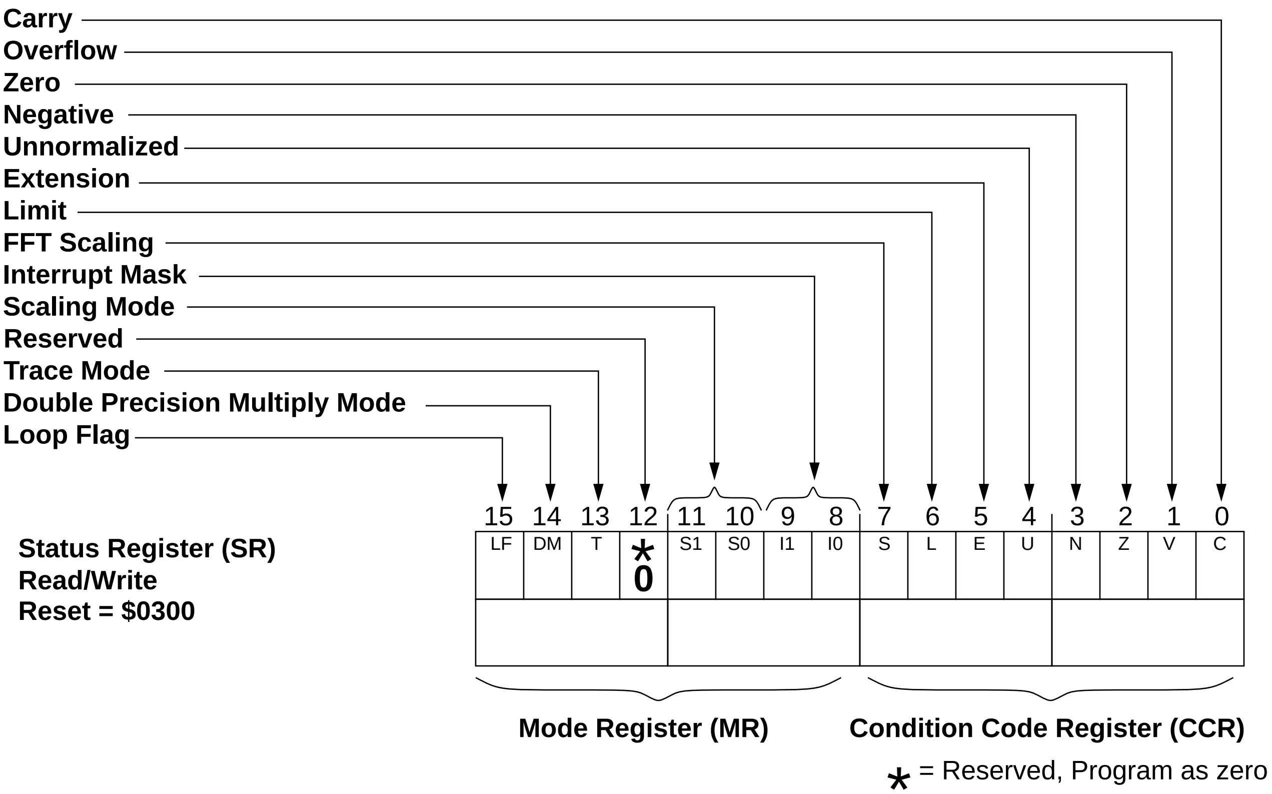


Figure B-2 Status Register (SR)

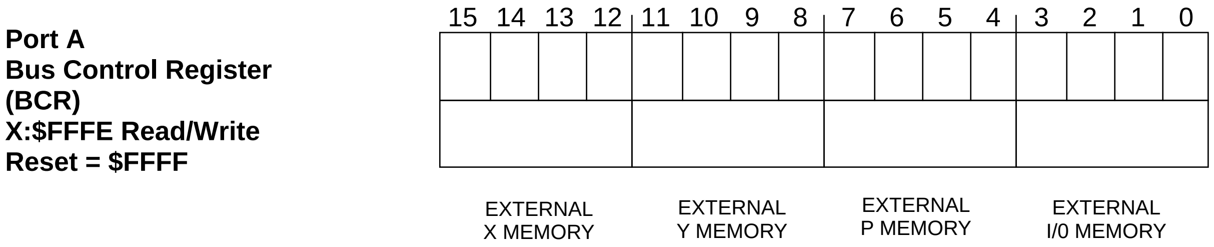


Figure B-3 Bus Control Register (BCR)

Application: ____________

Date: ____________

Programmer: ____________

Sheet 2 of 4

CENTRAL PROCESSOR

CENTRAL PROCESSOR

SSI **IPL**

SSL1	SSL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

SCI **IPL**

SCL1	SCL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

TIMER/COUNTER **IPL**

TIL1	TIL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

PWM **IPL**

PWL1	PWL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

$\overline{\text{IRQA}}$ Mode

IAL2	Trigger
0	Level
1	Neg. Edge

IAL1	IAL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

$\overline{\text{IRQB}}$ Mode

IBL2	Trigger
0	Level
1	Neg. Edge

IBL1	IBL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

Host IPL

HPL1	HPL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

$\overline{\text{IRQC}}$ Mode

ICL1	ICL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

$\overline{\text{IRQD}}$ Mode

IDL1	IDL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

Interrupt Priority Register (IPR)
X:$FFFF Read/Write
Reset = $000000

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	PWL1	PWL0	TIL1	TIL0	SCL1	SCL0	SSL1	SSL0	HPL1	HPL0	IDL1	IDL0	ICL1	ICL0	IBL2	IBL1	IBL0	IAL2	IAL1	IAL0
$0																							

* = Reserved, Program as zero

Figure B-4 Interrupt Priority Register (IPR)

Application: ____________________

Date: ____________________

Programmer: ____________________

Sheet 3 of 4

CENTRAL PROCESSOR

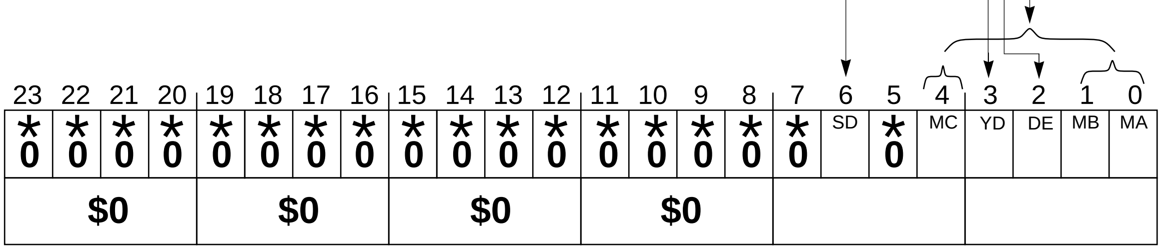
Mode	M M M C B A	Operating Mode
0	0 0 0	Single-Chip Mode
1	0 0 1	Bootstrap from EPROM at P: $C000
2	0 1 0	Normal Expanded Mode
3	0 1 1	Development Mode
4	1 0 0	Reserved
5	1 0 1	Bootstrap from Host
6	1 1 0	Bootstrap from SCI (external clock)
7	1 1 1	Bootstrap from EPROM at P: $8000

Data ROM Enable
0 = Disable ROMs
1 = Enable ROMs

Internal Y Memory Disable
0 = Y Memory controlled by DE bit
1 = All Y Memory external

Stop Delay
0 = 128K T Stabilization
1 = 16 T Stabilization

Operating Mode
Register (OMR)
Read/Write
Reset = $000000


23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*0 *0 *0 *0 *0 *0 *0 *0 *0 *0 *0 *0 *0 *0 *0 *0 *0 SD *0 MC YD DE MB MA
$0
$0
$0
$0

* = Bit 5 and bits 7 through 23 are reserved. Program as zero

Figure B-5 Operating Mode Register (OMR)

Application: ____

Date: ____

Programmer: ____

Sheet 4 of 4

CENTRAL PROCESSOR

XTAL Disable Bit (XTLD)
0 = Enable XTAL
1 = Disable XTAL

STOP Processing State Bit (PSTP)
0 = PLL Disabled During STOP Processing State
1 = PLL Enabled During STOP Processing State

Clock Output Disable Bits COD0 - COD1

COD1	COD0	CLKOUT Pin
0	0	Clock Out Enabled, Full Strength Output Buffer
0	1	Clock Out Enabled, 2/3 Strength Output Buffer
1	0	Clock Out Enabled, 1/3 Strength Output Buffer
1	1	Clock Out Disabled

Chip Clock Source Bit (CSRC)
0 = Output from Low Power Divider
1 = Output from VCO

PLL Enable Bit (PEN)
0 = Disable PLL
1 = Enable PLL

CKOUT Clock Source Bit (CKOS)
0 = Output from LPD
1 = Output from VCO

Multiplication Factor Bits MF0 - MF11

MF11 - MF0	Multiplication Factor MF
$000	1
$001	2
$002	3
.	.
.	.
.	.
$FFE	4095
$FFF	4096

Division Factor Bits DF0 - DF3

DF3 - DF0	Division Factor DF
$0	2^0
$1	2^1
$2	2^2
.	.
.	.
.	.
$E	2^{14}
$F	2^{15}

PLL Control Register (PCTL)
X:$FFFD Read/Write
Reset = $0X0000

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	CKOS	CSRC	COD1	COD0	PEN	PSTP	XTLD	DF3	DF2	DF1	DF0	MF11	MF10	MF9	MF8	MF7	MF6	MF5	MF4	MF3	MF2	MF1	MF0

Figure B-6 PLL Control Register (PCTL)

* = Reserved, Program as zero

Application: ______________________________ Date: ______________

______________________________ Programmer: ______________

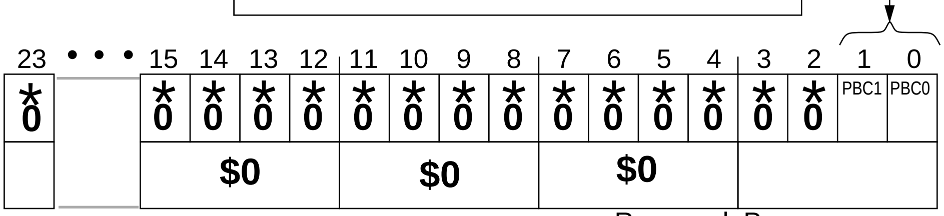
Sheet 1 of 2

GPIO

Port B

PBC1	PBC0	Function
0	0	General Purpose I/O (Reset Condition)
0	1	Host Interface
1	0	Host Interface (with $\overline{\text{HACK}}$ pin as GPIO)
1	1	Reserved

Port B
Control Register (PBC)
X:$FFE0 Read/Write
Reset = $000000

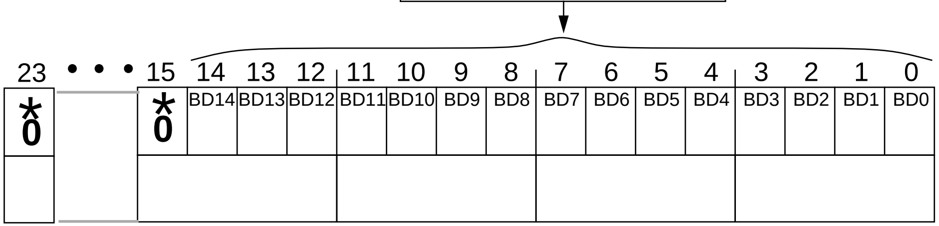


* = Reserved, Program as zero

Figure B-7 Port B Control Register (PBC)

Port B Data Direction Control
0 = Input
1 = Output

Port B
Data Direction
Register (PBDDR)
X:$FFE2 Read/Write
Reset = $000000

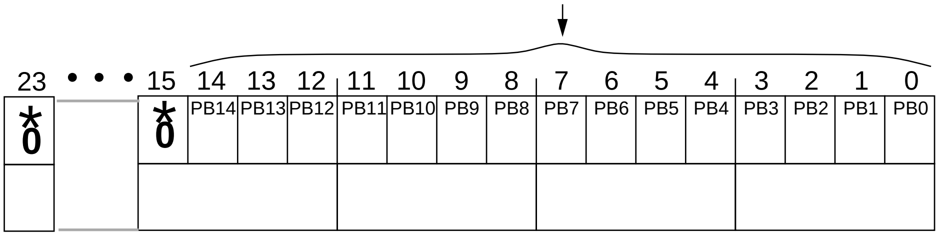


* = Reserved, Program as zero

Figure B-8 Port B Data Direction Register (PBDDR)

Port B Data (usually loaded by program)

Port B
Data Register (PBD)
X:$FFE4 Read/Write
Reset = $000000



* = Reserved, Program as zero

Figure B-9 Port B Data Register (PBD)

Application: ______________________________________ Date: ________________

______________________________________ Programmer: ____________

Sheet 2 of 2

GP I/O

Port C

Port C
Control Register (PCC)
X:$FFE1 Read/Write
Reset = $000000

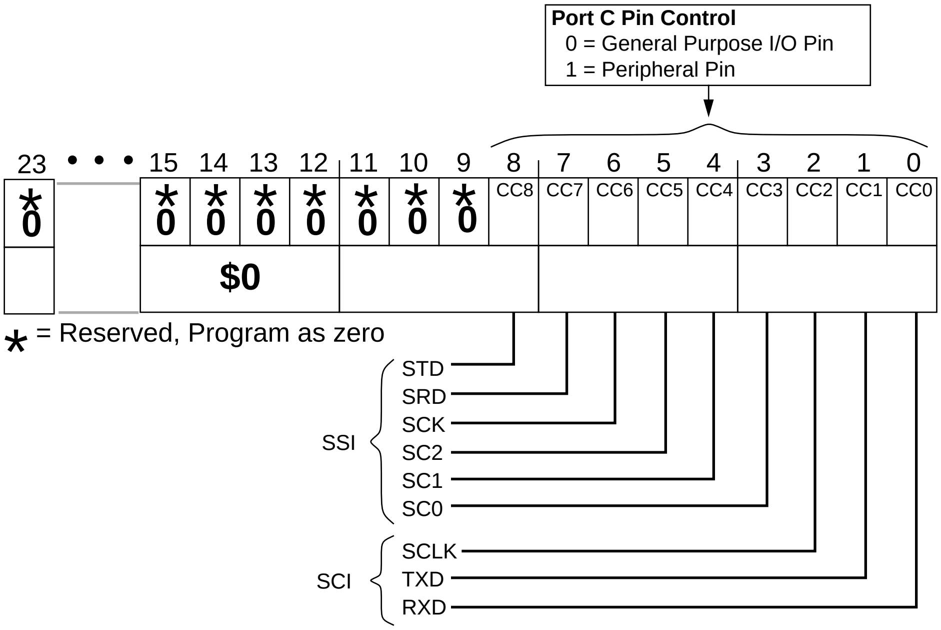


Figure B-10 Port C Control Register (PCC)

Port C
Data Direction
Register (PCDDR)
X:$FFE3 Read/Write
Reset = $000000

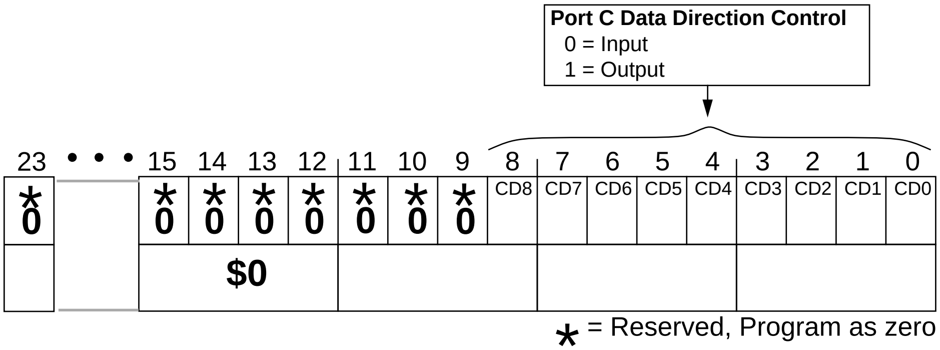

Port C Data Direction Control
0 = Input
1 = Output
23 • • • 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*
0
CD8 CD7 CD6 CD5 CD4 CD3 CD2 CD1 CD0
$0
* = Reserved, Program as zero

Figure B-11 Port C Data Direction Register (PCDDR)

Port C
Data Register (PCD)
X:$FFE5 Read/Write
Reset = $000000

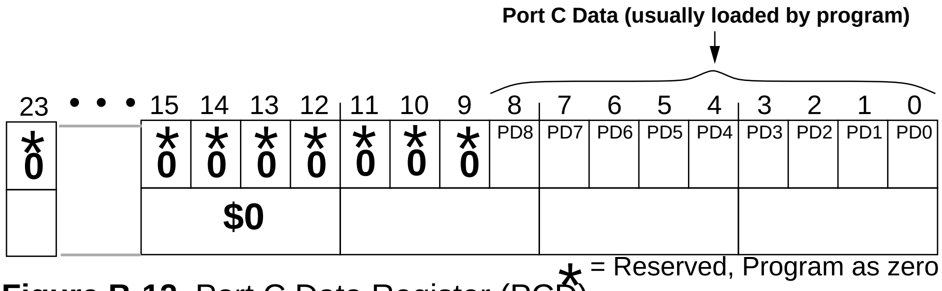

Port C Data (usually loaded by program)
23 • • • 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*
0
PD8 PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0
$0
* = Reserved, Program as zero

Figure B-12 Port C Data Register (PCD)

Application: ____ Date: ____________________

____ Programmer: ______________

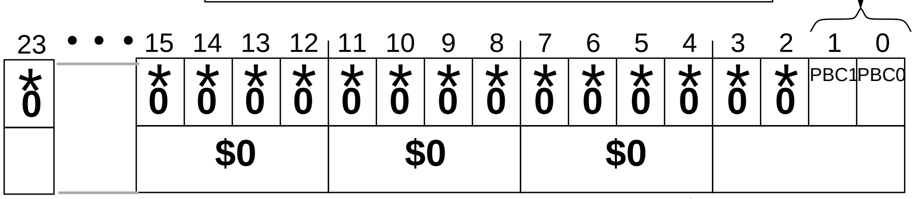
Sheet 1 of 5

HOST

Port B

PBC1	PBC0	Function
0	0	General Purpose I/O (Reset Condition)
0	1	Host Interface
1	0	Host Interface (with $\overline{\text{HACK}}$ pin as GPIO)
1	1	Reserved

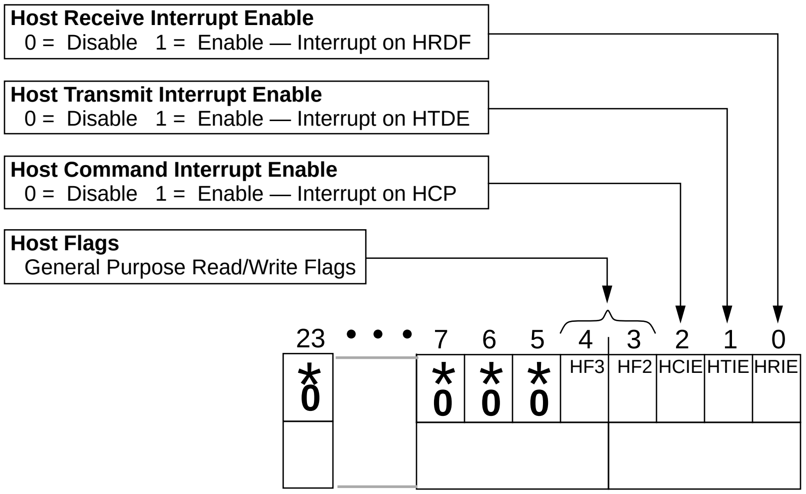
Port B
Control Register (PBC)
X:$FFE0 Read/Write
Reset = $000000



* = Reserved, Program as zero

Figure B-13 Port B Control Register (PBC)

DSP SIDE



Host Receive Interrupt Enable
0 = Disable 1 = Enable — Interrupt on HRDF

Host Transmit Interrupt Enable
0 = Disable 1 = Enable — Interrupt on HTDE

Host Command Interrupt Enable
0 = Disable 1 = Enable — Interrupt on HCP

Host Flags
General Purpose Read/Write Flags

Host Control Register (HCR)
X:$FFE8 Read/Write
Reset = $00

* = Reserved, Program as zero

Figure B-14 Host Control Register (HCR)

Application: ______________________________ Date: ______________

______________________________ Programmer: ______________

Sheet 2 of 5

HOST

DSP SIDE

Host Status Register (HSR)
X:$FFE9 Read Only
Reset = $000002

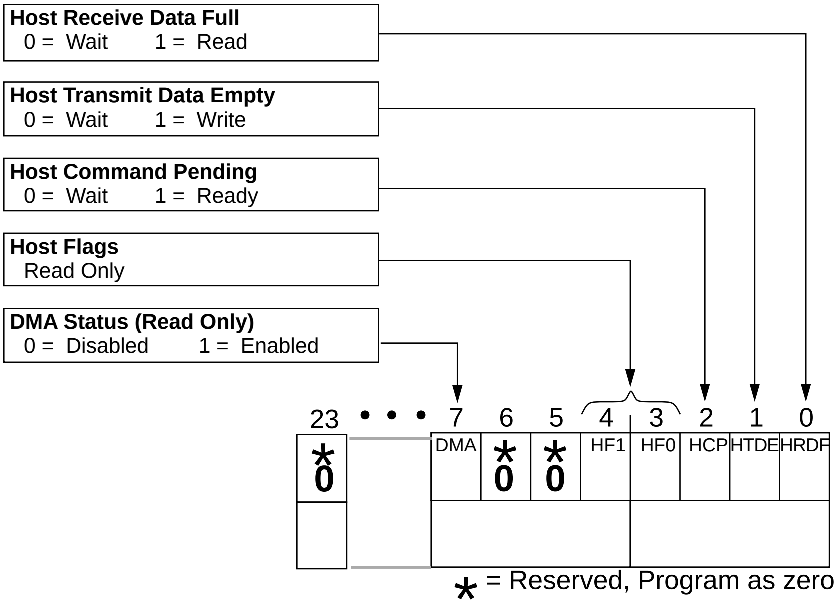


Figure B-15 Host Status Register (HSR)

Host Receive Data Register (HRX)
X:$FFEB Read Only
Reset = $000000

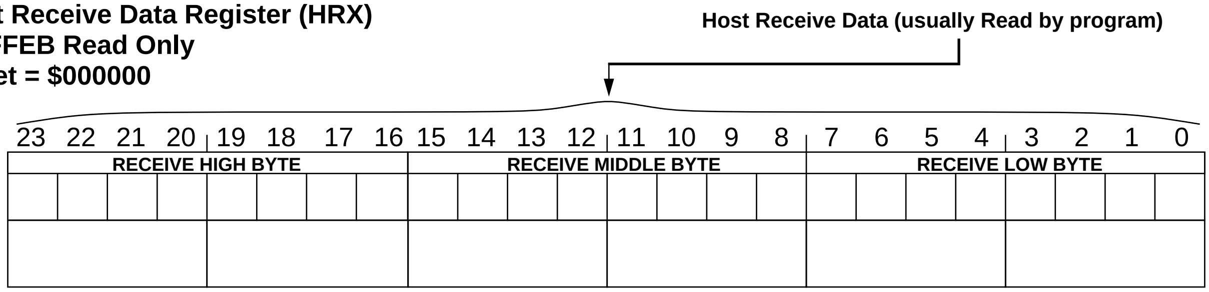


Figure B-16 Host Receive Data Register (HRX)

Host Transmit Data Register (HTX)
X:$FFEB Write Only
Reset = $000000

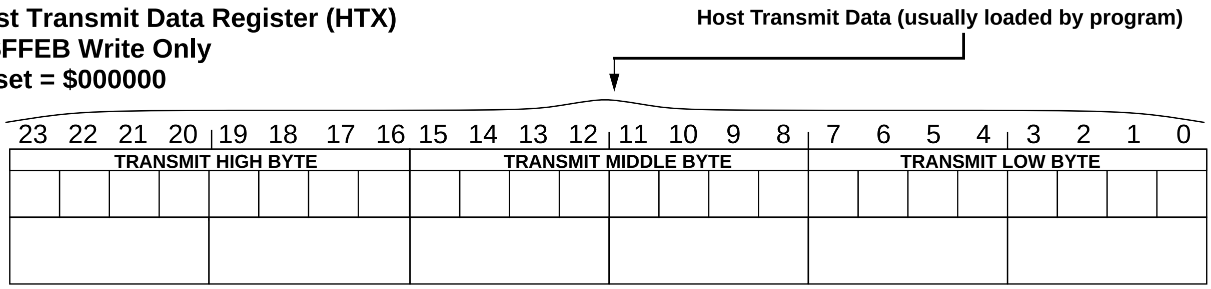


Figure B-17 Host Transmit Data Register (HTX)

Application: ______________________________ Date: ______________

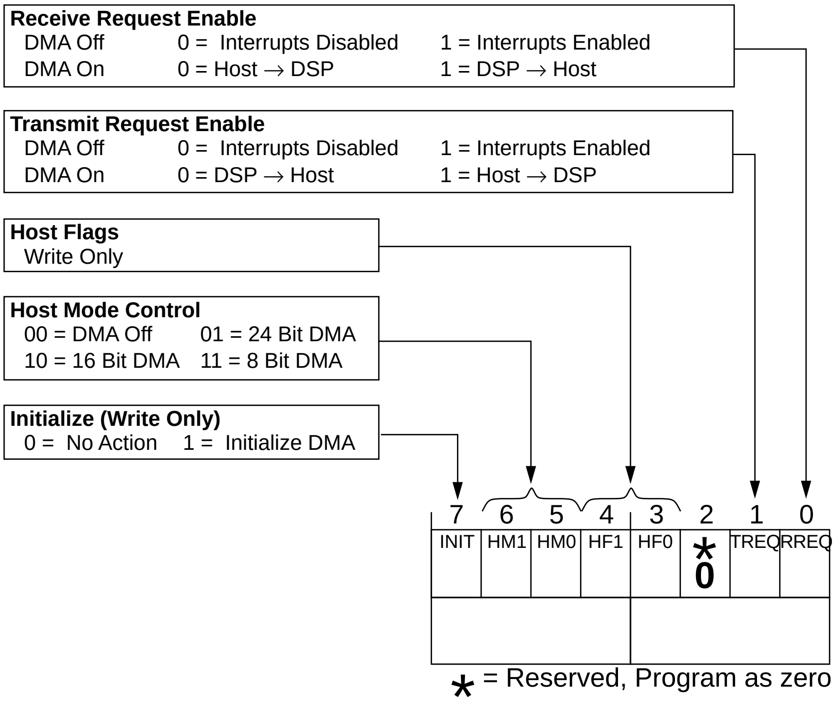
______________________________ Programmer: ______________

Sheet 3 of 5

HOST

PROCESSOR SIDE

Interrupt Control Register (ICR)
$0 Read/Write
Reset = $00



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

Host Flags
Write Only

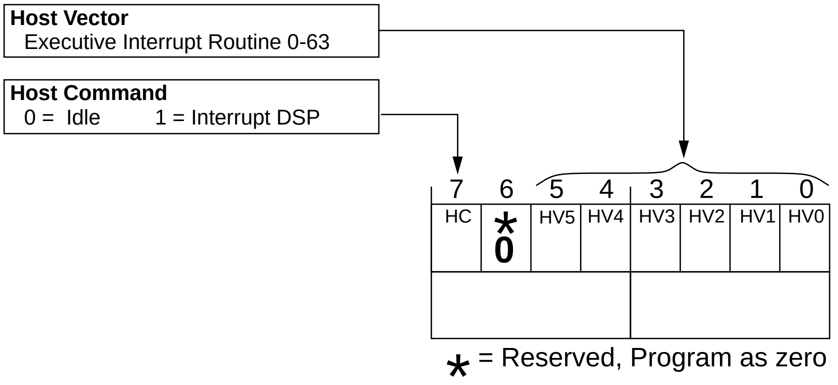
Host Mode Control
00 = DMA Off 01 = 24 Bit DMA
10 = 16 Bit DMA 11 = 8 Bit DMA

Initialize (Write Only)
0 = No Action 1 = Initialize DMA

* = Reserved, Program as zero

Figure B-18 Interrupt Control Register (ICR)

Command Vector Register (CVR)
$1 Read/Write
Reset = $12



Host Vector
Executive Interrupt Routine 0-63

Host Command
0 = Idle 1 = Interrupt DSP

* = Reserved, Program as zero

Figure B-19 Command Vector Register (CVR)

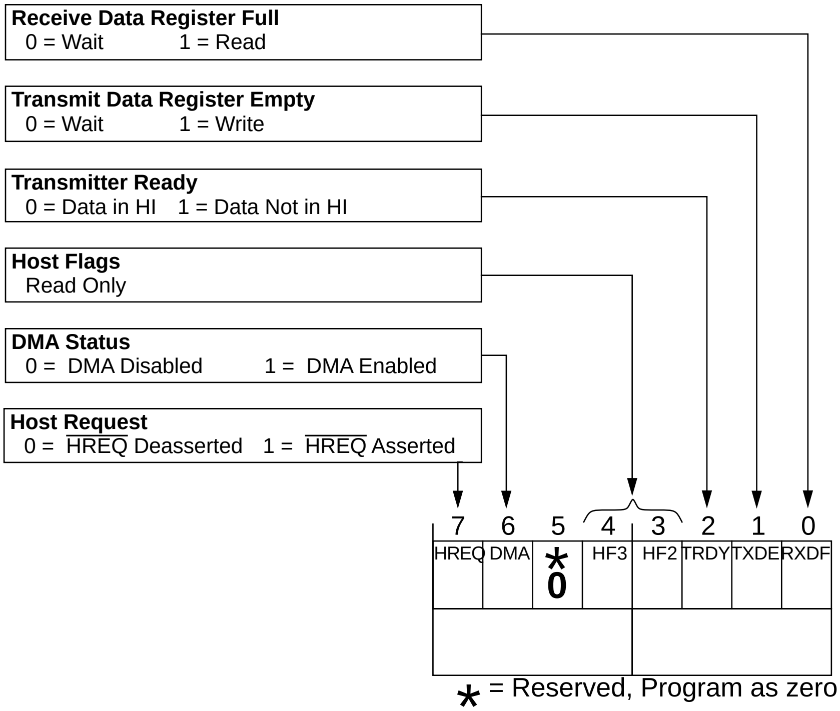
Application: ____ Date: ____________________

____ Programmer: ______________

Sheet 4 of 5

HOST

PROCESSOR SIDE



Receive Data Register Full
0 = Wait 1 = Read

Transmit Data Register Empty
0 = Wait 1 = Write

Transmitter Ready
0 = Data in HI 1 = Data Not in HI

Host Flags
Read Only

DMA Status
0 = DMA Disabled 1 = DMA Enabled

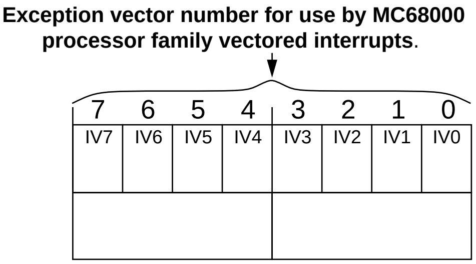
Host Request
0 = $\overline{\text{HREQ}}$ Deasserted 1 = $\overline{\text{HREQ}}$ Asserted

Interrupt Status Register (ISR)
$2 Read/Write
Reset = $06

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

* = Reserved, Program as zero

Figure B-20 Interrupt Status Register (ISR)



Exception vector number for use by MC68000 processor family vectored interrupts.

Interrupt Vector Register (IVR)
$3 Read/Write
Reset = $0F

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

Figure B-21 Interrupt Vector Register (IVR)

HOST

PROCESSOR SIDE

Receive Byte Registers
$7, $6, $5, $4 Read Only
Reset = $00

Host Receive Data (usually read by program)

7 0	7 0	7 0	7 0
RECEIVE LOW BYTE	RECEIVE MIDDLE BYTE	RECEIVE HIGH BYTE	NOT USED
			0 0 0 0 0 0 0 0
$7	$6	$5	$4

Figure B-22 Host Receive Byte Registers (RXH/RXM/RXL)

Transmit Byte Registers
$7, $6, $5, $4 Write Only
Reset = $00

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

Figure B-23 Host Transmit Byte Registers (TXH/TXM/TXL)

Application: ______

Date: ______

Programmer: ______

Sheet 5 of 5

Application: ______________________________________

____

Date: ____________________

Programmer: ______________

Sheet 1 of 3

SCI

Port C

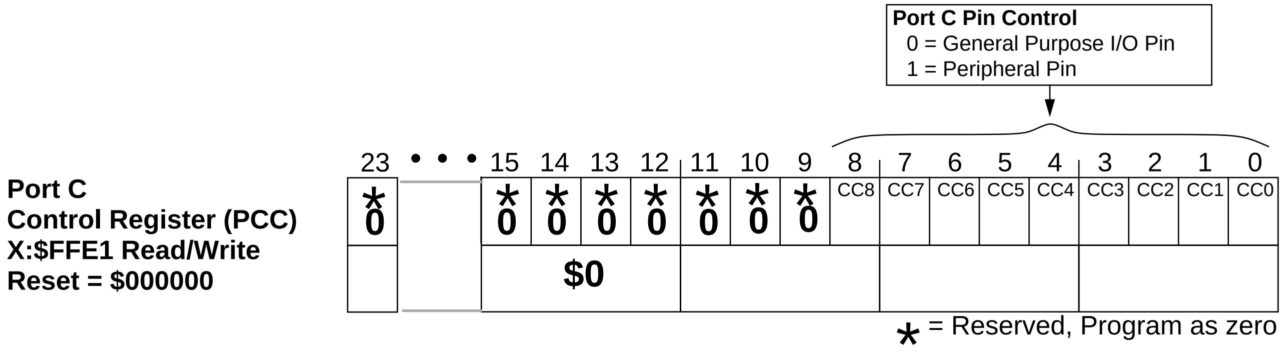


Figure B-24 Port C Control Register (PCC)

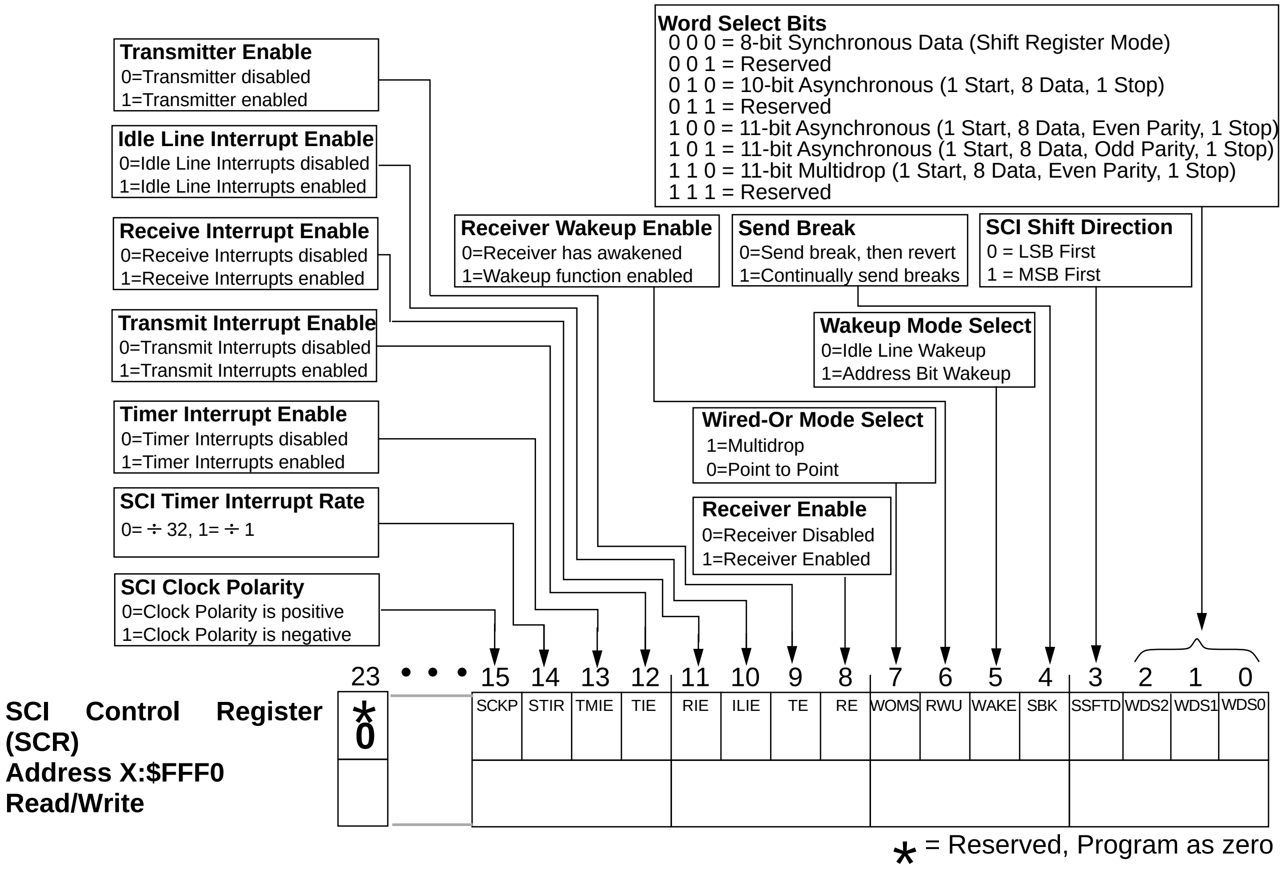

Word Select Bits
0 0 0 = 8-bit Synchronous Data (Shift Register Mode)
0 0 1 = Reserved
0 1 0 = 10-bit Asynchronous (1 Start, 8 Data, 1 Stop)
0 1 1 = Reserved
1 0 0 = 11-bit Asynchronous (1 Start, 8 Data, Even Parity, 1 Stop)
1 0 1 = 11-bit Asynchronous (1 Start, 8 Data, Odd Parity, 1 Stop)
1 1 0 = 11-bit Multidrop (1 Start, 8 Data, Even Parity, 1 Stop)
1 1 1 = Reserved
Transmitter Enable
0=Transmitter disabled
1=Transmitter enabled
Idle Line Interrupt Enable
0=Idle Line Interrupts disabled
1=Idle Line Interrupts enabled
Receive Interrupt Enable
0=Receive Interrupts disabled
1=Receive Interrupts 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
Receiver Wakeup Enable
0=Receiver has awakened
1=Wakeup function enabled
Send Break
0=Send break, then revert
1=Continually send breaks
SCI Shift Direction
0 = LSB First
1 = MSB First
Wakeup Mode Select
0=Idle Line Wakeup
1=Address Bit Wakeup
Wired-Or Mode Select
1=Multidrop
0=Point to Point
Receiver Enable
0=Receiver Disabled
1=Receiver Enabled
SCI Control Register
(SCR)
Address X:$FFF0
Read/Write
23 • • • 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*0
SCKP STIR TMIE TIE RIE ILIE TE RE WOMS RWU WAKE SBK SSFTD WDS2 WDS1 WDS0
* = Reserved, Program as zero

Figure B-25 SCI Control Register (SCR)

Application: ______________________________

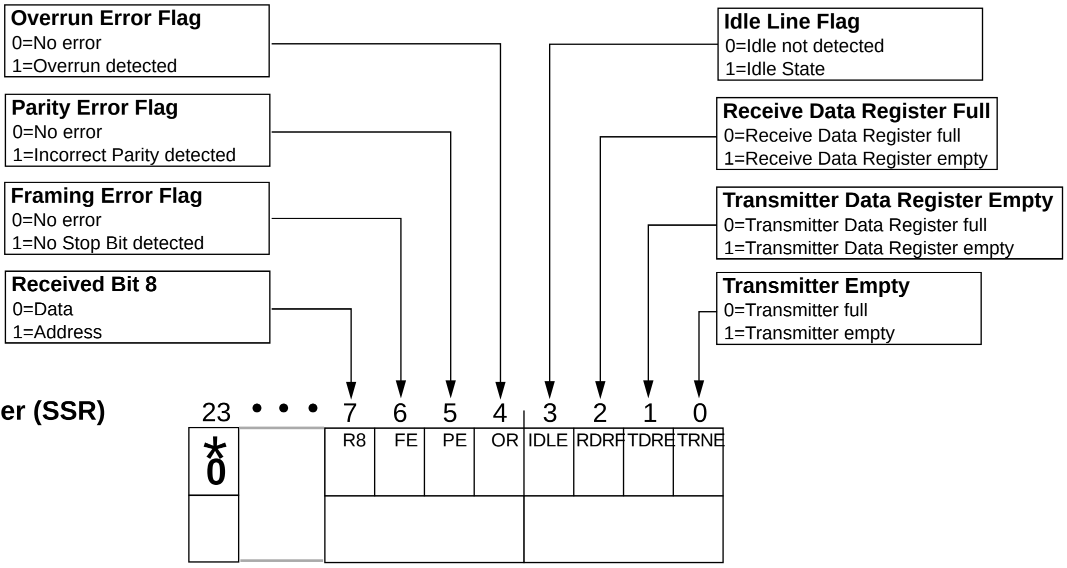
____

Date: ______________

Programmer: ______________

Sheet 2 of 3

SCI



Overrun Error Flag
0=No error
1=Overrun detected

Parity Error Flag
0=No error
1=Incorrect Parity detected

Framing Error Flag
0=No error
1=No Stop Bit detected

Received Bit 8
0=Data
1=Address

Idle Line Flag
0=Idle not detected
1=Idle State

Receive Data Register Full
0=Receive Data Register full
1=Receive Data Register empty

Transmitter Data Register Empty
0=Transmitter Data Register full
1=Transmitter Data Register empty

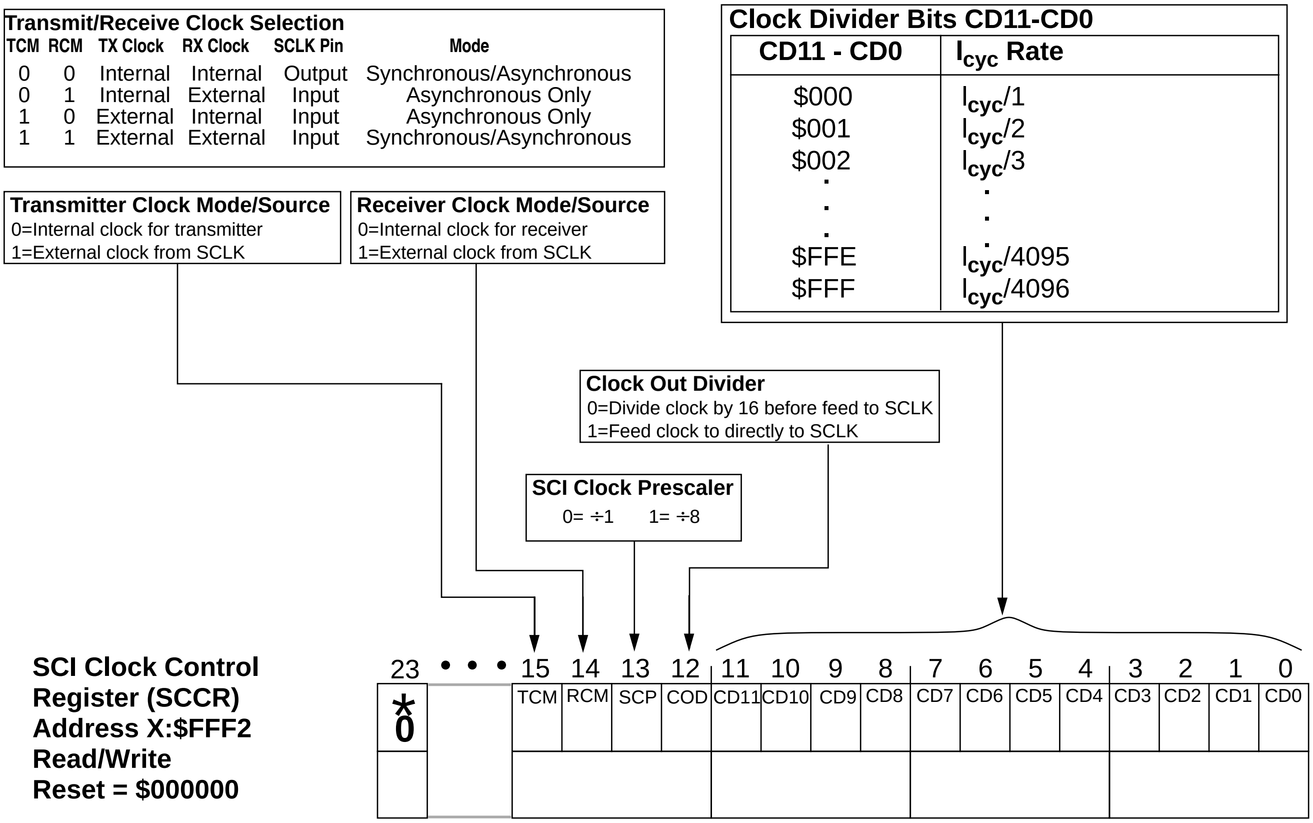
Transmitter Empty
0=Transmitter full
1=Transmitter empty

SCI Status Register (SSR)
Address X:$FFF1
Read Only
Reset = $000003

23	• • •	7	6	5	4	3	2	1	0
* 0		R8	FE	PE	OR	IDLE	RDRF	TDRE	TRNE

* = Reserved, Program as zero

Figure B-26 SCI Status Register (SSR)



Transmit/Receive Clock Selection

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:$FFF2
Read/Write
Reset = $000000

23	• • •	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0		TCM	RCM	SCP	COD	CD11	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0

* = Reserved, Program as zero

Figure B-27 SCI Clock Control Register (SCCR)

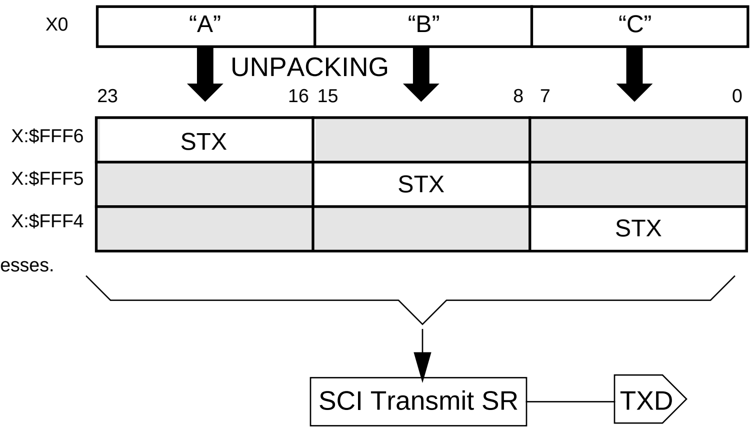
Application: ____ Date: __________________

____ Programmer: ______________

Sheet 3 of 3

SCI

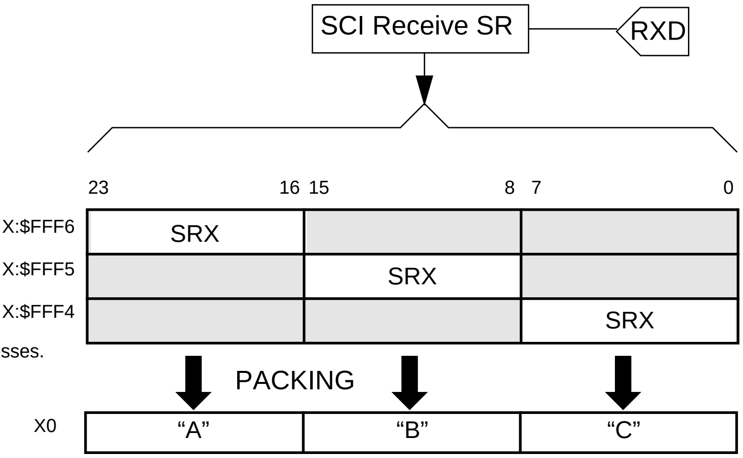
SCI Transmit Data Registers
Address X:$FFF4 – X:$FFF6 Read/Write
Reset = xxxxxx



NOTE: STX is the same register decoded at three different addresses.

Figure B-28 SCI Transmit Data Registers (STX)

SCI Receive Data Registers
Address X:$FFF4 - X:$FFF6 Read/Write
Reset = xxxxxx



NOTE: SRX is the same register decoded at three different addresses.

Figure B-29 SCI Receive Data Registers (SRX)

Application: ____________________________________ Date: ____________________

____________________________________ Programmer: ______________

Sheet 1 of 3

SSI

Port C

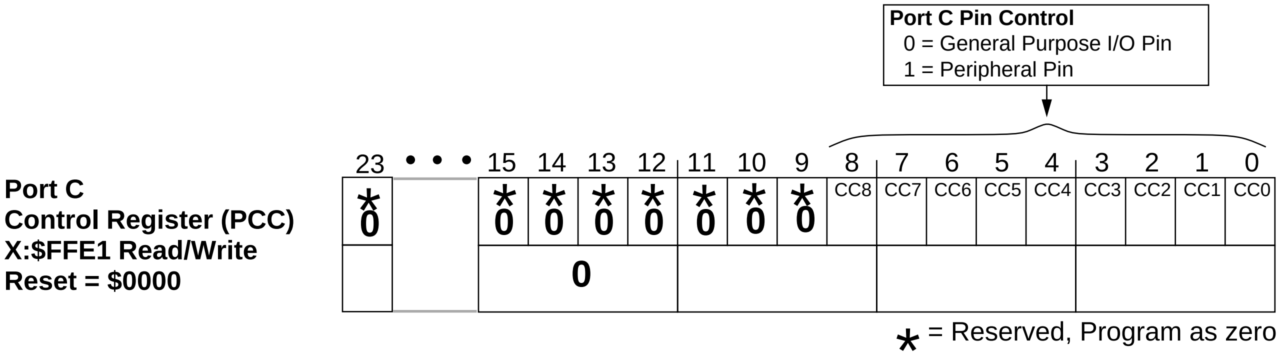


Figure B-30 SSI Control Register (PCC)

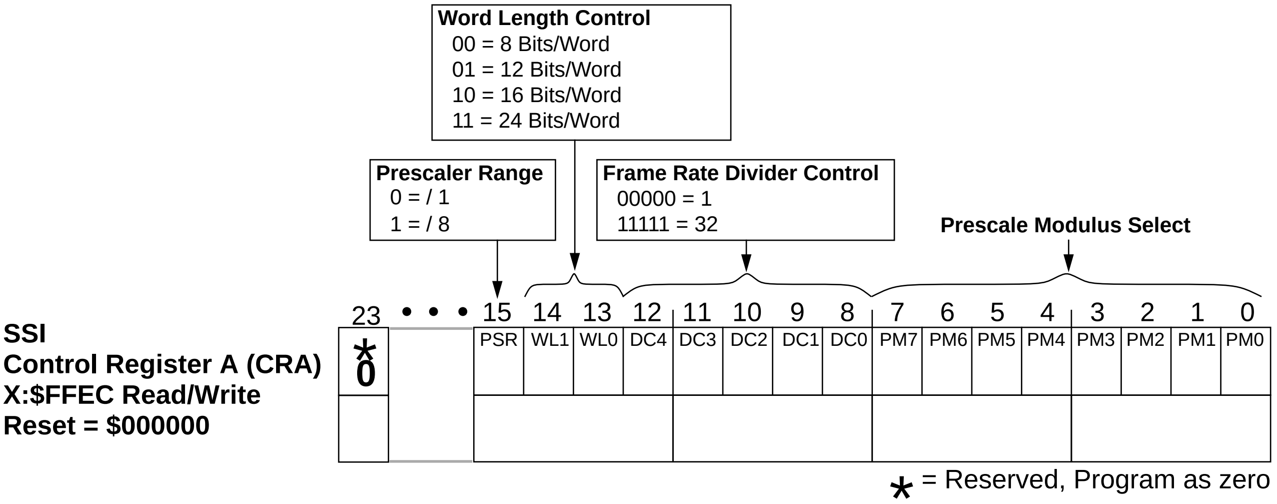


Figure B-31 SSI Control Register A (CRA)

Application: ____ Date: ____________________

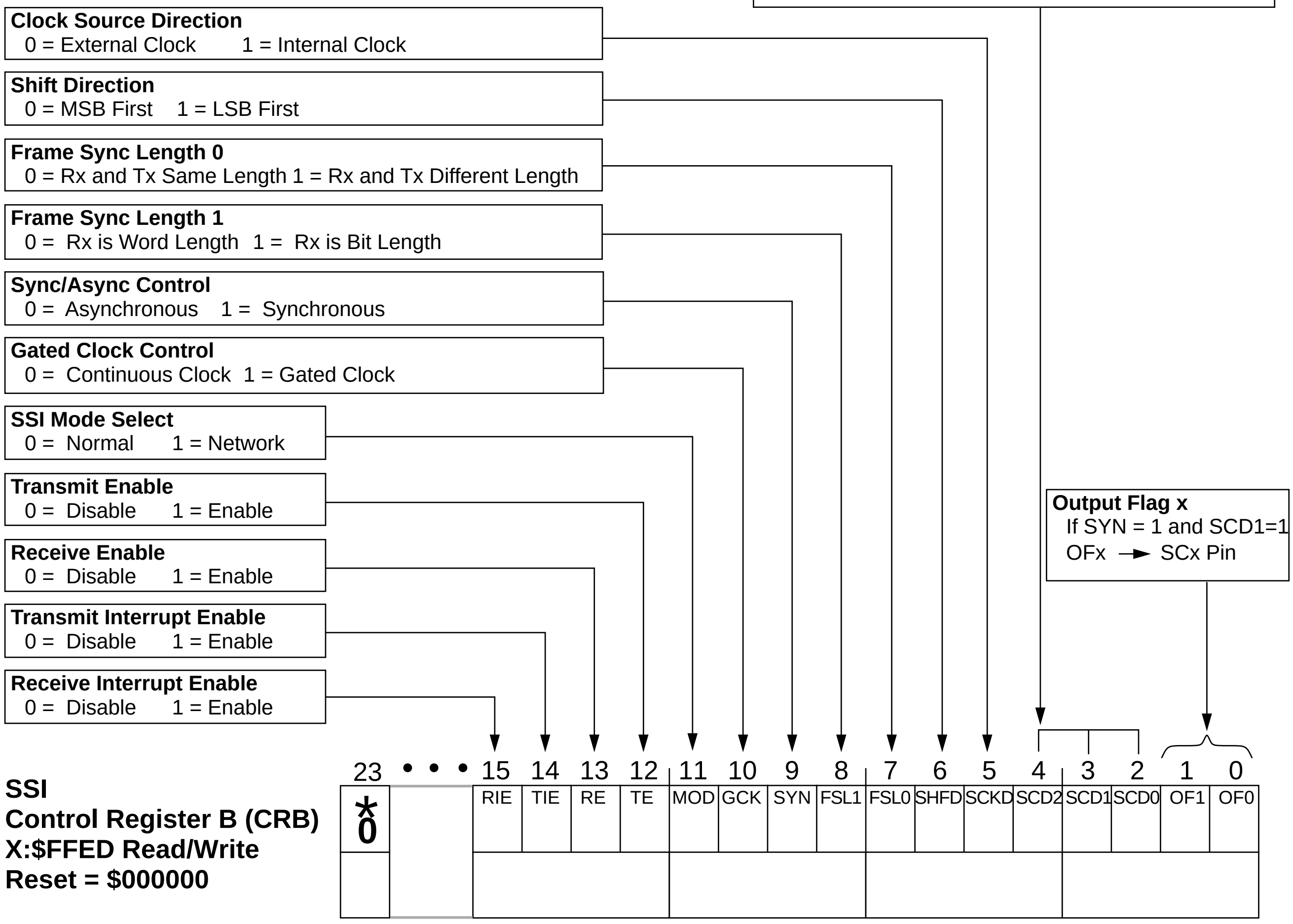
____ Programmer: ______________

Sheet 2 of 3

SSI

Serial Control Direction Bits

	SCDx=0 (Input)	**SCDx=1 (Output)**
SC0 Pin	Rx Clk	Flag 0
SC1 Pin	Rx Frame Sync	Flag 1
SC2 Pin	Tx Frame Sync	Tx, Rx Frame Sync



Clock Source Direction
0 = External Clock 1 = Internal Clock

Shift Direction
0 = MSB First 1 = LSB First

Frame Sync Length 0
0 = Rx and Tx Same Length 1 = Rx and Tx Different Length

Frame Sync Length 1
0 = Rx is Word Length 1 = Rx is Bit Length

Sync/Async Control
0 = Asynchronous 1 = Synchronous

Gated Clock Control
0 = Continuous Clock 1 = Gated Clock

SSI Mode Select
0 = Normal 1 = Network

Transmit Enable
0 = Disable 1 = Enable

Receive Enable
0 = Disable 1 = Enable

Transmit Interrupt Enable
0 = Disable 1 = Enable

Receive Interrupt Enable
0 = Disable 1 = Enable

Output Flag x
If SYN = 1 and SCD1=1
OFx → SCx Pin

SSI
Control Register B (CRB)
X:$FFED Read/Write
Reset = $000000

* = Reserved, Program as zero

Figure B-32 SSI Control Register B (CRB)

Application: ______________________________________

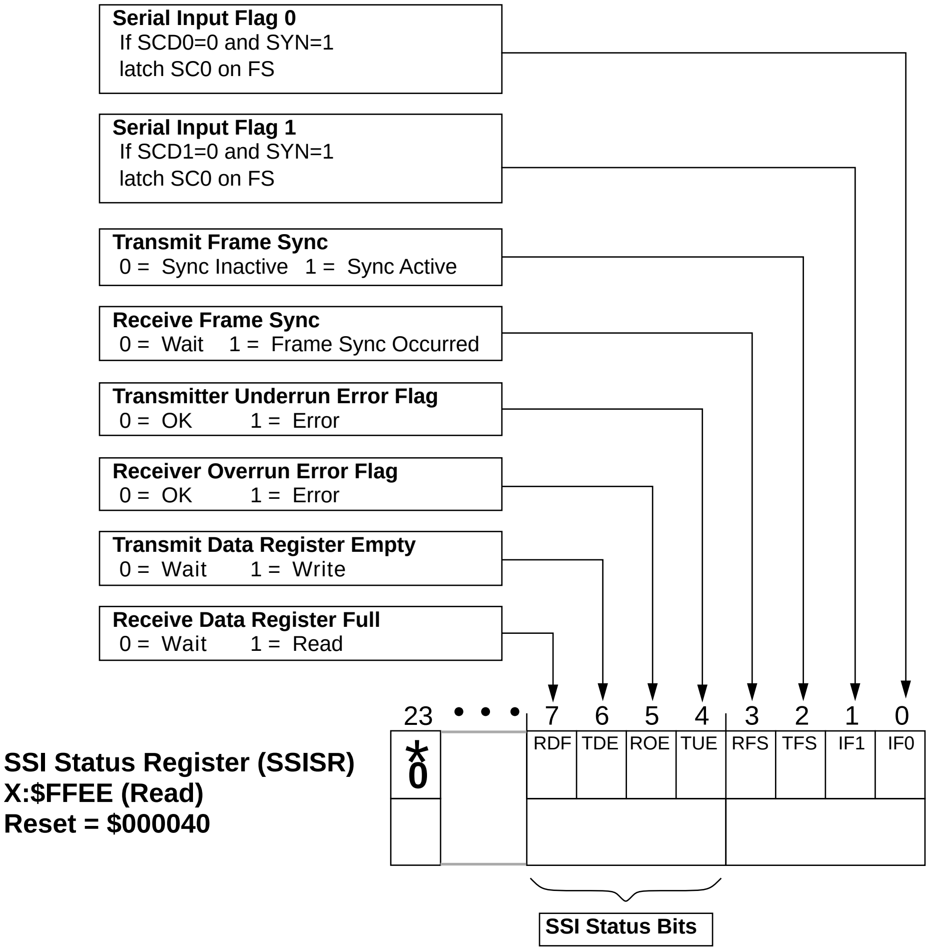
____

Date: ____________________

Programmer: ______________

Sheet 3 of 3

SSI



* = Reserved, Program as zero

Figure B-33 SSI Status Register (SSISR)

Application: ______________________________ Date: ______________

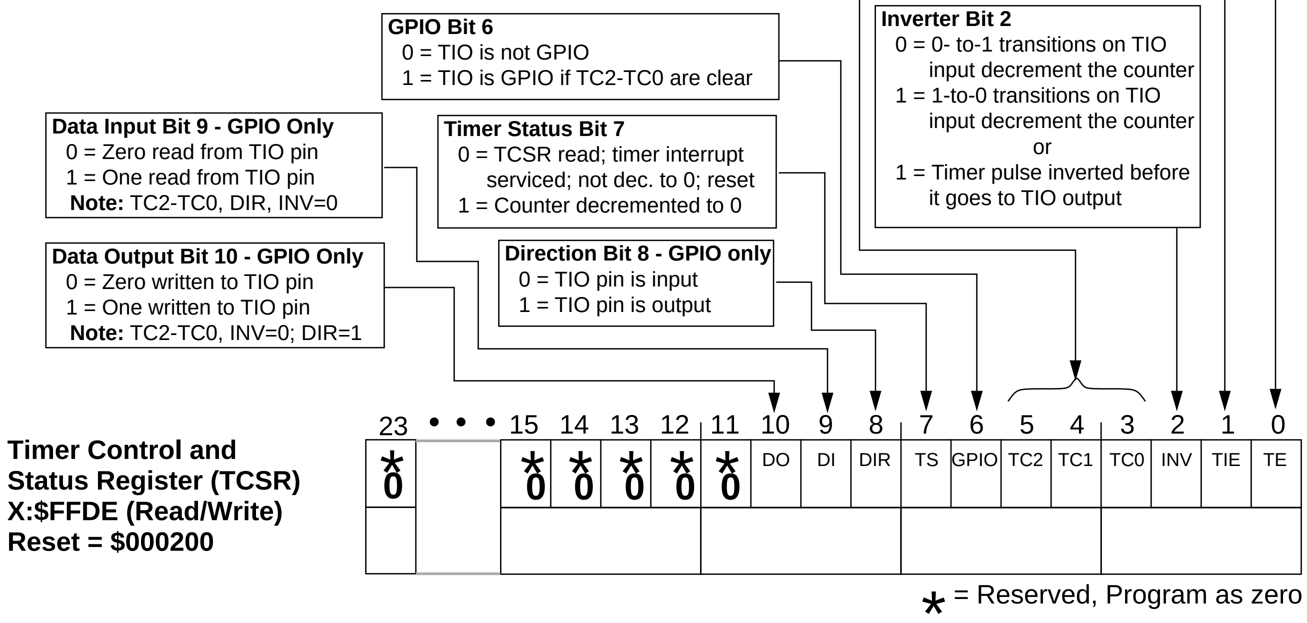
______________________________ ______________

Sheet 1 of 1

TIMER/COUNTER

Timer Control Bits 3-5 (TC0 - TC2)

TC2	TC1	TC0	TIO	Clock	Mode
0	0	0	GPIO	Internal	Timer
0	0	1	Output	Internal	Timer Pulse
0	1	0	Output	Internal	Timer Toggle
0	1	1	X	X	Undefined
1	0	0	Input	Internal	Input Width
1	0	1	Input	Internal	Input Period
1	1	0	Input	External	Standard Time Counter
1	1	1	Input	External	Event Counter



Timer Enable Bit 0
0 = Timer Disabled
1 = Timer Enabled

Timer Interrupt Enable Bit 1
0 = Interrupts Disabled
1 = Interrupts Enabled

Inverter Bit 2
0 = 0- to-1 transitions on TIO input decrement the counter
1 = 1-to-0 transitions on TIO input decrement the counter
or
1 = Timer pulse inverted before it goes to TIO output

GPIO Bit 6
0 = TIO is not GPIO
1 = TIO is GPIO if TC2-TC0 are clear

Timer Status Bit 7
0 = TCSR read; timer interrupt serviced; not dec. to 0; reset
1 = Counter decremented to 0

Direction Bit 8 - GPIO only
0 = TIO pin is input
1 = TIO pin is output

Data Input Bit 9 - GPIO Only
0 = Zero read from TIO pin
1 = One read from TIO pin
Note: TC2-TC0, DIR, INV=0

Data Output Bit 10 - GPIO Only
0 = Zero written to TIO pin
1 = One written to TIO pin
Note: TC2-TC0, INV=0; DIR=1

Timer Control and Status Register (TCSR)
X:$FFDE (Read/Write)
Reset = $000200

* = Reserved, Program as zero

Figure B-34 Timer Control/Status Register (TCSR)

Timer Count Register (TCR)
X:$FFDF (Read/Write)
Unaffected by Reset

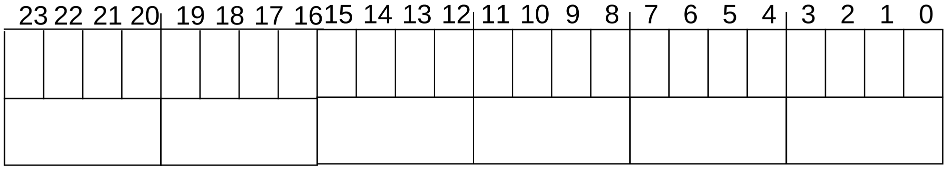


Figure B-35 Timer Count Register (TCR)

PULSE WIDTH MODULATOR

PWMA0 Count Register (PWACR0)
X:$FFDA Read/Write
Unaffected by Reset

Load Pulse Width Here. (bits 23–8)

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			

* = Reserved. Program as zero

Figure B-36 PWMA0 Count Register (PWACR0)

PWMA1 Count Register (PWACR1)
X:$FFDB Read/Write
Unaffected by Reset

Load Pulse Width Here. (bits 23–8)

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			

* = Reserved. Program as zero

Figure B-37 PWMA1 Count Register (PWACR1)

PWMA2 Count Register (PWACR2)
X:$FFDC Read/Write
Unaffected by Reset

Load Pulse Width Here. (bits 23–8)

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			

* = Reserved. Program as zero

Figure B-38 PWMA2 Count Register (PWACR2)

Application: __________ Date: __________

__________ Programmer: __________ Sheet 1 of 6

Application: ____ Date: ____________________

____ Programmer: ______________

Sheet 2 of 6

PULSE WIDTH MODULATOR

WAW2	WAW1	WAW0	Data Width
0	0	0	16
0	0	1	15
0	1	0	14
0	1	1	13
1	0	0	12
1	0	1	11
1	1	0	10
1	1	1	9

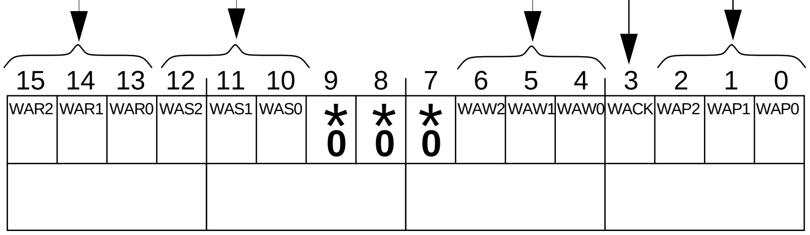
WAP2	WAP1	WAP0	Prescale Factor
0	0	0	2^0
0	0	1	2^1
0	1	0	2^2
0	1	1	2^3
1	0	0	2^4
1	0	1	2^5
1	1	0	2^6
1	1	1	2^7

PWMA Clock Source (WACK)
0 = External 1 = Internal

PWMA Status (WASn) — Read Only
(One bit for each pulse width modulator)
0 = PWACRn Written 1 = PWABUFn Written

PWMA Error (WARn) — Read Only
(One bit for each pulse width modulator)
0 = No Error 1 = Error

PWMA Control/status Register 0 (PWACSR0)
X:$FFD9 Read/Write
Reset = $1C00



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAR2	WAR1	WAR0	WAS2	WAS1	WAS0	* 0	* 0	* 0	WAW2	WAW1	WAW0	WACK	WAP2	WAP1	WAP0

* = Reserved, Program as zero

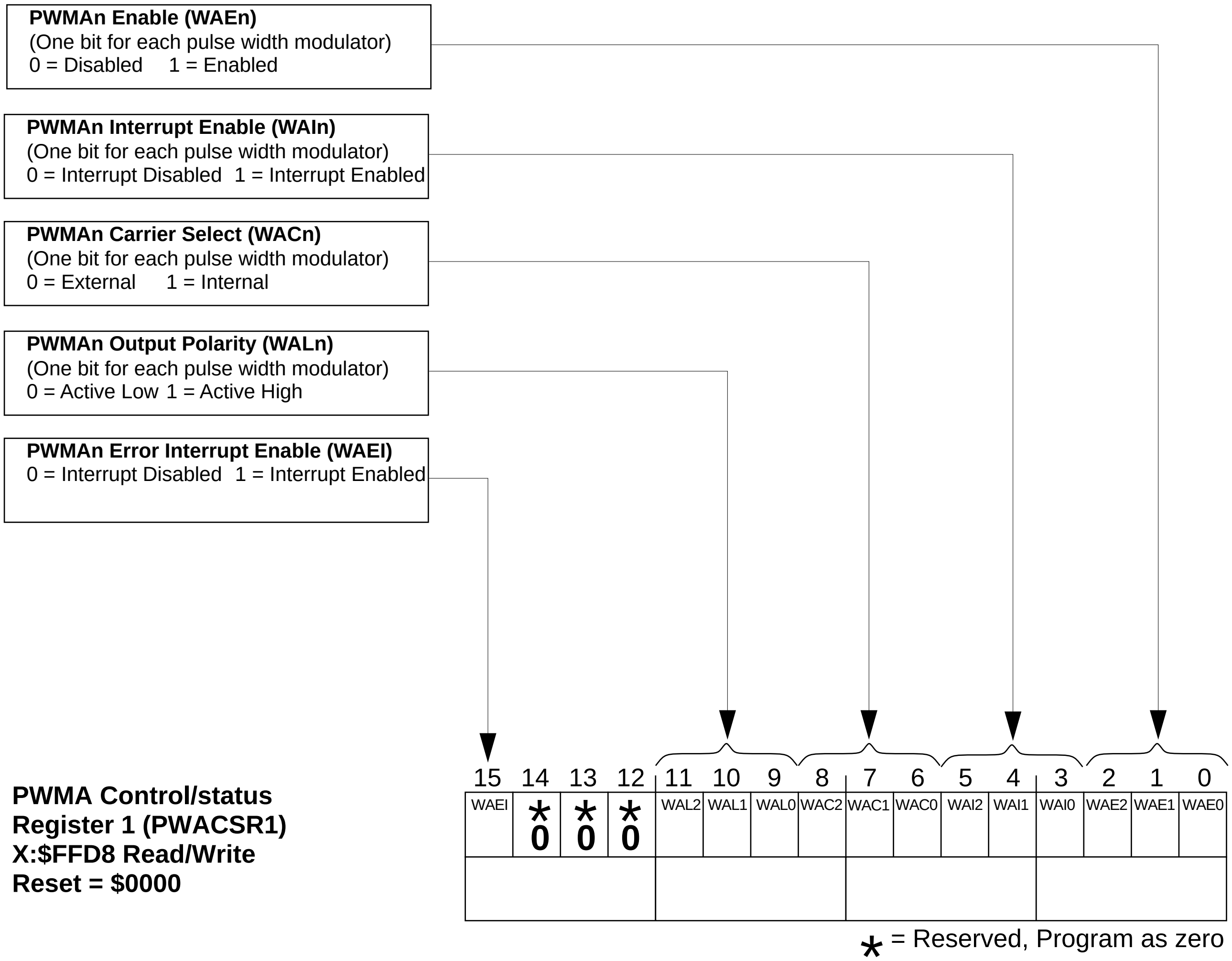
Figure B-39 PWMA Control/status Register 0 (PWACSR0)

Application: ______________________________ Date: ____________

______________________________ Programmer: ____________

Sheet 3 of 6

PULSE WIDTH MODULATOR



PWMAn Enable (WAEn)
(One bit for each pulse width modulator)
0 = Disabled 1 = Enabled

PWMAn Interrupt Enable (WAIn)
(One bit for each pulse width modulator)
0 = Interrupt Disabled 1 = Interrupt Enabled

PWMAn Carrier Select (WACn)
(One bit for each pulse width modulator)
0 = External 1 = Internal

PWMAn Output Polarity (WALn)
(One bit for each pulse width modulator)
0 = Active Low 1 = Active High

PWMAn Error Interrupt Enable (WAEI)
0 = Interrupt Disabled 1 = Interrupt Enabled

PWMA Control/status
Register 1 (PWACSR1)
X:$FFD8 Read/Write
Reset = $0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAEI	* 0	* 0	* 0	WAL2	WAL1	WAL0	WAC2	WAC1	WAC0	WAI2	WAI1	WAI0	WAE2	WAE1	WAE0

* = Reserved, Program as zero

Figure B-40 PWMA Control/status Register 1 (PWACSR1)

PULSE WIDTH MODULATOR

PWMB0 Count Register (PWBCR0) X:$FFD6 Read/Write Unaffected by Reset

Load Pulse Width Here. (bits 23–8)

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			

* = Reserved. Program as zero

Figure B-41 PWMB0 Count Register 0 (PWBCR0)

PWMB1 Count Register (PWBCR1) X:$FFD7 Read/Write Unaffected by Reset

Load Pulse Width Here. (bits 23–8)

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			

* = Reserved. Program as zero

Figure B-42 PWMB1 Count Register 1 (PWBCR1)

Application: ____________ Date: ____________

Programmer: ____________ Sheet 4 of 6

Application: ____ Date: ____________________

____ Programmer: _____________

Sheet 5 of 6

PULSE WIDTH MODULATOR

WBW2	WBW1	WBW0	Data Width
0	0	0	16
0	0	1	15
0	1	0	14
0	1	1	13
1	0	0	12
1	0	1	11
1	1	0	10
1	1	1	9

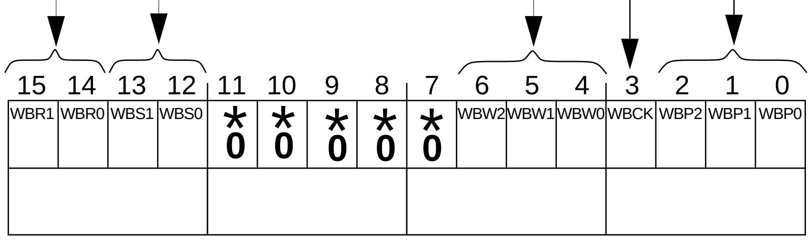
WBP2	WBP1	WBP0	Prescale Factor
0	0	0	2^0
0	0	1	2^1
0	1	0	2^2
0	1	1	2^3
1	0	0	2^4
1	0	1	2^5
1	1	0	2^6
1	1	1	2^7

PWMB Clock Source (WBCK)
0 = External 1 = Internal

PWMA Status (WBSn) — Read Only
(One bit for each pulse width modulator)
0 = PWACRn Written 1 = PWABUFn Written

PWMA Error (WBRn) — Read Only
(One bit for each pulse width modulator)
0 = No Error 1 = Error

PWMB Control/status Register 0 (PWBCSR0)
X:$FFD5 Read/Write
Reset = $0000



* = Reserved, Program as zero

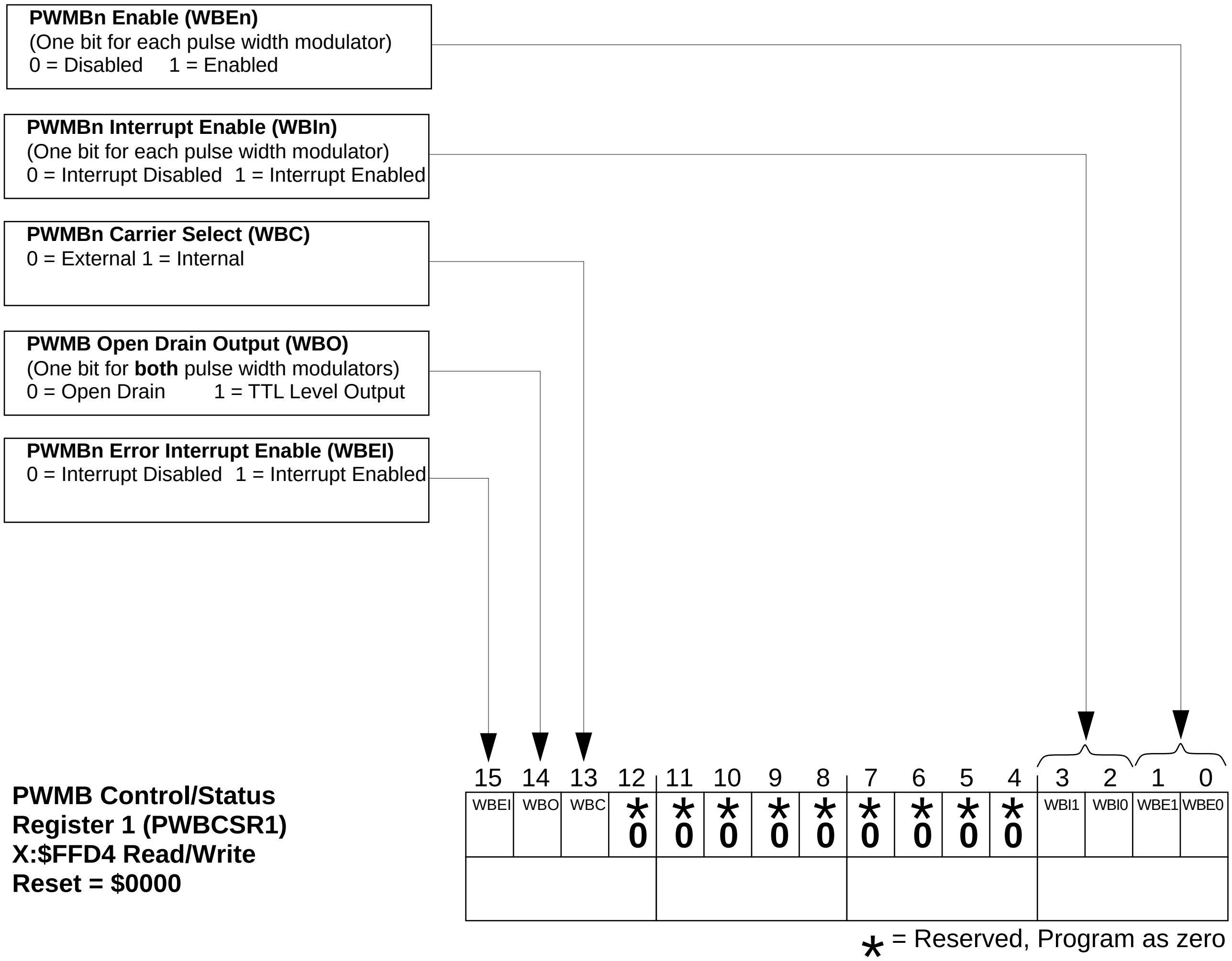
Figure B-43 PWMB Control/status Register 0 (PWBCSR0)

Application: ____ Date: ____________________

____ Programmer: ______________

Sheet 6 of 6

PULSE WIDTH MODULATOR



PWMBn Enable (WBEn)
(One bit for each pulse width modulator)
0 = Disabled 1 = Enabled

PWMBn Interrupt Enable (WBIn)
(One bit for each pulse width modulator)
0 = Interrupt Disabled 1 = Interrupt Enabled

PWMBn Carrier Select (WBC)
0 = External 1 = Internal

PWMB Open Drain Output (WBO)
(One bit for **both** pulse width modulators)
0 = Open Drain 1 = TTL Level Output

PWMBn Error Interrupt Enable (WBEI)
0 = Interrupt Disabled 1 = Interrupt Enabled

PWMB Control/Status Register 1 (PWBCSR1)
X:$FFD4 Read/Write
Reset = $0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WBEI	WBO	WBC	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	WBI1	WBI0	WBE1	WBE0

* = Reserved, Program as zero

Figure B-44 PWMB Control and Status Register 1 (PWBCSR1)

WATCHDOG TIMER

Application: ____________ Date: ____________

Programmer: ____________ Sheet 1 of 2

Watchdog Timer Status
0 = WDG not zero or WCSR was read
1 = WDG decremented to zero

WDG Timer Interrupt Enable
0 = WDG Interrupt Disabled
1 = WDG Interrupt Enabled

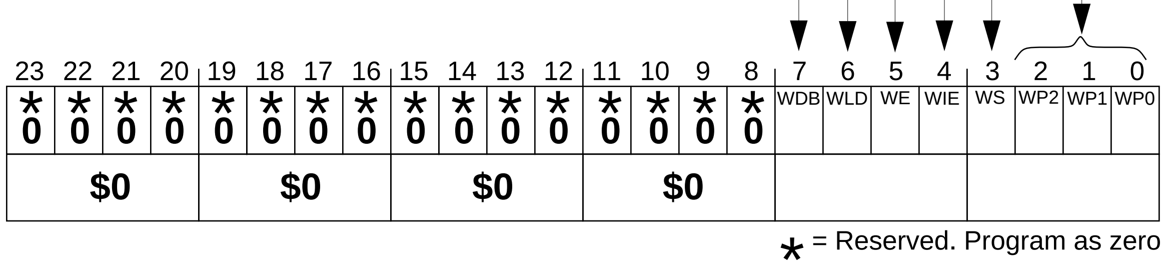
Watchdog Timer Enable
0 = WDG Disabled
1 = WDG Enabled

Watchdog Timer Load
0 = No change
1 = Load Counter and Prescaler

Watchdog Timer Debug
0 = Continue Timing During Debug
1 = Freeze Timing During Debug

WP2	WP1	WP0	Prescale Factor
0	0	0	2^0
0	0	1	2^1
0	1	0	2^2
0	1	1	2^3
1	0	0	2^4
1	0	1	2^5
1	1	0	2^6
1	1	1	2^7

Watchdog Timer Control/ Status Register (WCSR)
X:$FFE6 Read/Write
Reset = $000000



* = Reserved. Program as zero

Figure B-45 Watchdog Timer Control/status Register (WCSR)

WATCHDOG TIMER

Watchdog Timer Count Register (WCR)
X:$FFE7 Read/Write
Unaffected by Reset

Load 16-bit Time Delay Here.

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	WD15	WD14	WD13	WD12	WD11	WD10	WD9	WD8	WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0
$0				$0																			

* = Reserved. Program as zero

Figure B-46 Watchdog Timer Count Register (WCR)

Application: ____________

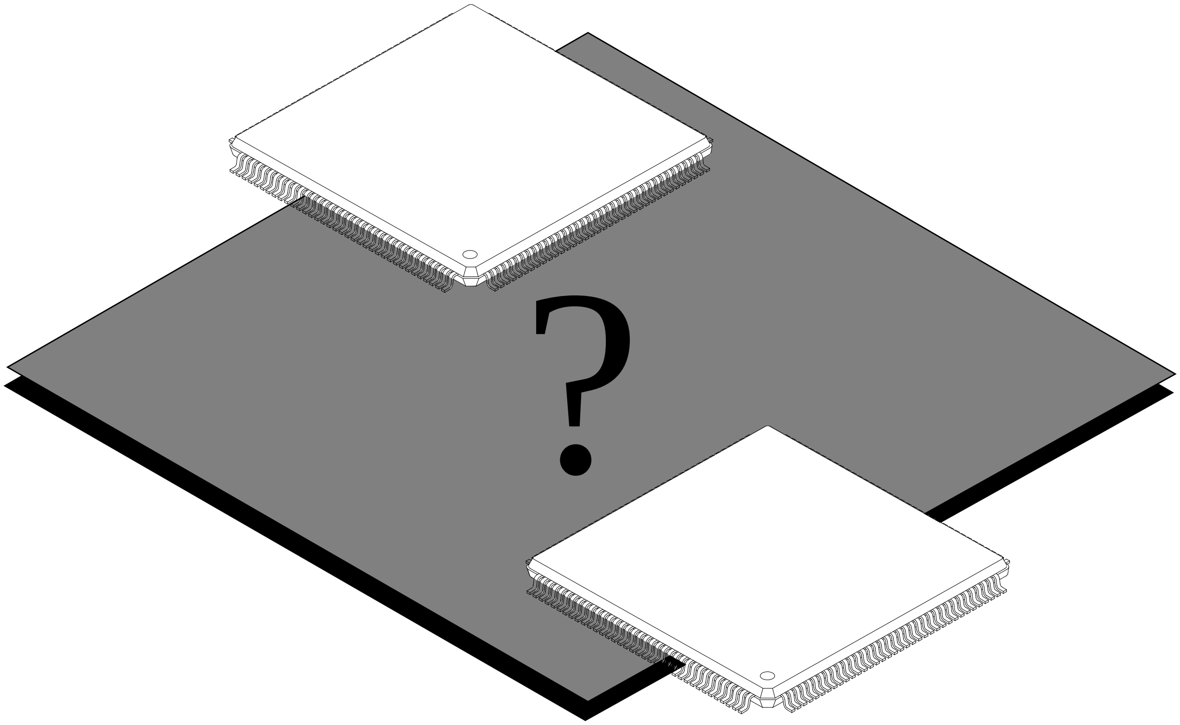
Date: ____________

Programmer: ____________

Sheet 2 of 2

APPENDIX C

DSP56003 AND DSP56005 DIFFERENCES



SECTION CONTENTS

C.1 INTRODUCTION

This manual describes both the DSP56003 and the DSP56005. These DSPs are basically identical; however, the DSP56003 is in a larger package with more pins than the DSP56005 and has several additional signals that are not available on the DSP56005. These additional pins are for bus arbitration, PLL lock, and PLL clock output polarity features. Vertical bars in the margin throughout this manual have been used to flag portions of the text that describe these signals and apply only to the DSP56003. This appendix collects those sections and describes the purpose of each feature.

C.2 DIFFERENCES

The additional DSP56003 features that differentiate it from the DSP56005 are:

- External Memory Bus Arbitration Signals
  - Bus Needed ($\overline{\text{BN}}$)
  - Bus Request ($\overline{\text{BR}}$)
  - Bus Grant ($\overline{\text{BG}}$)
  - Bus Strobe ($\overline{\text{BS}}$)
  - Bus Wait ($\overline{\text{WT}}$)
- PLL Lock Signal
  - Phase and Frequency Locked (PLOCK)
- PLL Clock Output Polarity Signal
  - CKOUT Polarity Control (CKP)

The DSP56003 is available in a 176 pin thin quad flat pack (TQFP), see Table C-1 and Table C-2. The DSP56005 is available in a 144 TQFP, see the *DSP56003/DSP56005 Data Sheet* for additional information.

C.3 SIGNAL DESCRIPTIONS

The pins are organized into the functional groups indicated in Table C-1. Some signals are discussed in the paragraphs that follow.

C.3.1 (2.2.2.1) Bus Needed ($\overline{\text{BN}}$) — active low output — DSP56003 Only

The $\overline{\text{BN}}$ output pin is asserted whenever the chip requires the external memory expansion port (Port A). During instruction cycles where the external bus is not required, $\overline{\text{BN}}$ is deasserted. If an external device has requested the bus by asserting the $\overline{\text{BR}}$ input and the DSP has granted the bus (by asserting $\overline{\text{BG}}$), the DSP will continue processing as long as no external accesses are required. If an external access is required and the chip is not the bus master, it will stop processing and remain in wait states until bus ownership is returned.

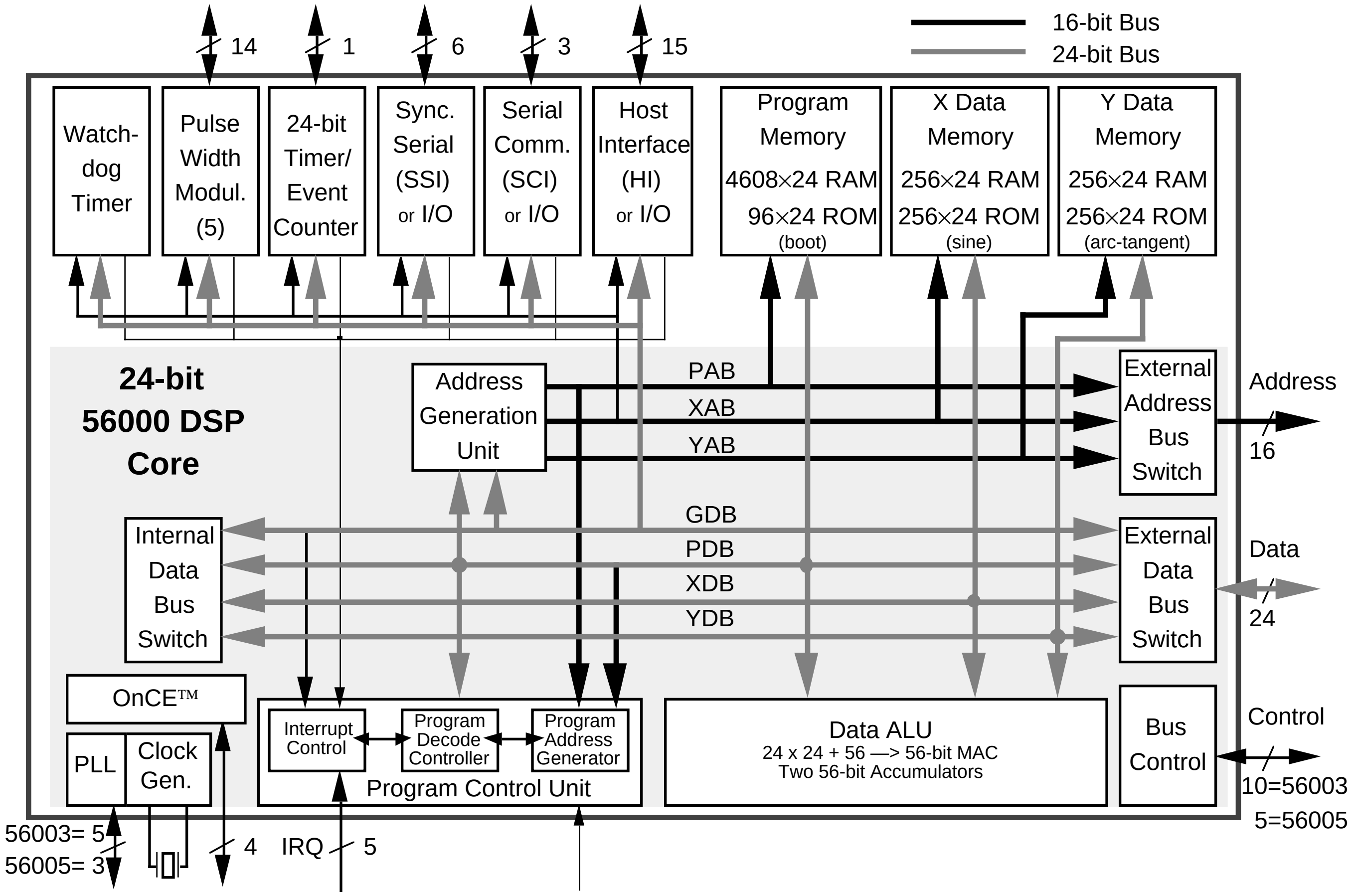

16-bit Bus
24-bit Bus
14
1
6
3
15
Watch-dog Timer
Pulse Width Modul. (5)
24-bit Timer/ Event Counter
Sync. Serial (SSI) or I/O
Serial Comm. (SCI) or I/O
Host Interface (HI) or I/O
Program Memory 4608×24 RAM 96×24 ROM (boot)
X Data Memory 256×24 RAM 256×24 ROM (sine)
Y Data Memory 256×24 RAM 256×24 ROM (arc-tangent)
24-bit 56000 DSP Core
Address Generation Unit
PAB
XAB
YAB
External Address Bus Switch
Address
16
GDB
PDB
XDB
YDB
Internal Data Bus Switch
External Data Bus Switch
Data
24
OnCE™
PLL
Clock Gen.
Interrupt Control
Program Decode Controller
Program Address Generator
Program Control Unit
Data ALU
24 x 24 + 56 —> 56-bit MAC
Two 56-bit Accumulators
Bus Control
Control
10=56003
5=56005
56003= 5
56005= 3
4
IRQ
5

Figure C-1 (1-1) DSP56003/005 Block Diagram

If the $\overline{\text{BN}}$ pin is asserted when the chip is not the bus master, the chip's processing has stopped and the DSP is waiting to acquire bus ownership. An external arbiter may use this pin to help decide when to return bus ownership to the DSP. During hardware reset, $\overline{\text{BN}}$ is deasserted.

Note: The $\overline{\text{BN}}$ pin cannot be used as an early indication of imminent external bus access because it is valid later than the other bus control signal $\overline{\text{BS}}$.

C.3.2 (2.2.2.2) Bus Request ($\overline{\text{BR}}$) — active low input — DSP56003 Only

The bus request $\overline{\text{BR}}$ allows another device such as a processor or DMA controller to become the master of the DSP external data bus D0-D23 and external address bus A0-A15. The DSP asserts $\overline{\text{BG}}$ after the $\overline{\text{BR}}$ input is asserted. The DSP bus controller releases control of the external data bus D0-D23, address bus A0-A15 and bus control pins $\overline{\text{PS}}$, $\overline{\text{DS}}$, X/$\overline{\text{Y}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ at the earliest time possible consistent with proper synchronization after the execution of the current instruction has been completed. These pins are then placed in the high impedance state and the $\overline{\text{BG}}$ pin is asserted. The DSP continues executing instructions only

Table C-1 (2-1) Functional Pin Groupings

Functional Group	DSP56003 Pins	DSP56005 Pins
Address Bus	16	16
Data Bus	24	24
Bus Control	11	6
Host Interface (HI)	15	15
Serial Communications Interface (SCI)	3	3
Synchronous Serial Interface (SSI)	6	6
Timer/Event Counter	1	1
Pulse Width Modulator A (PWMA)	10	10
Pulse Width Modulator B (PWMB)	4	4
On-chip Emulation (OnCE) Port	4	4
Power (V_{CC})	18	17
Ground (GND)	42	26
Interrupt and Mode Control	6	6
Phase-locked Loop (PLL) and Clock	7	5
Reserved	9	1
Total Number of Pins	176	144

if internal program and data memory resources are accessed. If the DSP requests the external bus while $\overline{\mathrm{BR}}$ input pin is asserted, the DSP bus controller inserts wait states until the external bus becomes available ($\overline{\mathrm{BR}}$ and $\overline{\mathrm{BG}}$ deasserted). When $\overline{\mathrm{BR}}$ is deasserted, the DSP will again assume bus mastership. $\overline{\mathrm{BR}}$ is an input during reset.

Notes: **1.** Interrupts are not serviced when a DSP instruction is waiting for the bus controller.

2. $\overline{\mathrm{BR}}$ is prevented from interrupting the execution of a read/modify/write instruction.

3. To prevent erroneous operation, the $\overline{\mathrm{BR}}$ pin should be pulled up when it is not in use.

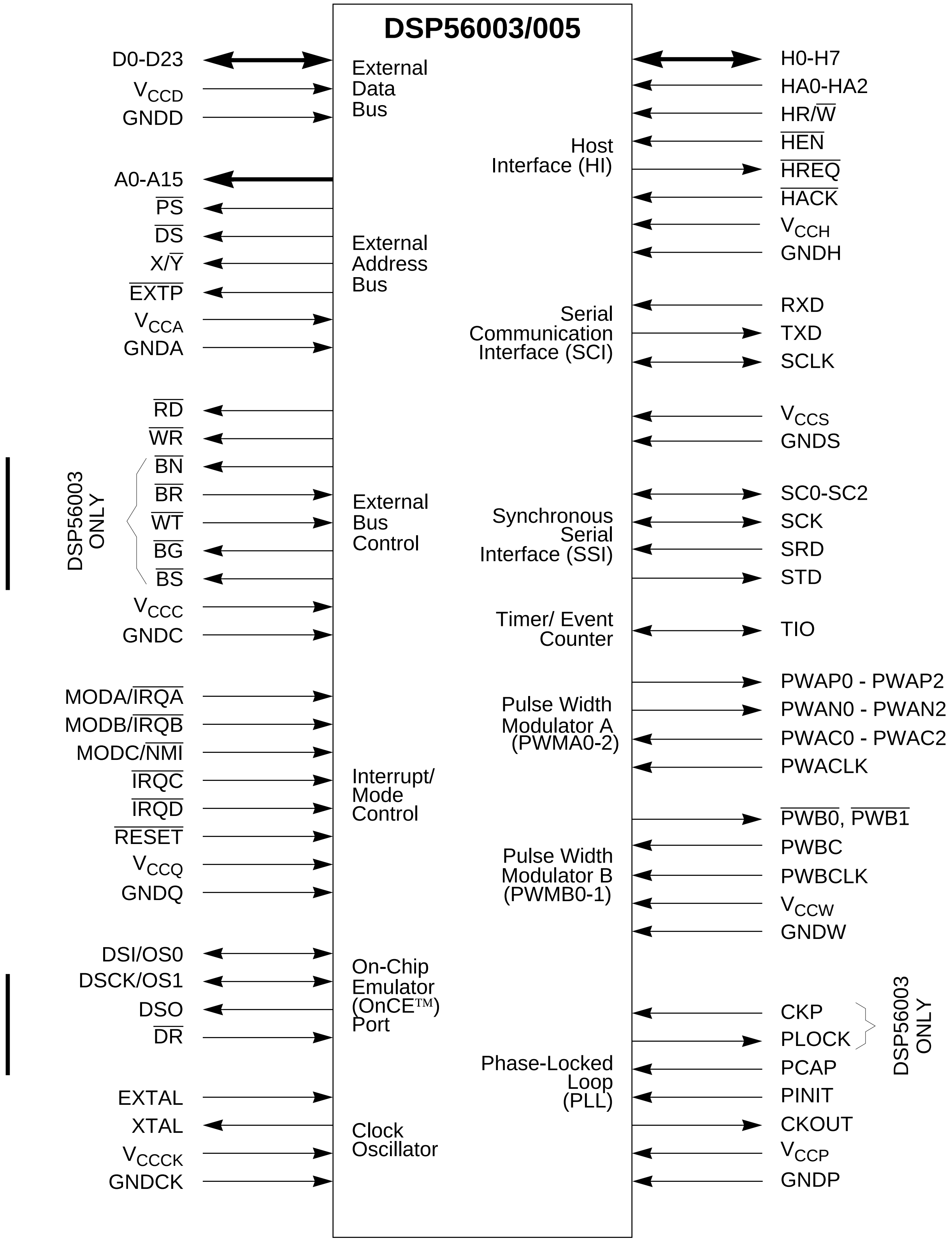


Figure C-2 (2-1) DSP56003/005 Signals

C.3.3 (2.2.2.3) Bus Grant ($\overline{\text{BG}}$) — active low output — DSP56003 Only

This pin is asserted to acknowledge an external bus request. It indicates that the DSP has released control of the external address bus A0-A15, data bus D0-D23 and bus control pins $\overline{\text{PS}}$, $\overline{\text{DS}}$, X/$\overline{\text{Y}}$, $\overline{\text{EXTP}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$. The $\overline{\text{BG}}$ output is asserted in response to a $\overline{\text{BR}}$ input. When the $\overline{\text{BG}}$ output is asserted, the external address bus A0-A15, data bus D0-D23 and bus control pins are in the high impedance state. $\overline{\text{BG}}$ assertion may occur in the middle of an instruction which requires more than one external bus cycle for execution. Note that $\overline{\text{BG}}$ assertion will not occur during indivisible read-modify-write instructions (BSET, BCLR, BCHG). When $\overline{\text{BR}}$ is deasserted, the $\overline{\text{BG}}$ output is deasserted and the DSP regains control of the external address bus, data bus, and bus control pins. This output is deasserted during hardware reset.

C.3.4 (2.2.2.4) Bus Strobe ($\overline{\text{BS}}$) — active low output — DSP56003 Only

Bus Strobe is asserted at the start of a bus cycle and deasserted at the end of the bus cycle. This pin can be used as an “early bus start” signal by an address latch and as an “early

Table C-2 (2-3) Power and Ground Pins

FUNCTION	PIN NAMES		DSP56003		DSP56005	
	V_{CC}	GND	V_{CC}	GND	V_{CC}	GND
Address Bus Output Buffer	V_{CCA}	GNDA	3	5	3	5
Data Bus Output Buffer	V_{CCD}	GNDD	3	6	3	6
Bus Control	V_{CCC}	GNDC	1	1	1	1
Host Interface (HI)	V_{CCH}	GNDH	2	4	2	4
Port C (Serial Communications Interface, Synchronous Serial Interface)	V_{CCS}	GNDS	1	2	1	2
Pulse Width Modulator (PWM)	V_{CCW}	GNDW	1	2	1	2
Internal Logic	V_{CCQ}	GNDQ	5	4	4	4
Phase-locked Loop (PLL)	V_{CCP}	GNDP	1	1	1	1
Clock	V_{CCCK}	GNDCK	1	1	1	1
Thermal	—	GND	0	16	0	0

bus end" signal by an external bus controller. It may also be used with the bus wait input, $\overline{\text{WT}}$, to generate wait states, a feature which provides capabilities such as:

- connecting slower asynchronous devices to the DSP
- allowing devices with differing timing requirements to reside in the same memory space
- allowing a bus arbiter to provide a fast multiprocessor bus access
- providing an alternative to the WAIT and STOP instructions to halt the DSP at a known program location and have a fast restart

This output is deasserted during hardware reset.

C.3.5 (2.2.2.5) Bus Wait ($\overline{\text{WT}}$) — active low input — DSP56003 Only

This input allows an external device to force the DSP to generate wait states for as long as $\overline{\text{WT}}$ is asserted. If $\overline{\text{WT}}$ is asserted while $\overline{\text{BS}}$ is asserted, wait states will be inserted into the current cycle. See the *DSP56003/005 Data Sheet* for timing details.

C.3.6 (2.2.10.2) Thermal Ground (GND) — DSP56003 Only

These pins provide a thermal enhancement (i.e. a heat sink) to the chip. The pins should be directly connected to the ground plane layer to help dissipate heat from the chip. This thermal connection is not necessary for operation. However, it will help keep the chip within the thermal specifications when thermal specification limits are otherwise being approached.

C.3.7 (2.2.11.6) Reset ($\overline{\text{RESET}}$) — input

This input is a direct hardware reset of the processor. When $\overline{\text{RESET}}$ is asserted, the DSP is initialized and placed in the reset state. A Schmitt trigger input is used for noise immunity. When the reset pin is deasserted, the initial chip operating mode is latched from the MODA, MODB, and MODC pins. The chip also samples the PINIT pin and writes its status into the PEN bit of the PLL Control Register. On the DSP56003 *only*, the DSP samples the CKP pin to determine the polarity of the CKOUT signal. When the chip comes out of the reset state, deassertion occurs at a voltage level and is not directly related to the rise time of the $\overline{\text{RESET}}$ signal. However, the probability that noise on $\overline{\text{RESET}}$ will generate multiple resets increases with increasing rise time of the $\overline{\text{RESET}}$ signal.

C.3.8 (2.2.12.2) CKOUT Polarity Control (CKP) — input — DSP56003 Only

This input pin defines the polarity of the CKOUT clock output. Strapping CKP through a resistor to GND will make the CKOUT polarity the same as the EXTAL polarity. Strapping CKP through a resistor to V_{CC} will make the CKOUT polarity the inverse of the EXTAL polarity. The CKOUT clock polarity is internally latched at the end of the hardware reset, so that any changes of the CKP pin logic state after deassertion of hardware reset will not affect the CKOUT clock polarity.

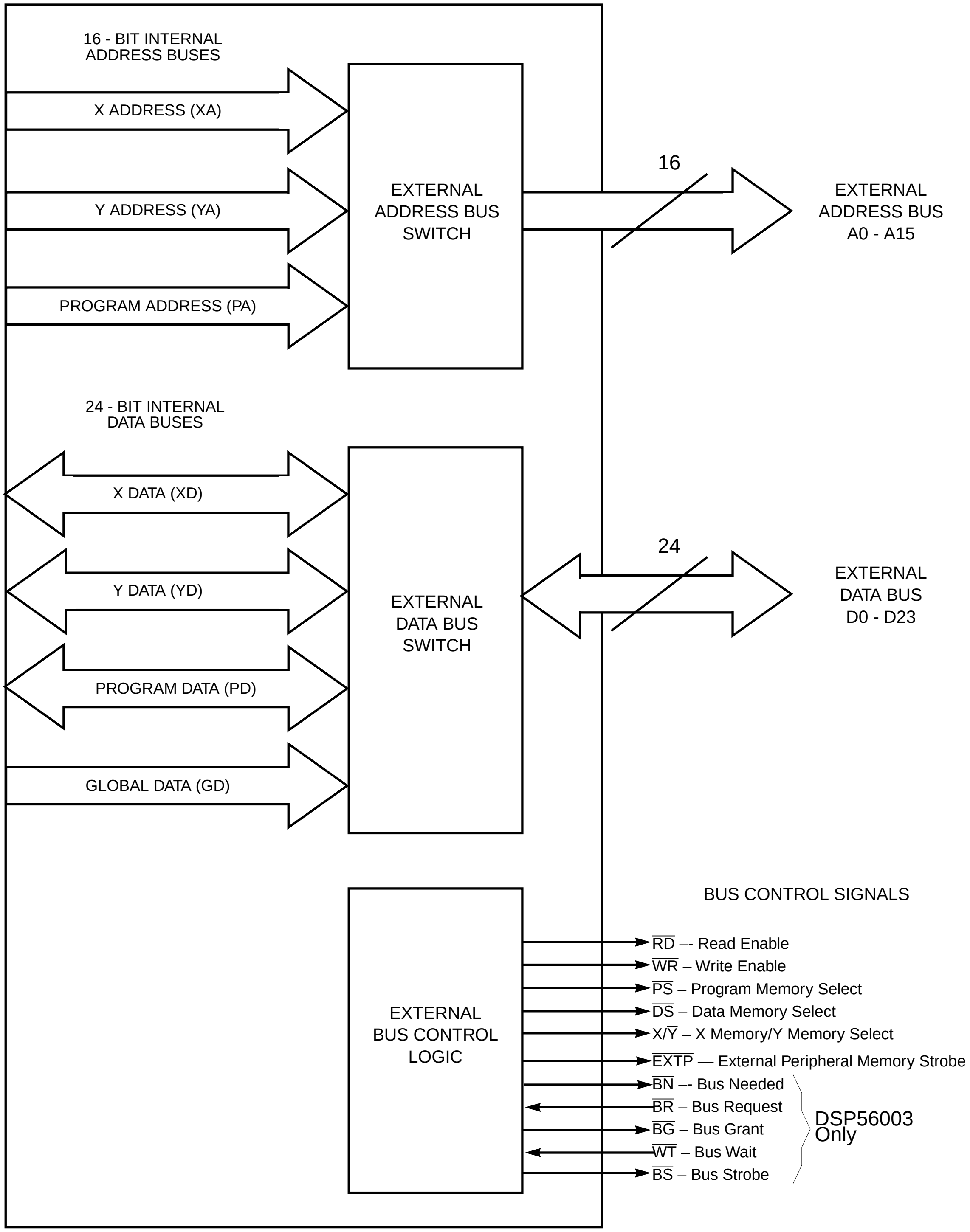

16 - BIT INTERNAL ADDRESS BUSES
X ADDRESS (XA)
Y ADDRESS (YA)
PROGRAM ADDRESS (PA)
EXTERNAL ADDRESS BUS SWITCH
16
EXTERNAL ADDRESS BUS A0 - A15
24 - BIT INTERNAL DATA BUSES
X DATA (XD)
Y DATA (YD)
PROGRAM DATA (PD)
GLOBAL DATA (GD)
EXTERNAL DATA BUS SWITCH
24
EXTERNAL DATA BUS D0 - D23
EXTERNAL BUS CONTROL LOGIC
BUS CONTROL SIGNALS
$\overline{RD}$ –- Read Enable
$\overline{WR}$ – Write Enable
$\overline{PS}$ – Program Memory Select
$\overline{DS}$ – Data Memory Select
X/$\overline{Y}$ – X Memory/Y Memory Select
$\overline{EXTP}$ — External Peripheral Memory Strobe
$\overline{BN}$ –- Bus Needed
$\overline{BR}$ – Bus Request
$\overline{BG}$ – Bus Grant
$\overline{WT}$ – Bus Wait
$\overline{BS}$ – Bus Strobe
DSP56003 Only

Figure C-3 (4-1) Port A Signals

C.3.9 (2.2.12.7) Phase and Frequency Locked (PLOCK) — output — DSP56003 Only

This signal originates from the PLL phase detector. The chip asserts PLOCK when the PLL is enabled and has locked on the proper phase and frequency of EXTAL. PLOCK is deasserted by the chip if the PLL is enabled and has not locked on the proper phase and frequency. The processor is halted when PLOCK is deasserted. PLOCK is asserted if the PLL is disabled. This signal is a reliable indicator of the PLL lock state only after the chip has exited the hardware reset state. During hardware reset, the PLOCK state is determined by PINIT and by the PLL lock condition.

C.4 APPLICATIONS OF THE EXTRA PINS

The external memory bus arbitration signals are used to allow multiple devices to use the external memory bus without bus arbitration conflicts.

C.4.1 Bus Control

The $\overline{\text{BN}}$ signal allows the DSP to tell an external device that the DSP needs access to the external bus. When the DSP gains access, the $\overline{\text{BS}}$ signal tells external devices that the DSP is either about to use the bus or that it is using the bus. The (essentially) equivalent signals from the external viewpoint are $\overline{\text{BR}}$ and $\overline{\text{BG}}$. $\overline{\text{BR}}$ is used by an external device to tell the DSP that the external device needs the bus. The $\overline{\text{BG}}$ signal tells the external device that the DSP has relinquished the bus and will wait to use the bus until after $\overline{\text{BR}}$ becomes inactive.

These four signals are useful in constructing:

- multiple DSP arrays
- mixed arrays of DSPs and other processors
- shared memory systems using single port memory
- external memory mapped peripherals

C.4.2 External Memory Interface Wait States

The DSP56003/005 features two methods to allow the user to accommodate slow memory and slow peripherals by changing the port A bus timing. The first method uses the bus control register (BCR), see Table C-3, which allows a fixed number of wait states to be inserted in a given memory access to all locations in each of the four memory spaces: X, Y, P, and I/O. The second method uses the bus strobe ($\overline{\text{BS}}$) and bus wait ($\overline{\text{WT}}$) facility (DSP56003 only), which allows an external device to insert an arbitrary number of wait states (see Table C-3) when accessing either a single location or multiple locations of external memory or I/O space. Wait states are executed until the external device releases the DSP to finish the external memory cycle.

Table C-3 (4-2) Wait State Control

BCR Contents	$\overline{\text{WT}}$ (DSP56003 only)	Number of Wait States Generated
0	Deasserted	0
0	Asserted — DSP56003 only	2 (minimum)
> 0	Deasserted	Equals value in BCR
> 0	Asserted — DSP56003 only	Minimum equals 2 or value in BCR. Maximum is determined by BCR or $\overline{\text{WT}}$, whichever is larger.

C.4.3 PLL and Clock Signal Applications

The PLL Locked signal indicates that the PLL is in phase and on frequency (PLOCK = 1) with the signal on EXTAL or that the PLL is adjusting its frequency (PLOCK = 0). If the PLL multiplier register (MF-MF11) has been changed, PLOCK will be deasserted (PLOCK = 0) and the clock will be cut off from the core processor until PLOCK = 1. This provides an external indicator that the multiplier was written and that the DSP core has paused until the PLL is locked.

The CKOUT Polarity Control allows the user to invert the clock out of the DSP without skewing it by the delay time of an inverter. The delay of an inverter can become critical when using fast static RAMs with access times of a few nano-seconds.

C.5 (4.6) BUS STROBE AND WAIT PINS — DSP56003 Only

The ability to insert wait states using $\overline{\text{BS}}$ and $\overline{\text{WT}}$ allows devices with differing timing requirements to reside in the same memory space, allows a bus arbiter to provide a fast multiprocessor bus access, and provides another means of halting the DSP at a known program location with a fast restart.

The timing of the $\overline{\text{BS}}$ and $\overline{\text{WT}}$ pins is illustrated in Figure C-4. $\overline{\text{BS}}$ is asserted at the same time as the external address lines. $\overline{\text{BS}}$ can be used by external wait-state logic to establish the start of an external access. $\overline{\text{BS}}$ is deasserted in T3 of each external bus cycle, signaling that the current bus cycle will complete. Since the $\overline{\text{WT}}$ signal is internally synchronized, it can be asserted asynchronously with respect to the system clock. The $\overline{\text{WT}}$ signal should only be asserted while $\overline{\text{BS}}$ is asserted. Asserting $\overline{\text{WT}}$ while $\overline{\text{BS}}$ is deasserted will give indeterminate results. However, for the number of inserted wait states to be deterministic, $\overline{\text{WT}}$ timing must satisfy setup and hold timing with respect to the

OPERATING MODE REGISTER

7	6	5	4	3	2	1	0
EM	SD	0	0	0	DE	MB	MA

SET EM = 1

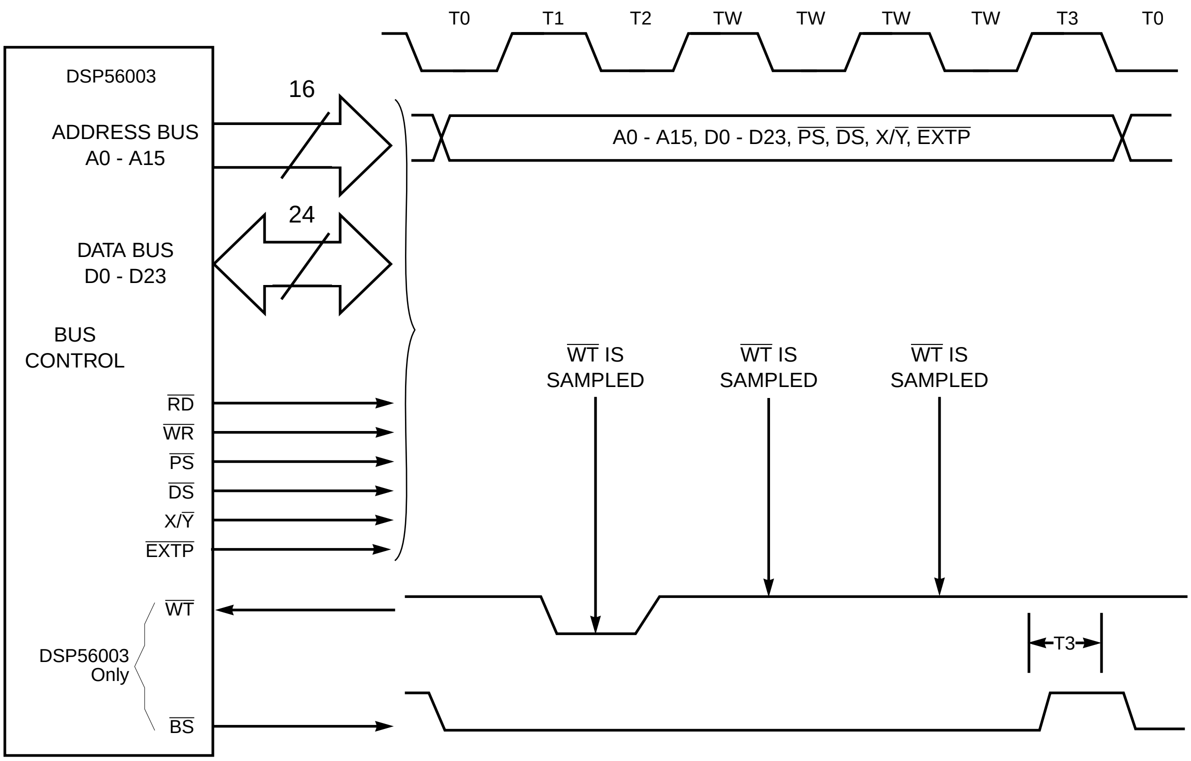


Figure C-4 (4-10) Bus Strobe/Wait Sequence — DSP56003 Only

negative-going edge of EXTAL. The setup and hold times are provided in the *DSP56003/005 Data Sheet*. The timing of $\overline{\text{WR}}$ is controlled by the BCR and is independent of $\overline{\text{WT}}$. The minimum number of wait states that can be inserted using the $\overline{\text{WT}}$ pin is two. The BCR is still operative when using $\overline{\text{BS}}$ and $\overline{\text{WT}}$ and defines the minimum number of wait states that are inserted. Table C-3 summarizes the effect of the BCR and $\overline{\text{WT}}$ pin on the number of wait states generated.

C.6 (4.7) BUS ARBITRATION AND SHARED MEMORY — DSP56003 Only

The DSP56003 has five pins that control the external memory interface. They are bus needed ($\overline{\text{BN}}$), bus request ($\overline{\text{BR}}$), bus grant ($\overline{\text{BG}}$), bus strobe ($\overline{\text{BS}}$) and bus wait ($\overline{\text{WT}}$) and they are described in Section 2 — DSP56003/005 Pin Descriptions.

The bus control signals provide the means to connect additional bus masters (which may be additional DSPs, microprocessors, direct memory access (DMA) controllers, etc.) to the external memory interface bus. They work together to arbitrate and determine what device gets access to the bus.

If an external device has requested the external bus by asserting the $\overline{\text{BR}}$ input, and the DSP has granted the bus by asserting $\overline{\text{BG}}$, the DSP will continue to process as long as it requires no external bus accesses itself. If the DSP **does** require an external access but is not the bus master, it will stop processing and remain in wait states until it regains bus ownership. The $\overline{\text{BN}}$ pin will be asserted, and an external device may use $\overline{\text{BN}}$ to help “arbitrate”, or decide when to return bus ownership to the chip.

- Four examples of bus arbitration will be described later in this section:
- bus arbitration using only $\overline{\text{BR}}$ and $\overline{\text{BG}}$ with internal control
- bus arbitration using $\overline{\text{BN}}$, $\overline{\text{BR}}$, and $\overline{\text{BG}}$ with external control
- bus arbitration using $\overline{\text{BR}}$, $\overline{\text{BG}}$ and $\overline{\text{WT}}$, $\overline{\text{BS}}$ with no overhead
- signaling using semaphores.

The $\overline{\text{BR}}$ input allows an external device to request and be given control of the external bus while the DSP continues internal operations using internal memory spaces. This independent operation allows a bus controller to arbitrate a multiple bus-master system independent of operation of each DSP. (A bus master can issue addresses on the bus; a bus slave can respond to addresses on the bus. A single device can be both a master and a slave, but can only be one or the other at any given time.)

Before $\overline{\text{BR}}$ is asserted, all the external memory interface signals may be driven by the DSP. When $\overline{\text{BR}}$ is asserted (see Figure C-5), the DSP will assert $\overline{\text{BG}}$ after the current external access cycle completes and will simultaneously three-state (high-impedance) the external memory interface signals (see the *DSP56003/005 Data Sheet* for exact timing of $\overline{\text{BR}}$ and $\overline{\text{BG}}$). The bus is then available to whatever external device has bus mastership. The external device will return bus mastership to the DSP by deasserting $\overline{\text{BR}}$. After the DSP completes the current cycle (an internally executed instruction with or without wait states), $\overline{\text{BG}}$ will be deasserted. When $\overline{\text{BG}}$ is deasserted, the A0-A15, $\overline{\text{PS}}$, $\overline{\text{DS}}$, X/$\overline{\text{Y}}$, $\overline{\text{EXTP}}$, and $\overline{\text{RD}}$, $\overline{\text{WR}}$ lines will be driven. However, the data lines will remain in three-state. All signals are now ready for a normal external access.

During the wait state (see SECTION 7 in the *DSP56000 Family Manual*), the $\overline{\text{BR}}$ and $\overline{\text{BG}}$ circuits remain active. However, the port is inactive - the control signals are deasserted, the data signals are inputs, and the address signals remain as the last address read or written. When $\overline{\text{BR}}$ is asserted, all signals are three-stated (high impedance). Table C-4 shows the status of $\overline{\text{BR}}$ and $\overline{\text{BG}}$ during the wait state.

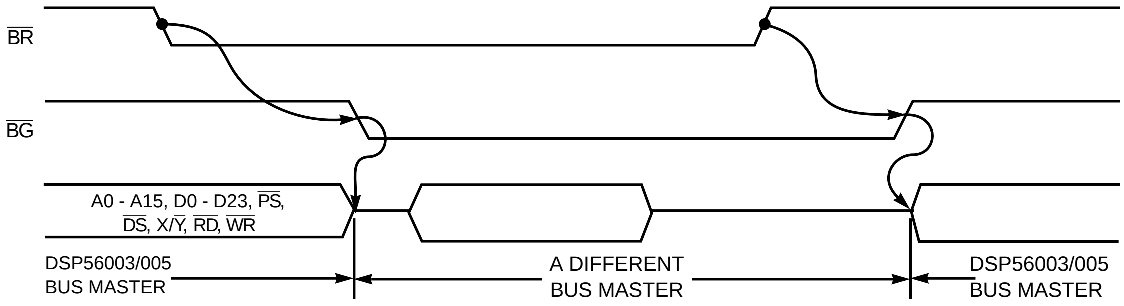

$\overline{\text{BR}}$
$\overline{\text{BG}}$
A0 - A15, D0 - D23, $\overline{\text{PS}}$,
$\overline{\text{DS}}$, X/$\overline{\text{Y}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$
DSP56003/005
BUS MASTER
A DIFFERENT
BUS MASTER
DSP56003/005
BUS MASTER

Figure C-5 (4-11) Bus Request/Bus Grant Sequence — DSP56003 Only

C.6.1 (4.7.1) Bus Arbitration Using Only $\overline{\text{BR}}$ and $\overline{\text{BG}}$ With Internal Control — DSP56003 Only

Perhaps the simplest example of a shared memory system using a DSP56003 is shown in Figure C-6. The bus arbitration is performed within the DSP#2 by using software. DSP#2 controls all bus operations by using I/O pin OUT2 to three-state its own external memory interface and by never accessing the external memory interface without first calling the subroutine that arbitrates the bus. When the DSP#2 needs to use external memory, it uses I/O pin OUT1 to request bus access and I/O pin IN1 to read bus grant. DSP#1 does not need any extra code for bus arbitration since the $\overline{\text{BR}}$ and $\overline{\text{BG}}$ hardware handles its bus arbitration automatically. The protocol for bus arbitration is as follows:

At reset: DSP#2 sets OUT2=0 ($\overline{\text{BR}}$#2=0) and OUT1=1 ($\overline{\text{BR}}$#1=1), which gives DSP#1 access to the bus and suspends DSP#2 bus access.

Table C-4 (4-3) BR and BG During Wait — DSP56003 Only

Signal	Before $\overline{\text{BR}}$	While $\overline{\text{BG}}$	After $\overline{\text{BR}}$	After Return to Normal State	After First

When DSP#2 wants control of the memory, the following steps are performed (see Figure C-7):

1. DSP# 2 sets OUT1=0 ($\overline{\text{BR}}$#1=0).
2. DSP# 2 waits for IN1=0 ($\overline{\text{BG}}$#1=0 and DSP#1 off the bus).
3. DSP#2 sets OUT2=1 ($\overline{\text{BR}}$#2=1 to let DSP#2 control the bus).
4. DSP#2 accesses the bus for block transfers, etc. at full speed.
5. To release the bus, DSP#2 sets OUT2=0 ($\overline{\text{BR}}$#2=0) after the last external access.
6. DSP#2 then sets OUT1=1 ($\overline{\text{BR}}$#1=1) to return control of the bus to DSP#1.
7. DSP#1 then acknowledges mastership by deasserting $\overline{\text{BG}}$#1.

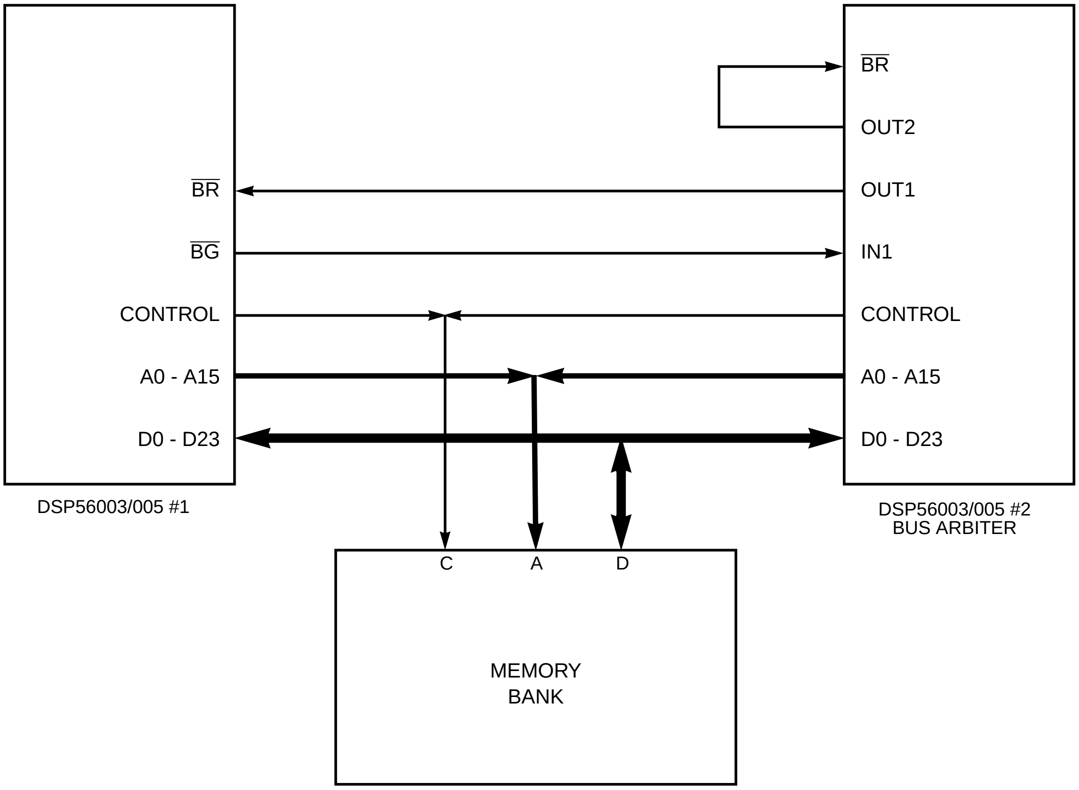


Figure C-6 (4-12) Bus Arbitration Using Only BR and BG with Internal Control — DSP56003 Only

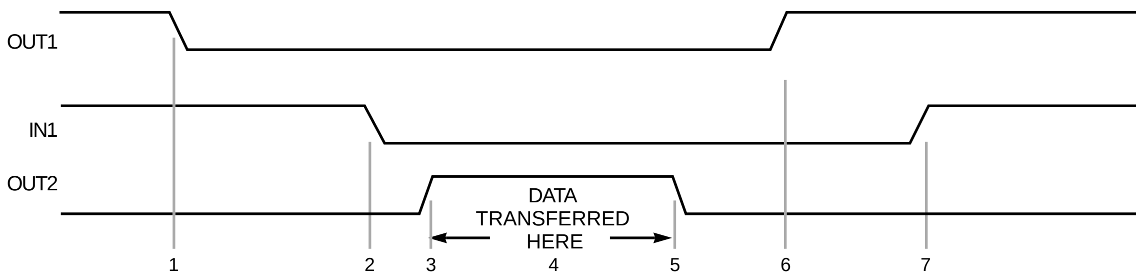


Figure C-7 (4-13) Two DSPs with External Bus Arbitration Timing

C.6.2 (4.7.2) Bus Arbitration Using $\overline{\text{BN}}$, $\overline{\text{BR}}$, and $\overline{\text{BG}}$ With External Control — DSP56003 Only

The system shown in Figure C-8 can be implemented with external bus arbitration logic, which will save processing capacity on the DSPs and can make bus access much faster at a cost of additional hardware. The bus arbitration logic takes control of the external bus by deasserting an enable signal (E1, E2, and E3) to all DSPs, which will then acknowledge by granting the bus ($\overline{\text{BG}}$=0). When a DSP (DSP#1 in Figure C-8) needs the bus, it will enter the wait state with $\overline{\text{BN}}$ asserted. If DSP#1 has highest priority of the pending bus requests, the arbitration logic grants the bus to DSP#1 by asserting E1 (E2 for DSP#2; E3 for DSP#3) to let the DSP know that it can have the bus. DSP#1 will then deassert $\overline{\text{BG}}$ to tell the arbiter it has taken control of the bus. When the DSP no longer needs to make an external access it will deassert $\overline{\text{BN}}$ and the arbiter deasserts E1, after which the DSP deasserts $\overline{\text{BG}}$.

C.6.3 (4.7.3) Arbitration Using $\overline{\text{BR}}$ and $\overline{\text{BG}}$, and $\overline{\text{WT}}$ and $\overline{\text{BS}}$ With No Overhead — DSP56003 Only

By using the circuit shown in Figure C-9, two DSPs can share memory with hardware arbitration that requires no software on the part of the DSPs. The protocol for bus arbitration in Figure C-9 is as follows:

At RESET assume DSP#1 is not making external accesses so that $\overline{\text{BR}}$ of DSP#2 is deasserted. Hence, $\overline{\text{BG}}$ of DSP#2 is deasserted, which three-states the buffers, giving DSP#2 control of the memory.

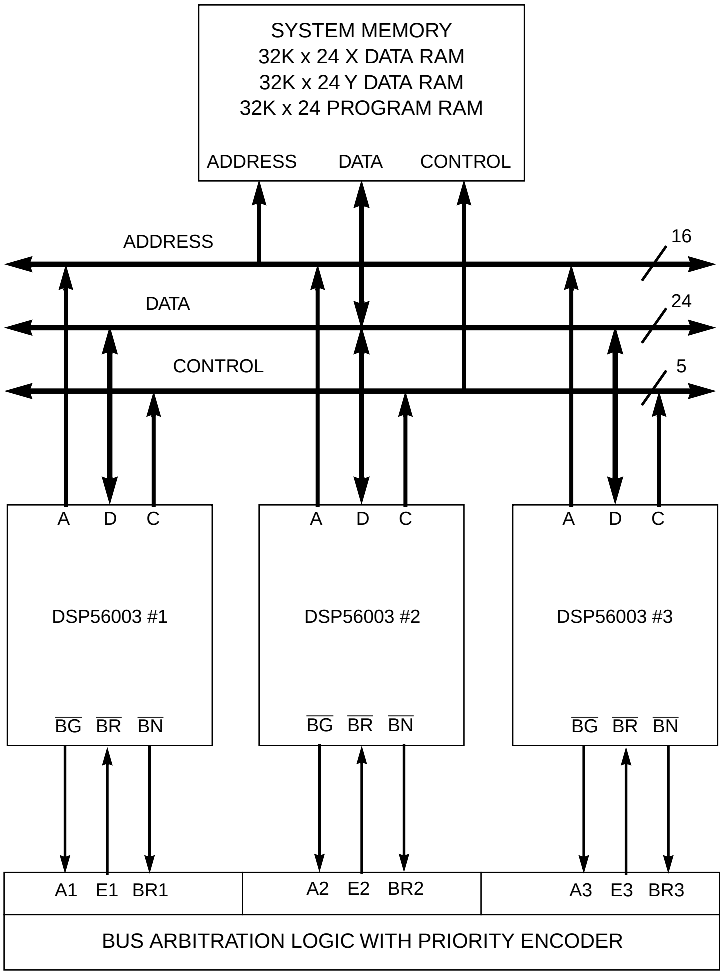

SYSTEM MEMORY
32K x 24 X DATA RAM
32K x 24 Y DATA RAM
32K x 24 PROGRAM RAM
ADDRESS
DATA
CONTROL
ADDRESS
16
DATA
24
CONTROL
5
A
D
C
DSP56003 #1
$\overline{BG}$ $\overline{BR}$ $\overline{BN}$
A
D
C
DSP56003 #2
$\overline{BG}$ $\overline{BR}$ $\overline{BN}$
A
D
C
DSP56003 #3
$\overline{BG}$ $\overline{BR}$ $\overline{BN}$
A1 E1 BR1
A2 E2 BR2
A3 E3 BR3
BUS ARBITRATION LOGIC WITH PRIORITY ENCODER

Figure C-8 (4-14) Bus Arbitration Using BN, BR, and BG with External Control — DSP56003 Only

When DSP#1 wants control of the memory the following steps are performed (see Figure C-10):

1. DSP#1 makes an external access, thereby asserting $\overline{\text{BS}}$, which asserts $\overline{\text{WT}}$ (causing DSP#1 to execute wait states in the current cycle) and asserts DSP#2 $\overline{\text{BR}}$ (requesting that DSP#2 release the bus).

2. When DSP#2 finishes its present bus cycle, it three-states its bus drivers and asserts $\overline{\text{BG}}$. Asserting $\overline{\text{BG}}$ enables the three-state buffers, placing the DSP#1 signals on the memory bus. Asserting $\overline{\text{BG}}$ also deasserts $\overline{\text{WT}}$, which allows DSP#1 to finish its bus cycle.

3. When DSP#1's memory cycle is complete, it releases $\overline{\text{BS}}$, which deasserts $\overline{\text{BR}}$. DSP#2 then deasserts $\overline{\text{BG}}$, three-stating the buffers and allowing DSP#2 to access the memory bus.

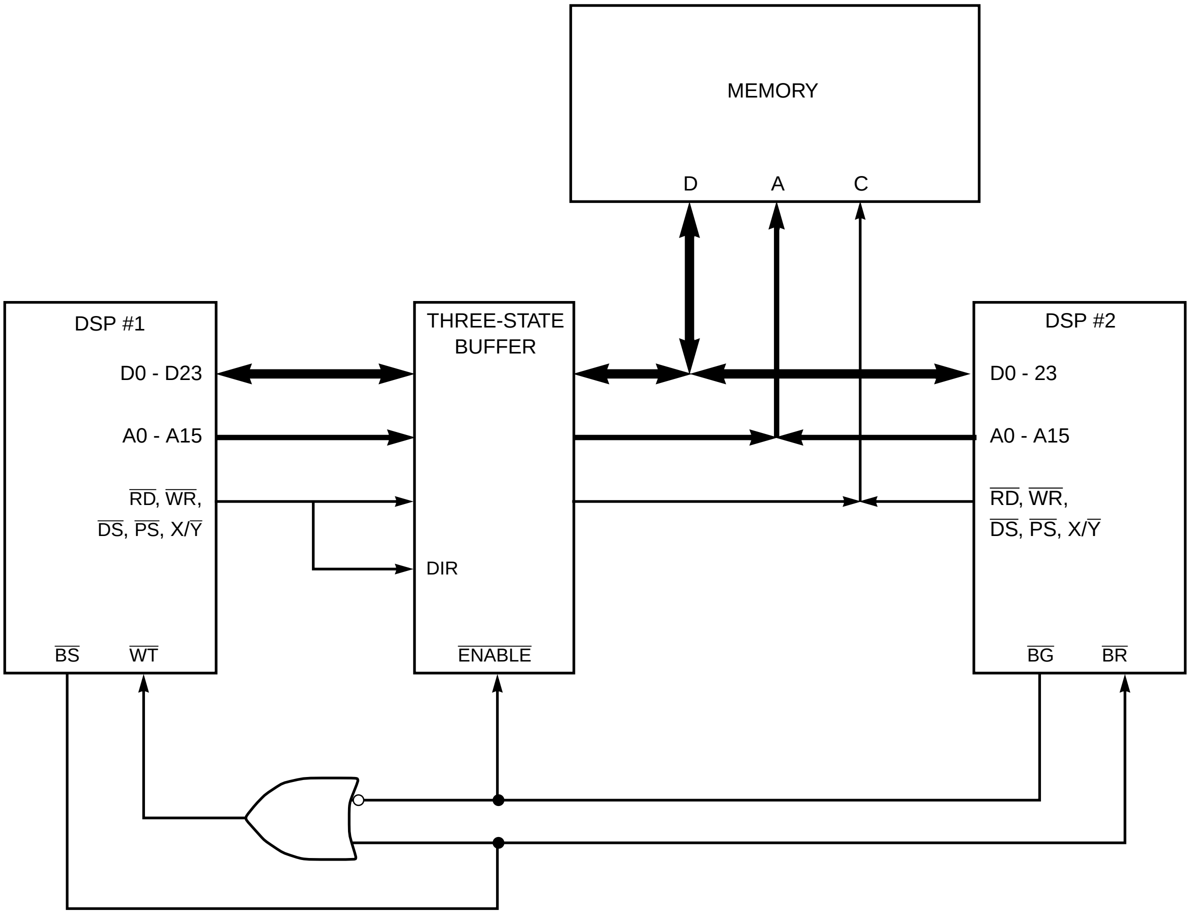


Figure C-9 (4-15) Bus Arbitration Using BR and BG, and WT and BS with No Overhead — DSP56003 Only

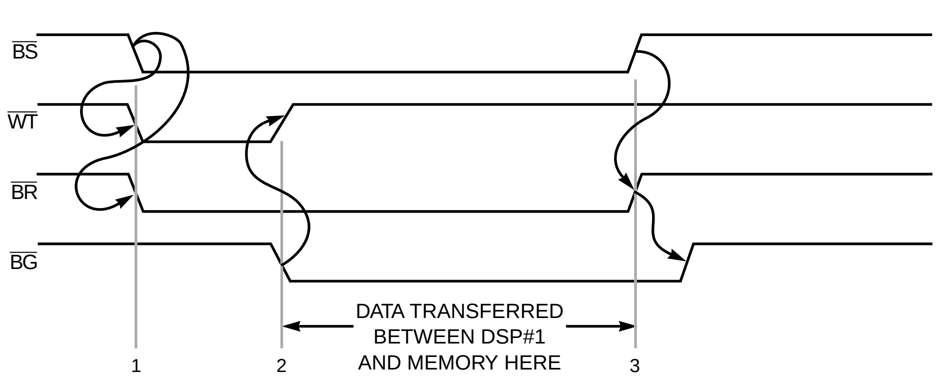


Figure C-10 (4-16) Two DSPs with External Bus Arbitration Timing — DSP56003 Only

Freescale Semiconductor, Inc.

Order this document by
DSP56003UMAD/AD

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

DSP56003/005

Addendum to
24-bit Digital Signal Processor User’s Manual

This document, containing changes, additional features, further explanations, and clarifications, is a supplement to the original document:

DSP56003UM/AD	**User’s Manual**	**DSP56003/005** **24-bit Digital Signal Processor**

Change the following:

Page 5-19, Figure 5-11 - Replace “X:FFE” in two places with “X:$FFE8” on top and “X:FFE9” on bottom.

Page 6-26, Program listing - Move: “MOVE (R0)+ ;and increment the packing pointer” to after the JCS instruction.

```
Replace  “RTI”
with     “RTI    X:”

Replace  “FLAG     MOVE     A,(R3)+”
with     “FLAG     MOVE     A,X:(R3)+”
```

Page 6-66, Section 6.3.9, third sentence - Replace “Bits CD11–CD0, SCP, and STIR in the SCCR work together to determine the time base.” with “Bits CD11–CD0 and SCP in the SCCR and the STIR bit in the SCR work together to determine the time base.”

Page 7-159, Section 7.3.7.2, second paragraph - Replace “MC15500” with “MC145500”.

Page 7-62, Figure 7-38 - Replace “MC1550x” with “MC14550x”.

Page 7-87, Figure 7-54 - Replace “MC15500” with “MC145500”.

Page B-26, Figure B-32 - Change CRB bits 2-4 description (see Figure B-32 below).

MOTOROLA

SSI

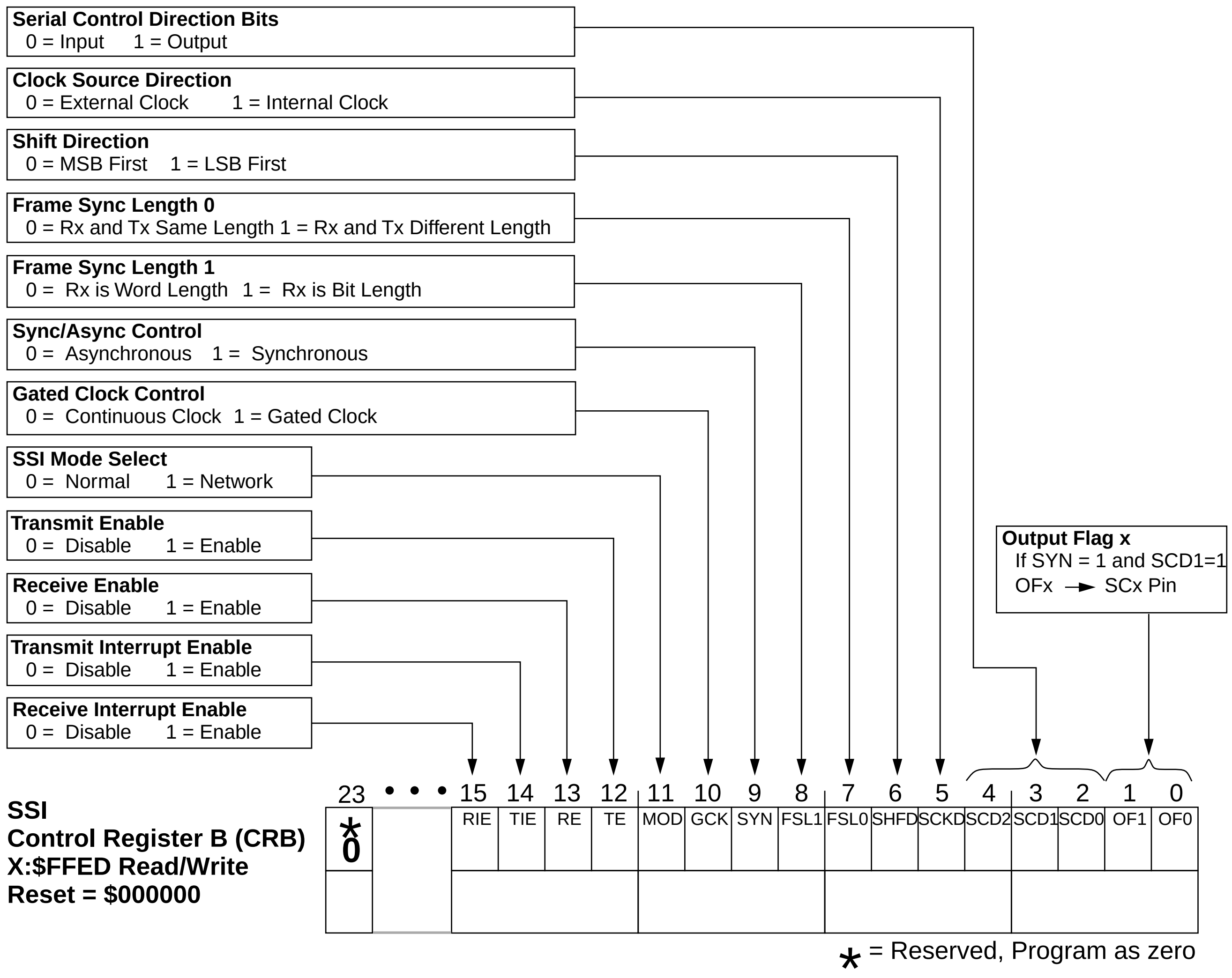


Figure B-32 SSI Control Register B (CRB)

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 can and do vary in different applications. All operating parameters, including “Typical”, must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola 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 Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part.

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, United Kingdom.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbor Center, No. 2 Dai King Street, Tai Po Industrial
Estate, Tai Po, N.T., Hong Kong.

MOTOROLA