

DSP56L811

16-bit Digital Signal Processor User's Manual

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

OnCE™ is a trademark of Motorola, Inc.

© MOTOROLA INC., 1996

Order this document by DSP56L811UM/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.

TABLE OF CONTENTS

SECTION 1 DSP56L811 OVERVIEW	1-1
1.1 INTRODUCTION	1-3
1.2 MANUAL ORGANIZATION	1-4
1.3 MANUAL CONVENTIONS	1-5
1.4 DSP56800 CORE DESCRIPTION	1-6
1.4.1 Data ALU	1-9
1.4.2 Address Generation Unit	1-10
1.4.3 Program Controller and Hardware Looping Unit	1-10
1.4.4 Address and Data Buses	1-10
1.4.5 Bus and Bit Manipulation Unit	1-11
1.4.6 On-Chip Emulation (OnCE) Unit	1-12
1.5 DSP56L811 ARCHITECTURE OVERVIEW	1-12
1.5.1 DSP56L811 Peripheral Blocks	1-13
1.5.1.1 On-chip Memory	1-14
1.5.1.2 External Memory Interface	1-14
1.5.1.3 General Purpose I/O Module	1-14
1.5.1.4 Programmable I/O Module	1-14
1.5.1.5 Serial Peripheral Interface	1-14
1.5.1.6 Synchronous Serial Interface	1-15
1.5.1.7 General-Purpose Timer Module	1-15
1.5.1.8 On-chip Clock Synthesis Block	1-15
1.5.1.9 COP/RTI Module	1-15
1.5.1.10 JTAG/OnCE Port	1-15
1.5.2 DSP56L811 Peripheral Interrupts	1-16
1.6 CODE DEVELOPMENT ON THE DSP56L811	1-16
 SECTION 2 PIN DESCRIPTIONS	 2-1
2.1 INTRODUCTION	2-3
2.2 PIN DESCRIPTIONS	2-6

SECTION 3	MEMORY CONFIGURATION AND OPERATING MODES	3-1
3.1	INTRODUCTION	3-3
3.2	DSP56L811 MEMORY MAP DESCRIPTION	3-3
3.2.1	Program Memory Map	3-6
3.2.2	On-Chip Peripheral Memory Map	3-7
3.3	DSP56L811 OPERATING MODES	3-9
3.3.1	Bootstrap Mode (Mode 0)	3-10
3.3.2	Bootstrap Mode (Mode 1)	3-10
3.3.3	Normal Expanded Mode (Mode 2)	3-10
3.3.4	Development Mode (Mode 3)	3-11
3.4	DSP56L811 BOOTSTRAPPING	3-11
3.5	EXECUTING PROGRAMS FROM XRAM	3-12
3.5.1	Description of the XRAM Execution Mode	3-12
3.5.2	Interrupts in the XRAM Execution Mode	3-14
3.5.3	Entering the XRAM Execution Mode	3-14
3.5.4	Exiting the XRAM Execution Mode	3-14
3.5.5	Restrictions in the XRAM Execution Mode	3-15
3.6	DSP56L811 RESET AND INTERRUPT VECTORS	3-15
3.6.1	DSP56L811 Status Register	3-17
3.6.2	DSP56L811 Interrupt Priority Register	3-19
SECTION 4	EXTERNAL MEMORY INTERFACE	4-1
4.1	INTRODUCTION	4-3
4.2	EXTERNAL MEMORY PORT ARCHITECTURE	4-3
4.3	EXTERNAL MEMORY PORT DESCRIPTION	4-5
4.3.1	External Memory Port Programing Model	4-5
4.3.1.1	Bus Control Register (BCR)	4-5
4.3.1.2	Drive (DRV) Bit 9	4-5
4.3.1.3	Wait State X-Data Memory (WSX) Bits 7-4	4-6
4.3.1.4	Wait State P Memory (WSPM) Bits 3-0	4-6
4.3.1.5	Reserved BCR Register Bits	4-8

4.3.2	Pins in Different Processing States	4-8
SECTION 5 PORT B GPIO FUNCTIONALITY		5-1
5.1	INTRODUCTION	5-3
5.2	PORT B PROGRAMMING MODEL	5-5
5.2.1	Port B Data Direction Register (PBDDR)	5-6
5.2.2	Port B Data Register (PBD)	5-6
5.2.2.1	PBD Bit Values for General Purpose Inputs	5-6
5.2.2.2	PBD Bit Values for General Purpose Outputs	5-6
5.2.2.3	PBD Bit Values for Interrupt Inputs	5-6
5.2.3	Port B Interrupt Register (PBINT)	5-7
5.2.3.1	Interrupt Mask (MSK7-MSK0) Bits 15-8	5-7
5.2.3.2	Interrupt Invert (INV7-INV0) Bits 7-0	5-8
5.3	PORT B INTERRUPT GENERATION	5-8
5.4	PORT B PROGRAMMING EXAMPLES	5-9
5.4.1	Receiving Data on Port B	5-10
5.4.2	Sending Data on Port B	5-11
5.4.3	Looping Data on Port B	5-13
5.4.4	Generating Interrupts on Port B	5-15
SECTION 6 PORT C GPIO FUNCTIONALITY		6-1
6.1	INTRODUCTION	6-3
6.2	PORT C PROGRAMMING MODEL	6-5
6.2.1	Port C Control Register (PCC)	6-5
6.2.2	Port C Data Direction Register (PCDDR)	6-6
6.2.3	Port C Data Register (PCD)	6-7
6.3	PORT C PROGRAMMING EXAMPLES	6-7
6.3.1	Receiving Data on Port C GPIO Pins	6-7
6.3.2	Sending Data on Port C GPIO Pins	6-9
6.3.3	Looping Data on Port C GPIO Pins	6-11
SECTION 7 SERIAL PERIPHERAL INTERFACE		7-1
7.1	INTRODUCTION	7-3
7.2	SPI ARCHITECTURE	7-5
7.3	SPI PROGRAMMING MODEL	7-6
7.3.1	SPI Control Registers (SPCR0 and SPCR1)	7-7

7.3.1.1	SPI Clock Rate Select (SPR2-0) Bits 8, 1-0	7-8
7.3.1.2	SPI Interrupt Enable (SPIE) Bit 7	7-9
7.3.1.3	SPI Enable (SPE) Bit 6.	7-9
7.3.1.4	Wired-OR Mode (WOM) Bit 5.	7-9
7.3.1.5	Master Mode Select (MST) Bit 4	7-10
7.3.1.6	Clock Polarity (CPL) Bit 3.	7-10
7.3.1.7	Clock Phase (CPH) Bit 2	7-10
7.3.1.8	Reserved SPCR Register Bits	7-10
7.3.2	SPI Status Register (SPSR0 and SPSR1)	7-10
7.3.2.1	SPI Interrupt Complete Flag (SPIF) Bit 7	7-10
7.3.2.2	Write Collision (WCOL) Bit 6	7-11
7.3.2.3	Mode Fault (MDF) Bit 4	7-11
7.3.2.4	Reserved SPSR Register Bits	7-11
7.3.3	SPI Data Registers (SPDR0 and SPDR1)	7-11
7.4	SPI DATA AND CONTROL PINS	7-11
7.5	SPI SYSTEM ERRORS.	7-14
7.5.1	SPI Mode-Fault Error	7-14
7.5.2	SPI Write-Collision Error.	7-15
7.5.3	SPI Overrun	7-16
7.6	CONFIGURING PORT C FOR SPI FUNCTIONALITY	7-16
7.6.1	SPI Low Power Operation.	7-17
7.7	PROGRAMMING EXAMPLES	7-18
7.7.1	Configuring an SPI Port as Master	7-18
7.7.2	Configuring an SPI Port as Slave	7-20
7.7.3	Sending Data from Master to Slave	7-23
SECTION 8 SYNCHRONOUS SERIAL INTERFACE		8-1
8.1	INTRODUCTION	8-3
8.2	SSI ARCHITECTURE	8-4
8.2.1	SSI Clocking	8-6
8.2.2	SSI Clock and Frame Sync Generation	8-6
8.3	SSI PROGRAMMING MODEL	8-8
8.3.1	SSI Transmit Shift Register (TXSR)	8-10
8.3.2	SSI Transmit Data Buffer Register	8-10
8.3.3	SSI Transmit Data Register (STX)	8-11
8.3.4	SSI Receive Shift Register (RXSR)	8-11

8.3.5	SSI Receive Data Buffer Register	8-11
8.3.6	SSI Receive Data Register (SRX)	8-12
8.3.7	SSI Transmit and Receive Control Registers (SCRTX and SCRRX)	8-12
8.3.7.1	Prescaler Range (PSR) Bit 15	8-12
8.3.7.2	Word Length Control (WL1-WL0) Bits 14-13	8-13
8.3.7.3	Frame Rate Divider (DC4-DC0) Bits 12-8	8-13
8.3.7.4	Prescale Modulus Select (PM7-PM0) Bits 7-0	8-13
8.3.8	SSI Control Register 2 (SCR2)	8-14
8.3.8.1	Receive Interrupt Enable (RIE) Bit 15	8-15
8.3.8.2	Transmit Interrupt Enable (TIE) Bit 14	8-15
8.3.8.3	Receive Enable (RE) Bit 13	8-16
8.3.8.4	Transmit Enable (TE) Bit 12	8-16
8.3.8.5	Receive Buffer Enable (RBF) Bit 11	8-17
8.3.8.6	Transmit Buffer Enable (TBF) Bit 10	8-17
8.3.8.7	Receive Direction (RXD) Bit 9	8-17
8.3.8.8	Transmit Direction (TXD) Bit 8	8-18
8.3.8.9	Synchronous Mode (SYN) Bit 7	8-18
8.3.8.10	Shift Direction (MSB/LSB Position) (SHFD) Bit 6	8-18
8.3.8.11	Clock Polarity (SCKP) Bit 5	8-18
8.3.8.12	SSI Enable (SSIEN) Bit 4	8-18
8.3.8.13	Network Mode (NET) Bit 3	8-19
8.3.8.14	Frame Sync Invert (FSI) Bit 2	8-19
8.3.8.15	Frame Sync Length (FSL) Bit 1	8-19
8.3.8.16	Early Frame Sync (EFS) Bit 0	8-19
8.3.9	SSI Status Register (SSR)	8-19
8.3.9.1	Receive Data Register Full (RDF) Bit 7	8-19
8.3.9.2	Transmit Data Register Empty (TDE) Bit 6	8-20
8.3.9.3	Receive Overrun Error (ROE) Bit 5	8-20
8.3.9.4	Transmit Underrun Error (TUE) Bit 4	8-20
8.3.9.5	Transmit Frame Sync (TFS) Bit 3	8-21
8.3.9.6	Receive Frame Sync (RFS) Bit 2	8-21
8.3.9.7	Receive Data Buffer Full (RDBF) Bit 1	8-21
8.3.9.8	Transmit Data Buffer Empty (TDBE) Bit 0	8-22
8.3.9.9	Reserved SSR Register Bits	8-22
8.3.10	SSI Time Slot Register (STSR)	8-22

8.4	SSI DATA AND CONTROL PINS	8-22
8.4.1	SSI Pin Definitions	8-25
8.5	SSI OPERATING MODES.	8-26
8.5.1	Normal Mode.	8-28
8.5.1.1	Normal Mode Transmit.	8-28
8.5.1.2	Normal Mode Receive	8-29
8.5.2	Network Mode.	8-31
8.5.2.1	Network Mode Transmit	8-31
8.5.2.2	Network Mode Receive	8-32
8.5.3	Gated Clock Operation	8-35
8.6	SSI RESET AND INITIALIZATION PROCEDURE	8-35
8.7	CONFIGURING PORT C FOR SSI FUNCTIONALITY	8-36
SECTION 9 TIMERS		9-1
9.1	INTRODUCTION	9-3
9.2	TIMER PROGRAMMING MODEL.	9-5
9.2.1	Timer Control Registers (TCR01 and TCR2)	9-6
9.2.1.1	Timer Enable (TE) Bit 15, Bit 7.	9-6
9.2.1.2	Invert (INV) Bit 6.	9-6
9.2.1.3	Overflow Interrupt Enable (OIE) Bit 12, Bit 4	9-7
9.2.1.4	Timer Output Enable (TO1-TO0) Bits 11-10, Bits 3-2	9-7
9.2.1.5	Event Select (ES1-ES0) Bits 9-8, Bits 1-0	9-8
9.2.1.6	Reserved TCR Register Bits	9-9
9.2.2	Timer Preload Register (TPR).	9-9
9.2.3	Timer Count Register (TCT)	9-9
9.3	TIMER RESOLUTION	9-10
9.4	TIMER INTERRUPT PRIORITIES.	9-11
9.5	EVENT COUNTING WITH THE TIMER MODULE	9-11
9.6	TIMER MODULE LOW POWER OPERATION	9-19
9.6.1	Turning Off the Entire Timer Module.	9-19
9.6.2	Turning Off Any Timer Not in Use.	9-20
9.6.3	Low Frequency Timer Operation.	9-20
9.6.4	Running a Timer in Wait Mode	9-20
9.6.5	Running a Timer in Stop Mode	9-20
9.7	TIMER MODULE TIMING DIAGRAMS	9-21

SECTION 10 ON-CHIP CLOCK SYNTHESIS	10-1
10.1 INTRODUCTION	10-3
10.2 DSP56L811 TIMING SYSTEM ARCHITECTURE	10-3
10.2.1 Oscillator	10-5
10.2.2 Phase-Locked Loop (PLL)	10-5
10.2.3 Prescaler	10-6
10.2.4 Clockout MUX	10-7
10.2.5 Control Registers	10-7
10.3 CLOCK SYNTHESIS PROGRAMMING MODEL	10-7
10.3.1 PLL Control Register 1 (PCR1)	10-8
10.3.1.1 PLL Enable (PLLE) and PLL Power Down (PLLD) Bits 14 and 13	10-8
10.3.1.2 Low Power Stop (LPST) Bit 12	10-9
10.3.1.3 Prescaler Divider (PS2-PS0) Bits 10-8	10-9
10.3.1.4 CLKO Select (CS1-CS0) Bits 7-6	10-10
10.3.1.5 Reserved PCR1 Register Bits	10-11
10.3.2 PLL Control Register 0 (PCR0)	10-11
10.3.2.1 Feedback Divider (YD9-YD0) Bits 13-5	10-11
10.3.2.2 Reserved PCR0 Register Bits	10-12
10.4 DSP56L811 STOP AND WAIT MODES	10-12
10.4.1 Options for Clock Synthesis in Stop Mode	10-12
10.4.2 Stop Mode—PLL, COP, Realtime Clock, Timer Module and CLKO Pin Enabled	10-13
10.4.3 Stop Mode—COP, Realtime Clock and CLKO Pin Enabled	10-13
10.4.4 Stop Mode—PLL and CLKO Pin Enabled	10-14
10.4.5 Stop Mode—CLKO Pin Enabled	10-14
10.4.6 Stop Mode—Everything Disabled	10-14
10.5 PLL LOCK	10-14
10.5.1 PLL Programming Sequence	10-15
10.5.2 Changing the PLL Frequency	10-16
10.5.3 Turning Off the PLL Before Entering Stop Mode	10-16
10.6 PLL MODULE LOW-POWER OPERATION	10-17
10.6.1 Turning Off the Entire Clock Synthesis Module	10-17

10.6.2	Turning Off the Prescaler Divider When Not in Use . . .	10-17
10.6.3	Turning Off the PLL When Not in Use.	10-17
10.6.4	Turning Off the CLK0 Pin When Not in Use.	10-18

SECTION 11 COP/RTI MODULE 11-1

11.1	INTRODUCTION	11-3
11.2	COP AND RTI	11-3
11.2.1	COP and Realtime Timer Architecture	11-4
11.2.2	COP and Realtime Timer Programming Model.	11-5
11.2.3	COP and RTI Control Register (COPCTL)	11-6
11.2.3.1	COP Enable (CPE) Bit 15	11-7
11.2.3.2	COP Timer Divider (CT) Bit 14.	11-7
11.2.3.3	Realtime Timer Enable (RTE) Bit 11	11-7
11.2.3.4	Realtime Timer Interrupt Enable (RTIE) Bit 10.	11-7
11.2.3.5	Realtime Timer Interrupt Flag (RTIF) Bit 9.	11-8
11.2.3.6	Realtime Prescaler (RP) Bit 8	11-8
11.2.3.7	Realtime/COP Divider (DV7-DV0) Bits 7-0	11-8
11.2.3.8	Reserved COPCTL Register Bits	11-8
11.2.4	COP and RTI Count Register (COPCNT).	11-8
11.2.4.1	Realtime/COP Divider (DV7-DV0) Bits 12-5	11-9
11.2.4.2	Realtime Prescaler (RP) Bits 4-3	11-9
11.2.4.3	Scaler (SC2-SC0) Bits 2-0	11-9
11.2.4.4	Reserved COPCNT Register Bits	11-9
11.2.5	COP Reset Register (COPRST).	11-9
11.3	PROGRAMMING THE COP AND RTI TIMERS	11-10
11.3.1	COP and Realtime Timer Resolution	11-11
11.3.2	COP/Realtime Timer Low Power Operation	11-11
11.3.3	Programming Example	11-12

SECTION 12 JTAG PORT 12-1

12.1	INTRODUCTION	12-3
12.2	JTAG PORT ARCHITECTURE	12-4
12.3	JTAG/ONCE PORT PINOUT.	12-5
12.4	JTAG REGISTERS	12-6
12.4.1	JTAG Instruction Register and Decoder	12-8
12.4.1.1	EXTEST (B[3:0]=0000).	12-8

12.4.1.2	SAMPLE/PRELOAD (B[3:0]=0001)	12-9
12.4.1.3	IDCODE (B[3:0]=0010)	12-9
12.4.1.4	EXTEST_PULLUP (B[3:0]=0011)	12-10
12.4.1.5	HIGHZ (B[3:0]=0100)	12-10
12.4.1.6	CLAMP (B[3:0]=0101)	12-10
12.4.1.7	ENABLE_ONCE (B[3:0]=0110)	12-11
12.4.1.8	DEBUG_REQUEST (B[3:0]=0111)	12-11
12.4.1.9	BYPASS (B[3:0]=1111)	12-11
12.4.2	JTAG Chip Identification Register	12-12
12.4.3	JTAG Boundary-Scan Register	12-13
12.4.4	JTAG Bypass Register	12-16
12.4.5	TAP Controller	12-17
12.5	DSP56L811 RESTRICTIONS	12-18
APPENDIX A BOOTSTRAP PROGRAM		A-1
APPENDIX B PROGRAMMER'S SHEETS		B-1
INDEX		I-1

LIST OF FIGURES

Figure 1-1	DSP56800 Core Block Diagram.	1-7
Figure 1-2	DSP56800 Bus Block Diagram	1-8
Figure 1-3	DSP56L811 Functional Block Diagram	1-13
Figure 1-4	Example of Code Development with Visibility on All Memory Accesses.	1-18
Figure 2-1	Top View of the DSP56L811 100-pin Plastic Thin Quad Flat Package	2-3
Figure 2-2	Functional Group Pin Allocations	2-4
Figure 3-1	DSP56L811 On-chip Memory Map	3-4
Figure 3-2	Bus Control Register Programming Model.	3-5
Figure 3-3	Operating Mode Register Programming Model	3-5
Figure 3-4	DSP56L811 Memory Map (XRAM Execution Mode)	3-13
Figure 3-5	Status Register Programming Model	3-18
Figure 3-6	DSP56L811 Interrupt Priority Register (IPR), X:\$FFFB.	3-19
Figure 4-1	DSP56L811 Input/Output Block Diagram.	4-4
Figure 4-2	External Memory Port Programming Model	4-5
Figure 4-3	Bus Operation (Read/Write, Zero Wait States)	4-7
Figure 4-4	Bus Operation (Read/Write—Three Wait States).	4-8
Figure 5-1	DSP56L811 Input/Output Block Diagram.	5-4
Figure 5-2	DSP56L811 Port B Programming Model	5-5
Figure 5-3	Port B Interrupt Block Diagram.	5-9

Figure 6-1	DSP56L811 Input/Output Block Diagram	6-4
Figure 6-2	DSP56L811 Port C Programming Model.	6-5
Figure 7-1	DSP56L811 Input/Output Block Diagram	7-4
Figure 7-2	SPI Block Diagram	7-5
Figure 7-3	SPI Programming Model	7-7
Figure 8-1	DSP56L811 Input/Output Block Diagram	8-4
Figure 8-2	SSI Block Diagram	8-5
Figure 8-3	SSI Clocking	8-6
Figure 8-4	SSI Transmit Clock Generator Block Diagram	8-7
Figure 8-5	SSI Transmit Frame Sync Generator Block Diagram	8-7
Figure 8-6	SSI Programming Model—Register Set	8-9
Figure 8-7	SSI Interrupt Vectors	8-10
Figure 8-8	SSI Bit Clock Equation	8-13
Figure 8-9	Asynchronous SSI Configurations—Continuous Clock.	8-23
Figure 8-10	Synchronous SSI Configurations—Continuous and Gated Clock.	8-24
Figure 8-11	Serial Clock and Frame Sync Timing	8-26
Figure 8-12	Normal Mode Timing—Continuous Clock	8-30
Figure 8-13	Normal Mode Timing—Gated Clock	8-30
Figure 8-14	Network Mode Timing—Continuous Clock	8-34
Figure 9-1	General Purpose Timer Module	9-4
Figure 9-2	General Purpose Timer Programming Model	9-5

Figure 9-3	Standard Timer Operation with Overflow Interrupt.	9-21
Figure 9-4	Write to the Count Register After Writing the Preload Register When Timer is Disabled	9-22
Figure 10-1	DSP56L811 Timing System	10-4
Figure 10-2	PLL Block Diagram.	10-6
Figure 10-3	Prescaler Clock Equations	10-6
Figure 10-4	Clock Synthesis Programming Model	10-8
Figure 11-1	Realtime and COP Timer Block Diagram.	11-4
Figure 11-2	Realtime Interrupt and COP Timer Programming Model	11-6
Figure 12-1	JTAG Block Diagram	12-5
Figure 12-2	JTAG Port Programming Model	12-7
Figure 12-3	Bypass Register	12-11
Figure 12-4	Chip Identification Register Configuration	12-12
Figure 12-5	TAP Controller State Diagram	12-17

LIST OF TABLES

Table 1-1	High True / Low True Signal Conventions.	1-6
Table 1-2	Data Buses	1-11
Table 2-1	Functional Group Pin Allocations	2-5
Table 2-2	Power Pins	2-6
Table 2-3	Ground Pins.	2-6
Table 2-4	PLL and Clock Pins	2-6
Table 2-5	Address Bus Pins	2-7
Table 2-6	Data Bus Pins	2-7
Table 2-7	Bus Control Pins	2-8
Table 2-8	Interrupt and Mode Control Pins	2-9
Table 2-9	Programmable Interrupt GPIO Pins.	2-10
Table 2-10	Dedicated General Purpose Input/Output (GPIO) Pins.	2-10
Table 2-11	Serial Peripheral Interface (SPI0 and SPI1) Pins	2-11
Table 2-12	Synchronous Serial Interface (SSI) Pins.	2-13
Table 2-13	Timer Module Pins.	2-15
Table 2-14	JTAG/On-Chip Emulation (OnCE) Pins.	2-16
Table 3-1	DSP56L811 I/O and On-Chip Peripheral Memory Map.	3-7
Table 3-2	DSP56L811 Program RAM Chip Operating Modes	3-9
Table 3-3	Interrupt Priority Structure	3-16
Table 3-4	Reset and Interrupt Vector Map	3-16

Table 3-5	Interrupt Mask Bit Definition	3-18
Table 4-1	Programming WSX Bits for Wait States	4-6
Table 4-2	Programming WSP Bits for Wait States	4-6
Table 4-3	Port A Operation with DRV Bit Set to 0	4-9
Table 4-4	Port A Operation with DRV Bit Set to 1	4-9
Table 5-1	PBDDR Bit Definition	5-6
Table 5-2	Reading the PBD Register	5-7
Table 5-3	MSK Bit Definition	5-7
Table 5-4	INV Bit Definition	5-8
Table 6-1	PCC Bit Definition.	6-6
Table 6-2	PCDDR Bit Definition	6-6
Table 7-1	SPR Divider Programming	7-8
Table 7-2	SPI Interrupt.	7-9
Table 7-3	SPI Mode Programming.	7-10
Table 7-4	PCC Register Programming for the SS Pin	7-16
Table 8-1	SSI Data Word Lengths	8-13
Table 8-2	SSI Bit Clock as a Function of Phi Clock and Prescale Modulus	8-14
Table 8-3	SSI Receive Data Interrupts.	8-15
Table 8-4	SSI Transmit Data Interrupts	8-16
Table 8-5	Frame Sync / Clock Pin Configuration	8-17
Table 8-6	SSI Operating Modes.	8-27
Table 8-7	SSI Control Bits Requiring Reset Before Change	8-36

Table 9-1	Timer Control Registers TCR01 and TCR02.	9-6
Table 9-2	INV Bit Definition	9-6
Table 9-3	Timer Interrupt Vectors	9-7
Table 9-4	TIO Pin Function	9-7
Table 9-5	ES Bit Definition.	9-8
Table 9-6	Timer Range and Resolution.	9-10
Table 9-7	Timer Interrupt Priorities	9-11
Table 10-1	PLL Operations	10-9
Table 10-2	PS Divider Programming.	10-10
Table 10-3	CLKOUT Pin Control	10-11
Table 11-1	COP Timer Divider Definition	11-7
Table 11-2	Realtime Prescaler Definition	11-8
Table 11-3	Realtime Prescaler Bits in COPCNT	11-9
Table 11-4	COP Timer Range and Resolution	11-11
Table 12-1	JTAG Pin Descriptions	12-6
Table 12-2	JTAG Instruction Register Encodings	12-8
Table 12-3	Device ID Register Bit Assignment	12-12
Table 12-4	Boundary-Scan Register Contents for DSP56L811	12-13
Table B-1	Instruction Set Summary	B-3
Table B-2	Condition Code Register (CCR) Symbols (Standard Definitions)	B-8
Table B-3	Condition Code Register Notation.	B-8
Table B-4	Interrupt Priority Structure	B-9

List of Tables

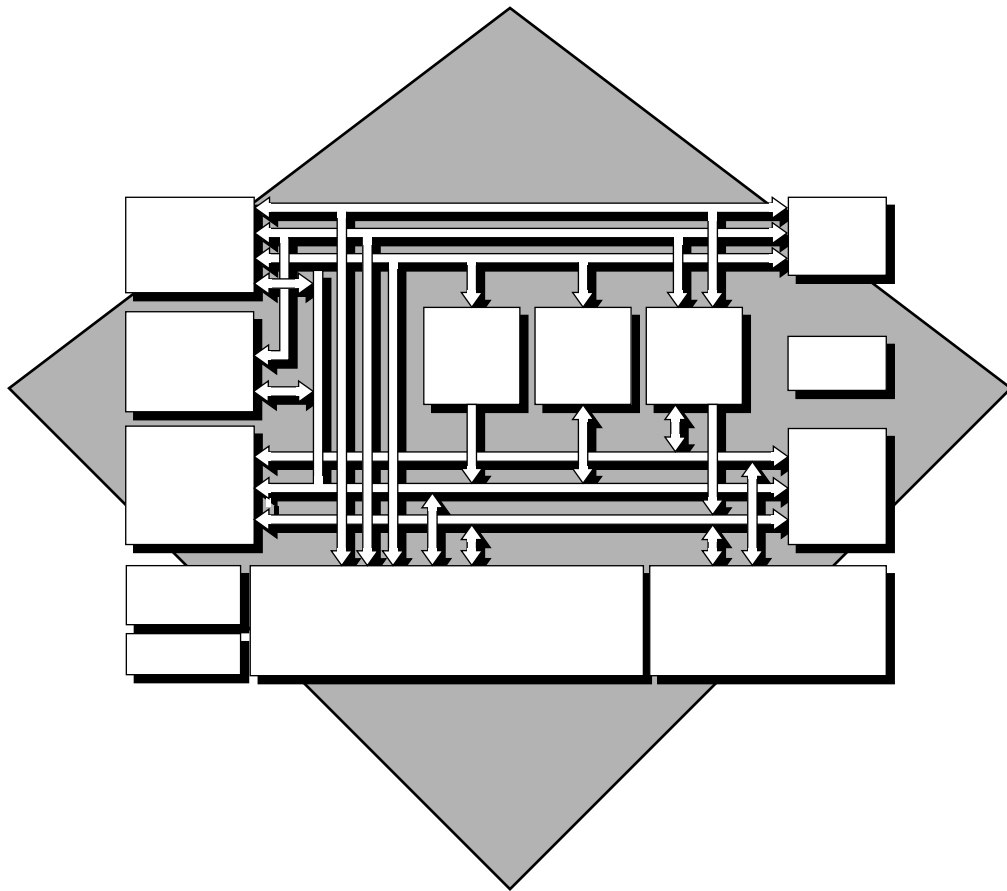
Table B-5	Reset and Interrupt Vectors	B-9
Table B-6	DSP56L811 I/O and On-Chip Peripheral Memory Map	B-11
Table B-7	List of Programmer's Sheets	B-13

LIST OF EXAMPLES

Example 1-1	Sample Code Listing	1-5
Example 1-2	On-Chip and Off-Chip Instruction Fetches	1-17
Example 5-1	Receiving Data on Port B	5-10
Example 5-2	Sending Data on Port B.	5-11
Example 5-3	Loop-back Example.	5-13
Example 5-4	Generating Interrupts on Port B.	5-15
Example 6-1	Receiving Data on Port C GPIO Pins	6-7
Example 6-2	Sending Data on Port C GPIO Pins.	6-9
Example 6-3	Loop-back Example.	6-11
Example 7-1	Configuring an SPI Port as Master	7-18
Example 7-2	Configuring an SPI Port as Slave	7-20
Example 7-3	Sending Data from Master to Slave.	7-23
Example 9-1	Counting Events on a Pin	9-12
Example 9-2	Decrementing to 0 and Generating an Interrupt.	9-14
Example 9-3	Timer Using 25% Duty Cycle.	9-16
Example 10-1	Programming the PLL	10-15
Example 11-1	Sending a COP Reset.	11-12
Example A-1	DSP56L811 Bootstrap Code.	A-4

SECTION 1

DSP56L811 OVERVIEW



1.1	INTRODUCTION	1-3
1.2	MANUAL ORGANIZATION	1-4
1.3	MANUAL CONVENTIONS	1-5
1.4	DSP56800 CORE DESCRIPTION	1-6
1.5	DSP56L811 ARCHITECTURE OVERVIEW	1-12
1.6	CODE DEVELOPMENT ON THE DSP56L811	1-16

1.1 INTRODUCTION

This manual describes the DSP56L811 16-bit digital signal processor (DSP), its memory and operating modes, and its peripheral modules. This manual is intended to be used with the *DSP56800 Family Manual (DSP56800FAM/AD)*, which describes the central processing unit, programming models, and instruction set details. The *DSP56L811 Data Sheet (DSP56L811/D)* provides electrical specifications, timing, pinout, and packaging descriptions. These documents as well as Motorola's DSP development tools can be obtained through a local Motorola Semiconductor Sales Office or authorized distributor.

To receive the latest information, access the Motorola DSP home page located at <http://www.motorola-dsp.com/dsp>

The DSP56L811 is a member of the DSP56800 core-based family of DSPs. This general-purpose DSP combines processing power with configuration flexibility, making it an excellent cost-effective solution for signal processing and control functions. To achieve its design goals, the DSP56L811 uses an efficient MPU-style, general-purpose, 16-bit DSP core, program and data memories, and support circuitry.

The central processing unit, the DSP56800 core, consists of three execution units operating in parallel, allowing as many as six operations during each instruction cycle. The MPU-style programming model and optimized instruction set allow straightforward generation of efficient, compact DSP and control code. The instruction set is also highly efficient for C compilers.

The DSP56L811 supports program execution from internal or external memories. Two data operands can be accessed per instruction cycle from the on-chip data RAM. The programmable peripherals and ports provide support for interfacing multiple external devices such as codecs, microprocessors, or other DSPs. The DSP56L811 also provides 16 to 32 GPIO lines, depending on how peripherals are configured, and two external dedicated interrupt lines. Because of its configuration flexibility, compact program code, and low cost, the DSP56800 core family is well-suited for cost-sensitive applications including digital wireless messaging, digital answering machines/feature phones, wireline and wireless modems, servo and AC motor control, and digital cameras.

1.2 MANUAL ORGANIZATION

This manual is arranged in the following sections:

- *Section 1—Introduction* provides a brief overview of the DSP56L811, describes the structure of this document, and lists other documentation necessary to use the DSP56L811.
- *Section 2—Pin Descriptions* provides a description of the pins on the DSP56L811 chip and how the pins are grouped into the various interfaces.
- *Section 3—Memory Configuration and Operating Modes* describes the on-chip memory, structures, registers, and interfaces.
- *Section 4—External Memory Interface* describes the DSP56L811 external memory interface, which is also referred to as Port A.
- *Section 5—Port B GPIO Functionality* describes the dedicated GPIO interface, which is also referred to as Port B.
- *Section 6—Port C* describes the 16 dual-function pins that constitute Port C; this section defines the GPIO functions, and sections 6, 7, and 9 describe these pins' alternate functionality.
- *Section 7—Serial Peripheral Interface* describes the Serial Peripheral Interface (SPI), which communicates with external devices such as LCDs and MCUs, and is a part of Port C.
- *Section 8—Synchronous Serial Interface* describes the Synchronous Serial Interface (SSI), which communicates with devices such as codecs, other DSPs, microprocessors, and peripherals, to provide the primary data input path, and is a part of Port C.
- *Section 9—Timers* describes the three internal timer/counter devices that are a part of Port C.
- *Section 10—On-Chip Clock Synthesis* describes the internal oscillator, PLL, and timer distribution chain for the DSP56L811 chip.
- *Section 11—COP/RTI Module* describes the on-chip watchdog timer and the realtime interrupt generator.
- *Section 12—JTAG Port* describes the specifics of the DSP56L811's JTAG port. The OnCE™ module, which is accessed through the JTAG port, is described in the *DSP56800 Family Manual (DSP56800FAM/AD)*.
- *Appendix A—Bootstrap Program* provides a listing of the bootstrap code used to start or reset the DSP56L811.

- *Appendix B—Programmer's Sheets* provides programming references and master programming sheets used to program the DSP56L811 registers.
- *Index* provides a cross-reference to topics in this manual.

1.3 MANUAL CONVENTIONS

The following conventions are used in this manual:

- Bits within registers are always listed from most significant byte (MSB) to least significant byte (LSB).

Note: Other manuals use the opposite convention, with bits listed from LSB to MSB.

- Bits within a register are indicated AA[n:0] when more than one bit is involved in a description. For purposes of description, the bits are presented as if they are contiguous within a register. However, this is not always the case. Refer to the programming model diagrams or to the programmer's sheets to see the exact location of bits within a register.
- When a bit is described as “set,” its value is set to 1. When a bit is described as “cleared,” its value is set to 0.
- Pins or signals that are asserted low (made active when pulled to ground) have an overbar over their name, for example, the $\overline{SS0}$ pin is asserted low.
- Hex values are indicated with a dollar sign (\$) preceding the hex value, as follows: \$FFFB is the X memory address for the Interrupt Priority Register (IPR).
- Code examples are displayed in a monospaced font, as shown in **Example 1-1**.

Example 1-1 Sample Code Listing

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

- Pins or signals listed in code examples that are asserted low have a tilde in front of their names. In the previous example, line 3 refers to the $\overline{SS0}$ pin (shown as ~SS0).
- The word “assert” means that a high true (active high) signal is pulled high to V_{CC} or that a low true (active low) signal is pulled low to ground. The word

“deassert” means that a high true signal is pulled low to ground or that a low true signal is pulled high to V_{CC} . See **Table 1-1**.

Table 1-1 High True / Low True Signal Conventions

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

1. 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).
2. 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).

- The word “reset” is used in three different contexts in this manual. There is a reset pin that is always written as “ $\overline{\text{RESET}}$,” a reset instruction that is always written as “RESET,” and the word reset that refers to the reset function and is written in lower case with a leading capital letter as grammar dictates. The word “pin” is a generic term for any pin on the chip.

1.4 DSP56800 CORE DESCRIPTION

The DSP56800 core consists of functional units that operate in parallel to increase throughput of the machine. Major features of the DSP56800 core include the following:

- Data Arithmetic Logic Unit (ALU)
- Address Generation Unit (AGU)
- Program Controller and Hardware Looping Unit
- Bus and Bit Manipulation Unit
- Address buses
- Data buses
- JTAG/OnCE port
- Clock generator

An overall block diagram of the DSP56800 core architecture is shown in **Figure 1-1**. The DSP56800 core is fed by internal program memory, a single internal data memory, an external memory interface, an on-chip phase-locked loop (PLL), 16 dedicated GPIO pins (eight of which can be programmed as interrupt inputs), two serial peripheral interface (SPI) ports for MCU support, a synchronous serial interface (SSI) port for codec support, three 16-bit general purpose timer/event counters, real-time and computer-operating-properly (COP) timers, two external mode select/interrupt lines, an external reset line, and a JTAG/On-Chip Emulation (OnCE) port.

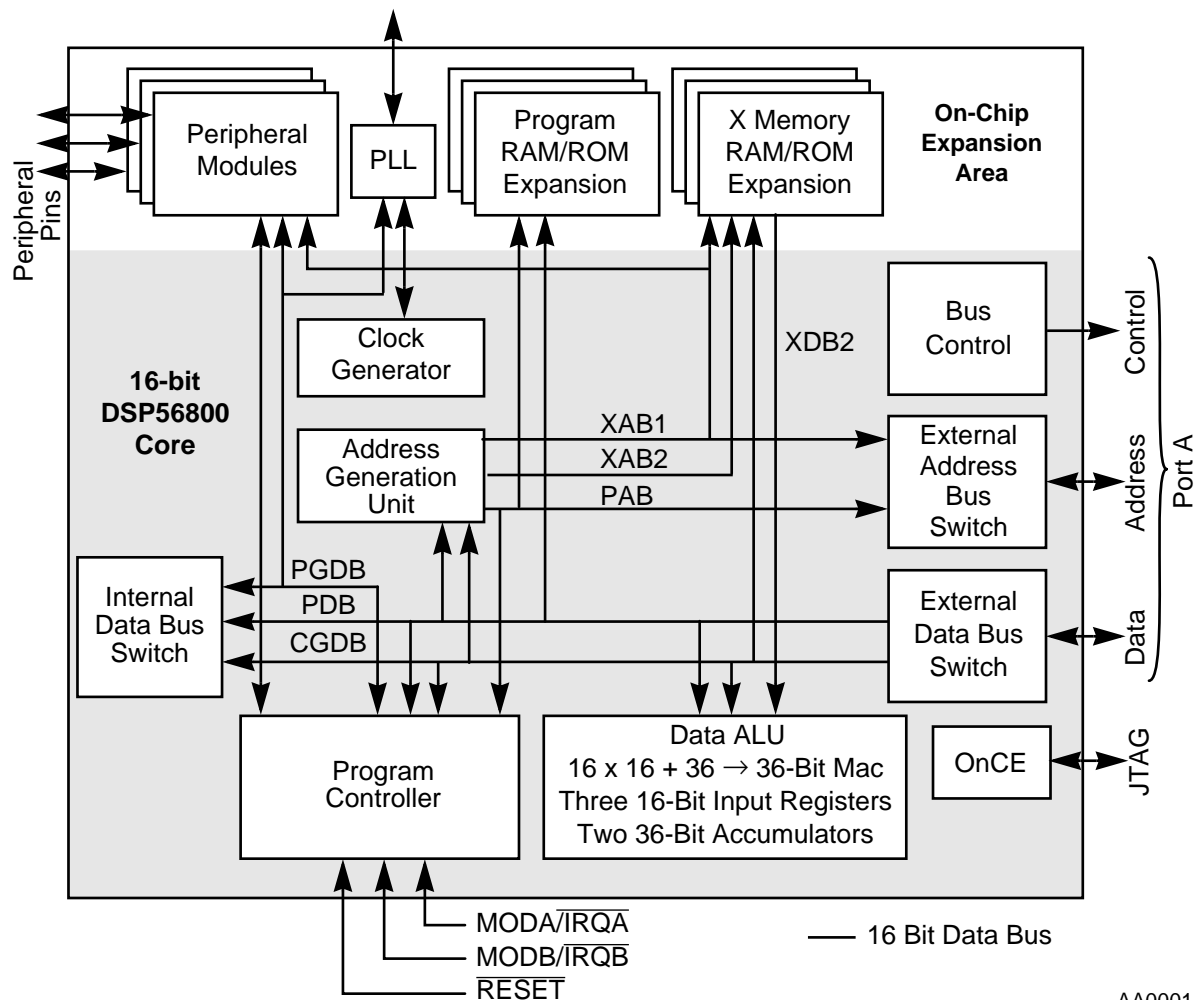
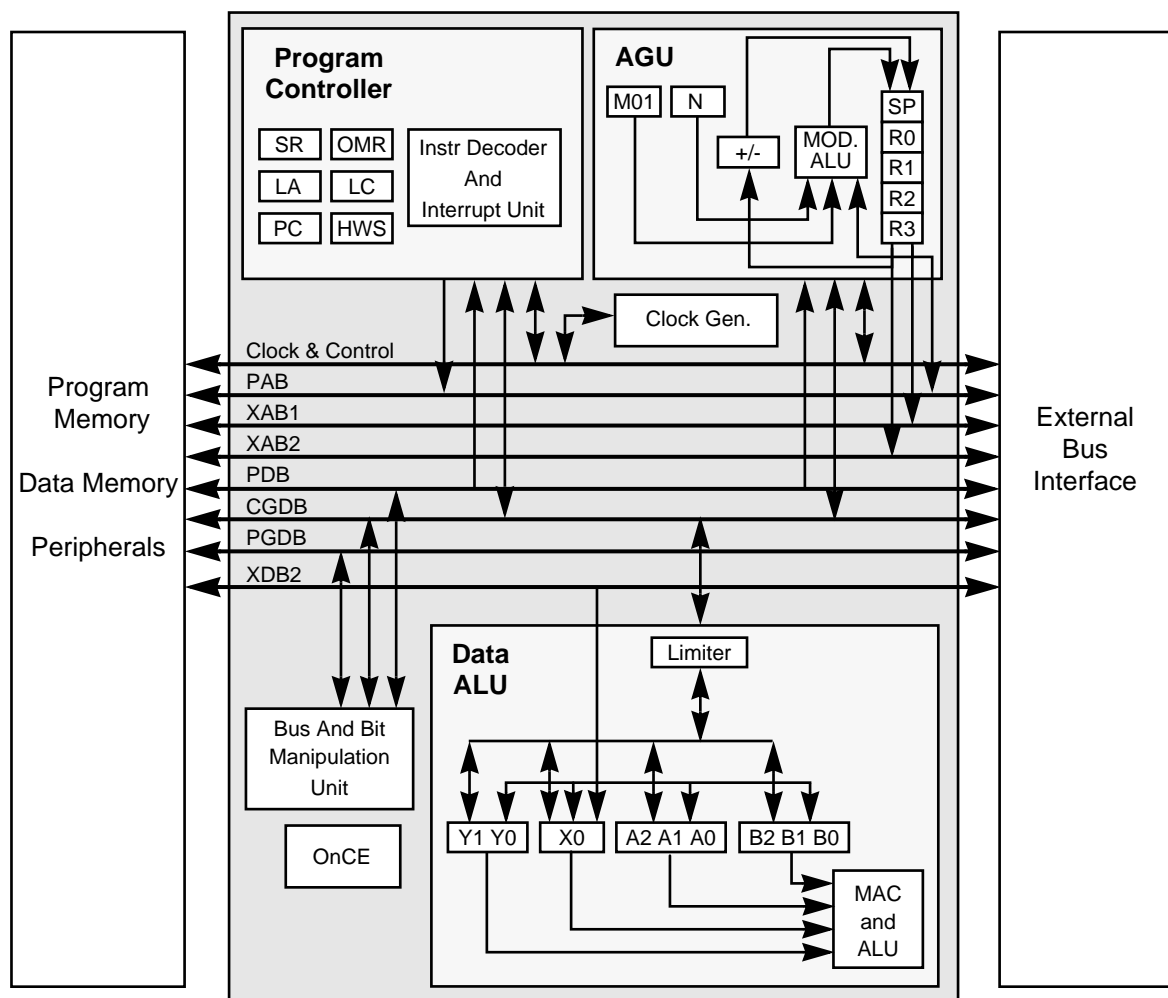


Figure 1-1 DSP56800 Core Block Diagram

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



AA0006

Figure 1-2 DSP56800 Bus Block Diagram

It is possible in a single instruction cycle for the Program Controller to be fetching a first instruction, the Address Generation Unit to generate two addresses for a second instruction, and the Data ALU to perform a multiply in a third instruction. In a similar manner, the bit manipulation unit can perform an operation of the third instruction described above instead of the multiplication in the Data ALU. The

architecture is pipelined to take advantage of the parallel units and significantly decrease the execution time of each instruction.

1.4.1 Data ALU

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

- Three 16-bit input registers
- Two 32-bit accumulator register
- Two 4-bit accumulator extension register
- Limiter
- Multi-bit shifting unit
- Parallel, single cycle, non-pipelined Multiply-Accumulator (MAC) unit.

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

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

Arithmetic operations are done using 2s complement fractional or integer arithmetic. Support is also provided for unsigned and multi-precision arithmetic.

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

1.4.2 Address Generation Unit

The Address Generation Unit performs all of the effective address calculations and address storage necessary to address data operands in memory. This unit operates in parallel with other chip resources to minimize address generation overhead. It contains two ALUs, allowing the generation of up to two 16-bit addresses every instruction cycle—one for either the XAB1 or PAB bus and one for the XAB2 bus. The ALU can directly address 65,536 locations on the XAB1 or XAB2 bus and 65,536 locations on the PAB bus, for a total capability of 131,072 16-bit data words. Hooks are provided on the DSP56800 core to expand this address space. Its arithmetic unit can perform linear and modulo arithmetic.

1.4.3 Program Controller and Hardware Looping Unit

The program controller performs instruction prefetch, instruction decoding, hardware loop control and interrupt (exception) processing. Instruction execution is carried out in other core units such as the Data ALU or Address Generation Unit. The program controller consists of a program counter unit, hardware looping control logic, interrupt control logic, and status and control registers.

Two mode and interrupt control pins provide input to the program interrupt controller. The Mode Select A/External Interrupt Request A(MODA/ $\overline{\text{IRQA}}$) and Mode Select B/External Interrupt Request B (MODB/ $\overline{\text{IRQB}}$) pins select the DSP56L811 operating mode and receive interrupt requests from external sources.

The $\overline{\text{RESET}}$ pin resets the DSP56L811. When it is asserted, it initializes the chip and places it in the reset state. When the $\overline{\text{RESET}}$ pin is deasserted, the DSP56L811 assumes the operating mode indicated by the MODA and MODB pins.

1.4.4 Address and Data Buses

Addresses are provided to the internal X Data Memory on two unidirectional 16-bit buses, X Address Bus One (XAB1) and X Address Bus Two (XAB2). Program memory addresses are provided on the unidirectional 19-bit Program Address Bus (PAB). Note that XAB1 can provide addresses for accessing both internal and external memory, whereas XAB2 can only provide addresses for accessing internal read-only memory. The External Address Bus (EAB) provides addresses for external memory.

Data movement on the DSP56L811 occurs over three bidirectional 16-bit buses—the Core Global Data Bus (CGDB), the Program Data Bus (PDB) and the Peripheral Data Bus (PGDB), and also over one unidirectional 16-bit bus, the X Data Bus Two (XDB2). Data transfer between the Data ALU and the X Data Memory occurs over the CGDB bus when one memory access is performed, over the CGDB bus and the XDB2 Data Bus when two simultaneous memory reads are performed. All other data transfers to core blocks occur over the CGDB bus, and all transfers to and from peripherals occur over the PGDB bus. Instruction word fetches occur simultaneously over the PDB bus. The External Data Bus (EDB) provides bi-directional access to external data memory.

The bus structure supports general register to register, register to memory, and memory to register, and can transfer up to three 16-bit words in the same instruction cycle. Transfers between buses are accomplished in the Bus and Bit Manipulation Unit. **Table 1-2** lists the address and data buses for the DSP56800 core.

Table 1-2 Data Buses

Bus	Bus Name	Bus width, direction, and use
XAB1	X Address Bus 1	16-bit, unidirectional, internal and external memory address
XAB2	X Address Bus 2	16-bit, unidirectional, internal memory address
PAB	Program Address Bus	19-bit, unidirectional, internal memory address
EAB	External Address Bus	16-bit, unidirectional, external memory address
CGDB	Core Global Data Bus	16-bit, bi-directional, internal data movement
PDB	Program Data Bus	16-bit, bi-directional, instruction word fetches
PGDB	Peripheral Data Bus	16-bit, bi-directional, internal data movement
XDB2	X Data Bus 2	16-bit, unidirectional, internal data movement
EDB	External Data Bus	16-bit, bi-directional, external data movement

1.4.5 Bus and Bit Manipulation Unit

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

DSP56L811 Architecture Overview

The bit manipulation unit performs bitfield manipulations on X memory words, peripheral registers, and registers on the DSP56800 core. It is capable of testing, setting, clearing, or inverting any bits specified in a 16-bit mask. For branch-on-bitfield instructions, this unit tests bits on the upper or lower byte of a 16-bit word. In other words, the mask tests a maximum of 8 bits at a time.

As a general rule, when reading any register less than 16 bits wide, unused bits are read as 0. Reserved and unused bits should always be written with 0 to insure future compatibility.

1.4.6 On-Chip Emulation (OnCE) Unit

The on-chip emulation (OnCE) unit allows the user to interact in a debug environment with the DSP56800 core and its peripherals non-intrusively. Its capabilities include examining registers, memory, or on-chip peripherals; setting breakpoints in memory; and stepping or tracing instructions. It provides simple, inexpensive, and speed independent access to the DSP56800 core for sophisticated debugging and economical system development. The JTAG port allows access to the OnCE module and through the DSP56L811 to its target system, retaining debug control without sacrificing other user accessible on-chip resources. This technique eliminates the costly cabling and the access to processor pins required by traditional emulator systems.

1.5 DSP56L811 ARCHITECTURE OVERVIEW

The DSP56L811 consists of the DSP56800 core, program and data memory, and peripherals useful for embedded control applications. **Figure 1-3** shows a block diagram of the DSP56L811 chip.

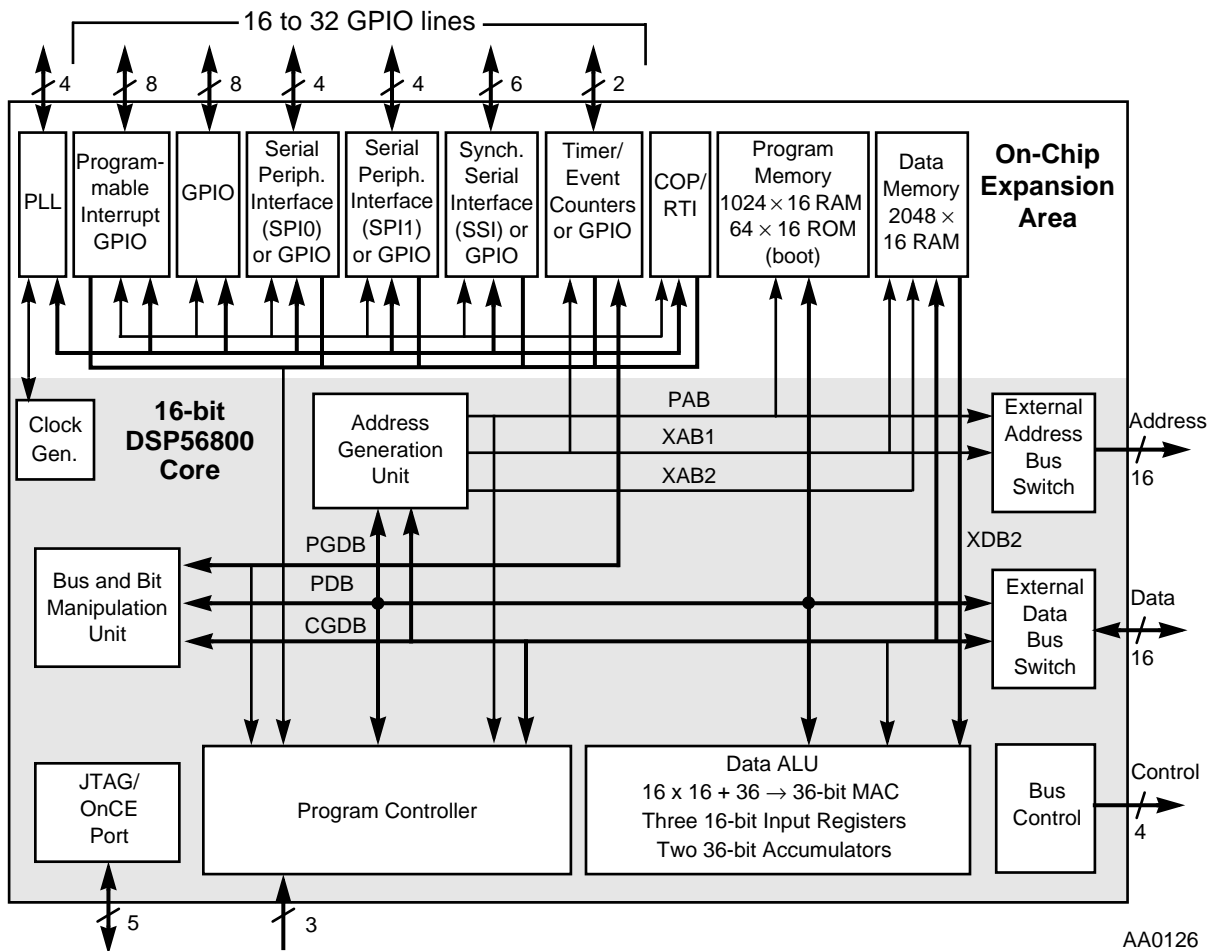


Figure 1-3 DSP56L811 Functional Block Diagram

1.5.1 DSP56L811 Peripheral Blocks

The DSP56L811 provides the following peripheral blocks:

- Program RAM
- Data RAM
- External Memory Interface
- General Purpose I/O Module
- Programmable I/O Module

DSP56L811 Architecture Overview

- Serial Peripheral Interface (two provided)
- Synchronous Serial Interface
- General-Purpose Timer Module
- On-chip Clock Synthesis Block (PLL)
- COP/RTI
- JTAG/OnCE Port

1.5.1.1 On-chip Memory

The DSP56L811 uses a Harvard architecture, which provides independent data and program memory. On-chip RAM is provided for both the X data (2K x 16-bit) and program (P) memory (1K x 16-bit), and both the program and data memories can be expanded off-chip. A 64 x 16-bit bootstrap ROM is also provided on-chip.

1.5.1.2 External Memory Interface

The DSP56L811 provides a port for external memory interfacing, also known as Port A. This port provides a total of 36 pins—16 pins for an external address bus, 16 pins for an external data bus, and four pins for bus control.

1.5.1.3 General Purpose I/O Module

A dedicated general-purpose input-output (GPIO) port, also known as Port B, provides 16 programmable I/O pins. This port is configured so that it is possible to generate interrupts when a transition is detected on any of its lower eight pins.

1.5.1.4 Programmable I/O Module

Port C provides 16 multiplexed general-purpose programmable I/O pins. Each pin can be individually selected as a GPIO pin, or allocated to on-chip peripherals—the general-purpose timer module, two Serial Peripheral Interfaces (SPI), and a Synchronous Serial Interface (SSI). Unlike the GPIO pins on Port B, the Port C pins cannot be configured to provide GPIO interrupts, but interrupts are available for the timer module, the two SPI ports, and the SSI port on Port C.

1.5.1.5 Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is an independent serial communications subsystem that allows the DSP56L811 to communicate synchronously with peripheral devices such as LCD display drivers, A/D subsystems, and MCU microprocessors. The SPI is also capable of inter-processor communication in a multiple master system. The SPI system can be configured as either a master or a slave device with high data rates. The SPI works in a demand-driven mode. In master mode, a transfer is initiated when data is written to the SPI Data Register. In slave mode, a transfer is initiated by the reception of a clock signal. Two separate SPIs are implemented on Port C.

1.5.1.6 Synchronous Serial Interface

The Synchronous Serial Interface (SSI) is a full-duplex serial port that allows the DSP56L811 to communicate with a variety of multiple serial devices, including industry-standard codecs, other DSPs, microprocessors, and peripherals that implement the Motorola SPI interface. It is typically used to transfer samples in a periodic manner. The SSI interface consists of independent transmitter and receiver sections with independent clock generation and frame synchronization. The SSI is implemented on Port C.

1.5.1.7 General-Purpose Timer Module

The timer module provides three independently programmable 16-bit timer/event counters. All three timer/counters can be clocked with clocks coming from one of three internal sources, including one of the other timers. In addition, the counters can be clocked with external clocking from the Timer Input / Output pins (TIO01 or TIO2) on Port C to count external events when configured as inputs. The same pins can be used as a timer pulse or for timer clock generation when used as outputs. The timer/event counters can be used to either interrupt the DSP56L811 or to signal an external device at periodic intervals. The timer I/O pins are implemented as part of Port C.

1.5.1.8 On-chip Clock Synthesis Block

The clock synthesis module generates the clocking for the DSP56L811. It generates three different clocks used by the DSP56800 core and DSP56L811 peripherals. It contains a phase-locked loop (PLL) that can multiply up the frequency or can be bypassed, as well as a prescaler/divider used to distribute clocks to peripherals and to lower power consumption on the DSP56L811. It also selects which clock, if any, is routed to the CLK0 pin of the DSP56L811.

1.5.1.9 COP/RTI Module

The Computer Operating Properly (COP) and Real Time Interrupt (RTI) module provides two separate functions: a watchdog timer, and an interrupt generator. These two functions monitor processor activity and provide an automatic reset signal if a failure occurs. Both functions are contained in the same block because the input clock for both comes from a common clock divider.

1.5.1.10 JTAG/OnCE Port

The JTAG/OnCE port allows the user to insert the DSP56L811 into a target system while retaining debug control. The JTAG port provides board-level testing capability in conformance with the *IEEE 1149.1a-1993 IEEE Standard Test Access Port and Boundary Scan Architecture* specification defined by the Joint Test Action Group (JTAG). Five dedicated pins interface to a test access port (TAP) that contains a 16-state controller.

In addition, on-chip emulation of the DSP56L811 is provided via the OnCE port, a Motorola-designed module used in DSP chips to debug application software used with the chip. The port is a separate on-chip block that allows non-intrusive interaction with the DSP and is accessible through the JTAG interface. The OnCE port makes it possible to examine registers, memory or on-chip peripherals contents in a special debug environment. This avoids sacrificing any user accessible on-chip resources to perform debugging.

The *DSP56800 Family Manual (DSP56800FAM/AD)*, provides complete details on the OnCE port.

1.5.2 DSP56L811 Peripheral Interrupts

The peripherals on the DSP56L811 use the interrupt channels found on the DSP56800 core. Each peripheral has its own interrupt vector (often more than one interrupt vector for each peripheral) and can selectively be enabled or disabled via the Interrupt Priority Register (IPR) found on the DSP56800 core.

Section 3 provides complete details on interrupt vectors.

1.6 CODE DEVELOPMENT ON THE DSP56L811

The DSP56L811 instruction set, described in detail in the *DSP56800 Family Manual (DSP56800FAM/AD)*, provides assembly level programming for this product. This manual provides a number of samples of source code to demonstrate the programming of certain features. These examples should not be considered as complete examples of how to program the DSP56L811, but as samples. See the *DSP56800 Family Manual (DSP56800FAM/AD)* for more information on development hardware and software products for the DSP56L811, including a low-cost evaluation module that allows access to most DSP56L811 functions and peripherals.

Two mechanisms on the DSP56L811 aid code development—full access by all instructions to the external data bus, and On-Chip Emulation (OnCE) hooks. The first is useful when code is first being developed on a hardware platform. The second is useful in all phases of debugging, especially when a unit is near completion and in its final packaging. This section describes the first option. The OnCE interface is fully described in the *DSP56800 Family Manual (DSP56800FAM/AD)*.

Instructions on the DSP56L811 can be executed without regard for whether the instruction fetch is on-chip or off-chip, and whether any data access is on-chip or

off-chip. However, executing an instruction (including parallel moves) may require as many as three memory accesses. If more than one of these memory accesses occurs off-chip, an additional instruction cycle is required for every external access, because only one access to external memory can occur at a time. This is shown in **Example 1-2** and in the following discussion.

Example 1-2 On-Chip and Off-Chip Instruction Fetches

```
mac x0,y0,a x:(r0)+,y0 x:(r3)+,x0
```

- Case 1. Instruction located on-chip, both x() data accesses performed to on-chip memory
 - In this case, since all memories are located on-chip, no external accesses are performed and the instruction runs in one instruction cycle, correctly performing all three accesses to on-chip memory.
- Case 2. Instruction located off-chip, both x() data accesses performed to on-chip memory
 - In this case only one external memory access occurs off-chip. The instruction fetch occurs over the external bus and the data accesses are made to on-chip memory. The instruction still runs in one instruction cycle, correctly performing all three accesses.
- Case 3. Instruction located on-chip, one x() data access performed to off-chip memory
 - In this case only one external memory access occurs off-chip. The instruction fetch is done to on-chip memory, one data access occurs over the external bus, and one data access is done to on-chip memory. The instruction still runs in one instruction cycle, correctly performing all three accesses.
- Case 4. Instruction located off-chip, one x() data access performed to off-chip memory
 - In this case, two external memory access occur off-chip. The instruction fetch occurs over the external bus, followed by the external data access. A data access to internal memory also takes place. The instruction now runs in two instruction cycles, correctly performing all three accesses.

Case 4 is often used during code development to provide the best visibility. This feature allows a logic analyzer to be placed on the external bus during code development on target hardware so that all memory accesses are visible. Separate \overline{PS} and \overline{DS} pins are provided to indicate whether the access is to external program or data memory.

Note: In this mode, accesses to on-chip peripherals are not visible because these memory-mapped registers are on-chip.

An example of a system where all program and memory accesses are visible is shown in **Figure 1-4**.

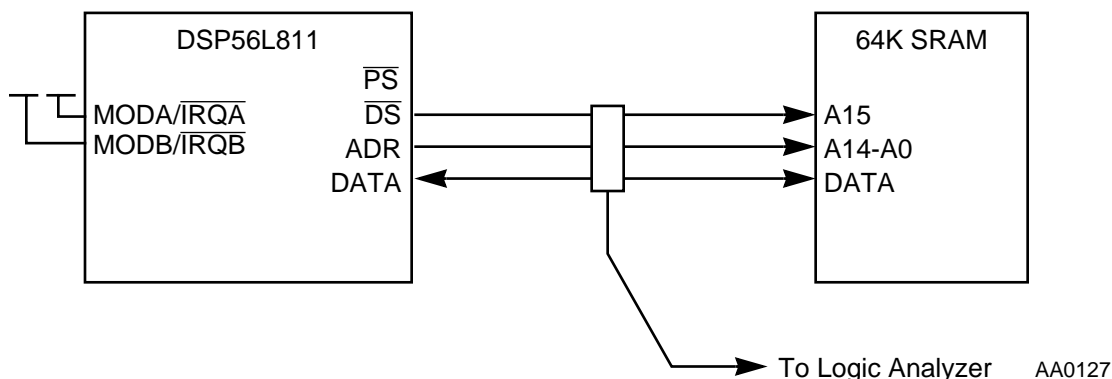


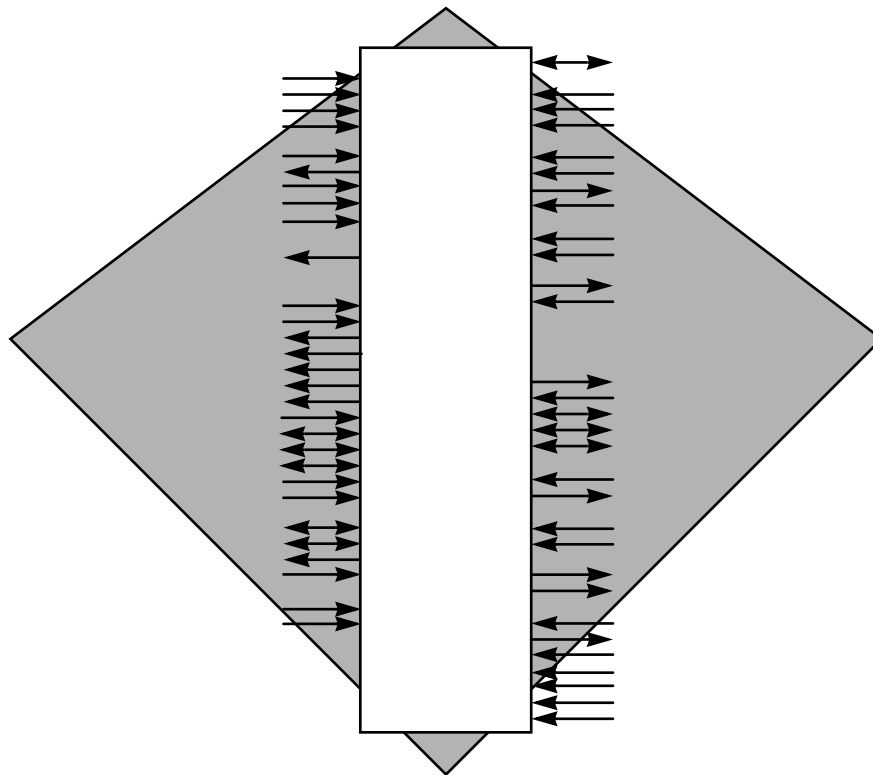
Figure 1-4 Example of Code Development with Visibility on All Memory Accesses

In this example, the DSP56L811 is programmed for Operating Mode 3, Development Mode, in the Operating Mode Register (OMR) to specify that all program accesses are done externally. See **3.2 DSP56L811 Memory Map Description** on page 3-3 for a detailed description of the OMR. Likewise, the EX bit (in the OMR) is set to specify that data accesses are done externally. An exception to this is the second access on any instruction which performs two reads in a single instruction. In this case, the second read using the R3 pointer always occurs to on-chip memory. If this is an issue, the instruction performing two data memory reads can be replaced by two instructions, each performing one of the two data memory accesses.



SECTION 2

PIN DESCRIPTIONS



2.1 INTRODUCTION..... 2-3

2.2 PIN DESCRIPTIONS..... 2-5

2.1 INTRODUCTION

The DSP56L811 is provided in a 100-pin TQFP package. **Figure 2-1** provides a pin-out diagram. For complete electrical specifications, refer to the *DSP56L811 Data Sheet (DSP56L811/D)*.

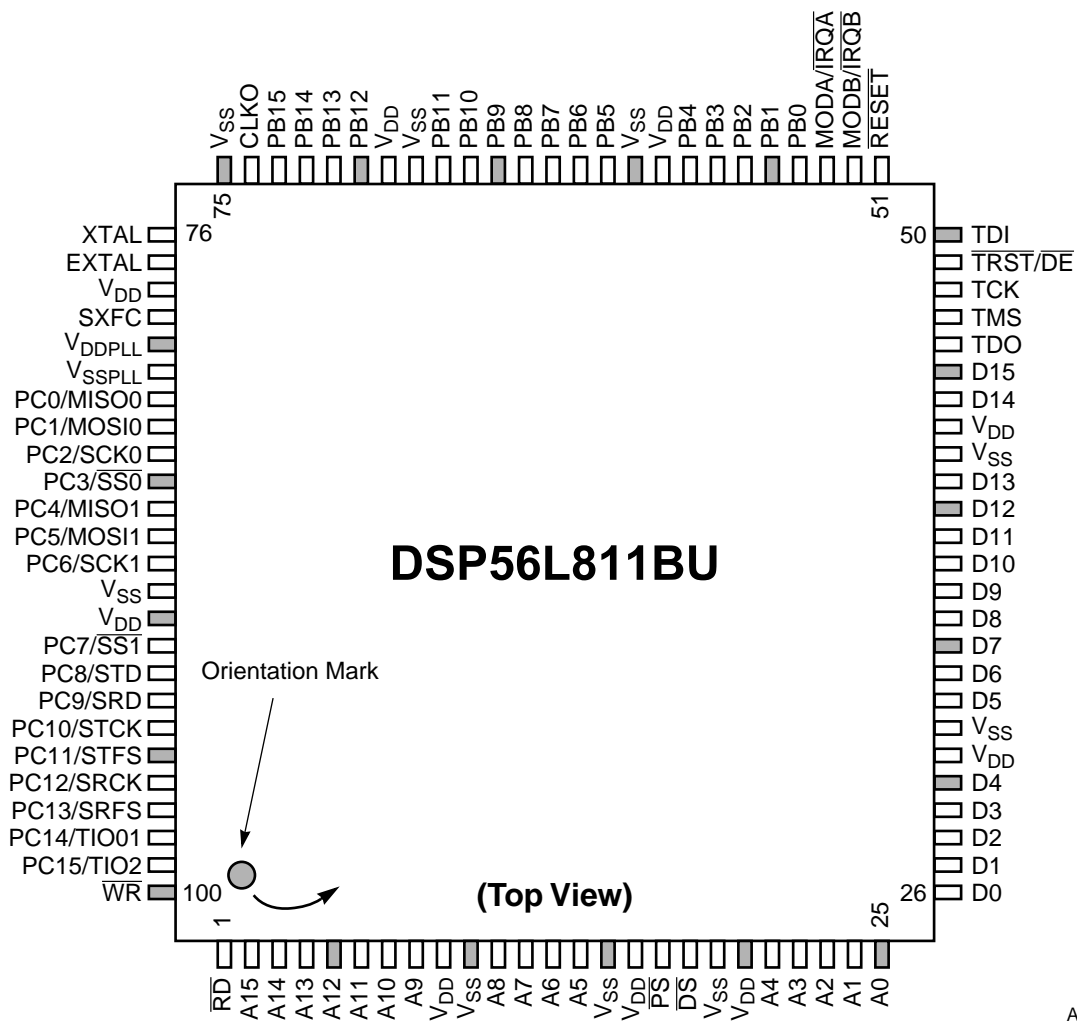
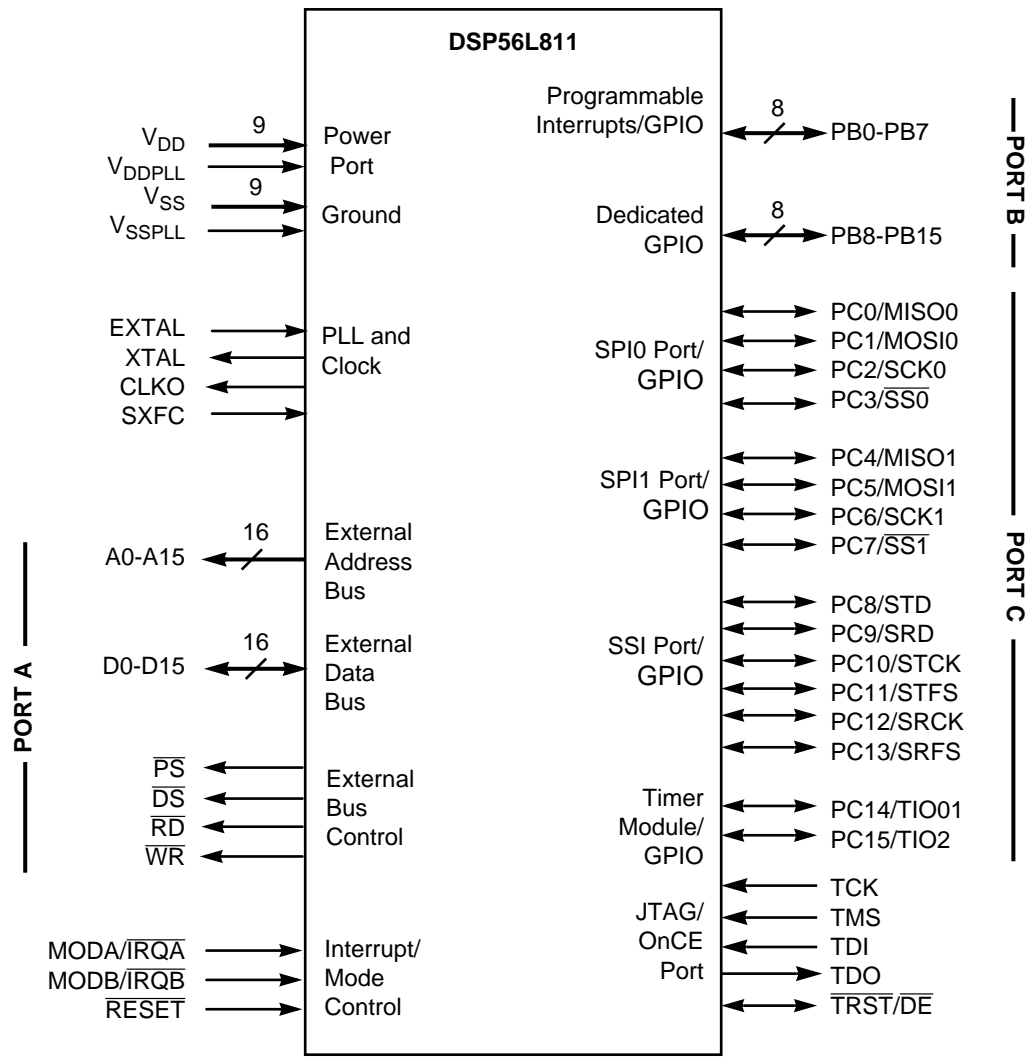


Figure 2-1 Top View of the DSP56L811 100-pin Plastic Thin Quad Flat Package

Figure 2-2 is a diagram of DSP56L811 pins by functional group.



AA0176

Figure 2-2 Functional Group Pin Allocations

The I/O signals are organized into the functional groups as summarized in **Table 2-1**.

Table 2-1 Functional Group Pin Allocations

Functional Group	Number of Pins	Detailed Description
Power (V_{DD} or V_{DDPLL})	10	Table 2-2
Ground (V_{SS} or V_{SSPLL})	10	Table 2-3
PLL and Clock	4	Table 2-4
Address Bus	16	Table 2-5
Data Bus	16	Table 2-6
Bus Control	4	Table 2-7
Interrupt and Mode Control	3	Table 2-8
Programmable Interrupt General Purpose Input/Output	8	Table 2-9
Dedicated General Purpose Input/Output	8	Table 2-10
Serial Peripheral Interface (SPI) Ports ¹	8	Table 2-11
Synchronous Serial Interface (SSI) Port ¹	6	Table 2-12
Timer Module ¹	2	Table 2-13
JTAG/On-chip emulation (OnCE)	5	Table 2-14
Note: 1. Alternately, general purpose I/O pins		

Note: All unused port pins configured as inputs should be properly terminated through a pull-up resistor. All power and ground pins should be connected to the appropriate low-impedance power and ground paths.

2.2 PIN DESCRIPTIONS

Table 2-2 Power Pins

Pin Name	Pin Description
V _{DD} (9)	Power —power pins
V _{DDPLL}	PLL Power —This pin supplies a quiet power source to the VCO to provide greater frequency stability.

Table 2-3 Ground Pins

Pin Name	Pin Description
V _{SS} (9)	GND —ground pins
V _{SSPLL}	PLL Ground —This pin supplies a quiet ground to the VCO to provide greater frequency stability.

Table 2-4 PLL and Clock Pins

Pin Name	Signal Type	State during Reset	Pin Description
EXTAL	Input	Input	External Clock/Crystal Input —This input should be connected to an external clock or to an external oscillator. After being squared, the input clock can be selected to provide the clock directly to the DSP core. The minimum instruction time is two input clock periods, broken up into four phases named T0, T1, T2, and T3. This input clock can also be selected as input clock for the on-chip PLL.
XTAL	Output	Chip-driven	Crystal Output —This output connects the internal crystal oscillator output to an external crystal. If an external clock is used, XTAL should not be connected.

Table 2-4 PLL and Clock Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
CLKO	Output	Chip-driven	Clock Output —This pin outputs a buffered clock signal. By programming two bits (CS1, CS0) inside the PLL control register PCR1, the user can select between outputting a squared version of the signal applied to EXTAL and a version of the DSP master clock at the output of the PLL. The clock frequency on this pin can also be disabled by programming the CS1, CS0 bits in the PLL control register, PCR1.
SXFC	Input	Input	External Filter Capacitor —This pin is used to add an external filter circuit to the phase locked loop.

Table 2-5 Address Bus Pins

Pin Names	Signal Type	State during Reset	Pin Description
A0-A15	Output	Tri-stated	Address Bus —Signals A0-A15 change in T0, and specify the address for an external program or data memory access. The value of the DRV bit in the bus control register (BCR) causes the address bus to retain the last external address (DRV = 1) or tri-stated (DRV = 0) during an internal access or in stop or wait mode.

Table 2-6 Data Bus Pins

Pin Names	Signal Type	State during Reset	Pin Description
D0-D15	Input/Output	Tri-stated	Data Bus —Read data is sampled in on the trailing edge of T2, while write data output is enabled on the leading edge of T2 and tri-stated on the leading edge of T0. D0-D15 are always tri-stated during internal access or in stop or wait mode.

Table 2-7 Bus Control Pins

Pin Name	Signal Type	State during Reset	Pin Description
\overline{PS}	Output	Pulled high internally	Program Memory Select — \overline{PS} is asserted low for external program memory access. If the external bus is not used during an instruction cycle (T0, T1, T2, T3), \overline{PS} is deasserted high in T0. During an internal access, in stop mode, or in wait mode, the value of the DRV bit in the Bus Control Register (BCR) determines whether the chip continues to drive \overline{PS} (DRV = 1) or tri-states \overline{PS} (DRV = 0).
\overline{DS}	Output	Pulled high internally	Data Memory Select — \overline{DS} is asserted low during T0 for external data memory access. If the external bus is not accessed during an instruction cycle (T0, T1, T2, T3), \overline{DS} is deasserted high in T0. During an internal access, in stop mode, or in wait mode, the value of the DRV bit in the BCR register determines whether the chip continues to drive \overline{DS} (DRV = 1) or tri-states \overline{DS} (DRV = 0).
\overline{WR}	Output	Pulled high internally	Write Enable — \overline{WR} is asserted low during external memory write cycles. When \overline{WR} is asserted low in T1, the data bus pins D0-D15 become outputs and the DSP puts data on the bus during the leading edge of T2. When \overline{WR} is deasserted high in T3, the external data is latched inside the external device. When \overline{WR} is asserted, it qualifies the A0-A15, \overline{PS} and \overline{DS} pins. \overline{WR} can be connected directly to the \overline{WE} pin of a static RAM. During an internal access, in stop mode, or in wait mode, the value of the DRV bit in the BCR register determines whether the chip continues to drive \overline{WR} (DRV = 1) or tri-states \overline{WR} (DRV = 0).
\overline{RD}	Output	Pulled high internally	Read Enable — \overline{RD} is asserted low during external memory read cycles. When \overline{RD} is asserted low during late T0/early T1, the data bus pins D0-D15 become inputs and an external device is enabled onto the DSP data bus. When \overline{RD} is deasserted high in T3, the external data is latched inside the DSP. When \overline{RD} is asserted, it qualifies the A0-A15 and \overline{PS} and \overline{DS} pins. \overline{RD} can be connected directly to the \overline{OE} pin of a static RAM or ROM. During an internal access, in stop mode, or in wait mode, the value of the DRV bit in the BCR register determines whether the chip continues to drive \overline{RD} (DRV = 1) or tri-states \overline{RD} (DRV = 0).

Table 2-8 Interrupt and Mode Control Pins

Pin Name	Signal Type	State during Reset	Pin Description
MODA/ $\overline{\text{IRQA}}$	Input	Input	<p>Mode Select A/External Interrupt Request A—This input has two functions:</p> <ol style="list-style-type: none"> 1. to select the initial chip operating mode and 2. after synchronization, to allow an external device to request a DSP interrupt <p>MODA is read and internally latched in the DSP when the processor exits the reset state. MODA and MODB select the initial chip operating mode. Several clock cycles (depending on PLL setup time) after leaving the reset state, the MODA pin changes to 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 which indicates that an external device is requesting service. It may be programmed to be level-sensitive or negative-edge sensitive. If level-sensitive triggering is selected, an external pull up resistor is required for wired-OR operation. If the processor is in the stop state and $\overline{\text{IRQA}}$ is asserted, the processor will exit the stop state.</p>
MODB/ $\overline{\text{IRQB}}$	Input	Input	<p>Mode Select B/External Interrupt Request B—This input has two functions:</p> <ol style="list-style-type: none"> 1. to select the initial chip operating mode and 2. after internal synchronization, to allow an external device to request a DSP interrupt <p>MODB is read and internally latched in the DSP when the processor exits the reset state. MODA and MODB select the initial chip operating mode. Several clock cycles (depending on PLL setup time) after leaving the reset state, the MODB pin changes to external interrupt request $\overline{\text{IRQB}}$. After reset, the chip operating mode can be changed by software. The $\overline{\text{IRQB}}$ input is an external interrupt request which indicates that an external device is requesting service. It may be programmed to be level sensitive or negative edge triggered. If level sensitive triggering is selected, an external pull up resistor is required for wired-OR operation.</p>

Table 2-8 Interrupt and Mode Control Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
RESET	Input	Input	Reset —This input is a direct hardware reset on the processor. When RESET is asserted low, 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 and MODB pins. The internal reset signal should be deasserted synchronous with the internal clocks.

Table 2-9 Programmable Interrupt GPIO Pins

Pin Name	Signal Type	State during Reset	Pin Description
PB0-PB7	Input or Output	Input	Port B GPIO —These eight pins can be programmed to generate an interrupt for any pin programmed as an input when there is a transition on that pin. Each pin can individually be configured to recognize a low-to-high or a high-to-low transition. In addition, these pins are dedicated General Purpose I/O (GPIO) pins which can individually be programmed as input or output pins. After reset, the default state is GPIO input.

Table 2-10 Dedicated General Purpose Input/Output (GPIO) Pins

Pin Name	Signal Type	State during Reset	Pin Description
PB8-PB15	Input or Output	Input	Port B GPIO —These eight pins are dedicated General Purpose I/O (GPIO) pins which can individually be programmed as input or output pins. After reset, the default state is GPIO input.

Table 2-11 Serial Peripheral Interface (SPI0 and SPI1) Pins

Pin Name	Signal Type	State during Reset	Pin Description
PC0/MISO0	Input/Output	Input	<p>SPI0 Master In/Slave Out (MISO0)—This serial data pin is an input to a master device and an output from a slave device. The MISO0 line of a slave device is placed in the high-impedance state if the slave device is not selected. The driver on this pin can be configured as an open-drain driver by the SPI's WOM bit when this pin is configured for SPI operation.</p> <p>Port C GPIO 0 (PC0)—This pin is a GPIO pin called PC0 when the SPI MISO0 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>
PC1/MOSI0	Input/Output	Input	<p>SPI0 Master Out/Slave In (MOSI0)—This serial data pin is an output from a master device and an input to a slave device. The master device places data on the MOSI0 line a half-cycle before the clock edge that the slave device uses to latch the data. The driver on this pin can be configured as an open-drain driver by the SPI's WOM bit when this pin is configured for SPI operation.</p> <p>Port C GPIO 1 (PC1)—This pin is a GPIO pin called PC1 when the SPI MOSI0 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>
PC2/SPCK0	Input/Output	Input	<p>SPI0 Serial Clock—This bidirectional pin provides a serial bit rate clock (SCK) for the SPI. This gated clock signal is an input to a slave device and is generated as an output by a master device. Slave devices ignore the SCK signal unless the slave select pin is active low. In both master and slave SPI devices, data is shifted on one edge of the SCK signal and is sampled on the opposite edge where data is stable. The driver on this pin can be configured as an open-drain driver by the SPI's WOM bit when this pin is configured for SPI operation.</p> <p>Port C GPIO 2 (PC2)—This pin is a GPIO pin called PC2 when the SPI SCK0 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-11 Serial Peripheral Interface (SPI0 and SPI1) Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
PC3/ $\overline{SS}0$	Input or Input/Output	Input	<p>SPI0 Slave Select—This input pin is used to select a slave device before a master device can exchange data with the slave device. \overline{SS} must be low before data transactions and must stay low for the duration of the transaction. The \overline{SS} line of the master must be held high.</p> <p>Port C GPIO 3 (PC3)—This pin is a GPIO pin called PC3 when the SPI $\overline{SS}0$ function is not being used.</p> <p>After reset, the default state is GPIO input.</p>
PC4/MISO1	Input/Output	Input	<p>SPI1 Master In/Slave Out—This serial data pin is an input to a master device and an output from a slave device. The MISO1 line of a slave device is placed in the high-impedance state if the slave device is not selected. The driver on this pin can be configured as an open-drain driver by the SPI's WOM bit when this pin is configured for SPI operation.</p> <p>Port C GPIO 4 (PC4)—This pin is a GPIO pin called PC4 when the SPI MISO1 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>
PC5/MOSI1	Input/Output	Input	<p>SPI1 Master Out/Slave In (MOSI1)—This serial data pin is an output from a master device and an input to a slave device. The master device places data on the MOSI0 line a half-cycle before the clock edge that the slave device uses to latch the data. The driver on this pin can be configured as an open-drain driver by the SPI's WOM bit when this pin is configured for SPI operation.</p> <p>Port C GPIO5 (PC5)—This pin is a GPIO pin called PC5 when the SPI MOSI1 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-11 Serial Peripheral Interface (SPI0 and SPI1) Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
PC6/SCK1	Input/Output	Input	<p>SPI1 Serial Clock—This bidirectional pin provides a serial bit rate clock for the SPI. This gated clock signal is an input to a slave device and is generated as an output by a master device. Slave devices ignore the SCK signal unless the slave select pin is active low. In both master and slave SPI devices, data is shifted on one edge of the SCK signal and is sampled on the opposite edge where data is stable. The driver on this pin can be configured as an open-drain driver by the SPI's WOM bit when this pin is configured for SPI operation.</p> <p>Port C GPIO 6 (PC6)—This pin is a GPIO pin called PC6 when the SPI SCK1 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>
PC7/ $\overline{SS}1$	Input or Input/Output	Input	<p>SPI1 Slave Select—This input pin is used to select a slave device before a master device can exchange data with the slave device. \overline{SS} must be low before data transactions and must stay low for the duration of the transaction. The \overline{SS} line of the master must be held high.</p> <p>Port C GPIO 7 (PC7)—This pin is a GPIO pin called PC7 when the SPI $\overline{SS}1$ function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-12 Synchronous Serial Interface (SSI) Pins

Pin Name	Signal Type	State during Reset	Pin Description
PC8/STD	Output or Input/Output	Input	<p>SSI Transmit Data (STD)—This output pin transmits serial data from the SSI Transmitter Shift Register.</p> <p>Port C GPIO 8 (PC8)—This pin is a GPIO pin called PC8 when the SSI STD function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-12 Synchronous Serial Interface (SSI) Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
PC9/SRD	Input or Input/Output	Input	<p>SSI Receive Data—This input pin receives serial data and transfers the data to the SSI Receive Shift Register.</p> <p>Port C GPIO 9 (PC9)—This pin is a GPIO pin called PC9 when the SSI SRD function is not being used.</p> <p>After reset, the default state is GPIO input.</p>
PC10/STCK	Input/Output	Input	<p>SSI Serial Transmit Clock—This bidirectional pin provides the serial bit rate clock for the Transmit section of the SSI interface. The clock signal can be continuous or gated and can be used by both the transmitter and receiver in synchronous mode.</p> <p>Port C GPIO 10 (PC10)—This pin is a GPIO pin called PC10 when the SSI STCK function is not being used.</p> <p>After reset, the default state is GPIO input.</p>
PC11/STFS	Input/Output	Input	<p>SSI Serial Transmit Frame Sync—This bidirectional pin is used by the Transmit section of the SSI serial interface as frame sync I/O or flag I/O. The STFS can be used by both the transmitter and receiver in synchronous mode. It is used to synchronize data transfer and can be an input or an output.</p> <p>Port C GPIO 11 (PC11)—This pin is a GPIO pin called PC11 when the SSI STFS function is not being used. This pin is not required by the SSI in gated clock mode.</p> <p>After reset, the default state is input.</p>
PC12/SRCK	Input/Output	Input	<p>SSI Serial Receive Clock—This bidirectional pin provides the serial bit rate clock for the Receive section of the SSI interface. The clock signal can be continuous or gated and can be used only by the receiver.</p> <p>Port C GPIO 12 (PC12)—This pin is a GPIO pin called PC12 when the SSI STD function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-12 Synchronous Serial Interface (SSI) Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
PC13/SRFS	Input/Output	Input	<p>SSI Serial Receive Frame Sync (SRFS)—This bidirectional pin is used by the Receive section of the SSI serial interface as frame sync I/O or flag I/O. The STFS can be used only by the receiver. It is used to synchronize data transfer and can be an input or an output.</p> <p>Port C GPIO 13 (PC13)—This pin is a GPIO pin called PC13 when the SSI SRFS function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-13 Timer Module Pins

Pin Name	Signal Type	State during Reset	Pin Description
PC14/TIO01	Input/Output	Input	<p>Timer 0 and Timer 1 Input/Output (TIO01)—This bidirectional pin receives external pulses to be counted by either the on-chip 16-bit Timer 0 or Timer 1 when configured as input and external clocking is selected. The pulses are internally synchronized to the DSP core internal clock. When configured as output, it generates pulses or toggles on a Timer 0 or Timer 1 overflow event. Selection of Timer 0 or Timer 1 is programmable through an internal register.</p> <p>Port C GPIO 14 (PC14)—This pin is a GPIO pin called PC14 when the Timer TIO01 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-13 Timer Module Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
PC15/TIO2	Input/Output	Input	<p>Timer 2 Input/Output (TIO2)—This bidirectional pin receives external pulses to be counted by the on-chip 16-bit Timer 2 when configured as input, and external clocking is selected. The pulses are internally synchronized to the DSP core internal clock. When configured as output, it generates pulses or toggles on a Timer 2 overflow event.</p> <p>Port C GPIO 15 (PC15)—This pin is a GPIO pin called PC15 when the Timer TIO2 function is not being used.</p> <p>After reset, the default state is GPIO input.</p>

Table 2-14 JTAG/On-Chip Emulation (OnCE) Pins

Pin Name	Signal Type	State during Reset	Pin Description
TCK	Input	Input	Test Clock Input —This input pin provides a gated clock to synchronize the test logic and shift serial data to the JTAG/OnCE port. The pin is connected internally to a pull-down resistor.
TMS	Input	Input	Test Mode Select Input —This input pin is used to sequence the JTAG test access port (TAP) controller's state machine. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
TDI	Input	Input	Test Data Input —This input pin provides a serial input data stream to the JTAG/OnCE port. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
TDO	Output	Tri-state	Test Data Output —This tri-statable output pin provides a serial output data stream from the JTAG/OnCE port. It is driven in the Shift-IR and Shift-DR controller states, and changes on the falling edge of TCK.

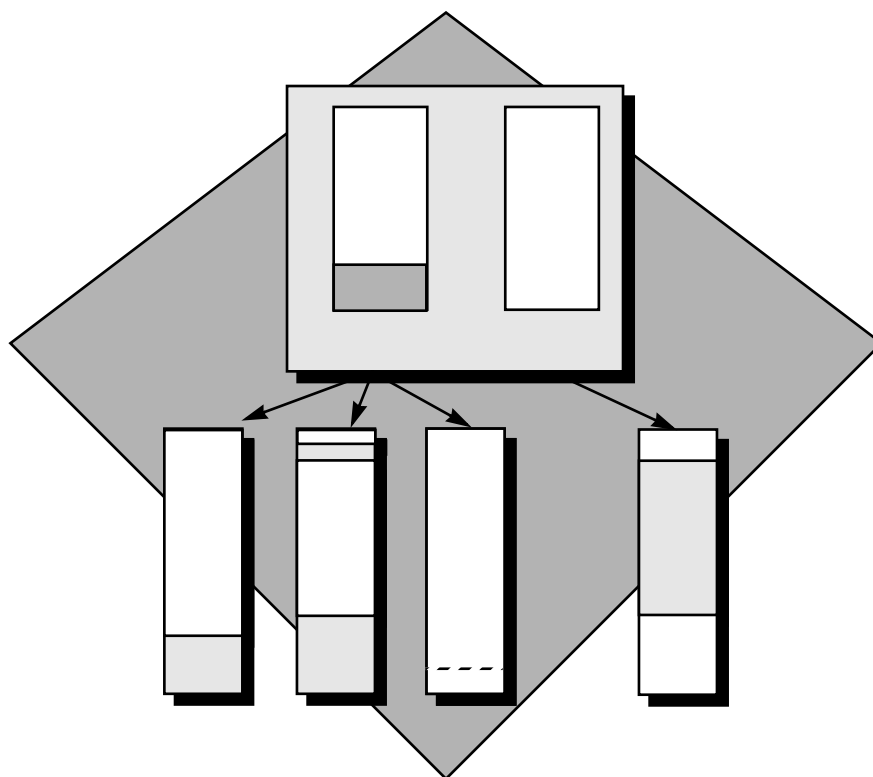
Table 2-14 JTAG/On-Chip Emulation (OnCE) Pins (Continued)

Pin Name	Signal Type	State during Reset	Pin Description
$\overline{\text{TRST}}/\overline{\text{DE}}$	Input or Output	Input	Test Reset/Debug Event —As an input, a low signal on this pin provides a reset signal to the JTAG TAP controller. It can also be programmed within the OnCE port as an output to provide a low pulse on recognized debug events; when configured as an output signal, the $\overline{\text{TRST}}$ input is disabled. This pin is connected internally to a pull-up resistor.



SECTION 3

MEMORY CONFIGURATION AND OPERATING MODES



3.1	INTRODUCTION	3-3
3.2	DSP56L811 MEMORY MAP DESCRIPTION	3-3
3.3	DSP56L811 OPERATING MODES	3-9
3.4	DSP56L811 BOOTSTRAPPING	3-11
3.5	EXECUTING PROGRAMS FROM XRAM	3-12
3.6	DSP56L811 RESET AND INTERRUPT VECTORS	3-15

3.1 INTRODUCTION

This section describes in detail the on-chip memories and the operating modes of the DSP56L811. In addition, the interrupt vectors, Interrupt Priority Register (IPR), and peripheral memory map are provided.

3.2 DSP56L811 MEMORY MAP DESCRIPTION

The DSP56L811 chip uses a Harvard memory architecture, in which two independent memory spaces, X data, and program (P), are provided. RAM is used for the on-chip data and program memory. The DSP56L811 has 2K words of on-chip data RAM. Also, 128 additional data memory locations are reserved for on-chip peripheral registers (X:\$FF80-FFFF). Both the program and data memories can be expanded off-chip. These are shown in **Figure 3-1**. A 64 x 16-bit bootstrap ROM is also provided.

The operating mode control bits (MA and MB) in the Operating Mode Register (OMR) control the program memory map and select the reset vector address.

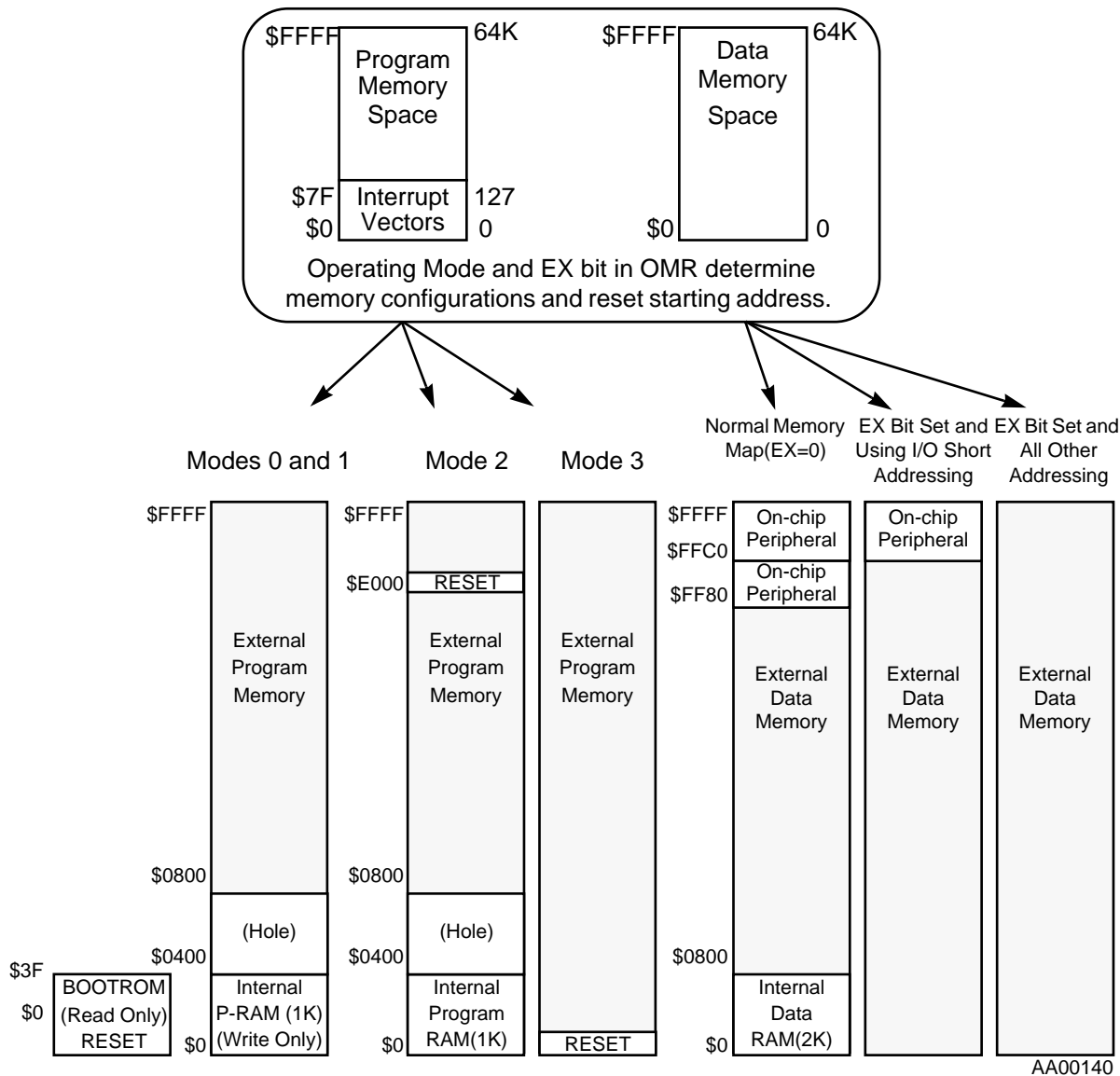
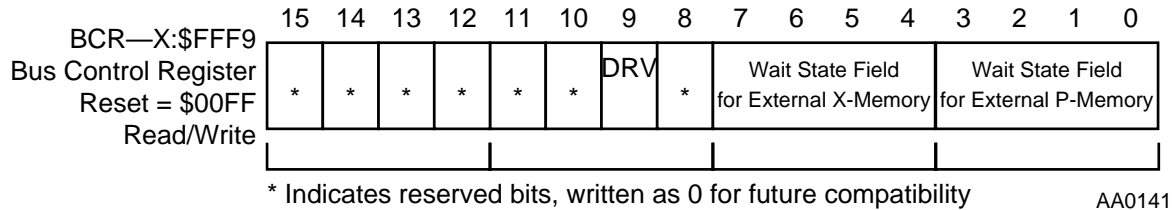


Figure 3-1 DSP56L811 On-chip Memory Map

Note: For DSP56800 core instructions that perform two reads from the X data memory in a single instruction, the *second* access using the R3 pointer always occurs to *on-chip* memory. In other words, the second read is Modulo 2048 on the DSP56L811 because there is 2K of on-chip data memory.

The X memory can be expanded off-chip for a total of 65,280 (65,536 - 256) addressable locations. The external data memory bus access time is controlled by four bits of an additional Bus Control Register (BCR) located at X:\$FFF9. This register

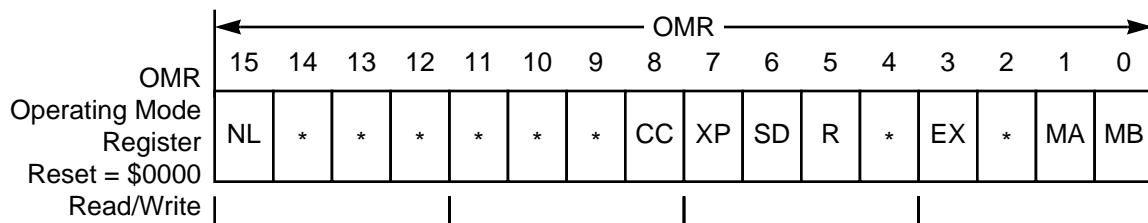
is shown in **Figure 3-2** and described in detail in **Section 7, External Memory Interface**.



AA0141

Figure 3-2 Bus Control Register Programming Model

The External X bit (EX, bit 3) of the OMR in the DSP56800 core determines the mapping of the X memory, as shown in **Figure 3-3**. Setting the EX bit to 1 completely disables the on-chip data memory and enables a full 64 K *external* memory map. The only exception to this rule is that if a MOVE or BIT FIELD instruction is used with the I/O short addressing mode, the EX bit is ignored. This allows on-chip peripheral registers to be accessed when the EX bit is set. When the EX bit is set, the access time to any data memory is controlled by the BCR register.



NL—Nested Looping
CC—Condition Codes
XP—X/P Memory
SD—Stop Delay
R—Rounding
EX—External X Memory
MA,MB—Operating Mode

* Indicates reserved bits, written as 0 for future compatibility

AA0142

Figure 3-3 Operating Mode Register Programming Model

Note: When the EX bit is set, only the upper 64 peripheral memory-mapped locations are accessible (X:\$FFC0 to X:\$FFFF) with the I/O short addressing mode. The other 64 memory-mapped locations (X:\$FF80 to X:\$FFC0) are not accessible when the EX bit is set. An access to these addresses results in an access to external memory.

DSP56L811 Memory Map Description

A complete description of the OMR register is provided in *DSP56800 Family Manual (DSP56800FAM/AD)*.

3.2.1 Program Memory Map

The DSP56L811 has 1K x 16 bits of available on-chip program RAM. It also contains a 64-word bootstrap ROM to assist in loading this on-chip program RAM. The first 128 locations of program memory are available for interrupt vectors, but many of these locations can also be used for program code, if not used for interrupt vectors. Program memory can be expanded off-chip for a total of 65,536 addressable locations. The external data bus access time is controlled by the BCR register, shown in **Figure 3-2**.

Figure 3-1 shows the on-chip memory map. The Bootstrap ROM is a 64 location ROM designed to initially load a program into the on-chip program RAM out of reset. The ROM is programmed at the factory with code that can load the program RAM from the external data bus or an on-chip peripheral. If the chip is configured in Modes 0 or 1, upon reset the chip begins executing the code from the Bootstrap ROM at location \$0.

The program memory map contains a non-accessible hole from 1K to 2K (\$0400 to \$07FF) in Modes 0, 1, and 2. In Mode 3, these locations are accessed as external memory. Any access to these addresses does not access external memory, except in Mode 3. These locations should not be used by an application except when Mode 3 is used exclusively. External program memory starts at location P:\$0800 (P:2048) for Modes 0, 1, and 2. When Mode 3 is selected, the complete 64 K words of program memory are external. These operating modes are described in the following subsection.

Appendix A provides more information on the Bootstrap ROM, including a listing of its contents.

3.2.2 On-Chip Peripheral Memory Map

Table 3-1 shows the on-chip memory mapped I/O registers on the DSP56L811.

Table 3-1 DSP56L811 I/O and On-Chip Peripheral Memory Map

X:\$FFFF	Reserved for On-Chip Emulation (OnCE)
X:\$FFFE	(reserved)
X:\$FFFD	(reserved)
X:\$FFFC	(reserved)
X:\$FFFB	IPR—Interrupt Priority Register
X:\$FFFA	(reserved)
X:\$FFF9	BCR—Bus Control Register (Port A)
X:\$FFF8	(reserved)
X:\$FFF7	(reserved)
X:\$FFF6	(reserved)
X:\$FFF5	(reserved)
X:\$FFF4	(reserved)
X:\$FFF3	PCR1—PLL Control Register 1
X:\$FFF2	PCR0—PLL Control Register 0
X:\$FFF1	COPCTL—COP/RTI Control Register
X:\$FFF0	COPCNT—COP/RTI Count Register (read only) COPRST—COP Reset Register (write only)
X:\$FFEF	PCD—Port C Data Register
X:\$FFEE	PCDDR—Port C Data Direction Register
X:\$FFED	PCC—Port C Control Register
X:\$FFEC	PBD—Port B Data Register
X:\$FFEB	PBDDR—Port B Data Direction Register
X:\$FFEA	PBINT—Port B Interrupt Register
X:\$FFE9	(reserved)
X:\$FFE8	(reserved)
X:\$FFE7	(reserved)
X:\$FFE6	SPCR1—SPI1 Control Register
X:\$FFE5	SPSR1—SPI1 Status Register
X:\$FFE4	SPDR1—SPI1 Data Register
X:\$FFE3	(reserved)
X:\$FFE2	SPCR0—SPI0 Control Register
X:\$FFE1	SPSR0—SPI0 Status Register
X:\$FFE0	SPDR0—SPI0 Data Register

DSP56L811 Memory Map Description

Table 3-1 DSP56L811 I/O and On-Chip Peripheral Memory Map (Continued)

X:\$FFDF	TCR01—Timer 0 and 1 Control Register
X:\$FFDE	TPR0—Timer 0 Preload Register
X:\$FFDD	TCT0—Timer 0 Count Register
X:\$FFDC	TPR1—Timer 1 Preload Register
X:\$FFDB	TCT1—Timer 1 Count Register
X:\$FFDA	TCR2—Timer 2 Control Register
X:\$FFD9	TPR2—Timer 2 Preload Register
X:\$FFD8	TCT2—Timer 2 Count Register
X:\$FFD7	(reserved)
X:\$FFD6	(reserved)
X:\$FFD5	(reserved)
X:\$FFD4	SCRRX—SSI Receive Control Register
X:\$FFD3	SCRTX—SSI Transmit Control Register
X:\$FFD2	SCR2—SSI Control Register 2
X:\$FFD1	SSR—SSI Status Register (read only) STSR—SSI Time Slot Register (write only)
X:\$FFD0	SRX—SSI Receive Register (read only) STX—SSI Transmit Register (write only)
X:\$FFCF	(reserved)
X:\$FFCE	(reserved)
X:\$FFCD	(reserved)
X:\$FFCC	(reserved)
X:\$FFCB	(reserved)
X:\$FFCA	(reserved)
X:\$FFC9	(reserved)
X:\$FFC8	(reserved)
X:\$FFC7	(reserved)
X:\$FFC6	(reserved)
X:\$FFC5	(reserved)
X:\$FFC4	(reserved)
X:\$FFC3	(reserved)
X:\$FFC2	(reserved)
X:\$FFC1	(reserved)
X:\$FFC0	(reserved)

3.3 DSP56L811 OPERATING MODES

The DSP56L811 has four operating modes that determine the memory maps for program and data memories and the start-up procedure when the chip leaves the reset state. Operating modes can be selected either by applying the appropriate signals to the MODA and MODB pins during reset, or by writing to the OMR register and changing the MA and MB bits, as shown in **Table 3-2**.

Table 3-2 DSP56L811 Program RAM Chip Operating Modes

MB or MODB value	MA or MODA value	Chip operating mode	Reset vector	Program memory configuration
0	0	Mode 0 Bootstrap 0	BOOTROM P:\$0000 Boot from external bus	Internal Program RAM is write-only.
0	1	Mode 1 Bootstrap 1	BOOTROM P:\$0000 Boot from peripheral	Internal Program RAM is write-only.
1	0	Mode 2 Normal expanded	External Program RAM P:\$E000	Internal Program RAM is enabled.
1	1	Mode 3 Development	External Program RAM P:\$0000	Internal Program RAM is disabled.

The MODA and MODB pins are sampled as the DSP56L811 leaves the reset state, and the initial operating mode of the chip is set accordingly. After the reset state is exited, the MODA and MODB pins become interrupt pins, $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$. One of four initial operating modes is selected, based on the values detected on MODA and MODB:

- Mode 0 (bootstrap from external bus)
- Mode 1 (bootstrap from peripheral)
- Mode 2 (normal expanded)
- Mode 3 (development)

Chip operating modes can also be changed by writing to the operating mode bits MB and MA in the OMR register. Changing operating modes does not reset the DSP56L811. Interrupts should be disabled immediately before changing the OMR register to prevent an interrupt from going to the wrong memory location. Also, one no-operation (NOP) instruction should be included after changing the OMR register to allow for remapping to occur.

DSP56L811 Operating Modes

Note: Depending on the operating mode chosen, the Computer Operating Properly (COP) reset vector differs. The COP reset uses reset vector P:\$0002 for Modes 0, 1, and 3, and P:\$E002 for Mode 2.

3.3.1 Bootstrap Mode (Mode 0)

Mode 0 is one of the two bootstrapping modes, where the on-chip program RAM is loaded one location at a time from the external bus. The DSP56800 core executes the bootstrapping program located in the bootstrap ROM. Mode 0 can be entered by either grounding both mode pins (MODA and MODB) before resetting the chip, or by writing to the OMR register and changing the MA and MB bits. The memory maps and reset vectors for Modes 0 and 1 are identical. The reset vector location in Mode 0 is P:\$0000 in the Bootstrap ROM (P:\$0002 for COP timer reset).

3.3.2 Bootstrap Mode (Mode 1)

Mode 1 is the second of the two bootstrapping modes, where the on-chip program RAM is loaded one location at a time from a peripheral. The DSP56800 core executes the bootstrapping program located in the Bootstrap ROM. This mode can be entered by either grounding the MODB pin and pulling the MODA pin high before resetting the chip, or by writing to the OMR register and changing the MA and MB bits. The memory maps and reset vectors for Modes 0 and 1 are identical. The reset vector location in Mode 0 is P:\$0000 in the Bootstrap ROM (P:\$0002 for COP timer reset).

3.3.3 Normal Expanded Mode (Mode 2)

The normal expanded mode (Mode 2) can be entered either by pulling the MODB pin high and grounding the MODA pin before resetting the chip, or by writing to the OMR and changing the MA and MB bits. The memory maps for Modes 0, 1, and 2 are identical. The difference between the modes is the location of the reset vector in program memory and Mode 2 does not cause the bootstrap program to be executed. The reset vector location in Mode 2 is located in the external program memory space at location P:\$E000 (P:\$E002 for COP timer reset).

3.3.4 Development Mode (Mode 3)

The development mode is similar to the normal expanded mode except that internal program memory is disabled. All references to program memory space are directed to external program memory, which is accessed on the external data bus. This mode can be entered by either pulling both the MODA and MODB pins high, or by writing to the OMR register and changing the MA and MB bits. The reset vector location in Mode 3 is located in the external program memory space at location P:\$0000 (P:\$0002 for COP timer reset).

3.4 DSP56L811 BOOTSTRAPPING

Bootstrapping is the technique used to load the on-chip program RAM immediately after an application comes out of reset, and then begin executing this code. It requires a bootstrap ROM, and is applicable only to program-RAM-based parts.

When the DSP56L811 exits the reset state in Mode 0 or 1, the following occurs:

1. Control logic maps the bootstrap ROM into the program memory space starting at location P:\$0000. The 64 locations in the bootstrap ROM are read-only. Likewise, the control logic maps the internal program RAM into the program memory space starting at location P:\$0000, but these locations are write-only.
2. The DSP56800 core begins to execute the instructions it fetches from P:\$0000 in the bootstrap ROM, which contains the bootstrap program. This program examines the OMR register to determine whether to load the internal memory with values read through the external bus, or with values received through one of the DSP56L811 peripherals.
3. Once the program determines where to get values to load into the on-chip program RAM, the values are loaded one-by-one into the program RAM.
4. When the loading of the on-chip program RAM has completed, the bootstrap program terminates bootstrapping and enters operating Mode 2 (by writing to the OMR register), then jumps to the first location in the internal program RAM, P:\$0000. At this point, fetches to the bootstrap ROM are disabled and all fetches are from the on-chip program RAM.

The bootstrap modes may also be selected by writing the appropriate bits to the OMR register for operating Modes 0 or 1. This technique allows the user to reboot the internal program RAM in software.

3.5 EXECUTING PROGRAMS FROM XRAM

The DSP56L811 chip is designed with the ability to execute programs stored in the X data RAM block (XRAM). This is useful for PROM-based parts for code that is downloaded externally to the DSP56L811 from one of its peripherals. The capabilities of this mode are presented in the following paragraphs.

3.5.1 Description of the XRAM Execution Mode

The XRAM execution mode provides a means by which programs previously downloaded into XRAM can be executed from this RAM. This is most useful on parts where the program space (P) contains program ROM only. In this mode, program instructions and interrupt vectors are downloaded into XRAM, where they can be executed later in this mode. Instructions execute in this mode at the same speed as P-memory execution mode, and most of the DSP56L811 instruction set can be executed in this mode. One notable exception is that instructions that perform two reads from the X data memory are not allowed. However, instructions that perform one parallel move operation are still allowed in this mode.

Note: An extra instruction cycle is inserted on all instructions that write to X data memory.

Figure 3-4 shows the memory map in this mode.

Note: There is no reset vector, because a reset clears the XP bit in the OMR register, placing the chip in its normal mode of operation where program instructions are fetched from the program memory.

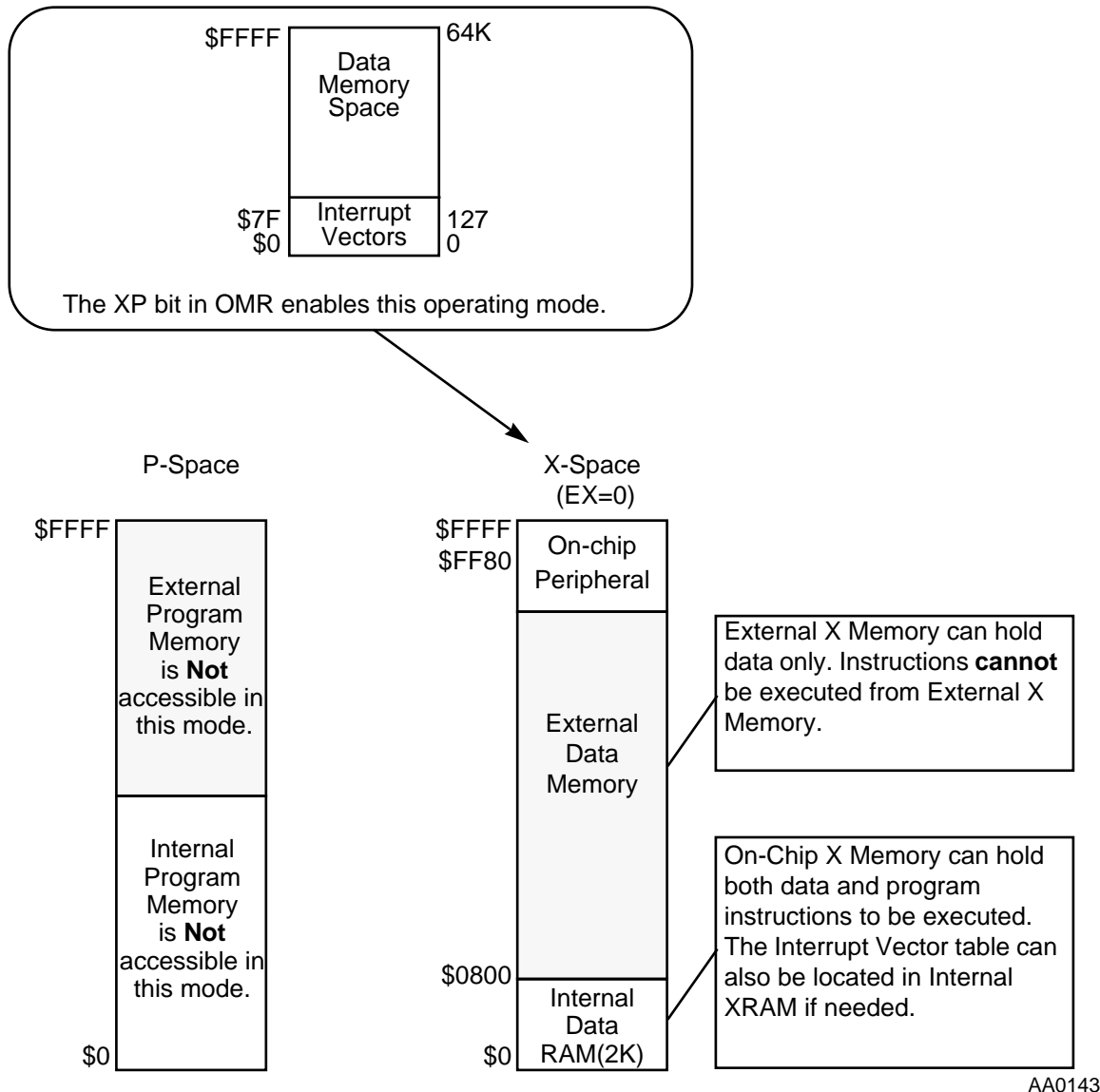


Figure 3-4 DSP56L811 Memory Map (XRAM Execution Mode)

It is still possible to access data located in off-chip X data memory in this mode, but not to execute instructions located in off-chip X data memory.

Note: The program will not operate correctly if an instruction is fetched at an address outside the on-chip XRAM space.

3.5.2 Interrupts in the XRAM Execution Mode

All interrupts are supported in this mode. The interrupt vector table is located at the beginning of the XRAM memory map, X:\$0. It is only necessary to place interrupt vectors for interrupts that can occur in this mode. The interrupt service routines must also be located in on-chip XRAM. Interrupts must be disabled when entering or exiting the XRAM Execution mode.

3.5.3 Entering the XRAM Execution Mode

In this case, the chip is initially operating in normal mode, where instructions are fetched from the program memory space, and where it is desired to begin executing instructions from the XRAM. When entering this mode, do the following:

1. Download the desired program into XRAM, including interrupt vectors and any necessary interrupt service routine.
2. Copy any constants located in program ROM into XRAM (if desired), because the program memory space cannot be accessed in XRAM execution mode.
3. Disable interrupts in the Status Register (SR). (See **Figure 3-5 Status Register Programming Model** on page 3-18.)
4. Set the XP bit in the Operating Mode Register (OMR) using a BFSET instruction.
5. Jump to the first instruction in the XRAM.
6. Re-enable interrupts from code in XRAM (if desired).

3.5.4 Exiting the XRAM Execution Mode

When the chip has finished executing instructions from XRAM and it is desired to begin executing instructions from the program memory space, the following sequence of operations is performed:

1. Disable interrupts in the SR register.
2. Clear the XP bit in the OMR register using a BFCLR instruction.
3. Jump to the return location in the program memory space.
4. Re-enable interrupts from code located in program memory space.

3.5.5 Restrictions in the XRAM Execution Mode

The following restrictions apply when executing programs from the X data memory:

- The EX bit in the OMR register must be set to 0.
- Programs must be entirely located in on-chip XRAM.
- Instructions that perform two reads from X data memory are not permitted.
- Instructions that perform accesses to the program memory space are not permitted.
- Interrupts must be disabled when entering or exiting the XRAM execution mode.
- Interrupt subroutines must be located in XRAM for any enabled interrupt.

3.6 DSP56L811 RESET AND INTERRUPT VECTORS

The interrupt vector map specifies the address to which the processor jumps when it recognizes an interrupt or encounters a reset condition. The instruction located at this address must be a JSR instruction for an interrupt, or a JSR instruction for a reset. The interrupt vector map for a given chip is specified by all possible interrupt sources on the DSP56800 core, as well as from the peripherals. No Interrupt Priority Level (IPL) is specified for hardware reset or for COP reset because these conditions reset the chip, and a reset takes precedence over all other interrupts.

Table 3-3 on page 3-16 provides the interrupt priority structure for the DSP56L811, including on-chip peripherals. **Table 3-4** lists the reset and interrupt vectors for the DSP56L811. A full description of interrupts is provided in the *DSP56800 Family Manual (DSP56800FAM/AD)*.

Note: In operating Mode 2, the hardware reset vector is located at address \$E000 and the COP reset vector is at \$E002. In all other modes, the hardware reset vector is at \$0000 and the COP reset vector is at \$0002.

Table 3-3 Interrupt Priority Structure

Priority	Exception	IPR Register Bits
Level 1 (Non-maskable)		
Highest	Hardware RESET	—
	COP Timer RESET	—
	Illegal Instruction Trap	—
	Hardware Stack Overflow	—
	OnCE Trap	—
Lower	SWI	—
Level 0 (Maskable)		
Higher	$\overline{\text{IRQA}}$ (External interrupt)	2, 1
	$\overline{\text{IRQB}}$ (External interrupt)	5, 4
	Channel 6 Peripheral Interrupt—SSI	9
	Channel 5 Peripheral Interrupt—Reserved	10
	Channel 4 Peripheral Interrupt—Timer Module	11
	Channel 3 Peripheral Interrupt—SPI1	12
	Channel 2 Peripheral Interrupt—SPI0	13
	Channel 1 Peripheral Interrupt—Realtime Timer	14
Lowest	Channel 0 Peripheral Interrupt—Port B GPIO	15

Table 3-4 Reset and Interrupt Vector Map

Interrupt Starting Address	Interrupt Priority Level	Interrupt Source
\$0000 / \$E000 *	—	Hardware RESET
\$0002 / \$E002 *	—	COP Timer RESET
\$0004	—	(Reserved)
\$0006	1	Illegal Instruction Trap
\$0008	1	Software Interrupt (SWI)
\$000A	1	Hardware Stack Overflow
\$000C	1	OnCE Trap
\$000E	1	(Reserved)
\$0010	0	$\overline{\text{IRQA}}$
\$0012	0	$\overline{\text{IRQB}}$

Table 3-4 Reset and Interrupt Vector Map (Continued)

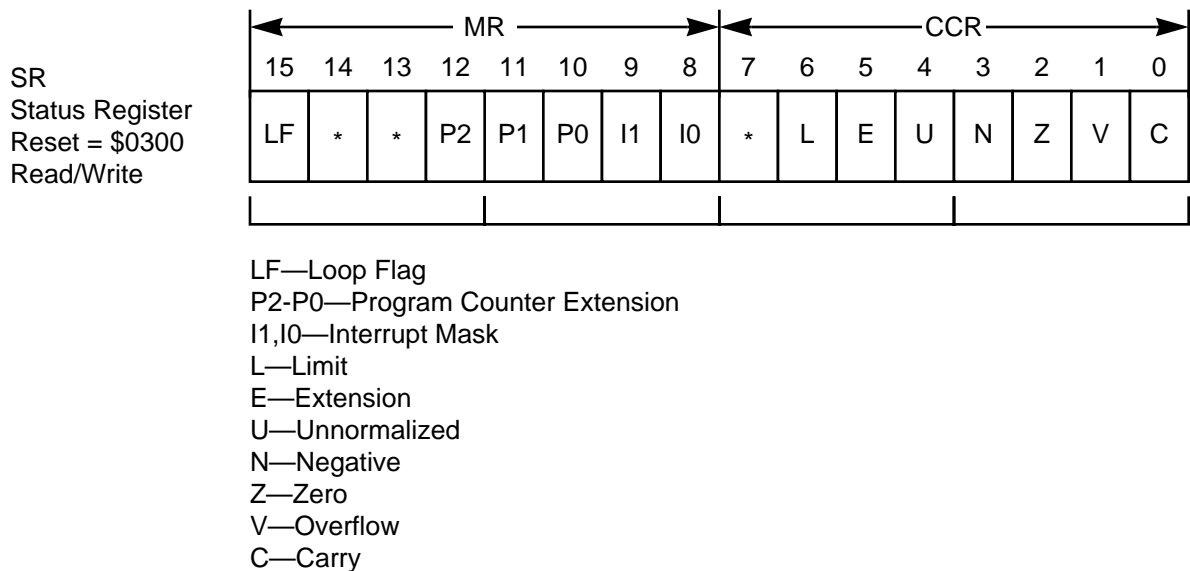
Interrupt Starting Address	Interrupt Priority Level	Interrupt Source
\$0014	0	Port B GPIO Interrupt
\$0016	0	Real-Time Interrupt
\$0018	0	Timer 0 Overflow
\$001A	0	Timer 1 Overflow
\$001C	0	Timer 2 Overflow
\$001E	0	(Reserved)
\$0020	0	SSI Receive Data w/ Exception Status
\$0022	0	SSI Receive Data
\$0024	0	SSI Transmit Data w/ Exception Status
\$0026	0	SSI Transmit Data
\$0028	0	SPI1 Serial System
\$002A	0	SPI0 Serial System
	.	(Reserved)
\$0040	0	(Reserved)
\$0042	0	(Reserved)
	.	(Reserved)
\$007C	0	(Reserved)
\$007E	0	(Reserved)

* Interrupt starting address when in Mode 2

3.6.1 DSP56L811 Status Register

The status register (SR) is a 16-bit register consisting of an 8-bit mode register (MR) and an 8-bit condition code register (CCR). The MR register is the high-order 8 bits of the SR; the CCR register is the low-order 8 bits. A full description of the SR register is provided in the *DSP56800 Family Manual (DSP56800FAM/AD)*. The programming model for the SR register is shown in **Figure 3-5**.

DSP56L811 Reset and Interrupt Vectors



* Indicates reserved bits, read as 0 and should be written with 0 for future compatibility AA0011

Figure 3-5 Status Register Programming Model

Within the SR register, the MR register is of special concern to DSP56L811 users, as it allows masking or enabling interrupts for the on-chip peripherals. On reset, the SR register is set to \$0300, which enables interrupts having IPL 1 (listed in Table 3-4) but masks interrupts with level IPL 0. All the on-chip peripheral interrupts have IPL 0. To enable these interrupts, set the I[1:0] bits in the SR register to 01. This should be done for all applications that use the on-chip peripherals. After enabling all the interrupts, selectively disable any unneeded interrupts by using the Interrupt Priority Register (IPR), described in DSP56L811 Interrupt Priority Register on page 3-19.

Table 3-5 shows the valid values to use for initializing the SR register, the values for the interrupt mask bits, and the interrupt masking.

Table 3-5 Interrupt Mask Bit Definition

Value of SR	I1	I0	Exceptions Permitted	Exceptions Masked
(Reserved)	0	0	(Reserved)	(Reserved)
\$0100	0	1	IPL 0, 1	None
(Reserved)	1	0	(Reserved)	(Reserved)
\$0300 (value at reset)	1	1	IPL 1	IPL 0

Note: Unless IPL 0 interrupts are enabled for on-chip peripheral interrupts in the SR register, setting the IPR register will have no effect.

For best results, use the following command to enable peripheral interrupts (IPL 0) in the SR register:

```
BFCLR #0200,SR
```

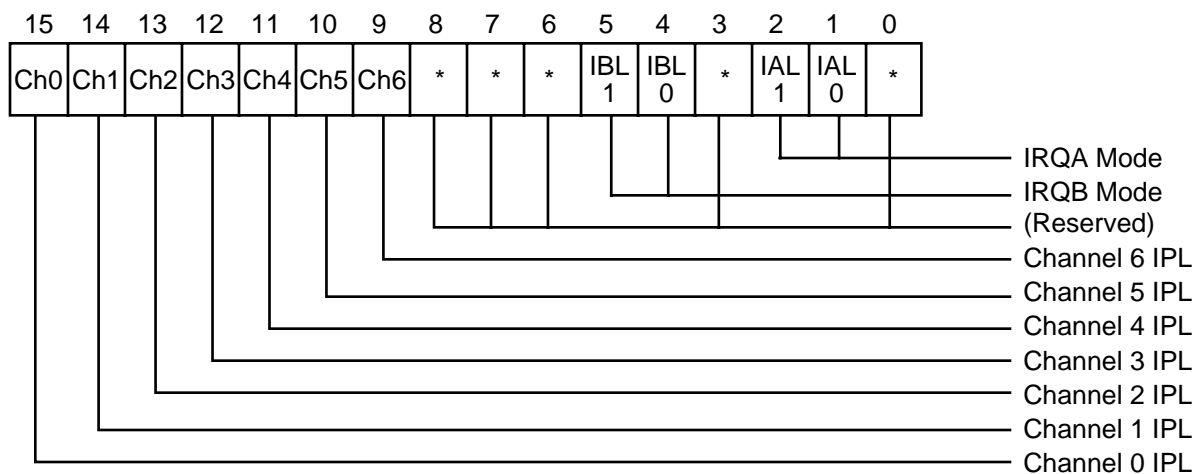
This command changes the I[1:0] bits from 11 to 01, clearing only the I1 bit and leaving all other bits in the SR register unaffected. If it is necessary to temporarily disable peripheral interrupts, issue the following command:

```
BFSET #0200,SR
```

This command changes the I[1:0] bits from 01 to 11, disabling all the peripheral interrupts. Afterwards, re-enable interrupts using the BFCLR #0200,SR command.

3.6.2 DSP56L811 Interrupt Priority Register

The interrupt arbiter on the DSP56800 core contains seven interrupt channels for use by peripherals, in addition to interrupts provided by the core. Peripheral interrupts are enabled or masked by writing to the IPR register after enabling them using the SR register. **Table 3-3** shows the interrupt priority order. **Figure 3-6** shows how this is programmed for the different peripherals on the DSP56L811.



* Indicates reserved bits, read as 0 and written with 0 for future compatibility

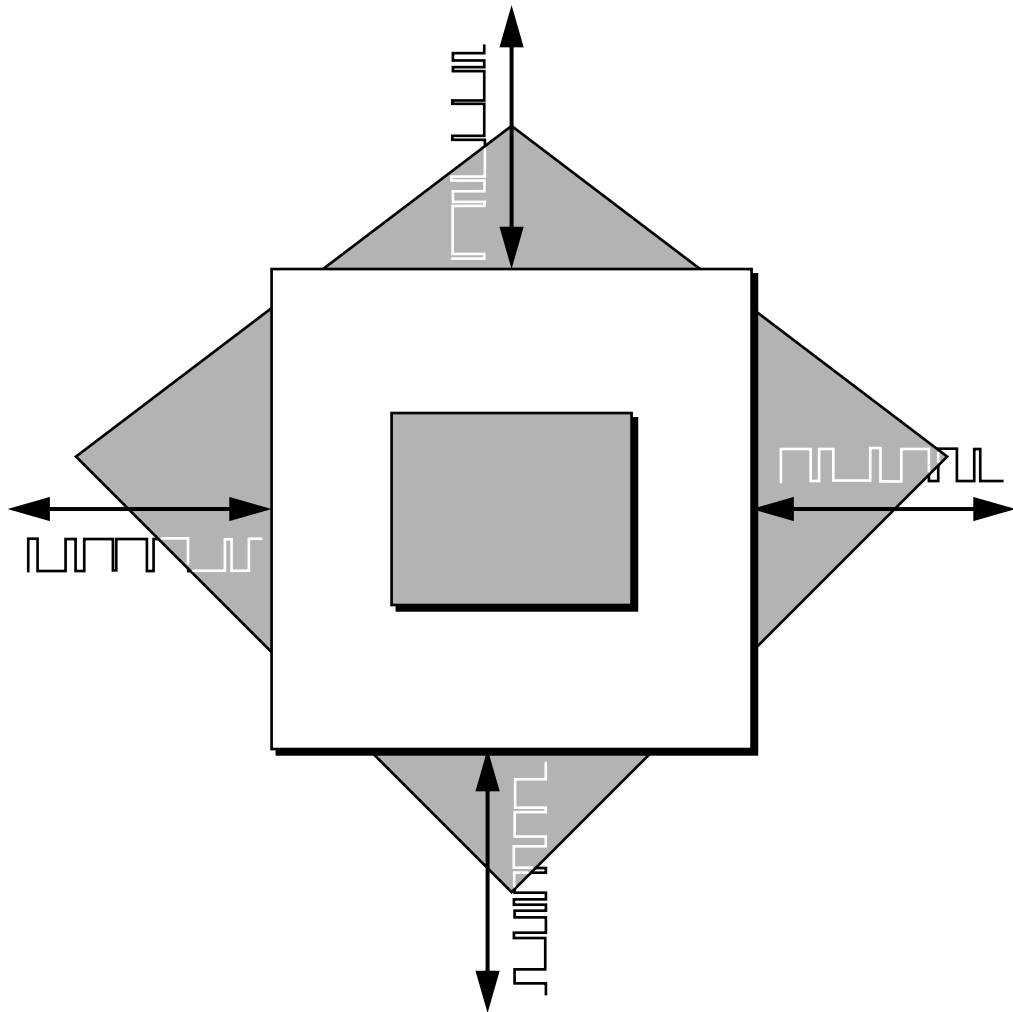
AA0057

Figure 3-6 DSP56L811 Interrupt Priority Register (IPR), X:\$FFFB



SECTION 4

EXTERNAL MEMORY INTERFACE



4.1	INTRODUCTION	4-3
4.2	EXTERNAL MEMORY PORT ARCHITECTURE	4-3
4.3	EXTERNAL MEMORY PORT DESCRIPTION	4-5

4.1 INTRODUCTION

The DSP56L811 provides a port for external memory interfacing. This section describes in detail the pins and programming specifics for the external memory port, also known as Port A. This port provides 16 pins for an external address bus, 16 pins for an external data bus, and four pins for bus control. Together, these 36 pins comprise the external memory port.

4.2 EXTERNAL MEMORY PORT ARCHITECTURE

Figure 4-1 on page 4-4 shows the general block diagram of the DSP56L811 I/O. It includes Ports A (described above), B, and C.

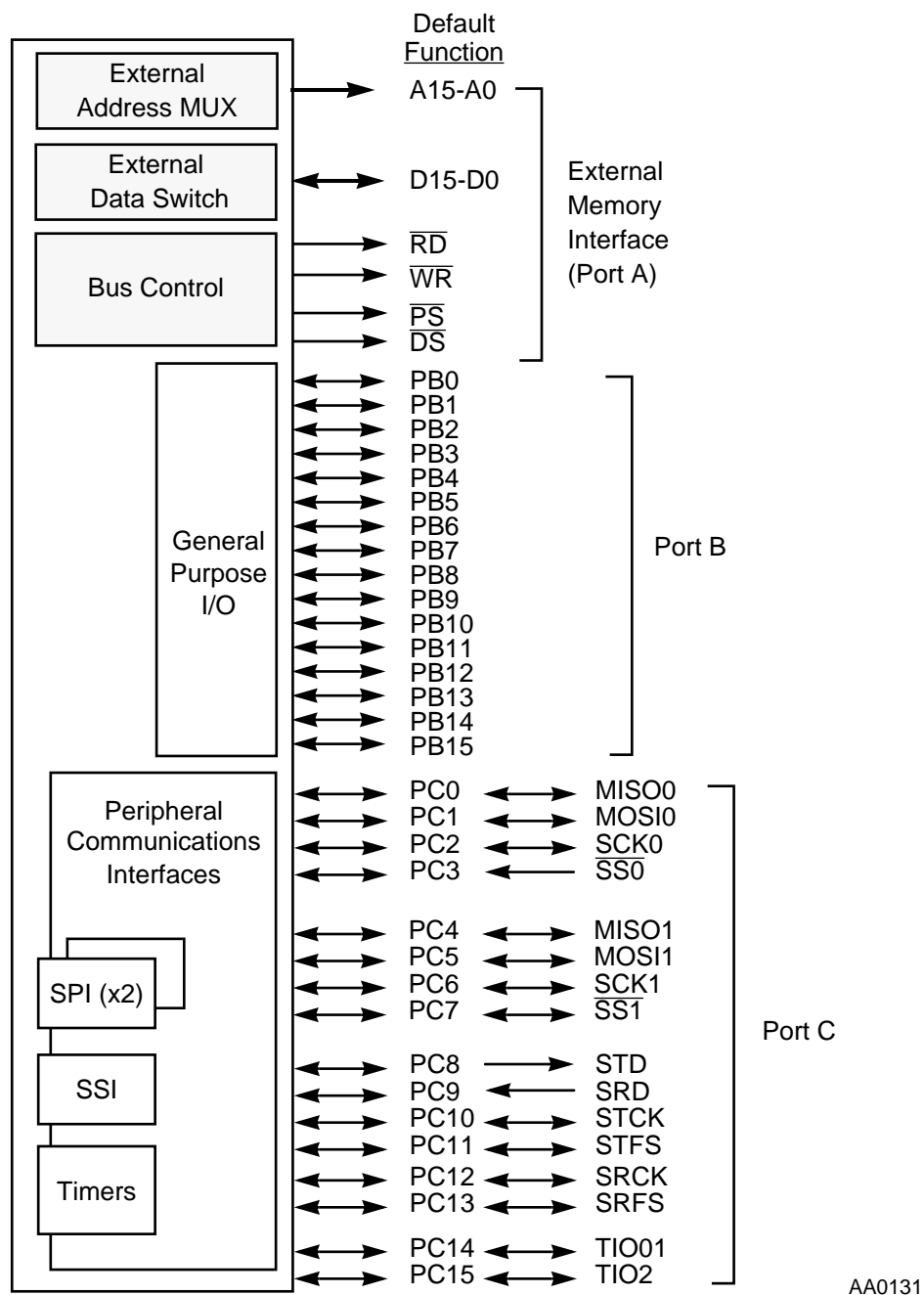


Figure 4-1 DSP56L811 Input/Output Block Diagram

4.3 EXTERNAL MEMORY PORT DESCRIPTION

The external memory port (also referred to as Port A) is the port through which all accesses to external memories and external memory-mapped peripherals are made. This port contains a 16-bit address bus, a 16-bit data bus, and four bus control pins for strobes.

External memory can be accessed at the maximum speed of the bus unit, or software wait states can be introduced when accessing slower memories or peripherals. Wait states are programmable using registers, and **Figure 4-3** and **Figure 4-4** on page 4-8 show examples of bus cycles with and without wait states.

4.3.1 External Memory Port Programming Model

The external memory port uses one programmable register for bus control, the Bus Control Register (BCR).

4.3.1.1 Bus Control Register (BCR)

The Bus Control Register (BCR), located at X:\$FFF9, is a 16-bit read/write register used for inserting software wait states on accesses to external program or data memory. Two 4-bit wait state fields are provided, each capable of specifying from 0 to 15 wait states. On processor reset, each wait state field is set to \$F so that 15 wait states are inserted, allowing slower memory to be used immediately after reset. All other BCR bits are cleared on processor reset.

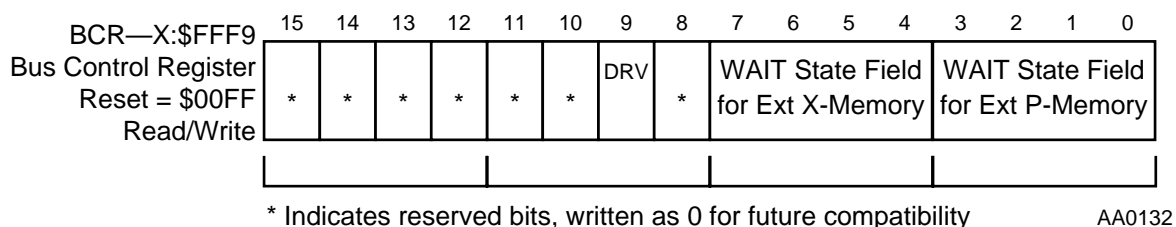


Figure 4-2 External Memory Port Programming Model

4.3.1.2 Drive (DRV) Bit 9

The Drive control bit (bit 9) is used to specify what occurs on the external memory port pins when no external access is performed—whether the pins remain driven or are placed in tri-state. **Table 4-3** on page 4-9 and **Table 4-4** on page 4-9 summarize the action of the DRV bit. The DRV bit is cleared on hardware reset.

4.3.1.3 Wait State X-Data Memory (WSX) Bits 7-4

The Wait State X-Data Memory control bits (bits 7-4) allow for programming of the wait states for external X data memory. These bits are programmed as follows:

Table 4-1 Programming WSX Bits for Wait States

Bit String	Hex value	Number of wait states
0000	\$0	0
0001	\$1	1
0010	\$2	2
0011	\$3	3
0100	\$4	4
0101	\$5	5
0110	\$6	6
0111	\$7	7
1000	\$8	8
1001	\$9	9
1010	\$A	10
1011	\$B	11
1100	\$C	12
1101	\$D	13
1110	\$E	14
1111	\$F	15

4.3.1.4 Wait State P Memory (WSPM) Bits 3-0

The Wait State Program Memory control bits (bits 3-0) allow for programming of the wait states for external program memory. These bits are programmed as follows:

Table 4-2 Programming WSP Bits for Wait States

Bit String	Hex value	Number of wait states
0000	\$0	0
0001	\$1	1
0010	\$2	2
0011	\$3	3
0100	\$4	4
0101	\$5	5

Table 4-2 Programming WSP Bits for Wait States (Continued)

Bit String	Hex value	Number of wait states
0110	\$6	6
0111	\$7	7
1000	\$8	8
1001	\$9	9
1010	\$A	10
1011	\$B	11
1100	\$C	12
1101	\$D	13
1110	\$E	14
1111	\$F	15

Figure 4-3 shows an example of bus cycles without wait states, and **Figure 4-4** shows an example of bus cycles with wait states. For more information on wait states, see the *DSP56L811 Data Sheet (DSP56L11/D)*.

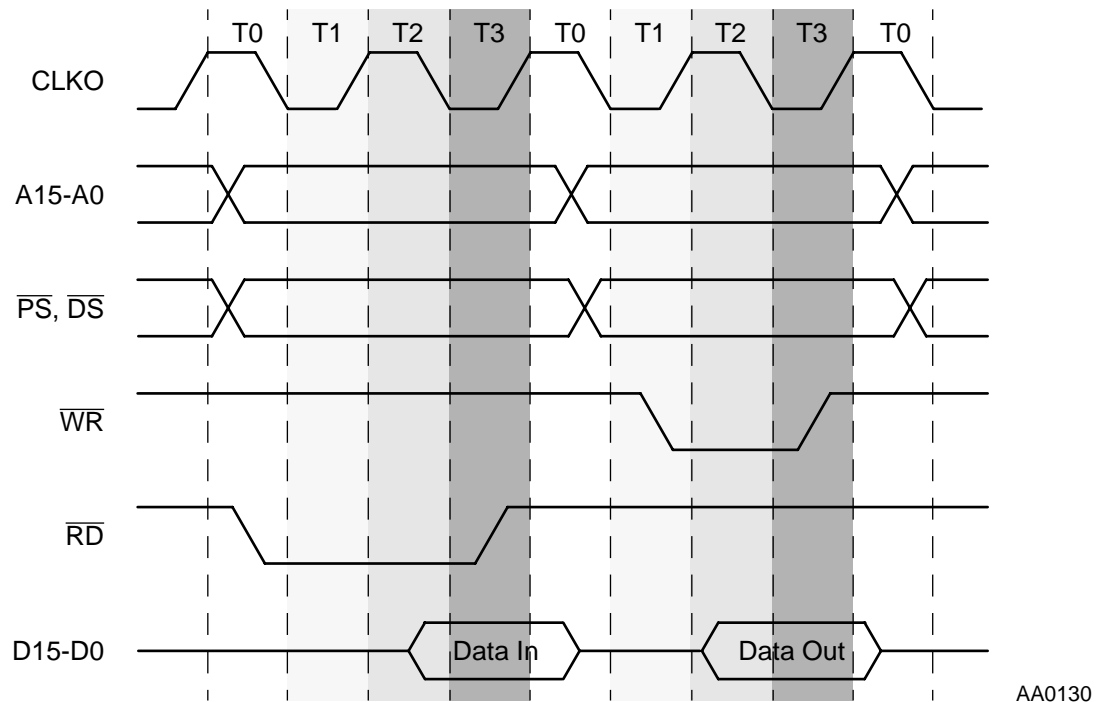


Figure 4-3 Bus Operation (Read/Write, Zero Wait States)

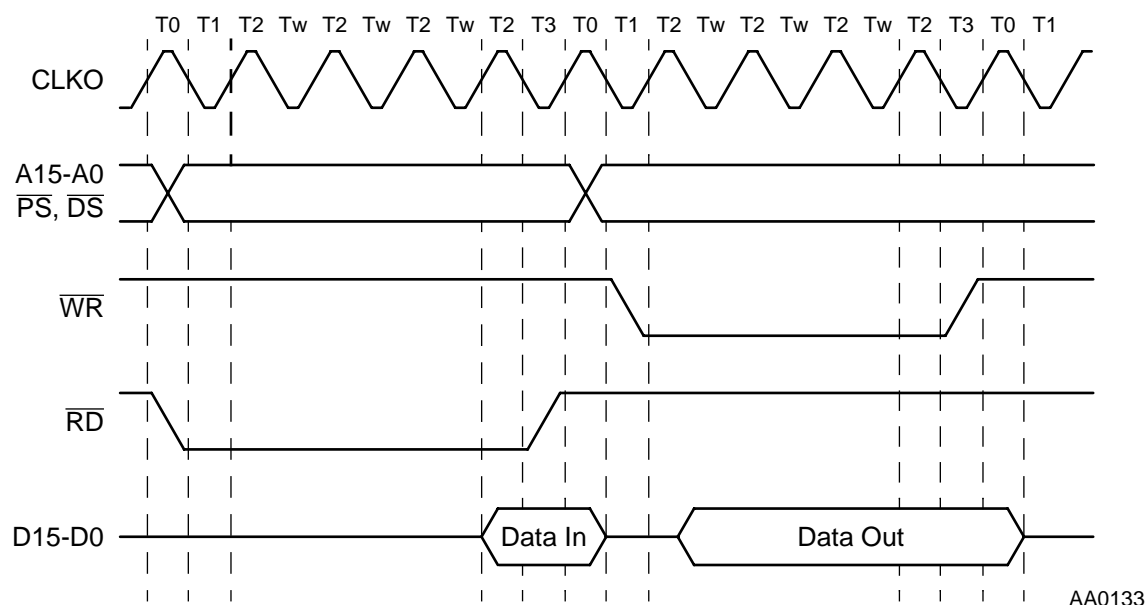


Figure 4-4 Bus Operation (Read/Write—Three Wait States)

4.3.1.5 Reserved BCR Register Bits

Bits 15-10 and 8 are reserved and are read as 0 during read operations. These bits should be written with 0 to ensure future compatibility.

4.3.2 Pins in Different Processing States

The DSP56800 core can be in one of six processing states:

- Normal
- Exception
- Reset
- Wait
- Stop
- Debug

In the normal processing state, each instruction cycle has two possible cases—either the processor needs to perform an external access, or all accesses are done internally. Exception processing is similar. For the case of an external access, the address and

bus control pins are driven to perform a normal bus cycle, and the data bus is also driven if the access is a write cycle.

For the case where there is no external access on a particular instruction cycle, one of two things can occur. The external address bus and control pins can remain driven with their previous values, or they can be tri-stated.

The reset processing state always tri-states the external address bus and internally pulls control pins high. The stop and wait processing states, however, either tri-state the address bus and control pins or let them remain driven with their previous values. This selection is made based on the value of the DRV bit in the BCR register.

The debug processing state is a special state used to debug and test programming code. This state is discussed in detail in **Section 9** of the *DSP56800 Family Manual (DSP56800FAM/AD)*, and is not described in the following tables.

Table 4-3 and **Table 4-4** describe the operation of the external memory port in these different modes.

Table 4-3 Port A Operation with DRV Bit Set to 0

Mode	Pins			
	A0-A15	PS, \overline{DS} , \overline{RD} , \overline{WR}	D0-D15	CLKO
Normal Mode External Fetch	Driven	Driven	Driven	Clock
Normal Mode Internal Fetch	Tri-state	Tri-state	Tri-state	Clock
Stop Mode	Tri-state	Tri-state	Tri-state	Clock
Wait Mode	Tri-state	Tri-state	Tri-state	Clock
Reset Mode	Tri-state	Internally pulled high	Tri-state	Clock

Table 4-4 Port A Operation with DRV Bit Set to 1

Mode	Pins			
	A0-A15	PS, \overline{DS} , \overline{RD} , \overline{WR}	D0-D15	CLKO
Normal Mode External Fetch	Driven	Driven	Driven	Clock
Normal Mode Internal Fetch	Driven	Driven	Tri-state	Clock

Table 4-4 Port A Operation with DRV Bit Set to 1 (Continued)

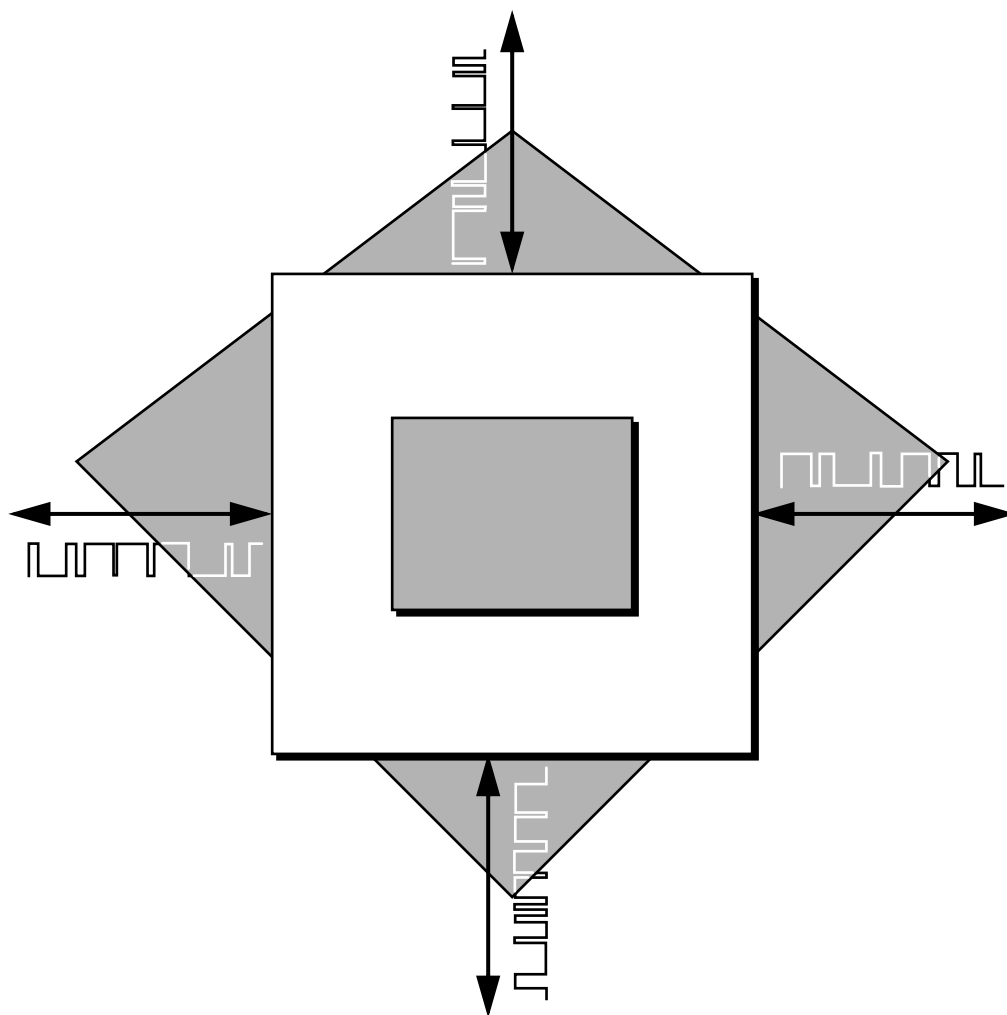
Mode	Pins			
	A0-A15	PS, \overline{DS} , \overline{RD} , \overline{WR}	D0-D15	CLKO
Stop Mode	Driven	Driven	Tri-state	Clock
Wait Mode	Driven	Driven	Tri-state	Clock
Reset Mode	Tri-state	Internally pulled high	Tri-state	Clock

The data lines are driven during normal mode external fetch only during an external write cycle.



SECTION 5

PORT B GPIO FUNCTIONALITY



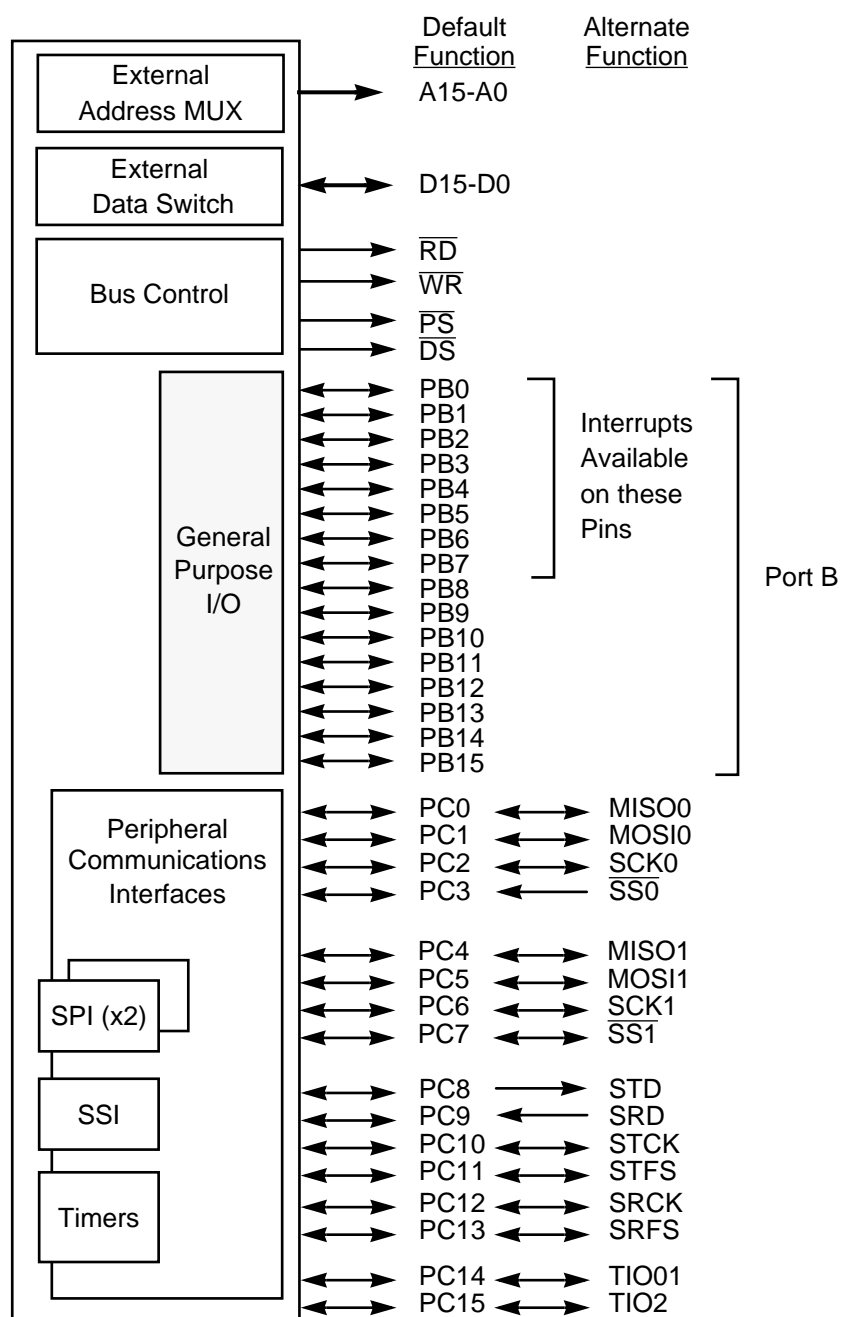
5.1	INTRODUCTION	5-3
5.2	PORT B PROGRAMMING MODEL	5-5
5.3	PORT B INTERRUPT GENERATION	5-8
5.4	PORT B PROGRAMMING EXAMPLES	5-9

5.1 INTRODUCTION

Port B is a dedicated general-purpose input-output (GPIO) port that provides 16 programmable I/O pins. Port B can be configured to generate an interrupt when a transition is detected on any of its lower eight pins, PB7-PB0, if they are configured as inputs. This section describes in detail the pins and programming specifics for Port B. **Figure 5-1** shows a block diagram of the I/O of the DSP56L811.

Note: After reset, all Port B pins are inputs.

Introduction



AA0134

Figure 5-1 DSP56L811 Input/Output Block Diagram

5.2 PORT B PROGRAMMING MODEL

There are three read/write registers in Port B:

- Port B Data Direction Register (PBDDR)
- Port B Data Register (PBD)
- Port B Interrupt Register (PBINT)

Figure 5-2 shows these the programming model for these three registers. Bit manipulation instructions can be used to access individual bits.

PBINT—X:\$FFEA		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Port B Interrupt Register		MSK	MSK	MSK	MSK	MSK	MSK	MSK	MSK	INV	INV	INV	INV	INV	INV	INV	INV
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Reset = \$0000																	
Read/Write																	
PBDDR—X:\$FFEB		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Port B Data Direction Register		BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD	BDD
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset = \$0000																	
Read/Write																	
PBD—X:\$FFEC		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Port B Data Register		BD	BD	BD	BD	BD	BD	BD	BD	BD	BD	BD	BD	BD	BD	BD	BD
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset = Uninitialized																	
Read/Write																	

GPIO Interrupt Vector:	
Port B GPIO Interrupt	P:\$0014
Enabling GPIO Interrupts in the Interrupt Priority Register:	
Set Bit 15 to 1 in the Interrupt Priority Register (X:\$FFFB)	

AA0135

Figure 5-2 DSP56L811 Port B Programming Model

5.2.1 Port B Data Direction Register (PBDDR)

The direction of each GPIO pin in Port B is determined by a corresponding control bit in the Port B Data Direction Register (PBDDR). The port pin is configured as an input if the corresponding Data Direction Register bit is cleared, and is configured as an output if the corresponding Data Direction Register bit is set. All Data Direction Register bits are cleared on processor reset, which configures all port pins as general-purpose input pins.

Table 5-1 PBDDR Bit Definition

BDD	Pin Direction
0	Input
1	Output

5.2.2 Port B Data Register (PBD)

The Port B Data register (PBD) is a read/write register used to access the Port B GPIO pins. The value of each bit in the register is determined by the individual pin configuration. All 16 pins can be configured as general purpose inputs (the default setting after reset) or as general purpose outputs. When configured as inputs, the lowest eight pins can also be configured as interrupts.

5.2.2.1 PBD Bit Values for General Purpose Inputs

If a Port B pin is configured as a general purpose input, the corresponding bit value reflects the value driven into the pin when the register is read. Writing to the register transfers the written value through the register to the output latch, but no value is transferred to the pin.

5.2.2.2 PBD Bit Values for General Purpose Outputs

If a Port B pin is configured as a general purpose output, writing a value to the PBD register drives the value to the output latch for that pin and to the pin itself. Reading the PBD register returns the value latched to the pin.

5.2.2.3 PBD Bit Values for Interrupt Inputs

For pins PB7-PB0, if a pin is configured as an input, it can be configured as an interrupt. The method for programming a bit as an interrupt input is described in **Port B Interrupt Register (PBINT)** on page 5-7. If a pin is configured as an interrupt input, the respective bit in the PBD register reflects the state of the pin when an interrupt event was detected. Detection of a defined interrupt event after the last register read causes the values of all interrupt-enabled pins at the time the interrupt

occurred to be locked into their respective bits in the PBD register. These values cannot be changed until a read of the register occurs again. Once the value is read, the bit values are unlocked and the pins operate like general purpose input pins until the next interrupt occurs and locks the values for those pins into the register until the next register read.

Table 5-2 Reading the PBD Register

BDD	MSK	Read of PBD Register Bit
0	0	Value on pin
0	1	Value in PBD latch
1	0	Value in PBD latch (and on pin)
1	1	Value in PBD latch (and on pin)

5.2.3 Port B Interrupt Register (PBINT)

The Port B Interrupt (PBINT) register is a 16-bit read/write control register used to set up and control the capability of generating interrupts from the lower eight Port B GPIO pins. The bits of this register are defined in the following text.

Note: The GPIO interrupt can be masked using the CH0 bit (bit 15) in the Interrupt Priority Register (IPR). See **DSP56L811 Interrupt Priority Register** on page 3-19 for information on the IPR register.

5.2.3.1 Interrupt Mask (MSK7-MSK0) Bits 15-8

The Interrupt Mask control bits (bits 15-8) are used to enable each of the lower eight Port B pins that can generate a GPIO interrupt. The programming for each of these eight bits is described in **Table 5-3**. The MSK bits are cleared on hardware reset.

Table 5-3 MSK Bit Definition

MSK	Function
0	Pin masked from generating interrupt.
1	Pin enabled for generating interrupt.

Note: Before any set MSK bit can enable a pin to generate an interrupt, the pin must be configured as an input pin in the PBDDR register.

Table 3-3 on page 3-16 lists the interrupt priority order for the DSP56L811.

5.2.3.2 Interrupt Invert (INV7-INV0) Bits 7-0

The Interrupt Invert bits (bits 7-0) are used to individually program whether a rising or falling transition is detected on a pin. The programming for each of these eight bits is described in **Table 5-4**. The INV bits are cleared on hardware reset.

Table 5-4 INV Bit Definition

INV	Transition detected
0	Rising Edge
1	Falling Edge

5.3 PORT B INTERRUPT GENERATION

The following information describes the capability for interrupt generation on the lower eight pins of Port B.

- Pins PB7-PB0 can be programmed to generate an interrupt by a transition on any of their input signals.
- Each pin can be programmed to detect a rising or a falling transition.
- Each pin can be individually enabled or masked.
- Each pin must be configured as an input pin to generate an interrupt.
- Any pin not used for generating an interrupt can be used as a GPIO pin.
- When the correct transition occurs on any pin enabled for interrupts, all pins enabled for interrupts are latched into the Port B Data register (PBD). Any pins not enabled for interrupt inputs are not latched.

As with all on-chip programmable peripheral interrupts for the DSP56L811, the Status Register (SR) must first be set to enable maskable interrupts (interrupts of level IPL1). See **3.6.1 DSP56L811 Status Register** on page 3-17 for more information about the SR register. Next, the IPR register must also be set to enable the interrupt. See **3.6.2 DSP56L811 Interrupt Priority Register** on page 3-19 for more information about the IPR register.

The interrupt feature of Port B is set up as follows:

1. Configure the SR to allow maskable interrupts.
2. Configure any pins desired for interrupt generation as inputs (in the PBDDR register).

3. Configure the associated INV bits for these pins, but leave associated MSK bits set to 0 (in the PBINT register).
4. Turn on the MSK bits for each associated pin.
5. Using the CH0 bit, enable the GPIO interrupt in the IPR register (see **3.6.2 DSP56L811 Interrupt Priority Register** on page 3-19).

Figure 5-3 shows the Port B interrupt block diagram.

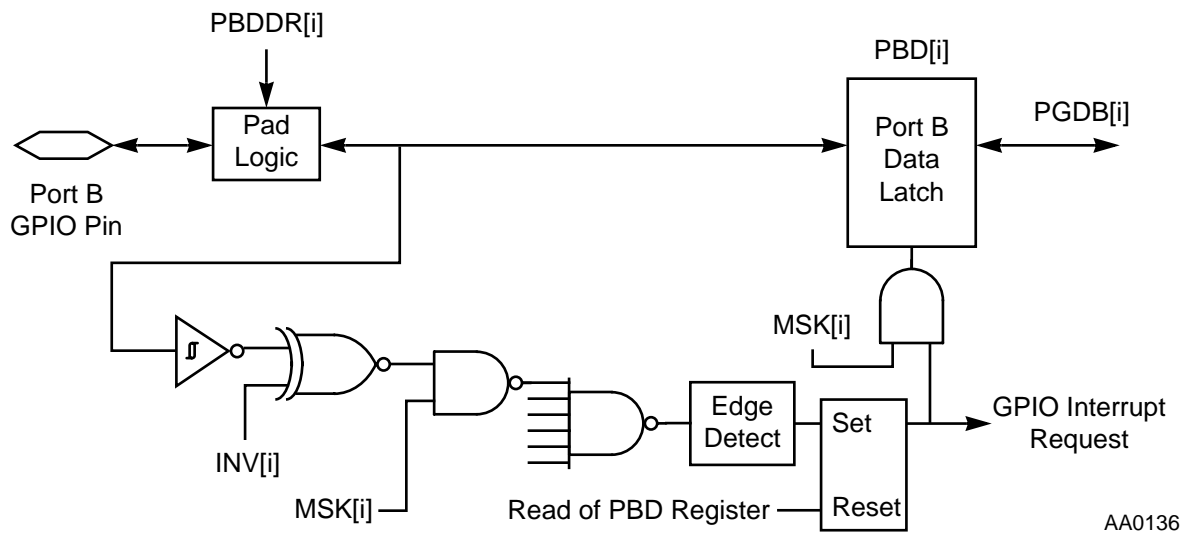


Figure 5-3 Port B Interrupt Block Diagram

5.4 PORT B PROGRAMMING EXAMPLES

The following examples show how to use Port B pins for receiving data (configured for input), sending data (configured as output), and for generating interrupts (configured as input).

5.4.1 Receiving Data on Port B

Example 5-1 shows how to configure Port B pins for as inputs, and use them for receiving data.

Example 5-1 Receiving Data on Port B

```
;*****
;* INPUT example      *
;* for Port B         *
;* of DSP56L811 chip *
;*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
PBD        EQU    $FFEC ; Port B Data
PBDDR      EQU    $FFEB ; Port B Data Direction Register
PBINT      EQU    $FFEA ; Port B Interrupt Register
data_i     EQU    $0001 ; data input

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+

      ORG    P:$0000      ; Cold Boot
      JMP    START        ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG    P:$E000      ; Warm Boot
      JMP    START        ; Hardware RESET vector (Mode 2)

      ORG    P:START      ; Start of program
;*****
;* General setup *
;*****

      MOVEP  #$0000,X:BCR; External Program memory has 0 wait states.
```


Example 5-1 Receiving Data on Port B (Continued)

```
                                ; External data memory has 0 wait states.
                                ; Port A pins tri-stated when no external access

;*****
;* Port B setup *
;*****

    MOVEP #$0000,X:PBINT ; Disable GPIO interrupt requests on
                        ; all lower eight Port B pins (default).
    MOVEP #$0000,X:PBDDR ; Select pins PB0-PB15 as input (default).

;*****
;* Main routine *
;*****

    ; ...
INPUT ; Input Loop
    ; ...
    MOVEP X:PBD,X0      ; Read PB0-PB15 into bits 0-15 of "data_i".
    MOVE  X0,X:data_i   ; Memory to memory move requires two MOVES.
    ; ...
    BRA   INPUT
```

5.4.2 Sending Data on Port B

Example 5-2 shows how to configure Port B pins as outputs and use them for sending data.

Example 5-2 Sending Data on Port B

```
;*****
;* OUTPUT example      *
;* for Port B          *
;* of DSP56L811 chip   *
;*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
```

Example 5-2 Sending Data on Port B (Continued)

```
PBD      EQU    $FFEC ; Port B Data
PBDDR    EQU    $FFEB ; Port B Data Direction Register
PBINT    EQU    $FFEA ; Port B Interrupt Register
data_o   EQU    $0000 ; data output

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then  |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+

      ORG     P:$0000      ; Cold Boot
      JMP     START        ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG     P:$E000      ; Warm Boot
      JMP     START        ; Hardware RESET vector (Mode 2)

      ORG     P:START      ; Start of program
;*****
;* General setup *
;*****
      MOVEP   #$0000,X:BCR  ; External Program memory has 0 wait states.
                               ; External data memory has 0 wait states.
                               ; Port A pins tri-stated when no external access.
;*****
;* Port B setup *
;*****
      MOVEP   #$0000,X:PBINT ; Disable GPIO interrupt requests on
                               ; all lower eight Port B pins (default).
      MOVEP   #$FFFF,X:PBDDR ; Select pins PB0-PB15 as output.

;*****
;* Main routine *
;*****
```

Example 5-2 Sending Data on Port B (Continued)

```

        ; ...
OUTPUT ; Output Loop
        ; ...
        MOVE  X:data_o,X0 ; Put bits 0-15 of "data_o" on pins PB0-PB15.
        MOVEP X0,X:PBD    ; Memory to memory move requires two MOVES.
        ; ...
        BRA   OUTPUT

```

5.4.3 Looping Data on Port B

Example 5-3 shows how to configure Port B pins to allow looping data from an output back to an input.

Example 5-3 Loop-back Example

```

;*****
;* LOOPBACK example *
;* for Port B      *
;* of DSP56L811 chip *
;*****

START      EQU   $0040 ; Start of program
BCR        EQU   $FFF9 ; Bus Control Register
PBD        EQU   $FFEC ; Port B Data
PBDDR      EQU   $FFEB ; Port B Data Direction Register
PBINT      EQU   $FFEA ; Port B Interrupt Register
data_o     EQU   $0000 ; data output
data_i     EQU   $0001 ; data input

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+

```

Example 5-3 Loop-back Example (Continued)

```
ORG    P:$0000      ; Cold Boot
JMP     START        ; also Hardware RESET vector (Mode 0, 1, 3)

ORG     P:$E000      ; Warm Boot
JMP     START        ; Hardware RESET vector (Mode 2)

ORG     P:START      ; Start of program
;*****
;* General setup *
;*****
        MOVEP #$0000,X:BCR ; External Program memory has 0 wait states.
                                ; External data memory has 0 wait states.
                                ; Port A pins tri-stated when no external access.
;*****
;* Port B setup *
;*****
        MOVEP #$0000,X:PBINT ; Disable GPIO interrupt requests on
                                ; all lower eight Port B pins (default).
        MOVEP #$FF00,X:PBDDR ; Select pins PB0-PB7 as input,
                                ; pins PB8-PB15 as output.

;*****
;* Main routine *
;*****
        ; ...
LOOPBACK ; Test Loop
        MOVE  X:data_o,X0 ; Put bits 8-15 of "data_o" on pins PB8-PB15.
        MOVEP X0,X:PBD    ; Bits going to input pins are ignored.
        ; ...
        MOVEP X:PBD,X0    ; Read PB0-PB7 into bits 0-7 of "data_i".
        MOVE  X0,X:data_i ; Bits 8-15 get values of PB8-PB15 as well.
        ; ...
        BRA   LOOPBACK
```

5.4.4 Generating Interrupts on Port B

Example 5-4 shows how to configure and use Port B for generating interrupts.

Example 5-4 Generating Interrupts on Port B

```
*****
;* INTERRUPT example *
;* for Port B          *
;* of DSP56L811 chip *
*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
IPR        EQU    $FFFB ; Interrupt Priority Register
PBD        EQU    $FFEC ; Port B Data
PBDDR      EQU    $FFE8 ; Port B Data Direction Register
PBINT      EQU    $FFEA ; Port B Interrupt Register
data_o     EQU    $0000 ; data output
data_i     EQU    $0001 ; data input

*****
;* Vector setup *
*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then  |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+

      ORG    P:$0000      ; Cold Boot
      JMP    START        ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG    P:$E000      ; Warm Boot
      JMP    START        ; Hardware RESET vector (Mode 2)

      ORG    P:$0014      ;
      JSR    GPIOISR      ; GPIO Interrupt vector

      ORG    P:START      ; Start of program
```

Example 5-4 Generating Interrupts on Port B (Continued)

```
*****
;* General setup *
*****

    MOVEP #$0000,X:BCR; External Program memory has 0 wait states.
                                ; External data memory has 0 wait states.
                                ; Port A pins tri-stated when no external access.

    BFCLR #$0200,SR    ; Allow IPL (Interrupt Priority Level) 0.
                                ; -- Enable maskable interrupts.
                                ; -- (peripherals, etc.)

*****
;* Port B setup *
*****

    MOVEP #$8000,X:PBINT ; Enable GPIO interrupt requests for
                                ; rising transitions on PB7.

    MOVEP #$FF00,X:PBDDR ; Select PB0-PB7 as input, PB8-PB15 as output.
    BFSET #$8000,X:IPR   ; Enable GPIO interrupts.

*****
;* Main routine *
*****

    ; ...

    MOVE  #$0000,X:data_o ; Initialize "data_o"
TESTPAT ; Test Pattern Loop

    MOVE  X:data_o,X0      ; Put bits 8-15 of "data_o" on pins PB8-PB15.
    MOVEP X0,X:PBD        ; Bits going to input pins are ignored.
    ADD   #$0100,X0       ; Change output pattern: increment by 1.
    MOVE  X0,X:data_o     ; Increment upper byte by 1.
    MOVEP X:PBD,X0        ; Read PB0-PB7 into bits 0-7 of "data_i".
    MOVE  X0,X:data_i     ; Bits 8-15 get values of PB8-PB15 as well.
    ; ...

    BRA   TESTPAT

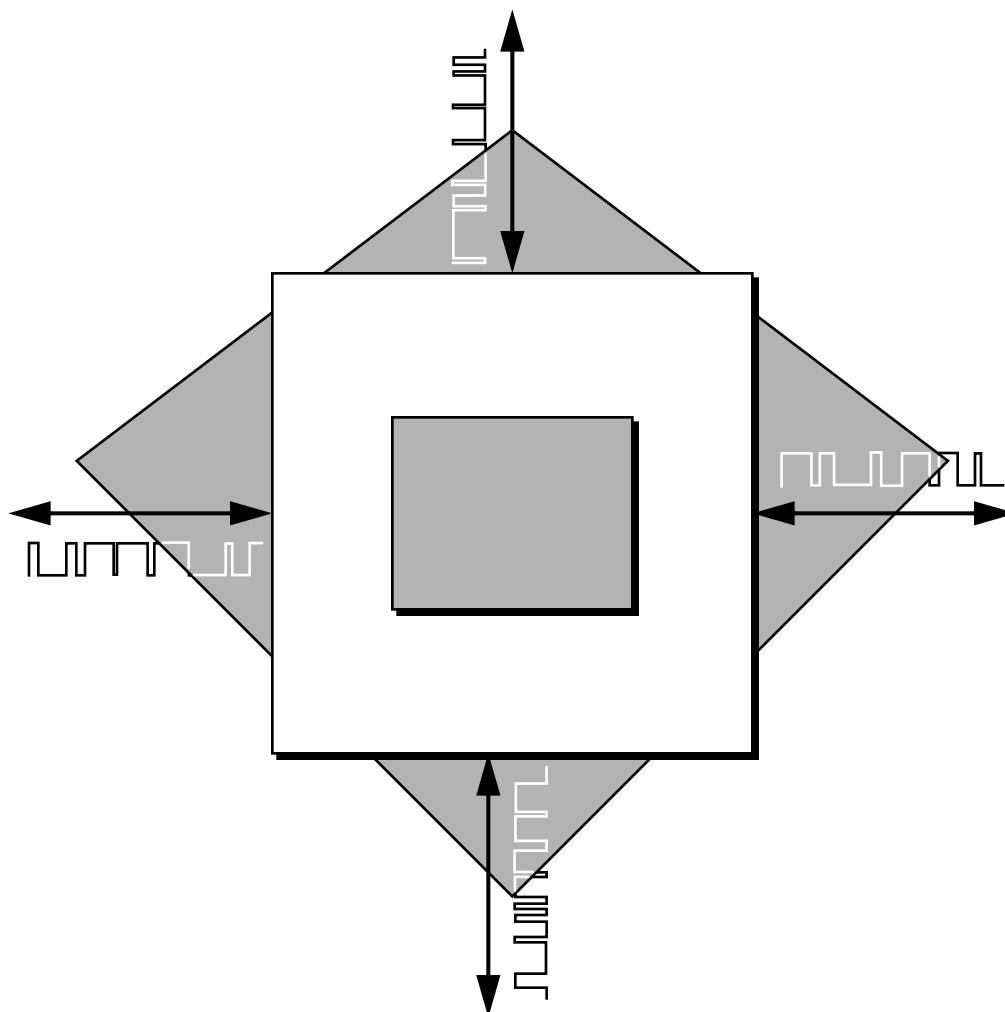
GPIOISR ; GPIO Interrupt Service Routine
    ; interrupt code

    RTI
```



SECTION 6

PORT C GPIO FUNCTIONALITY



6.1	INTRODUCTION	6-3
6.2	PORT C PROGRAMMING MODEL	6-5
6.3	PORT C PROGRAMMING EXAMPLES	6-7

6.1 INTRODUCTION

The Peripheral Communications Port (Port C) provides 16 multiplexed programmable I/O pins. These pins may be used as general purpose I/O (GPIO) pins or allocated to on-chip peripherals—a timer module, two Serial Peripheral Interfaces (SPI), and a Synchronous Serial Interface (SSI). Each pin is individually programmable.

This section describes how to program the pins for Port C and provides general information about their use as GPIO pins. Specifics for programming the various peripherals represented on Port C are provided in the appropriate sections, as follows:

- SPI module programming specifics are located in **Section 7, Serial Peripheral Interface**.
- SSI module programming specifics are located in **Section 8, Synchronous Serial Interface**.
- Timer module programming specifics are located in **Section 9, Timers**.

The Port C I/O interfaces are intended to minimize system chip count and “glue” logic in many DSP applications. Each I/O interface has its own control, status, and data registers that are treated as memory-mapped peripheral registers by the DSP56L811. Each interface has several dedicated interrupt vector addresses and control bits to enable/disable interrupts. This minimizes the overhead associated with servicing the device, since each interrupt source may have its own service routine.

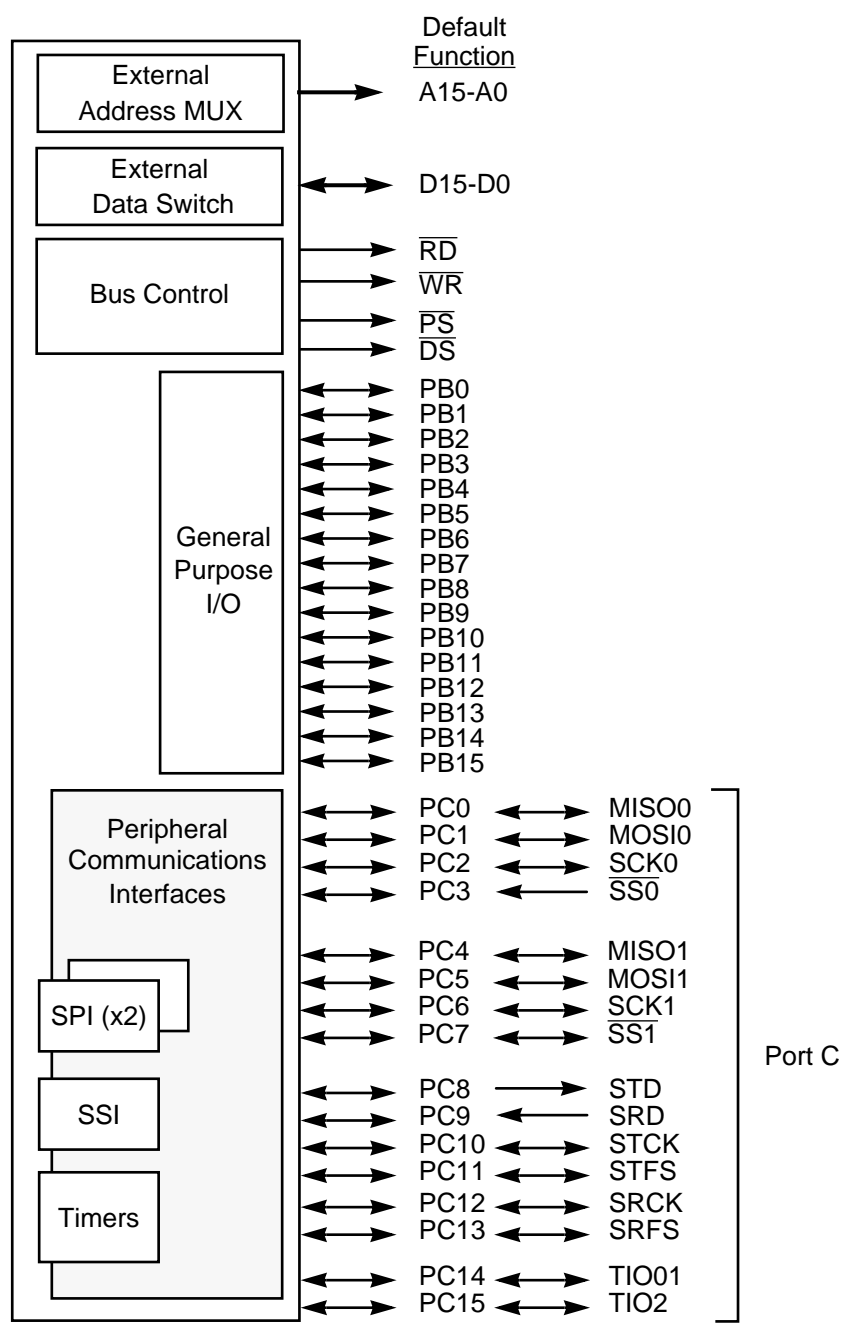


Figure 6-1 DSP56L811 Input/Output Block Diagram

6.2 PORT C PROGRAMMING MODEL

Port C provides three read/write registers:

- Port C Control Register (PCC)
- Port C Data Direction Register (PCDDR)
- Port C Data Register (PCD)

These registers are shown in **Figure 6-2**. The PCC register allows the programming of Port C pins as general purpose I/O pins or as dedicated on-chip peripheral pins. The PCDDR register specifies whether a pin programmed as a GPIO pin is an input or an output pin. The PCD register allows accessing data transmitted through a GPIO pin. Bit manipulation instructions can be used to access individual bits.

Port C pins associated with the SPI can also be configured as open-drain drivers by the Wired-OR Mode (WOM) control bit in the SPI Control Register (SPCR). More information on this register is available in **Section 7, Serial Peripheral Interface**.

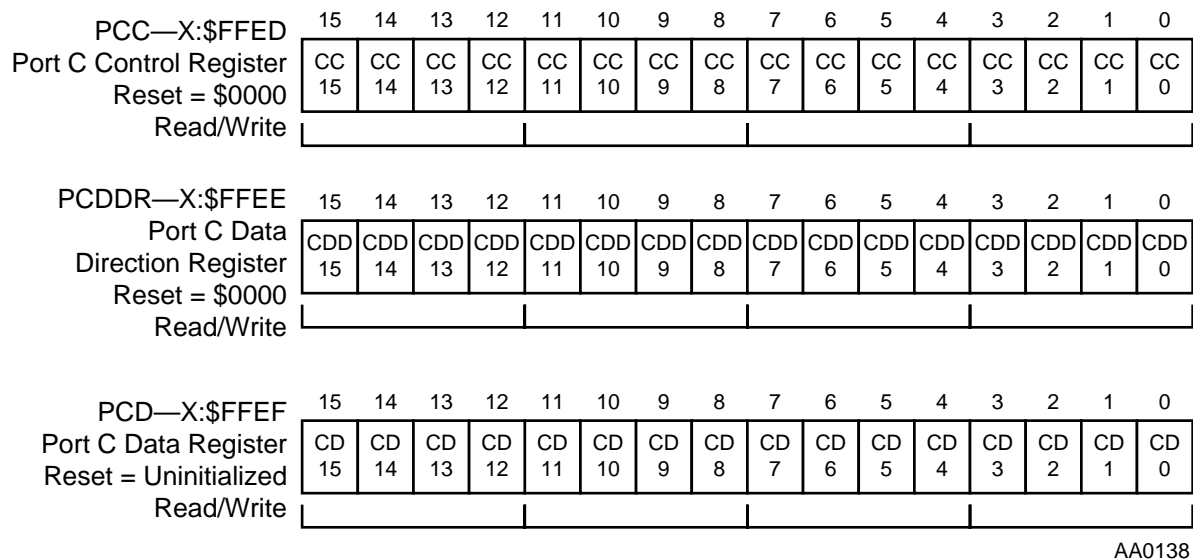


Figure 6-2 DSP56L811 Port C Programming Model

6.2.1 Port C Control Register (PCC)

Port C pins can be programmed under software control as general purpose I/O pins or as dedicated on-chip peripheral pins. The Port C Control Register (PCC) allows the

Port C Programming Model

port pins to be selected for one of these two functions. Each Port C pin is independently configured as a general purpose I/O pin if the corresponding CC bit is cleared and is configured as an SPI, SSI, or Timer pin if the corresponding PCC register bit is set. **Table 6-1** shows how this bit is defined.

Table 6-1 PCC Bit Definition

CC	Pin programmed as
0	General Purpose I/O Pin
1	Dedicated Peripheral Pin

Unlike the pins on Port B, none of the GPIO pins on Port C can be configured to provide interrupts.

Note: All CC bits are cleared on reset.

6.2.2 Port C Data Direction Register (PCDDR)

If a port pin is selected as a general purpose I/O pin, the direction of that pin is determined by a corresponding bit in the PCDDR register. The port pin is configured as an input if the corresponding CDD bit is cleared, and is configured as an output if the corresponding CDD bit is set. **Table 6-2** shows how this bit is defined.

Table 6-2 PCDDR Bit Definition

CDD	Pin direction
0	Input
1	Output

All PCC register bits and PCDDR register bits are cleared on processor reset, configuring all Port C pins as general purpose input pins. If the port pin is selected as an on-chip peripheral pin, the corresponding data direction bit is ignored and the direction of that pin is determined by the operating mode of the on-chip peripheral.

Note: All CDD bits are cleared on reset.

6.2.3 Port C Data Register (PCD)

The Port C Data Register (PCD) allows accessing data through a port pin that has been configured as a general purpose I/O pin. Data written to the PCD register is stored in an output latch. If the port pin is configured as an output, the output latch data is driven out on the port pin. When the PCD register is read, the logic value on the output port pin is read. If the port pin is configured as an input, data written to the PCD register is still stored in the output latch, but is not gated to the port pin. When the PCD register is read, the state of the port pin is read. That is, reading the port data register will reflect the state of the pins regardless of how they were configured.

When a port pin is configured as a dedicated on-chip peripheral pin, the port data register reads the state of the input pin or output driver.

6.3 PORT C PROGRAMMING EXAMPLES

The following examples show how to configure Port C pins as general-purpose I/O pins, using them for receiving data (configured as input) and for sending data (configured as output).

6.3.1 Receiving Data on Port C GPIO Pins

Example 6-1 shows how to configure Port C pins as general-purpose I/O pins, and how to use them for receiving data.

Example 6-1 Receiving Data on Port C GPIO Pins

```

;*****
;* INPUT example      *
;* for Port C         *
;* of DSP56L811 chip  *
;*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
PCC        EQU    $FFED ; Port C Control Register

```

Example 6-1 Receiving Data on Port C GPIO Pins (Continued)

```
PCD      EQU    $FFEF ; Port C Data
PCDDR    EQU    $FFEE ; Port C Data Direction Register
data_i   EQU    $0001 ; data input

;*****
;* Vector setup *
;*****
;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+
      ORG     P:$0000      ; Cold Boot
      JMP     START       ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG     P:$E000      ; Warm Boot
      JMP     START       ; Hardware RESET vector (Mode 2)

      ORG     P:START      ; Start of program
;*****
;* General setup *
;*****
      MOVEP   #$0000,X:BCR; External Program memory has 0 wait states.
                        ; External data memory has 0 wait states.
                        ; Port A pins are tri-stated when no
                        ; external access occurs.

;*****
;* Port C setup *
;*****
      MOVEP   #$0000,X:PCC  ; Configures PC0-PC15 as GPIO pins (default)
      MOVEP   #$0000,X:PCDDR; Selects pins PC0-PC15 as input (default)

;*****
;* Main routine *
;*****
      ; ...

INPUT ; Input Loop
```

Example 6-1 Receiving Data on Port C GPIO Pins (Continued)

```
; ...

MOVEP X:PCD,X0; Read PC0-PC15 into bits 0-15 of "data_i"
MOVE  X0,X:data_i; (memory to memory move requires two moves)
; ...

BRA   INPUT
```

6.3.2 Sending Data on Port C GPIO Pins

Example 6-2 shows how to configure Port C pins as general-purpose I/O pins, and how to use them for sending data.

Example 6-2 Sending Data on Port C GPIO Pins

```
;*****
;* OUTPUT example      *
;* for Port C          *
;* of DSP56L811 chip   *
;*****

START      EQU   $0040 ; Start of program
BCR        EQU   $FFF9 ; Bus Control Register
PCC        EQU   $FFED ; Port C Control Register
PCD        EQU   $FFEF ; Port C Data
PCDDR      EQU   $FFEE ; Port C Data Direction Register
data_o     EQU   $0000 ; data output

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).     |
;+-----+
```

Example 6-2 Sending Data on Port C GPIO Pins (Continued)

```
ORG    P:$0000      ; Cold Boot
JMP     START        ; also Hardware RESET vector (Mode 0, 1, 3)
ORG     P:$E000      ; Warm Boot
JMP     START        ; Hardware RESET vector (Mode 2)

ORG     P:START      ; Start of program
;*****
;* General setup *
;*****
        MOVEP #$0000,X:BCR ; External Program memory has 0 wait states.
                                ; External data memory has 0 wait states.
                                ; Port A pins are tri-stated when no
                                ; external access occurs.
;*****
;* Port C setup *
;*****
        MOVEP #$0000,X:PCC   ; Configure PC0-PC15 as GPIO pins (default).
        MOVEP $FFFF,X:PCDDR ; Select pins PC0-PC15 as output.
;*****
;* Main routine *
;*****
        ; ...
OUTPUT ; Output Loop
        ; ...
        MOVE  X:data_o,X0 ; Put bits 0-15 of "data_o" on pins PC0-PC15.
        MOVEP X0,X:PCD     ; Memory to memory move requires two moves.
        ; ...
        BRA   OUTPUT
```

6.3.3 Looping Data on Port C GPIO Pins

Example 6-3 shows how to configure Port C GPIO pins and then use them to loop data back from an output to an input.

Example 6-3 Loop-back Example

```

;*****
;* LOOPBACK example *
;* for Port C      *
;* of DSP56L811 chip *
;*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
PCC        EQU    $FFED ; Port C Control Register
PCD        EQU    $FFEF ; Port C Data
PCDDR      EQU    $FFEE ; Port C Data Direction Register
data_o     EQU    $0000 ; data output
data_i     EQU    $0001 ; data input

;*****
;* Vector setup *
;*****
;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+

      ORG    P:$0000      ; Cold Boot
      JMP    START        ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG    P:$E000      ; Warm Boot
      JMP    START        ; Hardware RESET vector (Mode 2)

      ORG    P:START      ; Start of program
;*****
;* General setup *
;*****

```

Example 6-3 Loop-back Example (Continued)

```
    MOVEP  #$0000,X:BCR    ; External Program memory has 0 wait states.
                                ; External data memory has 0 wait states.
                                ; Port A pins are tri-stated when no
                                ; external access occurs.

;*****
;* Port C setup *
;*****

    MOVEP  #$0000,X:PCC    ; Configure PC0-PC15 as GPIO pins (default).
    MOVEP  #$FF00,X:PCDDR  ; Select pins PC0-PC7 as input,
                                ; pins PC8-PC15 as output.

;*****
;* Main routine *
;*****

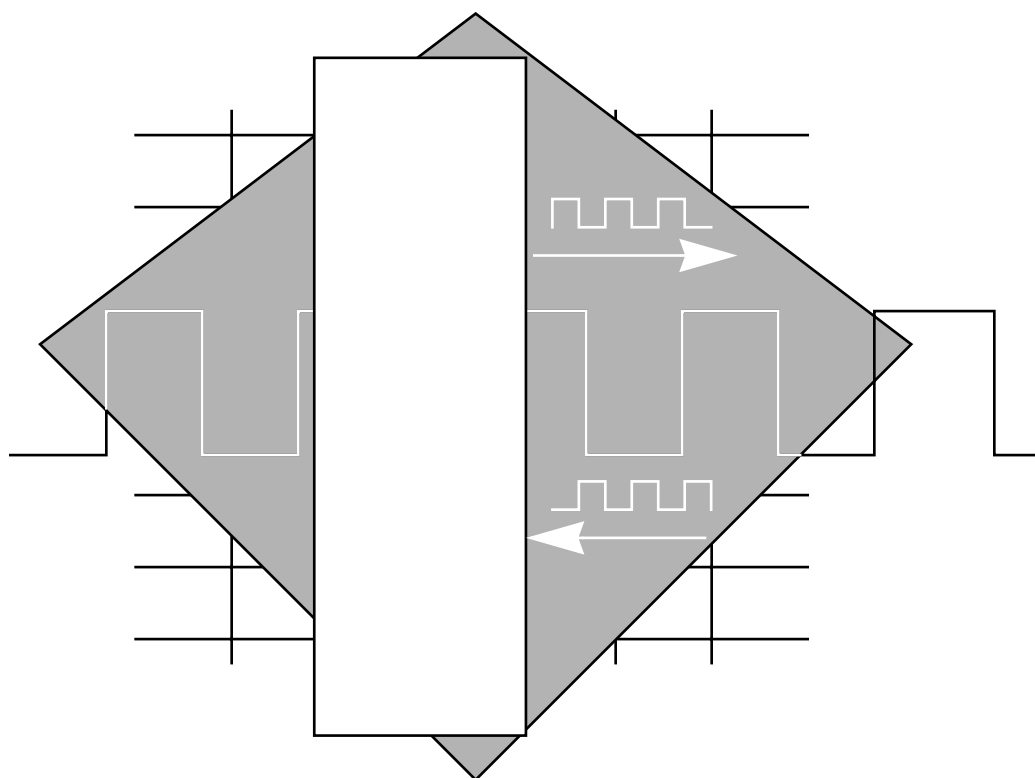
    ; ...

LOOPBACK ; Test Loop
    MOVE   X:data_o,X0     ; Put bits 8-15 of "data_o" on pins PC8-PC15.
    MOVEP  X0,X:PCD        ; Bits going to input pins are ignored.
    ; ...
    MOVEP  X:PCD,X0        ; Read PC0-PC7 into bits 0-7 of "data_i".
    MOVE   X0,X:data_i     ; Bits 8-15 get values of PC8-PC15 as well.
    ; ...
    BRA    LOOPBACK
```



SECTION 7

SERIAL PERIPHERAL INTERFACE



7.1	INTRODUCTION	7-3
7.2	SPI ARCHITECTURE	7-5
7.3	SPI PROGRAMMING MODEL	7-6
7.4	SPI DATA AND CONTROL PINS	7-11
7.5	SPI SYSTEM ERRORS.	7-14
7.6	CONFIGURING PORT C FOR SPI FUNCTIONALITY	7-16
7.7	PROGRAMMING EXAMPLES	7-18

7.1 INTRODUCTION

This section discusses the architecture of the Serial Peripheral Interface (SPI) provided on Port C, its pins, and its programming model. The section includes information on SPI system errors, a discussion of overrun on the SPI, correct programming of Port C when using the SPI, and low-power operation with the SPI.

The SPI is an independent serial communications subsystem that allows the DSP56L811 to communicate synchronously with peripheral devices such as LCD display drivers, A/D subsystems, and microprocessors. More sophisticated uses such as inter-processor communication in a multiple master system are also easy to implement. The SPI system can be configured as either a master or a slave device with high data rates. In master mode, a transfer is initiated when data is written to the SPI Data Register. In slave mode, a transfer is initiated by the reception of a clock signal.

Clock control logic allows a selection of clock polarity and a choice of two, fundamentally different clocking protocols to accommodate most available synchronous serial peripheral devices. In some cases, the phase and polarity are changed between transfers to allow a master device to communicate with peripheral slaves having different requirements. When the SPI is configured as a master, software selects one of eight different bit rates for the clock.

Error detection logic is included to support interprocessor communications. A write-collision detector indicates when an attempt is made to write data to the serial shift register while a transfer is in progress. A multiple-master mode-fault detector automatically disables SPI output drivers if more than one MCU simultaneously attempts to become bus master.

The DSP56L811 provides two identical SPIs, SPI0 and SPI1. **Figure 7-1** shows SPI0 and SPI1.

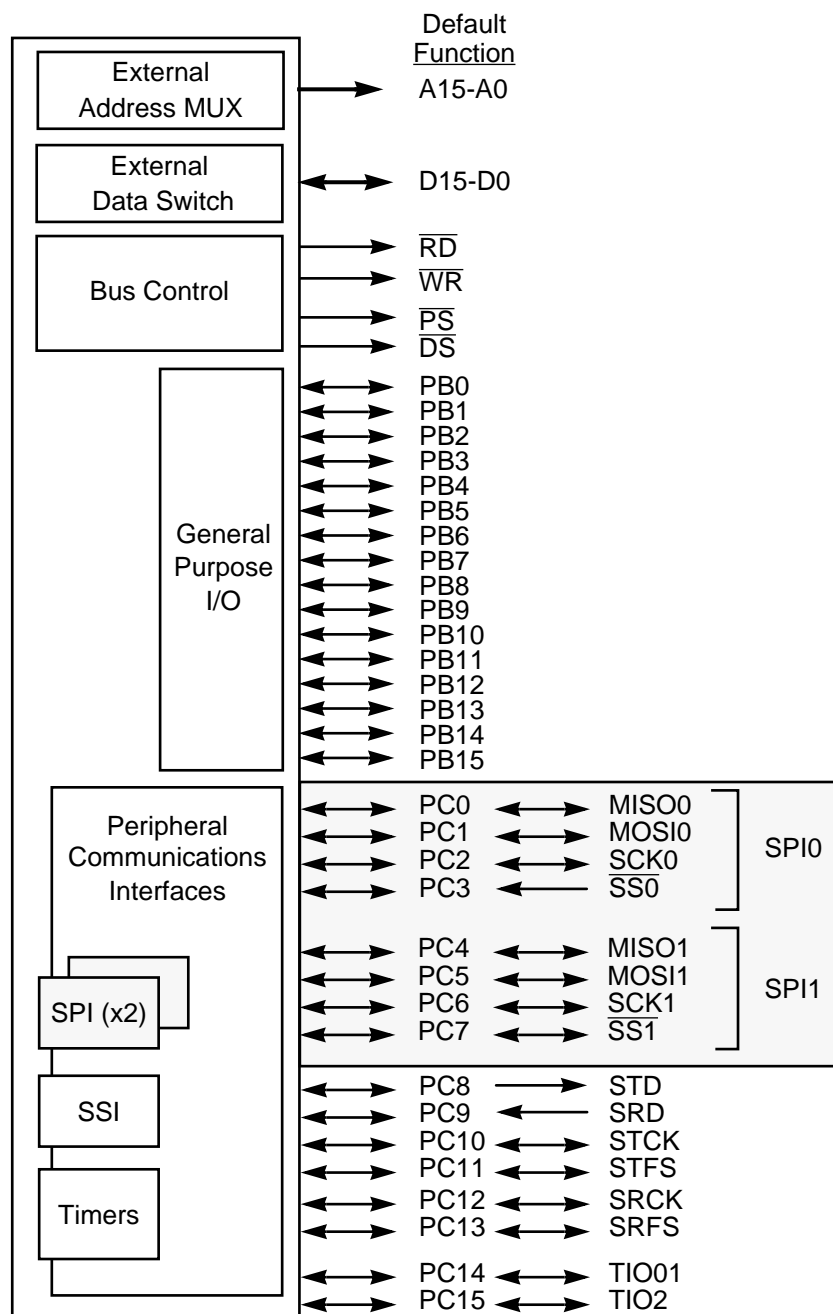


Figure 7-1 DSP56L811 Input/Output Block Diagram

Figure 7-2 shows a block diagram of the SPI subsystem. When an SPI transfer occurs, a byte is shifted out one data pin, while a different byte is simultaneously shifted in a second data pin. Another way to view this transfer is that an 8-bit shift register in the master device and another 8-bit shift register in the slave are connected as a circular 16-bit shift register. When a transfer occurs, this distributed shift register is shifted eight bit positions; thus, the bytes in the master and slave are effectively exchanged.



The central element in the SPI system is the block containing the shift register and receive data buffer. The system is single-buffered in the transmit direction and double-buffered in the receive direction. This means that new data for transmission cannot be written to the shifter until the previous transfer is complete. However,

SPI Programming Model

received data is transferred into a parallel read data buffer, so the shifter is free to accept a second serial byte. As long as the first byte is read out of the read data buffer before the next serial byte is ready to be transferred, no overrun condition occurs. A single memory address is used for reading data from the read buffer and writing data to the shifter.

The SPI status block represents the SPI status functions (transfer complete, write collision, and mode fault) located in the SPSR status register. The SPI control block contains the SPCR control register used to set up and control the SPI system.

7.3 SPI PROGRAMMING MODEL

Each SPI peripheral provides the following registers:

- SPI Control Register (SPCR)
- SPI Status Register (SPSR)
- SPI Data Register (SPDR)

These registers are shown in **Figure 7-3**. The descriptions of the registers in the following paragraphs apply equally to the corresponding registers on SPI0 and SPI1.



SPI1: Set Bit 12 to 1 in the Interrupt Priority Register (X:\$FFFB).
SPI0: Set Bit 13 to 1 in the Interrupt Priority Register (X:\$FFFB).

Figure 7-3 SPI Programming Model

The WOM, MST, CPL, CPH, and the SPR[2:0] bits should not be changed when a transfer is taking place. Typically, these bits are only modified in slave mode when the SPE bit is set to 0, when the device is not selected, or in master mode and no transfer is in progress.

SPI Programming Model

Note: When writing the SPCR0 or the SPCR1 register, it is necessary to first write the register with the SPE bit set to 0. Then a second write is performed to the SPCR register with the same value except that the SPE bit is set. The BFSET instruction can be used in place of this second write to set the SPE bit. This is true for all bits in the SPCR except the SPIE bit, which can be modified with a BFSET or BFCLR instruction at any time, even when the SPE bit is set.

The SPCR0 and SPCR1 registers are reset to \$0000 on hardware reset. The bits of the SPCR0 and SPCR1 registers are described in the following text.

7.3.1.1 SPI Clock Rate Select (SPR2-0) Bits 8, 1-0

The SPI Clock Rate Select bits (bits 8, 1-0) are used to program the divider of the Phi Clock to generate a serial bit clock for the SPI peripheral when the SPI is configured as a master device. For an SPI configured as a slave, the serial clock is input from the master and these bits have no meaning. The SPR bits are cleared on hardware reset.

Table 7-1 SPR Divider Programming

SPR[2:0]	Divider
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Note: The maximum frequency of the serial bit clock is limited to $1/4$ of the maximum rated clocking frequency of the part. This means that for a 40 MHz DSP56L811, the SPR bits must be programmed so that the clocking frequency of the serial bit clock is less than or equal to 10 MHz. Thus, setting the SPR[2:0] bits to 000 (divide by 1) can be used only when the frequency of the Phi clock is less than or equal to $1/4$ of the maximum clocking frequency of the DSP56800 core. Likewise, setting the SPR[2:0] bits to 001 (divide by 2) can be used only when the Phi clock used in an application is less than or equal to half the maximum clocking frequency of the DSP56800 core. All other combinations (divide by 4 through divide by 128) can be used with any Phi clock frequency.

7.3.1.2 SPI Interrupt Enable (SPIE) Bit 7

The SPI Interrupt Enable control bit (bit 7) is used to enable interrupts from the SPI port. When interrupts are disabled (the SPIE bit is set to 0), any pending interrupt is cleared and polling is used to sense the SPIF and MDF bits. When interrupts are enabled (the SPIE bit is set to 1), an SPI interrupt is requested if either the SPIF or the MDF bit is set. The SPIE bit is cleared on hardware reset.

As with all on-chip peripheral interrupts for the DSP56L811, the SR register must first be set to enable maskable interrupts (interrupts of level IPL1). Next, the IPR register must also be set to enable the interrupt. **Table 7-2** lists the appropriate bits to set in the IPR register, and the corresponding interrupt vector.

Table 7-2 SPI Interrupt

SPI port (Register)	Bit in IPR	Interrupt Vector	Interrupt Priority
SPI0 (SPCR0)	Bit 13 (CH2)	\$002A	0
SPI1 (SPCR1)	Bit 12 (CH3)	\$0028	0

Table 3-3 Interrupt Priority Structure on page 3-16 lists the interrupt priority order for the DSP56L811. Finally, SPI interrupts are configured within the SPI itself, using the SPIE bit.

7.3.1.3 SPI Enable (SPE) Bit 6

The SPI Enable control bit (bit 6) is used to enable the SPI port functionality. Clearing the SPE bit disables the SPI peripheral and the shuts off the Phi clock signal provided to the SPI port to reduce power consumption. The SPE bit is cleared on hardware reset.

7.3.1.4 Wired-OR Mode (WOM) Bit 5

The Wired-OR Mode control bit (bit 5) is used to select the nature of the SPI pins. When enabled (the WOM bit is set to 1), the SPI pins in Port C are configured as open-drain drivers, with the P-channel pullups disabled. When disabled (the WOM bit is cleared), the SPI pins are configured as push-pull drivers. The WOM bit is cleared on hardware reset.

7.3.1.5 Master Mode Select (MST) Bit 4

The Master Mode Select control bit (bit 4) is used to select master or slave mode. The MST bit is cleared on hardware reset. **Table 7-3** shows how this mode is set.

Table 7-3 SPI Mode Programming

Bit	SPI Mode
0	Slave
1	Master

7.3.1.6 Clock Polarity (CPL) Bit 3

The Clock Polarity control bit (bit 3) is used to define the polarity of the serial bit clock. When data is not being transferred and this bit is set to 0, the SCK pin of the master device idles as a logic low. When the CPL bit is set to 1, the SCK pin idles as a logic high. The CPL bit is cleared on hardware reset.

7.3.1.7 Clock Phase (CPH) Bit 2

The Clock Phase control bit (bit 2) is used in conjunction with the CPL bit to control the clock-data relationship between the master and slave. This bit selects one of two different clocking protocols. The CPH bit is cleared on hardware reset.

7.3.1.8 Reserved SPCR Register Bits

Bits 15-9 of the SPCR0 and SPCR1 registers are reserved and are read as 0 during read operations. These bits should be written with 0 for future compatibility.

7.3.2 SPI Status Register (SPSR0 and SPSR1)

The SP0 and SPSR1 registers are 16-bit read-only registers used to indicate the status of the SPI0 and SPI1 peripherals, respectively. The bits within these registers are arranged and interpreted identically. The only differences are that the two registers have different addresses and represent the status of different SPIs. The value of one register is not affected by the value of the other register. The bits of these register are defined in the following text.

7.3.2.1 SPI Interrupt Complete Flag (SPIF) Bit 7

The SPI Interrupt Complete Flag (bit 7) is set upon completion of data transfer between the processor and the external device. If the SPIF bit goes high and the SPIE bit is set, an interrupt is generated. To clear the SPIF bit, read the SPSR register with the SPIF bit set, then access the SPDR register. Unless the SPSR register is read first (with the SPIF bit set), attempts to write to the SPDR register are not permitted. The SPIF bit is cleared on hardware reset.

7.3.2.2 Write Collision (WCOL) Bit 6

The Write Collision flag (bit 6) is set when a write collision condition is detected. This bit is cleared by reading the SPSR register (with MDF set) followed by an access of the SPDR register. The WCOL flag is cleared on hardware reset.

7.3.2.3 Mode Fault (MDF) Bit 4

The Mode Fault flag (bit 4) is set when a mode fault has occurred. This bit is cleared by reading the SPSR register (with MDF set) followed by a write to the SPCR register. The MDF flag is cleared on hardware reset.

7.3.2.4 Reserved SPSR Register Bits

Bits 15-8, 5, and 3-0 of the SPSR0 and SPSR1 registers are reserved and are read as 0 during read operations. These bits should be written with 0 for future compatibility.

7.3.3 SPI Data Registers (SPDR0 and SPDR1)

The SPDR0 and SPDR1 registers are 16-bit read/write registers used when the SPI devices are transmitting or receiving data on the serial bus. Only a write to one of these registers initiates transmission or reception of a byte, and this only occurs on the master device. At the completion of transferring a byte of data, the SPIF status bit (in the corresponding SPSR register) is set in both the master and slave devices.

A read of an SPDR register is actually a read of a buffer. To prevent an overrun and the loss of the byte that caused the overrun, the first SPIF must be cleared by the time a second transfer of data from the shift register to the read buffer is initiated.

This register is double-buffered for input, and single-buffered for output.

7.4 SPI DATA AND CONTROL PINS

During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows individual selection of a slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the select line can optionally be used to indicate a multiple master bus contention.

Each SPI device has four dedicated I/O pins. The four pins on SPI0 are as follows:

- **MISO0/PC0 (SPI0 Master In Slave Out)**—The MISO0 pin carries one of two unidirectional serial data signals. It is an input to a master device and an output from a slave device. If a slave device is not selected, the MISO0 line of the slave device is placed in the high-impedance state. MISO0 can be programmed as a general purpose I/O pin called PC0 when the SPI MISO0 function is not being used.
- **MOSI0/PC1 (SPI0 Master Out Slave In)**—The MOSI0 pin carries the second of the two unidirectional serial data signals. It is an output from a master device and an input to a slave device. The master device places data on the MOSI0 line a half-cycle before the clock edge that the slave device uses to latch the data. MOSI0 can be programmed as a general purpose I/O pin called PC1 when the SPI MOSI0 function is not being used.
- **SCK0/PC2 (SPI0 Serial Clock)**—SCK0, an input to a slave device, is generated by the master device and synchronizes data movement in and out of the device through the MOSI0 and MISO0 lines. Master and slave devices are capable of exchanging a byte of information during a sequence of eight clock cycles. Slave devices ignore the SCK signal unless the slave select pin is active low.

Four possible timing relationships can be chosen by using control bits CPOL and CPH in the Serial Peripheral Control Register (SPCR). Both master and slave devices must operate with the same timing. The SPI Clock Rate Select bits, SPR[2:0], in the SPCR of the master device, select the clock rate. In a slave device, the SPR[2:0] bits have no effect on the operation of the SPI. The SCK0 pin can be programmed as a general purpose I/O pin (PC2) when the SPI SCK0 function is not being used.

- **\overline{SS} 0/PC3 (SPI0 Slave Select)**—The slave select (\overline{SS}) input of a slave device must be externally asserted before a master device can exchange data with the slave device. \overline{SS} must be low before data transactions and must stay low for the duration of the transaction. The \overline{SS} line of the master must be held high. If it goes low, a mode fault error flag (the MDF bit) is set in the SPSR register. To disable the mode fault circuit, program the \overline{SS} pin as a GPIO pin in the Port C Control register (PCC). This inhibits the mode fault flag, while the other three lines are dedicated to the SPI peripheral.

The state of the master and slave CPH bits (in the SPCR register) affects the operation of \overline{SS} . The CPH bit settings should be the same for both master and slave. When the CPH bit is set to 0, the shift clock is the OR of \overline{SS} with SCK. In this clock phase mode, \overline{SS} must go high between successive bytes in an SPI message. When the CPH bit is set to 1, \overline{SS} can be left low between successive

SPI bytes. In cases where there is only one SPI slave MCU, its \overline{SS} line can be tied to V_{SS} as long as only $CPH = 1$ clock mode is used. The $\overline{SS0}$ pin can be programmed as a general purpose I/O pin (PC3) when the SPI $\overline{SS0}$ function is not being used.

The four pins on SPI1 are as follows:

- **MISO1/PC4 (SPI1 Master In Slave Out)**—MISO1 is one of two unidirectional serial data signals. It is an input to a master device and an output from a slave device. If a slave device is not selected, the MISO0 line of the slave device is placed in the high-impedance state. MISO1 can be programmed as a general purpose I/O pin called PC4 when the SPI MISO1 function is not being used.
- **MOSI1/PC5 (SPI1 Master Out Slave In)**—The MOSI1 line is the second of the two unidirectional serial data signals. It is an output from a master device and an input to a slave device. The master device places data on the MOSI1 line a half-cycle before the clock edge that the slave device uses to latch the data. MOSI1 can be programmed as a general purpose I/O pin called PC5 when the SPI MOSI1 function is not being used.
- **SCK1/PC6 (SPI1 Serial Clock)**—SCK1, an input to a slave device, is generated by the master device and synchronizes data movement in and out of the device through the MOSI1 and MISO1 lines. Master and slave devices are capable of exchanging a byte of information during a sequence of eight clock cycles. Slave devices ignore the SCK signal unless the slave select pin is active low.

Four possible timing relationships can be chosen by using control bits CPOL and CPH in the SPCR register. Both master and slave devices must operate with the same timing. The SPI clock rate select bits, SPR[2:0], in the SPCR register of the master device, select the clock rate. In a slave device, the SPR[2:0] bits have no effect on the operation of the SPI. The SCK1 pin can be programmed as a general purpose I/O pin (PC6) when the SPI SCK1 function is not being used.

- **$\overline{SS1}$ /PC7 F(SPI1 Slave Select)**—The slave select (\overline{SS}) input of a slave device must be externally asserted before a master device can exchange data with the slave device. \overline{SS} must be low before data transactions and must stay low for the duration of the transaction. The \overline{SS} line of the master must be held high. If it goes low, a mode fault error flag (MDF) is set in the SPSR register. To disable the mode fault circuit, program the \overline{SS} pin as a GPIO pin in the PCC register. This inhibits the mode fault flag, while the other three lines are dedicated to the SPI peripheral.

The state of the master and slave CPH bits affects the operation of \overline{SS} . The

SPI System Errors

CPH bit settings should be the same for both master and slave. When the CPH bit is set to 0, the shift clock is the OR of \overline{SS} with SCK. In this clock phase mode, \overline{SS} must go high between successive bytes in an SPI message. When the CPH bit is set to 1, \overline{SS} can be left low between successive SPI bytes. In cases where there is only one SPI slave MCU, its \overline{SS} line can be tied to V_{SS} as long as only CPH = 1 clock mode is used. $\overline{SS1}$ can be programmed as a general purpose I/O pin (PC7) when the SPI $\overline{SS1}$ function is not being used.

SPI output pins can be configured as open-drain drivers. The WOM bit in the SPCR register is used to enable this option in each SPI. An external pullup resistor is required on each Port C output pin while this option is selected. In multiple-master systems, this option provides extra protection against CMOS latchup, because even if more than one SPI device tries to simultaneously drive the same bus line, there can be no destructive contention.

In some multiple SPI systems where one device is always the master device, it may make sense to bypass the Slave Select function and instead use this pin as a GPIO to drive the slave select of one of the slave SPI devices.

Note: Bypassing Slave Select disables the mode-fault detection capability.

Signal timing is described in the *DSP56L811 Data Sheet (DSP568L11/D)*.

7.5 SPI SYSTEM ERRORS

Two system errors can be detected by the SPI system. The first type of error arises in a multiple-master system when more than one SPI device simultaneously tries to be a master. This error is called a mode-fault error. The second type of error, a write-collision error, indicates that an attempt was made to write data to an SPDR register while a transfer was in progress.

7.5.1 SPI Mode-Fault Error

When the SPI system is configured as a master and the \overline{SS} input line goes to active low, a mode fault error has occurred, usually because two devices attempted to act as master at the same time. Only an SPI master can experience a mode-fault error. In cases where more than one device is concurrently configured as a master, a chance of contention between two pin drivers exists. For push-pull CMOS drivers, this contention can cause permanent damage. The mode fault attempts to protect the

device by disabling the drivers. When this error is detected, the following actions are taken immediately:

1. The SPI pins are reconfigured as all inputs.
2. The MST control bit is forced to 0 to reconfigure the SPI as a slave.
3. The SPE control bit is forced to 0 to disable the SPI system.
4. The MDF flag is set and an SPI interrupt is generated subject to masking by the SPIE bit, the appropriate bit in the IPR register, and the I1 bit in the SR register.

Note: The SPI pins are reconfigured as inputs regardless of the values of the corresponding PCDDR register bits. This condition is removed once the SPE bit is set again, and the pins resume their normal SPI functionality.

After software has corrected the problems that led to the mode fault, the MDF bit is cleared and the system returns to normal operation. The MDF bit is automatically cleared by reading the SPSR register while the MDF bit is set, followed by a write to the SPCR register.

Other precautions may need to be taken to prevent driver damage. If two devices are made masters at the same time, mode fault does not help protect either one unless one of them selects the other as slave. The amount of damage possible depends on the length of time both devices attempt to act as master.

Because the \overline{SS} pin is used to detect mode fault, it should not be configured as a GPIO pin on an SPI system where more than one SPI is capable of functioning as an SPI master. Maintaining the slave select functionality can help prevent driver damage if mode fault occurs.

7.5.2 SPI Write-Collision Error

A write-collision error occurs if the SPDR register is written while a transfer is in progress. Because the SPDR register is not double-buffered in the transmit direction, writes to the SPDR register cause data to be written directly into the SPI shift register. Because this write corrupts any transfer in progress, a write-collision error is generated. The transfer continues undisturbed, and the write data that caused the error is not written to the shifter. A write collision is normally a slave error because a slave has no control over when a master initiates a transfer. A master knows when a transfer is in progress, so there is no reason for a master to generate a write-collision error, although the SPI logic can detect write collisions in both master and slave devices.

Configuring Port C for SPI Functionality

The SPI configuration determines the characteristics of a transfer in progress. For a master, a transfer begins when data is written to the SPDR register and ends when the SPIF bit is set. For a slave with the CPH bit set to 0, a transfer starts when the \overline{SS} pin goes low and ends when the \overline{SS} pin returns high. In this case, the SPIF bit is set at the middle of the eighth SCK cycle (when data is transferred from the shifter to the parallel data register) but the transfer is still in progress until \overline{SS} goes high.

For a slave with the CPH bit set to 1, transfer begins when the SCK line goes to its active level, which is the edge at the beginning of the first SCK cycle. In this case, the transfer ends when the SPIF bit is set.

7.5.3 SPI Overrun

No mechanism is provided in the SPI for detecting overrun. Overrun is the condition where a second sample has been shifted in and transferred to the Receive Data Buffer before a first sample has been read from the Receive Data Buffer.

Overrun is avoided by reading the SPDR register from the previous transfer before the start of the eighth cycle of the current transfer. Any data from a previous transfer becomes invalid at the start of the eighth cycle.

7.6 CONFIGURING PORT C FOR SPI FUNCTIONALITY

The Port C Control (PCC) register is used to individually configure each pin as either an SPI pin or a GPIO pin. When the SPI is used in slave mode, all four SPI pins are programmed as SPI pins in the PCC register. When in master mode, the SCK, MISO, and MOSI pins are configured as SPI pins, and the \overline{SS} pin is optionally configured as an SPI pin if mode fault detection is desired. Otherwise, \overline{SS} can be configured as a GPIO pin in the PCC register. This is summarized in **Table 7-4**.

Table 7-4 PCC Register Programming for the \overline{SS} Pin

Mode	CC Bit	Functionality of the \overline{SS} Pin
Slave	0	Not allowed
Slave	1	Used as slave select pin
Master	0	Used as GPIO pin
Master	1	Used for mode fault detection

When the PCC register bit is set to 1 for an SPI pin, it is not necessary to program the corresponding PCDDR register bit. The SPI peripheral ensures the correct direction of this pin. Programming the PCDDR register is necessary only when a pin is programmed as a GPIO pin.

7.6.1 SPI Low Power Operation

In applications requiring minimum power consumption, the SPI module can be disabled by clearing the SPE bit in SPCR register. This also shuts off the Phi clock signal as it enters the block.

Stop mode automatically disables the SPI peripheral and any external clocks for devices configured in slave mode. The internal Phi clock is stopped in stop mode. If a device is in the middle of a transfer when it enters stop mode, all internal state machines are reset so that upon exiting stop mode, the device waits for a new transfer to be initiated.

In wait mode, the SPI peripheral continues operation and can complete transfers in progress. If the SPI interrupt is enabled in the SPCR, IPR, and SR registers, the SPI peripheral can generate an interrupt request in wait mode that brings the DSP56L811 out of wait mode.

7.7 PROGRAMMING EXAMPLES

The following examples show how to configure one SPI port as a master, one as a slave, and to send data from master to slave.

7.7.1 Configuring an SPI Port as Master

Example 7-1 shows how to configure an SPI port as a master.

Example 7-1 Configuring an SPI Port as Master

```

;*****
;* SPI master                                     *
;* for Serial Peripheral Interface (SPI) *
;* of DSP56L811 chip                             *
;*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
IPR        EQU    $FFFB ; Interrupt Priority Register
PCC        EQU    $FFED ; Port C Control Register
PCDDR      EQU    $FFEE ; Port C Data Direction Register
PCR0       EQU    $FFF2 ; PLL Control Register 0
PCR1       EQU    $FFF3 ; PLL Control Register 1
SPCR0      EQU    $FFE2 ; SPI0 Control Register
SPDR0      EQU    $FFE0 ; SPI0 Data Register
SPSR0      EQU    $FFE1 ; SPI0 Status Register

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).     |
;+-----+

      ORG    P:$0000      ; Cold Boot
      JMP    START        ; also Hardware RESET vector (Mode 0, 1, 3)

```

Example 7-1 Configuring an SPI Port as Master (Continued)

```
ORG    P:$E000      ; Warm Boot
JMP     START        ; Hardware RESET vector (Mode 2)

;
;   ORG    P:$0028      ;
;   JSR     [unused]    ; SPI1 Serial System vector

ORG     P:START      ; Start of program
;*****
;* General setup *
;*****

    MOVEP  #$0000,X:BCR ; External Program memory has 0 wait states.
                        ; External data memory has 0 wait states.
                        ; Port A pins are tri-stated when no
                        ; external access occurs.
;*****
;* Phi Clock included for serial bit clock of SPI Master *
;*****

    MOVEP  #$0180,X:PCR1 ; Configure:
                        ; (PLLE) PLL disabled (bypassed)
                        ; -- Oscillator supplies Phi Clock.
                        ; (PLLD) PLL Power Down disabled (PLL active).
                        ; -- PLL block active for PLL to attain lock.
                        ; (LPST) Low Power Stop disabled.
                        ; (PS[2:0]) Prescaler Clock disabled.
                        ; (CS[1:0]) Clockout pin (CLK0) sends Phi Clock.
    MOVEP  #$0260,X:PCR0 ; Set Feedback Divider to 1/20
    ; ...
    ; insert delay here: wait for PLL lock as specified in data sheet
    ; ...

    BFSET  #$4000,X:PCR1 ; Enable PLL for Phi Clock.
;*****
;* SPI Master setup *
;*****

    BFCLR  #$0040,X:SPCR0 ; (SPE) SPI disabled
    MOVEP  #$0116,X:SPCR0 ; Configure:
                        ; (SPR[2:0]) SPI Clock Rate Select at /64.
                        ; (SPIE) SPI Interrupt disabled.
```

Example 7-1 Configuring an SPI Port as Master (Continued)

```

; (WOM) Wired-OR Mode disabled:
; -- push-pull drivers.
; (MST) Master mode selected.
; (CPL) serial Clock Polarity:
; -- SCK pin idles as logic low.
; (CPH) Clock Phase protocol:
; -- ~SS line can be tied low if only one slave.
BFSET #$0007,X:PCC ; Configure:
; MISO0, MOSI0, SCK0 for SPI master,
; ~SS0 as PC3 for GPIO.
; Other pins remain as previously set.
; (reset default is GPIO).
BFSET #$0008,X:PCDDR ; Configure GPIO pin PC3 as output
BFCLR #$2000,X:IPR ; Disable SPI0 interrupts
; (unnecessary but mentioned for completeness).
BFSET #$0040,X:SPCR0 ; (SPE) SPI Enabled.

;*****
;* Main routine *
;*****
; ...
TEST ; Test Loop
; ...
BRA TEST

```

7.7.2 Configuring an SPI Port as Slave

Example 7-2 shows how to configure an SPI port as a slave.

Example 7-2 Configuring an SPI Port as Slave

```

;*****
;* SPI slave *
;* for Serial Peripheral Interface (SPI) *
;* of DSP56L811 chip *
;*****
START EQU $0040 ; Start of program

```

Example 7-2 Configuring an SPI Port as Slave (Continued)

```

BCR          EQU    $FFF9 ; Bus Control Register
IPR          EQU    $FFFB ; Interrupt Priority Register
PCC          EQU    $FFED ; Port C Control Register
;PCDDR [unused] EQU $FFEE; Port C Data Direction Register
SPCR1        EQU    $FFE6 ; SPI1 Control Register
SPDR1        EQU    $FFE4 ; SPI1 Data Register
SPSR1        EQU    $FFE5 ; SPI1 Status Register

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then  |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+

          ORG    P:$0000          ; Cold Boot
          JMP    START            ; also Hardware RESET vector (Mode 0, 1, 3)

          ORG    P:$E000          ; Warm Boot
          JMP    START            ; Hardware RESET vector (Mode 2)

;      ORG    P:$002A          ;
;      JSR    [unused]          ; SPI0 Serial System vector

          ORG    P:START          ; Start of program

;*****
;* General setup *
;*****

          MOVEP  #$0000,X:BCR      ; External Program memory has 0 wait states.
                                   ; External data memory has 0 wait states.
                                   ; Port A pins are tri-stated when no
                                   ; external access occurs.

;*****
;* SPI Slave setup *
;*****

```

Example 7-2 Configuring an SPI Port as Slave (Continued)

```
BFCLR #$0040,X:SPCR1 ; (SPE) SPI disabled.
MOVEP #$0004,X:SPCR1 ; Configure:
                        ; (SPR) SPI Clock Rate Select at /1
                        ; -- [irrelevant but mentioned for completeness].
                        ; (SPIE) SPI Interrupt disabled.
                        ; (WOM) Wired-OR Mode disabled:
                        ; -- push-pull drivers
                        ; (MST) Master mode: off (slave mode selected)
                        ; (CPL) serial Clock Polarity:
                        ; -- SCK pin idles as logic low
                        ; (CPH) Clock Phase protocol:
                        ; -- ~SS line can be tied low if only one slave.
BFSET #$00F0,X:PCC ; Configure:
                        ; MISO1, MOSI1, SCK1, ~SS1 for SPI slave
                        ; (~SS0 required for slave mode)
                        ; other pins remain as previously set
                        ; (reset default is GPIO)
;      [PCDDR unused]      ; SPI ensures correct direction of used SPI pins.
BFCLR #$1000,X:IPR ; Disable SPI1 interrupts
                        ; (unnecessary but mentioned for completeness).
BFSET #$0040,X:SPCR1; (SPE) SPI Enabled.

;*****
;* Main routine *
;*****
      ; ...
TEST ; Test Loop
      ; ...
      BRA  TEST
```

7.7.3 Sending Data from Master to Slave

Example 7-3 shows how to send data from an SPI port configured as a master to an SPI port configured as a slave.

Example 7-3 Sending Data from Master to Slave

```

;*****
;* Transfer from SPI master to SPI slave *
;* for Serial Peripheral Interface (SPI) *
;* of DSP56L811 chip                      *
;*****
;* SPI0: Master                          *
;* SPI1: Slave                          *
;*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
IPR        EQU    $FFFB ; Interrupt Priority Register
PCC        EQU    $FFED ; Port C Control Register
PCDDR      EQU    $FFEE ; Port C Data Direction Register
PCR0       EQU    $FFF2 ; PLL Control Register 0
PCR1       EQU    $FFF3 ; PLL Control Register 1
SPCR0      EQU    $FFE2 ; SPI0 Control Register
SPCR1      EQU    $FFE6 ; SPI1 Control Register
SPDR0      EQU    $FFE0 ; SPI0 Data Register
SPDR1      EQU    $FFE4 ; SPI1 Data Register
SPSR0      EQU    $FFE1 ; SPI0 Status Register
SPSR1      EQU    $FFE5 ; SPI1 Status Register

;*****
;* Vector setup *
;*****
;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set  |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then    |
;|      jumps to first location of internal program RAM (P:$0000).      |
;+-----+

```

```

ORG    P:$0000          ; Cold Boot
JMP     START           ; also Hardware RESET vector (Mode 0, 1, 3)


ORG    P:$E000          ; Warm Boot
JMP     START           ; Hardware RESET vector (Mode 2)


;      ORG    P:$0028      ;
;      JSR     [unused]   ; SPI1 Serial System vector


;      ORG    P:$002A      ;
;      JSR     [unused]   ; SPI0 Serial System vector


ORG     P:START         ; Start of program
;*****
;* General setup *
;*****

MOVEP  #$0000,X:BCR     ; External Program memory has 0 wait states.
                        ; External data memory has 0 wait states.
                        ; Port A pins are tri-stated when no external
                        ; access occurs.
;*****
;* Phi Clock included for serial bit clock of SPI Master *
;*****

MOVEP  #$0180,X:PCR1    ; Configure:
                        ; (PLLE) PLL disabled (bypassed)
                        ; -- Oscillator supplies Phi Clock.
                        ; (PLLDD) PLL Power Down disabled (PLL active)
                        ; -- PLL block active for PLL to attain lock.
                        ; (LPST) Low Power Stop disabled.
                        ; (PS[2:0]) Prescaler Clock disabled.
                        ; (CS[1:0]) Clockout pin (CLKO) sends Phi Clock.
MOVEP  #$0260,X:PCRO    ; Set Feedback Divider to 1/20.
; ...
; insert delay here: wait for PLL lock as specified in data sheet.
; ...

BFSET  #$4000,X:PCR1    ; Enable PLL for Phi Clock.
```

Example 7-3 Sending Data from Master to Slave (Continued)

```
;*****
;* SPI Master setup *
;*****

BFCLR #$0040,X:SPCR0 ; (SPE) SPI disabled.
MOVEP #$0116,X:SPCR0 ; Configure:
                        ; (SPR[2:0]) SPI Clock Rate Select at /64.
                        ; (SPIE) SPI Interrupt disabled.
                        ; (WOM) Wired-OR Mode disabled:
                        ; -- push-pull drivers.
                        ; (MST) Master mode selected.
                        ; (CPL) serial Clock Polarity:
                        ; -- SCK pin idles as logic low.
                        ; (CPH) Clock Phase protocol:
                        ; -- ~SS line can be tied low if only one slave.

BFSET #$0007,X:PCC     ; Configure:
                        ; MISO0, MOSI0, SCK0 for SPI master,
                        ; ~SS0 as PC3 for GPIO.
                        ; Other pins remain as previously set
                        ; (reset default is GPIO).

BFSET #$0008,X:PCDDR ; Configure GPIO pin PC3 as output.
BFCLR #$2000,X:IPR    ; Disable SPI0 interrupts
                        ; (unnecessary but mentioned for completeness).

BFSET #$0040,X:SPCR0 ; (SPE) SPI Enabled.
;*****
;* SPI Slave setup *
;*****

BFCLR #$0040,X:SPCR1; (SPE) SPI disabled.
MOVEP #$0004,X:SPCR1; Configure:
                        ; (SPR) SPI Clock Rate Select at /1
                        ; -- [irrelevant but mentioned
                        ;    for completeness]
                        ; (SPIE) SPI Interrupt disabled.
                        ; (WOM) Wired-OR Mode disabled:
                        ; -- push-pull drivers.
                        ; (MST) Master mode: off (slave mode selected).
                        ; (CPL) serial Clock Polarity:
                        ; -- SCK pin idles as logic low
```

Example 7-3 Sending Data from Master to Slave (Continued)

```

; (CPH) Clock Phase protocol:
; -- ~SS line can be tied low if only one slave
BFSET #$00F0,X:PCC ; Configure:
; MISO1, MOSI1, SCK1, ~SS1 for SPI slave
; (~SS0 required for slave mode).
; Other pins remain as previously set
; (reset default is GPIO).
; [PCDDR unused] ; SPI ensures correct direction of used SPI pins.
BFCLR #$1000,X:IPR ; Disable SPI1 interrupts.
; (unnecessary but mentioned for completeness).
BFSET #$0040,X:SPCR1; (SPE) SPI Enabled.

;*****
;* Main routine *
;*****
;+-----+
;| This example serves to illustrate the mechanics of transfer. |
;| Transfers can also be done with interrupts instead of polling. |
;| Data is transferred one byte at a time, lower byte first. |
;| SPI data lines are connected via the Master Out Slave In pins. |
;+-----+
        MOVE  #$0000,X0 ; Clear X0 to initialize output pattern
XLOOP ; Transfer Loop
        MOVE  X0,A1 ; Load output pattern into A1.
        DO    #2,XFER ; Send two bytes (lower byte of A1 goes first).
        JSR   TXBYTE ; Transmit byte out on MOSI0 line.
        JSR   RXBYTE ; Receive byte in from MOSI1 line.
XFER
        CMP   X0,B0 ; Did the data come back all right?
        BNE   XLOOP ; No, something is wrong with connection.
; -- Try again until successful.
        INC   X0 ; Increment X0 for next output pattern.
        BRA   XLOOP

TXBYTE ; Transmit Byte
        MOVEP X:SPSR0,A0 ; Read SPSR0 to clear SPIF flag so SPI can write
; to SPDR0, otherwise write is inhibited.

```

Example 7-3 Sending Data from Master to Slave (Continued)

```
MOVEP A1,X:SPDR0      ; Transmit lower byte of data from A1.

REP    #8              ; Shift upper byte of data for next transfer
ASR    A               ; (only needed during first pass of XFER loop).

RTS

RXBYTE ; Receive Byte
    BFTSTH#$0080,X:SPSR1 ; Test for SPIF flag high in Slave
    BCC    RXBYTE       ; Wait until data ready to be received

    MOVEP X:SPSR1,B1     ; Read SPSR1 to clear SPIF flag (for status)
    MOVEP X:SPDR1,B1     ; Receive data into B1

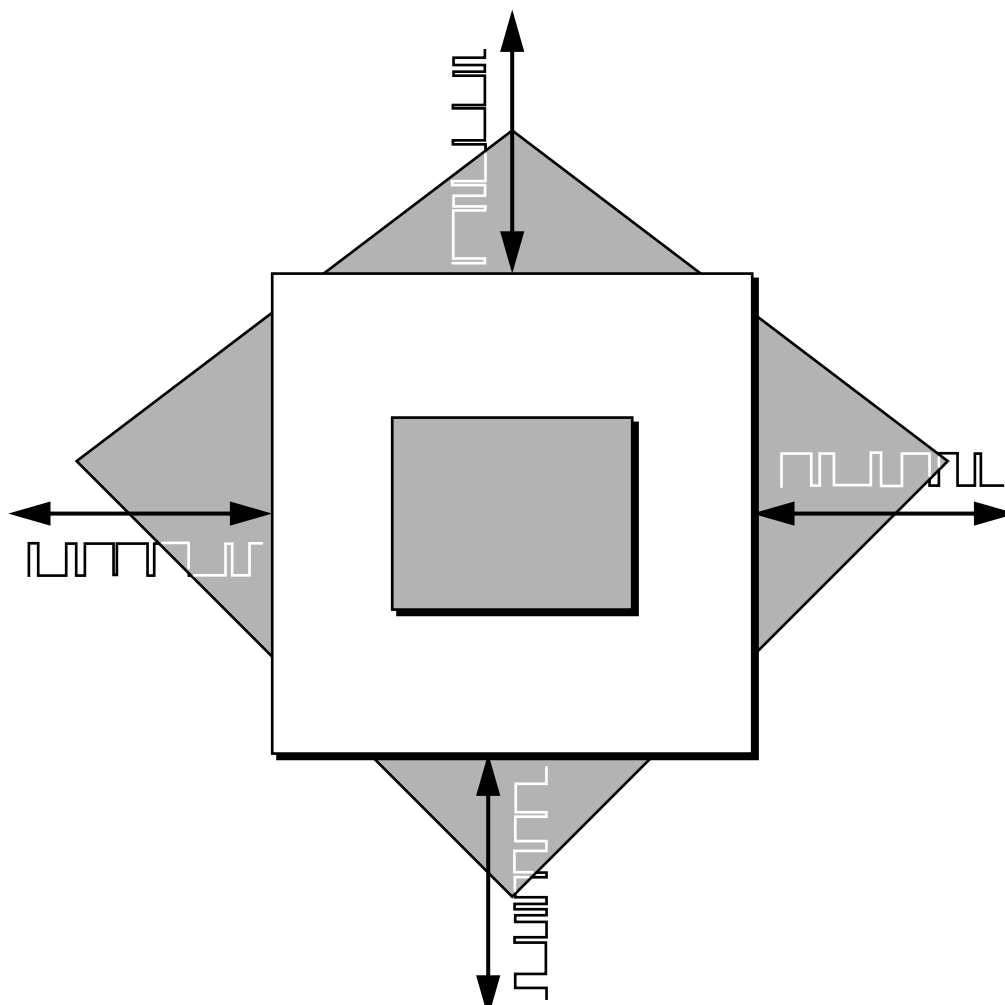
    REP    #8            ;
    ASR    B             ; Shift data into B0

    RTS
```



SECTION 8

SYNCHRONOUS SERIAL INTERFACE



8.1	INTRODUCTION	8-3
8.2	SSI ARCHITECTURE	8-4
8.3	SSI PROGRAMMING MODEL	8-8
8.4	SSI DATA AND CONTROL PINS	8-22
8.5	SSI OPERATING MODES.	8-26
8.6	SSI RESET AND INITIALIZATION PROCEDURE	8-35
8.7	CONFIGURING PORT C FOR SSI FUNCTIONALITY	8-36

8.1 INTRODUCTION

This section presents the Synchronous Serial Interface (SSI), and discusses the architecture, the programming model, the operating modes, and initialization of the SSI.

The SSI is a full-duplex serial port that allows the DSP56L811 to communicate with a variety of multiple serial devices, including industry-standard codecs, other DSPs, microprocessors, and peripherals that implement the Motorola SPI interface. It is typically used to transfer samples in a periodic manner. The SSI interface consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

The capabilities of the SSI include:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs
- Normal mode operation using frame sync
- Network mode operation with as many as 32 time slots
- Gated clock mode operation requiring no frame sync
- Programmable internal clock divider
- Programmable word length (8, 10, 12, or 16 bits)
- Program options for frame sync and clock generation
- SSI power-down feature

8.2 SSI ARCHITECTURE

Figure 8-1 shows the SSI provided on Port C of the DSP56L811.

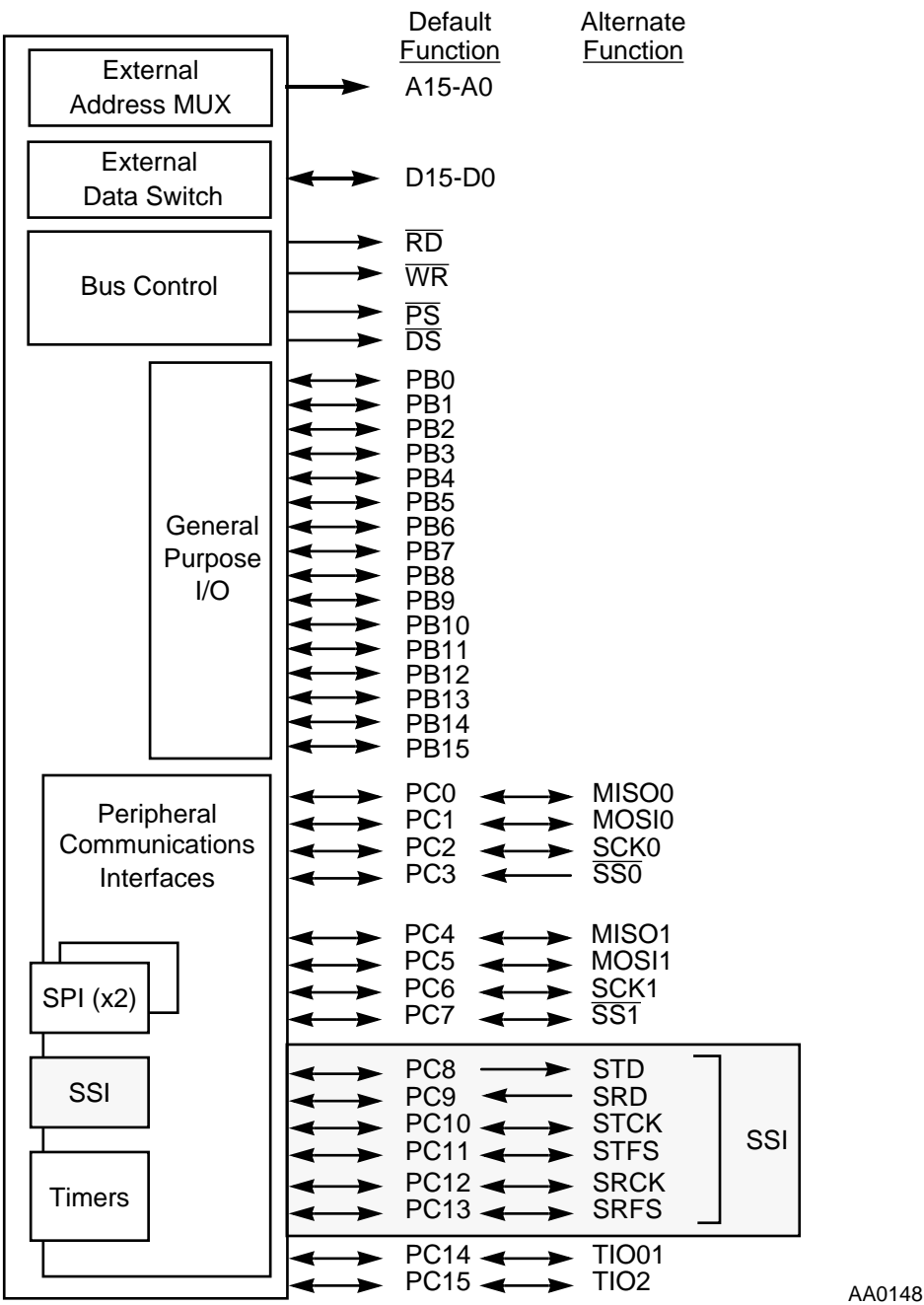


Figure 8-1 DSP56L811 Input/Output Block Diagram

Figure 8-2 shows a block diagram of the SSI. It consists of three control registers to set up the port, one status register, separate transmit and receive circuits with buffer registers, and separate serial clock and frame sync generation for the transmit and receive sections.

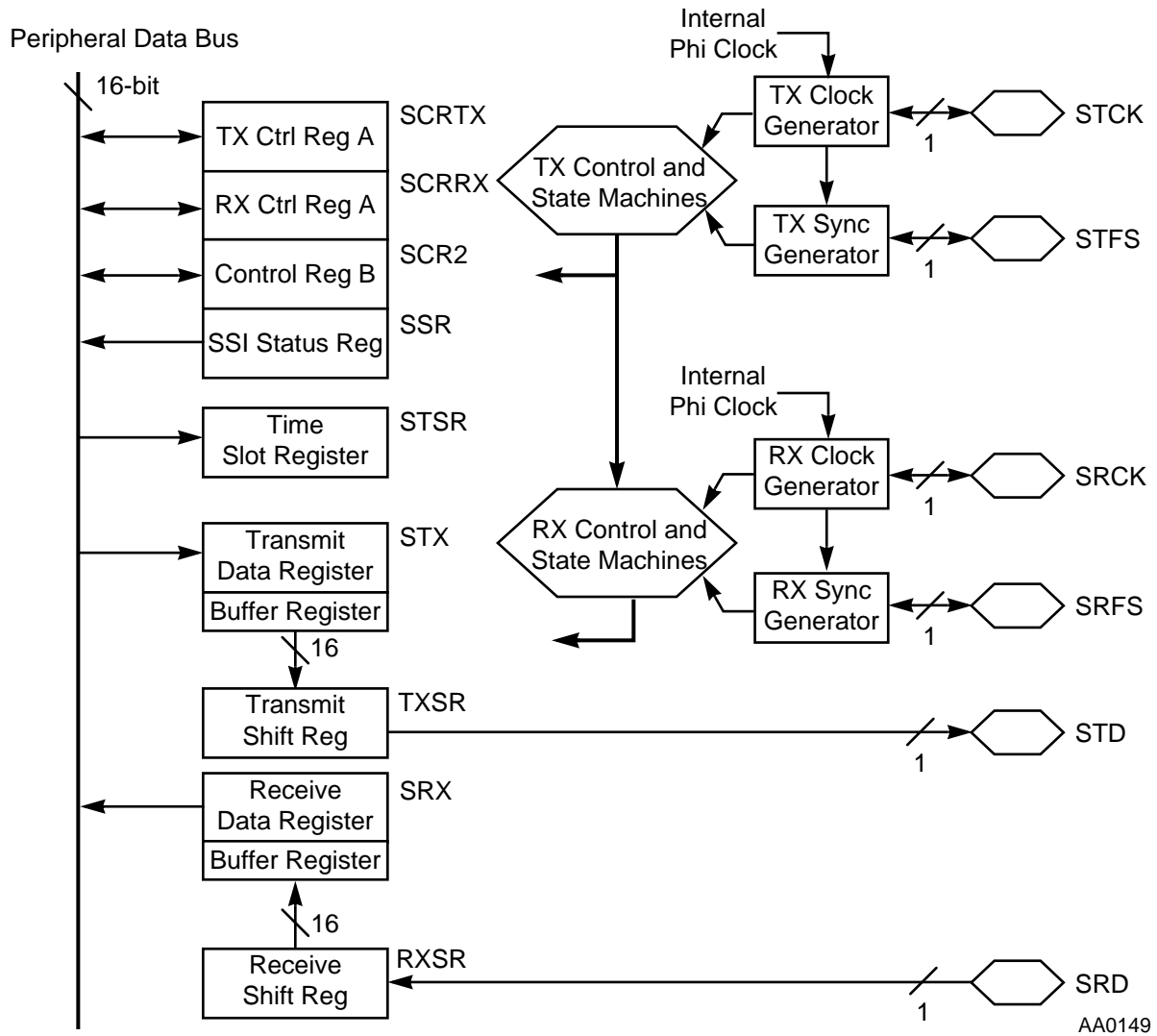


Figure 8-2 SSI Block Diagram

8.2.1 SSI Clocking

The SSI uses the following three clocks:

- Bit clock—Used to serially clock the data bits in and out of the SSI port
- Word clock—Used to count the number of data bits per word (8, 10, 12, or 16 bits)
- Frame clock—Used to count the number of words in a frame

The bit clock, used to serially clock the data, is visible on the STCK and SRCK pins. The word clock is an internal clock used to determine when transmission of an 8, 10, 12, or 16 bit word has completed. The word clock in turn then clocks the frame clock, which counts the number of words in the frame. The frame clock can be viewed on the STFS and SRFS frame sync pins, since a frame sync is generated after the correct number of words in the frame have passed. The relationship between the clock is shown in **Figure 8-3**. The bit clock can be received from an SSI clock pin or can be generated from the Phi clock through a divider, as shown in **Figure 8-4**.

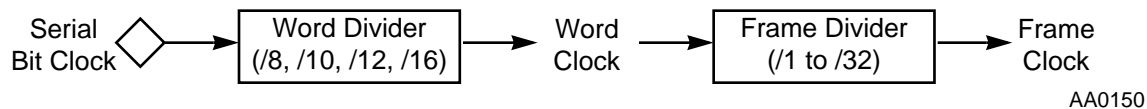


Figure 8-3 SSI Clocking

8.2.2 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally by the DSP56L811, or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the Phi clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation. In gated clock mode, the data clock is valid only when data is being transmitted. Otherwise the clock pin is tri-stated. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

Figure 8-4 shows a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the TXD bit in the SCR2 control register. The receive section contains an equivalent clock generator circuit.

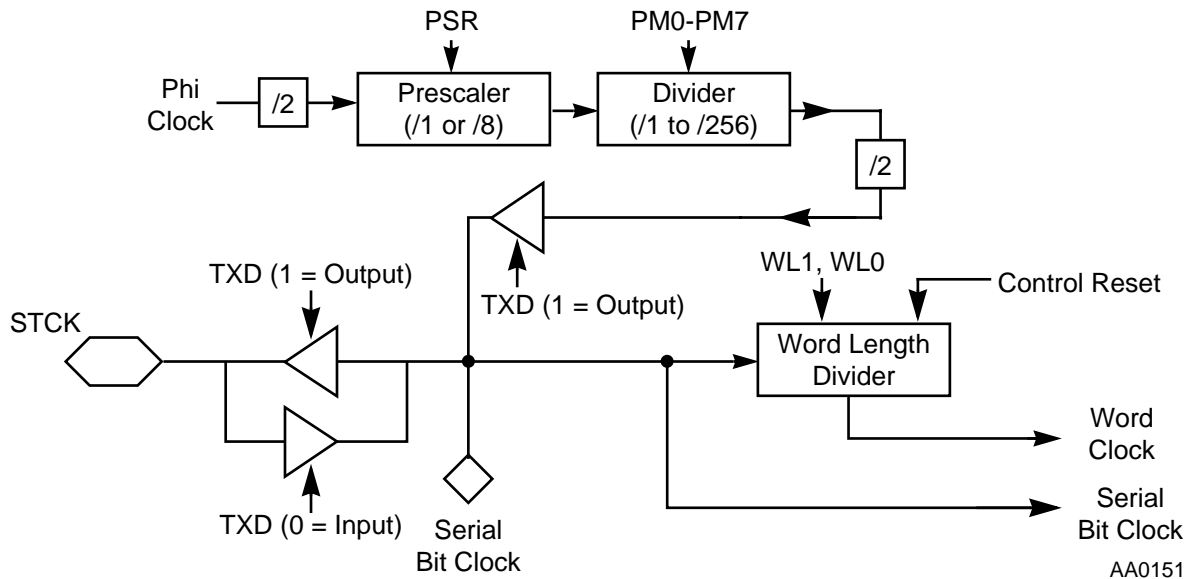


Figure 8-4 SSI Transmit Clock Generator Block Diagram

Figure 8-5 shows the Frame Sync Generator block for the transmit section. When internally generated, both receive and transmit frame sync are generated from the word clock, and are defined by the frame rate divider (DC[4:0]) bits and the word length (WL[1:0]) bits of the SCRTX register. The receive section contains an equivalent circuit for the Frame Sync Generator.

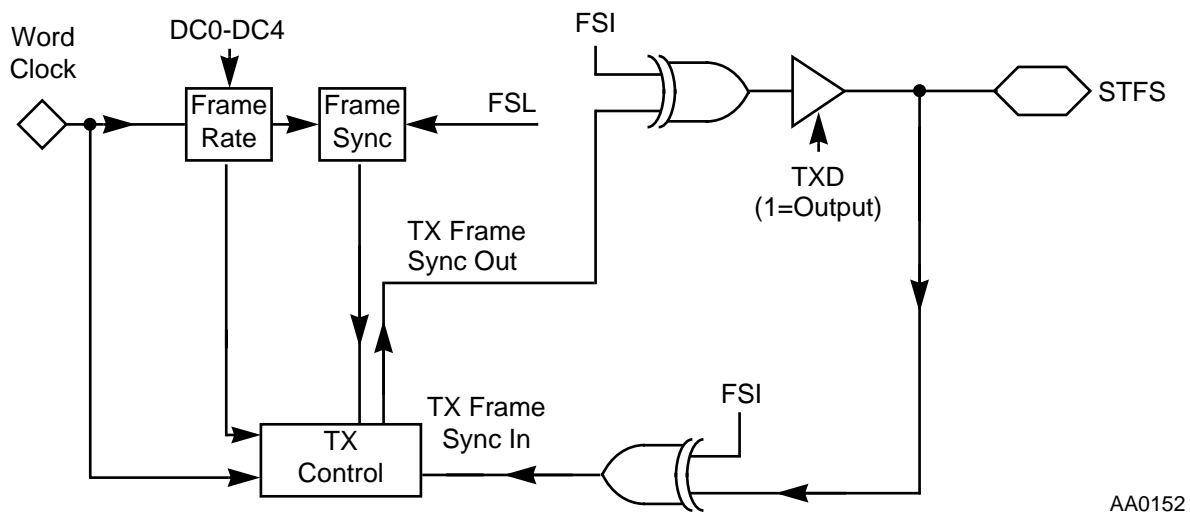


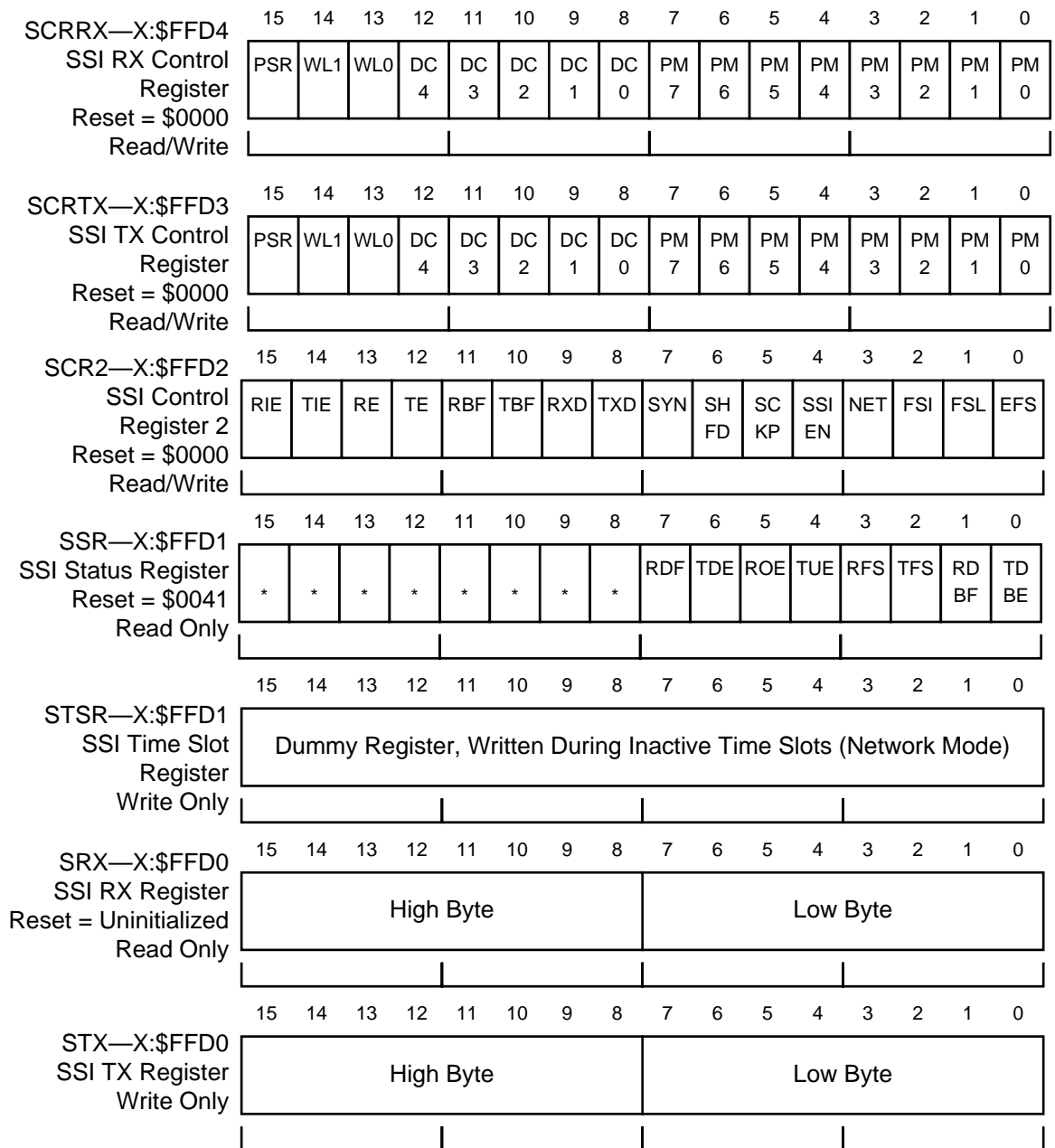
Figure 8-5 SSI Transmit Frame Sync Generator Block Diagram

8.3 SSI PROGRAMMING MODEL

The registers associated with the SSI include the following:

- SSI Transmit Shift Register (TXSR)
- SSI Receive Shift Register (RXSR)
- SSI Receive Data Buffer Register
- SSI Receive Data Register (SRX, read-only)
- SSI Transmit Control Register (SCRTX)
- SSI Receive Control Register (SCRRX)
- SSI Control Register 2 (SCR2)
- SSI Status Register (SSR, read-only)
- SSI Time Slot Register (STSR, write-only)
- SSI Transmit Data Register (STX, write-only)
- SSI Transmit Data Buffer Register

The control registers associated with the SSI interface are shown in **Figure 8-6**; **Figure 8-7** shows the programming information for SSI interrupts. **Table 3-3 Interrupt Priority Structure** on page 3-16 lists the interrupt priority order for the DSP56L811.



* Indicates reserved bits, written as 0 for future compatibility

AA0156

Figure 8-6 SSI Programming Model—Register Set

SSI Interrupt Vectors:

SSI Receive Data with Exception Status	P:\$0020
SSI Receive Data	P:\$0022
SSI Transmit Data with Exception Status	P:\$0024
SSI Transmit Data	P:\$0026

Enabling SSI Interrupts in the Interrupt Priority Register:

Set Bit 9 to 1 in the Interrupt Priority Register (X:\$FFFB)

AA0157

Figure 8-7 SSI Interrupt Vectors

8.3.1 SSI Transmit Shift Register (TXSR)

The SSI Transmit Shift Register (TXSR) is a 16-bit shift register that contains the data being transmitted. When a continuous clock is used, data is shifted out to the STD pin by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the STD pin by the selected (internal/external) gated clock. The Word Length control bits (WL[1:0]) in the SCRTX register (described in **SSI Transmit and Receive Control Registers (SCRTX and SCRRX)** on page 8-12) determine the number of bits to be shifted out of the TXSR register before it is considered empty and can be written to again. This word length can be 8, 10, 12, or 16 bits. The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is always shifted out of this register with the MSB first when the SHFD bit of the SCR2 is cleared. If this bit is set, the LSB is shifted out first.

8.3.2 SSI Transmit Data Buffer Register

The SSI Transmit Data Buffer Register is a 16-bit register used to buffer samples written to the STX register. It is written by the contents of the STX register whenever the transmit buffer feature is enabled. When enabled, the TXSR register receives its values from this buffer register. If the transmit buffer feature is not enabled, this register is bypassed and the contents of the STX register is transferred into the TXSR register.

When the TIE bit in the SCR2 register and TDE bit in the SSR register are set, the DSP56L811 is interrupted whenever both the STX register and the SSI Transmit Data Buffer Register become empty.

8.3.3 SSI Transmit Data Register (STX)

The SSI Transmit Data Register (STX) is a 16-bit write-only register. Data to be transmitted is written into this register. If the transmit buffer is used, data is transferred from this register to the Transmit Data Buffer register when it becomes empty. Otherwise, data written to this register is transferred to the TXSR register when shifting of previous data is completed. The data written occupies the most significant portion of the STX register. The unused bits (least significant portion) of the STX register are ignored. The DSP56L811 is interrupted whenever the STX register becomes empty (when both the STX register and SSI Transmit Data Buffer register are empty, if buffering is enabled) if both the TDE bit in the SSR register and the TIE bit in the SCR2 register are set.

8.3.4 SSI Receive Shift Register (RXSR)

The SSI Receive Shift Register (RXSR) is a 16-bit shift register that receives incoming data from the serial receive data SRD pin. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock. Data is assumed to be received MSB first if the SHFD bit of the SCR2 register is cleared. If this bit is set, the data is received LSB first. Data is transferred to the SSI Receive Data Register (SRX) or Receive Data Buffer Register if the receive buffer is enabled after 8, 10, 12, or 16 bits have been shifted in depending on the WL[1:0] control bits.

8.3.5 SSI Receive Data Buffer Register

The SSI Receive Data Buffer Register is a 16-bit buffer register used to buffer samples received in the Receive Shift Register. It is written by the Receive Shift Register, whenever the receive buffer feature is enabled, by setting the RBF bit in the SCR2 register. When enabled, the Receive Data Register then receives its values from this buffer register. The DSP56L811 is interrupted whenever both the SSI Receive Data Register and SSI Receive Data Buffer Register becomes full, if the associated interrupt

is enabled. If the receive buffer feature is not enabled, this register is bypassed and the Receive Shift Register is automatically transferred into the SRX register.

8.3.6 SSI Receive Data Register (SRX)

The SSI Receive Data Register (SRX) is a 16-bit read-only register. It accepts data contained in the Receive Data Buffer Register if receive buffering is enabled by setting the RBF bit in the SCR2 register. Otherwise, it accepts data from the Receive Shift Register as it becomes full. The data read occupies the most significant portion of the SRX register. The unused bits (least significant portion) are read as 0s. The DSP56L811 is interrupted whenever the SRX register becomes full (when both the SRX register and Receive Data Buffer Register are full if buffering is enabled), if the receive data full interrupt is enabled.

8.3.7 SSI Transmit and Receive Control Registers (SCRTX and SCRRX)

The SSI Transmit and Receive Control Registers (SCRTX and SCRRX) are two of three 16-bit read/write control registers used to direct the operation of the SSI. These registers control the SSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The SCRTX register is dedicated to the transmit section, and the SCRRX register is used for the receive section except in synchronous mode where the SCRTX register controls both the receive and transmit sections. The DSP reset clears all SCRTX and SCRRX bits. SSI reset and STOP reset do not affect the SCRTX and SCRRX bits. The control bits are described in the following paragraphs.

Although the bit patterns of the SCRRX and SCRTX registers are the same, the contents of these two registers can be programmed differently. See **Figure 8-6** on page 8-9 for the programming models of the SCRTX and SCRRX registers.

8.3.7.1 Prescaler Range (PSR) Bit 15

The Prescaler Range control bit (bit 15) controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is desired. When the PS bit is cleared, the fixed prescaler is bypassed. When the PSR bit is set, the fixed divide-by-eight prescaler is operational. This allows a 128 kHz master clock to be generated for Motorola MC1440x series codecs. The maximum internally generated bit clock frequency is $F_{osc}/4$ and the minimum internally generated bit clock frequency is $F_{osc}/(4 \times 8 \times 256)$.

8.3.7.2 Word Length Control (WL1-WL0) Bits 14-13

The Word Length Control bits (bits 14-13) are used to select the length of the data words being transferred by the SSI. Word lengths of 8, 10, 12 or 16 bits can be selected as shown in **Table 8-1**.

Table 8-1 SSI Data Word Lengths

WL1	WL0	Number of bits/word
0	0	8
0	1	10
1	0	12
1	1	16

These bits control the Word Length Divider shown in the SSI Clock Generator. The WL control bits also control the frame sync pulse length when the FSL bit is set to 0.

8.3.7.3 Frame Rate Divider (DC4-DC0) Bits 12-8

The Frame Rate Divider Control bits (bits 12 - 8) control the divide ratio for the programmable frame rate dividers. The divide ratio operates on the word clock. In normal mode, this ratio determines the word transfer rate. In network mode this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 (DC[4:0] = 00000 to 11111) in normal mode and from 2 to 32 (DC[4:0] = 00001 to 11111) in network mode.

8.3.7.4 Prescale Modulus Select (PM7-PM0) Bits 7-0

The Prescale Modulus Select control bits (bits 7-0) specify the divide ratio of the prescale divider in the SSI clock generator. This prescaler is used only in internal clock mode to divide the internal clock of the core. A divide ratio from 1 to 256 (PM[7:0] = \$00 to \$FF) can be selected. The bit clock output is available at the clock SCK. The bit clock on the SSI can be calculated from the Phi clock value using the equation in **Figure 8-8**.

$$SCK = \text{Phi Clock Frequency} / (4 \times (7 \times PSR + 1) \times (PM + 1))$$

$$\text{where } PM = PM[7:0]$$

AA0215

Figure 8-8 SSI Bit Clock Equation

SSI Programming Model

For example, in 8-bit word normal mode, with DC[4:0] set to 1 (00001), PM[7:0] set to 71 (0100 0111), and the PSR bit set to 0, and a 36.864 MHz Phi clock, a bit clock rate of $36.864 \text{ MHz} \div [1 \times 4 \times 72] = 128 \text{ kHz}$ is generated. Since the 8-bit word rate is equal to two, the sampling rate (FS rate) would then be $128 \text{ kHz} \div [2 \times 8] = 16 \text{ kHz}$.

The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. Careful choice of the crystal oscillator frequency and the prescaler modulus allows the telecommunication industry standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz to be generated. For example, a 24.576 MHz clock frequency can be used to generate the standard 2.048 MHz and 1.536 MHz rates, and a 24.704 MHz clock frequency can be used to generate the standard 1.544 MHz rate. **Table 8-2** gives examples of PM[7:0] values that can be used in order to generate different bit clocks.

Table 8-2 SSI Bit Clock as a Function of Phi Clock and Prescale Modulus

Phi Clock (MHz)	Max bit clock (MHz)	PM[7:0] Values for different SCK				
		2.048 MHz	1.544 MHz	1.536 MHz	128 kHz	64 kHz
16.384	4.096	1	—	—	31 (\$1F)	63 (\$3F)
18.432	4.608	—	—	2	35 (\$23)	71 (\$47)
20.480	5.12	—	—	—	39 (\$27)	79 (\$4F)
26.624	6.656	—	—	—	51 (\$33)	103 (\$67)
24.576	6.144	2	—	3	47 (\$2F)	95 (\$5F)
24.704	6.176	—	3	—	—	—
32.768	8.192	3	—	—	63 (\$3F)	127 (\$7F)
36.864	9.216	—	—	5	71 (\$47)	143 (\$8F)

8.3.8 SSI Control Register 2 (SCR2)

The SSI Control Register 2 (SCR2) is one of three 16 bit read/write control registers used to direct the operation of the SSI. The SSI reset is controlled by a bit in SCR2. SCR2 controls the direction of the bit clock and frame sync pins, STCK, SRCK, STFS, and SRFS. Interrupt enable bits for the receive and transmit sections are provided in this control register. SSI operating modes are also selected in this register. The DSP reset clears all SCR2 bits. However, SSI reset and STOP reset do not affect the SCR2 bits. The SCR2 bits are described in the following paragraphs. See **Figure 8-6** on page 8-9 for the programming model of the SCR2 register.

As with all on-chip peripheral interrupts for the DSP56L811, the SR register must first be set to enable maskable interrupts (interrupts of level IPL1). Next, the CH6 bit (bit 9) in the Interrupt Priority Register (IPR) must be set to 1 to enable the interrupt. Finally the interrupt can be enabled from within the SSI.

8.3.8.1 Receive Interrupt Enable (RIE) Bit 15

The SSI Receive Interrupt Enable control bit (bit 15) allows interrupting the program controller. When the RIE bit is set, the program controller is interrupted when the SSI Receive Data Register Full (RDF) bit in the SSR register is set.

When the receive buffer is enabled, two values are available to be read. If not enabled, then one value can be read from the SRX register. If the RIE bit is cleared, this interrupt is disabled. However, the RDF bit still indicates the receive data register full condition. Reading the SRX register clears the RDF bit, thus clearing the pending interrupt.

Two receive data interrupts, with separate interrupt vectors, are available: receive data with exception status, and receive data without exception. **Table 8-3** shows these vectors and the conditions under which these interrupts are generated.

Table 8-3 SSI Receive Data Interrupts

Interrupt	Vector	RIE	ROE	RDF
Receive Data with exception status	\$0020	1	1	1
Receive Data without exception	\$0022	1	0	1

8.3.8.2 Transmit Interrupt Enable (TIE) Bit 14

The SSI Transmit Interrupt Enable control bit (bit 14) allows interrupting the program controller. When the TIE bit is set, the program controller is interrupted when the SSI Transmit Data Register Empty flag (TDE) in the SSR register is set.

When the transmit buffer is enabled, two values can be written to the SSI. If not enabled, then one value can be written to the STX register. When the TIE bit is cleared, this interrupt is disabled. However, the TDE bit always indicates the transmit data register empty condition, even when the transmitter is disabled by the TE bit (in the SCR2 register). Writing data to the STX or STSR register clears the TDE bit, thus clearing the interrupt.

Two transmit data interrupts, with separate interrupt vectors, are available: transmit data with exception status, and transmit data without exceptions. **Table 8-4** shows

the conditions under which these interrupts are generated and lists the interrupt vectors.

Table 8-4 SSI Transmit Data Interrupts

Interrupt	Vector	TIE	TUE	TDE
Transmit Data with exception status	\$0024	1	1	1
Transmit Data without exception	\$0026	1	0	1

8.3.8.3 Receive Enable (RE) Bit 13

The SSI Receive Enable control bit (bit 13) enables the receive portion of the SSI. When the RE bit is set, the receive portion of the SSI is enabled. When the RE bit is cleared, the receiver is disabled by inhibiting data transfer into the RX buffer. If data is being received when this bit is cleared, the rest of the word is not shifted in and transferred to the SRX register. If the RE bit is re-enabled during a time slot before the second to last bit, then that word will be received.

Note: The function of disabling and enabling the RE bit in the DSP56L811 is different from the functionality of the RE bit in the DSP56156 SSI.

8.3.8.4 Transmit Enable (TE) Bit 12

The SSI Transmit Enable control bit (bit 12) enables the transfer of the contents of the STX register to the Transmit Shift Register and also enables the internal gated clock. When the TE bit is set and a word boundary is detected, the transmit portion of the SSI is enabled. When the TE bit is cleared, the transmitter continues to send the data currently in the SSI Transmit Shift Register, and then disables the transmitter. The serial output is tri-stated and any data present in the STX register is not transmitted. In other words, data can be written to the STX register with the TE bit cleared, and the TDE bit is cleared but data is not transferred to the Transmit Shift Register. If the TE bit is cleared and then set to 1 again during the same transmitted word, the data continues to be transmitted. If the TE bit is set to 1 again during a different time slot, data is not transmitted until the next word boundary.

The normal transmit enable sequence for transmit is to write data to the STX register or to the STSR register before setting the TE bit. The normal transmit disable sequence is to clear the TE bit and the TIE bit after the TDE bit is set to 1.

When an internal gated clock is being used, the gated clock runs during valid time slots if the TE bit is set. If the TE bit is cleared, the transmitter continues to send the data currently in the SSI Transmit Shift Register until it is empty. Then the clock stops. When the TE bit is set to 1 again, the gated clock starts immediately and runs during any valid time slots.

Note: The function of disabling and enabling the TE bit in the DSP56L811 is different from the functionality of the TE bit in the DSP56156 SSI.

8.3.8.5 Receive Buffer Enable (RBF) Bit 11

The Receive Buffer Enable control bit (bit 11) enables the buffer register for the receive section. When the RBF bit is set, this allows for two samples to be received by the SSI (a third sample can be shifting in) before the RDF bit is set, and an interrupt request generated when enabled by the RIE bit. When the RBF bit is cleared, the buffer register is not used, and an interrupt request is generated when a single sample is received by the SSI.

8.3.8.6 Transmit Buffer Enable (TBF) Bit 10

The Transmit Buffer Enable control bit (bit 10) enables the buffer register for the transmit section. When the TBF bit is set, two samples can be written to the SSI (a third sample can be shifting out) before the TDE bit is set and an interrupt request generated when enabled by the TIE bit. When the TBF bit is cleared, the buffer register is not used, and an interrupt request is generated when a single sample needs to be written to the SSI.

8.3.8.7 Receive Direction (RXD) Bit 9

The Receive Direction control bit (bit 9) selects the direction and source of the clock and frame sync signals used to clock the Receive Shift Register. When the RXD bit is set, the frame sync and clock are generated internally and are output to the SRFS and SRCK pins, respectively, if not configured as GPIO. When the RXD bit is cleared, (1) the clock source is external, (2) the internal clock generator is disconnected from the SRCK pin, and (3) an external clock source can drive this pin to clock the Receive Shift Register. The SRFS pin is an input, meaning that the receive frame sync is supplied from an external source. **Table 8-5** shows the frame sync/clock pin configuration.

Table 8-5 Frame Sync / Clock Pin Configuration

SYN	RXD	TXD	SRFS	STFS	SRCK	STCK
0	0	0	RFS in	TFS in	RCK in	TCK in
0	0	1	RFS in	TFS out	RCK in	TCK out
0	1	0	RFS out	TFS in	RCK out	TCK in
0	1	1	RFS out	TFS out	RCK out	TCK out
1	0	0	GPIO	FS in	GPIO	CK in
1	0	1	GPIO	FS out	GPIO	CK out
1	1	0	GPIO	GPIO	GPIO	Gated in
1	1	1	GPIO	GPIO	GPIO	Gated out

8.3.8.8 Transmit Direction (TXD) Bit 8

The Transmit Direction control bit (bit 8) selects the direction and source of the clock and frame sync signals used to clock the Transmit Shift Register. When the TXD bit is set, the frame sync and clock are generated internally and are output to the STFS and STCK pins, respectively, if not configured as GPIO. When the TXD bit is cleared, (1) the clock source is external, (2) the internal clock generator is disconnected from the STCK pin, and (3) an external clock source can drive this pin to clock the Transmit Shift Register. The STFS pin is an input, meaning that the transmit frame sync is supplied from an external source.

8.3.8.9 Synchronous Mode (SYN) Bit 7

The Synchronous Mode control bit (bit 7) enables the synchronous mode of operation. In this mode, the transmit and receive sections share a common clock pin (STCK) and frame sync pin (STFS).

8.3.8.10 Shift Direction (MSB/LSB Position) (SHFD) Bit 6

The Shift Direction (MSB/LSB Position) control bit (bit 6) controls whether the MSB or LSB is transmitted and received first. If the SHFD bit is set to 0, the data is transmitted and received MSB first. If the SHFD bit is set to 1, the LSB is transmitted and received first.

Note: The codec device labels the MSB as bit 0, whereas the DSP56L811 labels the LSB as bit 0. Therefore, when using a standard codec, the DSP56L811 MSB (or codec bit 0) is shifted out first, and the SHFD bit should be cleared.

8.3.8.11 Clock Polarity (SCKP) Bit 5

The Clock Polarity control bit (bit 5) controls which bit clock edge is used to clock out data and latch in data. If the SCKP bit is set to 0, the data is clocked out on the rising edge of the bit clock and latched in on the falling edge of the clock. If the SCKP bit is set to 1, the falling edge of the clock is used to clock the data out and the rising edge of the clock is used to latch the data in.

8.3.8.12 SSI Enable (SSIEN) Bit 4

The SSI Enable control bit (bit 4) enables and disables the SSI. If the SSIEN bit is set to 1, the SSI is enabled, which causes an output frame sync to be generated when set up for internal frame sync, or causes the SSI to wait for the input frame sync when set up for external frame sync. If the SSIEN bit is set to 0, the SSI is disabled. When disabled, the output clock and frame sync are tri-stated, the status register bits are preset to the same state produced by the DSP reset, and the control register bits are not affected.

8.3.8.13 Network Mode (NET) Bit 3

The Network Mode control bit (bit 3) selects the operational mode of the SSI. When the NET bit is cleared, normal mode is selected. When the NET bit is set to 1, network mode is selected.

8.3.8.14 Frame Sync Invert (FSI) Bit 2

The Frame Sync Invert control bit (bit 2) selects the logic of frame sync I/O. If the FSI bit is set to 1, the frame sync is active low. If the FSL bit is set to 0, the frame sync is active high.

8.3.8.15 Frame Sync Length (FSL) Bit 1

The Frame Sync Length control bit (bit 1) selects the length of the frame sync signal to be generated or recognized. If the FSL bit is set to 1, then a one-clock-bit-long frame sync is selected. If the FSL bit is set to 0, a one-word-long frame sync is selected. The length of this word-long frame sync is the same as the length of the data word selected by WL[1:0].

8.3.8.16 Early Frame Sync (EFS) Bit 0

The Early Frame Sync control bit (bit 0) controls when the frame sync is initiated for the transmit and receive sections. When the EFS bit is cleared, the frame sync is initiated as the first bit of data is transmitted or received. When the EFS bit is set, the frame sync is initiated one bit before the data is transmitted or received. The frame sync is disabled after one bit for bit length frame sync and after one word for word length frame sync.

8.3.9 SSI Status Register (SSR)

The SSI Status Register (SSR) is a 16-bit read-only status register used by the DSP56L811 to interrogate the status and serial input flags of the SSI. The status bits are described in the following paragraphs. See **Figure 8-6** on page 8-9 for the programming models of the SSR register.

Note: All the flags in the SSR are updated after the first bit of the next SSI word has completed transmission or reception.

8.3.9.1 Receive Data Register Full (RDF) Bit 7

The SSI Receive Data Register Full flag (bit 7) is set when the SRX register is loaded with a new value. If the receive buffer is enabled, the SRX register is loaded from the Receive Data Buffer Register. Otherwise, the SRX register is loaded from the Receive Shift Register. RDF is cleared when the DSP56L811 reads the SRX register. If the RIE bit is set, a DSP Receive Data interrupt request is issued when the RDF bit is set. The

interrupt request vector depends on the state of the Receiver Overrun (ROE) bit (in the SSR register). The RDF bit is cleared by DSP, SSI, and STOP reset.

8.3.9.2 Transmit Data Register Empty (TDE) Bit 6

The SSI Transmit Data Register Empty flag (bit 6) is set when there is no data waiting to be transferred to the STX register. If the transmit buffer is enabled, this occurs when the data in the STX register has been transferred first into the Transmit Data Buffer Register and then into the Transmit Shift Register (TXSR) before a new value has been written to the STX register. If the transmit buffer is not enabled, this occurs when the contents of the STX register is transferred into the TXSR register. When set, the TDE bit indicates that data should be written to the STX register or to the SSI Time Slot Register (STSR) before the transmit shift register becomes empty, which would cause an underrun error.

The TDE bit is cleared when the DSP56L811 writes to the SRX register or to the STSR register to disable transmission of the next time slot. If the TIE bit is set, an SSI Transmit Data interrupt request is issued when the TDE bit is set. The vector of the interrupt depends on the state of the TUE bit (in the SSR register). The TDE bit is set by DSP, SSI, and STOP reset.

8.3.9.3 Receive Overrun Error (ROE) Bit 5

The Receiver Overrun Error flag (bit 5) is set when the Receive Shift Register (RXSR) is filled and ready to transfer to the SRX register or the Receive Data Buffer Register (when enabled), and these registers are already full, as indicated by the RDF bit being set to 1. The RXSR register is not transferred in this case. A receive overrun error does not cause any interrupts. However, when the ROE bit is set, it causes a change in the interrupt vector used, allowing the use of a different interrupt handler for a receive overrun condition. If a receive interrupt occurs with the ROE bit set, the Receive Data With Exception Status interrupt (vector \$0020) is generated. If a receive interrupt occurs with the ROE bit cleared, the Receive Data With Exception Status interrupt (vector \$0020) is generated.

The ROE bit is cleared by DSP, SSI, or STOP reset and is cleared by reading the SSR with the ROE bit set, followed by reading the SRX register. Clearing the RE bit does not affect the ROE bit.

8.3.9.4 Transmit Underrun Error (TUE) Bit 4

The Transmitter Underrun Error flag (bit 4) is set when the TXSR register is empty (no data to be transmitted), as indicated by the TDE bit being set to 1, and a transmit time slot occurs. When a transmit underrun error occurs, the previously sent data is re-transmitted.

A transmit time slot in the Normal Mode occurs when the frame sync is asserted. In network mode, each time slot requires data transmission and is, therefore, a transmit time slot. (TE = 1.)

The TUE bit does not cause any interrupts. However, the TUE bit does cause a change in the interrupt vector used for transmit interrupts, so that a different interrupt handler can be used for a transmit underrun condition. If a transmit interrupt occurs with the TUE bit set, the Transmit Data With Exception Status interrupt (vector \$0024) is generated. If a transmit interrupt occurs with the TUE bit cleared, the Transmit Data With Exception Status (vector \$0026) interrupt is generated.

The TUE bit is cleared by DSP, SSI, or STOP reset. The TUE bit is also cleared by reading the SSR with the TUE bit set, followed by writing to the STX register or to the STSR register.

8.3.9.5 Transmit Frame Sync (TFS) Bit 3

When set, the Transmit Frame Sync flag (bit 3) indicates that a frame sync occurred during transmission of the last word written to the STX register. Data written to the STX register during the time slot when the TFS bit is set is sent during the second time slot (in Network Mode) or in the next first time slot (in normal mode). In network mode, the TFS bit is set during transmission of the first slot of the frame. It is then cleared when starting transmission of the next slot.

The TFS bit is cleared by DSP, SSI, or STOP reset.

Note: The function of the TFS bit in the DSP56L811 is different from the function of the TFS bit in the DSP56156 SSI.

8.3.9.6 Receive Frame Sync (RFS) Bit 2

When set, the Receive Frame Sync flag (bit 2) indicates that a frame sync occurred during receiving of the next word into the SRX register. In Network Mode, the RFS bit is set while the first slot of the frame is being received. It is cleared when the next slot of the frame begins to be received.

The RFS bit is cleared by DSP, SSI, or STOP reset.

Note: The function of the RFS bit in the DSP56L811 is different from the function of the RFS bit in the DSP56156 SSI.

8.3.9.7 Receive Data Buffer Full (RDBF) Bit 1

The Receiver Data Buffer Full flag (bit 1) is set when the receive section is programmed with the receive buffer enabled, and the contents of the RXSR register

SSI Data and Control Pins

are transferred to the Receive Data Buffer Register. When set, RDBF indicates that data can be read from the SRX register. Note that an interrupt is only generated if both the RDF and RIE bits are set.

The RDBF bit is cleared by DSP, SSI or STOP reset.

8.3.9.8 Transmit Data Buffer Empty (TDBE) Bit 0

The Transmitter Data Buffer Empty flag (bit 0) is set when the transmit section is programmed with the transmit buffer enabled and the contents of the Transmit Data Buffer Register are transferred to the TXSR register. When set, the TDBE bit indicates that data can be written to the STX register. Note that an interrupt is generated only if both the TDE and the TIE bits are set.

The TDBE bit is set by DSP, SSI or STOP reset.

8.3.9.9 Reserved SSR Register Bits

Bits 15-8 are reserved and are read as 0 during read operations. These bits should be written with 0 to ensure future compatibility.

8.3.10 SSI Time Slot Register (STSR)

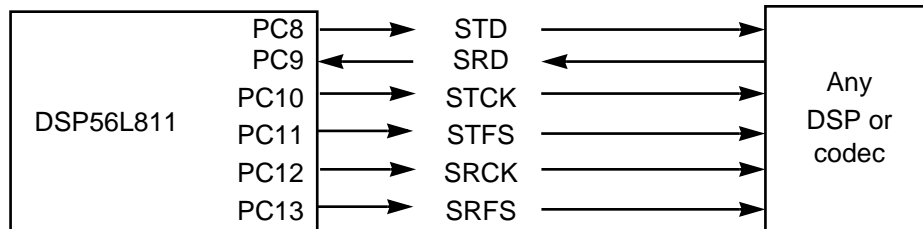
The SSI Time Slot Register (STSR) is used when data is not to be transmitted in an available transmit time slot. For the purposes of timing, the time slot register is a write-only register that behaves like an alternate transmit data register, except that instead of transmitting data, the STD pin is tri-stated. Using this register is important for avoiding overflow/underflow during inactive time slots.

8.4 SSI DATA AND CONTROL PINS

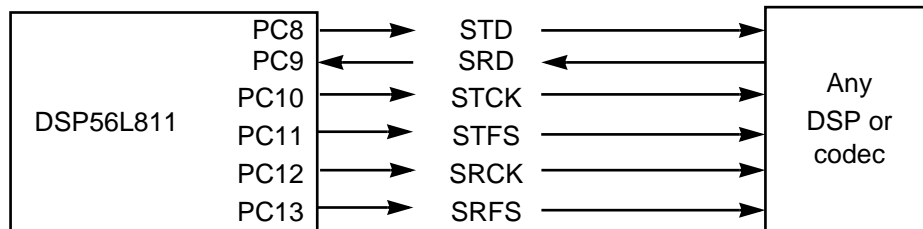
The SSI has the following six dedicated I/O pins:

- Serial Transmit Data (STD/PC8)
- Serial Receive Data (SRD/PC9)
- Serial Transmit Clock (STCK/PC10)
- Serial Transmit Frame Sync (STFS/PC11)
- Serial Receive Clock (SRCK/PC12)
- Serial Receive Frame Sync (SRFS/PC13)

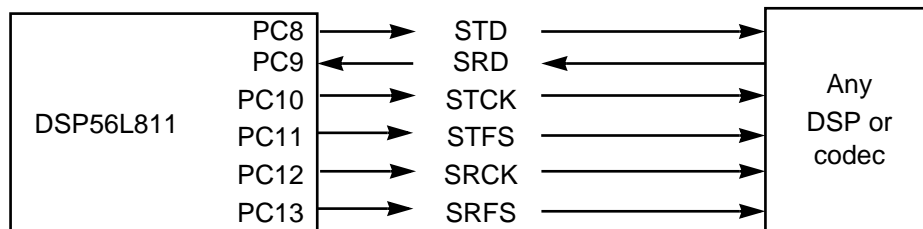
Figure 8-9 and **Figure 8-10** show the main SSI configurations. These pins support all transmit and receive functions with continuous or gated clock as shown. Note that gated clock implementations do not require the use of the frame sync pins (STFS and SRFS). In this case, these pins can be used as general purpose I/O pins, if desired.



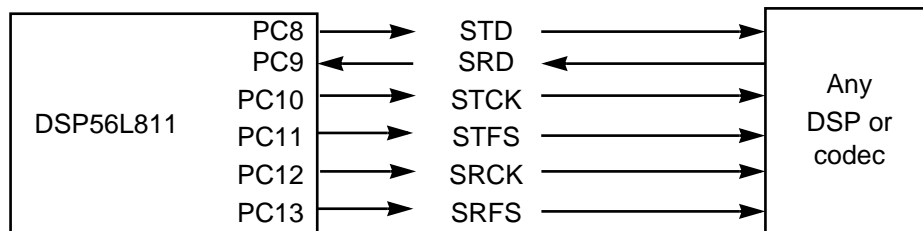
SSI Internal Continuous Clock (Asynchronous)



SSI External Continuous Clock (Asynchronous)



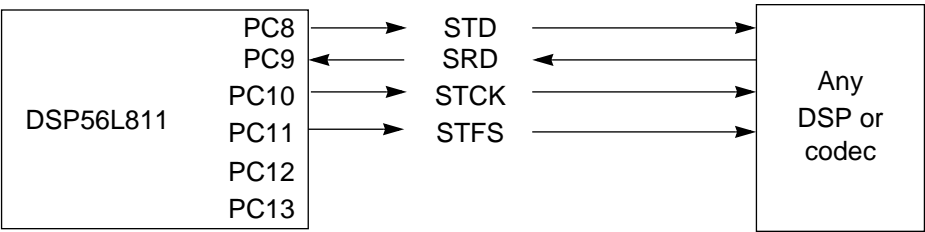
SSI Continuous Clock (RX=Internal, TX = External, Async)



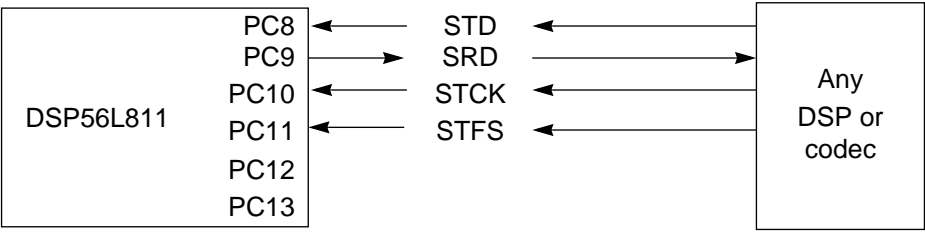
SSI Continuous Clock (RX=External, TX = Internal, Async)

AA0153

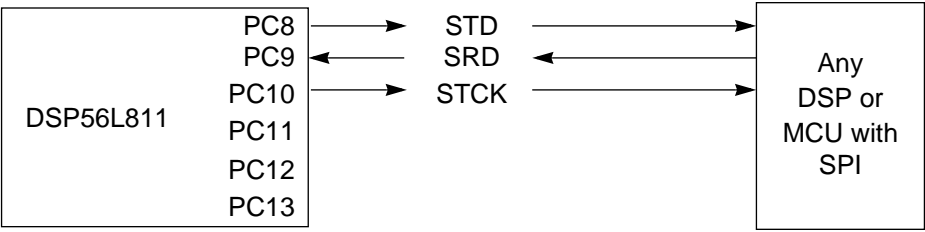
Figure 8-9 Asynchronous SSI Configurations—Continuous Clock



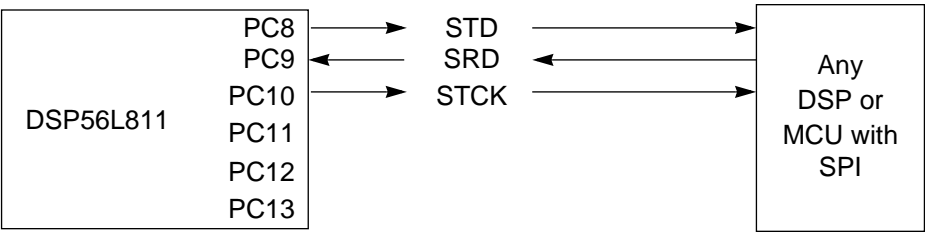
SSI Internal Continuous Clock (Synchronous)



SSI External Continuous Clock (Synchronous)



SSI Internal Gated Clock (Synchronous)



SSI External Gated Clock (Synchronous)

AA0154

Figure 8-10 Synchronous SSI Configurations—Continuous and Gated Clock

8.4.1 SSI Pin Definitions

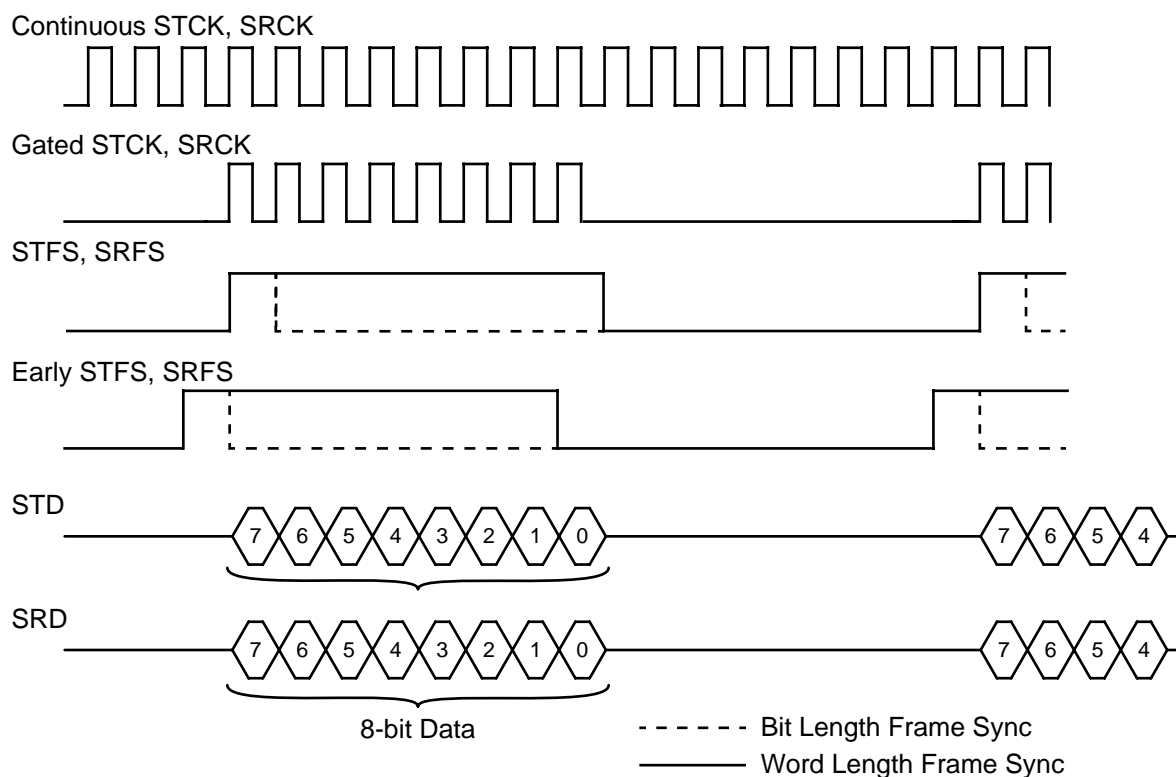
The following paragraphs describe the configuration of the SSI pins.

- **STD/PC8 (Serial Transmit Data)**—The STD pin transmits data from the Serial Transmit Shift Register. The STD pin is an output pin when data is being transmitted and is tri-stated between data word transmissions, and on the trailing edge of the bit clock after the last bit of a word is transmitted. Connect an external resistor to this pin to prevent the signal from floating when not being driven. (A floating pin may provide spurious edge transitions or may go into oscillation.) Since this pin is tri-stated, the external resistor can be either high or low, depending on the circuit designer's choice.
- **SRD/PC9 (Serial Receive Data)**—The SRD pin is used to bring serial data into the Receive Data Shift Register.
- **STCK/PC10 (Serial Transmit Clock)**—The STCK pin can be used as either an input or an output. This clock signal is used by the transmitter and can be either continuous or gated. During gated clock mode, data on the STCK pin is valid only during the transmission of data, otherwise it is tri-stated. In synchronous mode, this pin is used by both the transmit and receive sections. When using gated clock mode, an external resistor should be connected to this pin to prevent the signal from floating when not being driven.
- **STFS/PC11 (Serial Transmit Frame Sync)**—The STFS pin can be used as either an input or an output. The frame sync is used by the transmitter to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. In synchronous mode, this pin is used by both the transmit and receive sections. In gated clock mode, frame sync signals are not used.
- **SRCK/PC12 (Serial Receive Clock)**—The SRCK pin can be used as either an input or an output. This clock signal is used by the receiver and is always continuous. During gated mode, the STCK pin is used instead for clocking in data. This pin is not used in synchronous mode.
- **SRFS/PC13 (Serial Receive Frame Sync)**—The SRFS pin can be used as either an input or an output. The frame sync is used by the receiver to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data.

An example of the pin signals for an 8-bit data transfer is shown in **Figure 8-11**. Continuous and gated clock signals are shown, as well as the bit length frame sync signal and the word length frame sync signal. Note that the shift direction can be

SSI Operating Modes

defined as most significant bit (MSB) first or least significant bit (LSB) first, and that there are other options on the clock and frame sync.



AA0155

Figure 8-11 Serial Clock and Frame Sync Timing

8.5 SSI OPERATING MODES

The SSI has three basic operating modes, with the option of asynchronous or synchronous protocol, as listed in the following:

- Normal mode
 - Asynchronous protocol
 - Synchronous protocol
- Network mode
 - Asynchronous protocol
 - Synchronous protocol

- Gated clock mode
 - Synchronous protocol only

These modes can be programmed by several bits in the SSI control registers.

Table 8-6 lists these operating modes and some of the typical applications in which they can be used:

Table 8-6 SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Applications
Asynchronous	Continuous	Normal	Multiple Synchronous Codecs
Asynchronous	Continuous	Network	TDM Codec or DSP Networks
Synchronous	Continuous	Normal	Multiple Synchronous Codecs
Synchronous	Continuous	Network	TDM Codec or DSP Networks
Synchronous	Gated	Normal	SPI-Type Devices; DSP to MCU

The transmit and receive sections of the SSI can be synchronous or asynchronous. In *synchronous mode*, the transmitter and the receiver use a common clock and frame synchronization signal. In *asynchronous mode*, the transmitter and receiver each has its own clock and frame synchronization signals. Continuous or gated mode can be selected. In *continuous mode*, the clock runs continuously. In *gated mode*, the clock is only functioning during transmission.

Normal or network protocol can also be selected. In *normal protocol*, the SSI functions with one data word of I/O per frame. In *network protocol*, any number from two to 32 data words of I/O per frame can be used. Network mode is typically used in star or ring time division multiplex networks with other processors or codecs, allowing interface to time division multiplexed networks without additional logic. Use of the gated clock is not allowed in network mode. These distinctions result in the basic operating modes that allow the SSI to communicate with a wide variety of devices.

The SSI supports both normal and network modes, and these modes can be selected independent of whether the transmitter and receiver are synchronous or asynchronous. Typically these modes are used in a periodic manner, where data is transferred at regular intervals such as at the sampling rate of an external codec.

Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC[4:0] bits in either the SCRRX or SCRTX register, depending on whether data is being transferred or

SSI Operating Modes

received. The number of words transferred per frame depends on the mode of the SSI.

In normal mode, one data word is transferred per frame. In network mode, the frame is divided into anywhere between two and 32 time slots, where in each time slot one data word can optionally be transferred.

8.5.1 Normal Mode

Normal mode is the simplest mode of the SSI. It is used to transfer one word per frame. In continuous clock mode, a frame sync occurs at the beginning of each frame.

The length of the frame is determined by the following factors:

- The period of the Serial Bit Clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK pin)
- The number of bits per sample (WL[1:0] bits)
- The number of time slots per frame (DC[4:0] bits)

If normal mode is configured to provide more than one time slot per frame, data is transmitted only in the first time slot. No data is transmitted in subsequent time slots.

8.5.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in normal mode are:

1. SSI enabled (SSIEN = 1)
2. Transmitter enabled (TE = 1)
3. Frame sync active (for continuous clock case)
4. Bit clock begins (for gated clock case)

When the above conditions occur in normal mode, the next data word is transferred into the transmit shift register from the STX register, or from the Transmit Data Buffer Register, if transmit buffering is enabled. The new data word is transmitted immediately. If buffering is not enabled, the TDE bit is set (transmitter empty), and the transmit interrupt occurs if the TIE bit is set to 1 (transmit interrupt is enabled). If buffering is enabled, the TDE bit is set (transmitter empty), and the transmit interrupt occurs, if the TIE bit is set to 1 (transmit interrupt is enabled), when both values have been transferred to the Transmit Shift Register. If buffering is enabled, a second data word can be transferred and shifted before the DSP56L811 must write new data to the STX register.

The STD pin is tri-stated except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not tri-stated, even if both receiver and transmitter are disabled.

8.5.1.2 Normal Mode Receive

The conditions for data reception from the SSI are:

1. SSI enabled (SSIEN = 1)
2. Receiver enabled (RE = 1)
3. Frame sync active (for continuous clock case)
4. Bit clock begins (for gated clock case)

With the above conditions in normal mode with a continuous clock, each time the frame sync signal is generated (or detected) a data word is clocked in. With the above conditions and a gated clock, each time the clock begins, a data word is clocked in. If buffering is not enabled, after receiving the data word it is transferred from the RXSR register to the SRX register, the RDF flag is set (receiver full), and the Receive Interrupt occurs if it is enabled (the RIE bit is set to 1). If buffering is enabled, after receiving the data word it is transferred to the Receive Data Buffer Register. The RDF flag is set if both the SRX register and Receive Data Buffer Register are full, and the Receive Interrupt occurs if it is enabled (the RIE bit is set to 1).

The DSP56L811 program has to read the data from the SRX register before a new data word is transferred from the RXSR register, otherwise the ROE bit is set. If buffering is enabled, the ROE bit is set when both the SRX register and the Receive Data Buffer Register contain data and a new data word is ready to be transferred to the Receive Data Buffer Register.

Figure 8-12 shows transmitter and receiver timing for an 8-bit word with two words per time slot in normal mode, continuous clock with a late word length frame sync.

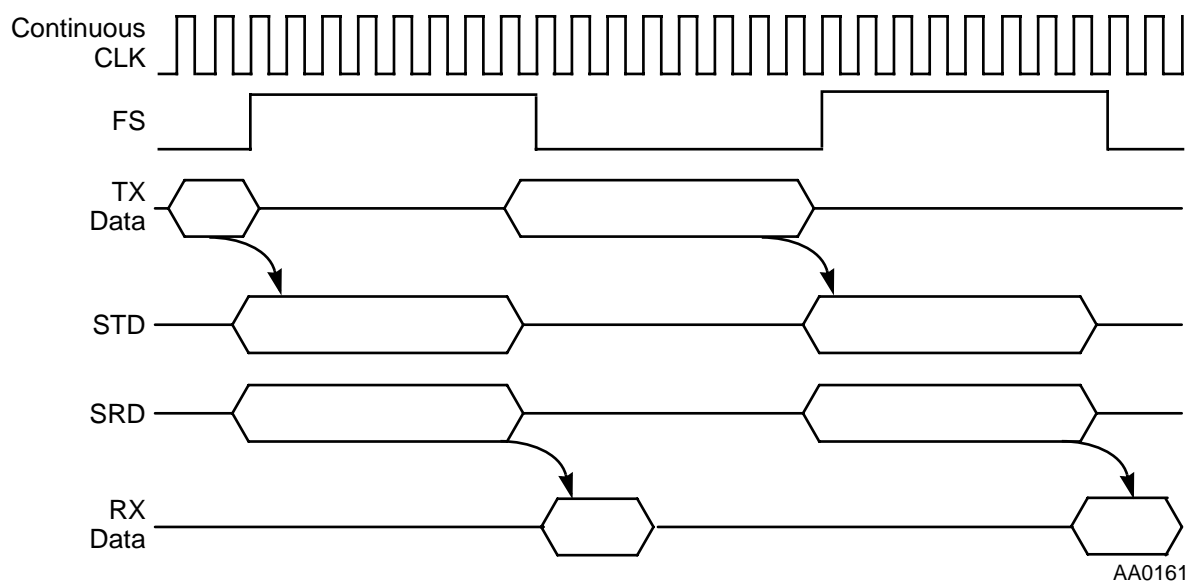


Figure 8-12 Normal Mode Timing—Continuous Clock

Figure 8-13 shows a similar case for gated clock. Note that a pulldown resistor is required in the gated clock case because the clock pin is tri-stated between transmissions.

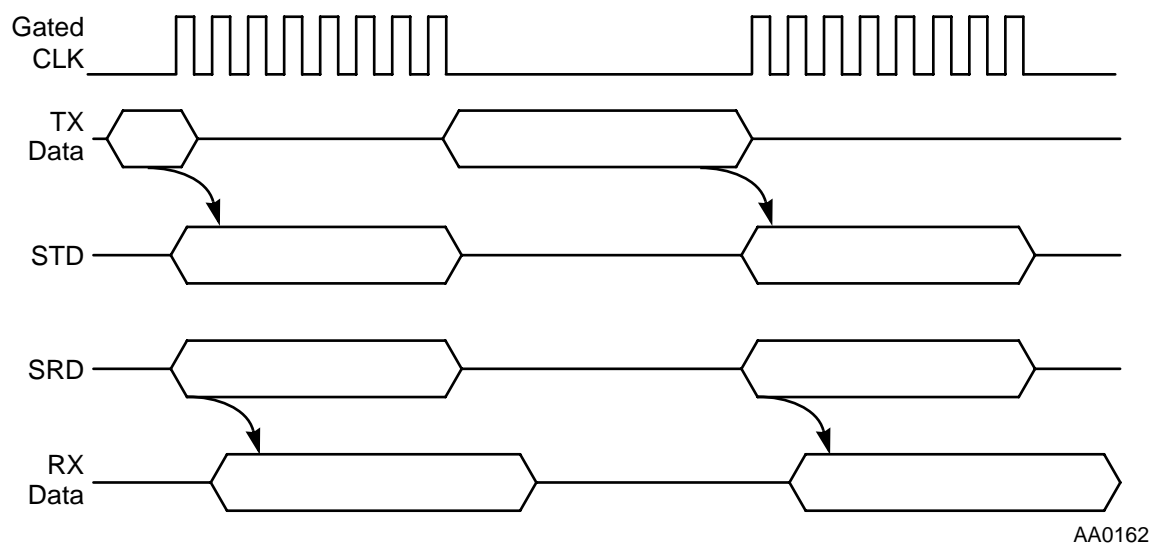


Figure 8-13 Normal Mode Timing—Gated Clock

8.5.2 Network Mode

Network mode is used for creating a TDM network, such as a TDM codec network or a network of DSPs. In continuous clock mode, a frame sync occurs at the beginning of each frame. In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate codec or DSP on the network. The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in normal mode). The frame rate dividers, controlled by the DC[4:0] bits select two to 32 time slots per frame. The length of the frame is determined by the following factors:

- The period of the Serial Bit Clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK pin)
- The number of bits per sample (WL[1:0] bits)
- The number of time slots per frame (DC[4:0] bits)

In network mode, data can be optionally transmitted in any time slot.

The distinction of the network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification allows the option of transmitting data during the time slot by writing to the STX register or ignoring the time slot by writing to STSR register. The receiver is treated in the same manner, except that data is always being shifted into the Receive Shift Register (RXSR) and transferred to the SRX register. The DSP56L811 reads the SRX register and either uses it or discards it.

8.5.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SCR2 register are both set to 1. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new time slot (if the TE bit is set to 1 during a slot other than the first). This functionality is different from the DSP56156 SSI. *Software has to find the start of the next frame.*

Normal start up sequence for transmission is to:

1. Write the data to be transmitted to the STX register. This clears the TDE flag.

SSI Operating Modes

2. Set the TE bit to enable the transmitter on the next word boundary (for continuous clock case).
3. Enable transmit interrupts.

Alternatively, the programmer may decide *not* to transmit in a time slot by writing to the SSI Time Slot Register (STSR). This clears the TDE flag just as if data were going to be transmitted, but the STD pin remains tri-stated during the time slot.

When the frame sync is detected or generated (continuous clock), the first enabled data word is transferred from the STX register to the Transmit Shift Register and is shifted out (transmitted). When the STX register is empty, the TDE bit is set, which causes a transmitter interrupt to be sent if the TIE bit is set. Software can poll the TDE bit or use interrupts to reload the STX register with new data for the next time slot or write to the STSR register to prevent transmitting in the next time slot. Failing to reload the STX register (or writing to the STSR register) before the Transmit Shift Register (TXSR) is finished shifting (empty) causes (1) a transmitter underrun, (2) the TUE error bit to be set and (3) the STD pin is tri-stated for the next time slot.

The operation of clearing the TE bit disables the transmitter after completion of transmission of the current data word. Setting the TE bit enables transmission of the next word. During that time the STD pin is tri-stated. The TE bit should be cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, the network mode transmitter generates interrupts every enabled time slot and requires the DSP program to respond to each enabled time slot. These responses can be:

- Write data register with data to enable transmission in the next time slot.
- Write the time slot register to disable transmission in the next time slot.
- Do nothing—transmit underrun occurs at the beginning of the next time slot and the previous data is re-transmitted.

8.5.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SCR2 register are set to 1. However, the receive enable only takes place during that time slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver is disabled immediately. This is different from the DSP56156 SSI. *Software has to find the start of the next frame.*

When the word is completely received, it is transferred to the SRX register, which sets the RDF bit (Receive Data register full). Setting the RDF bit causes a receive interrupt to occur if the receiver interrupt is enabled (the RIE bit is set to 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 SRX data register. The DSP program has to read the data from the RX data register (which clears RDF) before the second data word is completely received (ready to transfer to RX data register), or a receive overrun error occurs (the ROE bit is set).

An interrupt can occur after the reception of each enabled data word or the programmer can poll the RDF flag. The DSP56L811 program response can be:

- Read RX and use the data.
- Read RX and ignore the data.
- Do nothing—the receiver overrun exception occurs at the end of the current time slot.

Note: For a continuous clock, the optional frame sync output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame sync generation. The only way to disable the bit clock and the frame sync generation is to disable the SSIEN bit in the SCR2 register.

SSI Operating Modes

The transmitter and receiver timing for an eight-bit word with continuous clock, buffering disabled, three words per frame sync in Network Mode is shown in **Figure 8-14**.

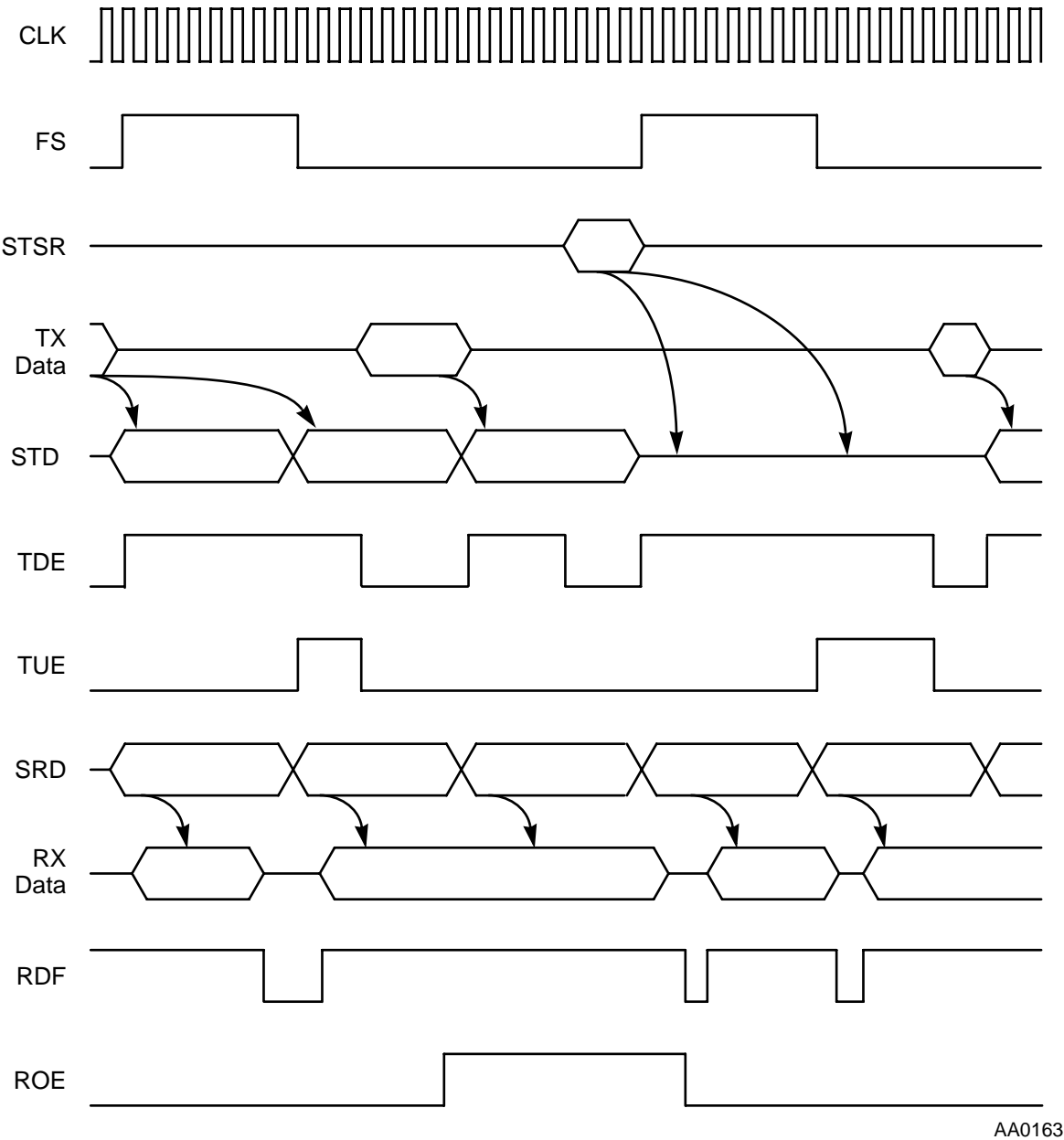


Figure 8-14 Network Mode Timing—Continuous Clock

8.5.3 Gated Clock Operation

Gated clock mode is often used to hook up to SPI-type interfaces on MCUs or external peripheral chips. In gated clock mode, the presence of the clock indicates that valid data is on the STD or SRD pins. For this reason, no frame sync is needed in this mode. Once transmission of data has completed, the clock pin is tri-stated. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock, and in normal mode. Gated clocks are not allowed in network mode.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. For the case of internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs (such as the first time slot in normal mode), the internal bit clock is enabled onto the appropriate clock pin. This allows data to be transferred out in periodic intervals in gated clock mode. With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in.

8.6 SSI RESET AND INITIALIZATION PROCEDURE

The SSI is affected by three types of reset:

- **DSP Reset**—The DSP reset is generated by asserting either the $\overline{\text{RESET}}$ pin or the COP timer reset. The DSP reset clears the SSIEN bit in SCR2, which disables the SSI. All other status and control bits in the SSI are affected as described in **SSI Programming Model** on page 8-8.
- **SSI Reset**—The SSI reset is generated when the SSIEN bit in the SCR2 register is cleared. The SSI status bits are preset to the same state produced by the DSP reset. The SSI control bits are unaffected. The SSI reset is useful for selective reset of the SSI interface without changing the present SSI control bits and without affecting the other peripherals.
- **STOP Reset**—The Stop reset is caused by executing the STOP instruction. While in the stopped state, no clock is active in the SSI, which is always powered down in stop mode. The SSI status bits are preset to the same state produced by the DSP reset. The SSI control bits are unaffected.

The correct sequence to initialize the SSI interface is as follows:

1. DSP reset or SSI reset
2. Program SSI control registers
3. Set the SSIEN bit in SCR2

Configuring Port C for SSI Functionality

To ensure proper operation of the SSI, the DSP programmer should use the DSP or SSI reset before changing any of the following control bits listed in **Table 8-7**:

Table 8-7 SSI Control Bits Requiring Reset Before Change

Bit	Control Register
EFS	SCR2
FSI	SCR2
FSL	SCR2
NET	SCR2
RBF	SCR2
RXD	SCR2
SCKP	SCR2
SHFD	SCR2
SYN	SCR2
TBF	SCR2
TXN	SCR2
WL0	SCRRX SCRTX
WL1	SCRRX SCRTX

These control bits should not be changed during SSI operation.

Note: The SSI bit clock must go low for at least one complete period to ensure proper SSI reset.

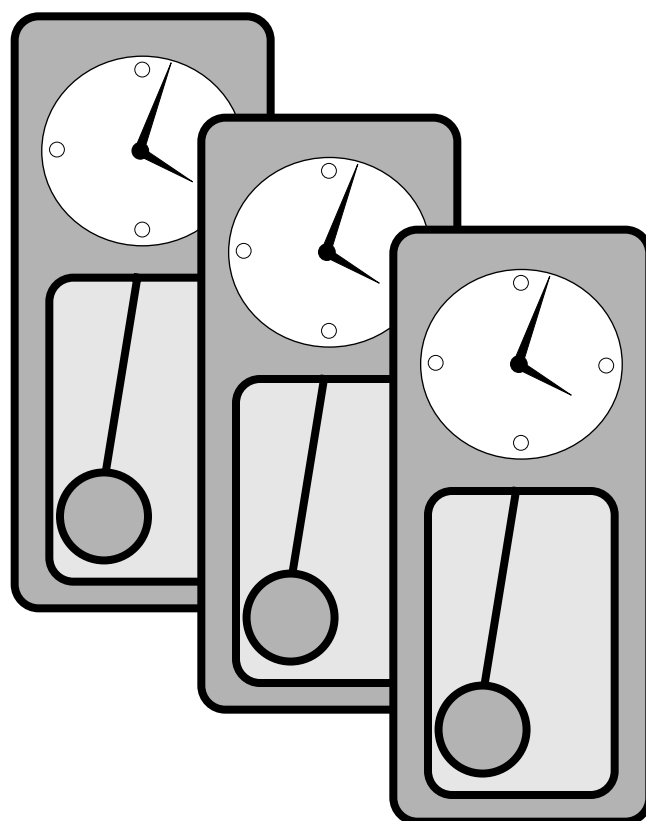
8.7 CONFIGURING PORT C FOR SSI FUNCTIONALITY

The Port C Control (PCC) register is used to individually configure each pin as either an SSI pin or a GPIO pin. Setting the corresponding CC bit in the PCC register to 1 configures the pin as an SSI pin. When the PCC register bit is set to 1, it is not necessary to program the corresponding PCDDR register bit. The SSI peripheral ensures the correct direction of this pin. Programming the PCDDR register is necessary only when a pin is programmed as a GPIO pin.



SECTION 9

TIMERS



9.1	INTRODUCTION	9-3
9.2	TIMER PROGRAMMING MODEL	9-5
9.3	TIMER RESOLUTION	9-10
9.4	TIMER INTERRUPT PRIORITIES	9-11
9.5	EVENT COUNTING WITH THE TIMER MODULE	9-11
9.6	TIMER MODULE LOW POWER OPERATION	9-19
9.7	TIMER MODULE TIMING DIAGRAMS	9-21

9.1 INTRODUCTION

This section describes the general purpose timer module provided on the DSP56L811, as a part of Port C.

The timer module provides three independently programmable 16-bit timer/event counters, which are referred to as Timer 0, Timer 1, and Timer 2. All three timer/event counters can be clocked with signals coming from one of two internal sources. Timer 1 and Timer 2 can also be clocked by the overflow events of Timer 0 and Timer 1, respectively. In addition, the counters can be clocked with external signals from the Timer I/O pins (TIO01 or TIO02) on Port C to count external events when configured as inputs. The same pins can be used to provide a timer pulse or for timer clock generation when configured as outputs. The timer/event counters can be used to either interrupt the DSP56L811 or to signal an external device at periodic intervals.

The capabilities of the Timer Module include the ability to:

- Decrement a timer to 0 and interrupt
- Decrement a timer to 0 and apply a pulse to a TIO pin
- Decrement a timer to 0 and toggle a TIO pin (50% duty cycle)
- Generate a wave form on a TIO pin with a duty cycle other than 50% using two timers
- Count events on an external TIO pin when selected as the input clock
- Operate timers independently or cascade timers together

Each timer can be clocked from:

- A TIO pin
- A clock running at $\frac{1}{2}$ the instruction rate of the chip
- A slow clock generated from the Prescaler Divider

In addition, Timer 1 can be clocked by the overflow events of Timer 0, and Timer 2 can be clocked by the overflow events of Timer 1.

When a timer is clocked using the slower clock from the Prescaler Divider, it can continue counting even when the DSP56800 core is in stop mode. All three timers are

Introduction

capable of operating in stop mode. However, only Timer 2 is capable of bringing the DSP56800 core out of stop mode when it times out.

Figure 9-1 shows a block diagram of the timer module.

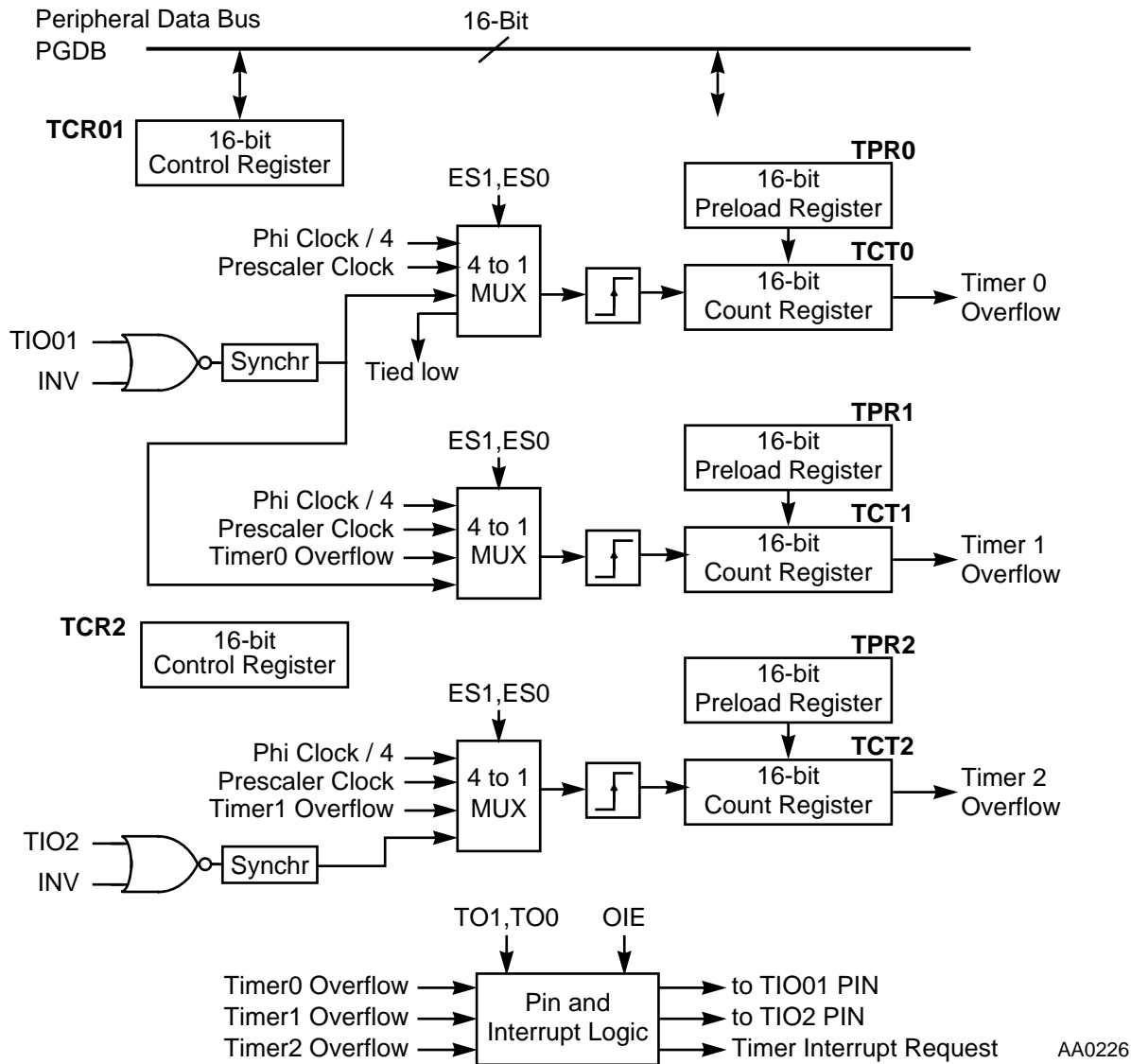
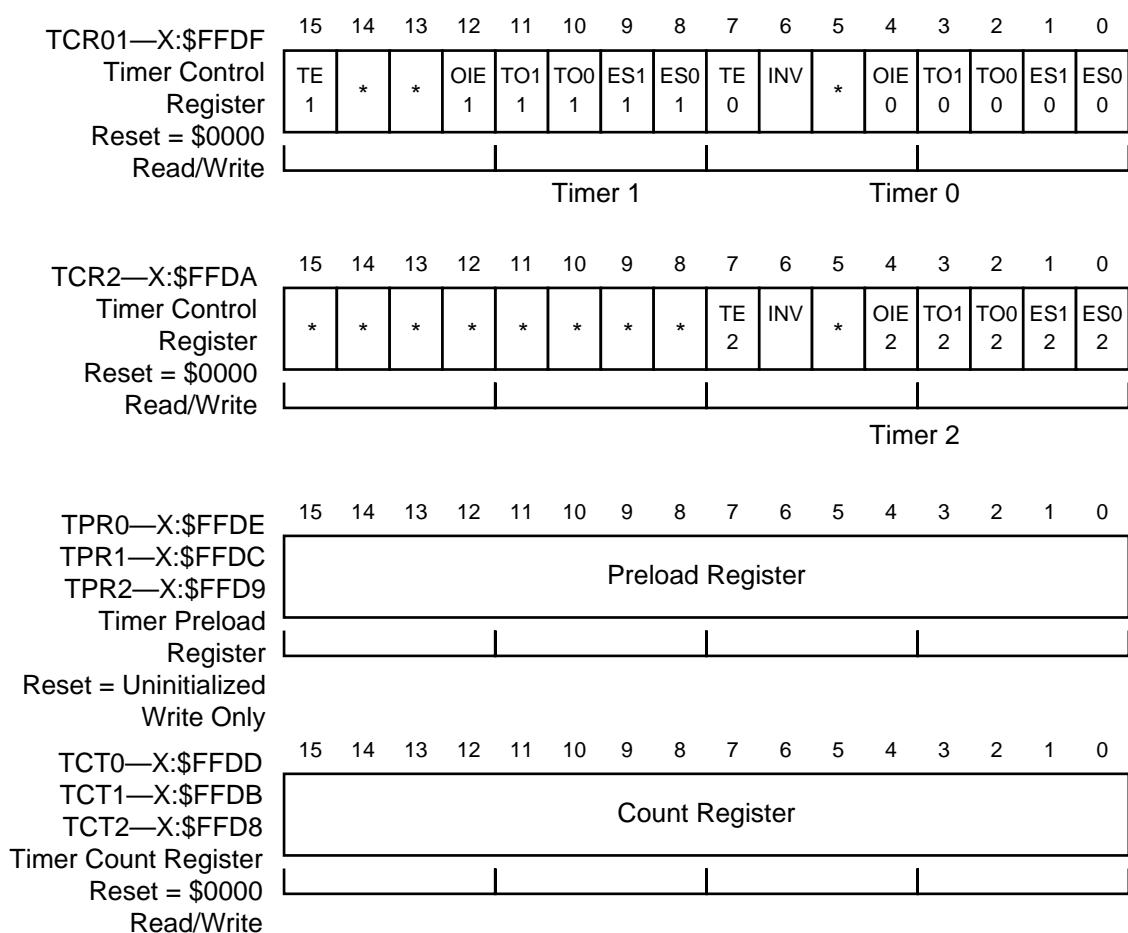


Figure 9-1 General Purpose Timer Module

9.2 TIMER PROGRAMMING MODEL

The General Purpose Timer Module contains eight read/write registers, all of which are memory-mapped in the X memory space. These eight registers include three sets of two 16-bit registers, one set for each timer: the Timer Count Register (TCT) and the Timer Preload Register (TPR). In addition, two Timer Control Registers (TCR01 and TCR2) control the operations of the timers. The TCR01 register provides control for Timers 0 and 1. The TCR2 register provides control for Timer 2 (the top eight bits of the TCR2 register are not used). The General Purpose Timer's programming model is shown in **Figure 9-2**



* Indicates reserved bits, written as 0 to ensure future compatibility

AA0227

Figure 9-2 General Purpose Timer Programming Model

9.2.1 Timer Control Registers (TCR01 and TCR2)

Two 16-bit read/write timer control registers contain the control bits for three 16-bit timers. The upper byte controls one timer, and the lower byte controls another timer, as shown in **Table 9-1**.

Table 9-1 Timer Control Registers TCR01 and TCR02

Timer	Control Register Used
0	TCR01[7:0]
1	TCR01[15:8]
2	TCR2[7:0]
(unused)	TCR2[15:8]

The timer control bits are defined in the following paragraphs.

9.2.1.1 Timer Enable (TE) Bit 15, Bit 7

The Timer Enable control bit (bit 7 in TCR01 for Timer 0, bit 15 in TCR01 for Timer 1, or bit 7 in TCR2 for Timer 2) is used to enable or disable the timer. Setting the TE bit to 1 enables the timer. The counter starts decrementing from its preset value each time an event comes in. Clearing the TE bit disables the timer. The count register is not affected by this operation. However, if a direct write to the count register occurs after the last count register reload, the value written is loaded into the count register instead of the preload value. The TE bit is cleared by reset.

9.2.1.2 Invert (INV) Bit 6

When the Invert control bit (bit 6 in the TCR01 register for the TIO01 pin, or bit 6 in the TCR2 register for the TIO2 pin) is set to 1, the external signal coming in the TIO pin (TIO01 or TIO2, depending on which register has its INV bit set) is inverted before being synchronized and entering the 16-bit counter. All 1-to-0 transitions of the TIO pin then decrement the 16-bit counter. When the INV bit is cleared, the external signal on TIO is not inverted and the 16-bit counter is decremented on all 0-to-1 transitions. The INV bit is cleared on reset.

Table 9-2 INV Bit Definition

INV	External Event on TIO Pin
0	Detects Rising Edges
1	Detects Falling Edges

9.2.1.3 Overflow Interrupt Enable (OIE) Bit 12, Bit 4

When the Overflow Interrupt Enable control bit (bit 4 in TCR01 for Timer 0, bit 12 in TCR01 for Timer 1, or bit 4 in TCR2 for Timer 2) is set to 1, an interrupt is requested to the DSP56L811 at the next event after the count register reaches 0. When the OIE bit is cleared, the interrupt is disabled and any pending interrupts are cleared. The OIE bit can be individually set for each timer, selectively enabling the interrupt capability independently for each timer. The OIE bit is cleared on reset.

As with all on-chip peripheral interrupts for the DSP56L811, the SR register must first be set to enable maskable interrupts (interrupts of level IPL1). Next, the CH4 bit (bit 11) in the IPR register (see **3.6.2 DSP56L811 Interrupt Priority Register** on page 3-19) must also be set to enable this interrupt. **Table 9-3** lists the interrupt vectors for the three timers.

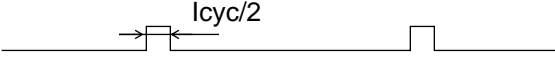

Table 9-3 Timer Interrupt Vectors

Timer	Interrupt Vector	Interrupt Priority
0	\$0018	0
1	\$001A	0
2	\$001C	0

9.2.1.4 Timer Output Enable (TO1-TO0) Bits 11-10, Bits 3-2

The Timer Output enable control bits (bits 3-2 in TCR01 for Timer 0, bits 11-10 in TCR01 for Timer 1, or bits 3-2 in TCR2 for Timer 2) are used to program the function of the timer output pin (TIO). **Table 9-4** shows the relationship between the value of the TO[1:0] bits and the function of the TIO pin. The TO bits are cleared on reset.

Table 9-4 TIO Pin Function

TO1	TO0	Function of TIO	Signal on TIO
0	0	TIO configured as input	_____
0	1	(reserved)	_____
1	0	Overflow pulse, TIO configured as output	
1	1	Overflow toggle, TIO configured as output	

Timer Programming Model

When the TO[1:0] bits are programmed for overflow pulse, the width of the pulse is the period of the internal Phi clock.

Note: For the overflow pulse and overflow toggle modes, it is possible to have more than one timer enabled to drive the TIO pin. In this case, the TIO pin is pulsed or toggled when either counter reaches 0. In this way, it is possible with two timers to generate square waves on the TIO pin with duty cycles other than 50%. Both timers are programmed with the same preload value, but their initial starting value differs. The amount by which their starting values differs determines the duty cycle of the resulting waveform.

It is also acceptable for all timers to be programmed such that no timer drives the TIO pin (TO[1:0] = 00 for the timers). The TIO pin is programmed in this manner when used as input.

Note: If the toggle mode is selected (TO[1:0] = 11) and the TE bit is written as 0 while the TIO pin is either high or low, the TIO pin remains in the same state. If the TO1 bit or TO0 bit is written as 0 with the TE bit equal to 0, the pin remains high, and is driven low when the timer is re-enabled.

9.2.1.5 Event Select (ES1-ES0) Bits 9-8, Bits 1-0

The Event Select control bits (bits 1-0 in TCR01 for Timer 0, bits 9-8 in TCR01 for Timer 1, or bits 1-0 in TCR2 for Timer 2) select the source of the timer clock. If ES[1:0] is 11, an external signal coming from the TIO pin is used as input to the decrement register. The external signal is synchronized to the internal clock as described in **Event Counting with the Timer Module** on page 9-11. The ES bit is cleared by reset.

Table 9-5 ES Bit Definition

ES1	ES0	Clock Selected
0	0	Internal Phi Clock / 4
0	1	Internal Prescaler Clock
1	0	Previous Timer Overflow
1	1	External Event from TIO Pin

Note: For Timer 0, setting the ES[1:0] bits to 10 causes the clock source to be pulled low, and no signal is applied to the timer. This is because no previous timer is available.

When a timer is clocked using the slower clock from the Prescaler Divider, it can continue counting even when the DSP56800 core is in stop mode.

9.2.1.6 Reserved TCR Register Bits

Bits 14, 13, and 5 of the TCR01 and TCR2 registers are reserved and are read as 0 during read operations. These bits should be written with 0 for future compatibility. In addition, the top byte of TCR2 is not used.

9.2.2 Timer Preload Register (TPR)

The Timer Preload Register (TPR) is a 16-bit write-only register that contains the value to be reloaded into the count register when a timer is enabled and when the Timer Count register (TCT) has decremented to 0. Three preload registers are provided, one for each timer.

The timer must be disabled (its TE bit in the TCR01 or TRC2 register is set to 0) when the user program writes a new value to its TPR register. This new value transfers immediately into the TCT register (described in the following section) unless a direct write to the TCT register has already been performed.

Note: The TPR register can be written only when the corresponding timer is disabled (its TE bit set to 0) in the Timer Control Register.

Since the TPR register is write-only and cannot be read, if it is necessary to read its value, this can be accomplished by writing the TPR register with the TE bit set to 0 and then reading the corresponding count register (with the TE bit set to 0). The TPR registers are initialized to 0 on reset.

9.2.3 Timer Count Register (TCT)

The Timer Count Register (TCT) is a 16-bit read/write register that contains the count for a timer. Three count registers are provided, one for each timer.

When a timer is enabled (its TE bit in the TCR01 or TRC2 register is set to 1), its TCT register is decremented by one with each clock. On the next event after the count register reaches 0, an overflow interrupt is generated if the Overflow Interrupt Enable bit (OIE) is set in the timer control register. Also, the state of the TIO pin can then be affected according to the mode selected by the Timer Output enable bits (TO[1:0]) of the timer control register. If “n” is the value stored in the count register when the timer is enabled, the overflow interrupt occurs after (n+1) input events. After reaching 0, the count register is reloaded with the contents of the preload register. The count register is loaded with a direct value when a direct write to the count register is executed.

Timer Resolution

The TCT register may also be read or written by a user program. When writing to the TCT register with the corresponding timer disabled, the value is immediately written to the count register, and is not overwritten by the value stored in the preload register when the timer is enabled (its TE bit set to 1). The value stored or written in the preload register is loaded into the count register on the next event after it reaches 0, unless another write to the count register is performed in the meantime. Refer to **Timer Module Timing Diagrams** on page 9-21 for more details.

Note: The count register can only be written when the corresponding timer is disabled (the TE bit is set to 0) in the timer's control register.

The count register may be *read* only if one of the following conditions is true:

- The PLL is enabled (the PLLE bit in the PCR1 register is set to 1) and the Prescaler clock is no more than half of the frequency of the Phi Clock, i.e., the frequency of the Prescaler clock is not the same as the frequency of the Phi Clock.
- The PLL is bypassed (the PLLE bit in the PCR1 register is set to 0) and the Prescaler within the PLL is set to divide by 1 (the PS[2:0] bits in the PCR1 register are set to 000).
- The Timer with the desired count register is disabled (the TE bit in the timer's control register is set to 0).

See **10.3.1 PLL Control Register 1 (PCR1)** on page 10-8 for information on the PCR1 register.

The count register is initialized to 0 on reset.

9.3 TIMER RESOLUTION

Table 9-6 shows the range of timer interrupt rates (overflow interrupt using the Phi clock) that are provided by the Timer Count Register (TCR0-TCR2) and the Timer Preload Register (TPR0-TPR2).

Table 9-6 Timer Range and Resolution

Input Clock Period	Timer Resolution (Preload = 0)	Timer Range (Preload = $2^{16}-1$)	Two Cascaded Timer Range
Phi Clock = 25 ns (20 MIPs)	100 ns	6.55 ms	429 s

Table 9-6 Timer Range and Resolution (Continued)

Input Clock Period	Timer Resolution (Preload = 0)	Timer Range (Preload = $2^{16}-1$)	Two Cascaded Timer Range
Phi Clock = 50 ns (10 MIPS)	200 ns	13.1 ms	859 s
Phi Clock = 100 ns (5 MIPS)	400 ns	26.2 ms	1718 s
Prescaler Clock = 31 us (32.00 KHz)	31.25 us	2.0 s	134,218 s

The value stored in the TPR register is the preload count. The overflow interrupt occurs every time the input clock provides a quantity of cycles equal to the Preload count +1.

9.4 TIMER INTERRUPT PRIORITIES

The timer module has a simple interrupt priority scheme among its different timers, as shown in **Table 9-7**. It is important to service timer interrupts with higher priorities in a timely fashion to determine if any timer interrupts with a lower priority are also present.

Table 9-7 Timer Interrupt Priorities

Timer	Priority
0	Highest
1	Middle
2	Lowest

9.5 EVENT COUNTING WITH THE TIMER MODULE

This section contains examples of how to program some of the timing/counting capabilities of the general-purpose timers. This set of examples is by no means a complete list of either the capabilities or the programming techniques that can be used. Instead, it offers a representative range of what can be accomplished.

The timer module can be used as an event counter. Setting the ES bits to 11 configures the timer to count the number of edges (external events) detected on the

Event Counting with the Timer Module

specified TIO pin. An external event is defined as a rising edge when the INV bit is set to 0, or a falling edge when the INV bit is set to 1.

After the pin is conditionally inverted, it is synchronized to the Phi Clock. Events on the TIO pin can be counted in wait mode, but cannot be counted in stop mode.

Note: The maximum allowed frequency on the TIO pin is the frequency of the Phi clock divided by four.

Example 9-1 shows how to count events on a pin.

Example 9-1 Counting Events on a Pin

```

;*****
;* example of Counting Events on a pin (with interrupt) *
;* for Timer Module                                     *
;* of DSP56L811 chip                                     *
;*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
IPR        EQU    $FFFB ; Interrupt Priority Register
TCR01      EQU    $FFDF ; Timer 0&1 Control Register
TCR2       EQU    $FFDA ; Timer 2 Control Register
TCT0       EQU    $FFDD ; Timer 0 Count Register
TPR0       EQU    $FFDE ; Timer 0 Preload Register

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set   |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then      |
;|      jumps to first location of internal program RAM (P:$0000).       |
;+-----+

      ORG    P:$0000      ; Cold Boot
      JMP    START        ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG    P:$E000      ; Warm Boot
      JMP    START        ; Hardware RESET vector (Mode 2)

```

Example 9-1 Counting Events on a Pin (Continued)

```

    ORG    P:$0018      ;
    JSR    TISR          ; Timer 0 Overflow vector

    ORG    P:START      ; Start of program
;*****
;* General setup *
;*****

    MOVEP  #$0000,X:BCR  ; External Program memory has 0 wait states.
                        ; External data memory has 0 wait states.
                        ; Port A pins tri-stated when no external access.

    BFCLR  #$0200,SR     ; Allow IPL (Interrupt Priority Level) 0
                        ; -- Enable maskable interrupts.
                        ; -- (peripherals, etc.)
;*****
;* Timer Module setup *
;*****

    MOVEP  #$0013,X:TCR01; Configure:
                        ; Timer 0 & 1 disabled.
                        ; Timer 0
                        ; -- don't Invert TIO input: detect rising edges.
                        ; -- Overflow Interrupt enabled.
                        ; -- TIO pin configured as input.
                        ; -- timer clock Event source is TIO pin.

    MOVEP  #$0000,X:TCR2 ; Timer 2 disabled.
    MOVEP  #234,X:TCT0   ; Set Timer 0 Count Register to 234 (235 events).
    MOVEP  #234,X:TPR0   ; Set Timer 0 Preload Register to 234.
    BFSET  #$0800,X:IPR  ; Enable Timer Module interrupts.
    ; ...

    BFSET  #$0080,X:TCR01; Enable Timer 0

;*****
;* Main routine *
;*****

TEST ; Test Loop
    ; ...

```

Event Counting with the Timer Module

Example 9-1 Counting Events on a Pin (Continued)

```
BRA    TEST

TISR ; Timer Interrupt Service Routine
      ; interrupt code
RTI
```

A timer can also be used to decrement to 0 and generate an interrupt, as shown in **Example 9-2**.

Example 9-2 Decrementing to 0 and Generating an Interrupt

```
*****
; * INTERRUPT example *
; * for Timer Module *
; * of DSP56L811 chip *
*****

START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
IPR        EQU    $FFFFB ; Interrupt Priority Register
PCR0       EQU    $FFF2 ; PLL Control Register 0
PCR1       EQU    $FFF3 ; PLL Control Register 1
TCR01      EQU    $FFDF ; Timer 0&1 Control Register
TCR2       EQU    $FFDA ; Timer 2 Control Register
TCT0       EQU    $FFDD ; Timer 0 Count Register
TPR0       EQU    $FFDE ; Timer 0 Preload Register

*****
; * Vector setup *
*****

; +-----+
; | Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
; |       chip operating mode for Mode 2 (Normal Expanded Mode), then |
; |       jumps to first location of internal program RAM (P:$0000). |
; +-----+

      ORG    P:$0000      ; Cold Boot
      JMP    START        ; also Hardware RESET vector (Mode 0, 1, 3)
```


Example 9-2 Decrementing to 0 and Generating an Interrupt (Continued)

```

    ORG    P:$E000        ; Warm Boot
    JMP    START          ; Hardware RESET vector (Mode 2)

    ORG    P:$0018        ;
    JSR    TISR            ; Timer 0 Overflow vector

    ORG    P:START        ; Start of program
;*****
;* General setup *
;*****
    MOVEP  #$0000,X:BCR    ; External Program memory has 0 wait states.
                                ; External data memory has 0 wait states.
                                ; Port A pins tri-stated when no external access.
    BFCLR  #$0200,SR       ; Allow IPL (Interrupt Priority Level) 0.
                                ; -- Enable maskable interrupts.
                                ; -- (peripherals, etc.)
;*****
;* PLL setup *
;* (to increase Phi Clock) *
;*****
    MOVEP  #$0180,X:PCRL; Configure:
                                ; (PLLE) PLL disabled (bypassed).
                                ; -- Oscillator supplies Phi Clock.
                                ; (PLLD) PLL Power Down disabled (PLL active).
                                ; -- PLL block active for PLL to attain lock.
                                ; (LPST) Low Power Stop disabled.
                                ; (PS[2:0]) Prescaler Clock disabled.
                                ; Select Phi Clock for Clockout pin (CLKO).
    MOVEP  #$0260,X:PCRO ; Set Feedback Divider to 1/20
    ; ...
    ; insert delay here: wait for PLL lock as specified in data sheet.
    ; ...
    BFSET  #$4000,X:PCRL ; Enable PLL for Phi Clock.
;*****
;* Timer Module setup *
;*****

```

Event Counting with the Timer Module**Example 9-2 Decrementing to 0 and Generating an Interrupt (Continued)**

```
MOVEP #$001C,X:TCR01 ; Configure:
                        ; Timer 0 & 1 disabled
                        ; Timer 0
                        ; -- don't Invert TIO input: detect rising edges
                        ; (irrelevant but mentioned for completeness).
                        ; -- Overflow Interrupt enabled.
                        ; -- Timer Output toggles TIO pin on overflow.
                        ; -- timer clock Event source is Phi Clock /4.

MOVEP #$0000,X:TCR2 ; Timer 2 disabled.
MOVEP #99,X:TCT0 ; Set Timer 0 Count Register to 99 (100 events).
MOVEP #99,X:TPR0 ; Set Timer 0 Preload Register to 99.
BFSET #$0800,X:IPR ; Enable Timer Module interrupts.

; ...

BFSET #$0080,X:TCR01 ; Enable Timer 0

;*****
;* Main routine *
;*****
; ...
TEST ; Test Loop
BRA TEST

TISR ; Timer Interrupt Service Routine
; interrupt code
RTI
```

Example 9-3 shows how to program Timer 0 and Timer 1 to generate a timing signal with a 25% duty cycle.

Example 9-3 Timer Using 25% Duty Cycle

```
;*****
;* 25% duty cycle example *
;* for Timer Module      *
;* of DSP56L811 chip      *
```

Example 9-3 Timer Using 25% Duty Cycle (Continued)

```

;*****
START      EQU    $0040 ; Start of program
BCR        EQU    $FFF9 ; Bus Control Register
IPR        EQU    $FFFB ; Interrupt Priority Register
PCR0       EQU    $FFF2 ; PLL Control Register 0
PCR1       EQU    $FFF3 ; PLL Control Register 1
TCR01      EQU    $FFDF ; Timer 0&1 Control Register
TCR2       EQU    $FFDA ; Timer 2 Control Register
TCT0       EQU    $FFDD ; Timer 0 Count Register
TCT1       EQU    $FFDB ; Timer 1 Count Register
TPR0       EQU    $FFDE ; Timer 0 Preload Register
TPR1       EQU    $FFDC ; Timer 1 Preload Register

;*****
;* Vector setup *
;*****
;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).    |
;+-----+

      ORG    P:$0000      ; Cold Boot
      JMP    START        ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG    P:$E000      ; Warm Boot
      JMP    START        ; Hardware RESET vector (Mode 2)

      ORG    P:START      ; Start of program
;*****
;* General setup *
;*****
      MOVEP  #$0000,X:BCR ; External Program memory has 0 wait states.
                        ; External data memory has 0 wait states.
                        ; Port A pins tri-stated when no external access.
;*****
;* PLL setup *
;* (to increase Phi Clock) *

```

Event Counting with the Timer Module

Example 9-3 Timer Using 25% Duty Cycle (Continued)

```

;*****
MOVEP #$0180,X:PCRL; Configure:
        ; (PLLE) PLL disabled (bypassed).
        ; -- Oscillator supplies Phi Clock.
        ; (PLLD) PLL Power Down disabled (PLL active).
        ; -- PLL block active for PLL to attain lock.
        ; (LPST) Low Power Stop disabled.
        ; (PS[2:0]) Prescaler Clock disabled.
        ; Select Phi Clock for Clockout pin (CLKO).
MOVEP #$0260,X:PCRO ; Set Feedback Divider to 1/20.
; ...
        ; insert delay here: wait for PLL lock as specified in data sheet.
; ...
        BFSET #$4000,X:PCRL ; Enable PLL for Phi Clock.
;*****
;* Timer Module setup *
;*****
MOVEP #$0C0C,X:TCR01; Configure:
        ; Timer 0 & 1 disabled
        ; Timer 0 & 1
        ; -- don't Invert TIO input: detect rising edges.
        ;   (irrelevant but mentioned for completeness).
        ; -- Overflow Interrupt disabled.
        ; -- Timer Output toggles TIO pin on overflow.
        ; -- timer clock Event source is Phi Clock /4.
MOVEP #$0000,X:TCR2 ; Timer 2 disabled
; Setup Timer 0 & 1 for 25% duty cycle:
; high for first 25 of 100 events
MOVEP #00,X:TCT0    ; Set Timer 0 Count Register to 00.
MOVEP #99,X:TPR0    ; Set Timer 0 Preload Register to 99.
MOVEP #25,X:TCT1    ; Set Timer 1 Count Register to 25.
MOVEP #99,X:TPR1    ; Set Timer 1 Preload Register to 99.
BFCLR #$0800,X:IPR  ; Disable Timer Module interrupts.
        ; (superfluous but done for thoroughness).
; ...
BFSET #$8080,X:TCR01 ; Enable Timer 0 & 1

```

Example 9-3 Timer Using 25% Duty Cycle (Continued)

```
;*****
;* Main routine *
;*****

; ...
TEST ; Test Loop
      BRA  TEST
```

9.6 TIMER MODULE LOW POWER OPERATION

In applications requiring minimum power consumption, there are several options for lowering the power consumption of the chip using the timer module. These include:

- Turn off the entire timer module
- Turn off timers not in use
- Lower the timer frequency
- Run a timer in wait mode
- Run a timer in stop mode

The following sections discuss these options individually.

9.6.1 Turning Off the Entire Timer Module

If the timer module is not required by an application, it is possible to shut off the entire module for lowest power consumption by resetting all the TE bits to 0 in the TCR01 and TCR2 registers and setting all the ES[1:0] bits to 01 in these same registers. This provides the prescaler clock to the timers, which is the lowest power setting possible when using the ES bits.

If no other module on the DSP56L811 is using the prescaler clock, it is possible to further reduce the overall power consumption by shutting down the prescaler divider that generates this clock.

9.6.2 Turning Off Any Timer Not in Use

For applications not requiring all of the timers, it is possible to turn off any timer not in use. Individual timers are shut off by resetting the TE bit to 0 for that individual timer and setting the ES[1:0] bits to 01 for that individual timer. These bits are located in register TCR01 or TCR2, depending on which timer is to be turned off.

9.6.3 Low Frequency Timer Operation

For applications requiring a timer to execute at a low frequency, less power is consumed if the timer is clocked using the prescaler clock instead of the Phi clock.

9.6.4 Running A Timer in Wait Mode

Overall power consumption on the DSP56L811 can be reduced using the wait mode of the DSP56800 core. It is possible to place the chip in wait mode for a predetermined amount of time. A timer is enabled and begins counting immediately before executing a WAIT instruction. When the timer reaches 0, a signal is generated that interrupts the DSP56800 core and brings it out of wait mode. All modes of operation in the timer module are available in wait mode.

9.6.5 Running A Timer in Stop Mode

Overall power consumption on the DSP56L811 can be greatly reduced using the stop mode of the DSP56800 core. It is possible to place the chip in stop mode for a predetermined amount of time by setting up Timer 2 using the prescaler clock as input. This timer is enabled and begins counting. Then the DSP56800 core executes a STOP instruction. When Timer 2 reaches 0, a signal is generated that wakes up the DSP core and brings it out of stop mode.

Note: The prescaler clock is the only clock available to the timer module that is active in stop mode. The TIO pin cannot be used as an event counter in stop mode. Also, only Timer 2 is capable of bringing the DSP56L811 out of stop mode, and it performs this function independently of the value of the OIE control bit (see **Overflow Interrupt Enable (OIE) Bit 12, Bit 4** on page 9-7) or the bits in the IPR register (see **3.6.2 DSP56L811 Interrupt Priority Register** on page 3-19).

9.7 TIMER MODULE TIMING DIAGRAMS

The figures in this section illustrate configurations in which the timer can be enabled, disabled, and used.

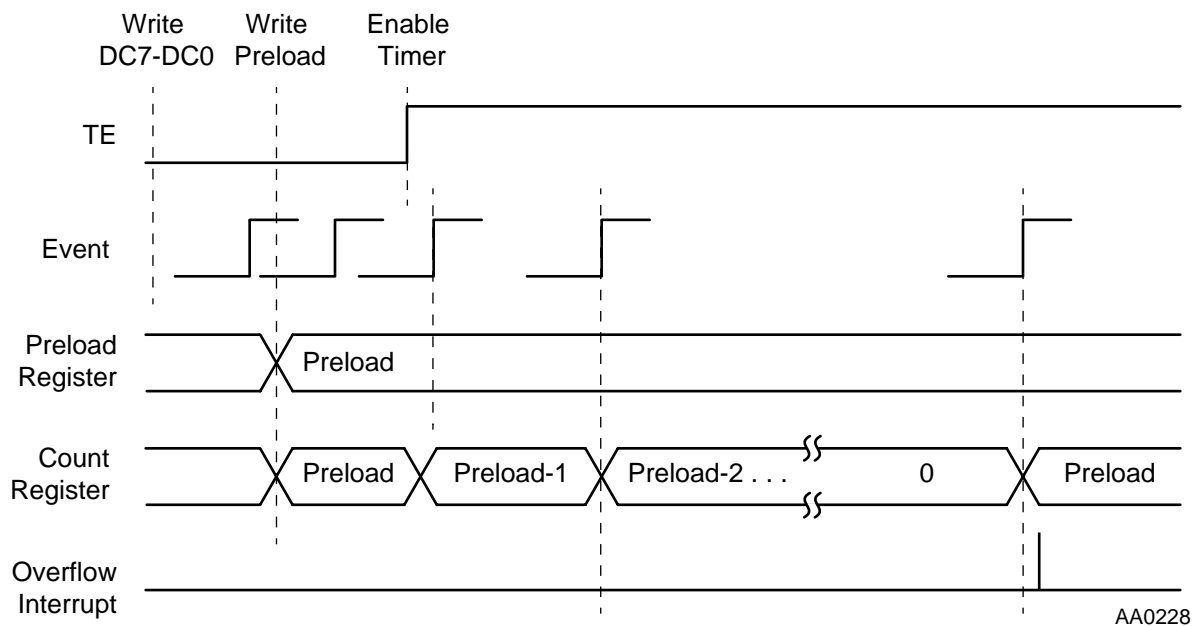


Figure 9-3 Standard Timer Operation with Overflow Interrupt

Timer Module Timing Diagrams

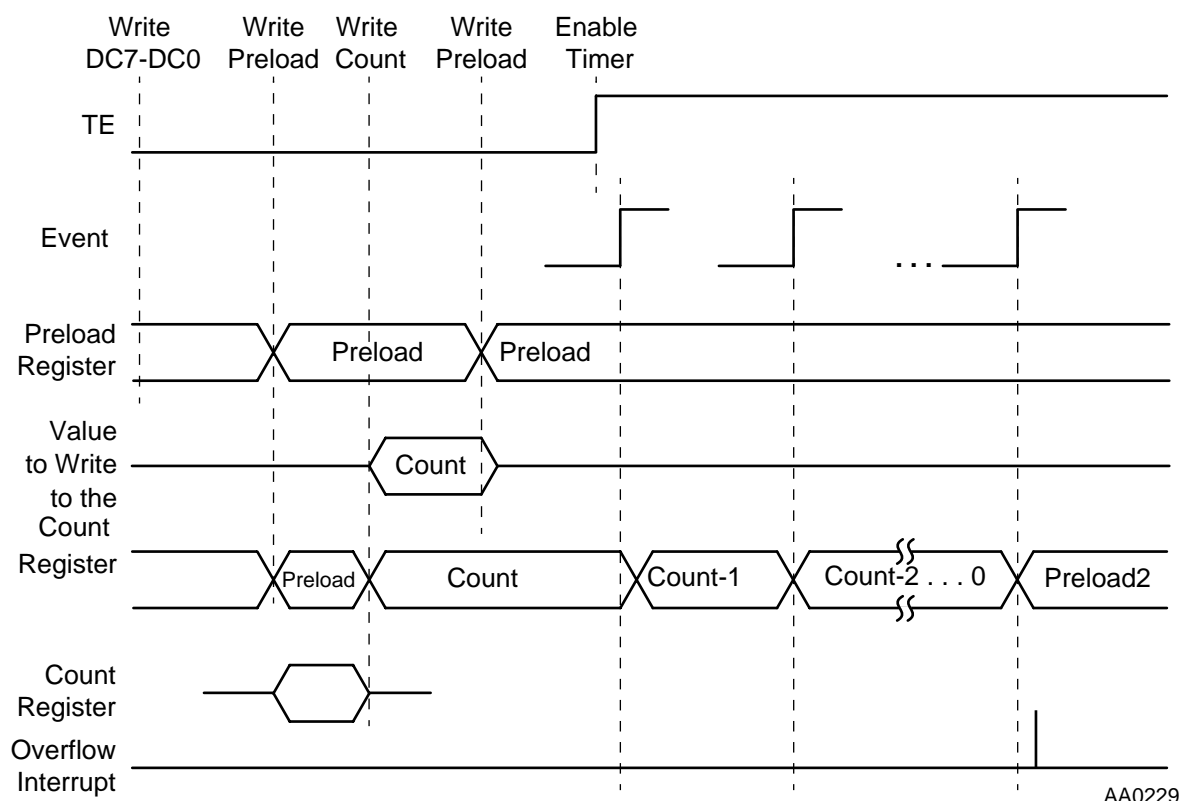
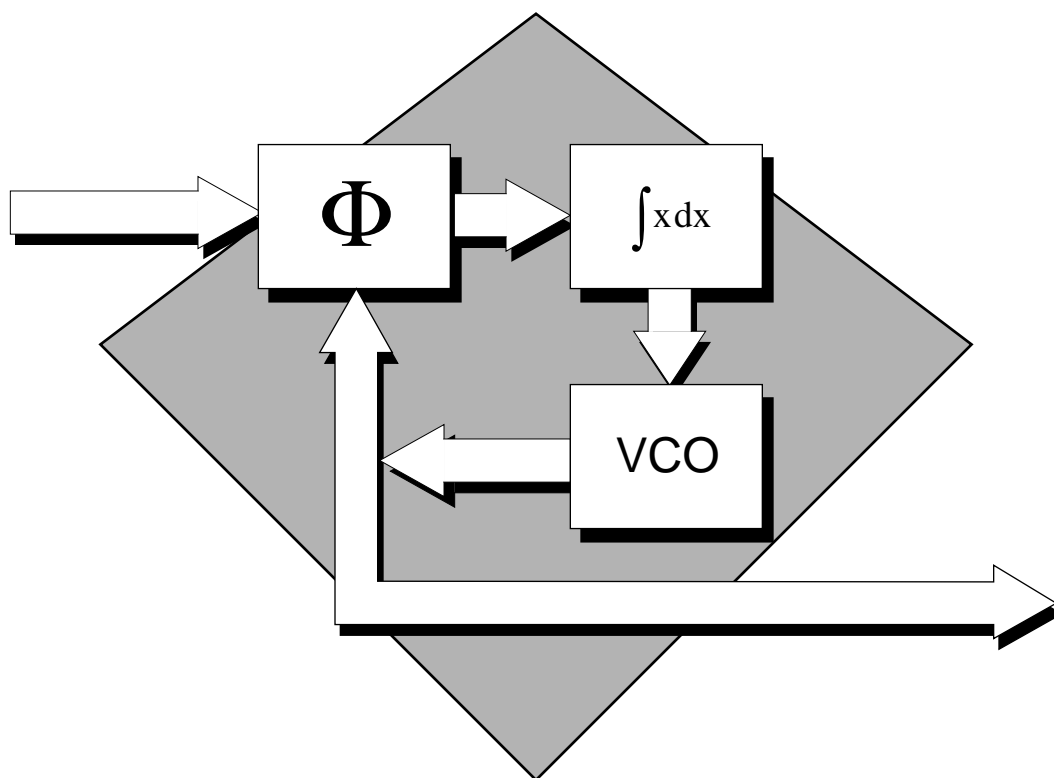


Figure 9-4 Write to the Count Register After Writing the Preload Register When Timer is Disabled



SECTION 10

ON-CHIP CLOCK SYNTHESIS



10.1	INTRODUCTION	10-3
10.2	DSP56L811 TIMING SYSTEM ARCHITECTURE	10-3
10.3	CLOCK SYNTHESIS PROGRAMMING MODEL	10-7
10.4	DSP56L811 STOP AND WAIT MODES	10-12
10.5	PLL LOCK	10-14
10.6	PLL MODULE LOW-POWER OPERATION	10-17

10.1 INTRODUCTION

This section describes the architecture of the clock synthesis module, its programming model, and different low-power modes of operation for the clock synthesis module, which generates the clocking for the DSP56L811. The module generates three clock signals for use by the DSP56800 core and DSP56L811 peripherals. It also contains a phase-locked loop (PLL) that can multiply the frequency, as well as a prescaler/divider used to distribute lower-frequency clocks to peripherals, leading to lower power consumption on the chip. It also selects which clock, if any, is routed to the CLK0 pin of the DSP56L811 chip.

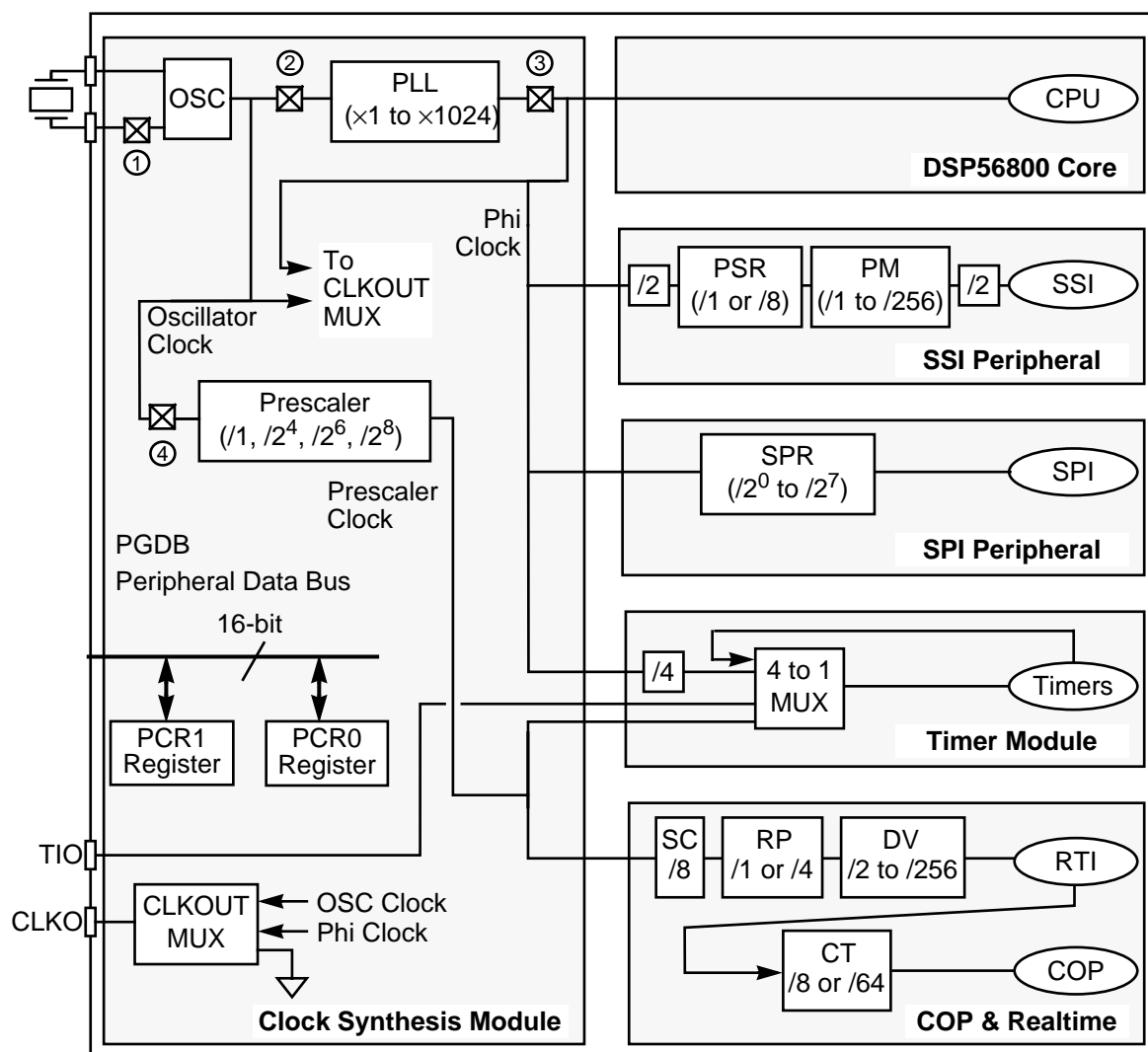
10.2 DSP56L811 TIMING SYSTEM ARCHITECTURE

The DSP56L811 timing system, shown in **Figure 10-1**, has the clock synthesis module at its core. This module is composed of the following five blocks:

- Oscillator
- Phase-locked loop (PLL)
- Prescaler
- Clockout multiplexer (MUX)
- Control registers

Together, these five blocks generate the following three clock signals used for core and peripheral operation:

- Oscillator clock
- Phi clock
- Prescaler clock



- ① All clocks can be disabled at this point in Stop Mode by the LPST control bit.
- ② Clocks are disabled beyond this point in Stop Mode if the PLL is powered down.
- ③ Clocks are disabled beyond this point in Stop Mode.
- ④ Clocks beyond this point can be powered down by the PS[2:0] control bits.

AA0170

Figure 10-1 DSP56L811 Timing System

Typically, the oscillator is attached to an external crystal. It can also be driven by an external oscillator. The output of the oscillator, called the oscillator clock signal, is provided to the prescaler and to the PLL blocks.

The prescaler divides the oscillator clock signal and provides it to the COP/RTI module (discussed in **Section 11, COP/RTI Module**), to the general purpose timer module, and to the Clockout MUX. This signal is called the prescaler clock.

The PLL multiplies up the oscillator clock signal and provides it to the DSP56800 core, to the SSI, to the SPI modules, to the general purpose timer module, and to the Clockout MUX. This multiplied signal is called the Phi clock.

The Clockout MUX delivers one of these three signals (or none) to the CLKO pin.

The two control registers PCR0 and PCR1 control PLL multiplication, power-down, and MUX selects as well as other PLL-related options.

10.2.1 Oscillator

The DSP56L811 supports frequencies from 32KHz to the maximum specified frequency of the chip. The oscillator derives a clock signal from an external crystal. It is also possible to input an external clock directly to the EXTAL pin. In this case, no crystal is used and the XTAL pin is left floating. The output of the oscillator drives the prescaler and the PLL inputs. This output also can provide an input for the Clockout MUX. Detailed information and design guidelines are furnished in the *DSP56L811 Data Sheet (DSP56L811/D)* in **Section 2 Specifications**.

10.2.2 Phase-Locked Loop (PLL)

The PLL is used to multiply up the oscillator clock frequency to the frequency needed by the core and peripherals for operation. For basic operation, the PCR1 register is configured with the PLLD bit set to 0 and the PLLE bit set to 1. When the PLLD bit is set to 0, the PLL loop is powered on, and the oscillator clock is multiplied by the value of the bits in YD[9:0] + 1 at the VCO (the YD bits are contained in the PCR0 register). When the PLLE bit is set to 1, the output of the VCO drives the Phi clock.

By setting the PLLE bit to 0, the PLL is bypassed and the Phi clock is driven directly by the oscillator clock.

The bits in the PCR0 and PCR1 registers are described in **Clock Synthesis Programming Model** on page 10-7. **Figure 10-2** shows a block diagram of the PLL.

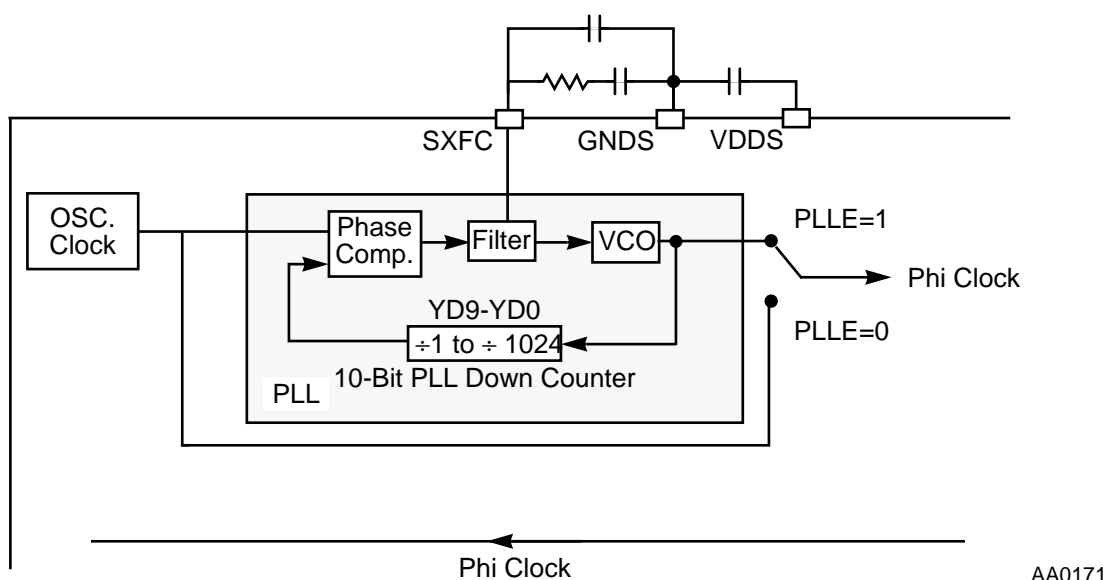


Figure 10-2 PLL Block Diagram

10.2.3 Prescaler

The prescaler is used when a higher frequency input clock or crystal (such as 1 MHz or more) is required in an application. The prescaler provides a slower clock signal as input to the Realtime Interrupt and COP timer. In addition, this low-frequency clock signal can also be used as input to the timers in the timer module. When a 32 kHz or 38.4 kHz crystal is used with the DSP56L811, it is appropriate to set the prescaler to divide by 1 (effectively bypassing the prescaler).

Based on the value of the PS[2:0] bits, the prescaler clock frequency is derived from the oscillator clock as shown in the equations in **Figure 10-3**.

$\text{Prescaler Clock Frequency} = \text{Oscillator Clock Frequency} / (4 \times 2^{\text{PS}[2:0]})$ $\text{Prescaler Clock Frequency} = \text{Oscillator Clock Frequency (when PS}[2:0] = 000)$
--

AA0214

Figure 10-3 Prescaler Clock Equations

Whenever the PS0 bit is set to 1, the prescaler is disabled, regardless of the value of the PS[2:0] bits. See **Clock Synthesis Programming Model** for detailed information on PS bit values and corresponding divide rates.

Note: The maximum frequency of the prescaler clock is limited to $1/16$ of the maximum clocking frequency of the part. This means that for a 40 MHz DSP56L811, the clocking frequency of the prescaler clock must be less than or equal to 2.5 MHz.

10.2.4 Clockout MUX

The Clockout MUX selects which clock is provided to the CLKO pin. Disable this pin for lowest power operation using the control bits in PCR1. For testing and some user applications, it is desirable to provide the Phi clock on this pin. In some cases, it may be desirable to provide the oscillator clock on the CLKO pin.

10.2.5 Control Registers

The PCR0 and PCR1 control registers provide the majority of the user control over the clock synthesis module. Individual bits and their functions are described in the next section.

10.3 CLOCK SYNTHESIS PROGRAMMING MODEL

The clock synthesis module provides two control registers to manage the frequency, signal paths, and outputs of the DSP56L811 clocks. These registers are:

- PCR1—PLL Control Register 1
- PCR0—PLL Control Register 0

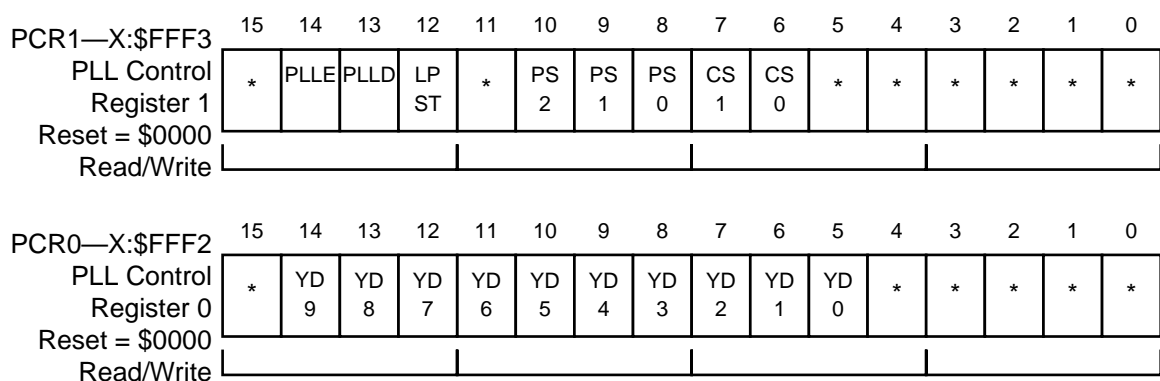
Clock signal control is also provided by additional registers within the following peripherals:

- SSI
- SPI
- COP/RTI

Clock Synthesis Programming Model

Note: Clock signals used within the peripherals can be provided as clock outputs. Users should be reminded that changing a clock may change the behavior of another peripheral.

The clock synthesis programming model is shown in **Figure 10-4**.



* Indicates reserved bits, written as 0 for future compatibility

AA0173

Figure 10-4 Clock Synthesis Programming Model

10.3.1 PLL Control Register 1 (PCR1)

The PLL Control Register 1 (PCR1) is a 16-bit read/write register used to direct the operation of the on-chip clock synthesizer. The PCR1 control bits are defined in the following text. All PCR1 bits are cleared on reset.

10.3.1.1 PLL Enable (PLLE) and PLL Power Down (PLLD) Bits 14 and 13

When the PLL Enable control bit (bit 14) is set, the DSP56L811 system clock (Phi Clock) is generated by the on-chip PLL. The state of the PLL is defined by the PLL Power Down control bit (bit 13). The PLLE and PLLD bits work together to control PLL operation, as shown in **Table 10-1**.

When the PLLD bit is set to 1, the PLL is in the power down mode, a low-current mode in which the VCO is inactive. When the PLLD bit is cleared (set to 0), the PLL is in the active mode. Before turning the PLL off, clear the PLLE bit to bypass the PLL. Then put the PLL in power down mode by setting PLLD to 1. Setting the PLLD bit to 1 powers down the complete PLL block, including the PS and YD registers, described in subsections **10.3.1.3** and **10.3.2.1**, respectively.

The PLLD bit should not be set when the PLLE bit is set. Both the PLLD bit and the PLLE bit are cleared (set to 0) when the chip is reset.

Note: The STOP instruction does not power down the PLL if the PLL is not powered down (PLLD=0) when entering stop mode.

Table 10-1 PLL Operations

PLLE	PLLD	Phi Clock	PLL Mode
0	0	Oscillator Clock	Active
0	1	Oscillator Clock	Power Down
1	0	Oscillator Clock x [YD+1]	Active
1	1	(Reserved)	(Reserved)

10.3.1.2 Low Power Stop (LPST) Bit 12

The Low Power Stop control bit (bit 12) is used to place the chip in the lowest power configuration when entering stop mode. If this bit is set to 1 when entering stop mode, the clock is disabled at the crystal oscillator. If this bit is 0, then the oscillator continues running in stop mode. The LPST bit is cleared (set to 0) on DSP reset.

10.3.1.3 Prescaler Divider (PS2-PS0) Bits 10-8

The Prescaler Divider control bits (bits 10-8) are used to pass, disable, or divide the oscillator clock by several different divide ratios: 2^4 , 2^6 , and 2^8 . The output of the divider can be used as the operating clock for the timer module or the COP and realtime timers. The PS bits are cleared on DSP reset.

The prescaler divider is used for systems with higher frequency crystals to provide a slower clocking frequency near 32 kHz for the realtime and COP timers, discussed in detail in **Section 11, COP/RTI Module**. Typically a user would set the divider to divide by one for systems with a 32.0 kHz or 38.4 kHz crystal, but would use the divider when a higher frequency crystal is used. Likewise, it is possible to disable this divider and its output clock for low-power applications that do not require a realtime or COP timer and do not require a low-frequency clock for the timer module.

The prescaler should always be set up with the correct division ratio before any peripheral using the prescaler clock is enabled.

Table 10-2 PS Divider Programming

PS[2:0]	Function	Comments
000	divide by 1	Used for 32.0 and 38.4 KHz crystals
001	Disabled	For low-power applications not requiring the Realtime or COP Timers
010	divide by 16	Typically for higher frequency crystals
011	(Reserved)	—
100	divide by 64	Typically for higher frequency crystals
101	(Reserved)	—
110	divide by 256	Typically for higher frequency crystals
111	(Reserved)	—

Note: The maximum frequency of the prescaler clock is limited to $1/16$ of the maximum clocking frequency of the part. This means that for a 40 MHz DSP56L811, the clocking frequency of the prescaler clock must be less than or equal to 2.5 MHz.

Changing the prescaler control bits when the COP timer is enabled via the CPE bit (in the COPCTL register) does *not* result in a change of the prescaler divider. This prevents an application from accidentally disabling the COP timer by disabling the prescaler clock. If the CPE bit is set, then the PS bits can still be written, but the prescaler division ratio is not changed. See **11.2.3 COP and RTI Control Register (COPCTL)** for a description of the COPCTL register.

Note: There is a restriction when setting the prescaler's division ratio. The case where the prescaler is set to divide by one *and* the PLL is set for a multiplication factor of one when the PLL provides the Phi clock (PLLE = 1) is *not allowed*. Violating this restriction results in faulty prescaler clock synchronization in the peripherals.

10.3.1.4 CLK0 Select (CS1-CS0) Bits 7-6

The CLK0 Select control bits (bits 7-6) are used to enable one of two different clocks to the CLK0 pin, or to disable all clocks to this pin. After DSP reset, the Phi Clock

output is provided on the CLK0 pin. The other options are presented in **Table 10-3** below. The CS bits are cleared on DSP reset.

Table 10-3 CLKOUT Pin Control

CS1	CS0	CLK0
0	0	Phi Clock
0	1	(Reserved)
1	0	Oscillator Clock
1	1	Disabled

10.3.1.5 Reserved PCR1 Register Bits

Bits 15, 11, and 5-0 are reserved and are read as 0 during read operations. These bits should be written with 0 to ensure future compatibility.

10.3.2 PLL Control Register 0 (PCR0)

The PLL Control Register (PCR0) is a 16-bit read/write register used to direct the operation of the on-chip clock synthesis. The PCR0 register controls the frequency programming of the PLL. The PCR0 control bits are defined in the following text.

All bits of PCR0 are cleared by DSP hardware reset.

10.3.2.1 Feedback Divider (YD9-YD0) Bits 14-5

The Feedback Divider control bits (bits 14-5) control the down counter in the feedback loop, causing it to divide by the value YD+1 where YD is the value contained in the control bits.

The resulting Phi clock signal must be within the limits specified in the *DSP56L811 Data Sheet (DSP56L811/D)*. The frequency of the VCO should also remain higher than the specified minimum value. Recommended PLL multiplication factors for the DSP56L811 are provided in **Section 2, Specifications**, of the *DSP56L811 Data Sheet (DSP56L811/D)*.

Note: Some multiplication factors require setting the PS[2:0] bits in the PCR1 register. This may affect timing on the various peripheral modules.

The YD bits are cleared on DSP reset.

10.3.2.2 Reserved PCR0 Register Bits

Bits 15 and 4-0 are reserved and are read as 0 during read operations. These bits should be written with 0 to ensure future compatibility.

10.4 DSP56L811 STOP AND WAIT MODES

The DSP56L811 can be configured for very low-power consumption in wait mode or minimal power consumption in stop mode, based on application needs. In wait mode, all internal core clocks are gated off, but the Phi clock continues to operate so that peripherals continue to function and can bring the chip out of wait mode by using a peripheral interrupt. In stop mode, the Phi clock and all internal core clocks are gated off. Stop mode uses less power than wait mode.

Before entering these modes, it is possible to enable or disable these blocks individually:

- CLKO pin
- PLL Module
- COP/RTI Module
- Timer Module
- SPI Module
- SSI Module

In addition, it is also possible to disable the Prescaler block, but this in turn disables the COP/RTI and the Timer Module blocks. All blocks left enabled continue to run in stop mode, except the SPI and SSI. The Timer Module and COP/RTI can bring the chip out of stop mode. The SPI and SSI are always powered down in stop mode.

10.4.1 Options for Clock Synthesis in Stop Mode

Several options for the clock synthesis block are available to the user in stop mode.

- Leave PLL enabled—low power, short wake-up time since PLL already stabilized
- Disable PLL—lower power, long wakeup time for PLL to stabilize
- Leave prescaler enabled—allows COP and realtime timers to continue operating

- Disable prescaler—lower power, disables clock to COP and realtime timers
- Disable oscillator—lowest power, all internal clocks disabled, longest wake-up time

Leaving the PLL enabled in stop or wait mode avoids the need to wait for the PLL to re-lock upon exiting. The following are examples of low-power configurations in stop mode.

10.4.2 Stop Mode—PLL, COP, Realtime Clock, Timer Module and CLKO Pin Enabled

In this mode, the DSP56800 core and the SPI and SSI peripherals are placed in low-power stop mode, in which all clocks are gated off. The PLL remains running and locked, the COP and realtime clock timers can still continue functioning, and a clock waveform can be provided on the CLKO pin, if desired. In this mode the oscillator is still running.

Note: It is also possible to leave the Timer Module running if clocked with the prescaler clock. This stop mode consumes the most power.

This mode is entered by executing a STOP instruction with the PLL, COP/RTI, and Timer Module peripherals still enabled. The Timer Module must be clocked with the Prescaler Clock.

Note: The CLKO pin can be disabled independently using the CS[1:0] control bits in the PCR1 register, described in **CLKO Select (CS1-CS0) Bits 7-6** on page 10-10.

10.4.3 Stop Mode—COP, Realtime Clock and CLKO Pin Enabled

In this mode, the DSP56800 core and the PLL, SPI, SSI, and Timer Module peripherals are placed in low-power stop mode where all clocks are gated off. The PLL loses lock in this condition. The COP and realtime clock timers can still continue functioning, and a clock waveform can be provided on the CLKO pin, if desired. In this mode the oscillator is still running.

This mode is entered by executing a STOP instruction with the PLL disabled and the COP/RTI peripheral still enabled. The CLKO pin can also be disabled independently using the CS[1:0] control bits in the PCR1 register.

10.4.4 Stop Mode—PLL and CLKO Pin Enabled

In this mode, the DSP56800 core and the COP/RTI, SPI, SSI, and Timer Module peripherals are placed in low-power stop mode where all clocks are gated off. The PLL remains locked and running in this condition. A clock waveform can be provided on the CLKO pin, if desired. In this mode the oscillator is still running.

This mode is entered by executing a STOP instruction with the PLL enabled and the COP/RTI peripheral disabled. The CLKO pin can also be disabled independently using the CS[1:0] control bits in the PCR1 register.

10.4.5 Stop Mode—CLKO Pin Enabled

In this mode, the DSP56800 core and the COP/RTI, PLL, SPI, SSI, and Timer Module peripherals are placed in low-power stop mode where all clocks are gated off. The PLL loses lock in this condition. A clock waveform is provided on the CLKO pin. In this mode the oscillator is still running. It may be necessary to wait for the PLL to re-lock after exiting stop mode.

This mode is entered by executing a STOP instruction when the PLL and COP/RTI peripherals are both disabled and the CLKO pin is enabled.

10.4.6 Stop Mode—Everything Disabled

In this mode, the DSP56800 core and the COP/RTI, PLL, SPI, SSI, and Timer Module peripherals are placed in low-power stop mode where all clocks are gated off. The PLL loses lock in this condition. The CLKO pin is disabled and the oscillator is disabled as well. If an external crystal is used, the processor must wait for the crystal to stabilize before exiting stop mode. It may also be necessary to wait for the PLL to re-lock.

This mode is entered by executing a STOP instruction when the LPST bit in the PCR1 register is set. This is the lowest power stop mode available.

10.5 PLL LOCK

There are several conditions when it is necessary to wait for the PLL to lock:

- When the chip first powers up
- When the PLL is taken out of its power down state (clearing the PLLD bit when it had previously been set)
- When the frequency of the PLL is changed by modifying the YD bits in the PCR0 register

Note: Changing the PLL frequency may require changing external filter components, and is not recommended. See the *DSP56L811 Data Sheet (DSP56L811/D)* for more information.

In each of these cases, it is necessary to wait until the PLL locks on its final frequency before the PLL clock is sent to the DSP56800 core and the peripherals (PLLE = 1). PLL lock time is provided in the *DSP56L811 Data Sheet (DSP56L811/D)*. Failure to wait until the PLL locks can result in improper processing states, software error, and other problems.

10.5.1 PLL Programming Sequence

Example 10-1 provides an example of how to program the PLL to multiply by a factor of 20 and wait for the PLL to stabilize.

Example 10-1 Programming the PLL

```

;*****
;* PLL Setup example *
;* of DSP56L811 chip *
;*****

PCR0      EQU    $FFF2 ; PLL Control Register 0
PCR1      EQU    $FFF3 ; PLL Control Register 1

;*****
;* PLL setup          *
;* (to increase Phi Clock) *
;*****

    MOVEP    #$0180,X:PCR1 ; Configure:
                                ; (PLLE) PLL disabled (bypassed)
                                ; -- Oscillator supplies Phi Clock.
                                ; (PLLD) PLL Power Down disabled (PLL active).
```

Example 10-1 Programming the PLL (Continued)

```
        ; -- PLL block active for PLL to attain lock.
        ; (LPST) Low Power Stop disabled.
        ; (PS[2:0]) Prescaler Clock disabled.
        ; (CS[1:0]) Clockout pin (CLKO) sends Phi Clock.
MOVEP #$0260,X:PCRO ; Set Feedback Divider to 1/20
; ...
; insert delay here: wait for PLL lock as specified in data sheet
; ...
BFSET #$4000,X:PCRL ; Enable PLL for Phi Clock.
```

10.5.2 Changing the PLL Frequency

To change the output frequency of the PLL (by reprogramming the YD bits) while the PLL output is used by the DSP56800 core (PLLE = 1; PLLD = 0), perform the following sequence of operations:

1. Clear the PLLE bit to switch back to the oscillator clock.
2. Program the YD bits (only after clearing PLLE).
3. Wait until the PLL has locked. See the *DSP56L811 Data Sheet (DSP56L811/D)* for settling time specifications.
4. Set the PLLE bit.

Note: Changing the PLL frequency may require changing external filter components, and is not recommended. See the *DSP56L811 Data Sheet (DSP56L811/D)* for more information on supported PLL frequencies and recommended external filter component values.

10.5.3 Turning Off the PLL Before Entering Stop Mode

To turn off the PLL before entering stop mode, execute the following sequence before issuing the STOP instruction:

1. Clear the PLLE bit to switch back to the Oscillator Clock.
2. Set the PLLD bit to 0 to power down the PLL.
3. Execute the STOP instruction.

Note: The PLL can be left running when entering stop mode. This allows a faster exit to normal mode. There is no wait for PLL lock because the PLL continues to run in stop mode.

10.6 PLL MODULE LOW-POWER OPERATION

In applications requiring minimum power consumption, there are several options for lowering the power consumption of the chip within the Clock Synthesis Module in stop or wait mode. These are discussed individually below.

10.6.1 Turning Off the Entire Clock Synthesis Module

If no internal clocks are required by an application in stop mode, shut off the entire module for lowest power consumption. This is done by resetting the PLLD bit to 1 (PLLE must already be set to 0), setting the PS[2:0] bits (in the PCR1 register) to 001, setting the CS[1:0] bits (in the PCR1 register) to 11, and setting the LPST bit (in the PCR1 register) to 1. See **PLL Control Register 1 (PCR1)** on page 10-8 for details on the PCR1 register.

When the LPST bit is set to 1, an additional period of time is required for crystal oscillator stabilization. Upon exiting stop mode, it is necessary to wait for the PLL to stabilize again (re-lock). Both these times are specified in the *DSP56L811 Data Sheet (DSP56L811/D)*.

10.6.2 Turning Off the Prescaler Divider When Not in Use

For applications not requiring a Prescaler Clock to any peripherals, turn off the Prescaler Divider by setting the PS[2:0] bits in PCR1 to 001.

Note: The Prescaler Divider can be turned off independently from the PLL or CLKO pin.

10.6.3 Turning Off the PLL When Not in Use

For applications in which the time required for the PLL to re-lock when exiting stop mode is not an issue, the PLL can be turned off by setting the PLLD bit in the PCR1

PLL Module Low-power Operation

register to 1. (See **PLL Control Register 1 (PCR1)** on page 10-8.) This can be done only after the PLLE bit in PCR1 has been set to 0.

Note: The PLL can be turned off independently from the Prescaler Divider or CLK0 pin. Upon exiting stop mode, the PLL must be re-enabled and it is necessary to wait for the PLL to stabilize again (re-lock).

10.6.4 Turning Off the CLK0 Pin When Not in Use

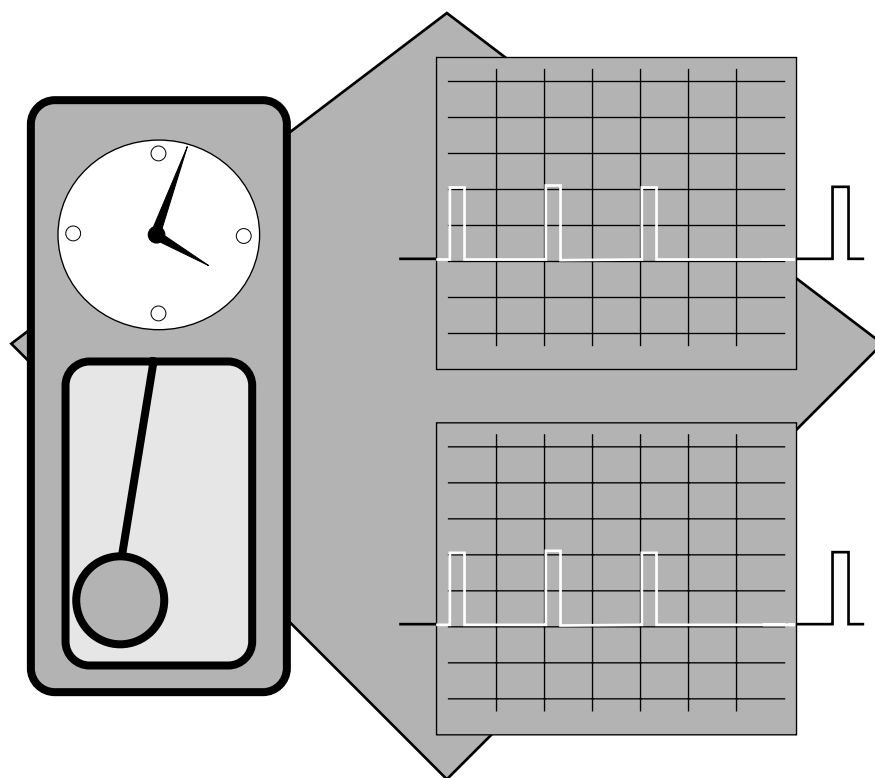
For applications where no external clock out pin is required, it is recommended to turn off the CLK0 pin to lower power and reduce switching on the pins. This can be accomplished by setting the CS[1:0] bits to 11. (See **PLL Control Register 1 (PCR1)** on page 10-8.)

Note: The CLK0 pin can be turned off independently from the PLL or prescaler divider.



SECTION 11

COP/RTI MODULE



11.1 INTRODUCTION 11-3

11.2 COP AND RTI 11-3

11.1 INTRODUCTION

This section describes the Computer Operating Properly (COP) and Realtime Interrupt (RTI) module (COP/RTI) provided on the DSP56L811.

The COP/RTI module provides two separate functions: a watchdog-like timer and a periodic interrupt generator. The COP timer guards processor activity and provides an automatic reset signal if a failure occurs. Both functions are contained in the same block because the input clock for both comes from a common clock divider.

11.2 COP AND RTI

The COP timer protects against system failures by providing a means to escape from unexpected input conditions, external events, or programming errors. Once started, the COP timer must be reset by software on a regular basis so that it never reaches its time-out value. When the COP timer reaches its time-out value, an internal reset is generated within the chip and the COP reset vector is fetched. It is assumed that if the COP reset is not received from the program, a system failure has occurred.

The COP functionality is typically used by software to ensure the chip is operating properly. Software must periodically service the COP timer by correctly writing to the COPRST register before the COP timer times out. The COP timer has its own reset vector. This allows the reset recovery from a COP time-out to differ from the reset procedure done after a hardware reset.

COP reset is very similar to a hardware reset through the $\overline{\text{RESET}}$ pin. The minor differences are the address from which the reset vector is fetched, and the duration during which reset is asserted.

The Realtime Interrupt capability provides a periodic interrupt in an application. It consists of a long decrementing counter chain that runs continuously when enabled. When it reaches 0, a status flag is set and an interrupt is generated (optionally, when enabled by the user). The Realtime Interrupt has its own interrupt vector location to reduce overhead in interrupt servicing.

11.2.1 COP and Realtime Timer Architecture

Figure 11-1 shows a block diagram of the COP/RTI timer module. This module contains a programmable divider chain for dividing the Prescaler Clock to a periodic rate that can be used as a realtime interrupt. This realtime clock is further divided down to get a COP timer reset. The COP and RTI Control Register (COPCTL) is used to set up the peripheral and program the divide ratios.

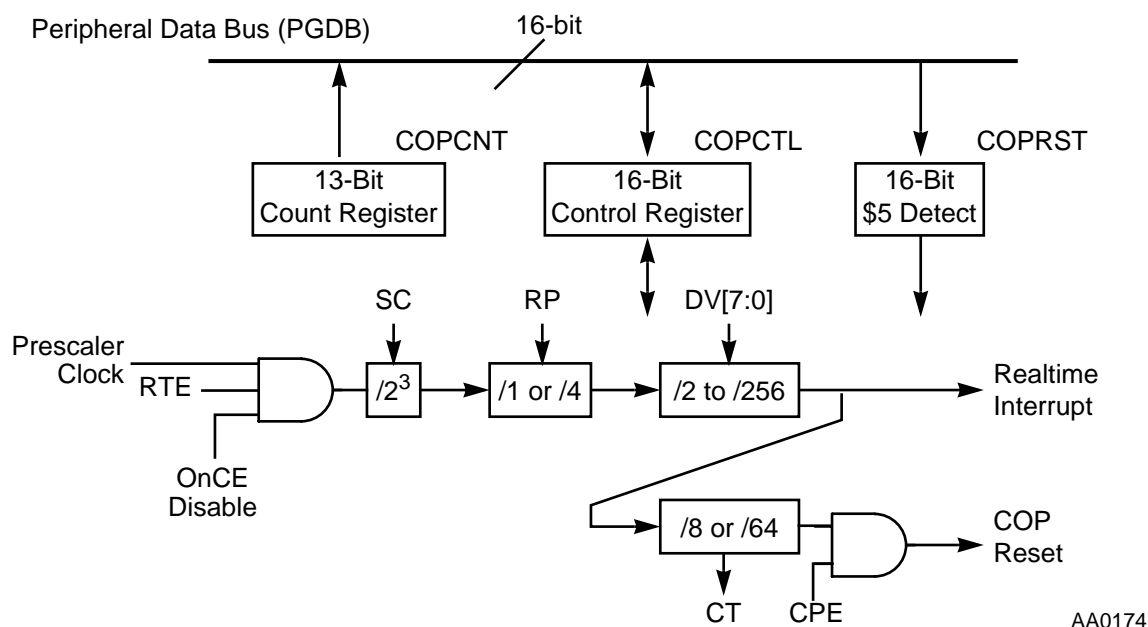


Figure 11-1 Realtime and COP Timer Block Diagram

Note: The maximum frequency of the prescaler clock is limited to $1/16$ of the maximum clocking frequency of the part. This means that for a 40 MHz DSP56L811, the clocking frequency of the Prescaler clock must be less than or equal to 2.5 MHz.

The Realtime timer and the COP timer share the Scaler (SC), Realtime Prescaler (RP), and Realtime/COP Divider (DV) dividers in the clock divider chain. The current value of the Realtime timer can be determined at any time by reading the COPCNT register, the bits of which reflect the shared components of the COP/RTI divider chain. These shared components comprise the RTI timer.

The COP timer is a timer whose clock source is cascaded from the Realtime timer. It consists of the COP Timer Divider (CT). The COP timer counts from either 7 or 63 down to 0, and then provides the COP reset signal, which resets the DSP chip. The

COP timer cannot be read. Its only function is to count down to 0 and send a COP reset signal, unless the COP timer itself is reset. The COP reset sequence, provided in **Example 11-1**, must be programmed to run periodically. Sending this reset sequence is analogous to renewing a library book before it is due.

The COPCTL register reflects the control status of both the Realtime timer and the COP timer. The COP/RTI peripheral is enabled by the Realtime Enable (RTE) bit in the COPCTL register. In addition, the On-Chip Emulation peripheral (OnCE) within the DSP56800 core can disable the counting in the Realtime and COP timers to prevent counting when the chip is no longer executing instructions, but instead is in debug mode. This is useful for debugging realtime systems.

11.2.2 COP and Realtime Timer Programming Model

The COP/RTI block contains the following registers:

- COP/RTI Control Register (COPCTL)
- COP/RTI Count Register (COPCNT)
- COP Reset Register (COPRST)

COP and RTI

These three registers are shown in **Figure 11-2** and explained in the following paragraphs.

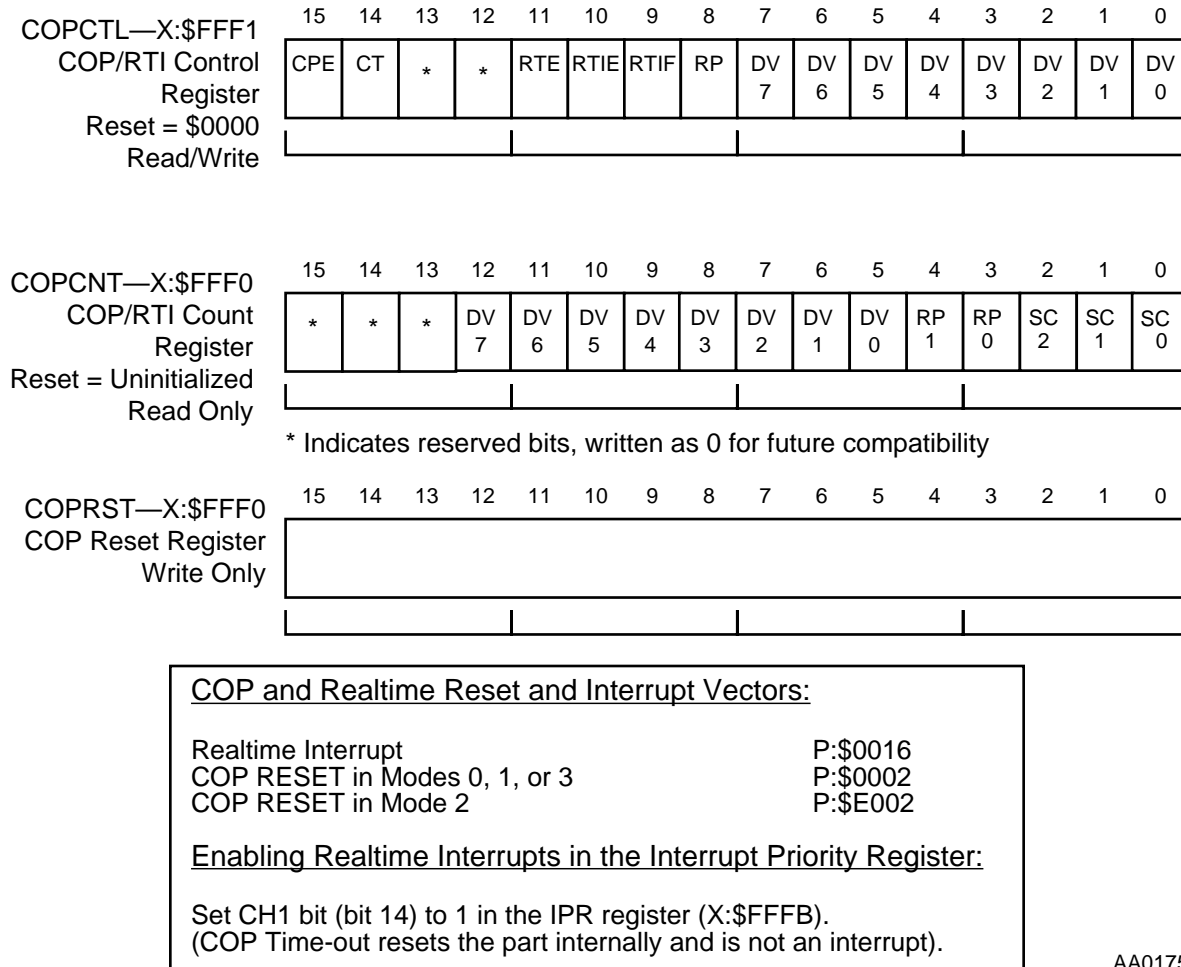


Figure 11-2 Realtime Interrupt and COP Timer Programming Model

11.2.3 COP and RTI Control Register (COPCTL)

The COP and RTI Control register (COPCTL) is a 16-bit read/write register used to program both the COP timer and the Realtime timer. The COPCTL register is reset to \$0000 on hardware reset. When changing the bits in this register, it is important to follow the guidelines in **Programming the COP and RTI Timers** on page 11-10. The bits of this register are defined in the following text.

Note: This register cannot be written unless preceded by the sequence documented in **Programming the COP and RTI Timers** on page 11-10. This ensures that the COPCTL register cannot be modified if the computer is not operating properly. The COPCTL register can be read at any time.

11.2.3.1 COP Enable (CPE) Bit 15

The COP Enable control bit (bit 15) enables the COP timer functionality. Both the RTE bit and the CPE bit must be set for the COP timer to function. The CPE bit is cleared on hardware reset.

11.2.3.2 COP Timer Divider (CT) Bit 14

The COP Timer Divider control bit (bit 14) is used to program the division ratio for the COP timer. The CT bit is cleared on hardware reset. **Table 11-1** shows how this bit is set.

Table 11-1 COP Timer Divider Definition

CT	Division
0	/8
1	/64

11.2.3.3 Realtime Timer Enable (RTE) Bit 11

The Realtime Timer Enable control bit (bit 11) is used to enable the Realtime and COP timer functionality. The Realtime timer can operate without the CPE bit being set, but both the RTE bit and the CPE bit must be set for the COP timer to operate. The RTE bit is cleared on hardware reset.

Note: Clearing the RTE bit disables the input clock to both the Realtime timer and COP timer, and can be used to reduce power consumption in systems that do not require these functions.

11.2.3.4 Realtime Timer Interrupt Enable (RTIE) Bit 10

The Realtime Interrupt Enable control bit (bit 10) is used to enable interrupts from the Realtime timer. When the RTIE bit is cleared, the interrupt is disabled and any pending Realtime interrupt is cleared. The RTIE bit is cleared on hardware reset.

The interrupt vector for the Realtime interrupt is \$0016. As with all on-chip peripheral interrupts for the DSP56L811, the SR register must first be set to enable maskable interrupts (interrupts of level IPL1). Next, the CH1 bit (bit 14) in the IPR register must also be set to enable this interrupt. (See **3.6.2 DSP56L811 Interrupt Priority Register** on page 3-19 for more information.) Finally, set the RTIE bit to enable the Realtime interrupt.

11.2.3.5 Realtime Timer Interrupt Flag (RTIF) Bit 9

The Realtime Interrupt Flag (bit 9) is automatically set to 1 at the end of every RTI period, that is, when the Realtime timer reaches 0. This read-only bit is cleared by writing a 1 to bit 9, RTIF, in the COPCTL register. The RTIF bit is cleared on hardware reset.

11.2.3.6 Realtime Prescaler (RP) Bit 8

The Realtime Prescaler control bit (bit 8) is used to program the prescaler for the Realtime timer. **Table 11-2** shows the different available selections. The RP bit is cleared on hardware reset.

Table 11-2 Realtime Prescaler Definition

RP	Division
0	/1
1	/4

11.2.3.7 Realtime/COP Divider (DV7-DV0) Bits 7-0

The Realtime/COP Divider control bits (bits 7-0) are used to program the last divider in the Realtime Clock chain. When the Realtime Counter decrements to 0, the eight bits of the Realtime Counter are reloaded with the value in the DV[7:0] bits. To set the counter for division by N counts, load the DV bits with the value (N-1). The DV bits are cleared on hardware reset.

Note: Divide by 1 (DV[7:0] = \$0) is not allowed for this divider. All other divider values from 2 to 256 are allowed. Since the value of the DV[7:0] bits is \$0 upon reset (an illegal value), this register must be written to before the Realtime or COP timer is first used.

11.2.3.8 Reserved COPCTL Register Bits

Bits 13-12 are reserved and are read as 0 during read operations. These bits should be written with 0 for future compatibility.

11.2.4 COP and RTI Count Register (COPCNT)

The COP and RTI Count Register is a 16-bit read-only register reflecting the value of the COP/RTI divider chain. The COPCNT register is written by the divider chain on the edge of the Prescaler Clock opposite that used to clock the divider chain. The COPCNT register can be read at anytime.

11.2.4.1 Realtime/COP Divider (DV7-DV0) Bits 12-5

When the COPCNT register is loaded, the Realtime/COP Divider bits (bits 12-5) show the value programmed into the last divider in the Realtime Clock chain.

11.2.4.2 Realtime Prescaler (RP) Bits 4-3

When the COPCNT register is loaded, the Realtime Prescaler bits (bits 4-3) show the value loaded into the prescaler by setting the RP bit in the COPCTL register.

Table 11-2 lists the resulting values.

Table 11-3 Realtime Prescaler Bits in COPCNT

RP in COPCTL	Division	RP[1:0] value
0	/1	00
1	/4	11

11.2.4.3 Scaler (SC2-SC0) Bits 2-0

When the COPCNT register is loaded, the Scaler bits (bits 2-0) are set to 111.

11.2.4.4 Reserved COPCNT Register Bits

Bits 15-13 are reserved and are read as 0 during read operations. When the COPCNT register is loaded, no value is written to these bits.

11.2.5 COP Reset Register (COPRST)

The COPRST register is a 16-bit write-only register, and is used for two purposes.

The first use of this register is for resetting the COP timer before it times out. The COP timer, once enabled, can only be reset by writing a sequence in the correct order to the COPRST register. The required sequence is described in the following section, **Programming the COP and RTI Timers**. This resets the COP timer to its maximum value, and it begins counting down again. The COP timer is the last divide-by-8 or divide-by-64 portion of the counter chain.

Note: Writing to this register does not affect the portion of the timing chain shared by both the Realtime and the COP timers.

The second use of this register is to enable writes to the COPCTL register. It is very important that the COPCTL register is not accidentally overwritten if the computer is not operating properly, so writes to this register are enabled only after a correct sequence is written to the COPRST register.

11.3 PROGRAMMING THE COP AND RTI TIMERS

Accidental writes to the COPCTL register are prevented by requiring the user to write a specific sequence to the COPRST register before writes to the COPCTL register are enabled.

Note: Writing this sequence also has the effect of resetting the COP timer.

The exact sequence is presented here:

1. Write the value \$5555 to the COPRST register.
2. Execute a NOP instruction.
3. Write the value \$AAAA to the COPRST register.
4. Execute a NOP instruction.
5. Write the value \$5555 to the COPRST register.
6. Execute a NOP instruction.
7. At this point in the sequence, it is now possible to write the COPCTL register. Use the following sequence to modify the bits in the COPCTL register.
8. Write the value \$AAAA to the COPRST register. At this point, the COP timer is reset to its maximum value—7 if CT=0 and 63 if CT=1. This final write also resets the sequence mechanism and disables writing to the COPCTL register, so it is necessary to begin again at step 1.
9. Execute a NOP instruction.

It is also important to modify the COPCTL bits in the proper order when programming the COPCTL register:

1. Write the correct sequence to COPRST register to enable writes to the COPCTL register, as shown in the previous example.
2. Clear the RTE bit in COPCTL using a BFCLR instruction.
3. Change any of the following COPCTL bits as desired: CPE, CT, RTIE, RP, or DV[7:0].
4. Set the RTE bit in COPCTL using a BFSET instruction.
5. Disable writes once again to the COPCTL register by writing the value \$AAAA to the COPRST register, as described in step 9 in the preceding sequence.

Note: If the RTE bit is set, bits in the COPCTL register can be written but a malfunction in the COP/RTI module can occur. Always clear the RTE bit before writing to the COPCTL register. This ensures proper operation of this module.

11.3.1 COP and Realtime Timer Resolution

Table 11-4 shows the resolution and range of the Realtime timer for different Prescaler Clock frequencies.

Table 11-4 COP Timer Range and Resolution

Crystal Frequency	RP Prescaler	Resolution (preload = 0)	Range (Preload= 2^8-1)
32.00 KHz	/1	250 us	64 ms
(31.25 us)	/4	1 ms	256 ms
38.4 KHz	/1	208.3 us	53.3 ms
(26.04 us)	/4	833.3 us	213.3 ms

The realtime interrupt occurs every $2^3 \times 4^{RP} \times (DV[7:0]+1)$ Prescaler clock cycles.

The COP time-out occurs every $2^3 \times 4^{RP} \times (DV[7:0]+1) \times 8^{(CT+1)}$ Prescaler clock cycles.

11.3.2 COP/Realtime Timer Low Power Operation

The COP/Realtime module can be shut off to reduce power consumption when the module is not required by an application. To shut off this module, set the RTE bit in the COPCTL register (described in **COP and RTI Control Register (COPCTL)** on page 11-6) to 0. This gates off all clocks to the COP/Realtime Module. If either the COP or realtime function is required, the RTE bit must remain set.

It is also possible to run the Realtime or COP timer when the chip is in stop mode. When the Realtime timer reaches 0, a signal is generated that wakes up the DSP56800 core and brings it out of stop mode. Caution should be taken that the COP timer is not allowed to time out when the DSP56L811 is in stop mode. The DSP56L811 is reset whenever the COP timer times out, regardless of the operating mode it is in when COP time-out occurs. Both timers can continue to operate in wait mode.

Programming the COP and RTI Timers

Note: The Realtime timer can bring the DSP56L811 out of stop mode independently of the value of the RTIE bit in the COPCTL register (described in **COP and RTI Control Register (COPCTL)** on page 11-6) or the bits in the IPR register (described in **3.6.2 DSP56L811 Interrupt Priority Register** on page 3-19).

11.3.3 Programming Example

Example 11-1 shows how to set up the COP timer.

Example 11-1 Sending a COP Reset

```

;*****
;* COP Timer example *
;* for COP/RTI Module *
;* of DSP56L811 chip *
;*****

START      EQU    $0040    ; Start of program
BCR        EQU    $FFF9    ; Bus Control Register
COPCTL     EQU    $FFF1    ; COP/RTI Control Register
COPRST     EQU    $FFF0    ; COP Reset Register [write only]
;PCR0 [unused] EQU $FFF2    ; PLL Control Register 0
PCR1       EQU    $FFF3    ; PLL Control Register 1

;*****
;* Vector setup *
;*****

;+-----+
;| Note: Bootstrap ROM configures OMR (Operating Mode Register) to set |
;|      chip operating mode for Mode 2 (Normal Expanded Mode), then   |
;|      jumps to first location of internal program RAM (P:$0000).     |
;+-----+

      ORG    P:$0000        ; Cold Boot
      JMP    START          ; also Hardware RESET vector (Mode 0, 1, 3)

      ORG    P:$E000        ; Warm Boot
      JMP    START          ; Hardware RESET vector (Mode 2)

```

Example 11-1 Sending a COP Reset (Continued)

```

ORG    P:$E002          ;
JMP     COPOUT           ; COP Watchdog RESET vector (Mode 2)

ORG     P:START          ; Start of program
;*****
;* General setup *
;*****

    MOVEP    #$0000,X:BCR    ; External Program memory has 0 wait states.
                                ; External data memory has 0 wait states.
                                ; Port A pins tri-stated if no external access.
;*****
;* Prescaler Clock setup      *
;* (source for divider chain) *
;*****

    MOVEP    #$06C0,X:PCRL    ; Configure:
                                ; (PLLE) PLL disabled (bypassed).
                                ; -- Oscillator supplies Phi Clock.
                                ; (PLLD) PLL Power Down disabled (PLL active).
                                ; -- PLL block active for PLL to attain lock.
                                ; (LPST) Low Power Stop disabled.
                                ; (PS[2:0]) Set Prescaler Divider to /256.
                                ; (CS[1:0]) Clockout pin (CLKO) disabled.
;    [PCR0 unused]           ; Feedback Divider unused (PLL bypassed).
;*****
;* COP Timer setup *
;*****

    MOVEP    #$5555,X:COPRST ; Begin COP reset sequence
    NOP                      ; .
    MOVEP    #$AAAA,X:COPRST ; .
    NOP                      ; .
    MOVEP    #$5555,X:COPRST ; .
    NOP                      ; COPCTL write-enabled
    BFCLR    #$0800,X:COPCTL ; (RTE) Realtime Timer disabled.
                                ; -- Realtime/COP divider chain disabled, thus
                                ; -- COP timer (and Realtime timer) disabled.
    MOVEP    #$C1FF,X:COPCTL ; Configure:

```

Example 11-1 Sending a COP Reset (Continued)

```

; (CPE) COP time-out (chip reset) Enabled.
; (CT) Set COP Timer divider to /64.
; (RTIE) Realtime Timer Interrupt disabled.
; (RTIF) Realtime Timer Interrupt flag remains.
; (RP) Set Realtime/COP Prescaler to /4.
; (DV[7:0]) Set Realtime/COP Divider to /256.
BFSET #$0800,X:COPCTL ; (RTE) Realtime Timer Enabled.
; -- Realtime/COP divider chain enabled, thus
; -- COP Timer (and Realtime Timer) enabled.
MOVEP #$AAAA,X:COPRST ; Reset (disable) write mechanism for COPCTL
; and reset COP Timer.
NOP ; -- End COP reset sequence.

;*****
;* Main routine *
;*****
; ...
TEST ; Test Loop
; ...
; code to be watched for potentially wayward execution
; ...
JSR COPCLR ; Let the COP know we have not left town.
BRA TEST

COPCLR ; COP CLEAR -- COP Timer Reset Routine
;*****
;* Clear the COP Timer: code execution is still valid *
;*****
MOVEP #$5555,X:COPRST; Begin COP reset sequence.
NOP ; .
MOVEP #$AAAA,X:COPRST; .
NOP ; .
MOVEP #$5555,X:COPRST; .
NOP ; COPCTL write-enabled
MOVEP #$AAAA,X:COPRST; Reset (disable) write mechanism for COPCTL
; and reset COP Timer.
NOP ; -- End COP reset sequence.

```


Example 11-1 Sending a COP Reset (Continued)

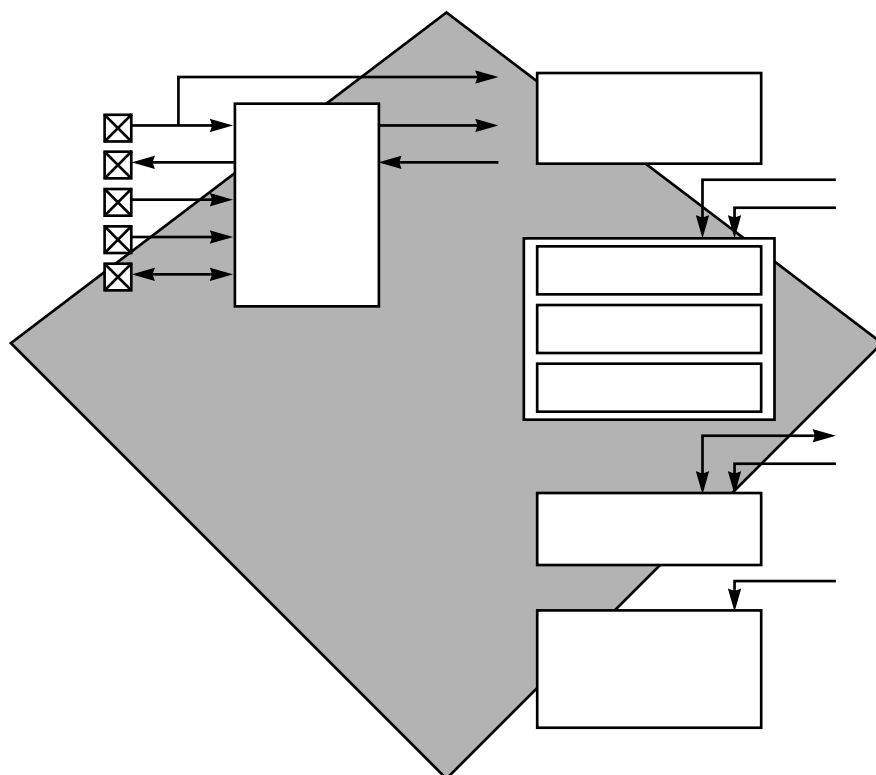
RTS

```
COPOUT ; COP time OUT -- COP Watchdog RESET Routine
        ; the COP timed out: system failure has occurred
        ; ...
        ; error recovery code
        ; ...
```



SECTION 12

JTAG PORT



12.1	INTRODUCTION	12-3
12.2	JTAG PORT ARCHITECTURE	12-4
12.3	JTAG/ONCE PORT PINOUT	12-5
12.4	JTAG REGISTERS	12-6
12.5	DSP56L811 RESTRICTIONS	12-18

12.1 INTRODUCTION

The DSP56L811 provides board and chip-level testing capability through two on-chip modules that are both accessed through the JTAG/OnCE interface. These modules are:

- On-Chip Emulation (OnCE) port
- Test access port (TAP) and 16-state controller (also called the JTAG port)

The presence of the JTAG/OnCE interface allows the user to insert the DSP chip into a target system while retaining debug control. This capability is especially important for devices without an external bus, because it eliminates the need for a costly cable to bring out the footprint of the chip, as required by a traditional emulator system.

The OnCE port is a Motorola-designed module used in DSP chips to debug application software used with the chip. The port is a separate on-chip block that allows non-intrusive interaction with the DSP and is accessible through the pins of the JTAG interface. The OnCE port makes it possible to examine registers, memory, or on-chip peripherals contents in a special debug environment. This avoids sacrificing any user accessible on-chip resources to perform debugging. See the *DSP56800 Family Manual (DSP56800FAM/AD)* for a detailed description of the OnCE port.

The JTAG port is a dedicated user-accessible TAP that is fully compatible with the *IEEE 1149.1a-1993 Standard Test Access Port and Boundary Scan Architecture*. Problems associated with testing high-density circuit boards have led to the development of this proposed standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The DSP56L811 supports circuit-board test strategies based on this standard.

Five dedicated pins interface to the TAP, which contains a 16-state controller. The TAP uses a boundary scan technique to test the interconnections between integrated circuits after they are assembled onto a printed circuit board. Boundary scan allows a tester to observe and control signal levels at each component pin through a shift register placed next to each pin. This is important for testing continuity and determining if pins are stuck at a 1 or 0 level.

The TAP port has the following capabilities:

- Perform boundary scan operations to test circuit-board electrical continuity.

JTAG Port Architecture

- Bypass the DSP for a given circuit-board test by replacing the Boundary Scan Register with a single bit register.
- Sample the DSP system pins during operation, and transparently shift out the result in the Boundary Scan Register. Pre-load values to output pins prior to invoking the EXTEST instruction.
- Disable the output drive to pins during circuit-board testing.
- Provide a means of accessing the OnCE controller and circuits to control a target system.
- Query identification information (manufacturer, part number, and version) from a DSP IC.
- Force test data onto the outputs of a DSP IC(s) while replacing its Boundary Scan Register in the serial data path with a single bit register.
- Enable a weak pull-up current device on all input signals of a DSP IC(s); this helps to ensure deterministic test results in the presence of a continuity fault during interconnect testing.

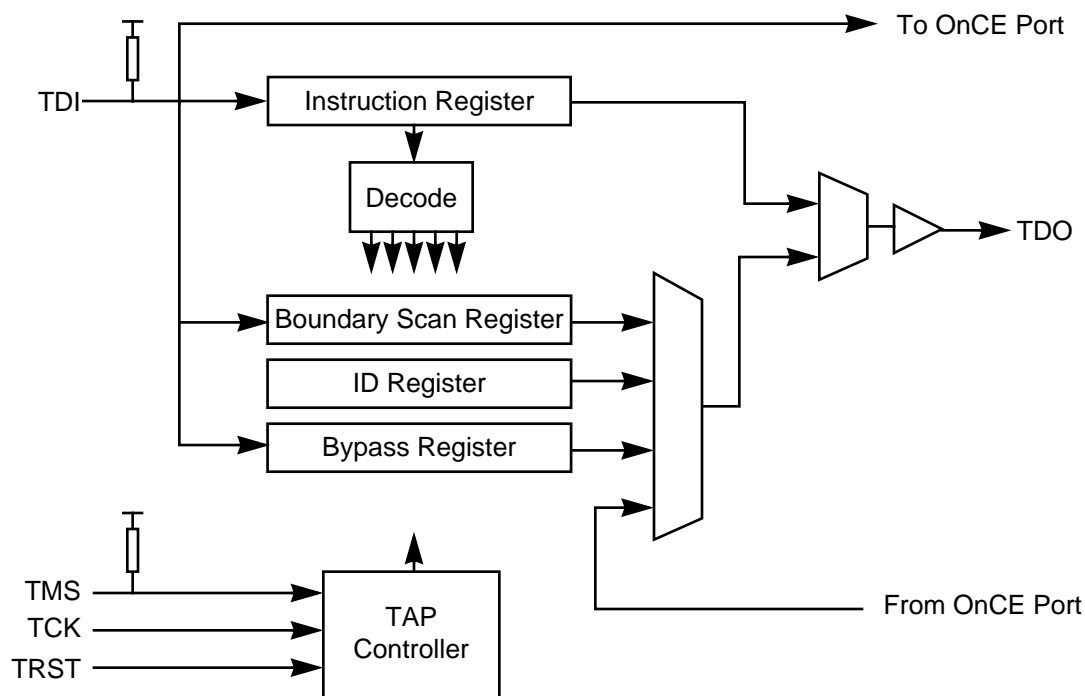
This section includes aspects of the JTAG implementation that are specific to the DSP56L811. It is intended to be used with the supporting IEEE 1149.1a-1993 document. The discussion includes those items required by the standard to be defined and, in certain cases, provides additional information specific to the DSP56L811. For internal details and applications of the standard, refer to the IEEE 1149.1a-1993 document.

12.2 JTAG PORT ARCHITECTURE

The TAP controller is a simple state machine used to sequence the JTAG port through its valid operations:

- Serially shift in or out a JTAG command
- Update (and decode) the JTAG Instruction Register
- Serially input or output a data value
- Update a JTAG (or OnCE) register

Note: The JTAG port oversees the shifting of data into and out of the OnCE port through the TDI and TDO pins, respectively. In this case, the shifting is guided by the same TAP controller used when shifting JTAG information.



AA0119

Figure 12-1 JTAG Block Diagram

A block diagram of the JTAG port is shown in **Figure 12-1**. The JTAG port has three read/write registers: the Instruction Register, the Boundary Scan Register, and the Bypass Register. There is also one read-only register, the ID Register.

The TAP controller provides access to the JTAG Instruction Register through the JTAG port. The other JTAG registers must be individually selected by the JTAG Instruction Register. The blocks and registers within the JTAG port are explained in the following paragraphs, and its corresponding programming model is shown in **Figure 12-2**.

12.3 JTAG/ONCE PORT PINOUT

As described in the IEEE 1149.1a-1993 specification, the JTAG port requires a minimum of four pins to support TDI, TDO, TCK, and TMS signals. The DSP56L811 also uses the optional $\overline{\text{TRST}}$ input signal and multiplexes it so that the same pin can support the debug event ($\overline{\text{DE}}$) output signal used by the OnCE interface. The pin functions are described in **Table 12-1**.

Table 12-1 JTAG Pin Descriptions

Pin Name	Pin Description
TDI	Test Data Input—This input pin provides a serial input data stream to the JTAG and the OnCE port. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
TDO	Test Data Output—This tri-state-able output pin provides a serial output data stream from the JTAG and the OnCE port. It is driven in the Shift-IR and Shift-DR controller states of the JTAG state machine, and changes on the falling edge of TCK.
TCK	Test Clock Input—This input pin provides a gated clock to synchronize the test logic and shift serial data to and from the JTAG/OnCE port. The maximum frequency for TCK is $\frac{1}{8}$ the maximum frequency specified for the DSP56L811 core (i.e., 5 MHz for TCK if the maximum CLK input is 40 MHz) The TCK pin has an on-chip pull-down resistor.
TMS	Test Mode Select Input—This input pin is used to sequence the JTAG TAP controller's state machine. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.
$\overline{\text{TRST}}/\overline{\text{DE}}$	Test Reset/Debug Event—This bidirectional pin, when configured as an input, provides a reset signal to the JTAG TAP controller. When configured as an output, it signals debug events detected on a trigger condition. The operational mode of the pin is configured by bit 14 of the OnCE Control Register (OCR). The $\overline{\text{TRST}}/\overline{\text{DE}}$ pin has an on-chip pull-down resistor.

12.4 JTAG REGISTERS

Figure 12-2 shows the programming models for the JTAG registers on the DSP56L811.

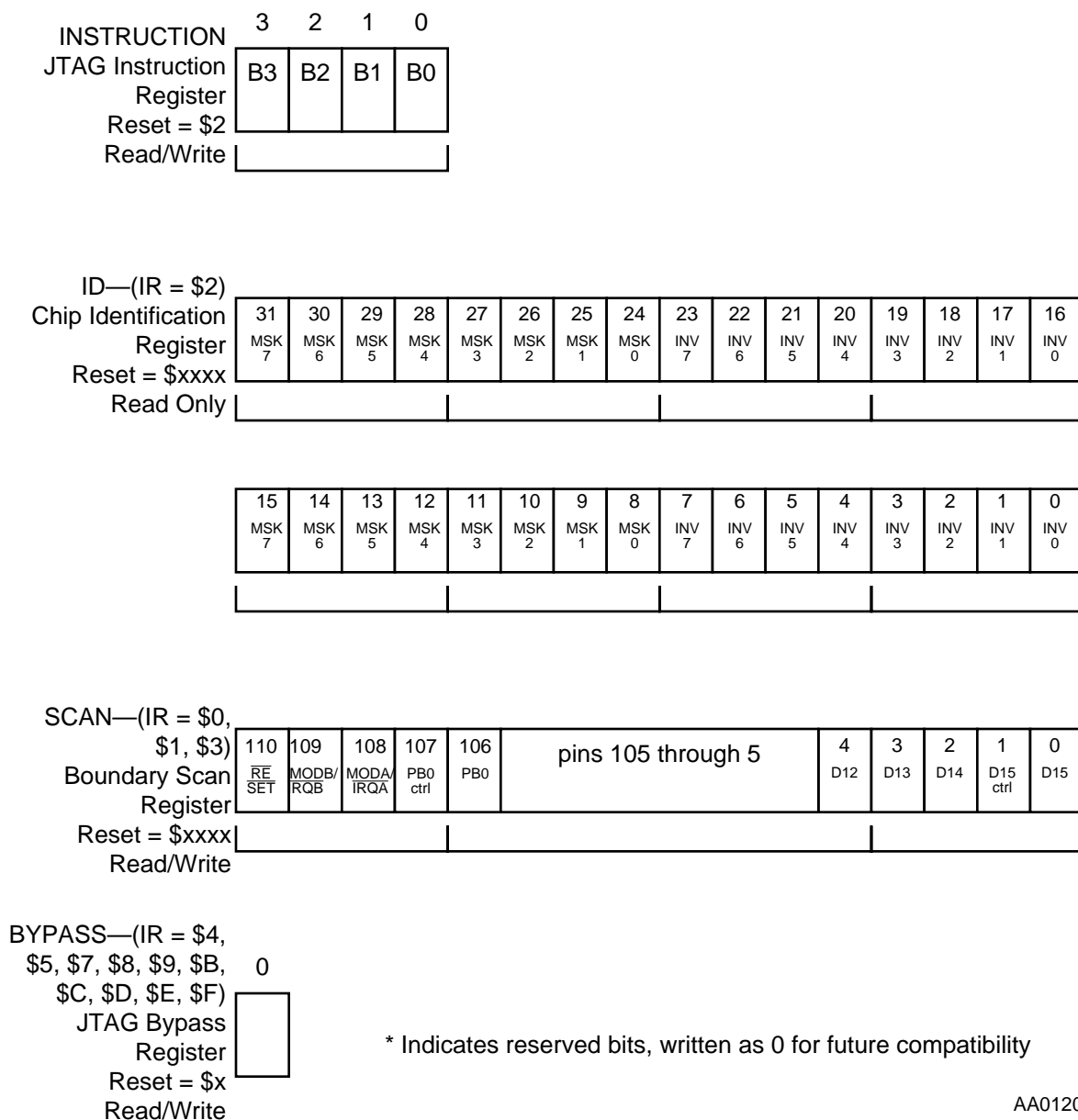


Figure 12-2 JTAG Port Programming Model

12.4.1 JTAG Instruction Register and Decoder

The TAP controller contains a 4-bit instruction register. The instruction is presented to an instruction decoder during the Update-IR state. See **TAP Controller** on page 12-17 for a description of the TAP controller operating states. The instruction decoder interprets and executes the instructions according to the conditions defined by the TAP controller state machine. The DSP56L811 includes the three mandatory public instructions (BYPASS, SAMPLE/PRELOAD, and EXTEST) and six public instructions (CLAMP, HIGHZ, EXTEST_PULLUP, IDCODE, ENABLE_OnCE, and DEBUG_REQUEST).

The four bits B[3:0] decode the 16 instructions as shown in **Table 12-2**.

Table 12-2 JTAG Instruction Register Encodings

B[3:0]	Instruction
0000	EXTEST
0001	SAMPLE/PRELOAD
0010	IDCODE
0011	EXTEST_PULLUP
0100	HIGHZ
0101	CLAMP
0110	ENABLE_OnCE
0111	DEBUG_REQUEST
1111	BYPASS

All other encodings are reserved for future enhancements and are decoded as BYPASS (1111). The JTAG Instruction Register is reset to 0010 in the Test-Logic-Reset controller state. Therefore, the IDCODE instruction is selected on JTAG reset. In the Capture-IR state, the two LSBs of the instruction shift register are preset to 01, where the 1 is in the LSB location as required by the standard. The two most significant bits may either capture status or be set to 0. New instructions are shifted into the instruction shift register stage on Shift-IR state.

12.4.1.1 EXTEST (B[3:0]=0000)

The external test (EXTEST) instruction enables the Boundary Scan Register between TDI and TDO, including cells for all digital device signals and associated control signals. The EXTAL pin and any codec pins associated with analog signals are not included in the Boundary Scan Register path.

In EXTEST, the Boundary Scan Register is capable of scanning user-defined values onto output pins, capturing values presented to input signals, and controlling the direction and value of bi-directional pins. The EXTEST instruction asserts internal system reset for the DSP system logic for the duration of EXTEST in order to force a predictable internal state while performing external boundary scan operations.

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

The SAMPLE/PRELOAD instruction enables the Boundary Scan Register between TDI and TDO. When this instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic, or on the flow of a signal between the system pin and the on-chip system logic, as specified by the IEEE 1149.1-1993a specification. This instruction provides two separate functions. First, it provides a means to obtain a snapshot of system data and control signals (SAMPLE). The snapshot occurs on the rising edge of TCK in the Capture-DR controller state. The data can be observed by shifting it transparently through the Boundary Scan Register. In a normal system configuration many signals require external pull-ups to ensure proper system operation. Consequently, the same is true for the SAMPLE/PRELOAD functionality. The data latched into the Boundary Scan Register during the Capture-DR controller state may not match the drive state of the package signal if the system-required pull-ups are not present within the test environment.

The second function of the SAMPLE/PRELOAD instruction is to initialize the Boundary Scan Register output cells (PRELOAD) prior to selection of the CLAMP, EXTEST, or EXTEST_PULLUP instruction. This initialization ensures that known data appears on the outputs when executing the EXTEST instruction. The data held in the shift register stage is transferred to the output latch on the falling edge of TCK in the Update-DR controller state. Data is not presented to the pins until the CLAMP, EXTEST, or EXTEST_PULLUP instruction is executed.

Note: Since there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results when sampling system values when using the SAMPLE/PRELOAD instruction.

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

The IDCODE instruction enables the IDREGISTER between TDI and TDO. It is provided as a public instruction to allow the manufacturer, part number, and version of a component to be determined through the TAP.

When the IDCODE instruction is decoded, it selects the IDREGISTER, a 32-bit test data register. IDREGISTER loads a constant logic 1 into its LSB. Since the Bypass Register loads a logic 0 at the start of a scan cycle, examination of the first bit of data shifted out of a component during a test data scan sequence immediately following

exit from the Test-Logic-Reset controller state shows whether an IDREGISTER is included in the design.

When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic, as required in the IEEE 1149.1a-1993 specification.

12.4.1.4 EXTEST_PULLUP (B[3:0]=0011)

The EXTEST_PULLUP instruction is provided as a public instruction to aid in fault diagnoses during boundary-scan testing of a circuit board. This instruction functions similarly to EXTEST, with the only difference being the presence of a weak pull-up device on all input signals. Given an appropriate charging delay, the pull-up current supplies a deterministic logic 1 result on an open input. When this instruction is used in board-level testing with heavily loaded nodes, it may require a charging delay greater than the two TCK periods needed to transition from the Update-DR state to the Capture-DR state. Two methods of providing an increase delay are available: traverse into the Run-Test/Idle state for extra TCK periods of charging delay, or limit the maximum TCK frequency (slow down the TCK) so that two TCK periods are adequate. The EXTEST_PULLUP instruction asserts internal system reset for the DSP system logic for the duration of EXTEST_PULLUP in order to force a predictable internal state while performing external boundary scan operations.

12.4.1.5 HIGHZ (B[3:0]=0100)

The HIGHZ instruction enables the single-bit Bypass Register between TDI and TDO. It is provided as a public instruction in order to prevent having to drive the output signals back during circuit board testing. When the HIGHZ instruction is invoked, all output drivers are placed in an inactive-drive state. HIGHZ asserts internal system reset for the DSP system logic for the duration of HIGHZ in order to force a predictable internal state while performing external boundary-scan operations.

12.4.1.6 CLAMP (B[3:0]=0101)

The CLAMP instruction enables the single-bit Bypass Register between TDI and TDO. It is provided as a public instruction. When the CLAMP instruction is invoked, the package output signals respond to the preconditioned values within the update latches of the Boundary Scan Register, even though the Bypass Register is enabled as the test data register. In-circuit testing can be facilitated by setting up guarding signal conditions that control the operation of logic not involved in the test with use of the SAMPLE/PRELOAD or EXTEST instructions. When the CLAMP instruction is executed, the state and drive of all signals remain static until a new instruction is invoked. A feature of the CLAMP instruction is that while the signals continue to supply the guarding inputs to the in-circuit test site, the bypass mode is enabled, thus

minimizing overall test time. Data in the boundary scan cell remains unchanged until a new instruction is shifted in or the JTAG state machine is set to its reset state.

CLAMP asserts internal system reset for the DSP system logic for the duration of CLAMP in order to force a predictable internal state while performing external boundary-scan operations.

12.4.1.7 ENABLE_ONCE (B[3:0]=0110)

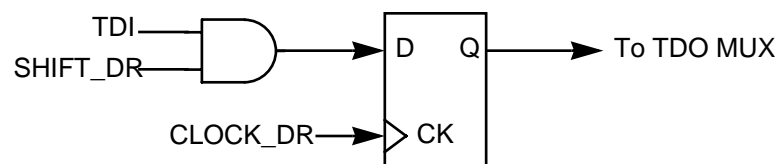
The ENABLE_ONCE instruction enables the JTAG port to communicate with the OnCE state machine and registers. It is provided as a Motorola public instruction to allow the user to perform system debug functions. When the ENABLE_ONCE instruction is invoked, the TDI and TDO pins are connected directly to the OnCE registers. The particular OnCE register connected between TDI and TDO is selected by the OnCE state machine and the OnCE instruction being executed. All communication with the OnCE instruction controller is done through the SELECT-DR-SCAN path of the JTAG state machine. Refer to the *DSP56800 Family Manual (DSP56800FAM/AD)* for more information.

12.4.1.8 DEBUG_REQUEST (B[3:0]=0111)

The DEBUG_REQUEST instruction asserts a request to halt the core for entry to debug mode. It is typically used in conjunction with ENABLE_ONCE to perform system debug functions. It is provided as a Motorola public instruction. When the DEBUG_REQUEST instruction is invoked, the TDI and TDO pins are connected to the Bypass Register. Refer to the *DSP56800 Family Manual (DSP56800FAM/AD)* for more information.

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

The BYPASS instruction enables the single-bit Bypass Register between TDI and TDO, as shown in **Figure 12-3**. This creates a shift register path from TDI to the Bypass Register and finally to the TDO signal, circumventing the Boundary Scan Register. This instruction is used to enhance test efficiency by shortening the overall path between TDI and TDO when no test operation of a component is required. In this instruction, the DSP system logic is independent of the test access port. When this instruction is selected, the test logic has no effect on the operation of the on-chip system logic, as required in the IEEE 1149.1-1993a specification.

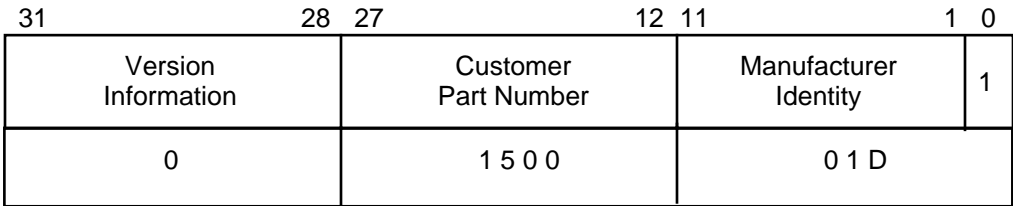


AA0122

Figure 12-3 Bypass Register

12.4.2 JTAG Chip Identification Register

The chip identification register (CID) is a 32-bit register that provides a unique JTAG ID for the DSP56L811. It is provided as a public instruction to allow the manufacturer, part number, and version of a component to be determined through the TAP. **Figure 12-4** shows the CID register configuration.



AA0121

Figure 12-4 Chip Identification Register Configuration

For the initial release of the DSP56L811, the Device ID Number is \$0150001D. The version code is \$0. Future revisions increment this version code. The value of the customer part number code is \$1500.

Motorola’s Manufacturer Identity is 00000001110. The Customer Part Number consists of two parts: Motorola Design Center Number (bits 27:22) and Design Center Assigned Sequence Number (bits 21:12). DSP’s Design Center Number is 000101. Version information and Design Center Assigned Sequence Number values vary depending on the current revision and implementation of a specific chip.

The bit assignment for the ID code is given in **Table 12-3**.

Table 12-3 Device ID Register Bit Assignment

Bit Number	Code Use	Value for DSP56L811
31-28	Version Number	0000 (for initial version only—these bits may vary)
27-22	DSP Identification	00 0101
21-12	Family and part ID	01 0000 0000
11-0	Motorola Manufacturer ID	0000 0001 1101

12.4.3 JTAG Boundary-Scan Register

The Boundary Scan Register is used to examine or control the “scan-able” pins on the DSP56L811. The Boundary Scan Register for the DSP56L811 contains 110 bits. One register bit is present for each scan-able pin on a chip.

Table 12-4 gives the contents of the Boundary Scan Register for the DSP56L811.

Table 12-4 Boundary-Scan Register Contents for DSP56L811

Bit Number	Pin/Bit Name	Pin Type	BSR Cell Type	Pin Number (TQFP Package)
0	D15	input/output	BC_6	45
1	D15 control	control	BC_2	n/a
2	D14	input/output	BC_6	44
3	D13	input/output	BC_6	41
4	D12	input/output	BC_6	40
5	D11	input/output	BC_6	39
6	D10	input/output	BC_6	38
7	D9	input/output	BC_6	37
8	D8	input/output	BC_6	36
9	D7	input/output	BC_6	35
10	D6	input/output	BC_6	34
11	D5	input/output	BC_6	33
12	D4	input/output	BC_6	30
13	D3	input/output	BC_6	29
14	D2	input/output	BC_6	28
15	D1	input/output	BC_6	27
16	D0	input/output	BC_6	26
17	A0	output	BC_2	25
18	A1	output	BC_2	24
19	A2	output	BC_2	23
20	A3	output	BC_2	22
21	A4	output	BC_2	21
22	\overline{DS}	output	BC_2	18
23	\overline{DS} control	control	BC_2	n/a

Table 12-4 Boundary-Scan Register Contents for DSP56L811 (Continued)

Bit Number	Pin/Bit Name	Pin Type	BSR Cell Type	Pin Number (TQFP Package)
24	PS	output	BC_2	17
25	\overline{PS} control	control	BC_2	n/a
26	A5	output	BC_2	14
27	A6	output	BC_2	13
28	A7	output	BC_2	12
29	A8	output	BC_2	11
30	A9	output	BC_2	8
31	A10	output	BC_2	7
32	A11	output	BC_2	6
33	A12	output	BC_2	5
34	A13	output	BC_2	4
35	A14	output	BC_2	3
36	A15	output	BC_2	2
37	A15 control	control	BC_2	n/a
38	\overline{RD}	output	BC_2	1
39	\overline{RD} control	control	BC_2	n/a
40	\overline{WR}	output	BC_2	100
41	\overline{WR} control	control	BC_2	n/a
42	PC15/TIO2	input/output	BC_6	99
43	PC15/TIO2 control	control	BC_2	n/a
44	PC14/TIO01	input/output	BC_6	98
45	PC14/TIO01 control	control	BC_2	n/a
46	PC13/SRFS	input/output	BC_6	97
47	PC13/SRFS control	control	BC_2	n/a
48	PC12/SRCK	input/output	BC_6	96
49	PC12/SRCK control	control	BC_2	n/a
50	PC11/STFS	input/output	BC_6	95
51	PC11/STFS control	control	BC_2	n/a
52	PC10/STCK	input/output	BC_6	94
53	PC10/STCK control	control	BC_2	n/a
54	PC9/SRD	input/output	BC_6	93
55	PC9/SRD control	control	BC_2	n/a

Table 12-4 Boundary-Scan Register Contents for DSP56L811 (Continued)

Bit Number	Pin/Bit Name	Pin Type	BSR Cell Type	Pin Number (TQFP Package)
56	PC8/STD	input/output	BC_6	92
57	PC8/STD control	control	BC_2	n/a
58	PC7/ $\overline{SS1}$	input/output	BC_6	91
59	PC7/ $\overline{SS1}$ control	control	BC_2	n/a
60	PC6/SCK1	input/output	BC_6	88
61	PC6/SCK1 control	control	BC_2	n/a
62	PC5/MOSI1	input/output	BC_6	87
63	PC5/MOSI1 control	control	BC_2	n/a
64	PC4/MISO1	input/output	BC_6	86
65	PC4/MISO1 control	control	BC_2	n/a
66	PC3/ $\overline{SS0}$	input/output	BC_6	85
67	PC3/ $\overline{SS0}$ control	control	BC_2	n/a
68	PC2/SCK0	input/output	BC_6	84
69	PC2/SCK0 control	control	BC_2	n/a
70	PC1/MOSI0	input/output	BC_6	83
71	PC1/MOSI0 control	control	BC_2	n/a
72	PC0/MISO0	input/output	BC_6	82
73	PC0/MISO0 control	control	BC_2	n/a
74	EXTAL	input	BC_4	77
75	CLKO	output	BC_2	74
76	PB15	input/output	BC_6	73
77	PB15 control	control	BC_2	n/a
78	PB14	input/output	BC_6	72
79	PB14 control	control	BC_2	n/a
80	PB13	input/output	BC_6	71
81	PB13 control	control	BC_2	n/a
82	PB12	input/output	BC_6	70
83	PB12 control	control	BC_2	n/a
84	PB11	input/output	BC_6	67
85	PB11 control	control	BC_2	n/a
86	PB10	input/output	BC_6	66
87	PB10 control	control	BC_2	n/a

Table 12-4 Boundary-Scan Register Contents for DSP56L811 (Continued)

Bit Number	Pin/Bit Name	Pin Type	BSR Cell Type	Pin Number (TQFP Package)
88	PB9	input/output	BC_6	65
89	PB9 control	control	BC_2	n/a
90	PB8	input/output	BC_6	64
91	PB8 control	control	BC_2	n/a
92	PB7	input/output	BC_6	63
93	PB7 control	control	BC_2	n/a
94	PB6	input/output	BC_6	62
95	PB6 control	control	BC_2	n/a
96	PB5	input/output	BC_6	61
97	PB5 control	control	BC_2	n/a
98	PB4	input/output	BC_6	58
99	PB4 control	control	BC_2	n/a
100	PB3	input/output	BC_6	57
101	PB3 control	control	BC_2	n/a
102	PB2	input/output	BC_6	56
103	PB2 control	control	BC_2	n/a
104	PB1	input/output	BC_6	55
105	PB1 control	control	BC_2	n/a
106	PB0	input/output	BC_6	54
107	PB0 control	control	BC_2	n/a
108	MODA/ $\overline{\text{IRQA}}$	input	BC_4	53
109	MODB/ $\overline{\text{IRQB}}$	input	BC_4	52
110	RESET	input	BC_2	51

12.4.4 JTAG Bypass Register

The JTAG Bypass Register is a 1-bit register used to provide a simple, direct path from the TDI pin to the TDO pin. This is useful in boundary scan applications where many chips are serially connected together in a daisy-chain. Individual DSPs or other devices can be programmed with the BYPASS instruction so that individually they become pass-through devices during testing. This allows testing of a specific chip, while still having all of the chips connected through the JTAG ports.

12.4.5 TAP Controller

The TAP controller is a synchronous finite state machine that contains 16 states as illustrated in **Figure 12-5**. The TAP controller responds to changes at the TMS and TCK signals. Transitions from one state to another occur on the rising edge of TCK. The value shown adjacent to each state transition in this figure represents the signal present at TMS at the time of a rising edge at TCK.

The TDO pin remains in the high impedance state except during the Shift-DR or Shift-IR controller states. In these controller states, TDO is updated on the falling edge of TCK. TDI is sampled on the rising edge of TCK.

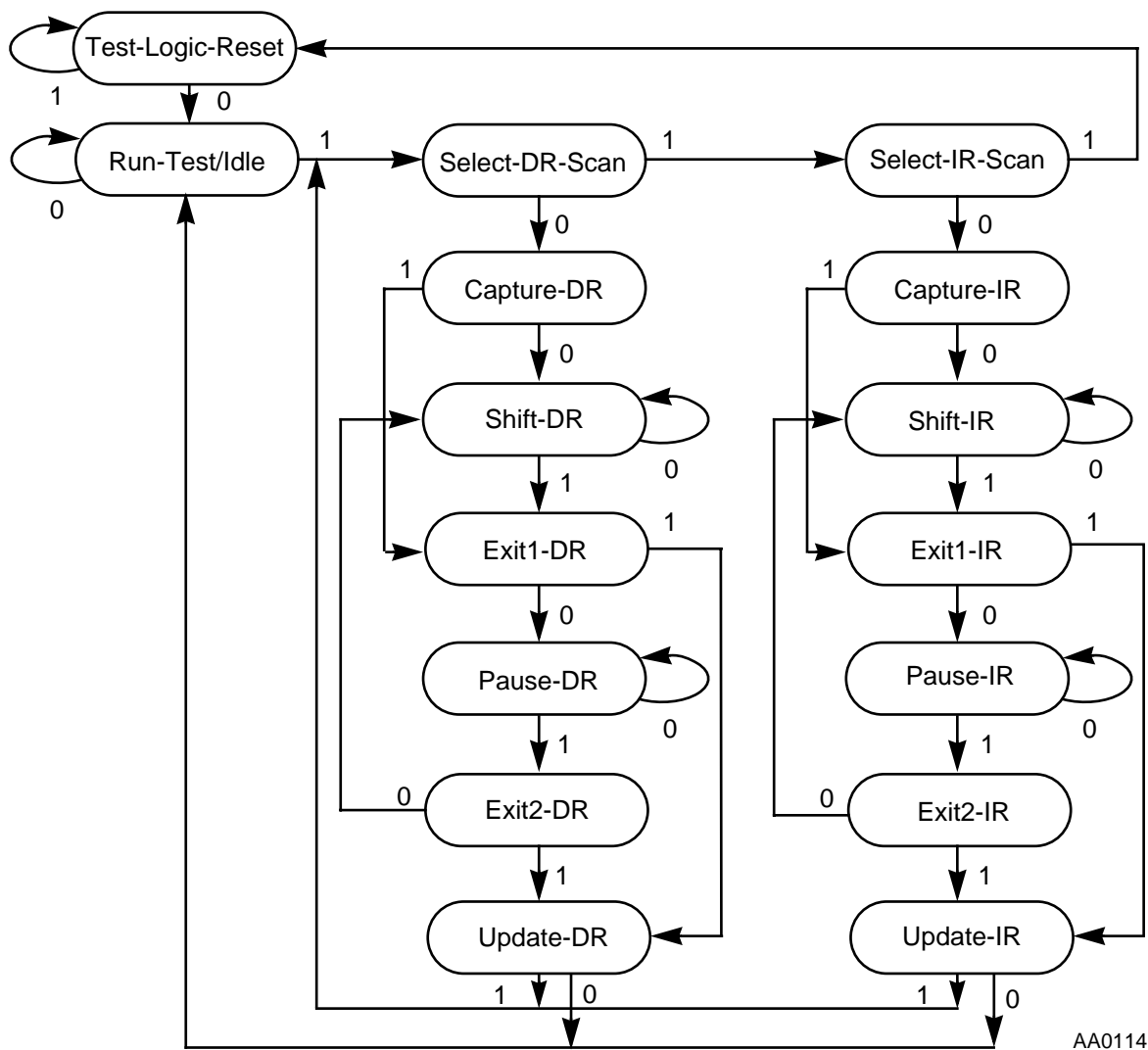


Figure 12-5 TAP Controller State Diagram

DSP56L811 Restrictions

There are two paths through the 16-state machine. The SHIFT-IR_SCAN path captures and loads JTAG instructions into the JTAG Instruction Register. The SHIFT-DR_SCAN path captures and loads data into the other JTAG registers. The TAP controller executes the last instruction decoded until a new instruction is entered at the Update-IR state, or until the Test-Logic-Reset state is entered.

When using the JTAG port to access OnCE port registers, accesses are first enabled by shifting the ENABLE_ONCE instruction into the JTAG Instruction Register. After this is selected, the OnCE port registers and commands are read and written through the JTAG pins using the SHIFT-DR_SCAN path. Asserting the JTAG's $\overline{\text{TRST}}$ pin asynchronously forces the JTAG state machine into the Test-Logic-Reset state.

12.5 DSP56L811 RESTRICTIONS

The control afforded by the output enable signals using the Boundary Scan Register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the DSP56L811 output drivers are enabled into actively driven networks.

Two constraints apply to the JTAG interface. First, while the TCK input includes an internal pull-down resistor, it should not be left unconnected. The second constraint is to ensure that the JTAG test logic is kept transparent to the system logic by forcing TAP into the test-logic-reset controller state, using either of two methods. During power-up, the $\overline{\text{TRST}}$ pin must be externally asserted to force the TAP controller into this state. After power-up is concluded, TMS must be sampled as a logic 1 for five consecutive TCK rising edges. If TMS either remains unconnected or is connected to V_{DD} , then the TAP controller cannot leave the test-logic-reset state, regardless of the state of TCK.

The DSP56L811 features a low-power stop mode that is invoked using an instruction called STOP. The interaction of the JTAG interface with low-power stop mode is as follows:

1. The TAP controller must be in the Test-Logic-Reset state to either enter or remain in the low-power stop mode. Leaving the TAP controller Test-Logic-Reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
2. The TCK input is not blocked in low-power stop mode. To consume minimal power, the TCK input should be externally connected to V_{DD} or ground.

3. The TMS and TDI pins include on-chip pull-up resistors. In low-power stop mode, these two pins should remain either unconnected or connected to V_{DD} to achieve minimal power consumption.

Since all DSP56L811 clocks are disabled during stop state, the JTAG interface provides the means of polling the device status (sampled in the capture-IR state).



A.1 INTRODUCTION A-3

A.2 BOOTSTRAP LISTING A-4

A.1 INTRODUCTION

This section presents the bootstrap program contained in the DSP56L811 Boot ROM. This program can load the internal program RAM starting at P:\$0 from an external EPROM or the external memory interface (Port A), and can load any program RAM segment from the Serial Peripheral Interface 0 (SPI0) or from the Synchronous Serial Interface (SSI).

- If bits MB and MA in the OMR register are both set to 0 (MB:MA = 00), the bootstrap program loads program RAM from 2048 consecutive byte-wide P: memory locations (typically EPROM), starting at P:\$C000 (lower byte of data bus), through the external memory interface.
- If bits MB and MA in the OMR register are set to 01, the bootstrap program loads program RAM from 13,824 consecutive byte-wide P: memory locations, starting at P:\$8000 (bits 7-0), through SPI0 or the SSI.
- If bit 15 of P:\$C000 is set to 0, program RAM is loaded through SPI0. If this bit is set to 1, program RAM is loaded through the SSI instead.

These instructions are stored in contiguous program RAM memory locations starting at P:\$0.

Note: This routine loads data starting with the least significant byte of P:\$0.

A.2 BOOTSTRAP LISTING

The bootstrap program listing is shown in **Example A-1**.

Example A-1 DSP56L811 Bootstrap Code

```
*****
; * boot_56811.asm
; *
; * Bootstrap source code for the Motorola DSP56L811.
; * (C) Copyright 1995 Motorola Inc.
; *
; * This is source code for the Bootstrap program contained in the DSP56L811.
; * This program can load the internal program memory from one of 3 external
; * sources. The program reads the OMR bits MA and MB typically out of reset
; * to decide which external source to access. The sources include:
; *
; * - Bootstrapping from external byte-wide memory (lower byte of data bus)
; * - Bootstrapping through the SPI serial port (SPI0)
; * - Bootstrapping through the SSI serial port
; *
; * The decision for which port is selected is shown here:
; *   if (MB:MA == 00)
; *       load from 2048 consecutive byte-wide P: memory locations
; *       (typically EPROM), starting at P:$C000 (lower byte of data bus)
; *   if (MB:MA = 01)
; *       if (bit 15 of P:$C000 == 0)
; *           load internal PRAM through the SPI (SPI0)
; *       if (bit 15 of P:$C000 == 1)
; *           load internal PRAM through the SSI
; *
; * Loading PRAM from external byte-wide memory:
; *   The bootrom code will load 2048 bytes from the external P memory
; *   space beginning at p:$c000 (bits 7-0). These will be condensed
; *   into 1024 16-bit words and stored in contiguous internal pram memory
; *   locations starting at p:$0. Note that the first routine loads data
; *   starting with the least significant byte of p:$0 first, so that the
```

Example A-1 DSP56L811 Bootstrap Code (Continued)

```

; *   data storage in external program memory is as follows:
; *
; *       P:$C000       lower byte of word to be stored at P:$0000
; *       P:$C001       upper byte of word to be stored at P:$0000
; *       P:$C002       lower byte of word to be stored at P:$0001
; *       P:$C003       upper byte of word to be stored at P:$0001
; *       .               .
; *       .               .
; *       .               .
; *
; * Loading PRAM from the SPI or SSI ports
; *   The bootrom code loads the internal pram from the SPI port or the SSI.
; *   It will load from SPI if bit 15 of p:$C000 is cleared and from the
; *   serial synchronuous interface SSI if bit 15 of p:$C000 is set.
; *   These will be condensed into 1024 16-bit words and stored in
; *   contiguous internal pram memory locations starting at p:$0. Note that
; *   when using the SPI or SSI, the routine loads data starting with the
; *   least significant byte of p:$0 first.  For the SPI or SSI, the maximum
; *   frequency allowed on the serial clock pin is 1/8 of the Phi Clock
; *   frequency used on the DSP core.
; *
; * NOTE:
; *   If this program is entered by changing the OMR in software to bootstrap
; *   mode, make certain that the M01 register has been set to $FFFF (linear
; *   addressing).  In addition, make sure the BCR register is set to $xxxF
; *   to support slow external EPROMs for the bootstrapping operation.
; *
; * Registers used by the program:
; *   R0  - pointer used to write 16-bit instructions to on-chip PRAM
; *   R1  - pointer used to read 8-bit data from off-chip P-memory when
; *         bootstrapping through the External Bus
; *   R2  - loop counter for main outer loop, counting #pram locations filled
; *   Y0  - contents of P:$C000, used to save MSB which selects SPI vs SSI
; *   A   - register used to combine two 8-bit words into a 16-bit value
; *   B   - register which holds an 8-bit sample from one of the 3 ports
; *   OMR - read to determine whether bootstrap occurs in Mode 0 or 1,
; *         and also used to change the chip's execution to on-chip PRAM

```

Example A-1 DSP56L811 Bootstrap Code (Continued)

```
; *          at the very end of the bootstrapping operation
; *****

        page 132,67
        opt  cc,now,mu

PRAMSIZ      EQU      1024 ; On-chip program RAM size is 1024

BOOT         EQU      $C000 ; The location in P: memory where the external
                           ; byte-wide EPROM is located when loading from
                           ; Port A. Also, the MSB of the value read at
                           ; this location is used to select SPI vs SSI
                           ; when bootstrapping in Mode 1.

PCC          EQU      $FFED ; Port C Control Register

SPCR0        EQU      $FFE2 ; SPI0 Control Register
SPSR0        EQU      $FFE1 ; SPI0 Status Register
SPDR0        EQU      $FFE0 ; SPI0 Data Register

SCRRX        EQU      $FFD4 ; SSI Receive Control Reg
SCRTX        EQU      $FFD3 ; SSI Transmit Control Reg
SCR2         EQU      $FFD2 ; SSI Control Reg 2
SSRTSR       EQU      $FFD1 ; SSI Status/Transmit Slot Reg
STXRX        EQU      $FFD0 ; SSI TX/RX Reg

; ***
; *** Initial Entry into the Bootstrap Program:
; ***      - Initialize R1 to address of external byte-wide ROM ($C000)
; ***      - Initialize R0 to address of 1st location in on-chip PRAM ($0000)
; ***      - read the contents of P:$C000 (external memory)
; ***      - if (LSB of OMR register is 0) goto main loop
; ***      - else initialize SPI or SSI port
; ***
```

Example A-1 DSP56L811 Bootstrap Code (Continued)

```

ORG    PL:$0                ; BOOTROM code starts at P:$0

MOVE   #BOOT,R1             ; Setup R1 as P: address of
                             ;     ext byte-wide ROM

CLR     R0                   ; Setup R0 = 0 as pointer to first
                             ;     location of on-chip PRAM for instrs

MOVE    P:(R1)+,Y0           ; Get P:$C000
LEA     (R1)-                ; Adjust pointer, because in the case
                             ; where bootstrapping from external
                             ; P-memory, P:$C000 will be read again

BRCLR   #$0001,OMR,_TO_LOOP; if (OMR's MODA bit == 0)
                             ;     goto _TO_LOOP
BRCLR   #$8000,Y0,_INIT_SSI; else if (P:$C000's MSB == 0)
                             ;     goto _INIT_SSI
                             ; else
                             ;     initialize SSI

_INIT_SSI
MOVE    R0,X:SCRRX           ; Set up SSI's SCRRX Control Register:
                             ;     - (R0 = 0 at this point)
                             ;     - set RX's prescaler to /1
                             ;     - RX set for 8 bits per word
                             ;     - RX set for 1 word per frame
                             ;     - set RX's PM divider to /1
MOVE    #$2010,X:SCR2        ; Set up SSI's SCR2 Control Register:
                             ;     - interrupts disabled
                             ;     - receive section enabled
                             ;     - receive buffer disabled
                             ;     - external clock, frame sync
                             ;     - MSB first
                             ;     - SSI enabled
                             ;     - normal mode
                             ;     - bit long frame sync

```

Example A-1 DSP56L811 Bootstrap Code (Continued)

```
BFSET $$3200,X:PCC      ; Set PC9,PC12,PC13 to SRD,SRCK,SRFS
BRA    _TO_LOOP
```

_INIT_SPI

```
BFSET $$0004,X:SPCR0    ; Set up SPIO's SPCR Control Register:
                        ;   - interrupts disabled
                        ;   - Wired-OR mode disabled
                        ;   - slave mode
                        ;   - CPL set to 0
                        ;   - CPH set to 1
                        ;   - SPR2-0 not used since slave mode
BFSET $$0040,X:SPCR0    ; Enable SPI via setting SPE bit
BFSET $$000F,X:PCC      ; Set PC0-PC3 to MISO0,MOSI0,SCK0,SS0
;BRA    _TO_LOOP        ; (falls through into main loop)
```

```
; ***
; *** Main Loop:
; ***   - branch to one of three code segments
; ***   - enter hardware do loop (executes two times)
; ***       - get an 8-bit sample from the appropriate code segment
; ***       - put in accumulator and shift 8 bits
; ***       - go back to top of inner loop one more time
; ***   - when loop completed twice, store the 16-bit word into program RAM
; ***   - go back to top of Main loop until all samples loaded into memory
; ***
```

_TO_LOOP

```
MOVE    #PRAMSIZE-1,R2; Load PRAM size into N
CLR     A              ; Initialize accumulator for simulation
```

Example A-1 DSP56L811 Bootstrap Code (Continued)

```
; ***** Beginning of Main Loop *****

_MAIN_LP

; At the top of each loop, the code determines what type of bootstrap
; is being performed - External Bus, SPI, or SSI. This code below
; selects and branches to the correct code segment which then gets
; 16 bits of data from one of these three ports.

        BRCLR #$0001,OMR,_MEMLD; if (OMR's MODA bit == 0)

                                ;    goto _MEMLD

        BRCLR #$8000,Y0,_SPILD; else if (Y0's MSB == 0)

                                ;    goto _SPILD

                                ; else
                                ;    goto _SSILD

; *** Code for getting two bytes from the SSI

_SSILD  DO #2,_LOOP2

_SSIWT                                ; Polling loop
        BRCLR #$0080,X:MSRTSR,_SSIWT;    - Wait for RDF flag to go high

        MOVE  X:STXR,X,B              ; Put SSI's byte into upper byte of B1
        REP   #8
        ASR   B
        BRA   <_PACK                  ; Branch to shared "packing" code

; *** Code for getting two bytes from the SPI

_SPILD  DO #2,_LOOP2
```

Example A-1 DSP56L811 Bootstrap Code (Continued)

```
_SPIWT                                ; Polling loop
    BRCLR #0080,X:SPSR0,_SPIWT;      - Wait for SPIF flag to go high

    MOVE  X:SPSR0,B                    ; Read of SPSR0 used to clear SPIF flag
    MOVE  X:SPDR0,B                    ; Put SPI's byte into lower byte of B1
    BRA   <_PACK                       ; Branch to shared "packing" code

; *** Code for getting two bytes from the External Bus

_MEMLD
    DO    #2,_LOOP2                    ; Each instruction has 2 bytes.
    MOVE  P:(R1)+,B                    ; Get 8-bit from external P-memory
                                           ; and put into lower byte of B1
    ;BRA   <_PACK                       ; Branch to shared "packing" code
                                           ; (Commented out since falls through)

; *** Code Common to all three techniques

_PACK                                ; Combine upper and lower bytes:
    MOVE  B1,A1                        ; - Move the 8-bit into A1
    REP   #8                           ; - Shift 8 bit data from A1 into A0
    ASR   A

_LOOP2                                ; Note: This End-of-Loop is actually
                                           ; shared by three different loops,
                                           ; where the three loops are not
                                           ; nested, but rather, where only
                                           ; one of the loops is active at
                                           ; any given time.

    NOP                                ;
    MOVE  A0,A                          ; Store 16-bit instr in on-chip PRAM

    MOVE  A,P:(R0)+
```


Example A-1 DSP56L811 Bootstrap Code (Continued)

```
TSTW  (R2)-
BGT   _MAIN_LP

; ***** End of Main Loop *****

; ***

; *** Exit Bootstrap Program

; ***   - set the MODB, MODA bits to 10 (Mode 2),
; ***   without modifying any other OMR bits
; ***   - insert 1 instruction cycle delay
; ***   - goto P:$0000.
; ***   This instruction is fetched in Mode 0 or 1 (one of the bootstrap
; ***   modes), but the time execution for this instruction completes,
; ***   the chip will be in Mode 2 and the instruction at P:$0000 will
; ***   be fetched from on-chip PRAM instead of the bootstrap ROM.
; ***

        ANDC  #$FF00,SR          ; clear SR as if HW reset
        ANDC  $FFFE,OMR         ; clear OMR bit 0
        ORC   $0002,OMR         ; set   OMR bit 1
                                   ; (trigger an exit from bootstrap mode)

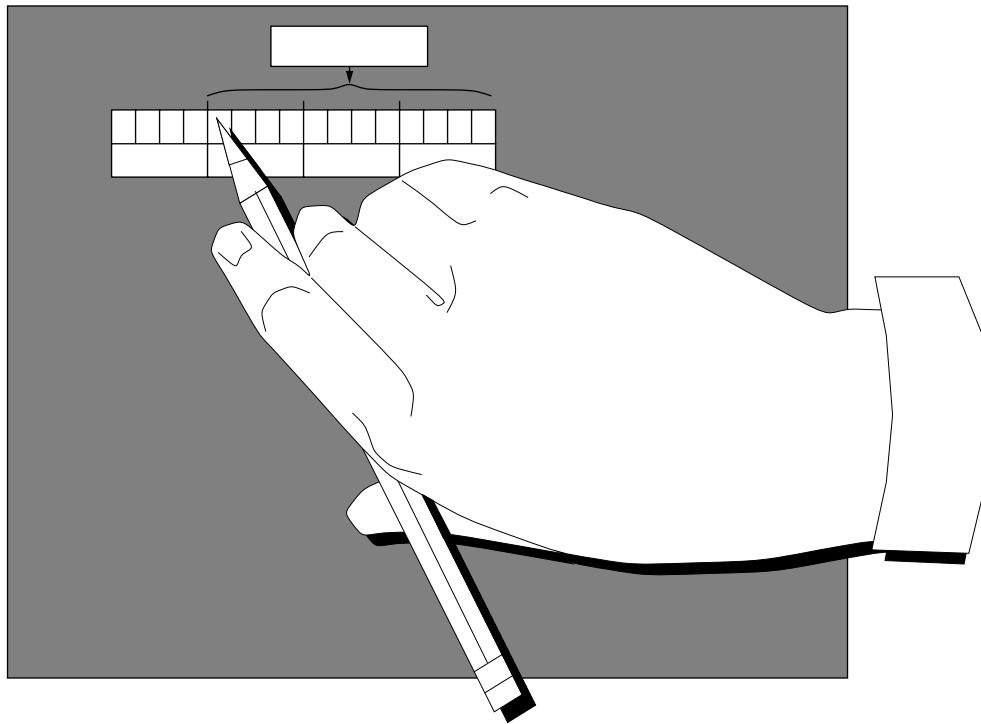
        NOP                    ; Introduce delay needed for
                                   ; operating mode change.

        BRA   <$0              ; Start fetching from PRAM.

END
```

APPENDIX B

PROGRAMMER'S SHEETS



B.1	INTRODUCTION	B-3
B.2	INSTRUCTION SET SUMMARY	B-3
B.3	INTERRUPT, VECTOR, AND ADDRESS TABLES	B-9
B.4	PROGRAMMER'S SHEETS	B-13

B.1 INTRODUCTION

The following pages provide a set of reference tables and programming sheets that are intended to simplify programming the DSP56L811. The programming sheets provide room to write in the value of each bit and the hexadecimal value for each register. The programmer can photocopy these sheets.

B.2 INSTRUCTION SET SUMMARY

The following tables provide a brief summary of the instruction set for the DSP56L811. **Table B-1** summarizes the instruction set. **Table B-2** and **Table B-3** on page B-8 provide a key to the abbreviations in the summary table. For complete instruction set details, see **Appendix A** of the *DSP56800 Family Manual (DSP56800FAM/AD)*.

Table B-1 Instruction Set Summary

Mnemonic	Syntax	Parallel Moves	Prog. Word	Clock Cycles	CCR						
					L	E	U	N	Z	V	C
ABS	D	(parallel move)	1	2+mv	*	*	*	*	*	*	—
ADC ¹	S,D	(no parallel move)	1	2	*	*	*	*	*	*	*
ADD	S,D	(parallel move)	1	2+mv	*	*	*	*	*	*	*
AND ¹	S,D	(parallel move)	1	2+mv	*	—	—	?	?	0	—
ANDC	#iiii,X:<ea> #iiii,D	...	2+ea	4+mvb	—	—	—	—	—	—	?
ASL	D	(parallel move)	1	2+mv	*	*	*	*	*	?	?
ASLL	S1,S2,D	(no parallel move)	1	2	?	?	?	*	*	?	?
ASR	D	(parallel move)	1	2+mv	*	*	*	*	*	0	?
ASRAC	S1,S2,D	(no parallel move)	1	2	?	?	?	*	*	?	?
ASRR	S1,S2,D	(no parallel move)	1	2	?	?	?	*	*	?	?
Bcc	xxxx ee Rn	...	2 1 1	4+jx	—	—	—	—	—	—	—

Table B-1 Instruction Set Summary (Continued)

Mne- monic	Syntax	Parallel Moves	Prog. Word	Clock Cycles	CCR						
					L	E	U	N	Z	V	C
BFCHG	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	...	2+ea	4+mvb	—	—	—	—	—	—	?
BFCLR	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	...	2+ea	4+mvb	—	—	—	—	—	—	?
BFSET	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	...	2+ea	4+mvb	—	—	—	—	—	—	?
BFTSTH	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	...	2+ea	4+mvb	—	—	—	—	—	—	?
BFTSTL	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	...	2+ea	4+mvb	—	—	—	—	—	—	?
BRA	xxxx aa Rn	...	2 1 1	4+jx	—	—	—	—	—	—	—
BRCLR	#iiii,X:<ea>,aa #iiii,D,aa	...	2+ea	8+mvb	—	—	—	—	—	—	?
BRSET			2+ea	8+mvb	—	—	—	—	—	—	?
BSR	xxxx Rn	...	2 1	4+jx							
CLR	D	(parallel move)	1	2+mv	*	*	*	*	*	0	—
CMP	S,D	(parallel move)	1	2+mv	*	*	*	*	*	*	*
DEBUG		...	1	4	—	—	—	—	—	—	—
DEC(W)	D	(parallel move)	1	2+mv	*	*	*	*	?	*	*
DIV	S,D	(parallel move)	1	2	*	—	—	—	—	?	?

Table B-1 Instruction Set Summary (Continued)

Mne- monic	Syntax	Parallel Moves	Prog. Word	Clock Cycles	CCR						
					L	E	U	N	Z	V	C
DO	X:(Rn),expr #xx,expr S,expr	...	2	6+mv if DO=0, else 10+mv	*	—	—	—	—	—	—
ENDDO		...	1	2	—	—	—	—	—	—	—
EOR ¹	S,D	(parallel move)	1	2+mv	*	—	—	?	?	0	—
EORC	#iiii,X:<ea> #iiii,D	...	2+ea	4+mvb	—	—	—	—	—	—	?
ILLEGAL		(no parallel move)	1	8	—	—	—	—	—	—	—
IMPY(16)	S1,S2,D	(no parallel move)	1	2	*	?	?	*	*	*	—
INC(W)	D	(parallel move)	1	2+mv	*	*	*	*	?	*	*
Jcc	xxxx (Rn)	...	1	4+jx	—	—	—	—	—	—	—
JMP	xxxx (Rn)	...	2 1	4+jx	—	—	—	—	—	—	—
JSR	xxxx AA Rn	...	2 1 1	4+jx	—	—	—	—	—	—	—
LEA	ea,D	(no parallel move)	1	4	—	—	—	—	—	—	—
LSL ¹	D	(parallel move)	1	2+mv	*	—	—	?	?	0	?
LSLL ¹	S1,S2,D	(no parallel move)									
LSR ¹	D	(parallel move)	1	2+mv	*	—	—	?	?	0	?
LSRAC ¹	S1,S2,D	(no parallel move)	1	2	?	?	?	*	*	?	?
LSRR ¹	S1,S2,D	(no parallel move)	1	2	?	?	?	*	*	?	?
MAC	(+)S2,S1,D S2,S1,D S1,S2,D	(no parallel move) (one parallel move) (two parallel reads)	1	2+mv	*	*	*	*	*	*	—
MACR	(+)S2,S1,D S2,S1,D S1,S2,D	(no parallel move) (one parallel move) (two parallel reads)	1	2+mv	*	*	*	*	*	*	—
MACSU	S1,S2,D	(no parallel move)	1	2	*	*	*	*	*	*	—
MOVE ⁴	X:<ea>,D S,X:<ea>	...	1	2+mv	*	—	—	—	—	—	—

Table B-1 Instruction Set Summary (Continued)

Mne- monic	Syntax	Parallel Moves	Prog. Word	Clock Cycles	CCR						
					L	E	U	N	Z	V	C
MOVE(C)	X:<ea>,D S,X:<ea> #xxxx,D S,D X:(R2+xx),D S,X:(R2+xx)	...	1+ea	2+mvc	?	?	?	?	?	?	?
MOVE(I)	#xx,D	...	1	2	—	—	—	—	—	—	—
MOVE(M)	P:<ea>,D S,P:<ea> P:(R2+xx),D S,P:(R2+xx) P:<ea>,X:<ea> X:<ea>,P:<ea>	...	1	2+mvm	*	—	—	—	—	—	—
MOVE(P)	X:<pp>,D X:<ea>,X:<pp> S,X:<pp> X:<pp>,X:<ea>	...	1	4+mvp	—	—	—	—	—	—	—
MOVE(S)	X:<a>,D S,X:<aa>	...	1	2+mvs	*	—	—	—	—	—	—
MPY	(±)S1,S2,D S1,S2,D S1,S2,D	(one parallel move) (two parallel reads) $\bar{D},X:(Rn)+(Nn)$	1	2+mv	*	*	*	*	*	*	—
MPYR	(±)S1,S2,D S1,S2,D	(one parallel move) (two parallel reads)	1	2+mv	*	*	*	*	*	*	—
MPYSU	S1,S2,D	(no parallel move)	1	2	*	*	*	*	*	*	—
NEG	D	(parallel move)	1	2+mv	*	*	*	*	*	*	*
NOP		...	1	2	—	—	—	—	—	—	—
NORM	Rn,D		1	2	*	*	*	*	*	?	—
NOT	D ₁	(parallel move)	1	2+mv	*	—	—	?	?	0	—
NOTC	X:<ea> D	...	2+ea	4+mvp	—	—	—	—	—	—	?
OR ¹	S,D	(parallel move)	1	2+mv	*	—	—	?	?	0	—
ORC	#iii,X:<ea> #iii,D	...	2+ea	4+mvp	—	—	—	—	—	—	?
POP	D	...	2	2+mv	?	?	?	?	?	?	?

Table B-1 Instruction Set Summary (Continued)

Mne- monic	Syntax	Parallel Moves	Prog. Word	Clock Cycles	CCR						
					L	E	U	N	Z	V	C
REP ⁴	X:(Rn) #xx S	...	1	6+mv if arg=0, else 4+mv	—	—	—	—	—	—	—
RESET					—	—	—	—	—	—	—
RND	D	(parallel move)	1	2+mv	*	*	*	*	*	*	—
ROL ¹	D	(parallel move)	1	2+mv	*	—	—	?	?	0	?
ROR ¹	D	(parallel move)	1	2+mv	*	—	—	?	?	0	?
RTI		...	1	4+rx	—	?	?	?	?	?	?
RTS		...	1	4+rx	—	—	—	—	—	—	—
SBC ₁	S,D	(parallel move)	1	2+mv	*	*	*	*	*	*	*
STOP ²		...	1	n/a	—	—	—	—	—	—	—
SUB	S,D S,D	(parallel move) (two parallel reads)	1+mva	2+mva	*	*	*	*	*	*	*
SWI		...	1	8	—	—	—	—	—	—	—
Tcc	S,D S,D R0,R1		1	2	—	—	—	—	—	—	—
TFR	S,D	(parallel move)	1	2+mv	—	—	—	—	—	—	—
TST	S	(parallel move)	1	2+mv	*	*	*	*	*	0	—
TST(W)	S	(no parallel move)	1	2+tst	*	*	*	*	*	0	—
WAIT ³		...	1	n/a	—	—	—	—	—	—	—

- Note:
1. These arithmetic instructions do not permit a parallel move.
 2. The STOP instruction disables the internal clock oscillator. After clock turn-on, an internal counter waits for 65,536 cycles before enabling the clock to the internal DSP circuits.
 3. The WAIT instruction takes a minimum of 16 cycles to execute when an internal interrupt is pending during the execution of the WAIT instruction.
 4. This MOVE instruction applies only to the case in which two reads are performed in parallel from the X memory.

Table B-2 Condition Code Register (CCR) Symbols (Standard Definitions)

Symbol	Description
L	Limit bit indicating arithmetic overflow and/or data limiting
E	Extension bit indicating if the integer portion is in use
U	Unnormalized bit indicating if the result is unnormalized
N	Negative bit indicating if bit 35 (or 31) of the result is set
Z	Zero bit indicating if the result equals 0
V	Overflow bit indicating if arithmetic overflow has occurred in the result
C	Carry bit indicating if a carry or borrow occurred in the result

Table B-3 Condition Code Register Notation

Notation	Description
*	Set according to the standard definition by the result of the operation
—	Not affected by the operation
0	Cleared
1	Set
U	Undefined
?	Set according to the special computation definition by the result of the operation

B.3 INTERRUPT, VECTOR, AND ADDRESS TABLES

Table B-4 Interrupt Priority Structure

Priority	Exception	Enabled By
Level 1 (Non-maskable)		
Highest	Hardware RESET	—
	COP Timer RESET	—
	Illegal Instruction Trap	—
	Hardware Stack Overflow	—
	OnCE Trap	—
Lower	Software Interrupt	—
Level 0 (Maskable)		
Higher	$\overline{\text{IRQA}}$ (External Interrupt)	IPR bits 2, 1
	$\overline{\text{IRQB}}$ (External Interrupt)	IPR bits 5, 4
	Channel 6 Peripheral Interrupt—SSI	IPR bit 9
	Channel 5 Peripheral Interrupt—Reserved	IPR bit 10
	Channel 4 Peripheral Interrupt—Timer Module	IPR bit 11
	Channel 3 Peripheral Interrupt—SPI1	IPR bit 12
	Channel 2 Peripheral Interrupt—SPI0	IPR bit 13
	Channel 1 Peripheral Interrupt—Realtime Timer	IPR bit 14
Lowest	Channel 0 Peripheral Interrupt—Port B GPIO	IPR bit 15

Table B-5 Reset and Interrupt Vectors

Interrupt Starting Address	Interrupt Priority Level	Interrupt Source
\$0000	—	Hardware RESET
\$0002	—	COP Timer RESET
\$0004	—	Reserved
\$0006	1	Illegal Instruction Trap
\$0008	1	Software Interrupt (SWI)
\$000A	1	Hardware Stack Overflow
\$000C	1	OnCE Trap

Table B-5 Reset and Interrupt Vectors (Continued)

Interrupt Starting Address	Interrupt Priority Level	Interrupt Source
\$000E	1	Reserved
\$0010	0	IRQA
\$0012	0	IRQB
\$0014	0	Port B GPIO Interrupt
\$0016	0	Real-Time Interrupt
\$0018	0	Timer 0 Overflow
\$001A	0	Timer 1 Overflow
\$001C	0	Timer 2 Overflow
\$001E	0	Reserved
\$0020	0	SSI Receive Data with Exception Status
\$0022	0	SSI Receive Data
\$0024	0	SSI Transmit Data with Exception Status
\$0026	0	SSI Transmit Data
\$0028	0	SPI1 Serial System
\$002A	0	SPI0 Serial System
\$002C	0	Reserved
\$002E	0	Reserved
\$0030	0	Reserved
\$0032	0	Reserved
\$0034	0	Reserved
\$0036	0	Reserved
\$0038	0	Reserved
\$003A	0	Reserved
\$003C	0	Reserved
\$003E	0	Reserved
\$0040	0	Reserved
\$0042	0	Reserved
\$007C	0	Reserved
\$007E	0	Reserved

Table B-6 DSP56L811 I/O and On-Chip Peripheral Memory Map

X:\$FFFF	Reserved for On-Chip Emulation (OnCE)
X:\$FFFE	(reserved)
X:\$FFFD	(reserved)
X:\$FFFC	(reserved)
X:\$FFFB	IPR—Interrupt Priority Register
X:\$FFFA	(reserved)
X:\$FFF9	BCR—Bus Control Register (Port A)
X:\$FFF8	(reserved)
X:\$FFF7	(reserved)
X:\$FFF6	(reserved)
X:\$FFF5	(reserved)
X:\$FFF4	(reserved)
X:\$FFF3	PCR1—PLL Control Register 1
X:\$FFF2	PCR0—PLL Control Register 0
X:\$FFF1	COPCTL—COP Control Register
X:\$FFF0	COPCNT—COP/RTI Count Register (read only) COPRST—COP Reset Register (write only)
X:\$FFEF	PCD—Port C Data Register
X:\$FFEE	PCDDR—Port C Data Direction Register
X:\$FFED	PCC—Port C Control Register
X:\$FFEC	PBD—Port B Data Register
X:\$FFEB	PBDDR—Port B Data Direction Register
X:\$FFEA	PBINT—Port B Interrupt Register
X:\$FFE9	(reserved)
X:\$FFE8	(reserved)
X:\$FFE7	(reserved)
X:\$FFE6	SPCR1—SPI1 Control Register
X:\$FFE5	SPSR1—SPI1 Status Register
X:\$FFE4	SPDR1—SPI1 Data Register
X:\$FFE3	(reserved)
X:\$FFE2	SPCR0—SPI0 Control Register
X:\$FFE1	SPSR0—SPI0 Status Register
X:\$FFE0	SPDR0—SPI0 Data Register
X:\$FFDF	TCR01—Timer 0 and 1 Control Register
X:\$FFDE	TPR0—Timer 0 Preload Register
X:\$FFDD	TCT0—Timer 0 Count Register

Table B-6 DSP56L811 I/O and On-Chip Peripheral Memory Map (Continued)

X:\$FFDC	TPR1—Timer 1 Preload Register
X:\$FFDB	TCT1—Timer 1 Count Register
X:\$FFDA	TCR2—Timer 2 Control Register
X:\$FFD9	TPR2—Timer 2 Preload Register
X:\$FFD8	TCT2—Timer 2 Count Register
X:\$FFD7	(reserved)
X:\$FFD6	(reserved)
X:\$FFD5	(reserved)
X:\$FFD4	SCRRX—SSI Receive Control Register
X:\$FFD3	SCRTX—SSI Transmit Control Register
X:\$FFD2	SCR2—SSI Control Register 2
X:\$FFD1	SSR—SSI Status Register (read only) STSR—SSI Time Slot Register (write only)
X:\$FFD0	SRX—SSI Receive Register (read only) STX—SSI Transmit Register (write only)
X:\$FFCF	(reserved)
X:\$FFCE	(reserved)
X:\$FFCD	(reserved)
X:\$FFCC	(reserved)
X:\$FFCB	(reserved)
X:\$FFCA	(reserved)
X:\$FFC9	(reserved)
X:\$FFC8	(reserved)
X:\$FFC7	(reserved)
X:\$FFC6	(reserved)
X:\$FFC5	(reserved)
X:\$FFC4	(reserved)
X:\$FFC3	(reserved)
X:\$FFC2	(reserved)
X:\$FFC1	(reserved)
X:\$FFC0	(reserved)

B.4 PROGRAMMER'S SHEETS

The following pages provide programmer's sheets that are intended to simplify programming the various registers in the DSP56L811. The programmer's sheets provide room to write in the value of each bit and the hexadecimal value for each register. The programmer can photocopy these sheets.

The programmer's sheets are provided in the same order as the sections in this document. **Table B-7** lists the sets of programmer's sheets, the registers described in the sheets, and the pages in this appendix where the sheets are located.

Table B-7 List of Programmer's Sheets

Type of Register	Register	Page
CPU	JTAG Instruction Register	B-15
	JTAG Bypass Register	B-15
	JTAG ID Register	B-15
	JTAG Boundary Scan Register	B-16
	SR—Status Register (includes Mode Register and Condition Code Register)	B-17
	IPR—Interrupt Priority Register	B-18
	BCR—Bus Control Register	B-19
Port B GPIO	PBINT—Port B Interrupt Register	B-20
	PBD—Port B Data Register	B-20
	PBINT—Port B Data Direction Register	B-20
Port C GPIO	PCC—Port C Control Register	B-21
	PCD—Port C Data Register	B-21
	PCDDR—Port C Data Direction Register	B-21
SPI0	SPCR0—SPI 0 Control Register	B-22
	SPSR0—SPI 0 Status Register	B-23
	SPDR0—SPI 0 Data Register	B-23
SPI1	SPCR1—SPI 1 Control Register	B-24
	SPCS1—SPI 1 Status Register	B-25
	SPDR1—SPI 1 Data Register	B-25

Table B-7 List of Programmer's Sheets

Type of Register	Register	Page
SSI	SCRRX—SSI Receive Control Register	B-26
	SCRTX—SSI Transmit Control Register	B-26
	SCR2—SSI Control Register 2	B-27
	SSR—SSI Status Register	B-28
	STSR—SSI Time Slot Register	B-28
	STX—SSI Transmit Register	B-29
	STX—SSI Receive Register	B-29
Timer 0	TCR01—Timer 0/1 Control Register	B-30
	TPR0—Timer 0 Preload Register	B-30
	TCT0—Timer 0 Count Register	B-30
Timer 1	TCR01—Timer 0/1 Control Register	B-31
	TPR1—Timer 1 Preload Register	B-31
	TCT1—Timer 1 Count Register	B-31
Timer 2	TCR2—Timer 2 Control Register	B-32
	TPR2—Timer 2 Preload Register	B-32
	TCT2—Timer 2 Count Register	B-32
PLL	PCR1—PLL Control Register 1	B-33
	PCR0—PLL Control Register 0	B-33
COP/RTI	COPCTL—COP Control Register	B-34
	COPCNT—COP Count Register	B-34

Application:_____

Date:_____

Programmer:_____

Sheet 1 of 5

CPU

JTAG Instruction Register

Reset = \$2

Read/Write

3210

B3B2B1B0

JTAG Bypass Register

Reset = \$0

Read/Write

0

JTAG ID Register

Read Only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK 7	MSK 6	MSK 5	MSK 4	MSK 3	MSK 2	MSK 1	MSK 0	INV 7	INV 6	INV 5	INV 4	INV 3	INV 2	INV 1	INV 0

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

Application: _____

Date: _____

Programmer: _____

Sheet 2 of 5

CPU

**JTAG Boundary Scan
Register
Read Only**

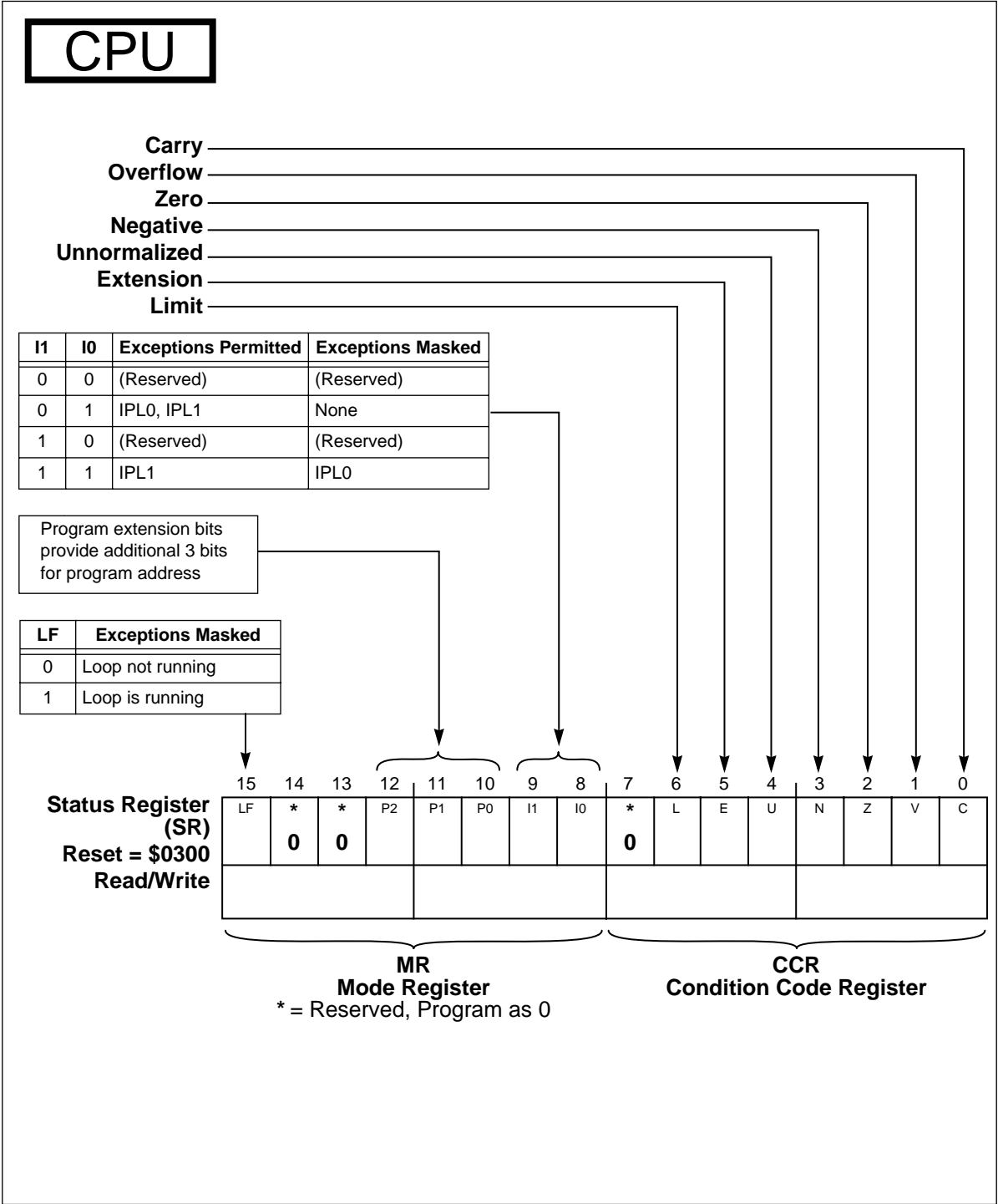
Scan
ister
Only

110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	
RE SET	MODB/ IRQB	MODA/ IRQA	PB0 ctrl	PB0	PB1 ctrl	PB1	PB2 ctrl	PB2	PB3 ctrl	PB3	PB4 ctrl	PB4	PB5 ctrl	PB5	
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
PB6 ctrl	PB6	PB7 ctrl	PB7	PB8 ctrl	PB8	PB9 ctrl	PB9	PB10 ctrl	PB10	PB11 ctrl	PB11	PB12 ctrl	PB12	PB13 ctrl	PB13
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
PB14 ctrl	PB14	PB15 ctrl	PB15	CLKO	EX	PC0 MISO0 ctrl	PC0 MISO0	PC1/ MOSIO ctrl	PC1/ MOSIO	PC2/ SCK0 ctrl	PC2/ SCK0	PC3/ SS0 ctrl	PC3/ SS0	PC4/ MISO1 ctrl	PC4/ MISO1
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
PC5/ MOSI1 ctrl	PC5/ MOSI1	PC6/ SCK1 ctrl	PC6/ SCK1	PC7/ SS1 ctrl	PC7/ SS1	PC8/ STD ctrl	PC8 STD	PC9/ SRD ctrl	PC9/ SRD	PC10/ STCK ctrl	PC10/ STCK	PC11/ STFS ctrl	PC11/ STFS	PC12/ SRCK ctrl	PC12/ SRCK
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
PC13/ SRFS ctrl	PC13/ SRFS	PC14/ TIO01 ctrl	PC14/ TIO01	PC15/ TIO2 ctrl	PC15/ TIO2	WR ctrl	WR	RD ctrl	RD	A15 ctrl	A15	A14	A13	A12	A11
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A10	A9	A8	A7	A6	A5	PS ctrl	PS	DS ctrl	DS	A4	A3	A2	A1	A0	D0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1	D2	D3	D4	D5	D6	D7	D8	D9 ctrl	D10	D11	D12	D13	D14	D15 ctrl	D15

Application: _____

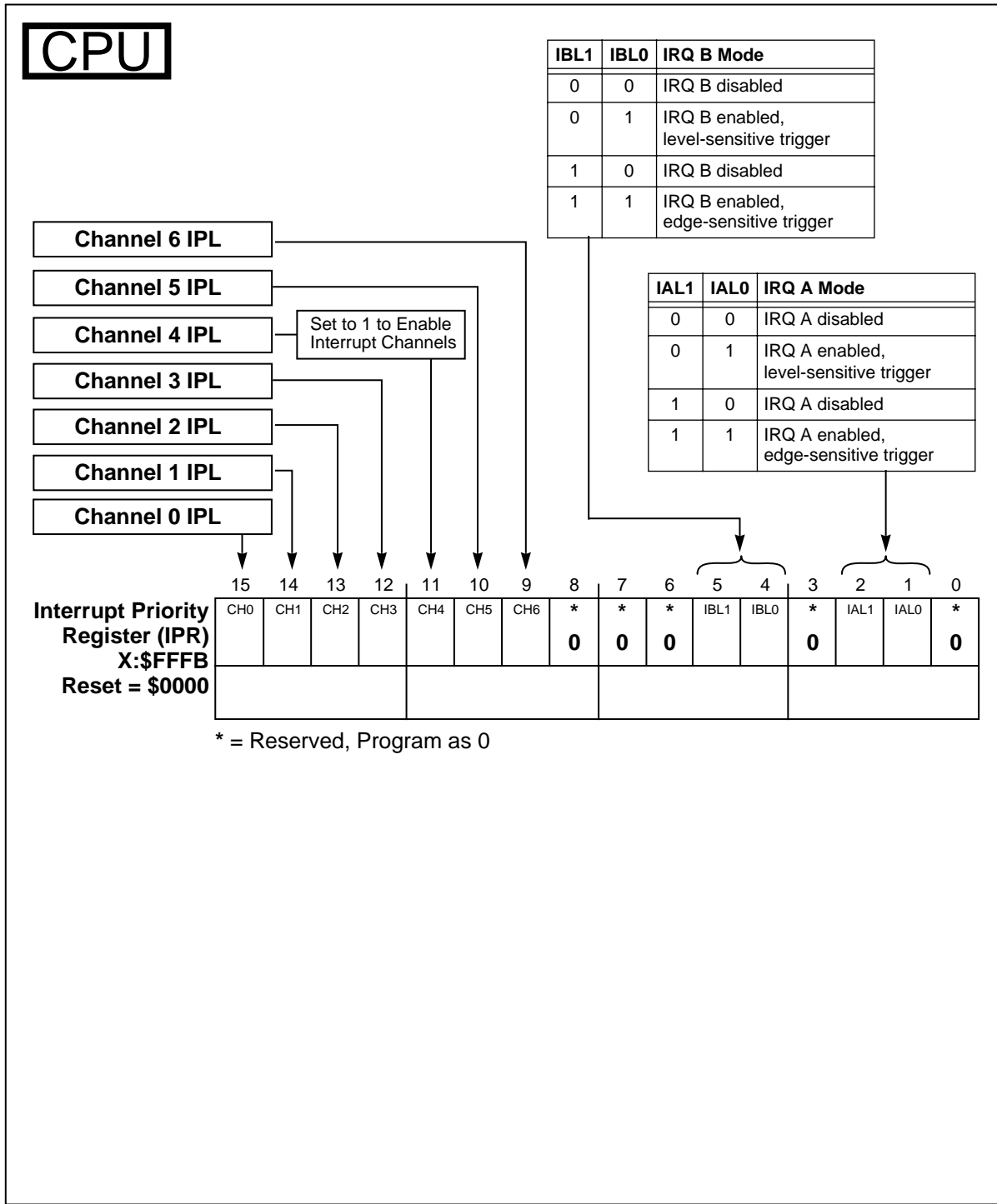
Date: _____
Programmer: _____

Sheet 3 of 5



Application: _____ Date: _____
Programmer: _____

Sheet 4 of 5

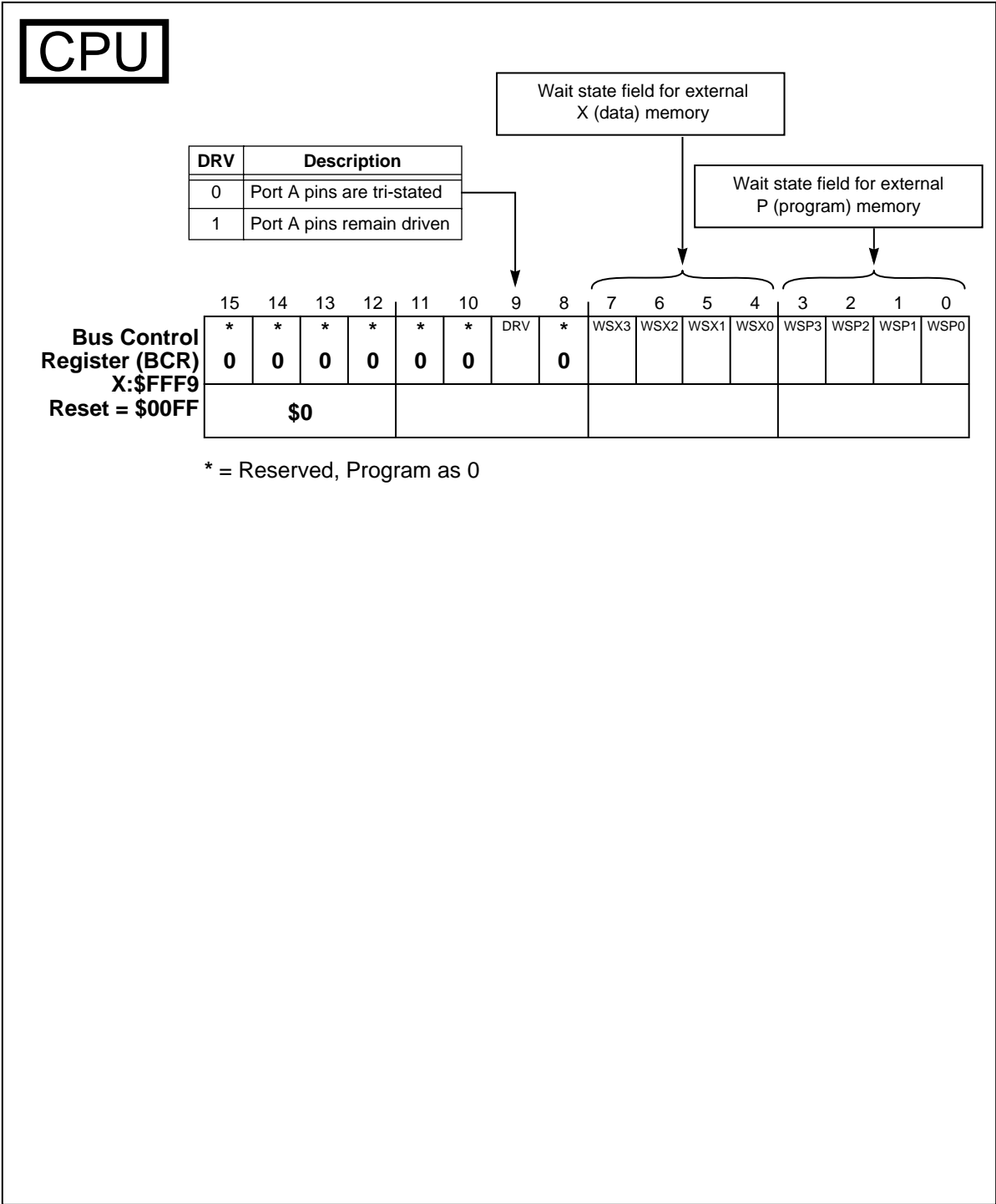


Application:_____

Date:_____

Programmer:_____

Sheet 5 of 5



Application: _____

Date: _____

Programmer: _____

Sheet 1 of 1

Port B
GPIO

INVn	Description
0	Detects rising edge
1	Detects falling edge

MSKn	Description
0	Pin masked from generating interrupt
1	Pin enabled for generating interrupt

Port B Interrupt Register (PBINT)
X:\$FFEA
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	INV7	INV6	INV5	INV4	INV3	INV2	INV1	INV0

Arrows from the MSK and INV tables point to these columns in the PBINT register.

Port B Data Register (PBD)
X:\$FFEC
Reset = Uninitialized

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BD15	BD14	BD13	BD12	BD11	BD10	BD9	BD8	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0

Port B Data Direction Register (PBDDR)
X:\$FFEB
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDD15	BDD14	BDD13	BDD12	BDD11	BDD10	BDD9	BDD8	BDD7	BDD6	BDD5	BDD4	BDD3	BDD2	BDD1	BDD0

Application: _____

Date: _____

Programmer: _____

Sheet 1 of 1

Port C GPIO

PCC register must be configured for Port C GPIO, Timers 0-2, SPI 0-1, and SSI. Use this programming sheet for all these peripherals.

Timer				Synchronous Serial Interface (SSI)				SPI1				SPI0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0

Port C Control Register (PCC)
X:\$FFED
Reset = \$0000

CCn	Description
0	Pin is GPIO Pin
1	Pin used for dedicated peripheral

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD15	CD14	CD13	CD12	CD11	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0

Port C Data Register (PCD)
X:\$FFEF
Reset = Uninitialized

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDD15	CDD14	CDD13	CDD12	CDD11	CDD10	CDD9	CDD8	CDD7	CDD6	CDD5	CDD4	CDD3	CDD2	CDD1	CDD0

Port C Data Direction Register (PCDDR)
X:\$FFEE
Reset = \$0000

Application: _____

Date: _____

Programmer: _____

Sheet 1 of 2

SPI0

CPL	Description
0	SCK Pin Idles at Logic Low
1	SCK Pin Idles at Logic High

WOM	Description
0	SPI Pins Configured as Push-Pull Drivers
1	SPI Pins Configured as Open-Drain Drivers

MST	Description
0	SPI in Slave Mode
1	SPI in Master Mode

SPE	Description
0	SPI Disabled
1	SPI Enabled

SPR2	SPR1	SPR0	Divider
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

SPTE	Description
0	SPI Interrupt Disabled
1	SPI Interrupt Enabled

CHP
Controls clock
phase—see Section 7.

**SPI 0 Control
Register (SPCR0)**
X:\$FFE2
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	SPR2	SPIE	SPE	WOM	MST	CPL	CPH	SPR1	SPR0
0	0	0	0	0	0	0									
\$0															

* = Reserved, Program as 0

Application: _____

Date: _____

Programmer: _____

Sheet 2 of 2

SPI0

WCOL	Description
0	(Bit is cleared)
1	Write Collision Detected

SPIF	Description
0	(Bit is cleared)
1	Data transfer is completed

MDF	Description
0	(Bit is cleared)
1	Mode Fault Detected

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI 0 Status Register (SPSR0)	*	*	*	*	*	*	*	*	SPIF	WCOL	*	MDF	*	*	*	*
X:\$FFE1	0	0	0	0	0	0	0	0			0		0	0	0	0
Reset = \$0000	\$0				\$0											

* = Reserved, Program as 0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI 0 Data Register (SPDR0)	*	*	*	*	*	*	*	*	data	data	data	data	data	data	data	data
X:\$FFE0	0	0	0	0	0	0	0	0								
Reset = Uninitialized	\$0				\$0											

* = Reserved, Program as 0

Application: _____

Date: _____

Programmer: _____

Sheet 1 of 2

SPI1

CPL	Description
0	SCK Pin Idles at Logic Low
1	SCK Pin Idles at Logic High

WOM	Description
0	SPI Pins Configured as Push-Pull Drivers
1	SPI Pins Configured as Open-Drain Drivers

MST	Description
0	SPI in Slave Mode
1	SPI in Master Mode

SPE	Description
0	SPI Disabled
1	SPI Enabled

SPR2	SPR1	SPR0	Divider
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

SPTE	Description
0	SPI Interrupt Disabled
1	SPI Interrupt Enabled

CHP
Controls clock
phase—see Section 10.

**SPI 1 Control
Register (SPCR1)**
X:\$FFE6
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	SPR2	SPIE	SPE	WOM	MST	CPL	CPH	SPR1	SPR0
0	0	0	0	0	0	0									
\$0															

* = Reserved, Program as 0

Application: _____

Date: _____

Programmer: _____

Sheet 2 of 2

SPI1

WCOL	Description
0	(Bit is cleared)
1	Write Collision Detected

SPIF	Description
0	(Bit is cleared)
1	Data transfer is completed

MDF	Description
0	(Bit is cleared)
1	Mode Fault Detected

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	*	*	*	*	*	*	*	*	SPIF	WCOL	*	MDF	*	*	*	*
SPI 1 Status Register (SPSR1)	0	0	0	0	0	0	0	0			0		0	0	0	0
X:\$FFE5																
Reset = \$0000	\$0				\$0											

* = Reserved, Program as 0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	*	*	*	*	*	*	*	*	data	data	data	data	data	data	data	data
SPI 1 Data Register (SPDR1)	0	0	0	0	0	0	0	0								
X:\$FFE4																
Reset = Uninitialized	\$0				\$0											

* = Reserved, Program as 0

Application: _____

Date: _____

Programmer: _____

Sheet 1 of 4

SSI

WL1	WL0	Description
0	0	8 bits per word
0	1	10 bits per word
1	0	12 bits per word
1	1	16 bits per word

PSR	Description
0	Prescaler Disabled
1	Prescaler ÷8 Enabled

Frame Rate Divider Bits

Prescale Modulus Bits

**SSI Receive Control
Register (SCRRX)**
X:\$FFD4
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

**SSI Transmit
Control Register
(SCRTX)**
X:\$FFD3
Reset = \$0000

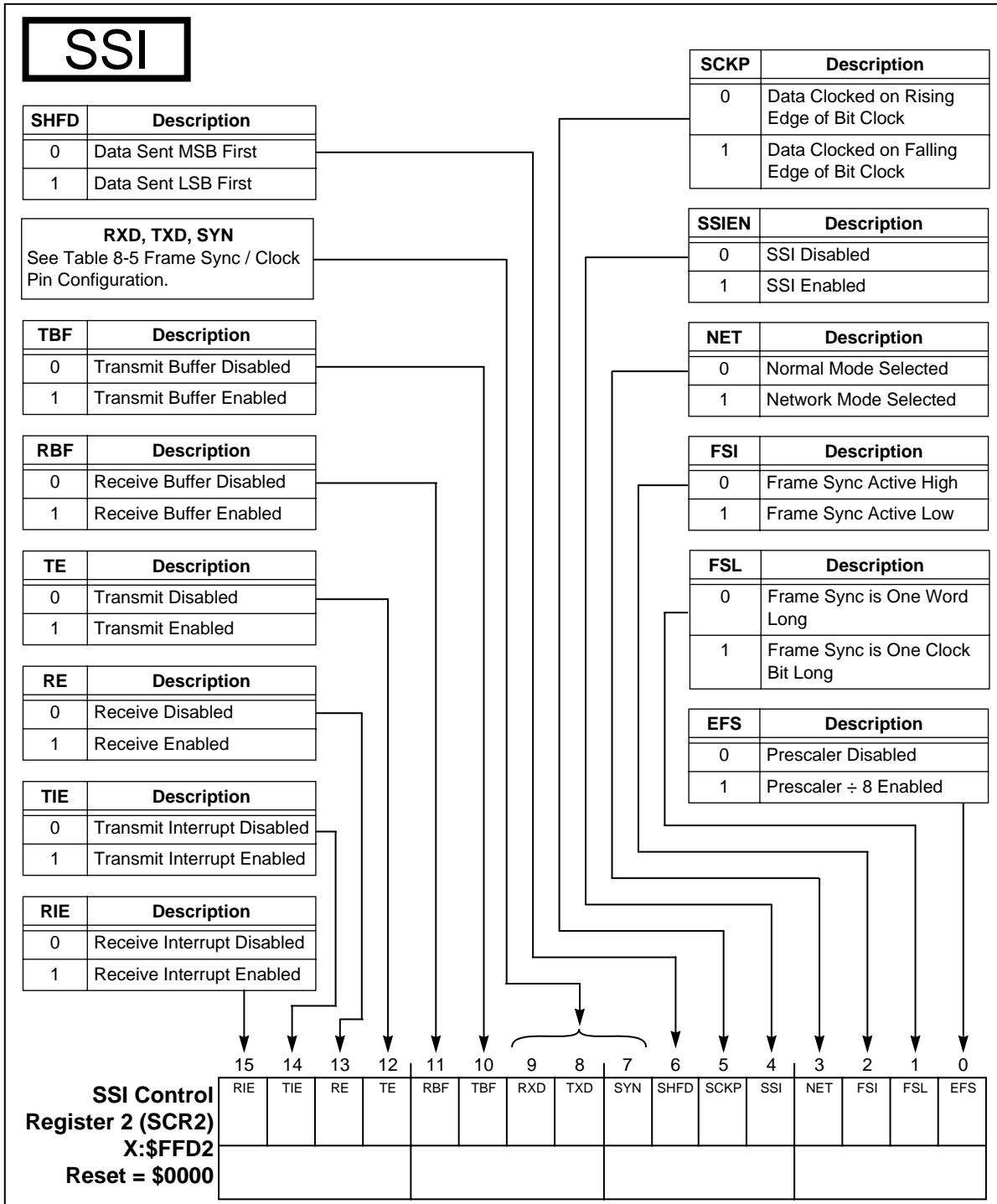
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

Application: _____

Date: _____

Programmer: _____

Sheet 2 of 4



Application:_____

Date:_____

Programmer:_____

Sheet 3 of 4

SSI

TUE	Description
0	(Bit is cleared)
1	TX Underrun Occurred

ROE	Description
0	(Bit is cleared)
1	RX Overrun Occurred

TDE	Description
0	(Bit is cleared)
1	TX Data Register Empty

RDF	Description
0	(Bit is cleared)
1	RX Data Register has data

TDBE	Description
0	Transmit Buffer Empty
1	Transmit Buffer Full

RDBF	Description
0	Receive Buffer Empty
1	Receive Buffer Full

TFS	Description
0	No Frame Sync
1	Frame Sync Occurred

RFS	Description
0	No Frame Sync
1	Frame Sync Occurred

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	RDF	TDE	ROE	TUE	RFS	TFS	RDBF	TDBE
0	0	0	0	0	0	0	0								
\$0				\$0											

* = Reserved, Program as 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Dummy Register, Written During Inactive Time Slots															

Application:_____

Date:_____

Programmer:_____

Sheet 4 of 4

SSI

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSI Transmit Register (STX)	High Byte								Low Byte							
X:\$FFD0																
Write-Only																

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSI Receive Register (SRX)	High Byte								Low Byte							
X:\$FFD0																
Read-Only																

Application: _____

Date: _____

Programmer: _____

Sheet 1 of 1

Timer 0

TO10	TO00	Description
0	0	TIO pin configured as input
0	1	(Reserved)
1	0	Overflow pulse
1	1	Overflow toggle

OIE0	Description
0	Overflow Interrupt Disabled
0	Overflow Interrupt Enabled

INV	Description
0	TIO Pin Detects Rising Edge
1	TIO Pin Detects Falling Edge

TE0	Description
0	Timer 0 Disabled
1	Timer 0 Enabled

ES10	ES00	Description
0	0	Internal Phi Clock ÷ 4 Selected
0	1	Internal Prescaler Clock Selected
1	0	Previous Timer Overflow Selected
1	1	External Event from TIO Pin

Timer 0/1
Control Register
(TCR01)
X:\$FFDF
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TE1	*	*	OIE1	TO11	TO01	ES11	ES01	TE0	INV	*	OIE0	TO10	TO00	ES10	ES00
X	0	0	X	X	X	X	X			0					
\$0															

* = Reserved, Program as 0

X = Used for Timer 1, Program Accordingly

Timer 0
Preload Register
(TPR0)
X:\$FFDE
Reset = Uninitialized
Write-Only

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

Timer 0
Count Register
(TCT0)
X:\$FFDD
Reset = Uninitialized

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

Application: _____

Date: _____

Programmer: _____

Sheet 1 of 1

Timer 1

TO11	TO01	Description
0	0	TIO pin configured as input
0	1	(Reserved)
1	0	Overflow pulse
1	1	Overflow toggle

OIE1	Description
0	Overflow Interrupt Disabled
1	Overflow Interrupt Enabled

TE1	Description
0	Timer 1 Disabled
1	Timer 1 Enabled

ES11	ES01	Description
0	0	Internal Phi Clock ÷ 4 Selected
0	1	Internal Prescaler Clock Selected
1	0	Previous Timer Overflow Selected
1	1	External Event from TIO Pin

INV	Description
0	TIO Pin Detects Rising Edge
1	TIO Pin Detects Falling Edge

**Timer 0/1
Control Register
(TCR01)**
X:\$FFDF
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TE1	*	*	OIE1	TO11	TO01	ES11	ES01	TE0	INV	*	OIE0	TO10	TO00	ES10	ES00
	0	0						X		0	X	X	X	X	X
\$0															

* = Reserved, Program as 0

X = Used for Timer 0, Program Accordingly

**Timer 1
Preload Register
(TPR1)**
X:\$FFDC
Reset = Uninitialized
Write-Only

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

**Timer 1
Count Register
(TCT1)**
X:\$FFDB
Reset = Uninitialized

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

Application:_____

Date:_____

Programmer:_____

Sheet 1 of 1

Timer 2

TO12	TO02	Description
0	0	TIO pin configured as input
0	1	(Reserved)
1	0	Overflow pulse
1	1	Overflow toggle

OIE2	Description
0	Overflow Interrupt Disabled
1	Overflow Interrupt Enabled

INV	Description
0	TIO Pin Detects Rising Edge
1	TIO Pin Detects Falling Edge

TE2	Description
0	Timer 2 Disabled
1	Timer 2 Enabled

ES12	ES02	Description
0	0	Internal Phi Clock ÷ 4 Selected
0	1	Internal Prescaler Clock Selected
1	0	Previous Timer Overflow Selected
1	1	External Event from TIO Pin

Timer 2 Control Register (TCR2) X:\$FFDA Reset = \$0000	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	*	*	*	*	*	*	*	*	TE2	INV	*	OIE2	TO10	TO02	ES12	ES02
	0	0	0	0	0	0	0	0			0					
	\$0				\$0											

* = Reserved, Program as 0

Timer 2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Preload Register (TPR2)																
X:\$FFD9																
Reset = Uninitialized																
Write-Only																

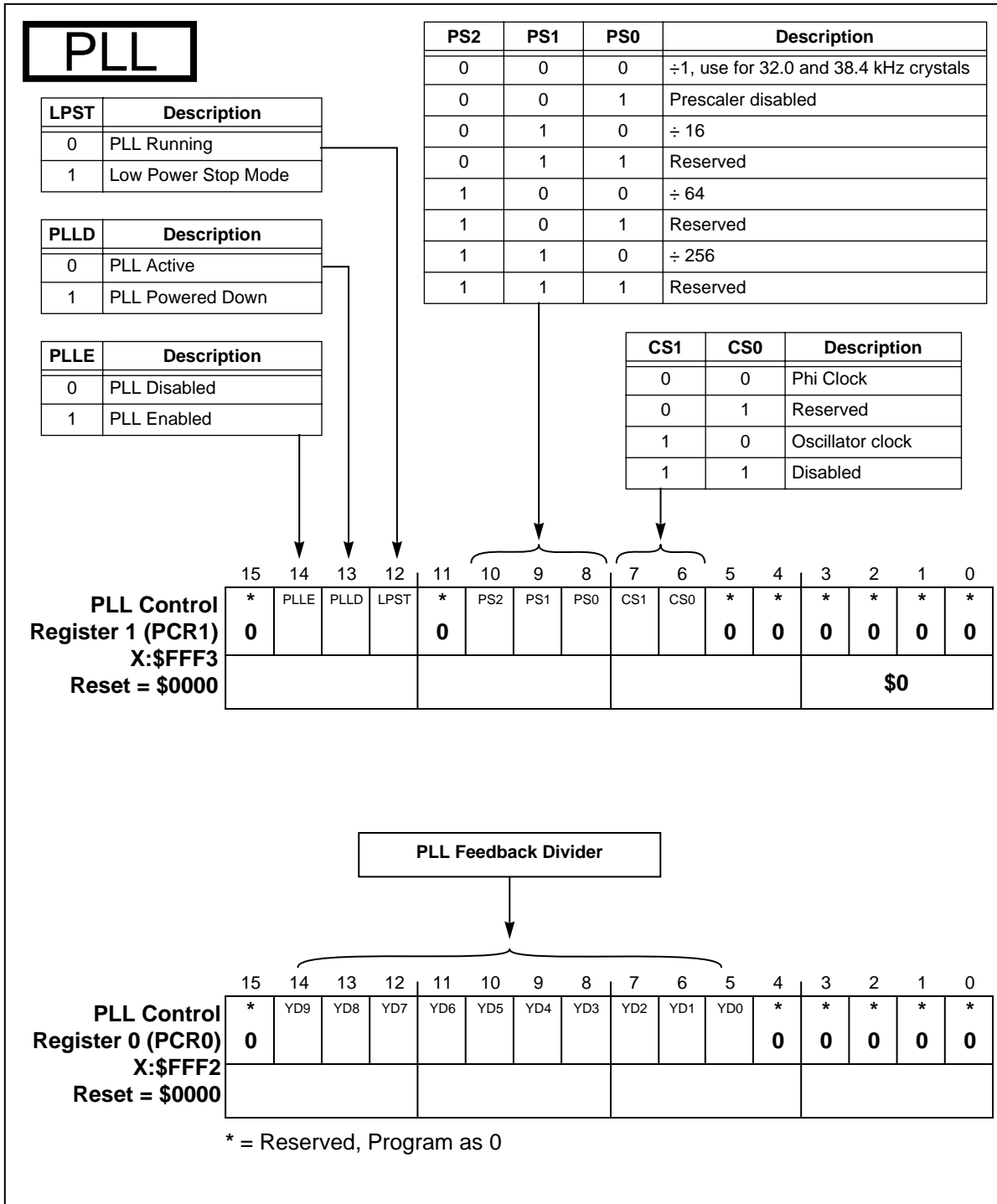
Timer 2 Count Register (TCT2) X:\$FFD8 Reset = Uninitialized	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Application: _____

Date: _____

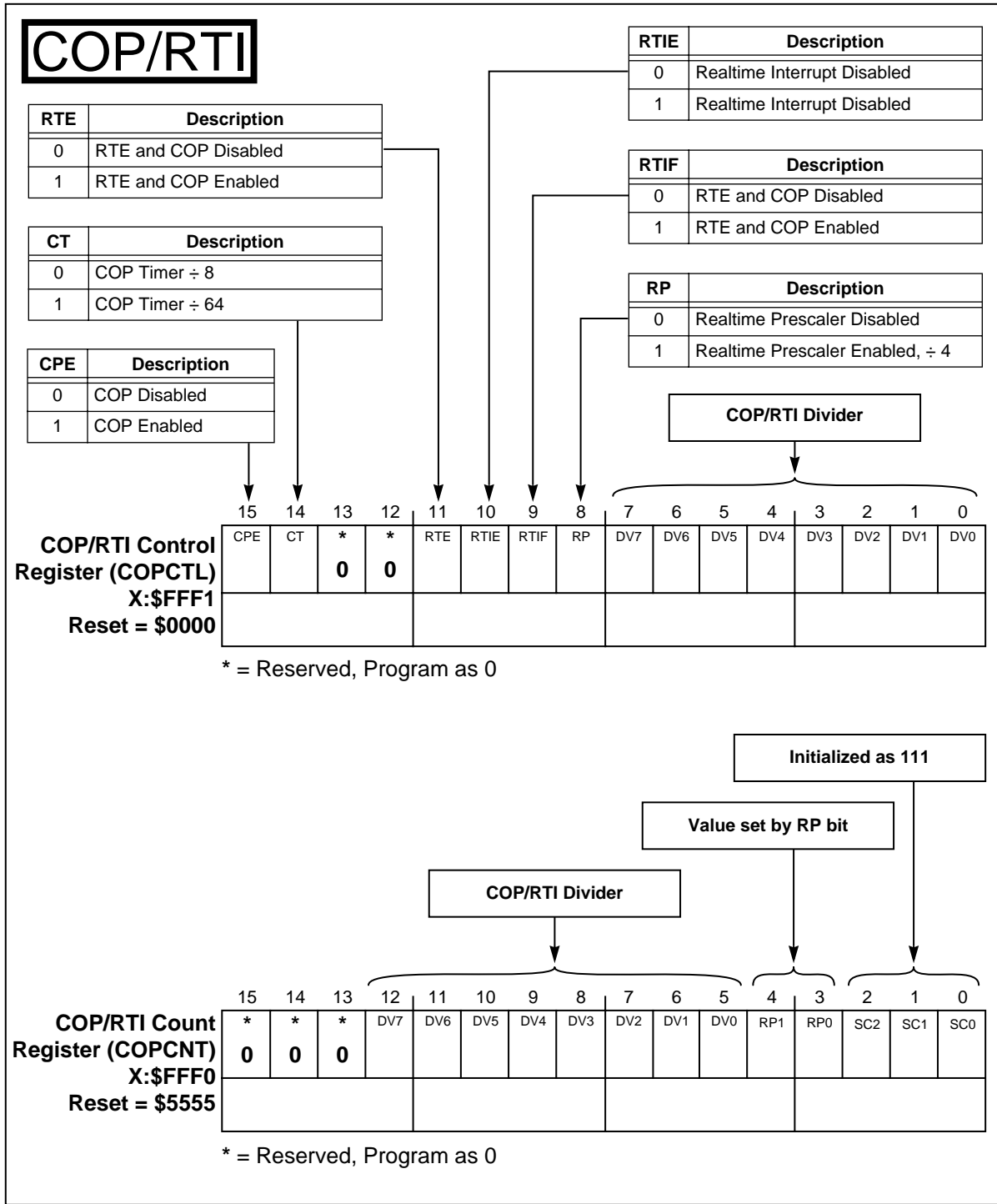
Programmer: _____

Sheet 1 of 1



Application: _____ Date: _____
Programmer: _____

Sheet 1 of 1



INDEX

A

A0-A15 Pins 2-7
 Address and Data Buses 1-10
 Address Bus Pins (A0-A15) 2-7
 Address Generation Unit 1-10

B

BCR Register 3-4, 4-5
 DRV Bit 4-5
 Reserved Bits 4-8
 WSPM Bits 4-6
 WSX Bits 4-6
 Bootstrap Mode
 Mode 0 3-10
 Mode 1 3-10
 Bootstrap ROM 3-6, A-3
 Bootstrapping 3-11
 Boundary-Scan Register 12-13
 Bus and Bit Manipulation Unit 1-11
 Bus Control Register (BCR) 3-4, 4-5
 BYPASS Instruction 12-11
 Bypass Register 12-16

C

CCR Register 3-17
 CDD Bit 6-6
 CLKO Pin 2-7
 Turning Off When Not in Use 10-18
 CLKO Select Bits (CSI-CS0) 10-10
 Clock Output Pin (CLKO) 2-7
 Clock Phase Bit (CPH) 7-10
 Clock Polarity (SCKP) Bit 8-18
 Clock Polarity Bit (CPL) 7-10
 Clock Synthesis Module 1-15
 Clocking SSI 8-6
 Clockout MUX 10-7
 Clocks
 Turning Off 10-17
 Computer Operating Properly (COP) Timer 11-3
 Condition Code Register (CCR) 3-17
 Configuring Port C for SPI Functionality 7-16
 Configuring Port C for SSI Functionality 8-36
 COP
 Programming 11-10
 COP and RTI Control Register (COPCTL) 11-6
 COP and RTI Count Register (COPCNT) 11-8
 COP Enable Bit (CPE) 11-7

COP Reset Register (COPRST) 11-9
 COP Timer Divider Bit (CT) 11-7
 COP/RTI Module 1-15
 COP/RTI Timer
 Low Power Operation 11-11
 Programming 11-10
 COPCNT Register 11-8
 Reserved Bits 11-9
 COPCTL Register 11-6
 Reserved Bits 11-8
 COPRST Register 11-9
 Core Global Data Bus (CGDB) 1-11
 CPE Bit 11-7
 CPH Bit 7-10
 CPL Bit 7-10
 Crystal Oscillator 10-5
 Crystal Output Pin (XTAL) 2-6
 CSI-CS0 Bits 10-10
 CT Bit 11-7

D

D0-D15 Pins 2-7
 Data ALU 1-9
 Data Bus 1-10
 Data Bus Pins (D0-D15) 2-7
 Data Memory Select Pin (DS) 2-8
 DC4-DC0 Bits 8-13
 DEBUG_REQUEST Instruction 12-11
 Development Mode (Mode 3) 3-11
 Drive Bit (DRV) 4-5
 DRV Bit 4-5
 \overline{DS} Pin 2-8
 DV7-DV0 Bits 11-8, 11-9

E

Early Frame Sync (EFS) Bit 8-19
 EFS Bit 8-19
 ENABLE_ONCE Instruction 12-11
 ES1-ES0 Bits 9-8
 Event Counting 9-11
 Event Select Bits (ES1-ES0) 9-8
 EX Bit
 in OMR Register 3-5
 Executing Programs from XRAM 3-12
 EXTAL Pin 2-6
 External Address Bus (EAB) 1-10
 External Clock/Crystal Input Pin (EXTAL) 2-6
 External Data Bus Pins (D0-D15) 2-7
 External Filter Capacitor Pin (SXFC) 2-7

F

External Memory 1-14
External Memory Port Architecture 4-3
External Memory Port Programing Model 4-5
EXTEST Instruction 12-8
EXTEST_PULLUP Instruction 12-10

F

Feedback Divider Bits (YD9-YD0) 10-11
Frame Rate Divider (DC4-DC0) Bits 8-13
Frame Sync Invert (FSI) Bit 8-19
Frame Sync Length (FSL) Bit 8-19
FSI Bit 8-19
FSL Bit 8-19

G

General Purpose I/O Module 1-14
General Purpose Input/Output 5-3
General Purpose Timers 9-3
General-Purpose Timer Module 1-15
GPIO 5-3
Ground Pins (V_{SS}) 2-6

H

Harvard Architecture
 For Memory 3-3
 for On-chip Memory 1-14
HIGHZ Instruction 12-10

I

IDCODE Instruction 12-9
Interrupt
 Priority 3-19
 Priority Register (IPR) 3-19
 Vector Map 3-15
Interrupt Invert Bits (INV7-INV0) 5-8
Interrupt Mask Bits (MSK7-MSK0) 5-7
INV Bit 9-6
INV7-INV0 Bits 5-8
Invert Bit (INV) 9-6
IPR Register 3-19

J

JTAG
 Boundary-Scan Register 12-13
 Bypass Register 12-16

Chip Identification Register 12-12
Decoder 12-8
Instruction Register 12-8
Registers 12-6
Restrictions 12-18
TCK Pin 2-16
TDI Pin 2-16
Test Clock Input Pin (TCK) 12-6
Test Data Input Pin (TDI) 12-6
Test Data Output Pin (TDO) 12-6
Test Mode Select Input Pin (TMS) 12-6
Test Reset/Debug Event Pin
 ($\overline{TRST/DE}$) 12-6
TMS Pin 2-16

JTAG Instruction
 BYPASS 12-11
 DEBUG_REQUEST 12-11
 ENABLE_ONCE 12-11
 EXTEST 12-8
 EXTEST_PULLUP 12-10
 HIGHZ 12-10
 IDCODE 12-9
 SAMPLE/PRELOAD 12-9
JTAG Port
 Architecture 12-4
 Pinout 12-5
JTAG/OnCE Port 1-15

L

Low Frequency Timer Operation 9-20
Low Power Operation
 SPI 7-17
Low Power Stop Bit (LPST) 10-9
Low-Power Operation
 PLL 10-17
LPST Bit 10-9

M

Master Mode Select Bit (MST) 7-10
MDF Bit 7-11
Memory Map Description 3-3
MISO0/PC0 Pin 7-12
MISO1/PC4 Pin 7-13
MODA/ \overline{IRQA} Pin 2-9
MODB/ \overline{IRQB} Pin 2-9
Mode Fault Bit (MDF) 7-11
Mode Register (MR) 3-17

Mode Select A/External Interrupt Request A Pin
(MODA/ $\overline{\text{IRQA}}$) 2-9
Mode Select B/External Interrupt Request B Pin
(MODB/ $\overline{\text{IRQB}}$) 2-9
MOSI0/PC1 Pin 7-12
MOSI1/PC5 Pin 7-13
MR Register 3-17
MSK7-MSK0 Bits 5-7
MST Bit 7-10

N

NET Bit 8-19
Network Mode (NET) Bit 8-19
Normal Expanded Mode (Mode 2) 3-10

O

OIE Bit 9-7
OMR (Operating Mode Register) 3-5
On-Chip Emulation (OnCE) 1-12
On-Chip Emulation (OnCE) Port 12-3
On-chip Memory 1-14
On-Chip Peripheral Memory Map 3-7
Operating Mode
 0 3-10
 1 3-10
 2 3-10
 3 3-11
Operating Mode Register (OMR) 3-5
Operating Modes 3-9
Overflow Interrupt Enable Bit (OIE) 9-7

P

PB0-PB7 Pins 2-10
PB8-PB15 Pins 2-10
PBD Register 5-6
PBDDR Register 5-6
PBINT Register 5-7
 INV7-INV0 Bits 5-8
 MSK7-MSK0 Bits 5-7
PC1/MOSI0 Pin 2-11
PC10/STCK Pin 2-14
PC11/STFS Pin 2-14
PC12/SRCK Pin 2-14
PC13/SRFS Pin 2-15
PC14/TIO01 Pin 2-15
PC15/TIO2 Pin 2-16
PC2/SPCK0 Pin 2-11
PC3/ $\overline{\text{SS0}}$ Pin 2-12

PC4/MISO1 Pin 2-12
PC5/MOSI1 Pin 2-12
PC6/SCK1 Pin 2-13
PC7/ $\overline{\text{SS1}}$ Pin 2-13
PC8/STD Pin 2-13
PC9/SRD Pin 2-14
PCC Bit 6-6
PCC Register 6-5
PCD Register 6-7
PCDDR Register 6-6
PCR0 10-11
PCR0 Register
 Reserved Bits 10-12
PCR1 Register 10-8
 Reserved Bits 10-11
Peripheral Data Bus (PGDB) 1-11
Peripheral Interrupts 1-16
Phase-Locked Loop (PLL) 10-5
PLL
 Architecture 10-3
 Changing Frequency 10-16
 Control Register 0 (PCR0) 10-11
 Control Register 1 (PCR1) 10-8
 Control Registers 10-7
 Enable Bit (PLLE) 10-8
 Lock 10-14
 Low-Power Operation 10-17
 Power Down Bit (PLLD) 10-8
 Programming Sequence 10-15
 Turning Off 10-16
 Turning Off When Not in Use 10-17
PLL Ground Pin (V_{SSPLL}) 2-6
PLL Power Pin (V_{DDPL}) 2-6
PLLD Bit 10-8
PLLE Bit 10-8
PM7-PM0 Bits 8-13
Port A Programming Model 4-5
Port B Data Direction Register (PBDDR) 5-6
Port B Data Register (PBD) 5-6
Port B GPIO Pins
 (PB0-PB7) 2-10
 (PB8-PB15) 2-10
Port B Interrupt Generation 5-8
Port B Interrupt Register (PBINT) 5-7
Port B Programming Examples 5-9
Port B Programming Model 5-5
Port C
 Configured for SPI 7-16
 Configured for SSI 8-36
 Control Bit (PCC) 6-6
 Control Register (PCC) 6-5

R

- Data Direction Bit (CDD) 6-6
- Data Direction Register (PCDDR) 6-6
- Data Register (PCD) 6-7
- GPIO 0 Pin (PC0/MISO0) 2-11
- GPIO 1 Pin (PC1/MOSI0) 2-11
- GPIO 10 Pin (PC10/STCK) 2-14
- GPIO 11 Pin (PC11/STFS) 2-14
- GPIO 12 Pin (PC12/SRCK) 2-14
- GPIO 13 Pin (PC13/SRFS) 2-15
- GPIO 14 Pin (PC14/TIO01) 2-15
- GPIO 15 Pin (PC15/TIO2) 2-16
- GPIO 2 Pin (PC2/SPCK0) 2-11
- GPIO 3 Pin (PC3/ $\overline{SS0}$) 2-12
- GPIO 4 Pin (PC4/MISO1) 2-12
- GPIO 5 Pin (PC5/MOSI1) 2-12
- GPIO 6 Pin (PC6/SCK1) 2-13
- GPIO 7 Pin (PC7/ $\overline{SS1}$) 2-13
- GPIO 8 Pin (PC8/STD) 2-13
- GPIO 9 Pin (PC9/SRD) 2-14
- Programming Example 6-7
- Programming Model 6-5
- Power Pins (V_{DD}) 2-6
- Prescale Modulus Select (PM7-PM0) Bits 8-13
- Prescaler 10-6
- Prescaler Clock
 - Turning Off 10-17
- Prescaler Divider Bits (PS2-PS0) 10-9
- Prescaler Range Bit (PSR) 8-12
- Program Address Bus (PAB) 1-10
- Program Controller 1-10
- Program Data Bus (PDB) 1-11
- Program Memory Map 3-6
- Program Memory Select Pin (\overline{PS}) 2-8
- Programmable I/O 1-14
- Programming Examples
 - Port B 5-9
- Programming Examples for SPI 7-18
- Programming Model
 - SPI 7-6
 - SSI 8-8
 - Timers 9-5
- \overline{PS} Pin 2-8
- PS2-PS0 Bits 10-9
- PSR Bit 8-12
- RDBF Bit 8-21
- RDF Bit 8-19
- RE Bit 8-16
- Read Enable Pin (\overline{RD}) 2-8
- Real Time Interrupt (RTI) Module 11-3
- Realtime Prescaler (RP) Bit 11-8
- Realtime Prescaler (RP) Bits 11-9
- Realtime Timer Enable Bit (RTE) 11-7
- Realtime Timer Interrupt Enable Bit (RTIE) 11-7
- Realtime Timer Interrupt Flag Bit (RTIF) 11-8
- Realtime/COP Divider (DV7-DV0) Bits 11-9
- Realtime/COP Divider Bits (DV7-DV0) 11-8
- Receive Buffer Enable (RBF) Bit 8-17
- Receive Data Buffer Full (RDBF) Bit 8-21
- Receive Data Register Full (RDF) Bit 8-19
- Receive Direction (RXD) Bit 8-17
- Receive Enable (RE) Bit 8-16
- Receive Frame Sync (RFS) Bit 8-21
- Receive Interrupt Enable (RIE) Bit 8-15
- Receive Overrun Error (ROE) Bit 8-20
- Reserved Bits
 - COPCNT Register 11-9
 - COPCTL Register 11-8
 - PCR0 Register 10-12
 - PCR1 Register 10-11
 - TCR Register 9-9
- \overline{RESET} Pin 2-10
- Reset Pin (RESET) 2-10
- Reset Vector Map 3-15
- RFS Bit 8-21
- RIE Bit 8-15
- ROE Bit 8-20
- RP Bit 11-8
- RP Bits 11-9
- RTE Bit 11-7
- RTI Timer
 - Programming 11-10
- RTIE Bit 11-7
- RTIF Bit 11-8
- Running Timer
 - in Wait Mode 9-20
 - in Stop Mode 9-20
- RXD Bit 8-17
- RXSR Register 8-11

R

- RBF Bit 8-17
- \overline{RD} Pin 2-8

S

- SAMPLE/PRELOAD Instruction 12-9
- SC2-SC0 Bits 11-9

- Scaler (SC2-SC0) Bits 11-9
- SCK0/PC2 Pin 7-12
- SCK1/PC6 Pin 7-13
- SCKP Bit 8-18
- SCR2 Register 8-14
- SCRRX Register 8-12
- SCRTX Register 8-12
- Serial Peripheral Interface (SPI) 1-14
- SHFD Bit 8-18
- Shift Direction (MSB/LSB Position) (SHFD)
 - Bit 8-18
- SPCR Register 7-7
 - Reserved Bits 7-10
- SPDR Register 7-11
- SPE Bit 7-9
- SPI
 - Architecture 7-5
 - Clock Phase Bit (CPH) 7-10
 - Clock Polarity Bit (CPL) 7-10
 - Clock Rate Select Bits (SPR2-0) 7-8
 - Control Register (SPCR) 7-7
 - Data and Control Pins 7-11
 - Data Register (SPDR) 7-11
 - Enable Bit (SPE) 7-9
 - Interrupt Complete Flag Bit (SPIF) 7-10
 - Interrupt Enable Bit (SPIE) 7-9
 - Master Mode Select Bit (MST) 7-10
 - Mode-Fault Error 7-14
 - Overrun 7-16
 - Programming 7-18
 - Programming Model 7-6
 - Status Register (SPSR) 7-10
 - System Errors 7-14
 - Wired-OR Mode Bit (WOM) 7-9
 - Write Collision Bit (WCOL) 7-11
 - Write-Collision Error 7-15
- SPI0
 - Master In/Slave Out Pin (PC0/MISO0) 2-11
 - Master Out/Slave In Pin (PC1/MOSI0) 2-11
 - Serial Clock Pin (PC2/SPCK0) 2-11
 - Slave Select Pin (PC3/ $\overline{SS0}$) 2-12
- SPI1
 - Master In/Slave Out Pin (PC4/MISO1) 2-12
 - Master Out/Slave In Pin (PC5/MOSI1) 2-12
 - Serial Clock Pin (PC6/SCK1) 2-13
 - Slave Select Pin (PC7/ $\overline{SS1}$) 2-13
- SPIE Bit 7-9
- SPIF Bit 7-10
- SPR2-0 Bits 7-8
- SPSR Register 7-10
 - Reserved Bits 7-11
- SR Register 3-17
- SRCK/PC12 Pin 8-25
- SRD/PC9 Pin 8-25
- SRFS/PC13 Pin 8-25
- SRX Register 8-12
- $\overline{SS0}$ /PC3 Pin 7-12
- $\overline{SS1}$ /PC7 Pin 7-13
- SSI
 - Architecture 8-4
 - Bit Clock 8-6
 - Clock Generation 8-6
 - Clocking 8-6
 - Control Register 2 (SCR2) 8-14
 - Data and Control Pins 8-22
 - Frame Clock 8-6
 - Frame Sync Generation 8-6
 - Frame Sync Generator 8-7
 - Gated Clock Operation 8-35
 - Network Mode 8-31
 - Receive 8-32
 - Transmit 8-31
 - Normal Mode 8-28
 - Receive 8-29
 - Transmit 8-28
 - Operating Modes 8-26
 - Programming Model 8-8
 - Receive Data Buffer Register 8-11
 - Receive Data Pin (PC9/SRD) 2-14
 - Receive Data Register (SRX) 8-12
 - Receive Shift Register (RXSR) 8-11
 - Reset and Initialization Procedure 8-35
 - Serial Receive Clock Pin (PC12/SRCK) 2-14
 - Serial Receive Frame Sync Pin
 - (PC13/SRFS) 2-15
 - Serial Transmit Clock Pin (PC10/STCK) 2-14
 - Serial Transmit Frame Sync Pin
 - (PC11/STFS) 2-14
 - Status Register (SSR) 8-19
 - Time Slot Register (STSR) 8-22
 - Transmit and Receive Control Registers
 - (SCRTX and SCRRX) 8-12
 - Transmit Data Buffer Register 8-10
 - Transmit Data Pin (PC8/STD) 2-13
 - Transmit Data Register (STX) 8-11
 - Transmit Shift Register (TXSR) 8-10
 - Word Clock 8-6
- SSI Enable (SSIEN) Bit 8-18
- SSIEN Bit 8-18
- SSR Register 8-19
 - Reserved Bits 8-22
- Status Register (SR) 3-17

T

- STCK/PC10 Pin 8-25
- STD/PC8 Pin 8-25
- STFS/PC11 Pin 8-25
- Stop Mode 10-12
 - Clocks Disabled 10-14
 - Clocks Enabled 10-13
 - Running a Timer in 9-20
 - Used with SPI 7-17
- STSR Register 8-22
- STX Register 8-11
- SXFC Pin 2-7
- SYN Bit 8-18
- Synchronous Mode (SYN) Bit 8-18
- Synchronous Serial Interface (SSI) 1-15

T

- TAP Controller 12-17
- TBF Bit 8-17
- TCK Pin 2-16, 12-6
- TCR Register
 - Reserved Bits 9-9
- TCR01 Register 9-6
- TCR2 Register 9-6
- TCT Register 9-9
- TDBE Bit 8-22
- TDE Bit 8-20
- TDI Pin 2-16, 12-6
- TDO Pin 12-6
- TE Bit 8-16, 9-6
- Test Access Port (TAP) 12-3
- Test Clock Input Pin (TCK) 2-16, 12-6
- Test Data Input Pin (TDI) 2-16, 12-6
- Test Data Output Pin (TDO) 12-6
- Test Mode Select Input Pin (TMS) 2-16, 12-6
- Test Reset/Debug Event Pin ($\overline{\text{TRST}}/\overline{\text{DE}}$) 12-6
- TFS Bit 8-21
- TIE Bit 8-15
- Timer
 - Control Register (TCR01) 9-6
 - Control Register (TCR2) 9-6
 - Count Register (TCT) 9-5
 - Interrupt Priorities 9-11
 - Preload Register (TPR) 9-5
 - Programming Model 9-5
 - Registers 9-5
 - Resolution 9-10
 - COP Timer 11-11
 - RTI Timer 11-11

- Timer 0 and Timer 1 Input/Output Pin (PC14/TIO01) 2-15
- Timer 2 Input/Output Pin (PC15/TIO2) 2-16
- Timer Architecture 9-5
- Timer Count Register (TCT) 9-9
- Timer Enable Bit (TE) 9-6
- Timer Module
 - Low Power Operation 9-19
 - Turning Off 9-19
- Timer Output Enable Bits (TO1-TO0) 9-7
- Timer Preload Register (TPR) 9-9
- Timers 0-2 9-3
- TMS Pin 2-16, 12-6
- TO1-TO0 Bits 9-7
- TPR Register 9-9
- Transmit Buffer Enable (TBF) Bit 8-17
- Transmit Data Buffer Empty (TDBE) Bit 8-22
- Transmit Data Register Empty (TDE) Bit 8-20
- Transmit Direction (TXD) Bit 8-18
- Transmit Enable (TE) Bit 8-16
- Transmit Frame Sync (TFS) Bit 8-21
- Transmit Interrupt Enable (TIE) Bit 8-15
- Transmit Underrun Error (TUE) Bit 8-20
- $\overline{\text{TRST}}/\overline{\text{DE}}$ Pin 12-6
- TUE Bit 8-20
- Turning Off Timer Not in Use 9-20
- TXD Bit 8-18
- TXSR Register 8-10

V

- V_{DD} Pins 2-6
- V_{DDPLL} Pin 2-6
- V_{SS} Pin 2-6
- V_{SSPLL} Pin 2-6

W

- Wait Mode 10-12
 - Running a Timer in 9-20
 - Used with SPI 7-17
- Wait State P Memory Bits (WSPM) 4-6
- Wait State X-Data Memory Bits (WSX) 4-6
- WCOL Bit 7-11
- Wired-OR Mode Bit (WOM) 7-9
- WL1-WL0 Bits 8-13
- WOM Bit 7-9
- Word Length Control Bits (WL1-WL0) 8-13
- WR Pin 2-8
- Write Collision Bit (WCOL) 7-11

Write Enable Pin (WR) 2-8

WSPM Bits 4-6

WSX Bits 4-6

X

X Address Bus One (XAB1) 1-10

X Address Bus Two (XAB2) 1-10

X Data Bus Two (XDB2) 1-11

XRAM Execution Mode

 Description 3-12

 Entering 3-14

 Exiting 3-14

 Interrupts 3-14

 Restrictions 3-15

XTAL Pin 2-6

Y

YD9-YD0 Bits 10-11

