DSP56374
24-Bit Digital Signal Processor

User Guide

Document Number: DSP56374UG
Rev. 1.2
07/2007

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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

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

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

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

Table of Contents

Paragraph Number		Page Number
	Preface i	
	Chapter 1 DSP56374 Overview	
1.1	Introduction	1-1
1.2	DSP56300 Core Description	1-2
1.3	DSP56374 Audio Processor Architecture	1-3
1.4	DSP56300 Core Functional Blocks	1-3
1.4.1	Data ALU	1-3
1.4.1.1	Data ALU Registers	1-3
1.4.1.2	Multiplier-Accumulator (MAC)	1-3
1.4.2	Address Generation Unit (AGU)	1-4
1.4.3	Program Control Unit (PCU)	1-4
1.4.4	Internal Buses	1-4
1.4.5	Direct Memory Access (DMA)	1-5
1.4.6	PLL-based Clock Oscillator	1-5
1.4.7	On-Chip Memory	1-5
1.4.8	Off-Chip Memory Expansion	1-5
1.4.9	Power Requirements	1-5
1.5	Peripheral Overview	1-6
1.5.1	General Purpose Input/Output (GPIO)	1-6
1.5.2	Triple Timer (TEC)	1-6
1.5.3	Enhanced Serial Audio Interface (ESAI)	1-7
1.5.4	Enhanced Serial Audio Interface 1 (ESAI_1)	1-7
1.5.5	Serial Host Interface (SHI)	1-7
1.5.6	Watchdog timer (WDT)	1-7
	Chapter 2 Signal/Connection Descriptions	
2.1	Signal Groupings	2-1
2.2	Power	2-1
2.3	Ground	2-3
2.4	SCAN	2-4
2.5	Clock and PLL	2-4
2.6	Interrupt and Mode Control	2-4
2.7	Serial Host Interface	2-6
2.8	Enhanced Serial Audio Interface	2-8
2.9	Enhanced Serial Audio Interface_1	2-12
2.10	Dedicated GPIO - Port G	2-16
2.11	Timer	2-18
2.12	JTAG/OnCE Interface	2-19
	Chapter 3 Memory Configuration	
3.1	Data and Program Memory Maps	3-1
3.1.1	Reserved Memory Spaces	3-5
3.1.2	Bootstrap CODE	3-5
3.1.3	Dynamic Memory Configuration Switching	3-5

Paragraph Number		Page Number
3.1.4	External Memory Support	3-5
3.1.5	DMA and Memory	3-5
3.1.6	Memory BLOCKS	3-6
3.2	Memory Patch Module	3-6
3.3	Internal I/O Memory Map	3-7

Chapter 4
Core Configuration

4.1	Introduction	4-1
4.2	Operating Mode Register (OMR)	4-1
4.2.1	RESERVED - Bits 4, 5, 10 - 15 and 23	4-1
4.3	Operating Modes	4-1
4.4	Interrupt Priority Registers	4-3
4.5	DMA Request Sources	4-9
4.6	PLL Initialization	4-10
4.6.1	PLL Pre-Divider Factor (PD0-PD4)	4-10
4.6.2	PLL Multiplication Factor (MF0-MF7)	4-10
4.6.3	PLL Feedback Multiplier (OD1)	4-10
4.6.4	PLL Output Divide Factor (OD0-OD1)	4-10
4.6.5	PLL Divider Factor (DF0-DF2)	4-10
4.6.6	PLL LOCK MUX (PLKM)	4-10
4.7	Device Identification (ID) Register	4-10
4.8	JTAG Identification (ID) Register	4-11

Chapter 5
PLL and Clock generator

5.1	Introduction	5-1
5.2	PLL and Clock Signals	5-1
5.3	PLL Block	5-1
5.3.1	Frequency Predivider	5-2
5.3.2	Phase Detector and Charge Pump Loop Filter	5-2
5.3.3	Voltage Controlled Oscillator (VCO)	5-2
5.3.4	PLL DividerS	5-2
5.3.5	PLL Multiplication Factor (MF)	5-3
5.4	PLL Operation	5-3
5.4.1	EXTAL Clock Input Division	5-3
5.4.2	PLL Frequency Multiplication	5-3
5.4.3	PLL Output Frequency (PLL Out)	5-4
5.5	Clock Generator	5-6
5.5.1	Low-Power Divider (LPD)	5-6
5.6	Operating Frequency (Fosc)	5-6
5.7	PLL Programming Model	5-7
5.8	PLL Initialization Procedure	5-10
5.9	PLL Programming Examples	5-11

Chapter 6
General Purpose Input/Output

6.1	Introduction	6-1
6.2	Programming Model	6-1

Paragraph Number		Page Number
6.2.1	Port C and E Signals and Registers	6-1
6.2.2	Port G Signals and Registers	6-1
6.2.2.1	Port G Control Register (PCRG)	6-1
6.2.2.2	Port G Direction Register (PRRG)	6-1
6.2.2.3	Port G Data register (PDRG)	6-2
6.2.2.4	ESAI/EXTAL clocking control	6-2
6.2.3	Port H Signals and Registers	6-3
6.2.3.1	Port H Control Register (PCRH)	6-3
6.2.3.2	Port H Direction Register (PRRH)	6-3
6.2.3.3	Port H Data register (PDRH)	6-4
6.2.4	Timer/Event Counter Signals	6-4

Chapter 7
Serial Host Interface

7.1	Introduction	7-1
7.2	Serial Host Interface Internal Architecture	7-1
7.3	SHI Clock Generator	7-2
7.4	Serial Host Interface Programming Model	7-2
7.4.1	SHI Input/Output Shift Register (IOSR)—Host Side	7-4
7.4.2	SHI Host Transmit Data Register (HTX)—DSP Side	7-4
7.4.3	SHI Host Receive Data FIFO (HRX)—DSP Side	7-5
7.4.4	SHI Slave Address Register (HSAR)—DSP Side	7-5
7.4.4.1	HSAR Reserved Bits—Bits 19, 17– 0	7-5
7.4.4.2	HSAR I^2C Slave Address (HA[6:3], HA1)—Bits 23–20,18	7-5
7.4.5	SHI Clock Control Register (HCKR)—DSP Side	7-5
7.4.5.1	Clock Phase and Polarity (CPHA and CPOL)—Bits 1–0	7-5
7.4.5.2	HCKR Prescaler Rate Select (HRS)—Bit 2	7-6
7.4.5.3	HCKR Divider Modulus Select (HDM[7:0])—Bits 10–3	7-7
7.4.5.4	HCKR Filter Mode (HFM[1:0]) — Bits 13–12	7-7
7.4.5.5	HCKR Reserved Bits—Bits 23–14, 11	7-7
7.4.6	SHI Control/Status Register (HCSR)—DSP Side	7-7
7.4.6.1	HCSR Host Enable (HEN)—Bit 0	7-7
7.4.6.1.1	SHI Individual Reset	7-8
7.4.6.2	HCSR I^2C/SPI Selection (HI2C)—Bit 1	7-8
7.4.6.3	HCSR Serial Host Interface Mode (HM[1:0])—Bits 3–2	7-8
7.4.6.4	HCSR I^2C Clock Freeze (HCKFR)—Bit 4	7-8
7.4.6.5	HCSR FIFO-Enable Control (HFIFO)—Bit 5	7-8
7.4.6.6	HCSR Master Mode (HMST)—Bit 6	7-8
7.4.6.7	HCSR Host-Request Enable (HRQE[1:0])—Bits 8–7	7-9
7.4.6.8	HCSR Idle (HIDLE)—Bit 9	7-9
7.4.6.9	HCSR Bus-Error Interrupt Enable (HBIE)—Bit 10	7-9
7.4.6.10	HCSR Transmit-Interrupt Enable (HTIE)—Bit 11	7-9
7.4.6.11	HCSR Receive Interrupt Enable (HRIE[1:0])—Bits 13–12	7-10
7.4.6.12	HCSR Host Transmit Underrun Error (HTUE)—Bit 14	7-10
7.4.6.13	HCSR Host Transmit Data Empty (HTDE)—Bit 15	7-10
7.4.6.14	HCSR Reserved Bits—Bits 23, 18 and 16	7-10
7.4.6.15	Host Receive FIFO Not Empty (HRNE)—Bit 17	7-10
7.4.6.16	Host Receive FIFO Full (HRFF)—Bit 19	7-10
7.4.6.17	Host Receive Overrun Error (HROE)—Bit 20	7-11

Paragraph Number		Page Number
7.4.6.18	Host Bus Error (HBER)—Bit 21	7-11
7.4.6.19	HCSR Host Busy (HBUSY)—Bit 22	7-11
7.5	Characteristics Of The SPI Bus	7-11
7.6	Characteristics Of The I^2C Bus	7-11
7.6.1	Overview	7-11
7.6.2	I^2C Data Transfer Formats	7-13
7.7	SHI Programming Considerations	7-13
7.7.1	SPI Slave Mode	7-13
7.7.2	SPI Master Mode	7-14
7.7.3	I^2C Slave Mode	7-14
7.7.3.1	Receive Data in I^2C Slave Mode	7-15
7.7.3.2	Transmit Data In I^2C Slave Mode	7-15
7.7.4	I^2C Master Mode	7-15
7.7.4.1	Receive Data in I^2C Master Mode	7-16
7.7.4.2	Transmit Data In I^2C Master Mode	7-16
7.7.5	SHI Operation During DSP Stop	7-17
7.7.6	GPIO- HREQ Signal and Registers	7-17

Chapter 8
Enhanced Serial Audio Interface (ESAI)

8.1	Introduction	8-1
8.2	ESAI Data and Control Pins	8-2
8.2.1	Serial Transmit 0 Data Pin (SDO0)	8-3
8.2.2	Serial Transmit 1 Data Pin (SDO1)	8-3
8.2.3	Serial Transmit 2/Receive 3 Data Pin (SDO2/SDI3)	8-3
8.2.4	Serial Transmit 3/Receive 2 Data Pin (SDO3/SDI2)	8-3
8.2.5	Serial Transmit 4/Receive 1 Data Pin (SDO4/SDI1)	8-3
8.2.6	Serial Transmit 5/Receive 0 Data Pin (SDO5/SDI0)	8-3
8.2.7	Receiver Serial Clock (SCKR)	8-4
8.2.8	Transmitter Serial Clock (SCKT)	8-4
8.2.9	Frame Sync for Receiver (FSR)	8-5
8.2.10	Frame Sync for Transmitter (FST)	8-6
8.2.11	High Frequency Clock for Transmitter (HCKT)	8-6
8.2.12	High Frequency Clock for Receiver (HCKR)	8-6
8.3	ESAI Programming Model	8-6
8.3.1	ESAI Transmitter Clock Control Register (TCCR)	8-6
8.3.1.1	TCCR Transmit Prescale Modulus Select (TPM7–TPM0) - Bits 7–0	8-7
8.3.1.2	TCCR Transmit Prescaler Range (TPSR) - Bit 8	8-8
8.3.1.3	TCCR Tx Frame Rate Divider Control (TDC4–TDC0) - Bits 13–9	8-8
8.3.1.4	TCCR Tx High Frequency Clock Divider (TFP3-TFP0) - Bits 17–14	8-8
8.3.1.5	TCCR Transmit Clock Polarity (TCKP) - Bit 18	8-9
8.3.1.6	TCCR Transmit Frame Sync Polarity (TFSP) - Bit 19	8-9
8.3.1.7	TCCR Transmit High Frequency Clock Polarity (THCKP) - Bit 20	8-9
8.3.1.8	TCCR Transmit Clock Source Direction (TCKD) - Bit 21	8-9
8.3.1.9	TCCR Transmit Frame Sync Signal Direction (TFSD) - Bit 22	8-9
8.3.1.10	TCCR Transmit High Frequency Clock Direction (THCKD) - Bit 23	8-9
8.3.2	ESAI Transmit Control Register (TCR)	8-9
8.3.2.1	TCR ESAI Transmit 0 Enable (TE0) - Bit 0	8-10
8.3.2.2	TCR ESAI Transmit 1 Enable (TE1) - Bit 1	8-10

Paragraph Number		Page Number
8.3.2.3	TCR ESAI Transmit 2 Enable (TE2) - Bit 2	8-10
8.3.2.4	TCR ESAI Transmit 3 Enable (TE3) - Bit 3	8-11
8.3.2.5	TCR ESAI Transmit 4 Enable (TE4) - Bit 4	8-11
8.3.2.6	TCR ESAI Transmit 5 Enable (TE5) - Bit 5	8-11
8.3.2.7	TCR Transmit Shift Direction (TSHFD) - Bit 6	8-11
8.3.2.8	TCR Transmit Word Alignment Control (TWA) - Bit 7	8-11
8.3.2.9	TCR Transmit Network Mode Control (TMOD1-TMOD0) - Bits 9-8	8-12
8.3.2.10	TCR Tx Slot and Word Length Select (TSWS4-TSWS0) - Bits 14-10	8-13
8.3.2.11	TCR Transmit Frame Sync Length (TFSL) - Bit 15	8-14
8.3.2.12	TCR Transmit Frame Sync Relative Timing (TFSR) - Bit 16	8-15
8.3.2.13	TCR Transmit Zero Padding Control (PADC) - Bit 17	8-16
8.3.2.14	TCR Reserved Bit - Bits 18	8-16
8.3.2.15	TCR Transmit Section Personal Reset (TPR) - Bit 19	8-16
8.3.2.16	TCR Transmit Exception Interrupt Enable (TEIE) - Bit 20	8-16
8.3.2.17	TCR Transmit Even Slot Data Interrupt Enable (TEDIE) - Bit 21	8-16
8.3.2.18	TCR Transmit Interrupt Enable (TIE) - Bit 22	8-16
8.3.2.19	TCR Transmit Last Slot Interrupt Enable (TLIE) - Bit 23	8-16
8.3.3	ESAI Receive Clock Control Register (RCCR)	8-16
8.3.3.1	RCCR Receiver Prescale Modulus Select (RPM7–RPM0) - Bits 7–0	8-17
8.3.3.2	RCCR Receiver Prescaler Range (RPSR) - Bit 8	8-17
8.3.3.3	RCCR Rx Frame Rate Divider Control (RDC4–RDC0) - Bits 13–9	8-17
8.3.3.4	RCCR Rx High Frequency Clock Divider (RFP3-RFP0) - Bits 17-14	8-17
8.3.3.5	RCCR Receiver Clock Polarity (RCKP) - Bit 18	8-18
8.3.3.6	RCCR Receiver Frame Sync Polarity (RFSP) - Bit 19	8-18
8.3.3.7	RCCR Receiver High Frequency Clock Polarity (RHCKP) - Bit 20	8-18
8.3.3.8	RCCR Receiver Clock Source Direction (RCKD) - Bit 21	8-18
8.3.3.9	RCCR Receiver Frame Sync Signal Direction (RFSD) - Bit 22	8-19
8.3.3.10	RCCR Receiver High Frequency Clock Direction (RHCKD) - Bit 23	8-19
8.3.4	ESAI Receive Control Register (RCR)	8-19
8.3.4.1	RCR ESAI Receiver 0 Enable (RE0) - Bit 0	8-20
8.3.4.2	RCR ESAI Receiver 1 Enable (RE1) - Bit 1	8-20
8.3.4.3	RCR ESAI Receiver 2 Enable (RE2) - Bit 2	8-20
8.3.4.4	RCR ESAI Receiver 3 Enable (RE3) - Bit 3	8-20
8.3.4.5	RCR Reserved Bits - Bits 5-4, 18-17	8-20
8.3.4.6	RCR Receiver Shift Direction (RSHFD) - Bit 6	8-20
8.3.4.7	RCR Receiver Word Alignment Control (RWA) - Bit 7	8-21
8.3.4.8	RCR Receiver Network Mode Control (RMOD1-RMOD0) - Bits 9-8	8-21
8.3.4.9	RCR Receiver Slot and Word Select (RSWS4-RSWS0) - Bits 14-10	8-21
8.3.4.10	RCR Receiver Frame Sync Length (RFSL) - Bit 15	8-22
8.3.4.11	RCR Receiver Frame Sync Relative Timing (RFSR) - Bit 16	8-22
8.3.4.12	RCR Receiver Section Personal Reset (RPR) - Bit 19	8-22
8.3.4.13	RCR Receive Exception Interrupt Enable (REIE) - Bit 20	8-23
8.3.4.14	RCR Receive Even Slot Data Interrupt Enable (REDIE) - Bit 21	8-23
8.3.4.15	RCR Receive Interrupt Enable (RIE) - Bit 22	8-23
8.3.4.16	RCR Receive Last Slot Interrupt Enable (RLIE) - Bit 23	8-23
8.3.5	ESAI Common Control Register (SAICR)	8-23
8.3.5.1	SAICR Serial Output Flag 0 (OF0) - Bit 0	8-23
8.3.5.2	SAICR Serial Output Flag 1 (OF1) - Bit 1	8-24
8.3.5.3	SAICR Serial Output Flag 2 (OF2) - Bit 2	8-24
8.3.5.4	SAICR Reserved Bits - Bits 5-3, 23-9	8-24

Paragraph Number		Page Number
8.3.5.5	SAICR Synchronous Mode Selection (SYN) - Bit 6	8-24
8.3.5.6	SAICR Transmit External Buffer Enable (TEBE) - Bit 7	8-24
8.3.5.7	SAICR Alignment Control (ALC) - Bit 8	8-24
8.3.6	ESAI Status Register (SAISR)	8-25
8.3.6.1	SAISR Serial Input Flag 0 (IF0) - Bit 0	8-26
8.3.6.2	SAISR Serial Input Flag 1 (IF1) - Bit 1	8-26
8.3.6.3	SAISR Serial Input Flag 2 (IF2) - Bit 2	8-26
8.3.6.4	SAISR Reserved Bits - Bits 5-3, 12-11, 23-18	8-26
8.3.6.5	SAISR Receive Frame Sync Flag (RFS) - Bit 6	8-26
8.3.6.6	SAISR Receiver Overrun Error Flag (ROE) - Bit 7	8-26
8.3.6.7	SAISR Receive Data Register Full (RDF) - Bit 8	8-27
8.3.6.8	SAISR Receive Even-Data Register Full (REDF) - Bit 9	8-27
8.3.6.9	SAISR Receive Odd-Data Register Full (RODF) - Bit 10	8-27
8.3.6.10	SAISR Transmit Frame Sync Flag (TFS) - Bit 13	8-27
8.3.6.11	SAISR Transmit Underrun Error Flag (TUE) - Bit 14	8-27
8.3.6.12	SAISR Transmit Data Register Empty (TDE) - Bit 15	8-27
8.3.6.14	SAISR Transmit Even-Data Register Empty (TEDE) - Bit 16	8-27
8.3.6.13	SAISR Transmit Odd-Data Register Empty (TODE) - Bit 17	8-28
8.3.7	ESAI Receive Shift Registers	8-29
8.3.8	ESAI Receive Data Registers (RX3, RX2, RX1, RX0)	8-30
8.3.9	ESAI Transmit Shift Registers	8-30
8.3.10	ESAI Transmit Data Registers (TX5, TX4, TX3, TX2,TX1,TX0)	8-30
8.3.11	ESAI Time Slot Register (TSR)	8-30
8.3.12	Transmit Slot Mask Registers (TSMA, TSMB)	8-30
8.3.13	Receive Slot Mask Registers (RSMA, RSMB)	8-31
8.4	Operating Modes	8-32
8.4.1	ESAI After Reset	8-32
8.4.2	ESAI Initialization	8-32
8.4.3	ESAI Interrupt Requests	8-33
8.4.4	Operating Modes – Normal, Network and On-Demand	8-33
8.4.4.1	Normal/Network/On-Demand Mode Selection	8-33
8.4.4.2	Synchronous/Asynchronous Operating Modes	8-34
8.4.4.3	Frame Sync Selection 3	8-34
8.4.4.4	Shift Direction Selection	8-34
8.4.5	Serial I/O Flags	8-34
8.5	GPIO - Pins and Registers	8-35
8.5.1	Port C (ESAI) GPIO - Pins and Registers	8-35
8.5.1.1	Port C Control Register (PCRC)	8-35
8.5.1.2	Port C Direction Register (PRRC)	8-35
8.5.1.3	Port C Data register (PDRC)	8-36
8.5.2	Port E (ESAI_1) GPIO - Pins and Registers	8-36
8.5.2.1	Port E Control Register (PCRE)	8-37
8.5.2.2	Port E Direction Register (PRRE)	8-37
8.5.2.3	Port E Data register (PDRE)	8-37
8.6	ESAI Initialization Examples	8-38
8.6.1	Initializing the ESAI Using Individual Reset	8-38
8.6.2	Initializing Just the ESAI Transmitter Section	8-38
8.6.3	Initializing Just the ESAI Receiver Section	8-38

Paragraph Number | Page Number

Chapter 9
Triple Timer Module

9.1 Overview 9-1
9.1.1 Triple Timer Module Block Diagram 9-1
9.1.2 Individual Timer Block Diagram 9-1
9.2 Operation 9-2
9.2.1 Timer After Reset 9-2
9.2.2 Timer Initialization 9-2
9.2.3 Timer Exceptions 9-3
9.3 Operating Modes 9-3
9.3.1 Triple Timer Modes 9-3
9.3.1.1 Timer GPIO (Mode 0) 9-4
9.3.1.2 Timer Pulse (Mode 1) 9-5
9.3.1.3 Timer Toggle (Mode 2) 9-7
9.3.1.4 Timer Event Counter (Mode 3) 9-9
9.3.2 Signal Measurement Modes 9-10
9.3.2.1 Measurement Input Width (Mode 4) 9-10
9.3.2.2 Measurement Input Period (Mode 5) 9-12
9.3.2.3 Measurement Capture (Mode 6) 9-13
9.3.3 Pulse Width Modulation (PWM, Mode 7) 9-14
9.3.4 Watchdog Modes 9-16
9.3.4.1 Watchdog Pulse (Mode 9) 9-16
9.3.4.2 Watchdog Toggle (Mode 10) 9-17
9.3.4.3 Reserved Modes 9-18
9.3.5 Special Cases 9-18
9.3.6 DMA Trigger 9-18
9.4 Triple Timer Module Programming Model 9-18
9.4.1 Prescaler Counter 9-18
9.4.2 Timer Prescaler Load Register (TPLR) 9-19
9.4.3 Timer Prescaler Count Register (TPCR) 9-20
9.4.4 Timer Control/Status Register (TCSR) 9-21
9.4.5 Timer Load Register (TLR) 9-25
9.4.6 Timer Compare Register (TCPR) 9-25
9.4.7 Timer Count Register (TCR) 9-25

Chapter 10
Watchdog Timer Module

10.1 Introduction 10-1
10.2 WDT Pin 10-1
10.3 WDT Operation 10-1
10.4 Description of Registers 10-2
10.4.1 Watchdog Control Register (WCR) 10-2
10.4.2 Watchdog Counter & WCNTR Register 10-2
10.4.3 Watchdog Modulus Register (WMR) 10-3
10.4.4 Watchdog Service Register (WSR) 10-3
10.5 Operation in Different Modes 10-3
10.5.1 WAIT Mode 10-3
10.5.2 DEBUG Mode 10-3
10.5.3 STOP MODE 10-3

Paragraph Number		Page Number
	Appendix A Bootstrap Source Code	
A.1	DSP56374 Bootstrap Program	A-1
A.2	Using The Serial EEPROM Boot Mode	A-5
	Appendix B Equates	
	Appendix C Programmer's Reference	
C.1	Introduction	C-1
C.1.1	Peripheral Addresses	C-1
C.1.2	Interrupt Addresses	C-1
C.1.3	Interrupt Priorities	C-1
C.1.4	Programming Sheets	C-1
C.1.5	Internal I/O Memory Map	C-1
C.1.6	Interrupt Vector Addresses	C-7
C.2	Interrupt Source Priorities (within an IPL)	C-10
C.3	Programming Sheets	C-11
	Appendix D BSDL	
D.1	52-pin BSDL	D-1
D.2	80-pin BSDL	D-6

List of Figures

Figure Number		Page Number
1-1	DSP56374 Block Diagram	1-1
2-1	80-pin Vdd Connections	2-2
2-2	52-pin Vdd Connections	2-3
3-1	Default Memory Map (MS 0)	3-2
3-2	Memory Map (MS 1, MSW(1-0) 11)	3-2
3-3	Memory Map (MS 1, MSW(1-0) 10)	3-3
3-4	Memory Map (MS 1, MSW(1-0) 01)	3-4
3-5	Memory Map (MS 1, MSW(1-0) 00)	3-4
4-1	Interrupt Priority Register P	4-4
4-2	Interrupt Priority Register C	4-4
4-3	PCTL Register	4-10
5-1	PLL Clock Generator Block Diagram	5-1
5-2	PLL Block Diagram	5-2
5-3	PLL Loop with One Divider when OD1=0 (FM = 2)	5-4
5-4	PLL Loop with Two Dividers when OD1=1 (FM = 4)	5-4
5-5	PLL Out = VCO Out/2 [OD1 = 0, OD0 = 1]	5-5
5-6	PLL Out = VCO Out/2 [OD1 = 1, OD0 = 0]	5-5
5-7	PLL Out = VCO Out/4 [OD1 = 1, OD0 = 1]	5-6
5-8	CLKGEN Block Diagram	5-6
5-9	PLL Control (PCTL) Register	5-7
6-1	PCRG Register	6-2
6-2	PRRG Register	6-2
6-3	PDRG Register	6-2
6-4	PCRH Register	6-4
6-5	PRRH Register	6-4
6-6	PDRH Register	6-4
7-1	Serial Host Interface Block Diagram	7-2
7-2	SHI Clock Generator	7-2
7-3	SHI Programming Model—Host Side	7-3
7-4	SHI Programming Model—DSP Side	7-3
7-5	SHI I/O Shift Register (IOSR)	7-4
7-6	SPI Data-To-Clock Timing Diagram	7-6
7-7	I^2C Bit Transfer	7-12
7-8	I^2C Start and Stop Events	7-12
7-9	Acknowledgment on the I^2C Bus	7-12
7-10	I^2C Bus Protocol For Host Write Cycle	7-13
7-11	I^2C Bus Protocol For Host Read Cycle	7-13
8-1	ESAI Block Diagram	8-2
8-2	TCCR Register	8-6
8-3	ESAI Clock Generator Functional Block Diagram	8-7
8-4	ESAI Frame Sync Generator Functional Block Diagram	8-8
8-5	TCR Register	8-10
8-6	Normal and Network Operation	8-13
8-7	Frame Length Selection	8-15
8-8	RCCR Register	8-17
8-9	RCR Register	8-20
8-10	SAICR Register	8-23
8-11	SAICR SYN Bit Operation	8-25

Figure Number		Page Number
8-12	SAISR Register	8-26
8-13	ESAI Data Path Programming Model ([R/T]SHFD=0)	8-28
8-14	ESAI Data Path Programming Model ([R/T]SHFD=1)	8-29
8-15	TSMA Register	8-30
8-16	RSMA Register	8-31
8-17	TSMB Register	8-31
8-18	RSMB Register	8-32
8-19	PCRC Register	8-36
8-20	PRRC Register	8-36
8-21	PDRC Register	8-36
8-22	PCRE Register	8-37
8-23	PRRE Register	8-37
8-24	PDRE Register	8-38
9-1	Triple Timer Module Block Diagram	9-1
9-2	Timer Module Block Diagram	9-2
9-3	Timer Mode (TRM = 1)	9-4
9-4	Timer Mode (TRM = 0)	9-5
9-5	Pulse Mode (TRM = 1)	9-6
9-6	Pulse Mode (TRM = 0)	9-7
9-7	Toggle Mode, TRM = 1	9-8
9-8	Toggle Mode, TRM = 0	9-8
9-9	Event Counter Mode, TRM = 1	9-9
9-10	Event Counter Mode, TRM = 0	9-10
9-11	Pulse Width Measurement Mode, TRM = 1	9-11
9-12	Pulse Width Measurement Mode, TRM = 0	9-11
9-13	Period Measurement Mode, TRM = 1	9-12
9-14	Period Measurement Mode, TRM = 0	9-13
9-15	Capture Measurement Mode, TRM = 0	9-14
9-16	Pulse Width Modulation Toggle Mode, TRM = 1	9-15
9-17	Pulse Width Modulation Toggle Mode, TRM = 0	9-16
9-18	Watchdog Pulse Mode	9-17
9-19	Watchdog Toggle Mode	9-18
9-20	Timer Module Programmer's Model	9-19
9-21	Timer Prescaler Count Register (TPCR)	9-20
10-1	Watchdog Timer Block Diagram	10-2
C-1	Status Register (SR)	C-12
C-2	Operating Mode Register (OMR)	C-13
C-3	Interrupt Priority Register–Core (IPR–C)	C-14
C-4	Interrupt Priority Register – Peripherals (IPR–P)	C-15
C-5	Phase Lock Loop Control Register (PCTL)	C-16
C-6	SHI Slave Address and Clock Control Registers	C-17
C-7	SHI Host Control/Status Register	C-18
C-8	ESAI Transmit Clock Control Register	C-19
C-9	ESAI Transmit Control Register	C-20
C-10	ESAI Receive Clock Control Register	C-21
C-11	ESAI Receive Control Register	C-22
C-12	ESAI Common Control Register	C-23
C-13	ESAI Status Register	C-24
C-14	ESAI_1 Transmit Clock Control Register	C-25
C-15	ESAI_1 Transmit Control Register	C-26

Figure Number		Page Number
C-16	ESAI_1 Receive Clock Control Register	C-27
C-17	ESAI_1 Receive Control Register	C-28
C-18	ESAI_1 Common Control Register	C-29
C-19	ESAI_1 Status Register	C-30
C-20	Timer Prescaler Load and Prescaler Count Registers (TPLR, TPCR)	C-31
C-21	Timer Control/Status Register	C-32
C-22	Timer Load, Compare and Count Registers	C-33
C-23	GPIO Port C	C-35
C-24	GPIO Port E	C-36
C-25	GPIO Port G	C-37

Notes

List of Tables

Table Number		Page Number
1-1	DSP56374 Memory Switch Configurations	1-2
2-1	DSP56374 Functional Signal Groupings	2-1
2-2	Power Inputs	2-1
2-3	Grounds	2-3
2-4	SCAN signals	2-4
2-5	Clock and PLL Signals	2-4
2-6	Interrupt and Mode Control	2-4
2-7	Serial Host Interface Signals	2-6
2-8	Enhanced Serial Audio Interface Signals	2-8
2-9	Enhanced Serial Audio Interface_1 Signals	2-12
2-10	Dedicated GPIO - Port G Signals	2-16
2-11	Timer Signal	2-18
2-12	JTAG/OnCE Interface	2-19
3-1	Internal Memory Configuration	3-1
3-2	Internal Memory Locations	3-1
3-3	Internal Memory Locations	3-2
3-4	Internal Memory Locations	3-3
3-5	Internal Memory Locations	3-3
3-6	Internal Memory Locations	3-4
3-7	Internal Memory Configurations	3-5
3-8	Internal I/O Memory Map (X Memory)	3-7
3-9	Internal I/O Memory Map (Y Memory)	3-10
4-1	Operating Mode Register (OMR)	4-1
4-2	DSP56374 Operating Modes	4-2
4-3	DSP56374 Mode Descriptions	4-2
4-4	Interrupt Priority Level Bits	4-3
4-5	Interrupt Sources Priorities Within an IPL	4-4
4-6	DSP56374 Interrupt Vectors	4-6
4-7	DMA Request Sources	4-9
4-8	Identification Register Configuration	4-10
4-9	JTAG Identification Register Configuration	4-11
5-1	Feedback Multiplier (FM); FM = 2(1 + OD1)	5-2
5-2	Output Divide Factor (OD)	5-3
5-3	Output Divide Factor (OD)	5-8
5-4	PLL Control (PCTL) Register Bit Definitions	5-8
5-5	PLL Programming Examples	5-11
6-1	PCRG and PRRG Bits Functionality	6-1
6-2	PCRH and PRRH Bits Functionality	6-3
7-1	SHI Interrupt Vectors	7-4
7-2	SHI Internal Interrupt Priorities	7-4
7-3	SHI Noise Reduction Filter Mode	7-7
7-4	SHI Data Size	7-8
7-5	HREQ Function In SPI Slave Mode	7-9
7-6	HCSR Receive Interrupt Enable Bits	7-10
8-1	Receiver Clock Sources (asynchronous mode only)	8-4
8-2	Transmitter Clock Sources	8-5
8-3	Transmitter High Frequency Clock Divider	8-9
8-4	Transmit Network Mode Selection	8-12
8-5	ESAI Transmit Slot and Word Length Selection	8-14
8-6	Receiver High Frequency Clock Divider	8-18
8-7	SCKR Pin Definition Table	8-18
8-8	FSR Pin Definition Table	8-19

Table Number		Page Number
8-9	HCKR Pin Definition Table	8-19
8-10	ESAI Receive Network Mode Selection	8-21
8-11	ESAI Receive Slot and Word Length Selection	8-21
8-12	PCRC and PRRC Bits Functionality	8-35
8-13	PCRE and PRRE Bits Functionality	8-37
9-1	Timer Prescaler Load Register (TPLR) Bit Definitions	9-20
9-2	Timer Prescaler Count Register (TPCR) Bit Definitions	9--20
9-3	Timer Control/Status Register (TCSR) Bit Definitions	9--21
9-4	Inverter (INV) Bit Operation	9-24
C-1	Internal I/O Memory Map (X Memory)	C-1
C-2	Internal I/O Memory Map (Y Memory)	C-4
C-3	DSP56374 Interrupt Vectors	C-7
C-4	Interrupt Sources Priorities Within an IPL	C-10

Preface

This manual describes the DSP56374 24-bit digital signal processor (DSP), its memory, operating modes and peripheral modules. The DSP56374 is a member of the DSP56300 family of programmable CMOS DSPs. Changes in core functionality specific to the DSP56374 are also described in this manual.

The SCF5250 is designed to support a multitude of digital signal processing applications that require a lot of horsepower in a small package. While generic in its signal-processing capabilities, the DSP56374 includes various built-in audio processing features designed to meet the needs of both consumer and automotive audio applications. The DSP56374 provides a wealth of audio-processing functions, including a basic operating system, various equalization algorithms, compression, signal generator, tone control, fade/balance, level meter/spectrum analyzer, and many more. The DSP56374 also supports various matrix decoders and sound-field processing algorithms. The SCF5250 uses the high performance, single-clock-per-cycle DSP56300 core family of programmable CMOS digital signal processors (DSPs) combined with the audio signal processing capability of the Freescale (formerly Motorola) Symphony™ DSP family. This design provides a two-fold performance increase over Freescale's popular DSP56000 family of DSPs while retaining code compatibility. Significant architectural enhancements include a barrel shifter, 24-bit addressing, and direct memory access (DMA).

This manual is intended to be used with the following publications:

- The *DSP56300 Family Manual (DSP56300FM/AD),* which describes the CPU, core programming models and instruction set details.
- The *DSP56374 Technical Data Sheet (DSP56374/D)*, which provides electrical specifications, timing, pinout and packaging descriptions of the DSP56374.

These documents, as well as Freescale's DSP development tools, can be obtained through a local Freescale Semiconductor Sales Office or authorized distributor.

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

This manual contains the following sections and appendices.

Section 1—DSP56374 Overview

- Provides a brief description of the DSP56374, including a features list and block diagram. Lists related documentation needed to use this chip and describes the organization of this manual.

Section 2—Signal/Connection Descriptions

- Describes the signals on the DSP56374 pins and how these signals are grouped into interfaces.

Section 3—Memory Configuration

- Describes the DSP56374 memory spaces, RAM and ROM configuration, memory configurations and their bit settings and memory maps.

Section 4—Core Configuration

- Describes the registers used to configure the DSP56300 core when programming the DSP56374, in particular the interrupt vector locations and the operation of the interrupt priority registers. Explains the operating modes and how they affect the processor's program and data memories.

Section 5—Phase-Locked Loop (PLL) and Clock Generator

- Describes the DSP56374 PLL and clock generator capability and the programming model for the PLL (operation, registers and control).

Section 6—General Purpose Input/Output (GPIO)

- Describes the DSP56374 GPIO capability and the programming model for the GPIO signals (operation, registers and control).

Section 7—Serial Host Interface (SHI)

- Describes the serial input/output interface providing a path for communication and program/coefficient data transfers between the DSP and an external host processor. The SHI can also communicate with other serial peripheral devices.

Section 8—Enhanced Serial Audio Interface (ESAI)

- Describes one of the full-duplex serial port for serial communication with a variety of serial devices.

Section 9—Triple Timer Module (TEC)

- Describes the Architecture, Programming model, and operating modes of three identical timer devices available for use as internals or event counters. Describes the operation of the Triple Timer and its many functions.

Section 10—Watchdog Timer Module (WDT)

- Describes the Architecture, Programming model and, operating modes of the watchdog timer.

Appendix A—Bootstrap Program

- Lists the bootstrap code used for the DSP56374.

Appendix B—Equates

- Lists equates for the DSP56374.

Appendix C—Programming Reference

- Lists peripheral addresses, interrupt addresses and interrupt priorities for the DSP56374. Contains programming sheets listing the contents of the major DSP56374 registers for programmer reference.

Appendix D—BSDL

- Provides the BSDL data for the DSP56374.

Revision History

The following table summarizes revisions to this document.

Revision	Date	Location	Comments
1.1	Nov 2004		Previous release.
1.2	July 2007	Chapter 6	• In section 6.2.2 Port G Signals and Registers, corrected register acronyms (PCRG, PRRG, PDRG). • In Table 6-2, corrected statements for ERI1, ETI0, ERI0.
		Appendix B	• In Equates statements, corrected addresses for PDRG, PRRG, PCRG on page B-4.
		Appendix C	• In Figure C-24, corrected register addresses for PCRG, PRRG, PDRG. • In Table C-2, corrected register acronyms for PCRG, PRRG, PDRG.

Manual Conventions

The following conventions are used in this manual:

- Bits within registers are always listed from most significant bit (MSB) to least significant bit (LSB).
- When several related bits are discussed, they are referenced as AA[n:m], where n>m. For purposes of description, the bits are presented as if they are contiguous within a register. However, this is not always the case. Refer to the programming model diagrams or to the programmer's sheets to see the exact location of bits within a register.
- When a bit is described as "set", its value is 1. When a bit is described as "cleared", its value is 0.
- The word "assert" means that a high true (active high) signal is pulled high to V_{DD} or that a low true (active low) signal is pulled low to ground. The word "de-assert" means that a high true signal is pulled low to ground or that a low true signal is pulled high to V_{DD}.

High True/Low True Signal Conventions

Signal/Symbol	Logic State	Signal State	Voltage
$\overline{\text{PIN}}$[1]	True	Asserted	Ground[2]
PIN	False	De-asserted	V_{DD}[3]
PIN	True	Asserted	V_{DD}
PIN	False	De-asserted	Ground

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

- Pins or signals that are asserted low (made active when pulled to ground)
  - In text, have an overbar (e.g., $\overline{\text{RESET}}$ is asserted low).
  - In code examples, have a tilde in front of their names. In example below, line 3 refers to the $\overline{\text{SS0}}$ pin (shown as `~SS0`).
- Sets of pins or signals are indicated by the first and last pins or signals in the set (e.g., SDO0–SDO5).
- Code examples are displayed in a monospaced font, as shown below:

Example Sample Code Listing

```
        BSET    #$000007,X:PCRC; Configure:PC7                  line 1
        BCLR    #$000007,X:PRRC; Configure:PDC7                 line 2
                         ; SDO4/SDI1 as PC7 for GPIO Input      line 3
```

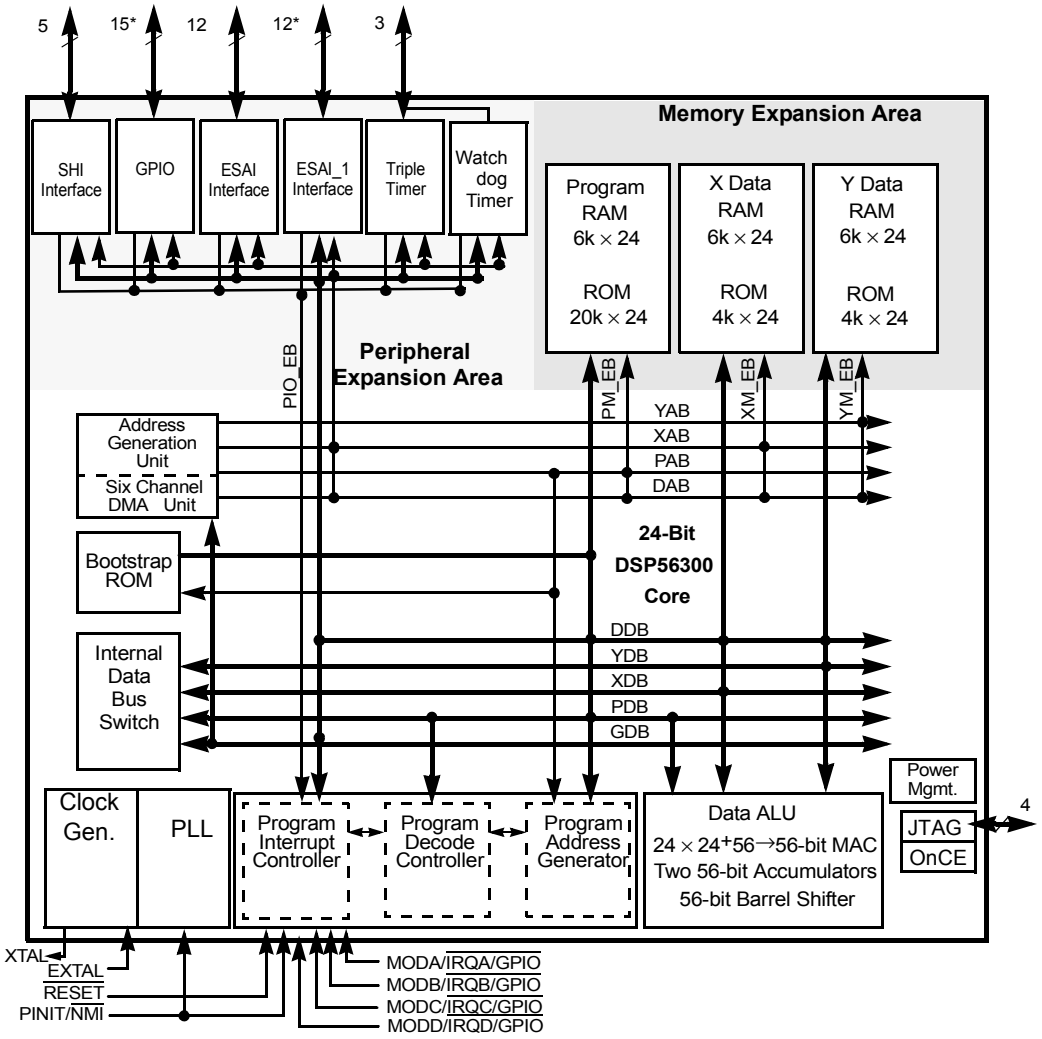
- Hex values are indicated with a dollar sign ($) preceding the hex value, as follows: $FFFFFF is the X memory address for the core interrupt priority register (IPR-C).
- The word "reset" is used in four different contexts in this manual:
  - the reset signal, written as "$\overline{\text{RESET}}$,"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

Chapter 1
DSP56374 Overview

1.1 Introduction

The DSP56374 is designed to support a multitude of digital signal processing applications requiring a lot of horsepower in a small package. This manual describes the DSP56374 24-bit digital signal processor (DSP), its memory, operating modes and peripheral modules.

The DSP56374 is a member of the Symphony™ family of programmable CMOS DSPs and is built on the high performance, single-clock-per-cycle DSP56300 core. The DSP56374 is provided in either an 80-pin or 52-pin package. This design provides a two-fold performance increase over Freescale's (formerly Motorola) popular DSP56000 Symphony family of DSPs while retaining code compatibility. Significant architectural enhancements include a barrel shifter, 24-bit addressing and direct memory access (DMA). Changes in core functionality specific to the DSP56374 are also described in this manual. See Figure 1-1 for the block diagram of the DSP56374.



* ESAI_1 and dedicated GPIO pins are not available in the 52-pin package.

Figure 1-1. DSP56374 Block Diagram

1.2 DSP56300 Core Description

The DSP56374 uses the DSP56300 core, a high-performance, single clock cycle per instruction engine that provides several times the performance of Freescale's (formerly Motorola's) popular DSP56000 core family while retaining code compatibility.

The DSP56300 core family offers a new level of performance in speed and power, provided by its rich instruction set and low power dissipation, thus enabling a new generation of wireless, telecommunications and multimedia products. For a description of the DSP56300 core, see Section 1.4, *DSP56300 Core Functional Blocks*. Significant architectural enhancements to the DSP56300 core family include a barrel shifter, 24-bit addressing, an instruction patch module and direct memory access (DMA).

The DSP56300 core family members contain the DSP56300 core and additional modules. The modules are chosen from a library of standard predesigned elements such as memories and peripherals. Note that new modules may be added to the library to meet customer specifications in future DSP56300 products. A standard interface between the DSP56300 core and the on-chip memory and peripherals supports a wide variety of memory and peripheral configurations. Refer to Chapter 3, *Memory Configuration*.

Core features are described fully in the *DSP56300 Family Manual*. Pinout, memory and peripheral features are described in this manual.

- DSP56300 modular chassis
  - 150 Million Instructions Per Second (MIPS) with a 150 MHz clock at an internal logic supply (QVDDL) of 1.25V.
  - Object Code Compatible with the 56k core.
  - Data ALU with a 24 x 24 bit multiplier-accumulator and a 56-bit barrel shifter;16-bit arithmatic support.
  - Program Control with position independent code support.
  - Six-channel DMA controller.
  - Provides a wide range of frequency multiplications (1 to 255), predivider factors (1 to 31), PLL feedback multiplier (2 or 4), Output divide factor (1, 2 or 4) and a power-saving clock divider (2^i: i = 0 to 7) to reduce clock noise
  - Internal address tracing support and OnCE for Hardware/Software debugging.
  - JTAG port, supporting boundary scan, compliant to IEEE 1149.1.
  - Very low-power CMOS design, fully static design with operating frequencies down to DC.
  - STOP and WAIT low-power standby modes.
- On-chip Memory Configuration
  - 6Kx24 Bit Y-Data RAM and 4Kx24 Bit Y-Data ROM.
  - 6Kx24 Bit X-Data RAM and 4Kx24 Bit X-Data ROM.
  - 20Kx24 Bit Program and Bootstrap ROM including a PROM patching mechanism.
  - 6Kx24 Bit Program RAM.
  - Various memory switches are available. See memory table below.

Table 1-1. DSP56374 Memory Switch Configurations

Bit Settings			Memory Sizes (24-bit words)					
MSW1	MSW0	MS	Prog RAM	X Data RAM	Y Data RAM	Prog ROM	X Data ROM	Y Data ROM
X	X	0	6k	6k	6k	20k	4k	4k
0	0	1	2k	10k	6k	20k	4k	4k
0	1	1	4k	8k	6k	20k	4k	4k
1	0	1	8k	4k	6k	20k	4k	4k
1	1	1	10K	4k	4k	20k	4k	4k

- Peripheral modules
  - Enhanced Serial Audio Interface (ESAI): up to 4 receiver pins and up to 6 transmitter pins, master or slave. I^2S, Sony, AC97, network and other programmable protocols.
  - Enhanced Serial Audio Interface I (ESAI_1): up to 4 receiver pins and up to 6 transmitter pins, master or slave. I^2S, Sony, AC97, network and other programmable protocols. *Note: Available in the 80-pin package only*
  - Serial Host Interface (SHI): SPI and I^2C protocols, 10-word receive FIFO, support for 8-, 16- and 24-bit words. Three noise reduction filter modes.
  - Triple Timer module (TEC).
  - Most pins of unused peripherals may be programmed as GPIO pins. Up to 47 pins can be configured as GPIO on the 80-pin package and 20 pins on the 52-pin package.

  - Hardware Watchdog Timer
- Packages
  - 80-pin and 52-pin plastic LQFP packages.

1.3 DSP56374 Audio Processor Architecture

This section defines the DSP56374 audio processor architecture. The audio processor is composed of the following units:

- The DSP56300 core is composed of the Data ALU, Address Generation Unit, Program Controller, DMA Controller, Memory Module Interface, Peripheral Module Interface and the On-Chip Emulator (OnCE). The DSP56300 core is described in the document *DSP56300 24-Bit Digital Signal Processor Family Manual,* Freescale (formerly Motorola) publication DSP56300FM/AD.
- Phased Lock Loop and Clock Generator
- Memory modules.
- Peripheral modules. The peripheral modules are defined in the following sections.

Memory sizes in the block diagram are defaults. Memory may be differently partitioned, according to the memory mode of the chip. See Table 1-1 and Section 1.4.7, *On-Chip Memory* for more details about memory size.

1.4 DSP56300 Core Functional Blocks

The DSP56300 core provides the following functional blocks:

- Data arithmetic logic unit (Data ALU)
- Address generation unit (AGU)
- Program control unit (PCU)
- DMA controller (with six channels)
- Instruction patch controller
- PLL-based clock oscillator
- OnCE module
- Memory

In addition, the DSP56374 provides a set of on-chip peripherals, described in Section 1.5, *Peripheral Overview*.

1.4.1 Data ALU

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

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

1.4.1.1 Data ALU Registers

The Data ALU registers can be read or written over the X memory data bus (XDB) and the Y memory data bus (YDB) as 24- or 48-bit operands (or as 16- or 32-bit operands in 16-bit arithmetic mode). The source operands for the Data ALU, which can be 24, 48, or 56 bits (16, 32, or 40 bits in 16-bit arithmetic mode), always originate from Data ALU registers. The results of all Data ALU operations are stored in an accumulator.

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

1.4.1.2 Multiplier-Accumulator (MAC)

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

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

1.4.2 Address Generation Unit (AGU)

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

The AGU is divided into two halves, each with its own Address ALU. Each Address ALU has four sets of register triplets, and each register triplet is composed of an address register, an offset register and a modifier register. The two Address ALUs are identical. Each contains a 24-bit full adder (called an offset adder).

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

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

Each Address ALU can update one address register from its respective address register file during one instruction cycle. The contents of the associated modifier register specifies the type of arithmetic to be used in the address register update calculation. The modifier value is decoded in the Address ALU.

1.4.3 Program Control Unit (PCU)

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

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

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

PCU features include the following:

- Position independent code support
- Addressing modes optimized for DSP applications (including immediate offsets)
- On-chip memory-expandable hardware stack
- Nested hardware DO loops
- Fast auto-return interrupts

The PCU implements its functions using the following registers:

- PC—program counter register
- SR—Status register
- LA—loop address register
- LC—loop counter register
- VBA—vector base address register
- SZ—stack size register
- SP—stack pointer
- OMR—operating mode register
- SC—stack counter register

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

1.4.4 Internal Buses

To provide data exchange between blocks, the following buses are implemented:

- Peripheral input/output expansion bus (PIO_EB) to peripherals
- Program memory expansion bus (PM_EB) to program memory

- X memory expansion bus (XM_EB) to X memory
- Y memory expansion bus (YM_EB) to Y memory
- Global data bus (GDB) between registers in the DMA, AGU, OnCE, PLL, BIU and PCU, as well as the memory-mapped registers in the peripherals
- DMA data bus (DDB) for carrying DMA data between memories and/or peripherals
- DMA address bus (DAB) for carrying DMA addresses to memories and peripherals
- Program Data Bus (PDB) for carrying program data throughout the core
- X memory Data Bus (XDB) for carrying X data throughout the core
- Y memory Data Bus (YDB) for carrying Y data throughout the core
- Program address bus (PAB) for carrying program memory addresses throughout the core
- X memory address bus (XAB) for carrying X memory addresses throughout the core
- Y memory address bus (YAB) for carrying Y memory addresses throughout the core

All internal buses on the DSP56300 family members are 24-bit buses. See Figure 1-1.

1.4.5 Direct Memory Access (DMA)

The DMA block has the following features:

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

1.4.6 PLL-based Clock Oscillator

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

- Allows change of low-power divide factor (DF) without loss of lock
- Provides output clock with skew elimination
- Provides a wide range of frequency multiplications (1 to 255), predivider factors (1 to 31), PLL feedback multiplier (2 or 4), Output divide factor (1, 2 or 4) and a power-saving clock divider (2^i: i = 0 to 7) to reduce clock noise

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

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

1.4.7 On-Chip Memory

The memory space of the DSP56300 core is partitioned into program memory space,
X data memory space and Y data memory space. The data memory space is divided into X and Y data memory in order to work with the two Address ALUs and to feed two operands simultaneously to the Data ALU. Memory space includes internal RAM and ROM and can not be expanded off-chip.

There is an instruction patch module. The patch module is used to patch program ROM. The memory switch mode is used to increase the size of program RAM as needed (switch from X data RAM and/or Y data RAM).

There are on-chip ROMs for program and bootstrap memory (20k x 24-bit), X ROM (4k x 24-bit) and Y ROM(4k x 24-bit).

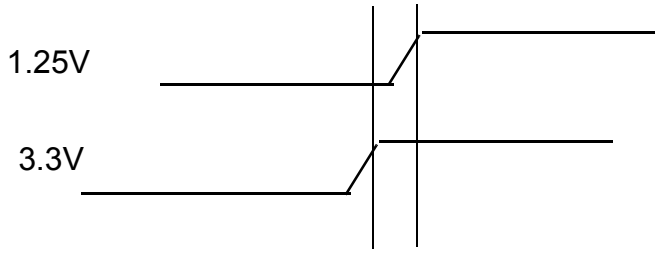
More information on the internal memory is provided in Chapter 3, *Memory Configuration*.

1.4.8 Off-Chip Memory Expansion

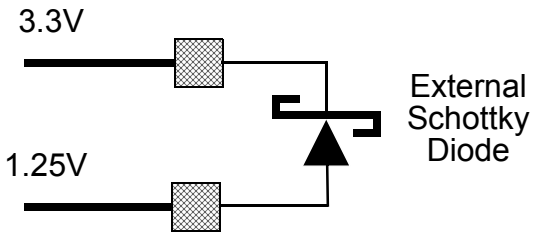
Memory cannot be expanded off-chip. There is no external memory bus.

1.4.9 Power Requirements

To prevent a high current condition and damage to the DSP upon power up, the 3.3V source must be applied ahead of the 1.25V source as shown below.



To prevent high current conditions due to possible improper sequencing of the power supplies, the connection shown below is recommended to be made between the DSP56374 3.3V and 1.25V power pins.



1.5 Peripheral Overview

The DSP56374 is designed to perform a wide variety of fixed-point digital signal processing functions. In addition to the core features previously discussed, the DSP56374 provides the following peripherals:

- As many as 47 dedicated or user-configurable general purpose input/output (GPIO) signals on the 80-pin package and 20 dedicated or user-configurable GPIO on the 52-pin package.
- Timer/event counter (TEC) module, containing three independent timers
- Memory switch mode in on-chip memory
- Four external interrupt/mode control lines and one external non-maskable interrupt line
- Enhanced serial audio interface (ESAI) with up to four receivers and up to six transmitters, master or slave, using the I^2S, Sony, AC97, network and other programmable protocols
- A second enhanced serial audio interface (ESAI_1) with up to four receivers and up to six transmitters, master or slave, using the I^2S, Sony, AC97, network and other programmable protocols. *Note: only available on the 80-pin package.*
- Serial host interface (SHI) using SPI and I^2C protocols, with multi-master capability, 10-word receive FIFO and support for 8-, 16- and 24-bit words
- A Hardware Watchdog Timer.

1.5.1 General Purpose Input/Output (GPIO)

The 80-pin DSP56374 provides 15 dedicated GPIO and 29 programmable pins that can operate either as GPIO pins or peripheral pins (ESAI, ESAI_1 and TEC). The four MOD pins, as well as the SHI HREQ pin, can also be utilized as GPIO. The ESAI and ESAI_1 pins are configured as GPIO after hardware reset. Register-programming techniques for all GPIO functionality among these interfaces are very similar and are described in the following sections.

1.5.2 Triple Timer (TEC)

This section describes a peripheral module composed of a common 21-bit prescaler and three independent and identical general purpose 24-bit timer/event counters, each one having its own register set.

Each timer can use internal or external clocking and can interrupt the DSP after a specified number of events (clocks). Each of the three timers can signal an external device after counting internal events. Each timer can also be used to trigger DMA transfers after a specified number of events (clocks) occurred. Each of the three timers connect to the external world through bidirectional pins (TIO0, TIO1 and TIO2). When a TIO pin is configured as input, the timer functions as an external event counter or can measure external pulse width/signal period. When a

TIO pin is used as output the timer is functioning as either a timer, a watchdog or a Pulse Width Modulator. When a TIO pin is not used by the timer it can be used as a General Purpose Input/Output Pin. Refer to Chapter 9, *Triple Timer Module*.

1.5.3 Enhanced Serial Audio Interface (ESAI)

The ESAI provides a full-duplex serial port for serial communication with a variety of serial devices including one or more industry-standard codecs, other DSPs, microprocessors and peripherals that implement the Freescale (formerly Motorola) SPI serial protocol. The ESAI consists of independent transmitter and receiver sections, each with its own clock generator. It is a superset of the DSP56300 family ESSI peripheral and of the DSP56000 family SAI peripheral. For more information on the ESAI, refer to Chapter 8, *Enhanced Serial Audio Interface (ESAI)*.

1.5.4 Enhanced Serial Audio Interface 1 (ESAI_1)

The ESAI_1 is a second ESAI interface. The ESAI_1 is functionally identical to ESAI. For more information on the ESAI_1, refer to Chapter 9, *Triple Timer Module*.

1.5.5 Serial Host Interface (SHI)

The SHI is a serial input/output interface providing a path for communication and program/coefficient data transfers between the DSP and an external host processor. The SHI can also communicate with other serial peripheral devices. The SHI can interface directly to either of two well-known and widely used synchronous serial buses: the Freescale (formerly Motorola) serial peripheral interface (SPI) bus and the Philips inter-integrated-circuit control (I^2C) bus. The SHI supports either the SPI or I^2C bus protocol, as required, from a slave or a single-master device. To minimize DSP overhead, the SHI supports single-, double- and triple-byte data transfers. The SHI has a 10-word receive FIFO that permits receiving up to 30 bytes before generating a receive interrupt, reducing the overhead for data reception. For more information on the SHI, refer to Chapter 7, *Serial Host Interface*.

1.5.6 Watchdog timer (WDT)

The watchdog timer (WDT) is a 16-bit timer used to help software recover from runaway code. The timer is a free-running down-counter used to generate a reset on underflow. Software must periodically service the watchdog timer in order to restart the count down. For more information on the WDT, refer to Chapter 10, *Watchdog Timer Module*.

Notes

Chapter 2
Signal/Connection Descriptions

2.1 Signal Groupings

The input and output signals of the DSP56374 are organized into functional groups, which are listed in Table 2-1.

The DSP56374 is operated from a 1.25 V and 3.3 V supply; however, some of the inputs can tolerate 5.0 V. A special notice for this feature is added to the signal descriptions of those inputs. Resistor values for pins with pull up or pull down resistors may vary with lot and will be between 40k ohms and 65k ohms.

Table 2-1. DSP56374 Functional Signal Groupings

Functional Group		Number of Signals[a]	Detailed Description
Power (V_{DD})		11	Table 2-2
Ground (GND)		9	Table 2-3
Scan Pins		1	Table 2-4.
Clock and PLL		3	Table 2-5.
Interrupt and mode control	Port H[1]	5	Table 2-6.
SHI	Port H[1]	5	Table 2-7.
ESAI	Port C[3]	12	Table 2-8.
ESAI_1	Port E[4]	12	Table 2-9.
Dedicated GPIO	Port G[2]	15	Table 2-10.
Timer		3	Table 2-11.
JTAG/OnCE Port		4	Table 2-12.

Note:
1. Port H signals are the GPIO port signals which are multiplexed with the MOD and HREQ signals.
2. Port G signals are the dedicated GPIO port signals.
3. Port C signals are the GPIO port signals which are multiplexed with the ESAI signals.
4. Port E signals are the GPIO port signals which are multiplexed with the ESAI_1 signals.

[a] Note: Pins are not 5 V. tolerant unless noted.

2.2 Power

Table 2-2. Power Inputs

Power Name	Description
PLLA_VDD (1)	**PLL Power**— The voltage (3.3 V) should be well-regulated, and the input should be provided with an extremely low impedance path to the 3.3 V_{DD} power rail. The user must provide adequate external decoupling capacitors between PLLA_VDD and PLLA_GND. PLLA_VDD requires a filter as shown in Figure 2-1. and Figure 2-2. below. See the DSP56374 technical data sheet for additional details.
PLLP_VDD(1)	**PLL Power**— The voltage (3.3 V) should be well-regulated, and the input should be provided with an extremely low impedance path to the 3.3 V_{DD} power rail. The user must provide adequate external decoupling capacitors between PLLP_VDD and PLLP_GND.
PLLD_VDD (1)	**PLL Power—** The voltage (1.25 V) should be well-regulated, and the input should be provided with an extremely low impedance path to the 1.25 V_{DD} power rail. The user must provide adequate external decoupling capacitors between PLLD_VDD and PLLD_GND.

Table 2-2. Power Inputs

Power Name	Description
CORE_VDD (4)	**Core Power**—The voltage (1.25 V) should be well-regulated, and the input should be provided with an extremely low impedance path to the 1.25 V_{DD} power rail. The user must provide adequate external decoupling capacitors.
IO_VDD (80-pin 4) (52-pin 3)	**SHI, ESAI, ESAI_1, WDT and Timer I/O Power** —The voltage (3.3 V) should be well-regulated, and the input should be provided with an extremely low impedance path to the 3.3 V_{DD} power rail. This is an isolated power for the SHI, ESAI, ESAI_1, WDT and Timer I/O. The user must provide adequate external decoupling capacitors.

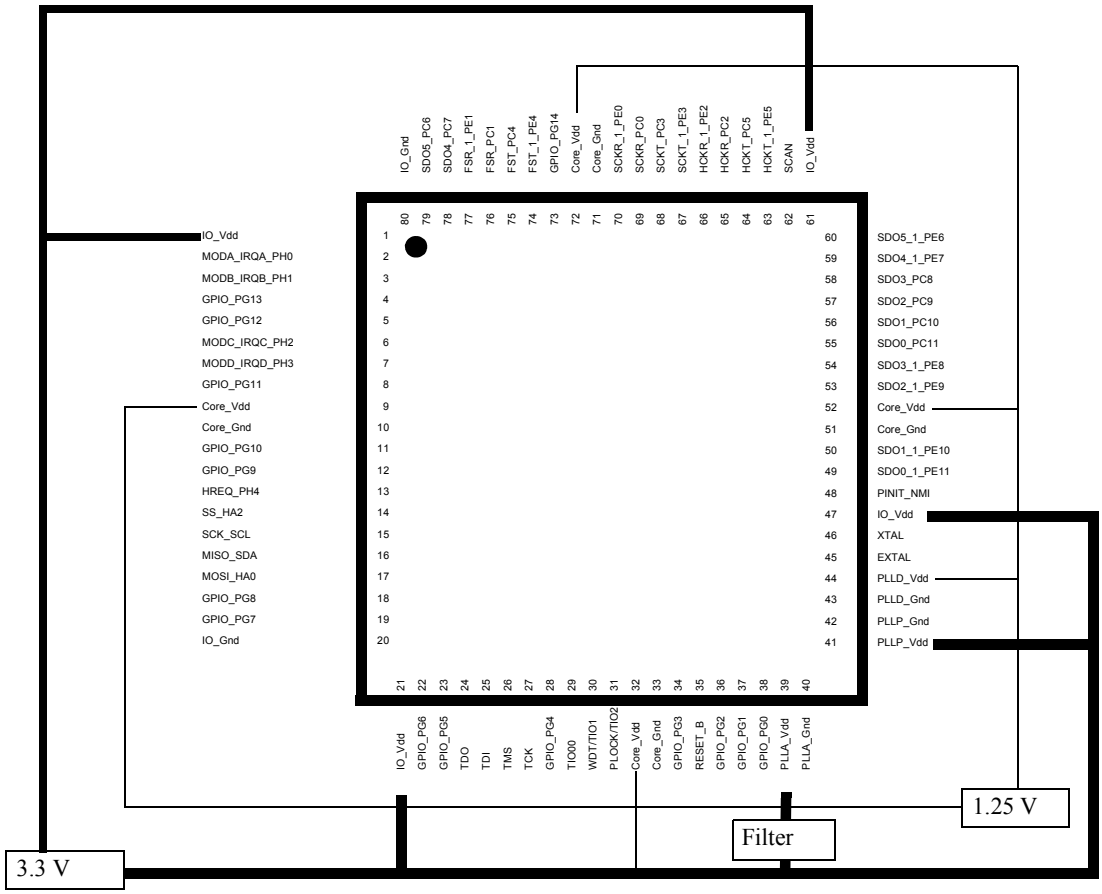


Figure 2-1. 80-pin Vdd Connections

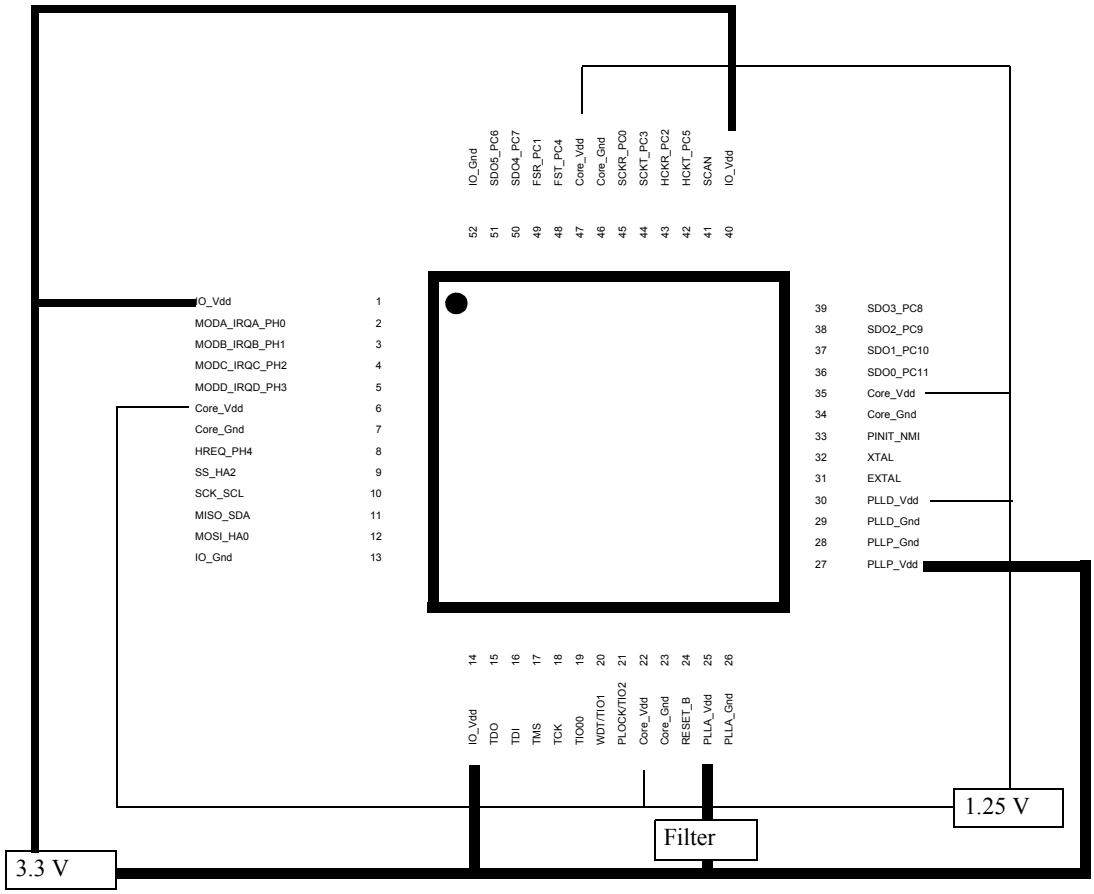


Figure 2-2. 52-pin Vdd Connections

2.3 Ground

Table 2-3. Grounds

Ground Name	Description
PLLA_GND(1)	**PLL Ground**—The PLL ground should be provided with an extremely low-impedance path to ground. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors between PLLA_VDD and PLLA_GND.
PLLP_GND(1)	**PLL Ground**—The PLL ground should be provided with an extremely low-impedance path to ground. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors between PLLP_VDD and PLLP_GND.
PLLD_GND(1)	**PLL Ground**—The PLL ground should be provided with an extremely low-impedance path to ground. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors between PLLD_VDD and PLLD_GND.
CORE_GND(4)	**Core Ground**—The Core ground should be provided with an extremely low-impedance path to ground. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors.
IO_GND(2)	**SHI, ESAI, ESAI_1, WDT and Timer I/O Ground**—IO_GND is the ground for the SHI, ESAI, ESAI_1, WDT and Timer I/O. This connection must be tied externally to all other chip ground connections. The user must provide adequate external decoupling capacitors.

2.4 SCAN

Table 2-4. SCAN signals

Signal Name	Type	State during Reset	Signal Description
SCAN	Input	Input	**SCAN**—Manufacturing test pin. This pin must be connected to ground. This pin has an internal pull-down resistor.

2.5 Clock and PLL

Table 2-5. Clock and PLL Signals

Signal Name	Type	State during Reset	Signal Description
EXTAL	Input	Input	**External Clock / Crystal Input**—An external clock source must be connected to EXTAL in order to supply the clock to the internal clock generator and PLL.
XTAL	Output	Chip Driven	**Crystal Output**—Connects the internal Crystal Oscillator output to an external crystal. If an external clock is used, leave XTAL unconnected.
PINIT/$\overline{\text{NMI}}$	Input	Input	**PLL Initial/Non-maskable Interrupt**—During assertion of $\overline{\text{RESET}}$, the value of PINIT/$\overline{\text{NMI}}$ is written into the PLL Enable (PEN) bit of the PLL control register, determining whether the PLL is enabled or disabled. After $\overline{\text{RESET}}$ de-assertion and during normal instruction processing, the PINIT/$\overline{\text{NMI}}$ Schmitt-trigger input is a negative-edge-triggered non-maskable interrupt (NMI) request internally synchronized to the internal system clock. This pin has an internal pull-up resistor. This input is 5 V tolerant.

2.6 Interrupt and Mode Control

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

Table 2-6. Interrupt and Mode Control

Signal Name	Type	State during Reset	Signal Description
MODA/$\overline{\text{IRQA}}$	Input	MODA Input	**Mode Select A/External Interrupt Request A—**MODA/$\overline{\text{IRQA}}$ is an active-low Schmitt-trigger input, internally synchronized to the DSP clock. MODA/$\overline{\text{IRQA}}$ selects the initial chip operating mode during hardware reset and becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. This pin can also be programmed as GPIO. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into the OMR when the $\overline{\text{RESET}}$ signal is de-asserted. If the processor is in the stop standby state and the MODA/$\overline{\text{IRQA}}$ pin is pulled to GND, the processor will exit the stop state. This pin has an internal pull-up resistor. This input is 5 V tolerant.
PH0	Input, Output, or Disconnected		**Port H0**—When the MODA/IRQA is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected.

Table 2-6. Interrupt and Mode Control (continued)

Signal Name	Type	State during Reset	Signal Description
MODB/IRQB	Input	MODB Input	**Mode Select B/External Interrupt Request B—**MODB/IRQB is an active-low Schmitt-trigger input, internally synchronized to the DSP clock. MODB/IRQB selects the initial chip operating mode during hardware reset and becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. This pin can also be programmed as GPIO. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into OMR when the RESET signal is de-asserted. This pin has an internal pull-up resistor. This input is 5 V tolerant.
PH1	Input, Output, or Disconnected		**Port H1**—When the MODB/IRQB is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected.
MODC/IRQC	Input	MODC Input	**Mode Select C/External Interrupt Request C—**MODC/IRQC is an active-low Schmitt-trigger input, internally synchronized to the DSP clock. MODC/IRQC selects the initial chip operating mode during hardware reset and becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. This pin can also be programmed as GPIO. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into OMR when the RESET signal is de-asserted. This pin has an internal pull-up resistor. This input is 5 V tolerant.
PH2	Input, Output, or Disconnected		**Port H2**—When the MODC/IRQC is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected.
MODD/IRQD	Input	MODD Input	**Mode Select D/External Interrupt Request D**—MODD/IRQD is an active-low Schmitt-trigger input, internally synchronized to the DSP clock. MODD/IRQD selects the initial chip operating mode during hardware reset and becomes a level-sensitive or negative-edge-triggered, maskable interrupt request input during normal instruction processing. This pin can also be programmed as GPIO. MODA, MODB, MODC, and MODD select one of 16 initial chip operating modes, latched into OMR when the RESET signal is de-asserted. This pin has an internal pull-up resistor. This input is 5 V tolerant.
PH3	Input, output, or disconnected		**Port H3**—When the MODD/IRQD is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected.
RESET	Input	Input	**Reset—**RESET is an active-low, Schmitt-trigger input. When asserted, the chip is placed in the Reset state and the internal phase generator is reset. The Schmitt-trigger input allows a slowly rising input (such as a capacitor charging) to reset the chip reliably. When the RESET signal is de-asserted, the initial chip operating mode is latched from the MODA, MODB, MODC, and MODD inputs. The RESET signal must be asserted during power up. A stable EXTAL signal must be supplied while RESET is being asserted. This pin has an internal pull-up resistor. This input is 5 V tolerant.

2.7 Serial Host Interface

The SHI has five I/O signals that can be configured to allow the SHI to operate in either SPI or I^2C mode.

Table 2-7. Serial Host Interface Signals

Signal Name	Signal Type	State during Reset	Signal Description
SCK	Input or Output	Tri-stated	**SPI Serial Clock**—The SCK signal is an output when the SPI is configured as a master and a Schmitt-trigger input when the SPI is configured as a slave. When the SPI is configured as a master, the SCK signal is derived from the internal SHI clock generator. When the SPI is configured as a slave, the SCK signal is an input, and the clock signal from the external master synchronizes the data transfer. The SCK signal is ignored by the SPI if it is defined as a slave and the slave select ($\overline{SS}$) signal is not asserted. In both the 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. Edge polarity is determined by the SPI transfer protocol.
SCL	Input or Output		**I^2C Serial Clock**—SCL carries the clock for I^2C bus transactions in the I^2C mode. SCL is a Schmitt-trigger input when configured as a slave and an open-drain output when configured as a master. SCL should be connected to V_{DD} through an external pull-up resistor according to the I^2C specifications. This signal is tri-stated during hardware, software, and individual reset. This pin has an internal pull-up resistor. This input is 5 V tolerant.
MISO	Input or Output	Tri-stated	**SPI Master-In-Slave-Out**—When the SPI is configured as a master, MISO is the master data input line. The MISO signal is used in conjunction with the MOSI signal for transmitting and receiving serial data. This signal is a Schmitt-trigger input when configured for the SPI Master mode, an output when configured for the SPI Slave mode, and tri-stated if configured for the SPI Slave mode when $\overline{SS}$ is de-asserted. An external pull-up resistor is not required for SPI operation.
SDA	Input or Open-drain Output		**I^2C Data and Acknowledge**—In I^2C mode, SDA is a Schmitt-trigger input when receiving and an open-drain output when transmitting. SDA should be connected to V_{DD} through a pull-up resistor. SDA carries the data for I^2C transactions. The data in SDA must be stable during the high period of SCL. The data in SDA is only allowed to change when SCL is low. When the bus is free, SDA is high. The SDA line is only allowed to change during the time SCL is high in the case of start and stop events. A high-to-low transition of the SDA line while SCL is high is a unique situation, and is defined as the start event. A low-to-high transition of SDA while SCL is high is a unique situation defined as the stop event. This signal is tri-stated during hardware, software, and individual reset. Thus, there is no need for an external pull-up in this state. This pin has an internal pull-up resistor. This input is 5 V tolerant.

Table 2-7. Serial Host Interface Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
MOSI	Input or Output	Tri-stated	**SPI Master-Out-Slave-In**—When the SPI is configured as a master, MOSI is the master data output line. The MOSI signal is used in conjunction with the MISO signal for transmitting and receiving serial data. MOSI is the slave data input line when the SPI is configured as a slave. This signal is a Schmitt-trigger input when configured for the SPI Slave mode.
HA0	Input		**I^{2}C Slave Address 0**—This signal uses a Schmitt-trigger input when configured for the I^{2}C mode. When configured for I^{2}C slave mode, the HA0 signal is used to form the slave device address. HA0 is ignored when configured for the I^{2}C master mode. This signal is tri-stated during hardware, software, and individual reset. Thus, there is no need for an external pull-up in this state. This pin has an internal pull-up resistor. This input is 5 V tolerant.
$\overline{SS}$	Input	Ignored Input	**SPI Slave Select**—This signal is an active low Schmitt-trigger input when configured for the SPI mode. When configured for the SPI Slave mode, this signal is used to enable the SPI slave for transfer. When configured for the SPI master mode, this signal should be kept de-asserted (pulled high). If it is asserted while configured as SPI master, a bus error condition is flagged. If $\overline{SS}$ is de-asserted, the SHI ignores SCK clocks and keeps the MISO output signal in the high-impedance state.
HA2	Input		**I^{2}C Slave Address 2**—This signal uses a Schmitt-trigger input when configured for the I^{2}C mode. When configured for the I^{2}C Slave mode, the HA2 signal is used to form the slave device address. HA2 is ignored in the I^{2}C master mode. This pin has an internal pull-up resistor. This input is 5 V tolerant.
$\overline{HREQ}$	Input or Output	Tri-stated	**Host Request**—This signal is an active low Schmitt-trigger input when configured for the master mode but an active low output when configured for the slave mode. When configured for the slave mode, $\overline{HREQ}$ is asserted to indicate that the SHI is ready for the next data word transfer and de-asserted at the first clock pulse of the new data word transfer. When configured for the master mode, $\overline{HREQ}$ is an input. When asserted by the external slave device, it will trigger the start of the data word transfer by the master. After finishing the data word transfer, the master will await the next assertion of $\overline{HREQ}$ to proceed to the next transfer. This pin can also be programmed as GPIO.
PH4	Input, Output, or Disconnected		**Port H4**—When $\overline{HREQ}$ is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-up resistor. This input is 5 V tolerant.

2.8 Enhanced Serial Audio Interface

Table 2-8. Enhanced Serial Audio Interface Signals

Signal Name	Signal Type	State during Reset	Signal Description
HCKR	Input or Output	GPIO Disconnected	**High Frequency Clock for Receiver**—When programmed as an input, this signal provides a high frequency clock source for the ESAI receiver as an alternate to the DSP core clock. When programmed as an output, this signal can serve as a high-frequency sample clock (e.g., for external digital to analog converters [DACs]) or as an additional system clock.
PC2	Input, Output, or Disconnected		**Port C2**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-up resistor. This input is 5 V tolerant.
HCKT	Input or Output	GPIO Disconnected	**High Frequency Clock for Transmitter**—When programmed as an input, this signal provides a high frequency clock source for the ESAI transmitter as an alternate to the DSP core clock. When programmed as an output, this signal can serve as a high frequency sample clock (e.g., for external DACs) or as an additional system clock.
PC5	Input, Output, or Disconnected		**Port C5**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-up resistor. This input is 5 V tolerant.

Table 2-8. Enhanced Serial Audio Interface Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
FSR	Input or Output	GPIO Disconnected	**Frame Sync for Receiver**—This is the receiver frame sync input/output signal. In the asynchronous mode (SYN=0), the FSR pin operates as the frame sync input or output used by all the enabled receivers. In the synchronous mode (SYN=1), it operates as either the serial flag 1 pin (TEBE=0), or as the transmitter external buffer enable control (TEBE=1, RFSD=1). When this pin is configured as serial flag pin, its direction is determined by the RFSD bit in the RCCR register. When configured as the output flag OF1, this pin will reflect the value of the OF1 bit in the SAICR register, and the data in the OF1 bit will show up at the pin synchronized to the frame sync in normal mode or the slot in network mode. When configured as the input flag IF1, the data value at the pin will be stored in the IF1 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.
PC1	Input, Output, or Disconnected		**Port C1**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
FST	Input or Output	GPIO Disconnected	**Frame Sync for Transmitter**—This is the transmitter frame sync input/output signal. For synchronous mode, this signal is the frame sync for both transmitters and receivers. For asynchronous mode, FST is the frame sync for the transmitters only. The direction is determined by the transmitter frame sync direction (TFSD) bit in the ESAI transmit clock control register (TCCR).
PC4	Input, Output, or Disconnected		**Port C4**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

Table 2-8. Enhanced Serial Audio Interface Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
SCKR	Input or Output	GPIO Disconnected	**Receiver Serial Clock**—SCKR provides the receiver serial bit clock for the ESAI. The SCKR operates as a clock input or output used by all the enabled receivers in the asynchronous mode (SYN=0), or as serial flag 0 pin in the synchronous mode (SYN=1). When this pin is configured as serial flag pin, its direction is determined by the RCKD bit in the RCCR register. When configured as the output flag OF0, this pin will reflect the value of the OF0 bit in the SAICR register, and the data in the OF0 bit will show up at the pin synchronized to the frame sync in normal mode or the slot in network mode. When configured as the input flag IF0, the data value at the pin will be stored in the IF0 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.
PC0	Input, Output, or Disconnected		**Port C0**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SCKT	Input or Output	GPIO Disconnected	**Transmitter Serial Clock**—This signal provides the serial bit rate clock for the ESAI. SCKT is a clock input or output used by all enabled transmitters and receivers in synchronous mode, or by all enabled transmitters in asynchronous mode.
PC3	Input, Output, or Disconnected		**Port C3**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SDO5	Output	GPIO Disconnected	**Serial Data Output 5**—When programmed as a transmitter, SDO5 is used to transmit data from the TX5 serial transmit shift register.
SDI0	Input		**Serial Data Input 0**—When programmed as a receiver, SDI0 is used to receive serial data into the RX0 serial receive shift register.
PC6	Input, Output, or Disconnected		**Port C6**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

Table 2-8. Enhanced Serial Audio Interface Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
SDO4	Output	GPIO Disconnected	**Serial Data Output 4**—When programmed as a transmitter, SDO4 is used to transmit data from the TX4 serial transmit shift register.
SDI1	Input		**Serial Data Input 1**—When programmed as a receiver, SDI1 is used to receive serial data into the RX1 serial receive shift register.
PC7	Input, Output, or Disconnected		**Port C7**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SDO3	Output	GPIO Disconnected	**Serial Data Output 3**—When programmed as a transmitter, SDO3 is used to transmit data from the TX3 serial transmit shift register.
SDI2	Input		**Serial Data Input 2**—When programmed as a receiver, SDI2 is used to receive serial data into the RX2 serial receive shift register.
PC8	Input, Output, or Disconnected		**Port C8**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SDO2	Output	GPIO Disconnected	**Serial Data Output 2**—When programmed as a transmitter, SDO2 is used to transmit data from the TX2 serial transmit shift register
SDI3	Input		**Serial Data Input 3**—When programmed as a receiver, SDI3 is used to receive serial data into the RX3 serial receive shift register.
PC9	Input,Ooutput, or Disconnected		**Port C9**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SDO1	Output	GPIO Disconnected	**Serial Data Output 1**—SDO1 is used to transmit data from the TX1 serial transmit shift register.
PC10	Input, Output, or Disconnected		**Port C10**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

Table 2-8. Enhanced Serial Audio Interface Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
SDO0	Output	GPIO Disconnected	**Serial Data Output 0**—SDO0 is used to transmit data from the TX0 serial transmit shift register.
PC11	Input, Output, or Disconnected		**Port C11**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

2.9 Enhanced Serial Audio Interface_1

Table 2-9. Enhanced Serial Audio Interface_1 Signals

Signal Name	Signal Type	State during Reset	Signal Description
HCKR_1	Input or Output	GPIO Disconnected	**High Frequency Clock for Receiver**—When programmed as an input, this signal provides a high frequency clock source for the ESAI_1 receiver as an alternate to the DSP core clock. When programmed as an output, this signal can serve as a high-frequency sample clock (e.g., for external digital to analog converters [DACs]) or as an additional system clock.
PE2	Input, Output, or Disconnected		**Port E2**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
HCKT_1	Input or Output	GPIO Disconnected	**High Frequency Clock for Transmitter**—When programmed as an input, this signal provides a high frequency clock source for the ESAI_1 transmitter as an alternate to the DSP core clock. When programmed as an output, this signal can serve as a high frequency sample clock (e.g., for external DACs) or as an additional system clock.
PE5	Input, Output, or Disconnected		**Port E5**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

Table 2-9. Enhanced Serial Audio Interface_1 Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
FSR_1	Input or Output	GPIO Disconnected	**Frame Sync for Receiver_1**—This is the receiver frame sync input/output signal. In the asynchronous mode (SYN=0), the FSR_1 pin operates as the frame sync input or output used by all the enabled receivers. In the synchronous mode (SYN=1), it operates as either the serial flag 1 pin (TEBE=0), or as the transmitter external buffer enable control (TEBE=1, RFSD=1). When this pin is configured as serial flag pin, its direction is determined by the RFSD bit in the RCCR_1 register. When configured as the output flag OF1, this pin will reflect the value of the OF1 bit in the SAICR_1 register, and the data in the OF1 bit will show up at the pin synchronized to the frame sync in normal mode or the slot in network mode. When configured as the input flag IF1, the data value at the pin will be stored in the IF1 bit in the SAISR_1 register, synchronized by the frame sync in normal mode or the slot in network mode.
PE1	Input, Output, or Disconnected		**Port E1**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
FST_1	Input or Output	GPIO Disconnected	**Frame Sync for Transmitter_1**—This is the transmitter frame sync input/output signal. For synchronous mode, this signal is the frame sync for both transmitters and receivers. For asynchronous mode, FST_1 is the frame sync for the transmitters only. The direction is determined by the transmitter frame sync direction (TFSD) bit in the ESAI_1 transmit clock control register (TCCR_1).
PE4	Input, Output, or Disconnected		**Port E4**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

Table 2-9. Enhanced Serial Audio Interface_1 Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
SCKR_1	Input or Output	GPIO Disconnected	**Receiver Serial Clock_1**—SCKR_1 provides the receiver serial bit clock for the ESAI_1. The SCKR_1 operates as a clock input or output used by all the enabled receivers in the asynchronous mode (SYN=0), or as serial flag 0 pin in the synchronous mode (SYN=1). When this pin is configured as serial flag pin, its direction is determined by the RCKD bit in the RCCR_1 register. When configured as the output flag OF0, this pin will reflect the value of the OF0 bit in the SAICR_1 register, and the data in the OF0 bit will show up at the pin synchronized to the frame sync in normal mode or the slot in network mode. When configured as the input flag IF0, the data value at the pin will be stored in the IF0 bit in the SAISR_1 register, synchronized by the frame sync in normal mode or the slot in network mode.
PE0	Input, Output, or Disconnected		**Port E0**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
SCKT_1	Input or Output	GPIO Disconnected	**Transmitter Serial Clock_1**—This signal provides the serial bit rate clock for the ESAI_1. SCKT_1 is a clock input or output used by all enabled transmitters and receivers in synchronous mode, or by all enabled transmitters in asynchronous mode.
PE3	Input, Output, or Disconnected		**Port E3**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
SDO5_1	Output	GPIO Disconnected	**Serial Data Output 5_1**—When programmed as a transmitter, SDO5_1 is used to transmit data from the TX5 serial transmit shift register.
SDI0_1	Input		**Serial Data Input 0_1**—When programmed as a receiver, SDI0_1 is used to receive serial data into the RX0 serial receive shift register.
PE6	Input, Output, or Disconnected		**Port E6**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant

Table 2-9. Enhanced Serial Audio Interface_1 Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
SDO4_1	Output	GPIO Disconnected	**Serial Data Output 4_1**—When programmed as a transmitter, SDO4_1 is used to transmit data from the TX4 serial transmit shift register.
SDI1_1	Input		**Serial Data Input 1_1**—When programmed as a receiver, SDI1_1 is used to receive serial data into the RX1 serial receive shift register.
PE7	Input, Output, or Disconnected		**Port E7**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SDO3_1	Output	GPIO Disconnected	**Serial Data Output 3**—When programmed as a transmitter, SDO3_1 is used to transmit data from the TX3 serial transmit shift register.
SDI2_1	Input		**Serial Data Input 2**—When programmed as a receiver, SDI2_1 is used to receive serial data into the RX2 serial receive shift register.
PE8	Input, Output, or Disconnected		**Port E8**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SDO2_1	Output	GPIO Disconnected	**Serial Data Output 2**—When programmed as a transmitter, SDO2_1 is used to transmit data from the TX2 serial transmit shift register.
SDI3_1	Input		**Serial Data Input 3**—When programmed as a receiver, SDI3_1 is used to receive serial data into the RX3 serial receive shift register.
PE9	Input, Output, or Disconnected		**Port E9**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.
SDO1_1	Output	GPIO Disconnected	**Serial Data Output 1**—SDO1_1 is used to transmit data from the TX1 serial transmit shift register.
PE10	Input, Output, or Disconnected		**Port E10**—When the ESAI is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

Table 2-9. Enhanced Serial Audio Interface_1 Signals (continued)

Signal Name	Signal Type	State during Reset	Signal Description
SDO0_1 PE11	Output Input, Output, or Disconnected	GPIO Disconnected	**Serial Data Output 0**—SDO0_1 is used to transmit data from the TX0 serial transmit shift register. **Port E11**—When the ESAI_1 is configured as GPIO, this signal is individually programmable as input, output, or internally disconnected. The default state after reset is GPIO disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant.

2.10 Dedicated GPIO - Port G

Table 2-10. Dedicated GPIO - Port G Signals

Signal Name	Type	State During Reset	Signal Description
PG0	Input, Output, or Disconnected	GPIO Disconnected	**Port G0**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG1	Input, Output, or Disconnected	GPIO Disconnected	**Port G1**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG2	Input, Output, or Disconnected	GPIO Disconnected	**Port G2**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG3	Input, Output, or Disconnected	GPIO Disconnected	**Port G3**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG4	Input, Output, or Disconnected	GPIO Disconnected	**Port G4**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG5	Input, Output, or Disconnected	GPIO Disconnected	**Port G5**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG6	Input, Output, or Disconnected	GPIO Disconnected	**Port G6**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant

Table 2-10. Dedicated GPIO - Port G Signals (continued)

Signal Name	Type	State During Reset	Signal Description
PG7	Input, Output, or Disconnected	GPIO Disconnected	**Port G7**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG8	Input, Output, or Disconnected	GPIO Disconnected	**Port G8**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG9	Input, Output, or Disconnected	GPIO Disconnected	**Port G9**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG10	Input, Output, or Disconnected	GPIO Disconnected	**Port G10**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG11	Input, Output, or Disconnected	GPIO Disconnected	**Port G11**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG12	Input, Output, or Disconnected	GPIO Disconnected	**Port G12**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG13	Input, Output, or Disconnected	GPIO Disconnected	**Port G13**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant
PG14	Input, Output, or Disconnected	GPIO Disconnected	**Port G14**—This signal is individually programmable as input, output, or internally disconnected. This pin has an internal pull-down resistor. This input is 5 V tolerant

2.11 Timer

Table 2-11. Timer Signal

Signal Name	Type	State during Reset	Signal Description
TIO0	Input or Output	GPIO Input	**Timer 0 Input/Output**—When timer 0 functions as an external event counter or in measurement mode, TIO0 is used as input. When timer 0 functions in watchdog, timer, or pulse modulation mode, TIO0 is used as output. The default mode after reset is GPIO input. This can be changed to output or configured as a timer input/output through the timer 0 control/status register (TCSR0). If TIO0 is not being used, it is recommended to either define it as GPIO output immediately at the beginning of operation or leave it defined as GPIO input. This pin has an internal pull-down resistor. This input is 5 V tolerant
TIO1	Input or Output	Watchdog Timer Output	**Timer 1 Input/Output**—When timer 1 functions as an external event counter or in measurement mode, TIO1 is used as input. When timer 1 functions in watchdog, timer, or pulse modulation mode, TIO1 is used as output. The default mode after reset is GPIO input. This can be changed to output or configured as a timer input/output through the timer 1control/status register (TCSR1). If TIO1 is not being used, it is recommended to either define it as GPIO output immediately at the beginning of operation or leave it defined as GPIO input.
WDT	Output		**WDT**—When this pin is configured as a hardware watchdog timer pin, this signal is asserted low when the hardware watchdog timer counts down to zero. This pin has an internal pull-down resistor. This input is 5 V tolerant
TIO2	Input or Output	PLOCK Output	**Timer 2 Input/Output**—When timer 2 functions as an external event counter or in measurement mode, TIO2 is used as input. When timer 2 functions in watchdog, timer, or pulse modulation mode, TIO2 is used as output. The default mode after reset is GPIO input. This can be changed to output or configured as a timer input/output through the timer control/status register (TCSR2). If TIO2 is not being used, it is recommended to either define it as GPIO output immediately at the beginning of operation or leave it defined as GPIO input.
PLOCK	Output		**PLOCK**—When this pin is configured as a PLL lock pin, this signal is asserted high when the on-chip PLL enabled and locked and de-asserted when the PLL enabled and unlocked. This pin is also asserted high when the PLL is disabled. This pin has an internal pull-down resistor. This input is 5 V tolerant

2.12 JTAG/OnCE Interface

Table 2-12. JTAG/OnCE Interface

Signal Name	Signal Type	State during Reset	Signal Description
TCK	Input	Input	**Test Clock**—TCK is a test clock input signal used to synchronize the JTAG test logic. This pin has an internal pull-up resistor. This input is 5 V tolerant.
TDI	Input	Input	**Test Data Input**—TDI is a test data serial input signal used for test instructions and data. TDI is sampled on the rising edge of TCK. This pin has an internal pull-up resistor. This input is 5 V tolerant.
TDO	Output	Tri-stated	**Test Data Output**—TDO is a test data serial output signal used for test instructions and data. TDO is tri-statable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TMS	Input	Input	**Test Mode Select**—TMS is an input signal used to sequence the test controller's state machine. TMS is sampled on the rising edge of TCK. This pin has an internal pull-up resistor. This input is 5 V tolerant.

Notes

Chapter 3
Memory Configuration

3.1 Data and Program Memory Maps

The DSP56374 provides 18k words of RAM divided between three memory spaces (X, Y, and P). The default memory allocation and memory block sizes are as follows (See Figure 3-1):

- Program RAM - 6k words (3 - 2k blocks) in the lowest memory addresses between $000000 - $0007FF.
- XRAM - 4k words (1 - 4k block) in the lowest memory addresses between $000000 - $000FFF and 2k words (1 - 2k block) in the memory addresses between $001000 - $0017FF.
- YRAM - 2k words (1 - 2k block) in the lowest memory addresses between $000000 - $0007FF and 4k words (4 - 1k blocks) in the memory addresses between $000800 - $0017FF.

The DSP56374 provides 28k words of ROM divided between three memory spaces (X, Y, and P). The memory allocations are as follows:

- Program ROM - 20k words.
- XROM - 4k words.
- YROM - 4k words.

The on-chip memory configuration of the DSP56374 is affected by the state of the MSW0, MSW1 and MS (Memory Switch) control bits in the OMR register in the Status Register. The internal data and program memory configurations are shown in Table 3-7. The address ranges for the internal memory are shown in Table 3-2. The memory maps for each memory configuration are shown in Figure 3-1 to Figure 3-5.

Table 3-1. Internal Memory Configuration

Program RAM	Program ROM	X Data RAM	X Data ROM	Y Data RAM	Y Data ROM
6k (3-2k blocks) $000000-$017FF	20k (20k blocks)	6k (1-4k block) $000000-$00FFF (1- 2k block) $001000-$017FF	4k (4k blocks)	6k (1-2k block) $000000-$007FF (4- 1k block) $000800-$017FF	4k (4k blocks)

Table 3-2. Internal Memory Locations

Program RAM	Program ROM	X Data RAM	X Data ROM	Y Data RAM	Y Data ROM
6k Words $000000 - $0017FF	20k Words $FF0000 - $FF4FFF	6k Words $000000 - $0017FF	4k Words $004000 - $004FFF	6k Words $000000 - $0017FF	4k Words $004000 - $004FFF

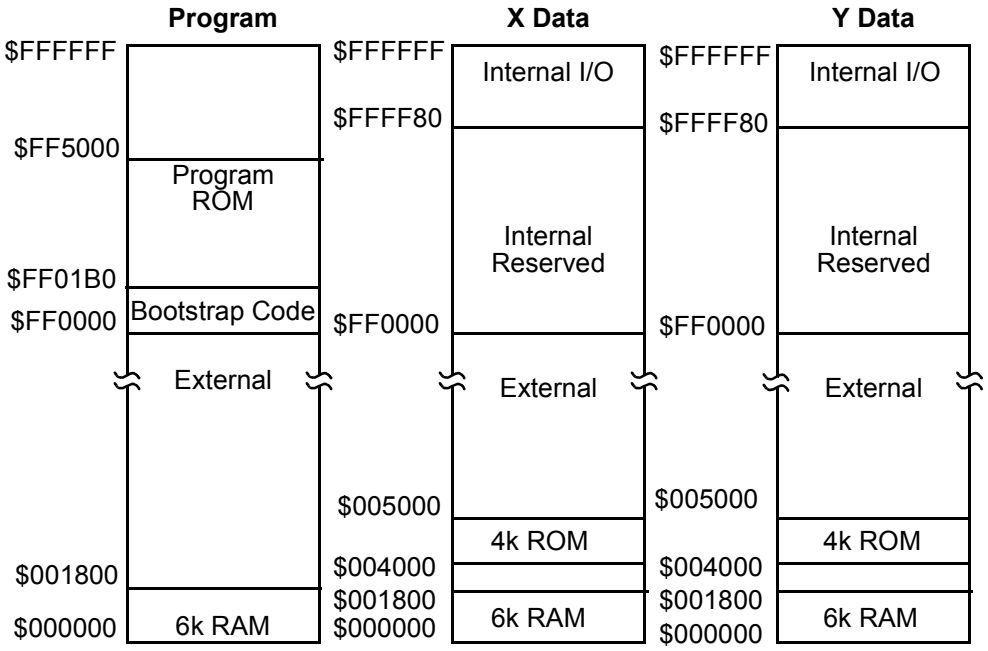


Figure 3-1. Default Memory Map (MS 0)

Table 3-3. Internal Memory Locations

Program RAM	Program ROM	X Data RAM	X Data ROM	Y Data RAM	Y Data ROM
10k Words $000000 - $0027FF	20k Words $FF0000 - $FF4FFF	4k Words $000000 - $000FFF	4k Words $004000 - $004FFF	4k Words $000000 - $000FFF	4k Words $004000 - $004FFF

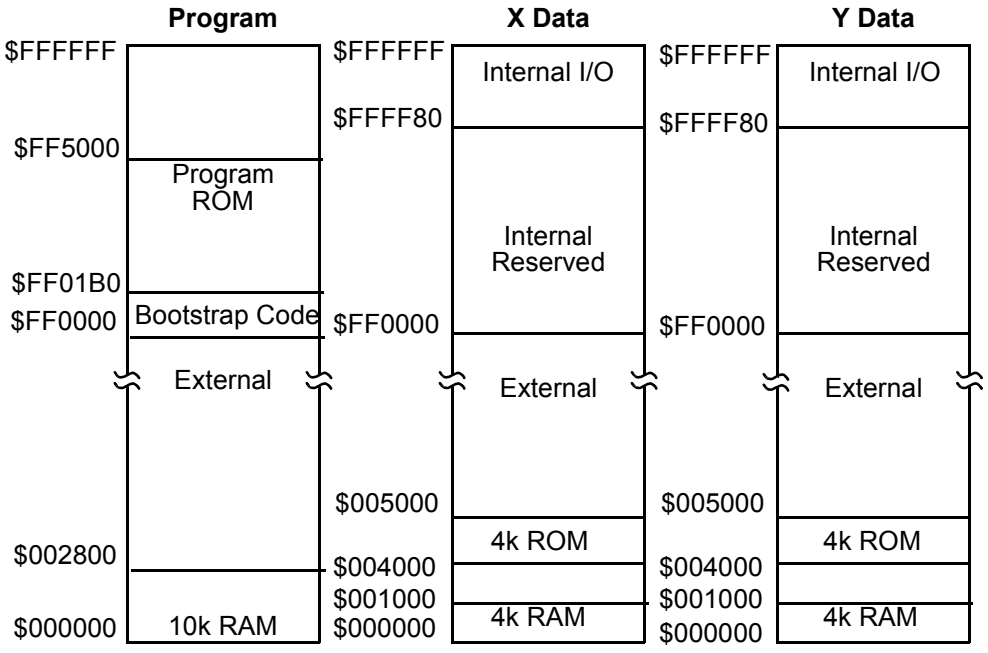


Figure 3-2. Memory Map (MS 1, MSW(1-0) 11)

Table 3-4. Internal Memory Locations

Program RAM	Program ROM	X Data RAM	X Data ROM	Y Data RAM	Y Data ROM
8k Words $000000 - $001FFF	20k Words $FF0000 - $FF4FFF	4k Words $000000 - $000FFF	4k Words $004000 - $004FFF	6k Words $000000 - $0017FF	4k Words $004000 - $004FFF

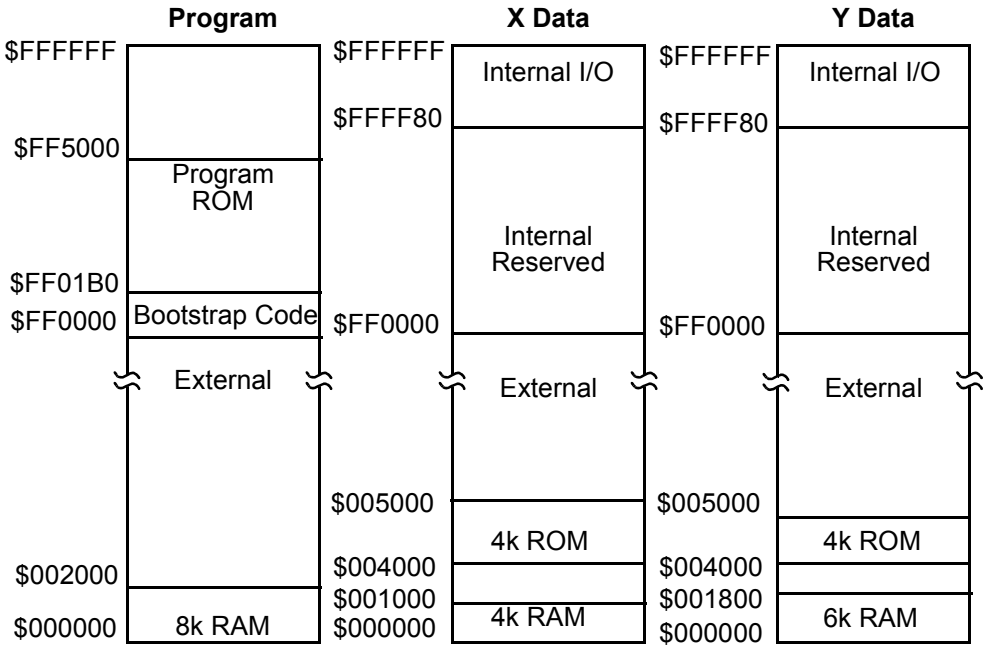


Figure 3-3. Memory Map (MS 1, MSW(1-0) 10)

Table 3-5. Internal Memory Locations

Program RAM	Program ROM	X Data RAM	X Data ROM	Y Data RAM	Y Data ROM
4k Words $000000 - $000FFF	20k Words $FF0000 - $FF4FFF	8k Words $000000 - $001FFF	4k Words $004000 - $004FFF	6k Words $000000 - $0017FF	4k Words $004000 - $004FFF

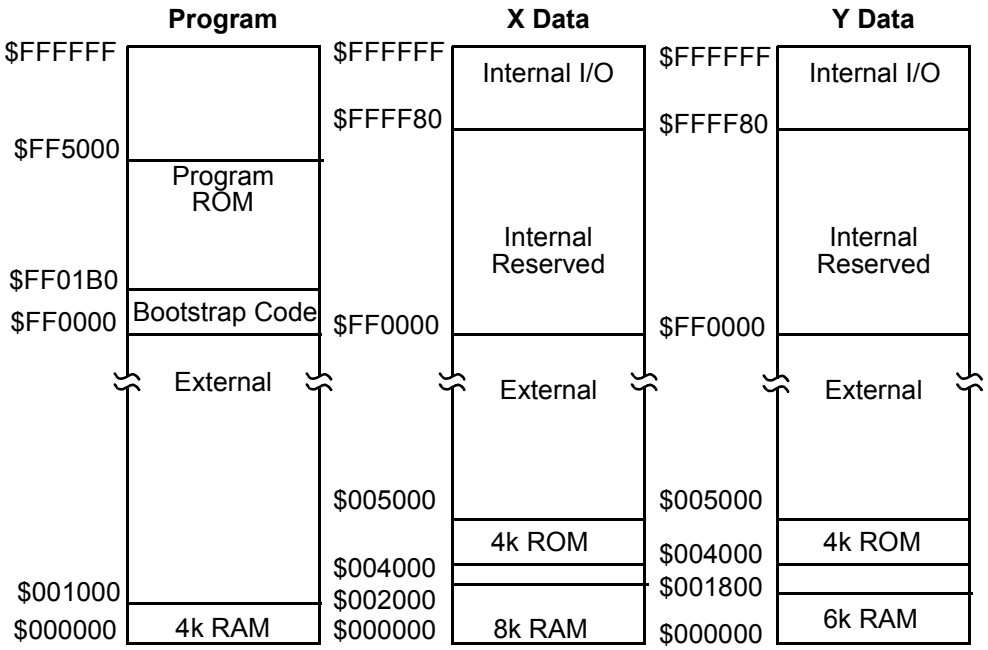


Figure 3-4. Memory Map (MS 1, MSW(1-0) 01)

Table 3-6. Internal Memory Locations

Program RAM	Program ROM	X Data RAM	X Data ROM	Y Data RAM	Y Data ROM
2k Words $000000 - $0007FF	20k Words $FF0000 - $FF4FFF	10k Words $000000 - $0027FF	4k Words $004000 - $004FFF	6k Words $000000 - $0017FF	4k Words $004000 - $004FFF

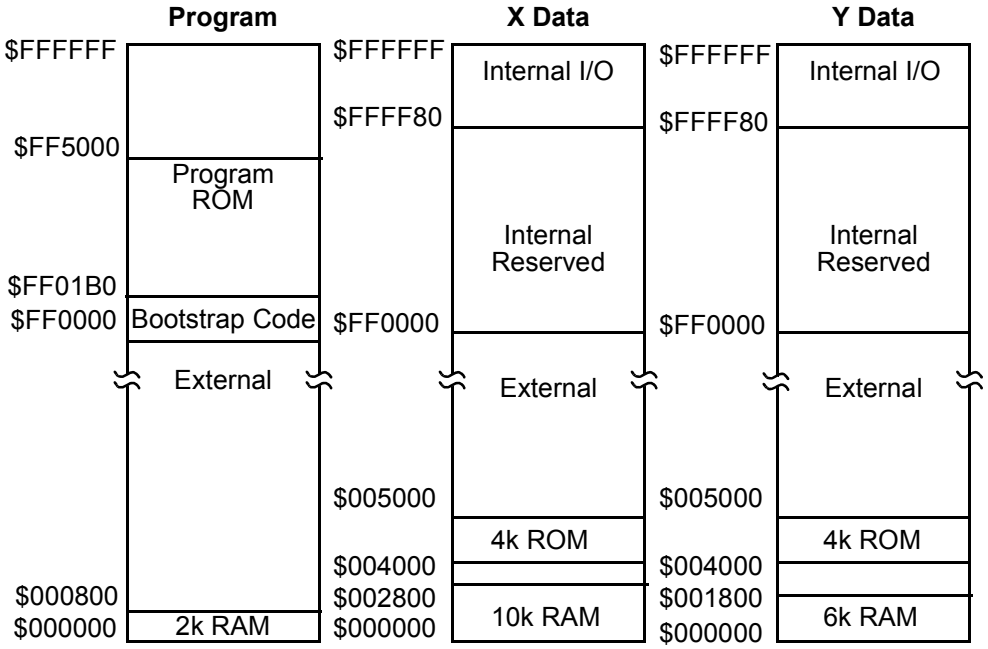


Figure 3-5. Memory Map (MS 1, MSW(1-0) 00)

3.1.1 Reserved Memory Spaces

The reserved memory spaces should not be accessed by the user. They are reserved for future expansion.

3.1.2 Bootstrap CODE

The bootstrap ROM is factory-programmed to perform the bootstrap operation following hardware reset. The Bootstrap source code is available in Appendix A, *Bootstrap Source Code*.

3.1.3 Dynamic Memory Configuration Switching

The internal memory configuration is altered by re-mapping RAM blocks from Y and X data memory into program memory space. The contents of the switched RAM blocks are preserved.

The memory can be dynamically switched from one configuration to another by changing the MS, MSW0 or MSW1 bits in OMR. The address ranges that are directly affected by the switch operation are specified in Table 3-2. The memory switch can be accomplished provided that the affected address ranges are not being accessed during the instruction cycle in which the switch operation takes place. Accordingly, the following condition must be observed for trouble-free dynamic switching:

NOTE

No accesses (including instruction fetches) to or from the affected address ranges in program and data memories are allowed during the switch cycle.

The switch cycle actually occurs 3 instruction cycles after the instruction that modifies the MS, MSW0 or MSW1bits.

Any sequence that complies with the switch condition is valid. For example, if the program flow executes in the address range that is not affected by the switch, the switch condition can be met very easily. In this case a switch can be accomplished by just changing the MS, MSW0 or MSW1 bits in OMR in the regular program flow, assuming no accesses to the affected address ranges of the data memory occur up to 3 instructions after the instruction that changes the OMR bit. Special care should be taken in relation to the interrupt vector routines since an interrupt could cause the DSP to fetch instructions out of sequence and might violate the switch condition.

Special attention should be given when running a memory switch routine using the OnCE™ port. Running the switch routine in Trace mode, for example, can cause the switch to complete after the MS bit change while the DSP is in Debug mode. As a result, subsequent instructions might be fetched according to the new memory configuration (after the switch) and, thus, might execute improperly.

Table 3-7. Internal Memory Configurations

Bit Settings			Memory Sizes (24-bit words)					
MSW1	MSW0	MS	Prog RAM	X Data RAM	Y Data RAM	Prog ROM	X Data ROM	Y Data ROM
X	X	0	6k	6K	6	20k	4k	4k
0	0	1	2k	10k	6k	20k	4k	4k
0	1	1	4k	8k	6k	20k	4k	4k
1	0	1	8k	4k	6k	20k	4k	4k
1	1	1	10k	4k	4k	20k	4k	4k

3.1.4 External Memory Support

The DSP56374 is not capable of directly accessing external memory.

3.1.5 DMA and Memory

Memory on the DSP56374 consists of 4 - 1k-word blocks, 5 - 2k-word blocks, and 1 - 4k word block (see Figure 3-1). It is important to understand that the DMA is designed for operation on 1k-word blocks. The implications are as follows:

When DMA accesses any of the 4 - 1k-word blocks (i.e., y:$000800..$0017FF), hardware will prevent potential loss of DMA accesses if the core happens to simultaneously access the same 1k-word block. If software does not prevent core/DMA access to the same 1k word block, hardware will delay the DMA access until the core has completed its operation. The DMA will continue to operate the expected access.

When DMA accesses any of the 5 - 2k-word blocks or 1 - 4k word block software should be written to prevent contention where hardware protection is not available. Protection for Core/DMA contention is only provided in a 2k or 4k word block if both the core and DMA are

accessing the same 1k section of the block. However, if DMA access is executed to a 2k or 4k word block while the core simultaneously accesses the same block, hardware protection may not be provided and DMA operation may not be performed during the core access. The DMA access may be missed. Software should be written to prevent contention where hardware protection is not available.

3.1.6 Memory BLOCKS

The RAM memory is implemented with a combination of 1k-word, 2k-words and 4k-word RAM memory blocks. The finer granularity of the 1k-word memory blocks permit DMA and core accesses to the same memory spaces with less possibility of contention. See Section 3.1.5, *DMA and Memory*.

3.2 Memory Patch Module

The patch module provides a means to replace instructions in program ROM with instructions written into the patch module's instruction registers. A program ROM instruction is replaced by storing the ROM memory address in a patch module address register. The new instruction is to be loaded into the corresponding patch module instruction register.

After reset (either hard reset or the RESET instruction), none of the patch module address registers are enabled for patching. The patch module address registers are initially cleared. Whenever an address register is written to, that address register (and corresponding instruction register) is enabled for patching.

Once an address register is enabled for patching, whenever there is an internal program read (by the core) of that address, the contents of the corresponding instruction register are substituted onto the Program Read Data bus, instead of the actual contents of memory at that address.

The Patch module has no effect on DMA accesses, or writes to program memory. It also has no effect if the address programmed is considered an external address.

Note that writing to an instruction register does not enable it for patching. Only writing to an address register enables patching. Therefore, if not used for patching, the instruction registers can be used as general purpose registers.

Patching example:

```
Begin

    bset   #23,omr                ; enable patch

    ; initialize patch module registers
    ; addresses first

    move      #$fff000,r1
    movep     r1,y:$ffffa0        ;load ROM address $fff000 into patch address 0
    move      #$fff001,r1
    movep     r1,y:$ffffa1        ;load ROM address $fff001 into patch address 1
    move      #$fff003,r1
    movep     r1,y:$ffffa2        ;load ROM address $fff003 into patch address 2

    ; instructions

    move      #patch_pattern,r2
    movep     p:(r2)+,y:$ffffa8   ;load instruction #1 into patch instruction 0
    movep     p:(r2)+,y:$ffffa9   ;load instruction #2 into patch instruction 1
    movep     p:(r2)+,y:$ffffaa   ;load instruction #3 into patch instruction 2
    jmp       program             ;start running program

       org    P:$800

patch_pattern

       bset   #0,a                ;instruction #1 = bset  #0,a
       bset   #1,a                ;instruction #2 = bset  #1,a
       bset   #2,a                ;instruction #3 = bset  #2,a
```

Anytime the program control unit fetches an instruction from P:$fff000, instruction 1 (bset #0,a) will be fetched in place of the instruction stored in ROM location P:$fff000. Also, when the program control unit fetches an instruction from P:$fff001, instruction 2 (bset #1,a) will be fetched instead. Also, when the program control unit fetches an instruction from P:$fff003, instruction 3 (bset #2,a) will be fetched instead.

3.3 Internal I/O Memory Map

The DSP56374 on-chip peripheral modules have their register files programmed to the addresses in the internal X-I/O memory range (the top 128 locations of the X data memory space) and internal Y-I/O memory range (48 locations of the Y data memory space) as shown in Table C-1 and Table C-2.

Chapter 4
Core Configuration

4.1 Introduction

This chapter contains DSP56300 core configuration information details specific to the SCF5250. These include the following:

- Operating modes
- Bootstrap program
- Interrupt sources and priorities
- DMA request sources
- OMR
- PLL control register
- JTAG

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

4.2 Operating Mode Register (OMR)

The contents of the Operating Mode Register (OMR) are shown in Table 4-1. Refer to the DSP56300 24-Bit Digital Signal Processor Family Manual, Freescale (formerly Motorola) publication DSP56300FM/AD for a description of the OMR bits.

Table 4-1. Operating Mode Register (OMR)

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

MSW1	- Memory Switch Mode 1			MS	- Master Memory Switch Mode
MSW0	- Memory Switch Mode 0			SD	- Stop Delay
SEN	- Stack Extension Enable				
WRP	- Extended Stack Wrap Flag			MD	- Operating Mode D
EOV	- Extended Stack Overflow Flag			MC	- Operating Mode C
EUN	- Extended Stack Underflow Flag	CDP1	- Core-DMA Priority 1	MB	- Operating Mode B
XYS	- Stack Extension Space Select	CDP0	- Core-DMA Priority 0	MA	- Operating Mode A

- Reserved bit. Read as zero, should be written with zero for future compatibility

4.2.1 RESERVED - Bits 4, 5, 10 - 15 and 23

These bits are reserved. They are read as zero and should be written with zero for future compatibility.

4.3 Operating Modes

The operating modes are defined as shown in Table 4-2. The operating modes are latched from MODA, MODB, MODC and MODD pins during reset. Each operating mode is briefly described below. The operation of all modes is defined by the Bootstrap ROM source code in Appendix A, *Bootstrap Source Code*.

Table 4-2. DSP56374 Operating Modes

Mode	MODD	MODC	MODB	MODA	Reset Vector	Description
0	0	0	0	0	$000000	Reserved
1	0	0	0	1	$FF0000	Reserved
2	0	0	1	0	$FF0000	Jump to PROM starting address (slave SPI mode)
3	0	0	1	1	$FF0000	Reserved
4	0	1	0	0	$FF0000	Reserved
5	0	1	0	1	$FF0000	Bootstrap from SHI (slave SPI mode)
6	0	1	1	0	$FF0000	Bootstrap from SHI (slave I^2C mode) (HCKFR=1, 100ns filter enabled)
7	0	1	1	1	$FF0000	Bootstrap from SHI (slave I^2C mode) (HCKFR=0)
8	1	0	0	0	$004000	Reserved
9	1	0	0	1	$FF0000	Bootstrap from SHI (Serial I^2C EEPROM mode) (HCKFR=1, 100ns filter enabled)
A	1	0	1	0	$FF0000	Reserved
B	1	0	1	1	$FF0000	Bootstrap from SHI (Serial SPI EEPROM mode)
C	1	1	0	0	$FF0000	Bootstrap from GPIO (Serial SPI EEPROM mode)
D	1	1	0	1	$FF0000	Jump to PROM at default HLX (slave SPI)
E	1	1	1	0	$FF0000	Jump to PROM starting address (slave I^2C) (HCKFR=0)
F	1	1	1	1	$FF0000	Jump to PROM starting address (slave I^2C) (HCKFR=1, 100ns filter enabled)

Table 4-3. DSP56374 Mode Descriptions

Mode 0 Reserved

Mode 1 Reserved

Mode 2 The DSP starts fetching instructions from the starting address of the on-chip Program ROM. SHI operates in SPI slave mode.

Mode 3 Reserved

Mode 4 Reserved

Mode 5 In this mode, the internal PRAM is loaded from the Serial Host Interface (SHI). The SHI operates in the SPI slave mode, with 24-bit word width.The bootstrap code expects to read a 24-bit word specifying the number of program words, a 24-bit word specifying the address to start loading the program words and then a 24-bit word for each program word to be loaded. The program words will be stored in contiguous PRAM memory locations starting at the specified starting address. After reading the program words, program execution starts from the same address where loading started.

Mode 6 Same as Mode 5 except SHI interface operates in the I^2C slave mode with HCKFR set to 1 and the 100ns filter enabled.

Mode 7 Same as Mode 5 except SHI interface operates in the I^2C slave mode with HCKFR set to 0.

Mode 8 Reserved

Table 4-3. DSP56374 Mode Descriptions

Mode 9	In this mode, the internal memory (PRAM, XRAM, or YRAM) is loaded from an external serial EPROM in I^2C mode withthe 100ns filter enabled. Supports ST M24256 and Atmel AT24C256. [a]
Mode A	Reserved
Mode B	In this mode, the internal memory (PRAM, XRAM, or YRAM) is loaded from an external serial EPROM in SPI mode. Supports ST M95256 and Atmel AT25256.
Mode C	In this mode, the internal memory (PRAM, XRAM, or YRAM) is loaded from an external serial EPROM in SPI mode via GPIO pins (PH0 - Chip Select, PH1 - Data in, PH2 - Data out and PH3 - clock). Supports ST M95256 and Atmel AT25256.
Mode D	The DSP starts operation of the default HLX. SHI operates in SPI slave mode.
Mode E	The DSP starts fetching instructions from the starting address of the on-chip Program ROM. SHI operates in in I^2C mode. No filter in enabled.
Mode F	The DSP starts fetching instructions from the starting address of the on-chip Program ROM. SHI operates in in I^2C mode with the 100ns filter enabled.

[a] See Appendix A, *Bootstrap Source Code* for details on using this boot mode.

4.4 Interrupt Priority Registers

There are two interrupt priority registers in the DSP56374:

1. IPR-C is dedicated for DSP56300 Core interrupt sources.
2. IPR-P is dedicated for DSP56374 Peripheral interrupt sources.

The interrupt priority registers are shown in Table 4-1 and Table 4-2. The Interrupt Priority Level bits are defined in Table 4-4. The interrupt vectors are shown in Table C-3 and the interrupt priorities are shown in Table C-4.

Table 4-4. Interrupt Priority Level Bits

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

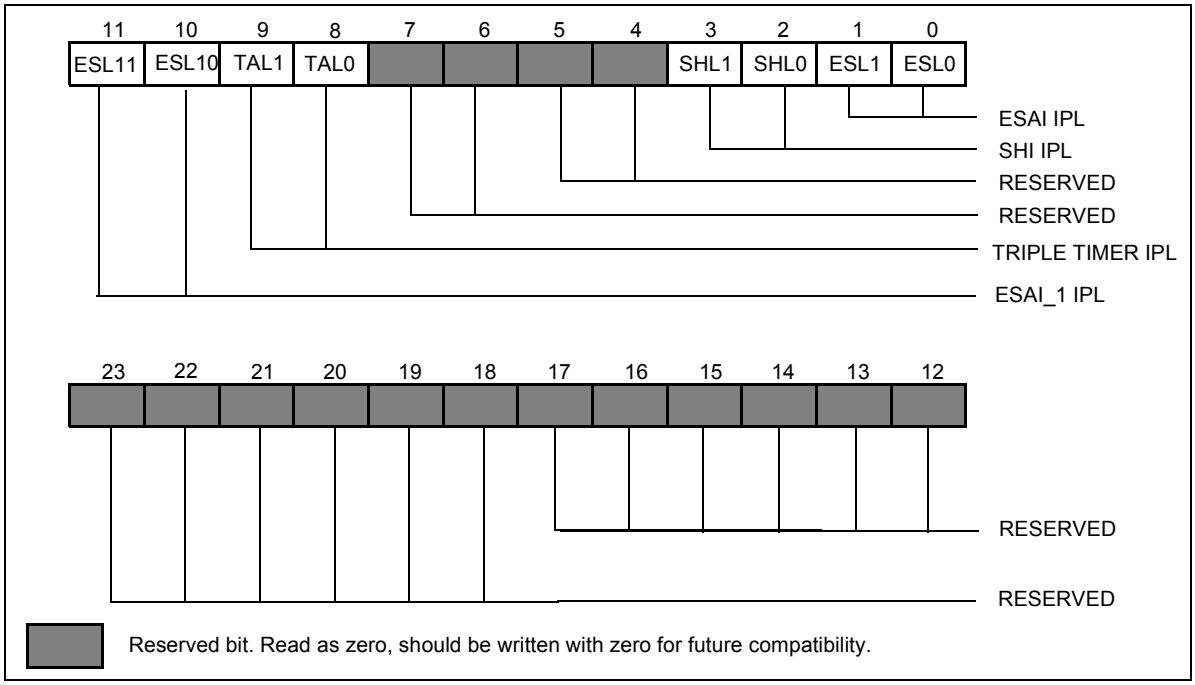


Figure 4-1. Interrupt Priority Register P

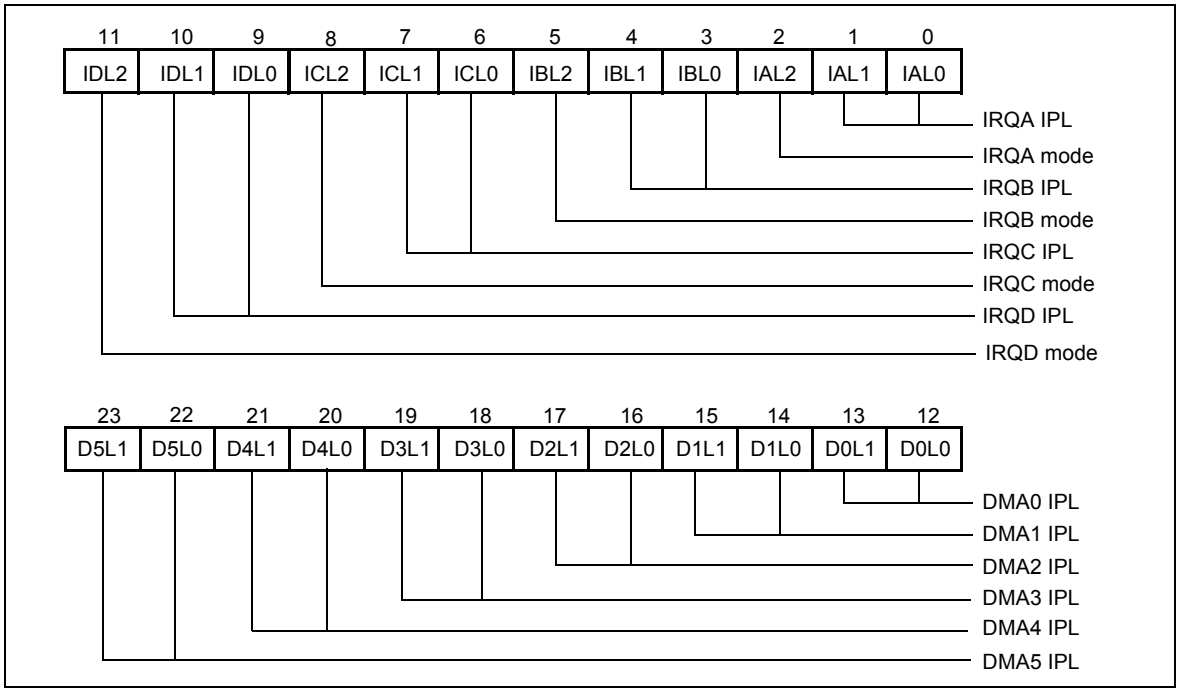


Figure 4-2. Interrupt Priority Register C

4.5 DMA Request Sources

The DMA Request Source bits (DRS4-DRS0 bits in the DMA Control/Status registers) encode the source of DMA requests used to trigger the DMA transfers. The DMA request sources may be the internal peripherals or external devices requesting service through the $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$ and $\overline{\text{IRQD}}$ pins. The DMA Request Sources are shown in Table 4-5.

Table 4-5. DMA Request Sources

DMA Request Source Bits DRS4...DRS0	Requesting Device
00000	External ($\overline{\text{IRQA}}$ pin)
00001	External ($\overline{\text{IRQB}}$ pin)
00010	External ($\overline{\text{IRQC}}$ pin)
00011	External ($\overline{\text{IRQD}}$ pin)
00100	Transfer Done from DMA channel 0
00101	Transfer Done from DMA channel 1
00110	Transfer Done from DMA channel 2
00111	Transfer Done from DMA channel 3
01000	Transfer Done from DMA channel 4
01001	Transfer Done from DMA channel 5
01010	Reserved
01011	ESAI Receive Data (RDF=1)
01100	ESAI Transmit Data (TDE=1)
01101	SHI HTX Empty
01110	SHI FIFO Not Empty
01111	SHI FIFO Full
10000	Reserved
10001	Reserved
10010	TIMER0 (TCF=1)
10011	TIMER1 (TCF=1)
10100	TIMER2 (TCF=1)
10101	ESAI_1 Receive Data (RDF=1)
10110	ESAI_1 Transmit Data (TDE=1)
10111	Reserved
11000	Reserved
11001-11111	Reserved

4.6 PLL Initialization

The following figure displays the PLL control register (PCTL). This register is used to control the PLL operation including its multiplication/divide factors and enabling bits.

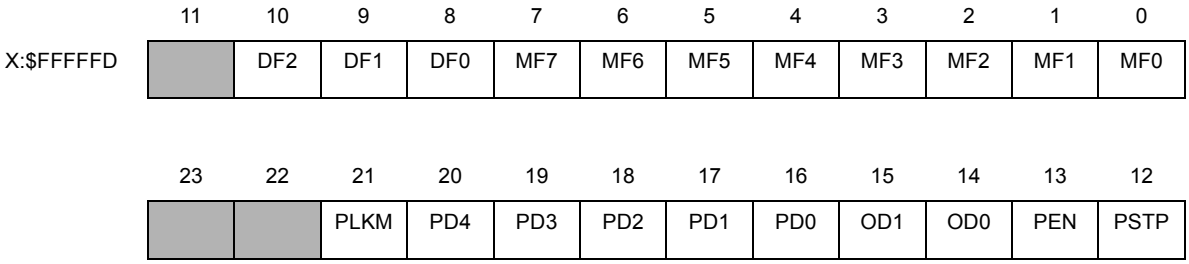


Figure 4-3. PCTL Register

4.6.1 PLL Pre-Divider Factor (PD0-PD4)

The DSP56374 PLL Pre-Divider factor is set to 4 during hardware reset, i.e., the Pre-Divider Factor Bits PD0-PD4 in the PLL Control Register (PCTL) are set to $4.

4.6.2 PLL Multiplication Factor (MF0-MF7)

The DSP56374 PLL multiplication factor is set to 29 during hardware reset, i.e., the Multiplication Factor Bits MF0-MF7 in the PLL Control Register (PCTL) are set to $1D.

4.6.3 PLL Feedback Multiplier (OD1)

The DSP56374 PLL Feedback Multiplier is set to 2 during hardware reset, i.e., OD1 is cleared ($0) in the PLL Control Register (PCTL).

4.6.4 PLL Output Divide Factor (OD0-OD1)

The DSP56374 PLL Output Divider factor is set to 2 during hardware reset, i.e., OD1 is cleared ($0) and OD0 is set ($1) in the PLL Control Register (PCTL).

4.6.5 PLL Divider Factor (DF0-DF2)

The DSP56374 PLL Divider factor is set to 1 during hardware reset, i.e., the Divider Factor Bits DF0-DF2 in the PLL Control Register (PCTL) are set to $0.

4.6.6 PLL LOCK MUX (PLKM)

The PLOCK Mux (PLKM) bit is a read/write bit that controls the operation of the PLOCK/TIO2 pin. When PLKM is set, the PLOCK/TIO2 pin operates as the PLL lock indicator (PLOCK). When the PLKM bit is cleared, the PLOCK/TIO2 pin operates as the TIO2 pin.

Note: The PLKM bit is set during hardware reset.

4.7 Device Identification (ID) Register

The Device Identification Register (IDR) is a 24 bit read only factory programmed register used to identify the different DSP56300 core-based family members located at x:$FFFFF5. This register specifies the derivative number and revision number. This information may be used in testing or by software. Table 4-6 shows the ID register configuration.

Table 4-6. Identification Register Configuration

23 16	15 12	11 0
Reserved	**Revision Number**	**Derivative Number**
$00	$0	$374

4.8 JTAG Identification (ID) Register

The JTAG Identification (ID) Register is a 32 bit, read only thought JTAG, factory programmed register used to distinguish the component on a board according to the IEEE 1149.1 standard. Table 4-7 shows the JTAG ID register configuration.

Table 4-7. JTAG Identification Register Configuration

31 28	27 22	21 12	11 1	0
Version Information	**Customer Part Number**	**Sequence Number**	**Manufacturer Identity**	**1**
0000	000111	0000000011	00000001110	1

Chapter 5
PLL and Clock generator

5.1 Introduction

The DSP56374 core features a Phase Locked Loop (PLL) clock generator in its central processing module. The PLL operation is controlled by a PLL control register (PCTL). The PLL allows the processor to operate at a high internal clock frequency derived from a low-frequency clock input, a feature that offers two immediate benefits. First, the lower frequency clock input can reduce the overall electromagnetic interference generated by a system. Second, the ability to oscillate at different frequencies reduces costs by eliminating the need to add additional oscillators to a system. Figure 5-1 shows the two main blocks of the clock generator in the DSP56374 core:

- Phase Locked Loop (PLL) that performs:
  - Clock input division
  - Frequency multiplication
  - Skew elimination
- Clock Generator (CLKGEN) that performs:
  - Low-power division
  - Internal clock generation

Note that the core is stopped when the PLL is enabled, but unlocked.

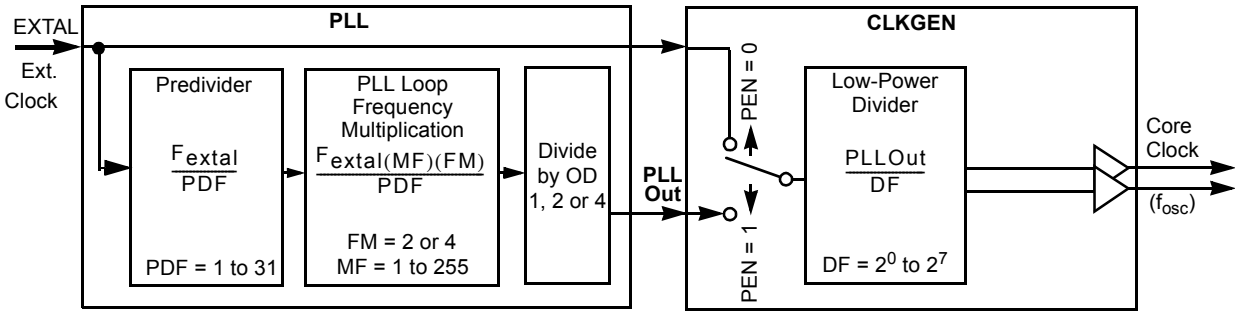


Figure 5-1. PLL Clock Generator Block Diagram

5.2 PLL and Clock Signals

The PLL and clock pin configuration for the DSP56374 is available in the device-specific technical data sheet. The following pins are dedicated to the PLL and clock operation:

- PINIT: During assertion of hardware reset, the value of the PINIT input pin is written into the PCTL PLL Enable (PEN) bit. After hardware reset is de-asserted, the PLL ignores the PINIT pin. The default PCTL setting when PINIT is asserted is: $04601D.
- EXTAL: An external clock is required to drive the DSP. The external clock is input via the EXTAL pin passing the clock through the PLL and Clock generator for optional frequency multiplication.
- XTAL: An external crystal between the values of 10MHz and 25MHz can be driven from the XTAL pin. The external crystal should be connected to both the XTAL and EXTAL pins to provide the source clock frequency.
- TIO2/PLOCK: PLOCK is muxed with the TIO2 pin and operates as a PLOCK pin upon exiting reset. When the PLKM bit is cleared, the TIO2/PLOCK pin operates as a timer pin. When the PLKM bit is set, the TIO2/PLOCK pin operates as a PLOCK pin. When operating as PLOCK, the following applies: The PLOCK pin is asserted (high) when the PLL is enabled and has locked on the proper phase and frequency of EXTAL (maximum lock time is 0.5ms). The PLOCK output is de-asserted (low) if the PLL is enabled and is not locked on the proper phase and frequency. PLOCK is asserted if the PLL is disabled. PLOCK should be a reliable indicator of the PLL lock state after exiting the hardware reset state.

5.3 PLL Block

This section describes the PLL control components and operation. Figure 5-2 shows the PLL block diagram.

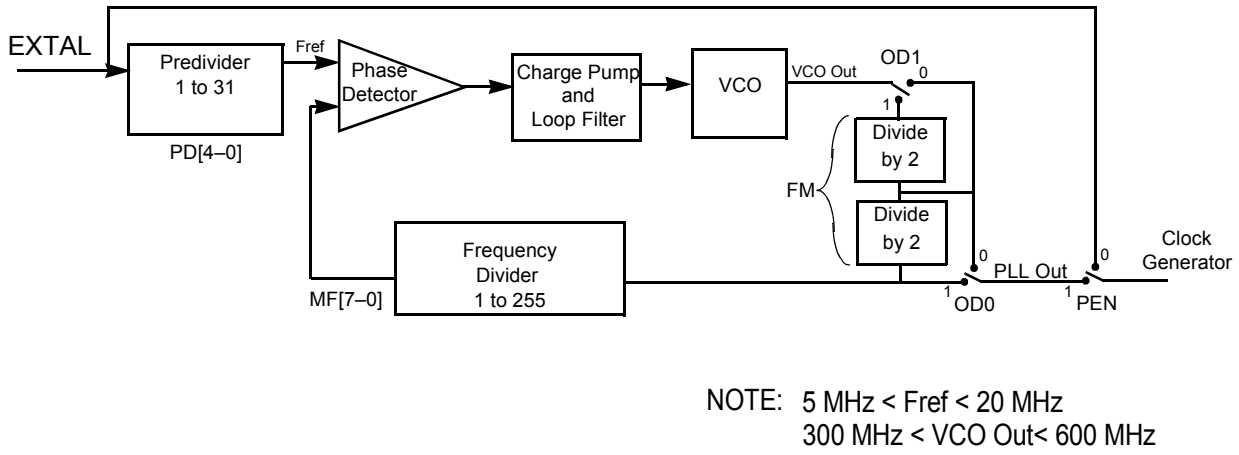


Figure 5-2. PLL Block Diagram

5.3.1 Frequency Predivider

Clock input frequency division is accomplished by means of a frequency predivider of the input frequency. The pre-divider ranges from 1 to 31. The pre-divider must never be set to zero. The output frequency of the pre-divider (Fref) must be between 5 MHz and 20 MHz to guarantee proper operation.

5.3.2 Phase Detector and Charge Pump Loop Filter

The Phase Detector detects any phase difference between the external clock (EXTAL) and the phase of the clock generated by the PLL. At the point where there is negligible phase difference and the frequency of the two inputs is identical, the PLL is in the locked state. The charge pump loop filter receives signals from the Phase Detector and either increases or decreases the voltage applied to the VCO based on the Phase Detector signals.

5.3.3 Voltage Controlled Oscillator (VCO)

The Voltage Controlled Oscillator (VCO) can oscillate at frequencies from 300 MHz to 600 MHz. The VCO output frequency is determined by the voltage applied to it by the charge pump that corresponds to the PLL input frequency (Fref). The VCO frequency is a function of the input frequency as well as the multiplication components (Multiplication factor (MF) and Feedback Multiplier (FM)).

5.3.4 PLL DividerS

As part of the PLL output stage, there are two divide modules (each is a divide by 2 module) controlled by the OD0 and OD1 bits in the PCTL register. These two bits control the PLL *feedback multiplier* (FM) as well as the *output divide* factor (OD). The feedback multiplier is a frequency divider implemented in the PLL feedback loop thus operating as a PLL multiplier and can be programmed to multiply the VCO output frequency up by a factor of 2 or 4. See Table 5-1. The number of divide modules in the PLL loop is determined by the OD1 bit. When one divide module is in the feedback loop (OD1=0) FM = 2. When two divide modules are in the feedback loop (OD1=1) FM = 4. Note that when OD1 is changed, the PLL will lose lock.

Table 5-1. Feedback Multiplier (FM); FM = 2(1 + OD1)

OD1	FM
0	2
1	4

The output divide factor (OD) determines the PLL output frequency as a function of the VCO frequency. The PLL output frequency can be programmed to be the VCO frequency divided by 2 or 4. The output divide factor (OD) is determined by both the OD1 and OD0 bits. See Table 5-2. Note that the PLL will not lose lock when OD0 is changed since OD0 is not in the PLL loop. The PLL will lose lock, however, when OD1 is changed. Also, note that the output divide factor (OD) should not be programmed such that both OD0 = 0 and OD1 = 0.

Table 5-2. Output Divide Factor (OD)

OD0	OD1	OD
0	0	Reserved
0	1	Div by 2
1	0	Div by 2
1	1	Div by 4

5.3.5 PLL Multiplication Factor (MF)

The Frequency Divider portion of the PLL feedback loop divides the VCO output by an additional programmable 8-bit value before entering the Phase Detector. The net result is an additional multiplication of the input clock by the programmed value. This is called the *Multiplication Factor (MF)* and is programmed using the PCTL[MF] bits. The Multiplication Factor can range from 1 to 255. The MF must never be set to zero and although there are 8 bits for programming the MF ranging from 1 to 255, due to VCO output frequency and Fref frequency limitations, the MF should always be programmed between the values of 4 and 60.

5.4 PLL Operation

The PLL uses two major control elements in its circuitry:

- Clock input division
- Frequency multiplication

The following describes the operation of the PLL and its components.

5.4.1 EXTAL Clock Input Division

The PLL can divide the input frequency (EXTAL) by any integer between 1 and 31. The *Division Factor* can be modified by changing the value of the PCTL *Predivider Factor* (PDF) bits (PD[4–0]). The output frequency of the predivider is called the reference frequency (Fref) and is determined using the following formula:

$$Fref = \frac{Fextal}{PDF} \quad ; 5\ MHz < Fref < 20\ MHz$$

5.4.2 PLL Frequency Multiplication

The PLL can multiply the reference frequency by using the MF and FM multipliers in the PLL feedback loop. This is performed by writing to the Multiplication Factor (MF[7–0]) bits and the OD[1] bit (affecting the Feedback Multiplier) in the PCTL register. The output frequency of the VCO (that is, VCO Out as shown in Figure 5-2) is computed using the following formula:

$$VCO\ Out = \frac{Fextal \times MF \times FM}{PDF} \quad ; 300\,MHz < VCO < 600\,MHz$$

The following figures display how the OD1 bit affects the PLL loop and the VCO output. Figure 5-3 shows that when OD1 is clear, only one divider module is in the PLL loop, effectively applying a feedback multiplier of 2. Figure 5-4 shows that when OD1 is set, two divider modules are in the PLL loop, effectively applying a feedback multiplier of 4. Note that, since OD1 is in the closed loop of the PLL, changes to OD1 do cause a loss of lock condition.

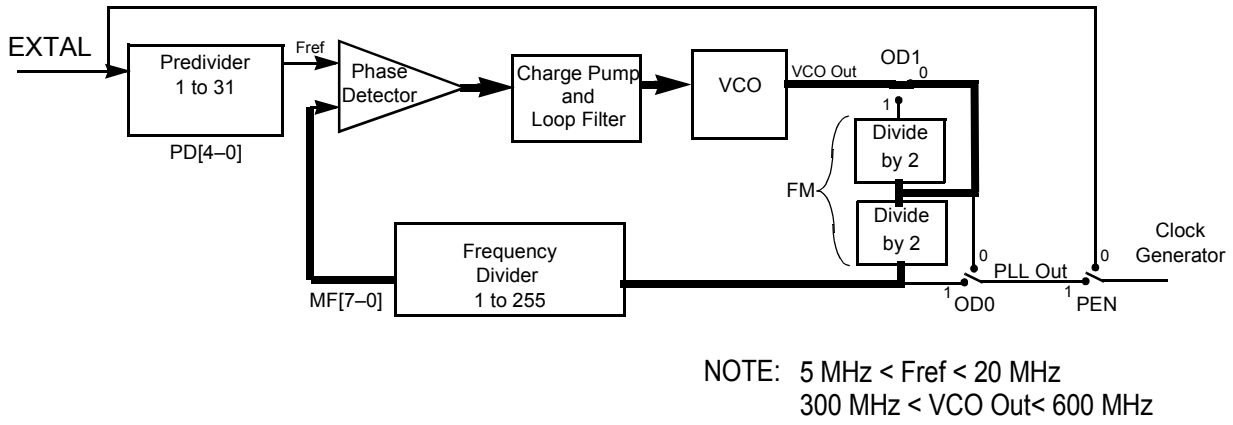


Figure 5-3. PLL Loop with One Divider when OD1=0 (FM = 2)

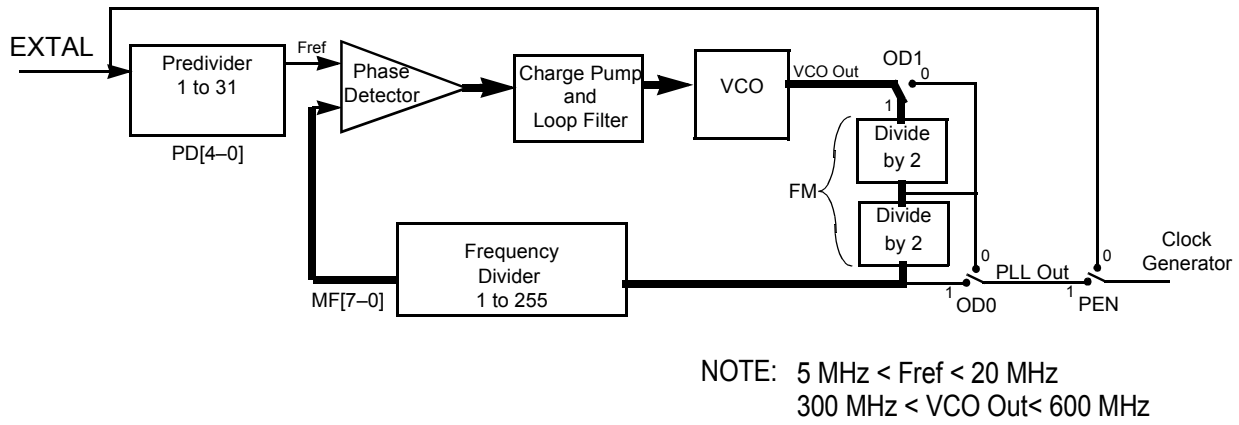


Figure 5-4. PLL Loop with Two Dividers when OD1=1 (FM = 4)

5.4.3 PLL Output Frequency (PLL Out)

The PLL Output frequency is a function of the VCO frequency as follows:

$$\text{PLL Out} = \frac{\text{VCO}}{\text{OD}}$$

As described above the Output Divider Factor is 2 or 4 as determined by the OD1 and OD0 bits. Note that since OD0 is not in the closed loop of the PLL, changes to OD0 do not cause a loss of lock condition. The figures below show how the OD [1-0] bits affect the PLL Output frequency by dividing the VCO Output. Figure 5-5 displays how setting OD1 = 0 and OD0 = 1 divides the VCO output to generate a PLL Output that is VCO Out/2.

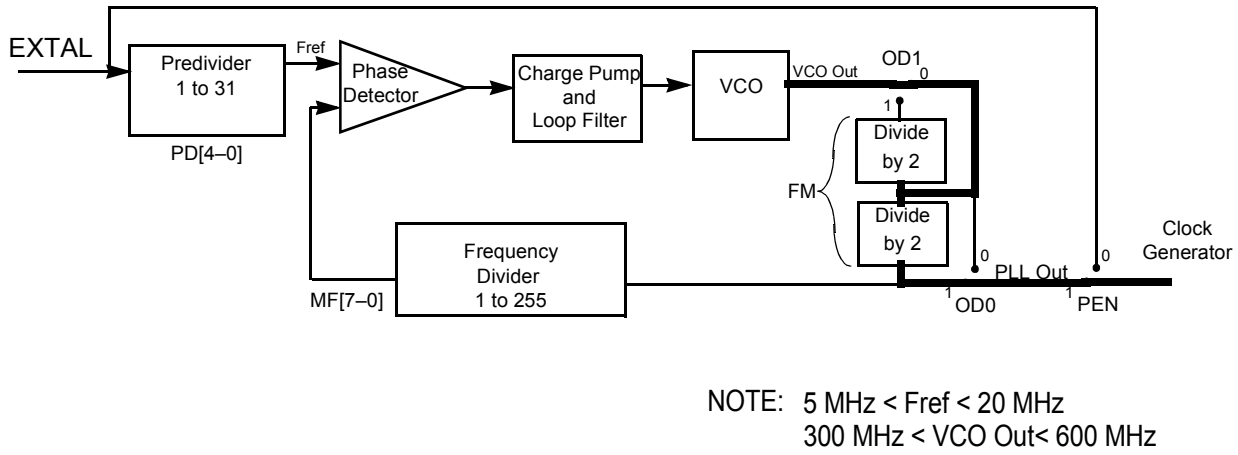


Figure 5-5. PLL Out = VCO Out/2 [OD1 = 0, OD0 = 1]

Figure 5-6 displays how setting OD1 = 1 and OD0 = 0 divides the VCO output to generate a PLL Output that is VCO Out/2.

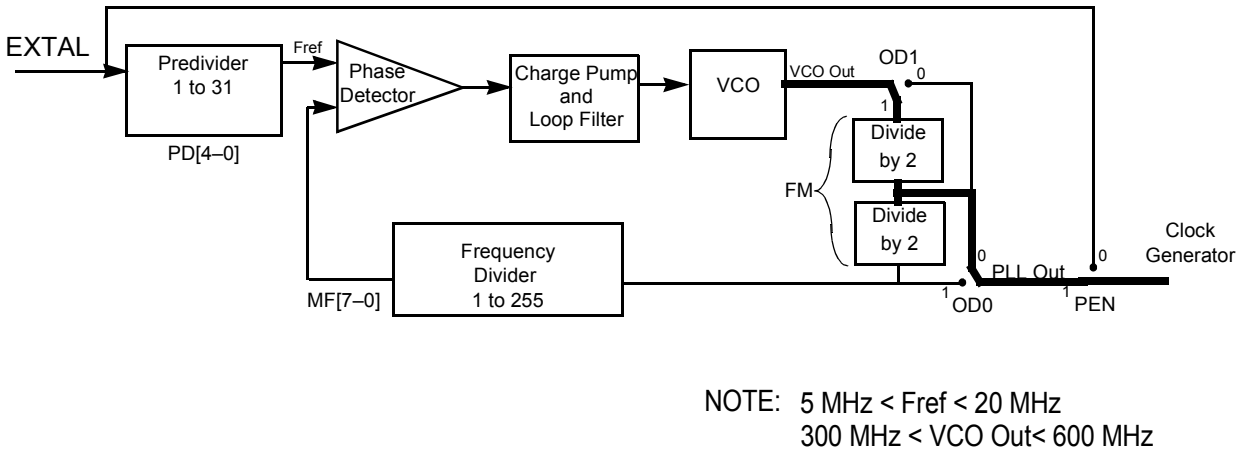


Figure 5-6. PLL Out = VCO Out/2 [OD1 = 1, OD0 = 0]

Figure 5-7 displays how setting OD1 = 1 and OD0 = 1 divides the VCO output to generate a PLL Output that is VCO Out/4.

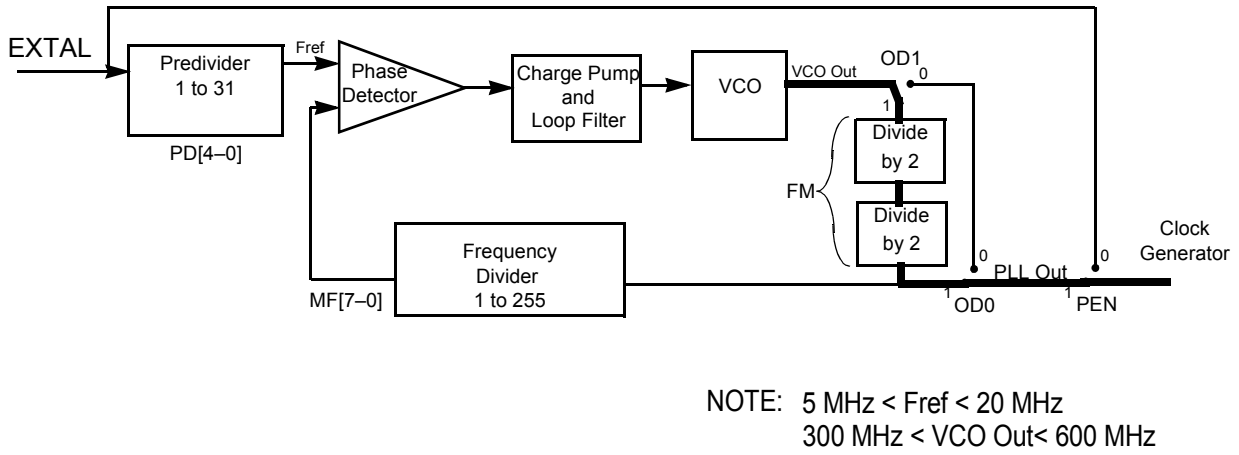


Figure 5-7. PLL Out = VCO Out/4 [OD1 = 1, OD0 = 1]

5.5 Clock Generator

Figure 5-8 shows the Clock Generator block diagram. The components of the Clock Generator are described in the following sections.

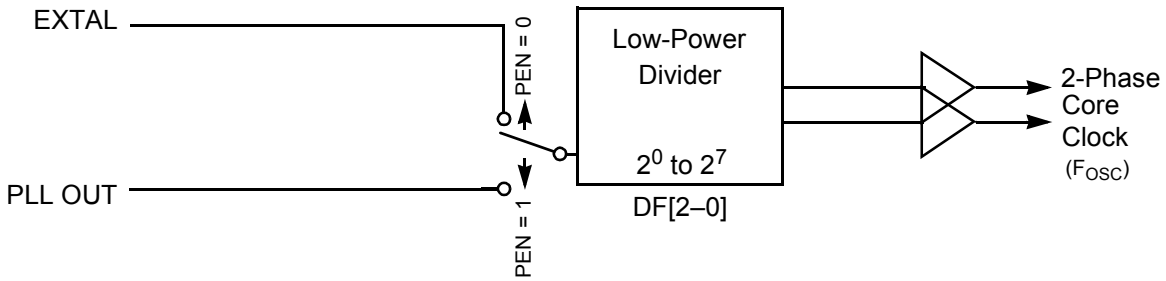


Figure 5-8. CLKGEN Block Diagram

5.5.1 Low-Power Divider (LPD)

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

5.6 Operating Frequency (Fosc)

The output stage of the Clock Generator generates the clock signals to the core and the device peripherals. The input source to the clock generator is selected between:

- EXTAL (PEN = 0, PLL disabled), which generates the device frequency from the EXTAL clock directly.

$$Fosc = \frac{Fextal}{DF}$$

- PLL Output (PEN = 1, PLL enabled), which generates a device frequency defined by the following formula:

$$Fosc = \frac{Fextal \times MF \times FM}{PDF \times DF \times OD}$$

where:

- MF is the Multiplication Factor defined by MF[7–0]
- PDF is the Predivider Factor defined by PD[4–0]
- DF is the Division Factor defined by DF[2–0]
- OD is the Output Divide Factor defined by OD[1-0].
- FM is the Feedback Multiplication Factor defined by OD[1]
- F_{OSC} is the device operating frequency
- F_{EXTAL} is the external EXTAL input

5.7 PLL Programming Model

The PLL clock generator uses a single register, the PCTL Register, for PLL control. The PCTL is an X I/O mapped 24-bit read/write register used to direct the operation of the on-chip PLL.
Figure 5-9 shows the PCTL control bits. The PCTL bits are described in Table 5-3.

23	22	21	20	19	18	17	16	15	14	13	12
		PLKM	PD4	PD3	PD2	PD1	PD0	OD1	OD0	PEN	PSTP
Reset:											
			0	0	1	0	0	0	1	a	0
11	10	9	8	7	6	5	4	3	2	1	0
	DF2	DF1	DF0	MF7	MF6	MF5	MF4	MF3	MF2	MF1	MF0
Reset:											
	0	0	0	0	0	0	1	1	1	0	1

a The reset value of the PEN bit is based on the value of the PLL PINIT input.

Figure 5-9. PLL Control (PCTL) Register

Table 5-3. PLL Control (PCTL) Register Bit Definitions

Bit Number	Bit Name	Reset Value	Description
21	PLKM	$0	PLL LOCK MUX The PLOCK Mux (PLKM) bit is a read/write bit that controls the operation of the PLOCK/TIO2 pin. When PLKM is set, the PLOCK/TIO2 pin operates as the PLL lock indicator (PLOCK). When the PLKM bit is cleared, the PLOCK/TIO2 pin operates as the TIO2 pin. **NOTE:** The PLKM bit is set during hardware reset.
20–16	PD[4–0]	$4	**Predivider Factor** Defines the PDF value that is applied to the input frequency. PDF can be any integer from 1 to 31. The VCO is a function of PDF and oscillates at a frequency defined by the following formula: $\frac{(Fextal \times MF \times FM)}{PDF}$ PDF must be chosen to ensure that Fref lies in a range specified in the device-specific technical data sheet (5 MHz - 20 MHz) and the resulting VCO output frequency lies in the range specified in the device-specific technical data sheet (300 MHz - 600 MHz). Any time a new value is written into the PD[4–0] bits, the PLL loses the lock condition. The PDF bits (PD[4–0]) are set to $4 during hardware reset. The PDF value should never be set to $0.
15–14	OD[1–0]	01	**Output Divider Factor and Feedback Multiplier** Defines the OD and FM values that are applied to the output VCO frequency. The VCO oscillates at a frequency defined by the following formula: $\frac{(Fextal \times MF \times FM)}{PDF}$ FM = 2(1 + OD1). OD1 must be chosen to ensure that the resulting VCO output frequency lies in the range specified in the device-specific technical data sheet (300 MHz - 600 MHz). Any time OD1 is changed, the PLL loses the lock condition. OD1 is initially cleared (0) following reset. OD0 is initially set (1) following reset. Changes to OD0 do not cause the PLL to lose the lock condition. OD0 and OD1 bits together define the output divide factor (OD). The output divide factor divides the VCO output frequency by a factor of 2 or 4 according to Table 5-4.

Table 5-4. Output Divide Factor (OD)

OD0	OD1	OD
0	0	Reserved
0	1	Div by 2
1	0	Div by 2
1	1	Div by 4

Note that OD0 and OD1 should not simultaneously be cleared. The resulting Fosc frequency will exceed the maximum operating frequency when in this case. The PLL Output is defined by the following formula when OD = 1:

$$\frac{VCO\ Out}{OD}$$

Table 5-3. PLL Control (PCTL) Register Bit Definitions (continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th>Description</th></tr>
<tr><td>13</td><td>PEN</td><td>a</td><td>PLL Enable
Enables PLL operation. When PEN is set, the PLL is enabled and the internal clocks are derived from the PLL VCO output. When PEN is cleared, the PLL is disabled and the internal clocks are derived directly from the EXTAL signal. When the PLL is disabled, the VCO stops to minimize power consumption. The PEN bit may be set or cleared by software any time during the device operation. During hardware reset, this bit is set or cleared based on the value of the PLL PINIT input. Note that the core is stopped when the PLL is enabled, but unlocked.</td></tr>
<tr><td>12</td><td>PSTP</td><td>0</td><td>PLL Stop State
Controls PLL and on-chip crystal oscillator behavior during the Stop processing state. When PSTP is set, the PLL remains operating when the chip is in the Stop state. When PSTP is cleared and the device enters the Stop state, the PLL is disabled, to further reduce power consumption. This however results in longer recovery time upon exit from the Stop state. To enable rapid recovery when exiting the Stop state (but at the cost of higher power consumption during the Stop state), PSTP should be set.
<table>
<tr><th rowspan="2">PSTP</th><th rowspan="2">PEN</th><th colspan="2">Operation During Stop State</th><th rowspan="2">Recovery Time From Stop State</th><th rowspan="2">Power Consumption During Stop State</th></tr>
<tr><th>PLL</th><th>Oscillator</th></tr>
<tr><td>0</td><td>x</td><td>Disabled</td><td>Disabled</td><td>Long</td><td>Minimal</td></tr>
<tr><td>1</td><td>0</td><td>Disabled</td><td>Enabled</td><td>Short</td><td>Lower</td></tr>
<tr><td>1</td><td>1</td><td>Enabled</td><td>Enabled</td><td>Short</td><td>Higher</td></tr>
</table></td></tr>
<tr><td>10–8</td><td>DF[2–0]</td><td>0</td><td>Division Factor
Define the DF of the low-power divider. These bits specify the DF as a power of two in the range from 2^0 to 2^7. Changing the value of the DF[2–0] bits does not cause a loss of lock condition. Whenever possible, changes of the operating frequency of the device (for example, to enter a low-power mode) should be made by changing the value of the DF[2–0] bits rather than changing the MF[7–0] bits.
<table>
<tr><td>DF[2–0]</td><td>DF Value</td></tr>
<tr><td>000</td><td>$2^0 = 1$</td></tr>
<tr><td>001</td><td>$2^1 = 2$</td></tr>
<tr><td>010</td><td>$2^2 = 4$</td></tr>
<tr><td>011</td><td>$2^3 = 8$</td></tr>
<tr><td>100</td><td>$2^4 = 16$</td></tr>
<tr><td>101</td><td>$2^5 = 32$</td></tr>
<tr><td>110</td><td>$2^6 = 64$</td></tr>
<tr><td>111</td><td>$2^7 = 128$</td></tr>
</table></td></tr>
</table>

Table 5-3. PLL Control (PCTL) Register Bit Definitions (continued)

Bit Number	Bit Name	Reset Value	Description
7–0	MF[7–0]	$1D	**Multiplication Factor** Defines the Multiplication Factor (MF) that is applied to the PLL input frequency. The MF can be any integer from 1 to 255. The VCO oscillates at a frequency defined by the following formula where PDF is the Predivider Division Factor and FM is the Feedback Multiplier: $$\frac{(Fextal \times MF \times FM)}{PDF}$$ The MF must be chosen to ensure that the resulting VCO output frequency is in the range specified in the device-specific technical data sheet (300 MHz - 600 MHz). Any time a new value is written into the MF[7–0] bits, the PLL loses the lock condition. The Multiplication Factor bits MF[7–0] are set to $1D (29) during hardware reset.
a	The reset value of the PEN bit is based on the value of the PLL PINIT input		

5.8 PLL Initialization Procedure

The DSP56374 PLL is programmed via the PCTL register. Unlike the DSP56371, the DSP56374 does not require a two step initialization process. However, the DSP56374 PLL is backwards compatible with the DSP56371 and will support the two step initialization process. The following programming example illustrates the initialization process.

PLL Programming Example:

Input Frequency (EXTAL) - 24.576 MHz

Target Operating Frequency - 150 MHz

Program the PLL control register (PCTL) - $23E012.

- This enables the TIO2/PLOCK pin as a PLOCK pin
- This multiplies up the input frequency to a VCO frequency of 589.824 MHz
- The PLL output is VCO / 4 via the output divider to 147.456 MHz
- The Fosc frequency is VCO / 4 via the output divider and low power divider to 147.456 MHz

Note that the default PCTL value is $04601D.

Use Table 5-5 to determine the appropriate PCTL value for generating the maximum operating frequency (Fosc). Locate the maximum Fref value in the table that is larger than the target Fref. Use the corresponding final PCTL value to generate the maximum operating frequency given the target Fref.

Table 5-5. PCTL Value Guide

150 MHz		
	Maximum Fref (MHz)	**Final PCTL**
1	5.00	$2xE01E
2	5.15	$2xE01D
3	5.35	$2xE01C
4	5.55	$2xE01B
5	5.75	$2xE01A
6	6.00	$2xE019
7	6.25	$2xE018

Table 5-5. PCTL Value Guide

150 MHz		
	Maximum Fref (MHz)	**Final PCTL**
8	6.521	$2xE017
9	6.82	$2xE016
10	7.142	$2xE015
11	7.5	$2xE014
12	7.89	$2xE013
13	8.33	$2xE012
14	8.82	$2xE011
15	9.4	$2xE010
16	10	$2xE00F
17	10.7	$2xE00E
18	11.54	$2xE00D
19	12.5	$2xE00C
20	13.636	$2xE00B
21	15	$2xE00A
22	16.666	$2xE009
23	18.75	$2xE008

Example:

Maximum Operating Frequency = 150 MHz

EXTAL Frequency = 11.2896 MHz

Target Fref = EXTAL / PD (where PD = 2) = 5.6448 MHz

The maximum Fref value that is greater than the Target Fref is #5: 5.75 MHz. The corresponding final PCTL value is $2xE01A. x represents the PLL Pre-Divider (PD) used to determine the target Fref value. In the example the Pre-Divider (PD) = 2. Thus, the PCTL value is $22E01A. The following PLL parameters can be determined from the final PCTL setting.

VCO frequency = 5.6448 MHz *4 * 26 = 589.056 MHz

PLL Output = VCO / 4= 147.264 MHz

Fosc = PLL Output = 147.264 MHz

This represents the maximum operating frequency obtainable with a target Fref frequency of 5.6448 MHz.

5.9 PLL Programming Examples

Table 5-6. PLL Programming Examples

EXTAL (MHz)	PDF	Fref (MHz) 5 MHz - 20 MHz	OD1	OD0	FM	MF	VCO Output (MHz) 300 - 600 MHz	OD	PLL Output (MHz)	LPD	Fosc (MHz)	PCTL
27.00	3	9.0	1	1	4	16	576.0	4	144.0	0	144	$23E010
27.0	4	6.75	1	1	4	22	594.0	4	148.5	0	148.5	$24E016

Table 5-6. PLL Programming Examples (continued)

EXTAL (MHz)	PDF	Fref (MHz) 5 MHz - 20 MHz	OD1	OD0	FM	MF	VCO Output (MHz) 300 - 600 MHz	OD	PLL Output (MHz)	LPD	Fosc (MHz)	PCTL
27.0	5	5.4	1	1	4	27	583.2	4	145.8	0	145.8	$25E01B
24.576	4	6.144	1	1	4	24	589.824	4	147.456	0	147.456	$24E018
24.576	3	8.192	1	1	4	18	589.824.	4	147.456	0	147.456	$23E012
12.288	1	12.288	1	1	4	12	589.824	4	147.456	0	147.456	$21E00C
12.288	2	6.144	1	1	4	24	589.824	4	147.456	0	147.456	$22E018
11.2896	2	5.6448	1	1	4	26	587.0592	4	146.7648	0	146.764	$22E01A

NOTE

The default PLL setting ($04601D) established upon reset when the PINIT pin is pulled high should not be used at or below 150 MHz operation.

Chapter 6
General Purpose Input/Output

6.1 Introduction

The DSP56374 provides up to 47 programmable signals that are dedicated GPIO pins or pins that can operate either as GPIO pins or peripheral pins (ESAI, ESAI_1, and TEC). Up to 20 pins can be programmed as signal or GPIO pins in the 52-pin package. The signals (except for MODA - MODD, and HREQ) are configured as GPIO after hardware reset. The techniques for register programming for all GPIO functionality is very similar between these interfaces. This section describes how signals may be used as GPIO.

6.2 Programming Model

The signals description section of this manual describes the special uses of these signals in detail. There are six groups of these signals which can be controlled separately or as groups:

- Port C: twelve GPIO signals (shared with the ESAI signals).
- Port E: twelve GPIO signals (shared with the ESAI_1 signals) **NOTE**: 80-pin package only.
- Port G: fifteen GPIO signals (dedicated GPIO signals) **NOTE**: 80-pin package only.
- Port H: five GPIO signals (shared with the MODA - MODD and HREQ signals).
- Timer: three GPIO signals (shared with the timer/event counter signals).

6.2.1 Port C and E Signals and Registers

Each of the 12 port C signals not used as an ESAI signal can be configured individually as a GPIO signal. The GPIO functionality of port C is controlled by three registers: port C control register (PCRC), port C direction register (PRRC) and port C data register (PDRC). These registers are described in Chapter 8, *Enhanced Serial Audio Interface (ESAI)*.

Each of the 12 port E signals not used as an ESAI_1 signal can be configured individually as a GPIO signal. The GPIO functionality of port E is controlled by three registers: port E control register (PCRE), port E direction register (PRRE) and port E data register (PDRE). These registers are described in Chapter 8, *Enhanced Serial Audio Interface (ESAI)*.

6.2.2 Port G Signals and Registers

Each of the 15 Port G signals can be configured individually as a GPIO signal. This establishes the appropriate wait state setting for correct operation of the interface. The GPIO functionality of Port G is controlled by three registers: Port G control register (PCRG), Port G direction register (PRRG), and Port G data register (PDRG). These registers are described below.

6.2.2.1 Port G Control Register (PCRG)

The read/write 24-bit Port G Control Register (PCRG) in conjunction with the Port G Direction Register (PRRG) controls the functionality of the dedicated GPIO pins. Each of the PG(14:0) bits controls the functionality of the corresponding port pin. See Table 6-1. for the port-pin configurations. Hardware and software reset clear all PCRG bits.

6.2.2.2 Port G Direction Register (PRRG)

The read/write 24-bit Port G Direction Register (PRRG) in conjunction with the Port G Control Register (PCRG) controls the functionality of the dedicated GPIO pins. Table 6-1. describes the port-pin configurations. Hardware and software reset clear all PRRG bits.

Table 6-1. PCRG and PRRG Bits Functionality

PDG[i]	PG[i]	Port Pin[i] Function
0	0	Disconnected
0	1	GPIO input
1	0	GPIO output
1	1	Open Drain Output

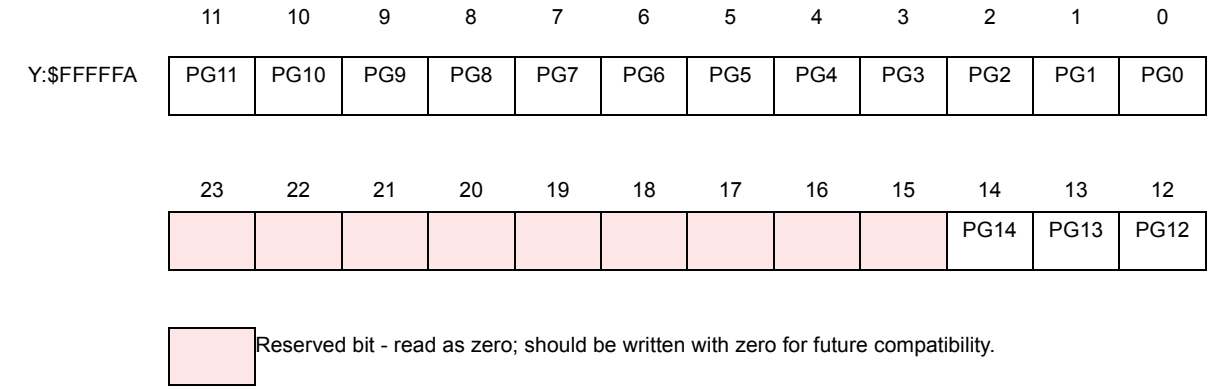


Figure 6-1. PCRG Register

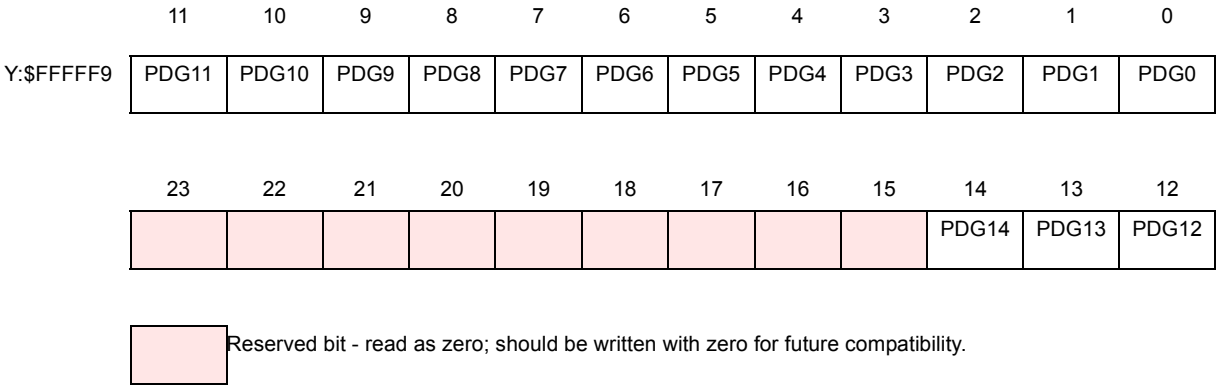


Figure 6-2. PRRG Register

6.2.2.3 Port G Data Register (PDRG)

The read/write 24-bit Port G Data Register (see Figure 6-3) is used to read or write data to/from the dedicated GPIO pins. Bits PG(14:0) are used to read or write data from/to the corresponding port pins. If a port pin [i] is configured as a GPIO input, the corresponding PG[i] bit reflects the value present on this pin. If a port pin [i] is configured as a GPIO output, the value written into the corresponding PG[i] bit is reflected on this pin. If a port pin [i] is configured as disconnected, the corresponding PG[i] bit is not reset and contains undefined data.

The PDG and PG bits should not be set simultaneously.

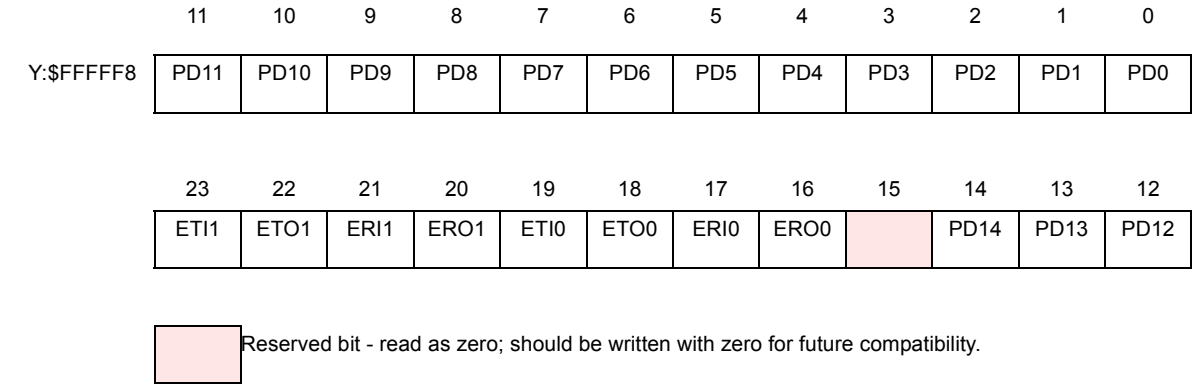


Figure 6-3. PDRG Register

6.2.2.4 ESAI/EXTAL Clocking Control

ESAI/EXTAL clock bits optionally direct the EXTAL clock to the ESAI clocking chain for generating the corresponding high frequency clock, bit clock and framesync clock. There are 8 ESAI/EXTAL clock control bits as described in Table 6-2. These bits are cleared upon reset.

Table 6-2. ESAI/EXTAL clock bit descriptions

ESAI/EXTAL control bit	Bit description
ETI1	When this bit is set, the EXTAL clock can be used to generate the ESAI_1 transmitter clocks : HCKT_1, SCKT_1 and FST_1. When this bit is cleared, the Fosc clock can be used to generate the ESAI_1 transmitter clocks : HCKT_1, SCKT_1 and FST_1.
ETO1	When this bit is set, the EXTAL clock is directed to the HCKT_1 pin. When this bit is cleared, EXTAL clock is not directed to the HCKT_1 pin
ERI1	When this bit is set, the EXTAL clock can be used to generate the ESAI_1 receiver clocks : HCKR_1, SCKR_1 and FSR_1. When this bit is cleared, the Fosc clock can be used to generate the ESAI_1 receiver clocks : HCKR_1, SCKR_1 and FSR_1.
ERO1	When this bit is set, the EXTAL clock is directed to the HCKR_1 pin. When this bit is cleared, the EXTAL clock is not directed to the HCKR_1 pin.
ETI0	When this bit is set, the EXTAL clock can be used to generate the ESAI transmitter clocks : HCKT, SCKT and FST. When this bit is cleared, the Fosc clock can be used to generate the ESAI transmitter clocks : HCKT, SCKT and FST.
ETO0	When this bit is set, the EXTAL clock is directed to the HCKT pin. When this bit is cleared, the EXTAL clock is not directed to the HCKT pin.
ERI0	When this bit is set, the EXTAL clock can be used to generate the ESAI receiver clocks : HCKR, SCKR and FSR. When this bit is cleared, the Fosc clock can be used to generate the ESAI receiver clocks : HCKR, SCKR and FSR.
ERO0	When this bit is set, the EXTAL clock is directed to the HCKR pin. When this bit is cleared, the EXTAL clock is not directed to the HCKR pin.

6.2.3 Port H Signals and Registers

Each of the five Port H signals (MODA, MODB, MODC, MODD and HREQ) can be configured individually as a GPIO signal. The GPIO functionality of Port H is controlled by three registers: Port H control register (PCRH), Port H direction register (PRRH) and Port H data register (PDRH).

6.2.3.1 Port H Control Register (PCRH)

The read/write 24-bit Port H Control Register (PCRH) in conjunction with the Port H Direction Register (PRRH) controls the functionality of the dedicated GPIO pins. Each of the PH(4:0) bits controls the functionality of the corresponding port pin. See Table 6-3. for the port-pin configurations. Hardware and software reset sets all PCRH bits.

6.2.3.2 Port H Direction Register (PRRH)

The read/write 24-bit Port H Direction Register (PRRH) in conjunction with the Port H Control Register (PCRH) controls the functionality of the dedicated GPIO pins. Table 6-3. describes the port-pin configurations. Hardware and software reset sets all PRRH bits.

Table 6-3. PCRH and PRRH Bits Functionality

PDH[i]	PH[i]	Port Pin[i] Function
0	0	Disconnected
0	1	GPIO input

Table 6-3. PCRH and PRRH Bits Functionality

PDH[i]	PH[i]	Port Pin[i] Function
1	0	GPIO output
1	1	Respective Functionality (MODx or HREQ)

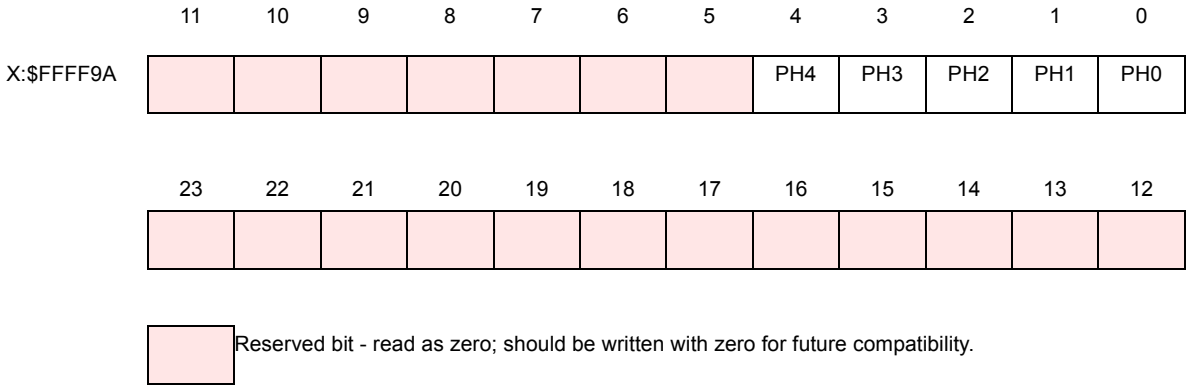


Figure 6-4. PCRH Register

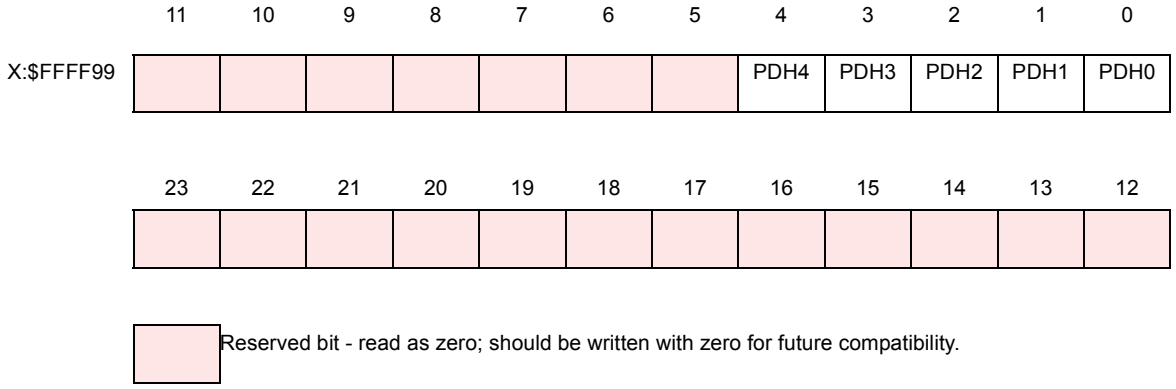


Figure 6-5. PRRH Register

6.2.3.3 Port H Data RSegister (PDRH)

The read/write 24-bit Port H Data Register (see <Blue>Figure 6-6.) is used to read or write data to/from the dedicated GPIO pins. Bits PH(4:0) are used to read or write data from/to the corresponding port pins if they are configured as GPIO. If a port pin [i] is configured as a GPIO input, the corresponding PH[i] bit reflects the value present on this pin. If a port pin [i] is configured as a GPIO output, the value written into the corresponding PH[i] bit is reflected on this pin. If a port pin [i] is configured as disconnected, the corresponding PH[i] bit is not reset and contains undefined data.

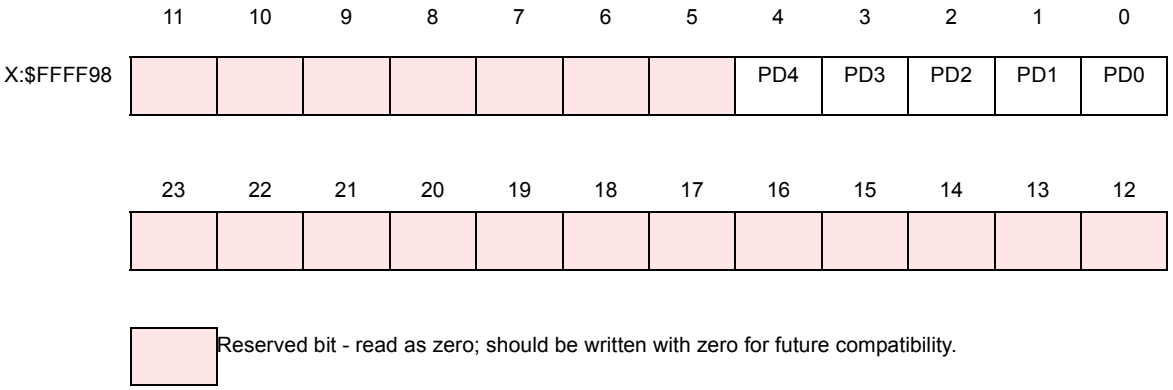


Figure 6-6. PDRH Register

6.2.4 Timer/Event Counter Signals

The timer/event counter signals (TIO0, TIO1 and TIO2), when not used as timer signals can be configured as GPIO signals. These signals are controlled by the appropriate timer control status register (TCSR). The register is described in Chapter 9, *Triple Timer Module*

Chapter 7
Serial Host Interface

7.1 Introduction

The Serial Host Interface (SHI) is a serial I/O interface that provides a path for communication and program/coefficient data transfers between the DSP and an external host processor. The SHI can also communicate with other serial peripheral devices. The SHI supports two well-known and widely used synchronous serial buses: the Freescale (previously known as Motorola) Serial Peripheral Interface (SPI) bus and the Philips Inter-Integrated-Circuit Control (I^2C) bus. The SHI supports either bus protocol as either a slave or a single-master device. To minimize DSP overhead, the SHI supports 8-bit, 16-bit and 24-bit data transfers. The SHI has a 1 or 10-word receive FIFO that permits receiving up to 30 bytes before generating a receive interrupt, reducing the overhead for data reception.

When configured in the SPI mode, the SHI can perform the following functions:

- Identify its slave selection (in slave mode)
- Simultaneously transmit (shift out) and receive (shift in) serial data
- Directly operate with 8-, 16- and 24-bit words
- Generate vectored interrupts separately for receive and transmit events and update status bits
- Generate a separate vectored interrupt for a receive exception
- Generate a separate vectored interrupt for a transmit exception
- Generate a separate vectored interrupt for a bus-error exception
- Generate the serial clock signal (in master mode)
- Trigger DMA to service the transmit and receive events

When configured in the I^2C mode, the SHI can perform the following functions:

- Detect/generate start and stop events
- Identify its slave (ID) address (in slave mode)
- Identify the transfer direction (receive/transmit)
- Transfer data byte-wise according to the SCL clock line
- Generate ACK signal following a byte receive
- Inspect ACK signal following a byte transmit
- Directly operate with 8-, 16- and 24-bit words
- Generate vectored interrupts separately for receive and transmit events and update status bits
- Generate a separate vectored interrupt for a receive exception
- Generate a separate vectored interrupt for a transmit exception
- Generate a separate vectored interrupt for a bus error exception
- Generate the clock signal (in master mode)
- Trigger DMA to service the transmit and receive events

7.2 Serial Host Interface Internal Architecture

The DSP views the SHI as a memory-mapped peripheral in the X data memory space. The DSP uses the SHI as a normal memory-mapped peripheral using standard polling, interrupt programming techniques, or DMA transfers. Memory mapping allows DSP communication with the SHI registers to be accomplished using standard instructions and addressing modes. In addition, the MOVEP instruction allows interface-to-memory and memory-to-interface data transfers without going through an intermediate register. The DMA controller may be used to service the receive or transmit data path. The single master configuration allows the DSP to directly connect to dumb peripheral devices. For that purpose, a programmable baud-rate generator is included to generate the clock signal for serial transfers. The host side invokes the SHI for communication and data transfer with the DSP through a shift register that may be accessed serially using either the I^2C or the SPI bus protocols. Figure 7-1 shows the SHI block diagram.

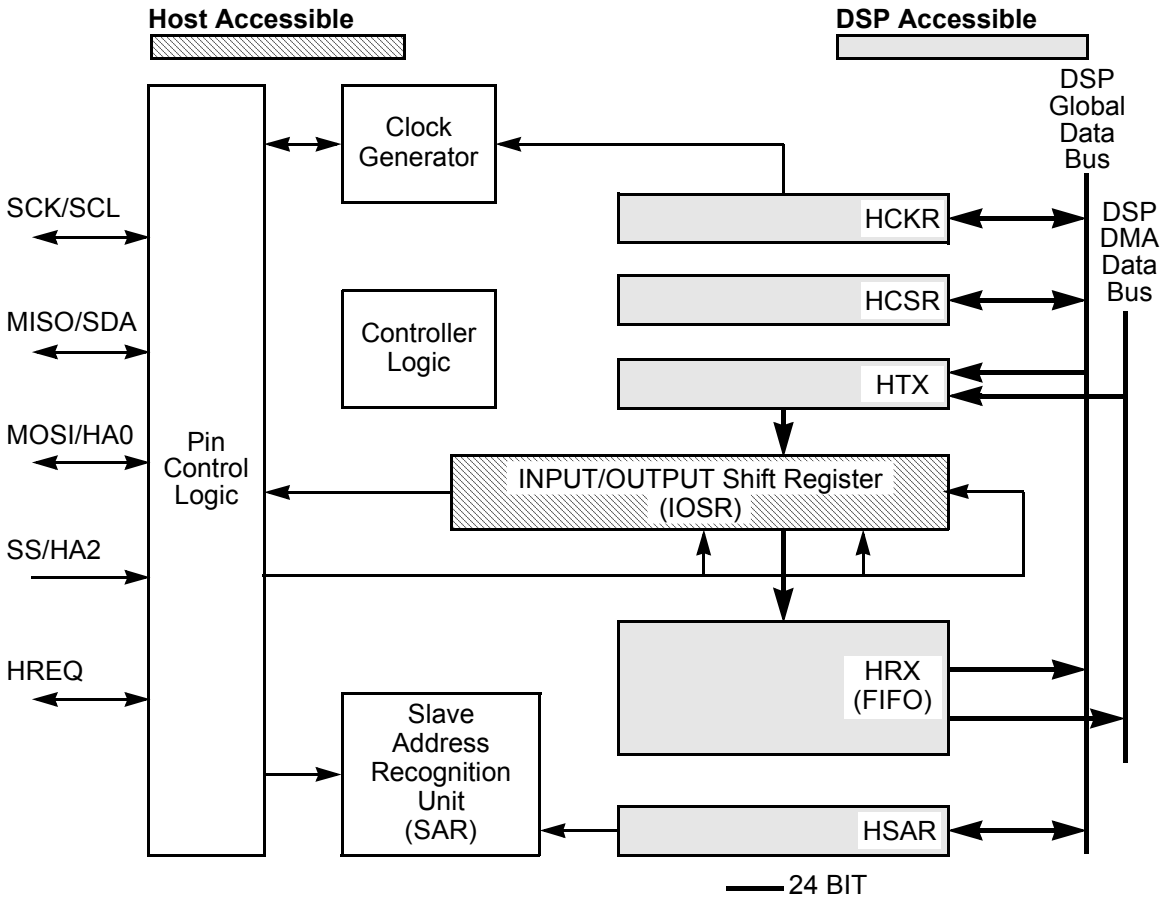


Figure 7-1. Serial Host Interface Block Diagram

7.3 SHI Clock Generator

The SHI clock generator generates the SHI serial clock if the interface operates in the master mode. The clock generator is disabled if the interface operates in the slave mode, except in I^2C mode when the HCKFR bit is set in the HCKR register. When the SHI operates in the slave mode, the clock is external and is input to the SHI (HMST = 0). Figure 7-2 illustrates the internal clock path connections. It is the user's responsibility to select the proper clock rate within the range as defined in the I^2C and SPI bus specifications.

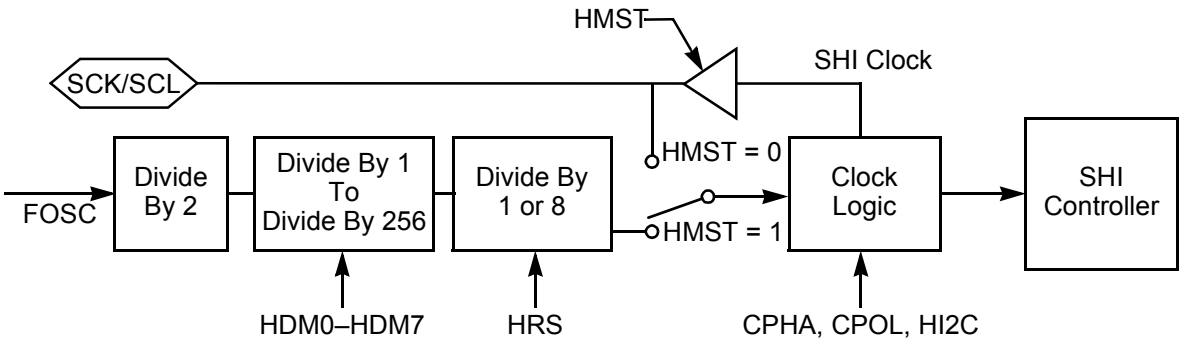


Figure 7-2. SHI Clock Generator

7.4 Serial Host Interface Programming Model

The Serial Host Interface programming model has two parts:

- **Host side**—see Figure 7-3 below and Section 7.4.1, *SHI Input/Output Shift Register (IOSR)—Host Side*
- **DSP side**—see Figure 7-4 and Section 7.4.2, *SHI Host Transmit Data Register (HTX)—DSP Side* through Section 7.4.6, *SHI Control/Status Register (HCSR)—DSP Side* for detailed information.

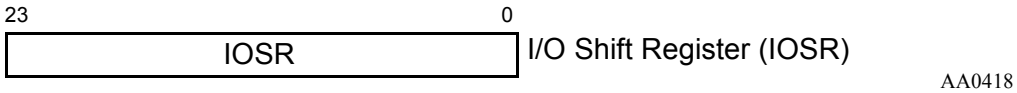


Figure 7-3. SHI Programming Model—Host Side

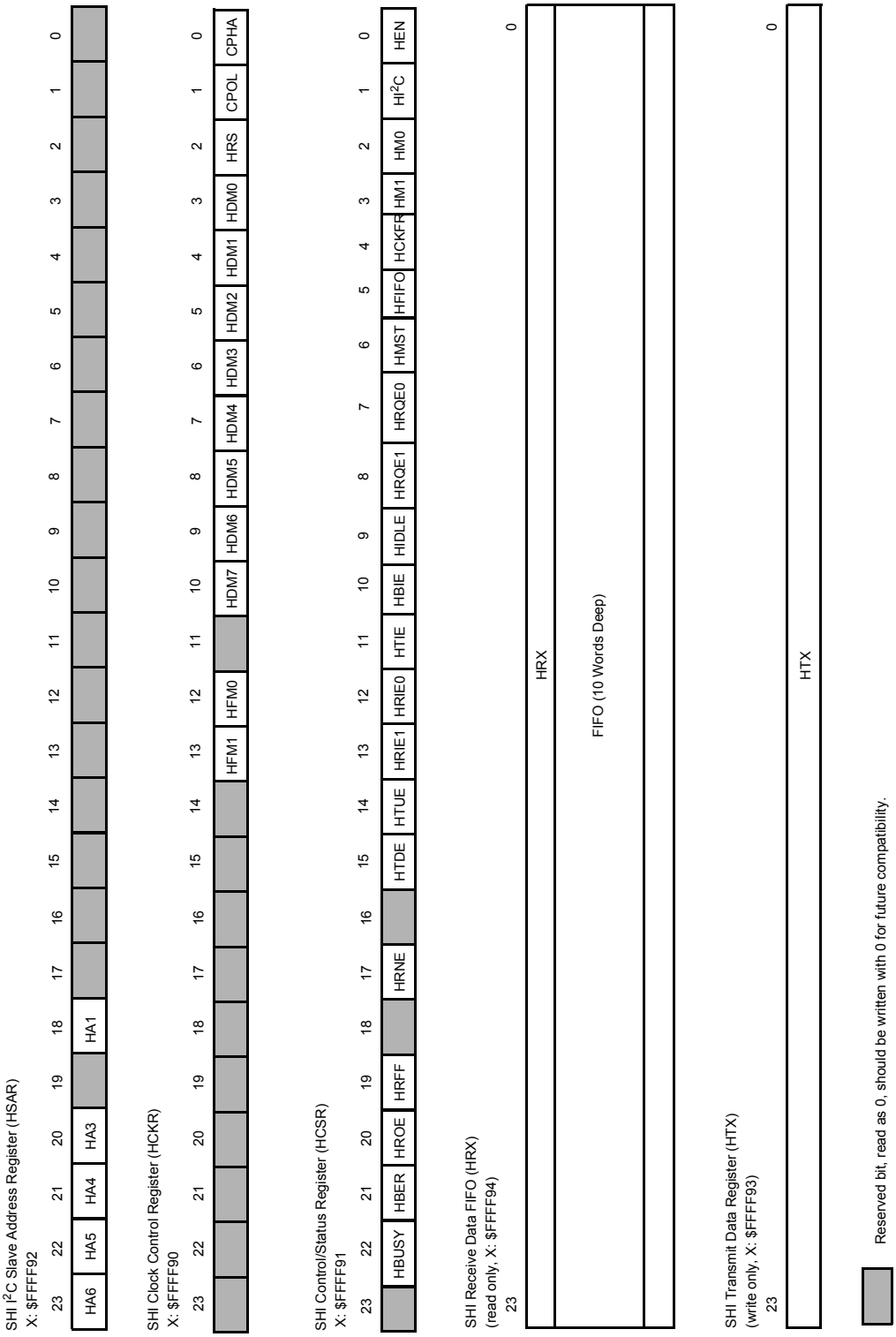


Figure 7-4. SHI Programming Model—DSP Side

The SHI interrupt vector table is shown in Table 7-1 and the exception priorities generated by the SHI are shown in Table 7-2.

Table 7-1. SHI Interrupt Vectors

Program Address	Interrupt Source
VBA:$0040	SHI Transmit Data
VBA:$0042	SHI Transmit Underrun Error
VBA:$0044	SHI Receive FIFO Not Empty
VBA:$0048	SHI Receive FIFO Full
VBA:$004A	SHI Receive Overrun Error
VBA:$004C	SHI Bus Error

Table 7-2. SHI Internal Interrupt Priorities

Priority	Interrupt
Highest	SHI Bus Error
	SHI Receive Overrun Error
	SHI Transmit Underrun Error
	SHI Receive FIFO Full
	SHI Transmit Data
Lowest	SHI Receive FIFO Not Empty

7.4.1 SHI Input/Output Shift Register (IOSR)—Host Side

The variable length Input/Output Shift Register (IOSR) can be viewed as a serial-to-parallel and parallel-to-serial buffer in the SHI. The IOSR is involved with every data transfer in both directions (read and write). In compliance with the I^2C and SPI bus protocols, data is shifted in and out MSB first. In 8-bit data transfer modes, the most significant byte of the IOSR is used as the shift register. In 16-bit data transfer modes, the two most significant bytes become the shift register. In 24-bit transfer modes, the shift register uses all three bytes of the IOSR (see Table 7-5).

NOTE

The IOSR cannot be accessed directly either by the host processor or by the DSP. It is fully controlled by the SHI controller logic.

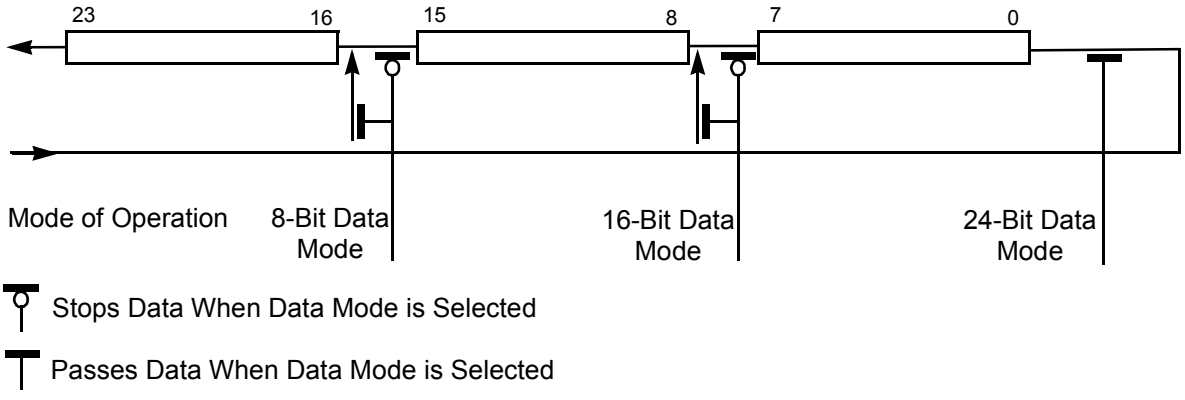


Figure 7-5. SHI I/O Shift Register (IOSR)

7.4.2 SHI Host Transmit Data Register (HTX)—DSP Side

The host transmit data register (HTX) is used for DSP-to-Host data transfers. The HTX register is 24 bits wide. Writing to the HTX register by DSP core instructions or by DMA transfers clears the HTDE flag. The DSP may program the HTIE bit to cause a host transmit data interrupt when HTDE is set (see Section 7.4.6.10, *HCSR Transmit-Interrupt Enable (HTIE)—Bit 11*). Data should not be written to the HTX

until HTDE is set in order to prevent overwriting the previous data. HTX is reset to the empty state when in stop mode and during hardware reset, software reset and individual reset.

In the 8-bit data transfer mode, the most significant byte of the HTX is transmitted; in the 16-bit mode, the two most significant bytes are transmitted, and, in the 24-bit mode, all the contents of HTX are transferred.

7.4.3 SHI Host Receive Data FIFO (HRX)—DSP Side

The 24-bit host receive data FIFO (HRX) is a 10-word deep, First-In-First-Out (FIFO) register used for Host-to-DSP data transfers. The serial data is received via the shift register and then loaded into the HRX. In the 8-bit data transfer mode, the most significant byte of the shift register is transferred to the HRX (the other bits are cleared); in the 16-bit mode the two most significant bytes are transferred (the least significant byte is cleared), and, in the 24-bit mode, all 24 bits are transferred to the HRX. The HRX may be read by the DSP while the FIFO is being loaded from the shift register. Reading all data from HRX clears the HRNE flag. The HRX may be read by DSP core instructions or by DMA transfers. The HRX FIFO is reset to the empty state when the chip is in stop mode, as well as during hardware reset, software reset and individual reset.

7.4.4 SHI Slave Address Register (HSAR)—DSP Side

The 24-bit slave address register (HSAR) is used when the SHI operates in the I^2C slave mode and is ignored in the other operational modes. HSAR holds five bits of the 7-bit slave device address. The SHI also acknowledges the general call address specified by the I^2C protocol (eight zeroes comprising a 7-bit address and a R/$\overline{W}$ bit), but treats any following data bytes as regular data. That is, the SHI does not differentiate between its dedicated address and the general call address. HSAR cannot be accessed by the host processor.

7.4.4.1 HSAR Reserved Bits—Bits 19, 17– 0

These bits are reserved. They read as zero and should be written with zero for future compatibility.

7.4.4.2 HSAR I^2C Slave Address (HA[6:3], HA1)—Bits 23–20,18

Part of the I^2C slave device address is stored in the read/write HA[6:3], HA1 bits of HSAR. The full 7-bit slave device address is formed by combining the HA[6:3], HA1 bits with the HA0 and HA2 pins to obtain the HA[6:0] slave device address. The full 7-bit slave device address is compared to the received address byte whenever an I^2C master device initiates an I^2C bus transfer. During hardware reset or software reset, HA[6:3] = 1011 and HA1 is cleared; this results in a default slave device address of 1011[HA2]0[HA0].

7.4.5 SHI Clock Control Register (HCKR)—DSP Side

The HCKR is a 24-bit read/write register that controls SHI clock generator operation. The HCKR bits should be modified only while the SHI is in the individual reset state (HEN = 0 in the HCSR).

For proper SHI clock setup, please consult the data sheet. The programmer should not use the combination HRS = 1 and HDM[7:0] = 00000000, since it may cause synchronization problems and improper operation (it is an illegal combination).

The HCKR bits are cleared during hardware reset or software reset, except for CPHA, which is set. The HCKR is not affected by the stop state.

The HCKR bits are described in the following paragraphs.

7.4.5.1 Clock Phase and Polarity (CPHA and CPOL)—Bits 1–0

The Clock Phase (CPHA) bit controls the relationship between the data on the master-in-slave-out (MISO) and master-out-slave-in (MOSI) pins and the clock produced or received at the SCK pin. The CPOL bit determines the clock polarity (1 = active-high, 0 = active-low).

The clock phase and polarity should be identical for both the master and slave SPI devices. CPHA and CPOL are functional only when the SHI operates in the SPI mode and are ignored in the I^2C mode. The CPHA bit is set and the CPOL bit is cleared during hardware reset and software reset.

The programmer may select any of four combinations of serial clock (SCK) phase and polarity when operating in the SPI mode (See Figure 7-6).

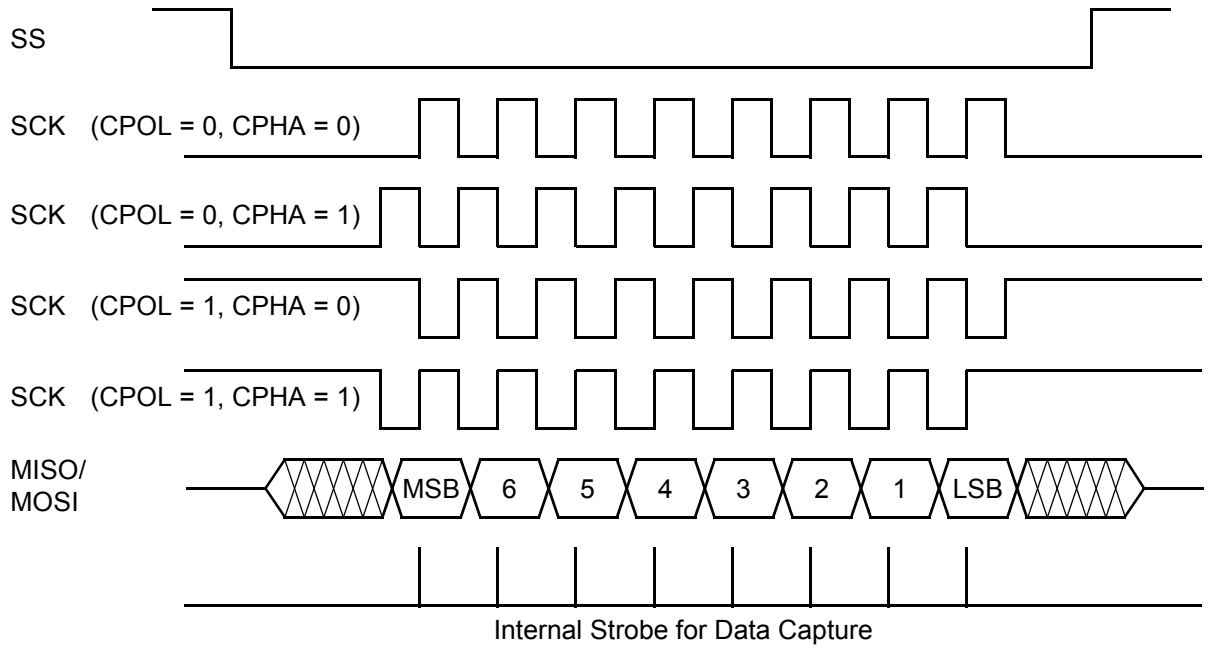


Figure 7-6. SPI Data-To-Clock Timing Diagram

If CPOL is cleared, it produces a steady-state low value at the SCK pin of the master device whenever data is not being transferred. If the CPOL bit is set, it produces a high value at the SCK pin of the master device whenever data is not being transferred.

CPHA is used with the CPOL bit to select the desired clock-to-data relationship. The CPHA bit, in general, selects the clock edge that captures data and allows it to change states. It has its greatest impact on the first bit transmitted (MSB) in that it does or does not allow a clock transition before the data capture edge.

When the SHI is in slave mode and CPHA = 0, the $\overline{SS}$ line must be de-asserted and asserted by the external master between each successive word transfer. $\overline{SS}$ must remain asserted between successive bytes within a word. The DSP core should write the next data word to HTX when HTDE = 1, clearing HTDE. However, the data is transferred to the shift register for transmission only when $\overline{SS}$ is de-asserted. HTDE is set when the data is transferred from HTX to the shift register.

When the SHI is in slave mode and CPHA = 1, the $\overline{SS}$ line may remain asserted between successive word transfers. The $\overline{SS}$ must remain asserted between successive bytes within a word. The DSP core should write the next data word to HTX when HTDE = 1, clearing HTDE. The HTX data is transferred to the shift register for transmission as soon as the shift register is empty. HTDE is set when the data is transferred from HTX to the shift register.

When the SHI is in master mode and CPHA = 0, the DSP core should write the next data word to HTX when HTDE = 1, clearing HTDE. The data is transferred immediately to the shift register for transmission. HTDE is set only at the end of the data word transmission.

NOTE

The master is responsible for de-asserting and asserting the slave device $\overline{SS}$ line between word transmissions.

When the SHI is in master mode and CPHA = 1, the DSP core should write the next data word to HTX when HTDE = 1, clearing HTDE. The HTX data is transferred to the shift register for transmission as soon as the shift register is empty. HTDE is set when the data is transferred from HTX to the shift register.

7.4.5.2 HCKR Prescaler Rate Select (HRS)—Bit 2

The HRS bit controls a prescaler in series with the clock generator divider. This bit is used to extend the range of the divider when slower clock rates are desired. When HRS is set, the prescaler is bypassed. When HRS is cleared, the fixed divide-by-eight prescaler is operational. HRS is ignored when the SHI operates in the slave mode, except for I^2C when HCKFR is set. The HRS bit is cleared during hardware reset and software reset.

NOTE

Use the equations in the SHI data sheet to determine the value of HRS for the specific serial clock frequency required.

7.4.5.3 HCKR Divider Modulus Select (HDM[7:0])—Bits 10–3

The HDM[7:0] bits specify the divide ratio of the clock generator divider. A divide ratio between 1 and 256 (HDM[7:0] = $00 to $FF) may be selected. When the SHI operates in the slave mode, the HDM[7:0] bits are ignored (except for I^2C when HCKFR is set). The HDM[7:0] bits are cleared during hardware reset and software reset.

NOTE

Use the equations in the SHI data sheet to determine the value of HDM[7:0] for the specific serial clock frequency required.

7.4.5.4 HCKR Filter Mode (HFM[1:0]) — Bits 13–12

The read/write control bits HFM[1:0] specify the operational mode of the noise reduction filters, as described in Table 7-3.. The filters are designed to eliminate undesired spikes that might occur on the clock and data-in lines and allow the SHI to operate in noisy environments when required. One filter is located in the input path of the SCK/SCL line and the other is located in the input path of the data line (i.e., the SDA line when in I^2C mode, the MISO line when in SPI master mode, and the MOSI line when in SPI slave mode).

Table 7-3. SHI Noise Reduction Filter Mode

HFM1	HFM0	Description
0	0	Bypassed (Disabled)
0	1	Very Narrow Spike Tolerance.
1	0	Narrow Spike Tolerance
1	1	Wide Spike Tolerance

When HFM[1:0] = 00, the filter is bypassed (spikes are not filtered out). This mode is useful when higher bit-rate transfers are required and the SHI operates in a noise-free environment.

When HFM[1:0] = 01, the very narrow-spike-tolerance filter mode is selected. In this mode the filters eliminate spikes with durations of up to 10ns. This mode is useful when very high bit-rate transfers are required and the SHI operates in a nearly noise-free environment.

When HFM[1:0] = 10, the narrow-spike-tolerance filter mode is selected. In this mode the filters eliminate spikes with durations of up to 50ns. This mode is suitable for use in mildly noisy environments and imposes some limitations on the maximum achievable bit-rate transfer.

When HFM[1:0] = 11, the wide-spike-tolerance filter mode is selected. In this mode the filters eliminate spikes up to 100 ns. This mode is recommended for use in noisy environments; the bit-rate transfer is strictly limited. The wide-spike- tolerance filter mode is highly recommended for use in I^2C bus systems as it fully conforms to the I^2C bus specification and improves noise immunity.

NOTE

HFM[1:0] are cleared during hardware reset and software reset.

After changing the filter bits in the HCKR to a non-bypass mode (HFM[1:0] not equal to ‘00’), the programmer should wait at least ten times the tolerable spike width before enabling the SHI (setting the HEN bit in the HCSR). Similarly, after changing the H I^2C bit in the HCSR or the CPOL bit in the HCKR, while the filter mode bits are in a non-bypass mode (HFM[1:0] not equal to ‘00’), the programmer should wait at least ten times the tolerable spike width before enabling the SHI (setting HEN in the HCSR).

7.4.5.5 HCKR Reserved Bits—Bits 23–14, 11

These bits in HCKR are reserved. They are read as zero and should be written with zero for future compatibility.

7.4.6 SHI Control/Status Register (HCSR)—DSP Side

The HCSR is a 24-bit register that controls the SHI operation and reflects its status. The control bits are read/write. The status bits are read-only. The bits are described in the following paragraphs. When in the stop state or during individual reset, the HCSR status bits are reset to their hardware-reset state, while the control bits are not affected.

7.4.6.1 HCSR Host Enable (HEN)—Bit 0

The read/write control bit HEN, when set, enables the SHI. When HEN is cleared, the SHI is disabled (that is, it is in the individual reset state, see below). The HCKR and the HCSR control bits are not affected when HEN is cleared. When operating in master mode, HEN should be cleared only when the SHI is idle (HBUSY = 0). HEN is cleared during hardware reset and software reset.

7.4.6.1.1 SHI Individual Reset

While the SHI is in the individual reset state, SHI input pins are inhibited, output and bidirectional pins are disabled (high impedance), the HCSR status bits and the transmit/receive paths are reset to the same state produced by hardware reset or software reset. The individual reset state is entered following a one-instruction-cycle delay after clearing HEN.

7.4.6.2 HCSR I²C/SPI Selection (HI²C)—Bit 1

The read/write control bit H I²C selects whether the SHI operates in the I²C or SPI modes. When HI²C is cleared, the SHI operates in the SPI mode. When HI²C is set, the SHI operates in the I²C mode. H I²C affects the functionality of the SHI pins as described in Section 2, *Signal/Connection Descriptions* . It is recommended that an SHI individual reset be generated (HEN cleared) before changing H I²C. H I²C is cleared during hardware reset and software reset.

7.4.6.3 HCSR Serial Host Interface Mode (HM[1:0])—Bits 3–2

The read/write control bits HM[1:0] select the size of the data words to be transferred, as shown in Table 7-4.. HM[1:0] should be modified only when the SHI is idle (HBUSY = 0). HM[1:0] are cleared during hardware reset and software reset.

Table 7-4. SHI Data Size

HM1	HMO	Description
0	0	8-bit data
0	1	16-bit data
1	0	24-bit data
1	1	Reserved

7.4.6.4 HCSR I²C Clock Freeze (HCKFR)—Bit 4

The read/write control bit HCKFR determines the behavior of the SHI when the SHI is unable to service the master request, when operating in the I²C slave mode. The HCKFR bit is used only in the I²C slave mode; it is ignored otherwise.

If HCKFR is set, the SHI holds the clock line to GND if it is not ready to send data to the master during a read transfer or if the input FIFO is full when the master attempts to execute a write transfer. In this way, the master may detect that the slave is not ready for the requested transfer, without causing an error condition in the slave. When HCKFR is set for transmit sessions, the SHI clock generator must be programmed as if to generate the same serial clock as produced by the external master, otherwise erroneous operation may result. The programmed frequency should be in the range of 1 to 0.75 times the external clock frequency.

If HCKFR is cleared, any attempt from the master to execute a transfer when the slave is not ready results in an overrun or underrun error condition.

It is recommended that an SHI individual reset be generated (HEN cleared) before changing HCKFR. HCKFR is cleared during hardware reset and software reset.

7.4.6.5 HCSR FIFO-Enable Control (HFIFO)—Bit 5

The read/write control bit HFIFO selects the receive FIFO size. When HFIFO is cleared, the FIFO has one level. When HFIFO is set, the FIFO has 10 levels. It is recommended that an SHI individual reset be generated (HEN cleared) before changing HFIFO. HFIFO is cleared during hardware reset and software reset.

7.4.6.6 HCSR Master Mode (HMST)—Bit 6

The read/write control bit HMST determines the SHI operating mode. If HMST is set, the interface operates in the master mode. If HMST is cleared, the interface operates in the slave mode. The SHI supports a single-master configuration in both I²C and SPI modes.

When configured as an SPI master, the SHI drives the SCK line and controls the direction of the data lines MOSI and MISO. The $\overline{SS}$ line must be held de-asserted in the SPI master mode; if the $\overline{SS}$ line is asserted when the SHI is in SPI master mode, a bus error is generated (the HCSR HBER bit is set—see Section 7.4.6.18, *Host Bus Error (HBER)—Bit 21*).

When configured as an I²C master, the SHI controls the I²C bus by generating start events, clock pulses and stop events for transmission and reception of serial data.

It is recommended that an SHI individual reset be generated (HEN cleared) before changing HMST. HMST is cleared during hardware reset and software reset.

7.4.6.7 HCSR Host-Request Enable (HRQE[1:0])—Bits 8–7

The read/write control bits HRQE[1:0] are used to control the $\overline{\text{HREQ}}$ pin. When HRQE[1:0] are cleared, the $\overline{\text{HREQ}}$ pin is disabled and held in the high impedance state. If either of HRQE[1:0] are set and the SHI is in a master mode, the $\overline{\text{HREQ}}$ pin becomes an input controlling SCK: de-asserting $\overline{\text{HREQ}}$ suspends SCK. If either of HRQE[1:0] are set and the SHI is in SPI slave mode, $\overline{\text{HREQ}}$ becomes an output and its operation is defined in Table 7-5. HRQE[1:0] should be changed only when the SHI is idle (HBUSY = 0). HRQE[1:0] are cleared during hardware reset and software reset. Note the HREQ can also be programmed as a GPIO. See Section 6.2.3, *Port H Signals and Registers*

Table 7-5. $\overline{\text{HREQ}}$ Function In SPI Slave Mode

HRQE1	HRQE0	$\overline{\text{HREQ}}$ Pin Operation
0	0	High impedance
0	1	Asserted if IOSR is ready to receive a new word
1	0	Asserted if IOSR is ready to transmit a new word
1	1	SPI: Asserted if IOSR is ready to transmit and receive

7.4.6.8 HCSR Idle (HIDLE)—Bit 9

The read/write control/status bit HIDLE is used only in the I^2C master mode; it is ignored otherwise. It is only possible to set the HIDLE bit during writes to the HCSR. HIDLE is cleared by writing to HTX. To ensure correct transmission of the slave device address byte, HIDLE should be set only when HTX is empty (HTDE = 1). After HIDLE is set, a write to HTX clears HIDLE and causes the generation of a stop event, a start event, and then the transmission of the eight MSBs of the data as the slave device address byte. While HIDLE is cleared, data written to HTX is transmitted as is. If the SHI completes transmitting a word and there is no new data in HTX, the clock is suspended after sampling ACK. If HIDLE is set when the SHI completes transmitting a word with no new data in HTX, a stop event is generated.

HIDLE determines the acknowledge that the receiver sends after correct reception of a byte. If HIDLE is cleared, the reception is acknowledged by sending a 0 bit on the SDA line at the ACK clock tick. If HIDLE is set, the reception is not acknowledged (a 1 bit is sent). It is used to signal an end-of-data to a slave transmitter by not generating an ACK on the last byte. As a result, the slave transmitter must release the SDA line to allow the master to generate the stop event. If the SHI completes receiving a word and the HRX FIFO is full, the clock is suspended before transmitting an ACK. While HIDLE is cleared the bus is busy, that is, the start event was sent but no stop event was generated. Setting HIDLE causes a stop event after receiving the current word.

HIDLE is set while the SHI is not in the I^2C master mode, while the chip is in the stop state, and during hardware reset, software reset and individual reset.

NOTE

Programmers should take care to ensure that all DMA channel service to HTX is disabled before setting HIDLE.

7.4.6.9 HCSR Bus-Error Interrupt Enable (HBIE)—Bit 10

The read/write control bit HBIE is used to enable the SHI bus-error interrupt. If HBIE is cleared, bus-error interrupts are disabled, and the HBER status bit must be polled to determine if an SHI bus error occurred. If both HBIE and HBER are set, the SHI requests an SHI bus-error interrupt service from the interrupt controller. HBIE is cleared by hardware reset and software reset.

NOTE

Clearing HBIE masks a pending bus-error interrupt only after a one instruction cycle delay. If HBIE is cleared in a long interrupt service routine, it is recommended that at least one other instruction separate the instruction that clears HBIE and the RTI instruction at the end of the interrupt service routine.

7.4.6.10 HCSR Transmit-Interrupt Enable (HTIE)—Bit 11

The read/write control bit HTIE is used to enable the SHI transmit data interrupts. If HTIE is cleared, transmit interrupts are disabled, and the HTDE status bit must be polled to determine if HTX is empty. If both HTIE and HTDE are set and HTUE is cleared, the SHI requests an SHI transmit-data interrupt service from the interrupt controller. If both HTIE and HTUE are set, the SHI requests an SHI transmit-underrun-error interrupt service from the interrupt controller. HTIE is cleared by hardware reset and software reset.

NOTE

Clearing HTIE masks a pending transmit interrupt only after a one instruction cycle delay. If HTIE is cleared in a long interrupt service routine, it is recommended that at least one other instruction separate the instruction that clears HTIE and the RTI instruction at the end of the interrupt service routine.

7.4.6.11 HCSR Receive Interrupt Enable (HRIE[1:0])—Bits 13–12

The read/write control bits HRIE[1:0] are used to enable the SHI receive-data interrupts. If HRIE[1:0] are cleared, receive interrupts are disabled, and the HRNE and HRFF (bits 17 and 19, see below) status bits must be polled to determine if there is data in the receive FIFO. If HRIE[1:0] are not cleared, receive interrupts are generated according to Table 7-6. HRIE[1:0] are cleared by hardware and software reset.

Table 7-6. HCSR Receive Interrupt Enable Bits

HRIE[1:0]	Interrupt	Condition
00	Disabled	Not applicable
01	Receive FIFO not empty Receive Overrun Error	HRNE = 1 and HROE = 0 HROE = 1
10	Reserved	Not applicable
11	Receive FIFO full Receive Overrun Error	HRFF = 1 and HROE = 0 HROE = 1

NOTE

Clearing HRIE[1:0] masks a pending receive interrupt only after a one instruction cycle delay. If HRIE[1:0] are cleared in a long interrupt service routine, it is recommended that at least one other instruction separate the instruction that clears HRIE[1:0] and the RTI instruction at the end of the interrupt service routine.

7.4.6.12 HCSR Host Transmit Underrun Error (HTUE)—Bit 14

The read-only status bit HTUE indicates whether a transmit-underrun error occurred. Transmit-underrun errors can occur only when operating in the SPI slave mode or the I^2C slave mode when HCKFR is cleared. In a master mode, transmission takes place on demand and no underrun can occur. HTUE is set when both the shift register and the HTX register are empty and the external master begins reading the next word:

- When operating in the I^2C mode, HTUE is set in the falling edge of the ACK bit. In this case, the SHI retransmits the previously transmitted word.
- When operating in the SPI mode, HTUE is set at the first clock edge if CPHA = 1; it is set at the assertion of $\overline{SS}$ if CPHA = 0.

If a transmit interrupt occurs with HTUE set, the transmit-underrun interrupt vector is generated. If a transmit interrupt occurs with HTUE cleared, the regular transmit-data interrupt vector is generated. HTUE is cleared by reading the HCSR and then writing to the HTX register. HTUE is cleared by hardware reset, software reset, SHI individual reset and during the stop state.

7.4.6.13 HCSR Host Transmit Data Empty (HTDE)—Bit 15

The read-only status bit HTDE indicates whether the HTX register is empty and can be written by the DSP. HTDE is set when the data word is transferred from HTX to the shift register, except in SPI master mode when CPHA = 0 (see HCKR). When in the SPI master mode with CPHA = 0, HTDE is set after the end of the data word transmission. HTDE is cleared when the DSP writes the HTX either with write instructions or DMA transfers. HTDE is set by hardware reset, software reset, SHI individual reset and during the stop state.

7.4.6.14 HCSR Reserved Bits—Bits 23, 18 and 16

These bits are reserved. They read as zero and should be written with zero for future compatibility.

7.4.6.15 Host Receive FIFO Not Empty (HRNE)—Bit 17

The read-only status bit HRNE indicates that the Host Receive FIFO (HRX) contains at least one data word. HRNE is set when the FIFO is not empty. HRNE is cleared when HRX is read by the DSP (read instructions or DMA transfers), reducing the number of words in the FIFO to zero. HRNE is cleared during hardware reset, software reset, SHI individual reset and during the stop state.

7.4.6.16 Host Receive FIFO Full (HRFF)—Bit 19

The read-only status bit HRFF indicates, when set, that the Host Receive FIFO (HRX) is full. HRFF is cleared when HRX is read by the DSP (read instructions or DMA transfers) and at least one place is available in the FIFO. HRFF is cleared by hardware reset, software reset, SHI individual reset and during the stop state.

7.4.6.17 Host Receive Overrun Error (HROE)—Bit 20

The read-only status bit HROE indicates, when set, that a data-receive overrun error has occurred. Receive-overrun errors cannot occur when operating in the I^2C master mode, because the clock is suspended if the receive FIFO is full; nor can they occur in the I^2C slave mode when HCKFR is set.

HROE is set when the shift register (IOSR) is filled and ready to transfer the data word to the HRX FIFO and the FIFO is already full (HRFF is set). When a receive-overrun error occurs, the shift register is not transferred to the FIFO. If a receive interrupt occurs with HROE set, the receive-overrun interrupt vector is generated. If a receive interrupt occurs with HROE cleared, the regular receive-data interrupt vector is generated.

HROE is cleared by reading the HCSR with HROE set, followed by reading HRX. HROE is cleared by hardware reset, software reset, SHI individual reset and during the stop state.

7.4.6.18 Host Bus Error (HBER)—Bit 21

The read-only status bit HBER indicates, when set, that an SHI bus error occurred when operating as a master (HMST set). In I^2C mode, HBER is set if the transmitter does not receive an acknowledge after a byte is transferred; then a stop event is generated and transmission is suspended. In SPI mode, HBER is set if $\overline{SS}$ is asserted; then transmission is suspended at the end of transmission of the current word. HBER is cleared only by hardware reset, software reset, SHI individual reset and during the stop state.

7.4.6.19 HCSR Host Busy (HBUSY)—Bit 22

The read-only status bit HBUSY indicates that the I^2C bus is busy (when in the I^2C mode) or that the SHI itself is busy (when in the SPI mode). When operating in the I^2C mode, HBUSY is set after the SHI detects a start event and remains set until a stop event is detected. When operating in the slave SPI mode, HBUSY is set while $\overline{SS}$ is asserted. When operating in the master SPI mode, HBUSY is set if the HTX register is not empty or if the IOSR is not empty. HBUSY is cleared otherwise. HBUSY is cleared by hardware reset, software reset, SHI individual reset and during the stop state.

7.5 Characteristics Of The SPI Bus

The SPI bus consists of two serial data lines (MISO and MOSI), a clock line (SCK) and a Slave Select line ($\overline{SS}$). During an SPI transfer, a byte is shifted out one data pin while a different byte is simultaneously shifted in through a second data pin. It can be viewed as two 8-bit shift registers connected together in a circular manner, with one shift register on the master side and the other on the slave side. Thus the data bytes in the master device and slave device are exchanged. The MISO and MOSI data pins are used for transmitting and receiving serial data. When the SPI is configured as a master, MISO is the master data input line, and MOSI is the master data output line. When the SPI is configured as a slave device, MISO is the slave data output line, and MOSI is the slave data input line.

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. When the SPI is configured as a master, the control bits in the HCKR select the appropriate clock rate, as well as the desired clock polarity and phase format (see Figure 7-6).

The $\overline{SS}$ line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activity, i.e., they keep their MISO output pin in the high-impedance state. When the SHI is configured as an SPI master device, the $\overline{SS}$ line should be held high. If the $\overline{SS}$ line is driven low when the SHI is in SPI master mode, a bus error is generated (the HCSR HBER bit is set).

7.6 Characteristics Of The I^2C Bus

The I^2C serial bus consists of two bidirectional lines, one for data signals (SDA) and one for clock signals (SCL). Both the SDA and SCL lines must be connected to a positive supply voltage via a pull-up resistor.

NOTE

In the I^2C bus specifications, the standard mode (100KHz clock rate) and a fast mode (400KHz clock rate) are defined. The SHI can operate in either mode.

7.6.1 Overview

The I^2C bus protocol must conform to the following rules:

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is high. Changes in the data line when the clock line is high are interpreted as control signals (see Table 7-7).

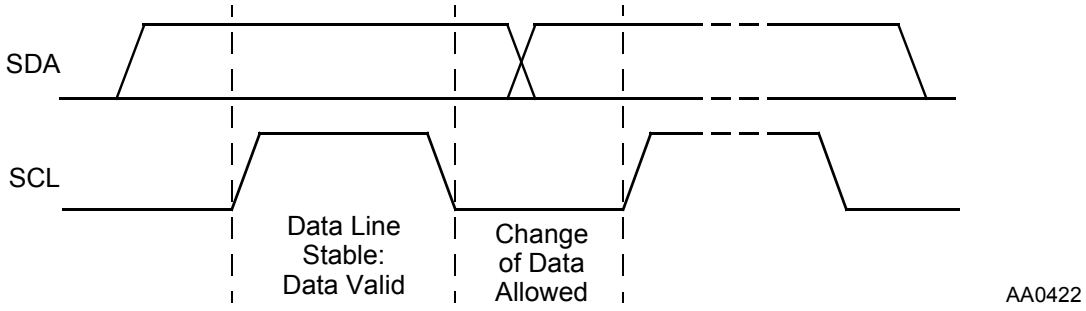


Figure 7-7. I2C Bit Transfer

Accordingly, the I2C bus protocol defines the following events:

- **Bus not busy**—Both data and clock lines remain high.
- **Start data transfer**—The start event is defined as a change in the state of the data line, from high to low, while the clock is high (see Figure 7-8).
- **Stop data transfer**—The stop event is defined as a change in the state of the data line, from low to high, while the clock is high (see Figure 7-8).
- **Data valid**—The state of the data line represents valid data when, after a start event, the data line is stable for the duration of the high period of the clock signal. The data on the line may be changed during the low period of the clock signal. There is one clock pulse per bit of data.

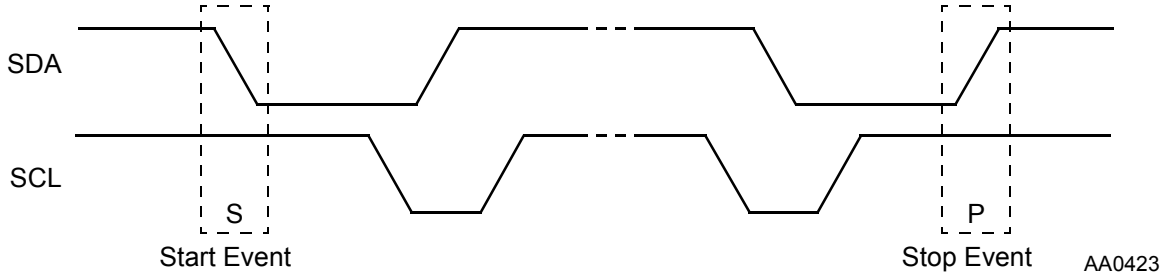


Figure 7-8. I2C Start and Stop Events

Each 8-bit word is followed by one acknowledge bit. This acknowledge bit is a high level put on the bus by the transmitter when the master device generates an extra acknowledge-related clock pulse. A slave receiver that is addressed must generate an acknowledge after each byte is received. Also, a master receiver must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter. The acknowledging device must pull down the SDA line during the acknowledge clock pulse so that the SDA line is stable low during the high period of the acknowledge-related clock pulse (see Table 7-9).

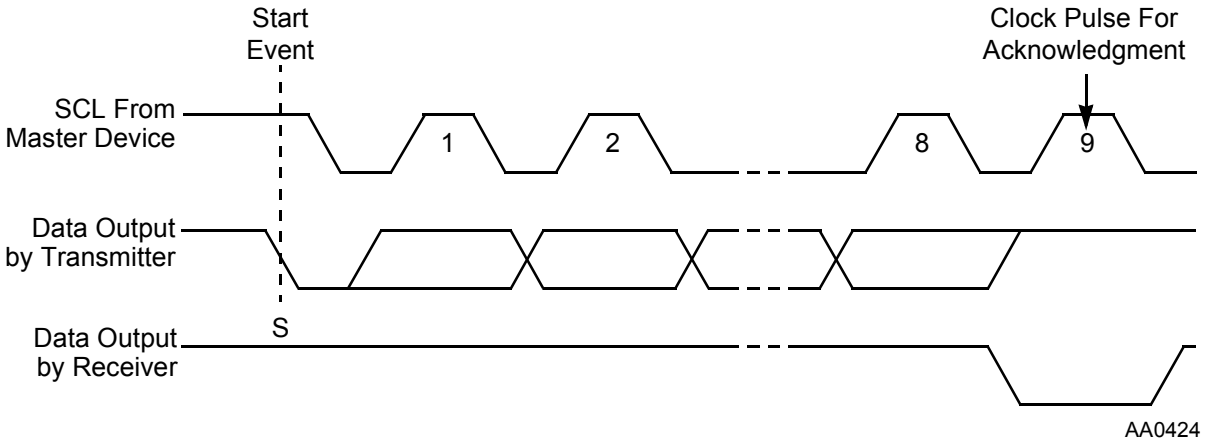


Figure 7-9. Acknowledgment on the I2C Bus

A device generating a signal is called a transmitter, and a device receiving a signal is called a receiver. A device controlling a signal is called a master and devices controlled by the master are called slaves. A master receiver must signal an end-of-data to the slave transmitter by not generating an acknowledge on the last byte clocked out of the slave device. In this case the transmitter must leave the data line high to enable the master to generate the stop event. Handshaking may also be accomplished by using the clock synchronizing mechanism. Slave devices

can hold the SCL line low, after receiving and acknowledging a byte, to force the master into a wait state until the slave device is ready for the next byte transfer. The SHI supports this feature when operating as a master device and waits until the slave device releases the SCL line before proceeding with the data transfer.

7.6.2 I²C Data Transfer Formats

The I²C bus data transfers follow the following process: after the start event, a slave device address is sent. The address consists of seven address bits and an eighth bit as a data direction bit (R/W). In the data direction bit, zero indicates a transmission (write), and one indicates a request for data (read). A data transfer is always terminated by a stop event generated by the master device. However, if the master device still wishes to communicate on the bus, it can generate another start event and address another slave device without first generating a stop event. (The SHI does not support this feature when operating as an I²C master device.) This method is also used to provide indivisible data transfers. Various combinations of read/write formats are illustrated in Table 7-10 and Figure 7-11.

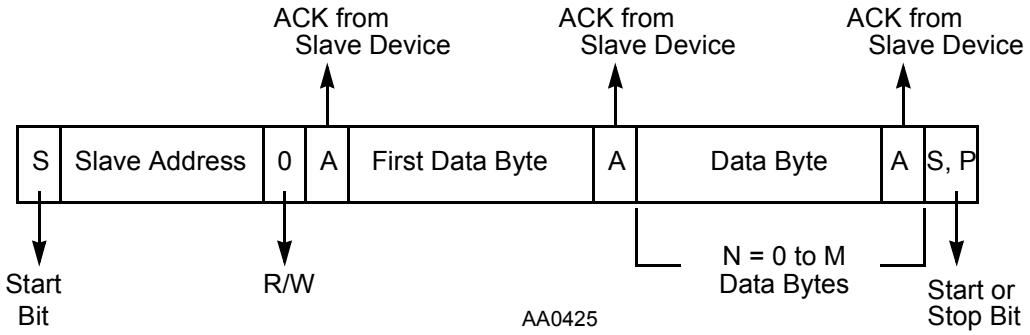


Figure 7-10. I²C Bus Protocol For Host Write Cycle

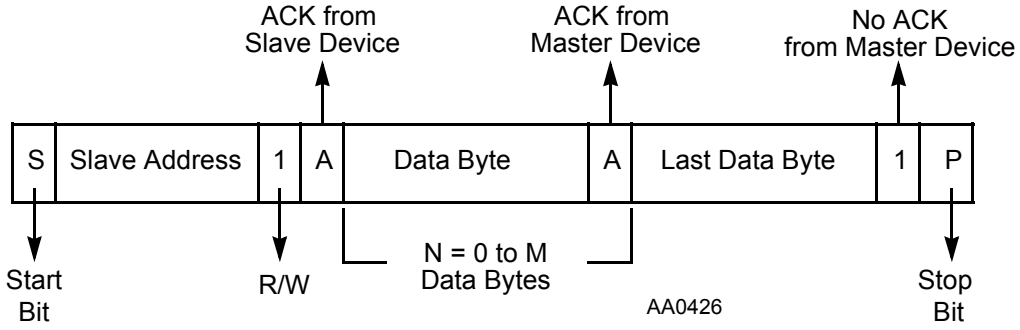


Figure 7-11. I²C Bus Protocol For Host Read Cycle

NOTE

The first data byte in a write-bus cycle can be used as a user-predefined control byte (e.g., to determine the location to which the forthcoming data bytes should be transferred).

7.7 SHI Programming Considerations

The SHI implements both SPI and I²C bus protocols and can be programmed to operate as a slave device or a single-master device. Once the operating mode is selected, the SHI may communicate with an external device by receiving and/or transmitting data. Before changing the SHI operational mode, an SHI individual reset should be generated by clearing the HEN bit. The following paragraphs describe programming considerations for each operational mode.

7.7.1 SPI Slave Mode

The SPI slave mode is entered by enabling the SHI (HEN=1), selecting the SPI mode (HI²C=0) and selecting the slave mode of operation (HMST=0). The programmer should verify that the CPHA and CPOL bits (in the HCKR) correspond to the external host clock phase and polarity. Other HCKR bits are ignored. When configured in the SPI slave mode, the SHI external pins operate as follows:

- SCK/SCL is the SCK serial clock input.
- MISO/SDA is the MISO serial data output.
- MOSI/HA0 is the MOSI serial data input.
- $\overline{SS}$/HA2 is the $\overline{SS}$ slave select input.
- $\overline{HREQ}$ is the Host Request output.

In the SPI slave mode, a receive, transmit, or full-duplex data transfer may be performed. Actually, the interface performs data receive and transmit simultaneously. The status bits of both receive and transmit paths are active; however, the programmer may disable undesired interrupts and ignore irrelevant status bits. It is recommended that an SHI individual reset (HEN cleared) be generated before beginning data reception in order to reset the HRX FIFO to its initial (empty) state (e.g., when switching from transmit to receive data).

If a write to HTX occurs, its contents are transferred to IOSR between data word transfers. The IOSR data is shifted out (via MISO) and received data is shifted in (via MOSI). The DSP may write HTX with either DSP instructions or DMA transfers if the HTDE status bit is set. If no writes to HTX occur, the contents of HTX are not transferred to IOSR, so the data shifted out when receiving is the data present in the IOSR at the time. The HRX FIFO contains valid receive data, which the DSP can read with either DSP instructions or DMA transfers (if the HRNE status bit is set).

The $\overline{\text{HREQ}}$ output pin, if enabled for receive (HRQE[1:0] = 01), is asserted when the IOSR is ready for receive and the HRX FIFO is not full; this operation guarantees that the next received data word is stored in the FIFO. The $\overline{\text{HREQ}}$ output pin, if enabled for transmit (HRQE[1:0] = 10), is asserted when the IOSR is loaded from HTX with a new data word to transfer. If $\overline{\text{HREQ}}$ is enabled for both transmit and receive (HRQE[1:0] = 11), it is asserted when the receive and transmit conditions are both true. $\overline{\text{HREQ}}$ is de-asserted at the first clock pulse of the next data word transfer. The $\overline{\text{HREQ}}$ line may be used to interrupt the external master device. Connecting the $\overline{\text{HREQ}}$ line between two SHI-equipped DSPs, one operating as an SPI master device and the other as an SPI slave device, enables full hardware handshaking if operating with CPHA = 1.

The $\overline{\text{SS}}$ line should be kept asserted during a data word transfer. If the $\overline{\text{SS}}$ line is de-asserted before the end of the data word transfer, the transfer is aborted and the received data word is lost.

7.7.2 SPI Master Mode

The SPI master mode is initiated by enabling the SHI (HEN = 1), selecting the SPI mode (HI^{2}C = 0) and selecting the master mode of operation (HMST = 1). Before enabling the SHI as an SPI master device, the programmer should program the proper clock rate, phase and polarity in HCKR. When configured in the SPI master mode, the SHI external pins operate as follows:

- SCK/SCL is the SCK serial clock output.
- MISO/SDA is the MISO serial data input.
- MOSI/HA0 is the MOSI serial data output.
- $\overline{\text{SS}}$/HA2 is the $\overline{\text{SS}}$ input. It should be kept de-asserted (high) for proper operation.
- $\overline{\text{HREQ}}$ is the Host Request input.

The external slave device can be selected either by using external logic or by activating a GPIO pin connected to its $\overline{\text{SS}}$ pin. However, the $\overline{\text{SS}}$ input pin of the SPI master device should be held de-asserted (high) for proper operation. If the SPI master device $\overline{\text{SS}}$ pin is asserted, the host bus error status bit (HBER) is set. If the HBIE bit is also set, the SHI issues a request to the DSP interrupt controller to service the SHI bus error interrupt.

In the SPI master mode the DSP must write to HTX to receive, transmit or perform a full-duplex data transfer. Actually, the interface performs simultaneous data receive and transmit. The status bits of both receive and transmit paths are active; however, the programmer may disable undesired interrupts and ignore irrelevant status bits. In a data transfer, the HTX is transferred to IOSR, clock pulses are generated, the IOSR data is shifted out (via MOSI) and received data is shifted in (via MISO). The DSP programmer may write HTX (if the HTDE status bit is set) with either DSP instructions or DMA transfers to initiate the transfer of the next word. The HRX FIFO contains valid receive data, which the DSP can read with either DSP instructions or DMA transfers, if the HRNE status bit is set.

It is recommended that an SHI individual reset (HEN cleared) be generated before beginning data reception in order to reset the receive FIFO to its initial (empty) state (e.g., when switching from transmit to receive data).

The $\overline{\text{HREQ}}$ input pin is ignored by the SPI master device if the HRQE[1:0] bits are cleared and considered if any of them is set. When asserted by the slave device, $\overline{\text{HREQ}}$ indicates that the external slave device is ready for the next data transfer. As a result, the SPI master sends clock pulses for the full data word transfer. $\overline{\text{HREQ}}$ is de-asserted by the external slave device at the first clock pulse of the new data transfer. When de-asserted, $\overline{\text{HREQ}}$ prevents the clock generation of the next data word transfer until it is asserted again. Connecting the $\overline{\text{HREQ}}$ line between two SHI-equipped DSPs, one operating as an SPI master device and the other as an SPI slave device, enables full hardware handshaking if CPHA = 1. For CPHA = 0, $\overline{\text{HREQ}}$ should be disabled by clearing HRQE[1:0].

7.7.3 I^{2}C Slave Mode

The I^{2}C slave mode is entered by enabling the SHI (HEN=1), selecting the I^{2}C mode (HI^{2}C=1) and selecting the slave mode of operation (HMST=0). In this operational mode the contents of HCKR are ignored. When configured in the I^{2}C slave mode, the SHI external pins operate as follows:

- SCK/SCL is the SCL serial clock input.
- MISO/SDA is the SDA open drain serial data line.
- MOSI/HA0 is the HA0 slave device address input.
- $\overline{\text{SS}}$/HA2 is the HA2 slave device address input.
- $\overline{\text{HREQ}}$ is the Host Request output.

When the SHI is enabled and configured in the I2C slave mode, the SHI controller inspects the SDA and SCL lines to detect a start event. Upon detection of the start event, the SHI receives the slave device address byte and enables the slave device address recognition unit. If the slave device address byte was not identified as its personal address, the SHI controller fails to acknowledge this byte by not driving low the SDA line at the ninth clock pulse (ACK = 1). However, it continues to poll the SDA and SCL lines to detect a new start event. If the personal slave device address was correctly identified, the slave device address byte is acknowledged (ACK = 0 is sent) and a receive/transmit session is initiated according to the eighth bit of the received slave device address byte, i.e., the R/$\overline{W}$ bit.

7.7.3.1 Receive Data in I2C Slave Mode

A receive session is initiated when the personal slave device address has been correctly identified and the R/$\overline{W}$ bit of the received slave device address byte has been cleared. Following a receive initiation, data in the SDA line is shifted into IOSR MSB first. Following each received byte, an acknowledge (ACK = 0) is sent at the ninth clock pulse via the SDA line. Data is acknowledged byte wise, as required by the I2C bus protocol, and is transferred to the HRX FIFO when the complete word (according to HM[1:0]) is filled into IOSR. It is the responsibility of the programmer to select the correct number of bytes in an I2C frame so that they fit in a complete number of words. For this purpose, the slave device address byte does not count as part of the data; therefore, it is treated separately.

In a receive session, only the receive path is enabled and HTX to IOSR transfers are inhibited. The HRX FIFO contains valid data, which may be read by the DSP with either DSP instructions or DMA transfers (if the HRNE status bit is set).

If HCKFR is cleared, when the HRX FIFO is full and IOSR is filled, an overrun error occurs and the HROE status bit is set. In this case, the last received byte is not acknowledged (ACK=1 is sent) and the word in the IOSR is not transferred to the HRX FIFO. This may inform the external I2C master device of the occurrence of an overrun error on the slave side. Consequently the I2C master device may terminate this session by generating a stop event.

If HCKFR is set, when the HRX FIFO is full the SHI holds the clock line to GND not letting the master device write to IOSR, which eliminates the possibility of reaching the overrun condition.

The $\overline{HREQ}$ output pin, if enabled for receive (HRQE[1:0] = 01), is asserted when the IOSR is ready to receive and the HRX FIFO is not full; this operation guarantees that the next received data word is stored in the FIFO. $\overline{HREQ}$ is de-asserted at the first clock pulse of the next received word. The $\overline{HREQ}$ line may be used to interrupt the external I2C master device. Connecting the $\overline{HREQ}$ line between two SHI-equipped DSPs, one operating as an I2C master device and the other as an I2C slave device, enables full hardware handshaking.

7.7.3.2 Transmit Data In I2C Slave Mode

A transmit session is initiated when the personal slave device address has been correctly identified and the R/$\overline{W}$ bit of the received slave device address byte has been set. Following a transmit initiation, the IOSR is loaded from HTX (assuming the latter was not empty) and its contents are shifted out, MSB first, on the SDA line. Following each transmitted byte, the SHI controller samples the SDA line at the ninth clock pulse and inspects the ACK status. If the transmitted byte was acknowledged (ACK = 0), the SHI controller continues and transmits the next byte. However, if it was not acknowledged (ACK = 1), the transmit session is stopped and the SDA line is released. Consequently, the external master device may generate a stop event in order to terminate the session.

HTX contents are transferred to IOSR when the complete word (according to HM[1:0]) has been shifted out. It is, therefore, the responsibility of the programmer to select the correct number of bytes in an I2C frame so that they fit in a complete number of words. For this purpose, the slave device address byte does not count as part of the data; therefore, it is treated separately.

In a transmit session, only the transmit path is enabled and the IOSR-to-HRX FIFO transfers are inhibited. When the HTX transfers its valid data word to IOSR, the HTDE status bit is set and the DSP may write a new data word to HTX with either DSP instructions or DMA transfers.

If HCKFR is cleared and both IOSR and HTX are empty when the master device attempts a transmit session, an underrun condition occurs, setting the HTUE status bit, and the previous word is retransmitted.

If HCKFR is set and both IOSR and HTX are empty when the master device attempts a transmit session, the SHI holds the clock line to GND to avoid an underrun condition.

The $\overline{HREQ}$ output pin, if enabled for transmit (HRQE[1:0] = 10), is asserted when HTX is transferred to IOSR for transmission. When asserted, $\overline{HREQ}$ indicates that the slave device is ready to transmit the next data word. $\overline{HREQ}$ is de-asserted at the first clock pulse of the next transmitted data word. The $\overline{HREQ}$ line may be used to interrupt the external I2C master device. Connecting the $\overline{HREQ}$ line between two SHI-equipped DSPs, one operating as an I2C master device and the other as an I2C slave device, enables full hardware handshaking.

7.7.4 I2C Master Mode

The I2C master mode is entered by enabling the SHI (HEN=1), selecting the I2C mode (HI2C=1) and selecting the master mode of operation (HMST=1). Before enabling the SHI as an I2C master, the programmer should program the appropriate clock rate in HCKR.

When configured in the I2C master mode, the SHI external pins operate as follows:

- SCK/SCL is the SCL open drain serial clock output.
- MISO/SDA is the SDA open drain serial data line.
- MOSI/HA0 is the HA0 slave device address input.

- $\overline{SS}$/HA2 is the HA2 slave device address input.
- $\overline{HREQ}$ is the Host Request input.

In the I^2C master mode, a data transfer session is always initiated by the DSP by writing to the HTX register when HIDLE is set. This condition ensures that the data byte written to HTX is interpreted as being a slave address byte. This data byte must specify the slave device address to be selected and the requested data transfer direction.

NOTE

The slave address byte should be located in the high portion of the data word, whereas the middle and low portions are ignored. Only one byte (the slave address byte) is shifted out, independent of the word length defined by the HM[1:0] bits.

In order for the DSP to initiate a data transfer the following actions are to be performed:

- The DSP tests the HIDLE status bit.
- If the HIDLE status bit is set, the DSP writes the slave device address and the R/$\overline{W}$ bit to the most significant byte of HTX.
- The SHI generates a start event.
- The SHI transmits one byte only, internally samples the R/$\overline{W}$ direction bit (last bit) and accordingly initiates a receive or transmit session.
- The SHI inspects the SDA level at the ninth clock pulse to determine the ACK value. If acknowledged (ACK = 0), it starts its receive or transmit session according to the sampled R/$\overline{W}$ value. If not acknowledged (ACK = 1), the HBER status bit in HCSR is set, which causes an SHI Bus Error interrupt request if HBIE is set, and a stop event is generated.

The $\overline{HREQ}$ input pin is ignored by the I^2C master device if HRQE[1:0] are cleared, and it is considered if either of them is set. When asserted, $\overline{HREQ}$ indicates that the external slave device is ready for the next data transfer. As a result, the I^2C master device sends clock pulses for the full data word transfer. $\overline{HREQ}$ is de-asserted by the external slave device at the first clock pulse of the next data transfer. When de-asserted, $\overline{HREQ}$ prevents the clock generation of the next data word transfer until it is asserted again. Connecting the $\overline{HREQ}$ line between two SHI-equipped DSPs, one operating as an I^2C master device and the other as an I^2C slave device, enables full hardware handshaking.

7.7.4.1 Receive Data in I^2C Master Mode

A receive session is initiated if the R/$\overline{W}$ direction bit of the transmitted slave device address byte is set. Following a receive initiation, data in the SDA line is shifted into IOSR MSB first. Following each received byte, an acknowledge (ACK = 0) is sent at the ninth clock pulse via the SDA line if the HIDLE control bit is cleared. Data is acknowledged byte-wise, as required by the I^2C bus protocol, and is transferred to the HRX FIFO when the complete word (according to HM[1:0]) is filled into IOSR. It is the responsibility of the programmer to select the correct number of bytes in an I^2C frame so that they fit in a complete number of words. For this purpose, the slave device address byte does not count as part of the data; therefore, it is treated separately.

If the I^2C slave transmitter is acknowledged, it should transmit the next data byte. In order to terminate the receive session, the programmer should set the HIDLE bit at the last required data word. As a result, the last byte of the next received data word is not acknowledged, the slave transmitter releases the SDA line, and the SHI generates the stop event and terminates the session.

In a receive session, only the receive path is enabled and the HTX-to-IOSR transfers are inhibited. If the HRNE status bit is set, the HRX FIFO contains valid data, which may be read by the DSP with either DSP instructions or DMA transfers. When the HRX FIFO is full, the SHI suspends the serial clock just before acknowledge. In this case, the clock is reactivated when the FIFO is read (the SHI gives an ACK = 0 and proceeds receiving).

7.7.4.2 Transmit Data In I^2C Master Mode

A transmit session is initiated if the R/$\overline{W}$ direction bit of the transmitted slave device address byte is cleared. Following a transmit initiation, the IOSR is loaded from HTX (assuming HTX is not empty) and its contents are shifted out, MSB-first, on the SDA line. Following each transmitted byte, the SHI controller samples the SDA line at the ninth clock pulse and inspects the ACK status. If the transmitted byte was acknowledged (ACK=0), the SHI controller continues transmitting the next byte. However, if it was not acknowledged (ACK=1), the HBER status bit is set to inform the DSP side that a bus error (or overrun, or any other exception in the slave device) has occurred. Consequently, the I^2C master device generates a stop event and terminates the session.

HTX contents are transferred to the IOSR when the complete word (according to HM[1:0]) has been shifted out. It is, therefore, the responsibility of the programmer to select the right number of bytes in an I^2C frame so that they fit in a complete number of words. Remember that for this purpose, the slave device address byte does not count as part of the data.

In a transmit session, only the transmit path is enabled and the IOSR-to-HRX FIFO transfers are inhibited. When the HTX transfers its valid data word to the IOSR, the HTDE status bit is set and the DSP may write a new data word to HTX with either DSP instructions or DMA transfers. If both IOSR and HTX are empty, the SHI suspends the serial clock until new data is written into HTX (when the SHI proceeds with the transmit session) or HIDLE is set (the SHI reactivates the clock to generate the stop event and terminate the transmit session).

7.7.5 SHI Operation During DSP Stop

The SHI operation cannot continue when the DSP is in the stop state, because no DSP clocks are active. While the DSP is in the stop state, the SHI remains in the individual reset state.

While in the individual reset state the following is true:

- If the SHI was operating in the I^2C mode, the SHI signals are disabled (high impedance state).
- If the SHI was operating in the SPI mode, the SHI signals are not affected.
- The HCSR status bits and the transmit/receive paths are reset to the same state produced by hardware reset or software reset.
- The HCSR and HCKR control bits are not affected.

NOTE

It is recommended that the SHI be disabled before entering the stop state.

7.7.6 GPIO- HREQ Signal and Registers

Note that the HREQ pin can also be programmed as a GPIO. See Section 6.2.3, *Port H Signals and Registers*.

Notes

Chapter 8
Enhanced Serial Audio Interface (ESAI)

8.1 Introduction

The Enhanced Serial Audio Interface (ESAI) provides a full-duplex serial port for serial communication with a variety of devices, including industry-standard codecs, SPDIF transceivers, and other DSPs. The ESAI consists of independent transmitter and receiver sections, each section with its own clock generator. It is a superset of the 56300 Family ESSI peripheral and of the 56000 Family SAI peripheral.

NOTE

There are two independent and identical ESAIs in the DSP56374:ESAI and ESAI_1. For simplicity, a single generic ESAI is described here.

The ESAI block diagram is shown in Figure 8-1. The ESAI is named synchronous because all serial transfers are synchronized to a clock. Additional synchronization signals are used to delineate the word frames. The normal mode of operation is used to transfer data at a periodic rate, one word per period. The network mode is similar in that it is also intended for periodic transfers; however, it supports up to 32 words (time slots) per period. This mode can be used to build time division multiplexed (TDM) networks. In contrast, the on-demand mode is intended for non-periodic transfers of data and to transfer data serially at high speed when the data becomes available.

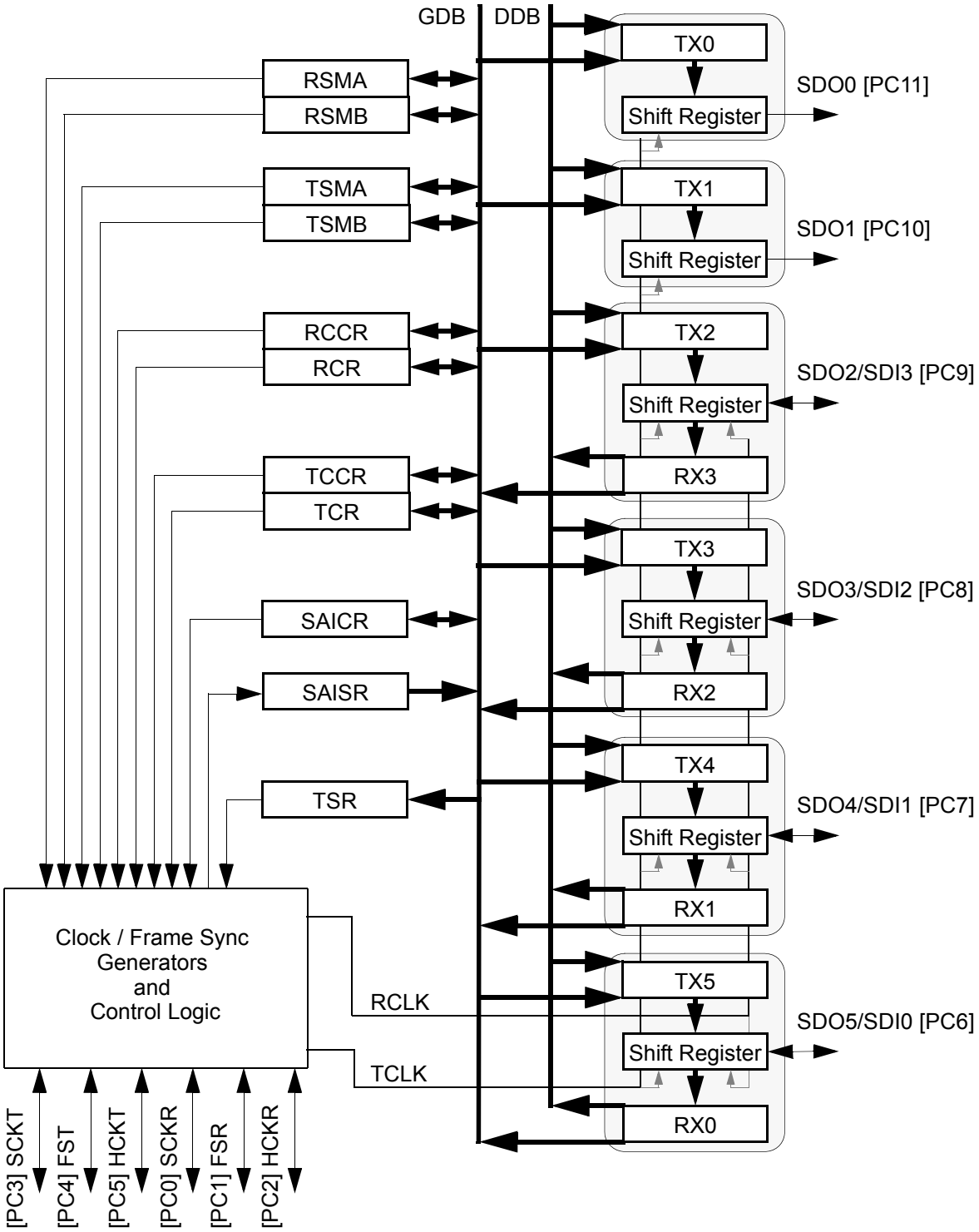


Figure 8-1. ESAI Block Diagram

8.2 ESAI Data and Control Pins

Three to twelve pins are required for operation, depending on the operating mode selected and the number of transmitters and receivers enabled. The SDO0 and SDO1 pins are used by transmitters 0 and 1 only. The SDO2/SDI3, SDO3/SDI2, SDO4/SDI1 and SDO5/SDI0 pins are shared by transmitters 2 to 5 with receivers 0 to 3. The actual mode of operation is selected under software control. All transmitters operate fully synchronized under control of the same transmitter clock signals. All receivers operate fully synchronized under control of the same receiver clock signals.

8.2.1 Serial Transmit 0 Data Pin (SDO0)

SDO0 is used for transmitting data from the TX0 serial transmit shift register. SDO0 is an output when data is being transmitted from the TX0 shift register. In the on-demand mode with an internally generated bit clock, the SDO0 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO0 may be programmed as a general-purpose I/O pin (PC11) when the ESAI SDO0 function is not being used.

8.2.2 Serial Transmit 1 Data Pin (SDO1)

SDO1 is used for transmitting data from the TX1 serial transmit shift register. SDO1 is an output when data is being transmitted from the TX1 shift register. In the on-demand mode with an internally generated bit clock, the SDO1 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO1 may be programmed as a general-purpose I/O pin (PC10) when the ESAI SDO1 function is not being used.

8.2.3 Serial Transmit 2/Receive 3 Data Pin (SDO2/SDI3)

SDO2/SDI3 is used as the SDO2 for transmitting data from the TX2 serial transmit shift register when programmed as a transmitter pin, or as the SDI3 signal for receiving serial data to the RX3 serial receive shift register when programmed as a receiver pin. SDO2/SDI3 is an input when data is being received by the RX3 shift register. SDO2/SDI3 is an output when data is being transmitted from the TX2 shift register. In the on-demand mode with an internally generated bit clock, the SDO2/SDI3 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO2/SDI3 may be programmed as a general-purpose I/O pin (PC9) when the ESAI SDO2 and SDI3 functions are not being used.

8.2.4 Serial Transmit 3/Receive 2 Data Pin (SDO3/SDI2)

SDO3/SDI2 is used as the SDO3 signal for transmitting data from the TX3 serial transmit shift register when programmed as a transmitter pin, or as the SDI2 signal for receiving serial data to the RX2 serial receive shift register when programmed as a receiver pin. SDO3/SDI2 is an input when data is being received by the RX2 shift register. SDO3/SDI2 is an output when data is being transmitted from the TX3 shift register. In the on-demand mode with an internally generated bit clock, the SDO3/SDI2 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO3/SDI2 may be programmed as a general-purpose I/O pin (PC8) when the ESAI SDO3 and SDI2 functions are not being used.

8.2.5 Serial Transmit 4/Receive 1 Data Pin (SDO4/SDI1)

SDO4/SDI1 is used as the SDO4 signal for transmitting data from the TX4 serial transmit shift register when programmed as transmitter pin, or as the SDI1 signal for receiving serial data to the RX1 serial receive shift register when programmed as a receiver pin. SDO4/SDI1 is an input when data is being received by the RX1 shift register. SDO4/SDI1 is an output when data is being transmitted from the TX4 shift register. In the on-demand mode with an internally generated bit clock, the SDO4/SDI1 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO4/SDI1 may be programmed as a general-purpose I/O pin (PC7) when the ESAI SDO4 and SDI1 functions are not being used.

8.2.6 Serial Transmit 5/Receive 0 Data Pin (SDO5/SDI0)

SDO5/SDI0 is used as the SDO5 signal for transmitting data from the TX5 serial transmit shift register when programmed as transmitter pin, or as the SDI0 signal for receiving serial data to the RX0 serial shift register when programmed as a receiver pin. SDO5/SDI0 is an input when data is being received by the RX0 shift register. SDO5/SDI0 is an output when data is being transmitted from the TX5 shift register. In the on-demand mode with an internally generated bit clock, the SDO5/SDI0 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO5/SDI0 may be programmed as a general-purpose I/O pin (PC6) when the ESAI SDO5 and SDI0 functions are not being used

8.2.7 Receiver Serial Clock (SCKR)

SCKR is a bidirectional pin providing the receivers serial bit clock for the ESAI interface. The direction of this pin is determined by the RCKD bit in the RCCR register.The SCKR operates as a clock input or output used by all the enabled receivers in the asynchronous mode (SYN=0), or as serial flag 0 pin in the synchronous mode (SYN=1).

When this pin is configured as serial flag pin, its direction is determined by the RCKD bit in the RCCR register. When configured as the output flag OF0, this pin reflects the value of the OF0 bit in the SAICR register, and the data in the OF0 bit shows up at the pin synchronized to the frame sync being used by the transmitter and receiver sections. When this pin is configured as the input flag IF0, the data value at the pin is stored in the IF0 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.

SCKR may be programmed as a general-purpose I/O pin (PC0) when the ESAI SCKR function is not being used.

NOTE

Although the external ESAI serial clock can be independent of and asynchronous to the DSP system clock, the DSP clock frequency (Fosc) must be at least four times the external ESAI serial clock frequency and each ESAI serial clock phase must exceed the minimum of 2 DSP clock periods.

For more information on pin mode and definition, see Table 8-7 and on receiver clock signals see Table 8-1.

Table 8-1. Receiver Clock Sources (asynchronous mode only)

RHCKD	RFSD	RCKD	ERI0	ERO0	Receiver Bit Clock Source	OUTPUTS		
0	0	0	N/A	N/A	SCKR			
0	0	1	N/A	N/A	HCKR			SCKR
0	1	0	N/A	N/A	SCKR		FSR	
0	1	1	N/A	N/A	HCKR		FSR	SCKR
1	0	0	0	0	SCKR	HCKR		
1	0	0	0	1	SCKR	HCKR		
1	0	0	1	0	SCKR	HCKR		
1	0	0	1	1	SCKR	HCKR		
1	0	1	0	0	Fosc	HCKR		SCKR
1	0	1	0	1	Fosc	HCKR		SCKR
1	0	1	1	0	EXTAL	HCKR		SCKR
1	0	1	1	1	EXTAL	HCKR		SCKR
1	1	0	0	0	SCKR	HCKR	FSR	
1	1	0	0	1	SCKR	HCKR	FSR	
1	1	0	1	0	SCKR	HCKR	FSR	
1	1	0	1	1	SCKR	HCKR	FSR	
1	1	1	0	0	Fosc	HCKR	FSR	SCKR
1	1	1	0	1	Fosc	HCKR	FSR	SCKR
1	1	1	1	0	EXTAL	HCKR	FSR	SCKR
1	1	1	1	1	EXTAL	HCKR	FSR	SCKR

8.2.8 Transmitter Serial Clock (SCKT)

SCKT is a bidirectional pin providing the transmitters serial bit clock for the ESAI interface. The direction of this pin is determined by the TCKD bit in the TCCR register. The SCKT is a clock input or output used by all the enabled transmitters in the asynchronous mode (SYN=0) or by all the enabled transmitters and receivers in the synchronous mode (SYN=1) (see Table 8-2).

Table 8-2. Transmitter Clock Sources

THCKD	TFSD	TCKD	ETI0	ETO0	Transmitter Bit Clock Source	OUTPUTS		
0	0	0	N/A	N/A	SCKT			
0	0	1	N/A	N/A	HCKT			SCKT
0	1	0	N/A	N/A	SCKT		FST	
0	1	1	N/A	N/A	HCKT		FST	SCKT
1	0	0	0	0	SCKT	HCKT		
1	0	0	0	1	SCKT	HCKT		
1	0	0	1	0	SCKT	HCKT		
1	0	0	1	1	SCKT	HCKT		
1	0	1	0	0	Fosc	HCKT		SCKT
1	0	1	0	1	Fosc	HCKT		SCKT
1	0	1	1	0	EXTAL	HCKT		SCKT
1	0	1	1	1	EXTAL	HCKT		SCKT
1	1	0	0	0	SCKT	HCKT	FST	
1	1	0	0	1	SCKT	HCKT	FST	
1	1	0	1	0	SCKT	HCKT	FST	
1	1	0	1	1	SCKT	HCKT	FST	
1	1	1	0	0	Fosc	HCKT	FST	SCKT
1	1	1	0	1	Fosc	HCKT	FST	SCKT
1	1	1	1	0	EXTAL	HCKT	FST	SCKT
1	1	1	1	1	EXTAL	HCKT	FST	SCKT

SCKT may be programmed as a general-purpose I/O pin (PC3) when the ESAI SCKT function is not being used.

NOTE

Although the external ESAI serial clock can be independent of and asynchronous to the DSP system clock, the DSP clock frequency must be at least three times the external ESAI serial clock frequency and each ESAI serial clock phase must exceed the minimum of 1.5 DSP clock periods.

8.2.9 Frame Sync for Receiver (FSR)

FSR is a bidirectional pin providing the receivers frame sync signal for the ESAI interface. The direction of this pin is determined by the RFSD bit in RCR register. In the asynchronous mode (SYN=0), the FSR pin operates as the frame sync input or output used by all the enabled receivers. In the synchronous mode (SYN=1), it operates as either the serial flag 1 pin (TEBE=0), or as the transmitter external buffer enable control (TEBE=1, RFSD=1). For further information on pin mode and definition, see Table 8-8 and on receiver clock signals see Table 8-1.

When this pin is configured as serial flag pin, its direction is determined by the RFSD bit in the RCCR register. When configured as the output flag OF1, this pin reflects the value of the OF1 bit in the SAICR register, and the data in the OF1 bit shows up at the pin synchronized to the frame sync being used by the transmitter and receiver sections. When configured as the input flag IF1, the data value at the pin is stored in the IF1 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.

FSR may be programmed as a general-purpose I/O pin (PC1) when the ESAI FSR function is not being used.

8.2.10 Frame Sync for Transmitter (FST)

FST is a bidirectional pin providing the frame sync for both the transmitters and receivers in the synchronous mode (SYN=1) and for the transmitters only in asynchronous mode (SYN=0) (see Table 8-2). The direction of this pin is determined by the TFSD bit in the TCR register. When configured as an output, this pin is the internally generated frame sync signal. When configured as an input, this pin receives an external frame sync signal for the transmitters (and the receivers in synchronous mode).

FST may be programmed as a general-purpose I/O pin (PC4) when the ESAI FST function is not being used.

8.2.11 High Frequency Clock for Transmitter (HCKT)

HCKT is a bidirectional pin providing the transmitters high frequency clock for the ESAI interface. The direction of this pin is determined by the THCKD bit in the TCCR register. In the asynchronous mode (SYN=0), the HCKT pin operates as the high frequency clock input or output used by all enabled transmitters. In the synchronous mode (SYN=1), it operates as the high frequency clock input or output used by all enabled transmitters and receivers. When programmed as input this pin is used as an alternative high frequency clock source to the ESAI transmitter rather than the DSP main clock. When programmed as output it can serve as a high frequency sample clock (to external DACs for example) or as an additional system clock. See Table 8-2.

HCKT may be programmed as a general-purpose I/O pin (PC5) when the ESAI HCKT function is not being used.

8.2.12 High Frequency Clock for Receiver (HCKR)

HCKR is a bidirectional pin providing the receivers high frequency clock for the ESAI interface. The direction of this pin is determined by the RHCKD bit in the RCCR register. In the asynchronous mode (SYN=0), the HCKR pin operates as the high frequency clock input or output used by all the enabled receivers. In the synchronous mode (SYN=1), it operates as the serial flag 2 pin. For further information on pin mode and definition, see Table 8-9 and on receiver clock signals see Table 8-1.

When this pin is configured as serial flag pin, its direction is determined by the RHCKD bit in the RCCR register. When configured as the output flag OF2, this pin reflects the value of the OF2 bit in the SAICR register, and the data in the OF2 bit shows up at the pin synchronized to the frame sync being used by the transmitter and receiver sections. When configured as the input flag IF2, the data value at the pin is stored in the IF2 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.

HCKR may be programmed as a general-purpose I/O pin (PC2) when the ESAI HCKR function is not being used.

8.3 ESAI Programming Model

The ESAI can be viewed as five control registers, one status register, six transmit data registers, four receive data registers, two transmit slot mask registers, two receive slot mask registers and a special-purpose time slot register. The following paragraphs give detailed descriptions and operations of each bit in the ESAI registers.

The ESAI pins can also function as GPIO pins, described in Section 8.5, *GPIO - Pins and Registers*.

8.3.1 ESAI Transmitter Clock Control Register (TCCR)

The read/write Transmitter Clock Control Register (TCCR) controls the ESAI transmitter clock generator bit and frame sync rates, the bit clock and high frequency clock sources and the directions of the HCKT, FST and SCKT signals. (See Figure 8-2). The PDRC register provides additional clocking options by allowing the use of EXTAL as the clock source to the ESAI transmitter as shown in Table 8-2. (Also see Figure 8-21). In the synchronous mode (SYN=1), the bit clock defined for the transmitter determines the receiver bit clock as well. TCCR also controls the number of words per frame for the serial data. Hardware and software reset clear all the bits of the TCCR register.

11	10	9	8	7	6	5	4	3	2	1	0
TDC2	TDC1	TDC0	TPSR	TPM7	TPM6	TPM5	TPM4	TPM3	TPM2	TPM1	TPM0

23	22	21	20	19	18	17	16	15	14	13	12
THCKD	TFSD	TCKD	THCKP	TFSP	TCKP	TFP3	TFP2	TFP1	TFP0	TDC4	TDC3

Figure 8-2. TCCR Register

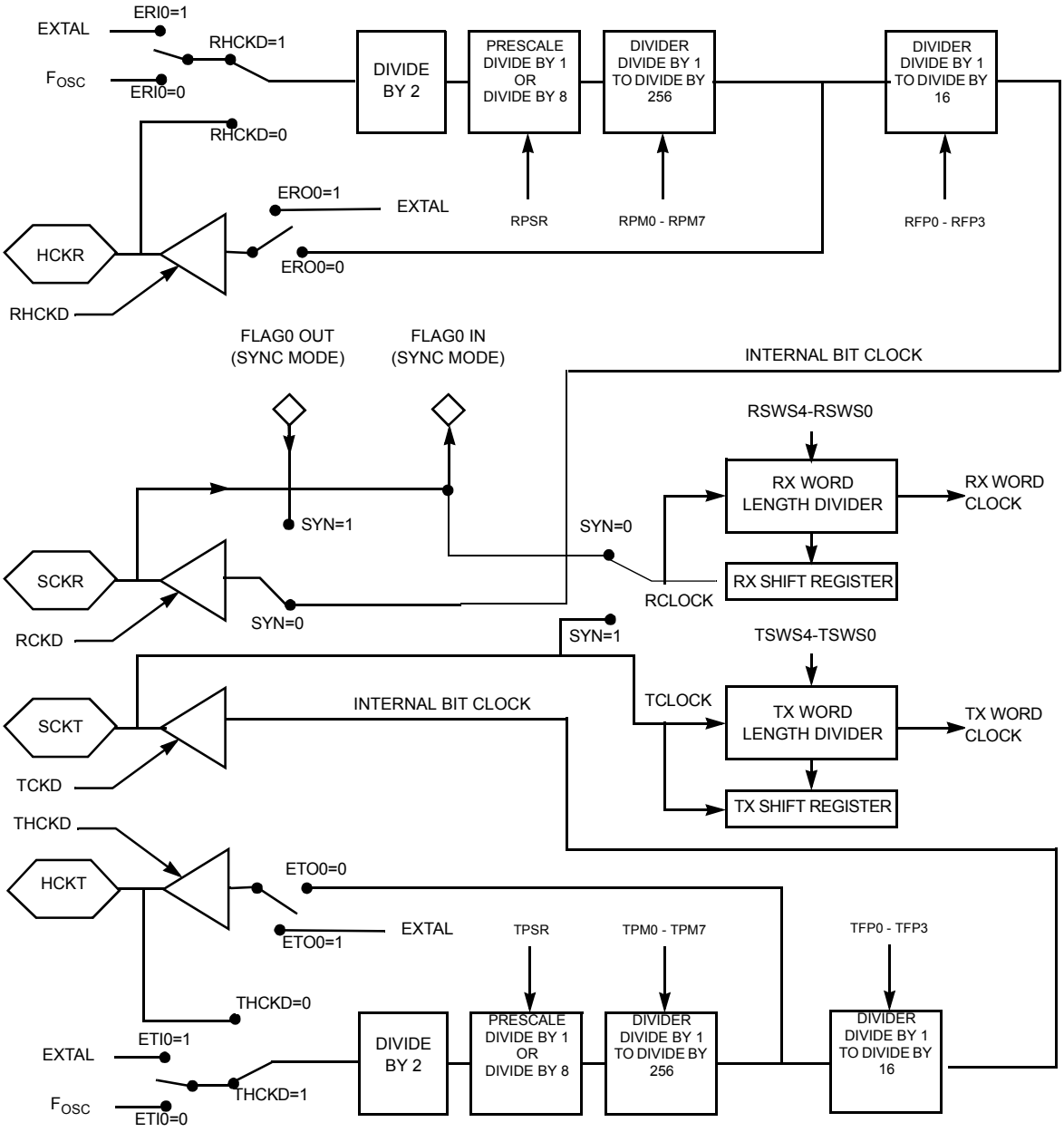
The ESAI TCCR register is located at x:$FFFFB6. The ESAI_1 TCCR register is located at y:$FFFF96.

Note that care should be taken in asynchronous mode whenever the framesync clock (FSR, FST) is not sourced directly from its associated bit clock (SCKR,SCKT). Proper phase relationships must be maintained between these clocks in order to guarantee proper operation of the ESAI.

The TCCR control bits are described in the following paragraphs.

8.3.1.1 TCCR Transmit Prescale Modulus Select (TPM7–TPM0) - Bits 7–0

The TPM7–TPM0 bits specify the divide ratio of the prescale divider in the ESAI transmitter clock generator. A divide ratio from 1 to 256 (TPM[7:0]=$00 to $FF) may be selected. The bit clock output is available at the transmit serial bit clock (SCKT) pin of the DSP. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. The ESAI transmit clock generator functional diagram is shown in Figure 8-3.



Notes:
1.ETIx, ETOx, ERIx and EROx bit descriptions are covered in Section 6.2.2.4, *ESAI/EXTAL Clocking Control*
2. Fosc is the DSP56300 Core internal clock frequency.

Figure 8-3. ESAI Clock Generator Functional Block Diagram

8.3.1.2 TCCR Transmit Prescaler Range (TPSR) - Bit 8

The TPSR bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When TPSR is set, the fixed prescaler is bypassed. When TPSR is cleared, the fixed divide-by-eight prescaler is operational (see Figure 8-3). The maximum internally generated bit clock frequency is Fosc/4; the minimum internally generated bit clock frequency is Fosc/(2 x 8 x 256)=Fosc/4096.

NOTE

Do not use the combination TPSR=1, TPM7-TPM0=$00, and TFP3-TFP0=$0 which causes synchronization problems when using the internal DSP clock as source (TCKD=1 or THCKD=1).

8.3.1.3 TCCR Tx Frame Rate Divider Control (TDC4–TDC0) - Bits 13–9

The TDC4–TDC0 bits control the divide ratio for the programmable frame rate dividers used to generate the transmitter frame clocks.

In network mode, this ratio may be interpreted as the number of words per frame minus one. The divide ratio may range from 2 to 32 (TDC[4:0]=00001 to 11111) for network mode. A divide ratio of one (TDC[4:0]=00000) in network mode is a special case (on-demand mode).

In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (TDC[4:0]=00000 to 11111) for normal mode. In normal mode, a divide ratio of 1 (TDC[4:0]=00000) provides continuous periodic data word transfers. A bit-length frame sync (TFSL=1) must be used in this case.

The ESAI frame sync generator functional diagram is shown in Figure 8-4.

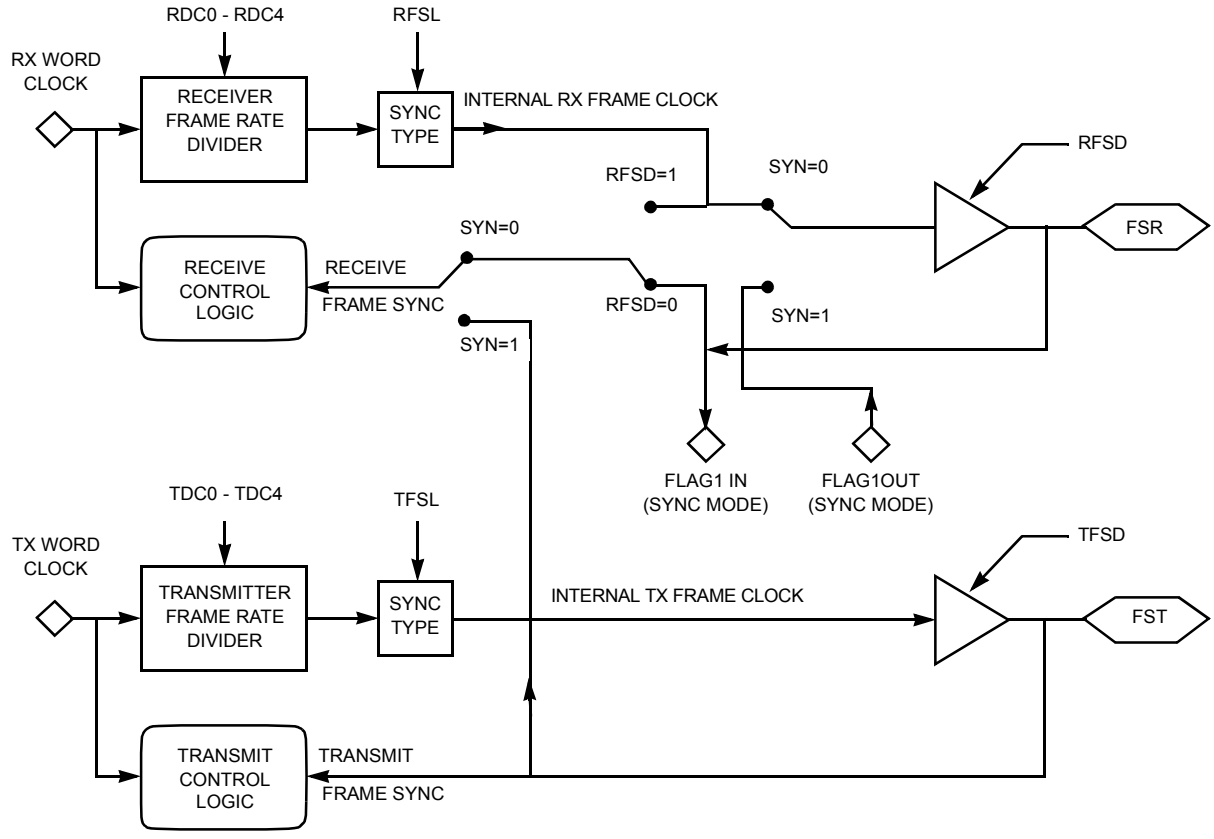


Figure 8-4. ESAI Frame Sync Generator Functional Block Diagram

8.3.1.4 TCCR Tx High Frequency Clock Divider (TFP3-TFP0) - Bits 17–14

The TFP3–TFP0 bits control the divide ratio of the transmitter high frequency clock to the transmitter serial bit clock when the source of the high frequency clock and the bit clock is the internal DSP clock. When the HCKT input is being driven from an external high frequency clock, the TFP3-TFP0 bits specify an additional division ratio in the clock divider chain. See Table 8-3 for the specification of the divide ratio. The ESAI high frequency clock generator functional diagram is shown in Figure 8-3.

Table 8-3. Transmitter High Frequency Clock Divider

TFP3-TFP0	Divide Ratio
$0	1
$1	2
$2	3
$3	4
...	...
$F	16

8.3.1.5 TCCR Transmit Clock Polarity (TCKP) - Bit 18

The Transmit Clock Polarity (TCKP) bit controls on which transmit bit clock edge the transmit data lines are clocked out, and the transmit frame sync is either clocked out if defined as an output or latched in if defined as an input.

If the TCKP bit is cleared, the transmit data lines are clocked out on the rising edge of the transmit bit clock. The transmit frame sync is clocked out on the rising edge of the transmit bit clock if it is defined as an output, or it is latched in on the falling edge of the transmit bit clock if defined as an input.

If the TCKP bit is set, the transmit data lines are clocked out on the falling edge of the transmit bit clock. The transmit frame sync is clocked out on the falling edge of the transmit clock if it is defined as an output, or it is latched in on the rising edge of the transmit bit clock if defined as an input.

8.3.1.6 TCCR Transmit Frame Sync Polarity (TFSP) - Bit 19

The Transmitter Frame Sync Polarity (TFSP) bit determines the polarity of the transmit frame sync signal. When TFSP is cleared, the frame sync signal polarity is positive, i.e., the frame start is indicated by a high level on the frame sync pin. When TFSP is set, the frame sync signal polarity is negative, i.e., the frame start is indicated by a low level on the frame sync pin.

8.3.1.7 TCCR Transmit High Frequency Clock Polarity (THCKP) - Bit 20

The Transmitter High Frequency Clock Polarity (THCKP) bit controls on which bit clock edge data and frame sync are clocked out and latched in. If THCKP is cleared the data and the frame sync are clocked out on the rising edge of the transmit high frequency bit clock and latched in on the falling edge of the transmit bit clock. If THCKP is set the falling edge of the transmit clock is used to clock the data out and frame sync and the rising edge of the transmit clock is used to latch the data and frame sync in.

8.3.1.8 TCCR Transmit Clock Source Direction (TCKD) - Bit 21

The Transmitter Clock Source Direction (TCKD) bit selects the source of the clock signal used to clock the transmit shift registers in the asynchronous mode (SYN=0) and the transmit shift registers and the receive shift registers in the synchronous mode (SYN=1). When TCKD is set, the internal clock source becomes the bit clock for the transmit shift registers and word length divider and is the output on the SCKT pin. When TCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCKT pin, and an external clock source may drive this pin. See Table 8-2.

8.3.1.9 TCCR Transmit Frame Sync Signal Direction (TFSD) - Bit 22

TFSD controls the direction of the FST pin. When TFSD is cleared, FST is an input; when TFSD is set, FST is an output. See Table 8-2.

8.3.1.10 TCCR Transmit High Frequency Clock Direction (THCKD) - Bit 23

THCKD controls the direction of the HCKT pin. When THCKD is cleared, HCKT is an input; when THCKD is set, HCKT is an output. See Table 8-2.

8.3.2 ESAI Transmit Control Register (TCR)

The read/write Transmit Control Register (TCR) controls the ESAI transmitter section. Interrupt enable bits for the transmitter section are provided in this control register. Operating modes are also selected in this register.See Figure 8-5.

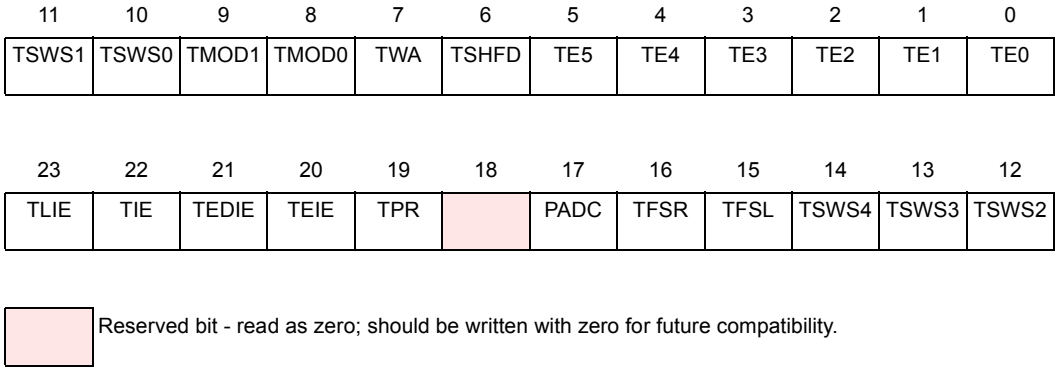


Figure 8-5. TCR Register

Hardware and software reset clear all the bits in the TCR register. The ESAI TCR register is located at x:$FFFFB5. The ESAI_1 TCR register is located at y:$FFFF95.

The TCR bits are described in the following paragraphs.

8.3.2.1 TCR ESAI Transmit 0 Enable (TE0) - Bit 0

TE0 enables the transfer of data from TX0 to the transmit shift register #0. When TE0 is set and a frame sync is detected, the transmit #0 portion of the ESAI is enabled for that frame. When TE0 is cleared, the transmitter #0 is disabled after completing transmission of data currently in the ESAI transmit shift register. The SDO0 output is tri-stated, and any data present in TX0 is not transmitted, i.e., data can be written to TX0 with TE0 cleared, but data is not transferred to the transmit shift register #0.

The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one.

In the network mode, the operation of clearing TE0 and setting it again disables the transmitter #0 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO0 pin remains in the high-impedance state.The on-demand mode transmit enable sequence can be the same as the normal mode, or TE0 can be left enabled.

8.3.2.2 TCR ESAI Transmit 1 Enable (TE1) - Bit 1

TE1 enables the transfer of data from TX1 to the transmit shift register #1. When TE1 is set and a frame sync is detected, the transmit #1 portion of the ESAI is enabled for that frame. When TE1 is cleared, the transmitter #1 is disabled after completing transmission of data currently in the ESAI transmit shift register. The SDO1 output is tri-stated, and any data present in TX1 is not transmitted, i.e., data can be written to TX1 with TE1 cleared, but data is not transferred to the transmit shift register #1.

The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one.

In the network mode, the operation of clearing TE1 and setting it again disables the transmitter #1 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO1 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE1 can be left enabled.

8.3.2.3 TCR ESAI Transmit 2 Enable (TE2) - Bit 2

TE2 enables the transfer of data from TX2 to the transmit shift register #2. When TE2 is set and a frame sync is detected, the transmit #2 portion of the ESAI is enabled for that frame. When TE2 is cleared, the transmitter #2 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX2 when TE2 is cleared but the data is not transferred to the transmit shift register #2.

The SDO2/SDI3 pin is the data input pin for RX3 if TE2 is cleared and RE3 in the RCR register is set. If both RE3 and TE2 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE3 and TE2 should not be set at the same time.

The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one.

In the network mode, the operation of clearing TE2 and setting it again disables the transmitter #2 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO2/SDI3 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE2 can be left enabled.

8.3.2.4 TCR ESAI Transmit 3 Enable (TE3) - Bit 3

TE3 enables the transfer of data from TX3 to the transmit shift register #3. When TE3 is set and a frame sync is detected, the transmit #3 portion of the ESAI is enabled for that frame. When TE3 is cleared, the transmitter #3 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX3 when TE3 is cleared but the data is not transferred to the transmit shift register #3.

The SDO3/SDI2 pin is the data input pin for RX2 if TE3 is cleared and RE2 in the RCR register is set. If both RE2 and TE3 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE2 and TE3 should not be set at the same time.

The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one.

In the network mode, the operation of clearing TE3 and setting it again disables the transmitter #3 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO3/SDI2 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE3 can be left enabled.

8.3.2.5 TCR ESAI Transmit 4 Enable (TE4) - Bit 4

TE4 enables the transfer of data from TX4 to the transmit shift register #4. When TE4 is set and a frame sync is detected, the transmit #4 portion of the ESAI is enabled for that frame. When TE4 is cleared, the transmitter #4 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX4 when TE4 is cleared but the data is not transferred to the transmit shift register #4.

The SDO4/SDI1 pin is the data input pin for RX1 if TE4 is cleared and RE1 in the RCR register is set. If both RE1 and TE4 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE1 and TE4 should not be set at the same time.

The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one.

In the network mode, the operation of clearing TE4 and setting it again disables the transmitter #4 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO4/SDI1 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE4 can be left enabled.

8.3.2.6 TCR ESAI Transmit 5 Enable (TE5) - Bit 5

TE5 enables the transfer of data from TX5 to the transmit shift register #5. When TE5 is set and a frame sync is detected, the transmit #5 portion of the ESAI is enabled for that frame. When TE5 is cleared, the transmitter #5 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX5 when TE5 is cleared but the data is not transferred to the transmit shift register #5.

The SDO5/SDI0 pin is the data input pin for RX0 if TE5 is cleared and RE0 in the RCR register is set. If both RE0 and TE5 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE0 and TE5 should not be set at the same time.

The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one.

In the network mode, the operation of clearing TE5 and setting it again disables the transmitter #5 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO5/SDI0 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE5 can be left enabled.

8.3.2.7 TCR Transmit Shift Direction (TSHFD) - Bit 6

The TSHFD bit causes the transmit shift registers to shift data out MSB first when TSHFD equals zero or LSB first when TSHFD equals one (see Figure 8-13 and Figure 8-14).

8.3.2.8 TCR Transmit Word Alignment Control (TWA) - Bit 7

The Transmitter Word Alignment Control (TWA) bit defines the alignment of the data word in relation to the slot. This is relevant for the cases where the word length is shorter than the slot length. If TWA is cleared, the data word is left-aligned in the slot frame during transmission. If TWA is set, the data word is right-aligned in the slot frame during transmission.

Since the data word is shorter than the slot length, the data word is extended until achieving the slot length, according to the following rule:

1. If the data word is left-aligned (TWA=0), and zero padding is disabled (PADC=0), the last data bit is repeated after the data word has been transmitted. If zero padding is enabled (PADC=1), zeroes are transmitted after the data word has been transmitted.
2. If the data word is right-aligned (TWA=1), and zero padding is disabled (PADC=0), the first data bit is repeated before the transmission of the data word. If zero padding is enabled (PADC=1), zeroes are transmitted before the transmission of the data word.

8.3.2.9 TCR Transmit Network Mode Control (TMOD1-TMOD0) - Bits 9-8

The TMOD1 and TMOD0 bits are used to define the network mode of ESAI transmitters according to Figure 8-4. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot, as shown in Figure 8-6. In network mode, it is possible to transfer a word for every time slot, as shown in Figure 8-6. For more details, see Section 8.4, *Operating Modes*.

In order to comply with AC-97 specifications, TSWS4-TSWS0 should be set to 00011 (20-bit slot, 20-bit word length), TFSL and TFSR should be cleared, and TDC4-TDC0 should be set to $0C (13 words in frame). If TMOD[1:0]=$11 and the above recommendations are followed, the first slot and word will be 16 bits long, and the next 12 slots and words will be 20 bits long, as required by the AC97 protocol.

Table 8-4. Transmit Network Mode Selection

TMOD1	TMOD0	TDC4-TDC0	Transmitter Network Mode
0	0	$0-$1F	Normal Mode
0	1	$0	On-Demand Mode
0	1	$1-$1F	Network Mode
1	0	X	Reserved
1	1	$0C	AC97

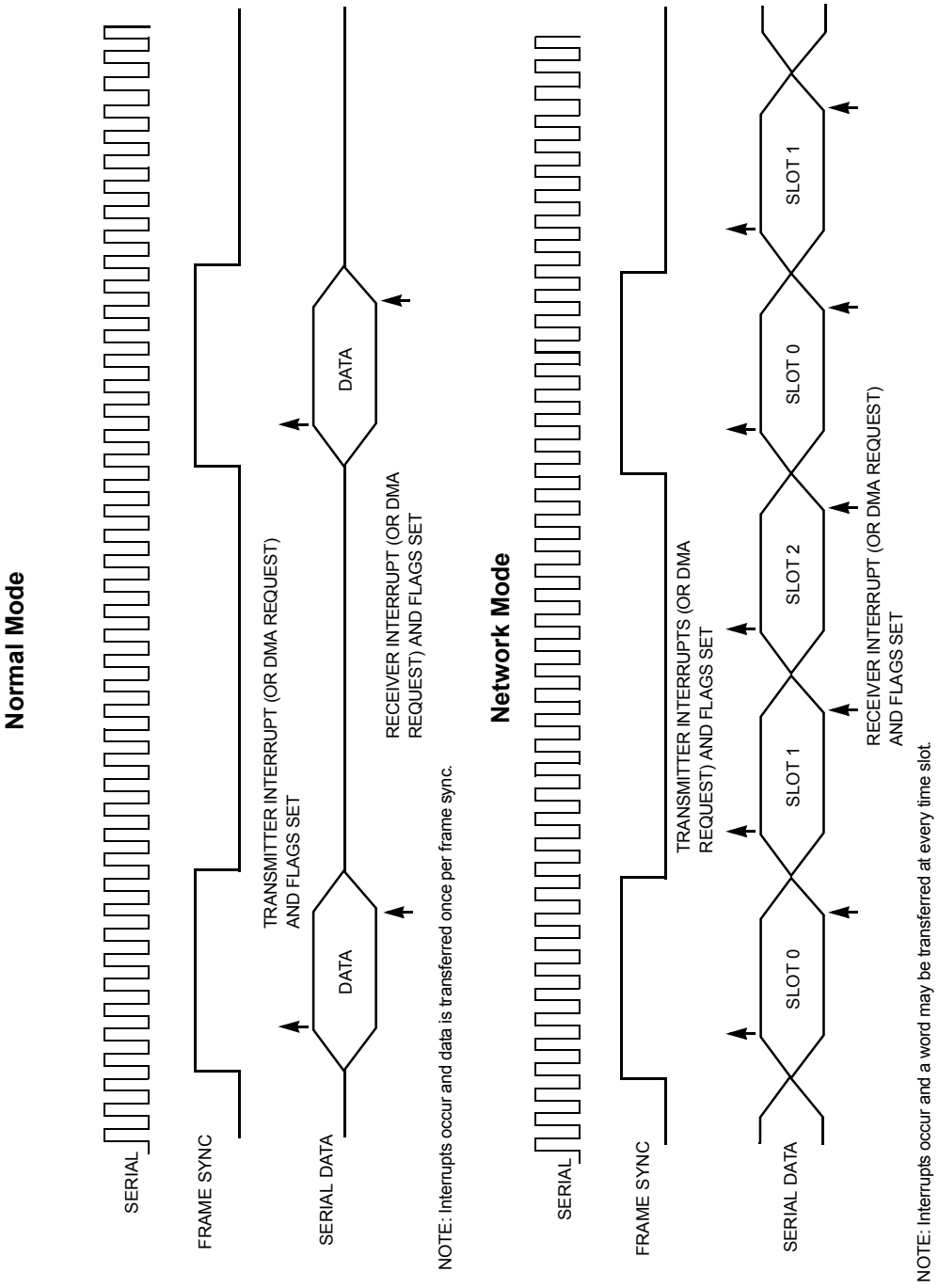


Figure 8-6. Normal and Network Operation

8.3.2.10 TCR Tx Slot and Word Length Select (TSWS4-TSWS0) - Bits 14-10

The TSWS4-TSWS0 bits are used to select the length of the slot and the length of the data words being transferred via the ESAI. The word length must be equal to or shorter than the slot length. The possible combinations are shown in Table 8-5. See also the ESAI data path programming model in Figure 8-13 and Figure 8-14.

Table 8-5. ESAI Transmit Slot and Word Length Selection

TSWS4	TSWS3	TSWS2	TSWS1	TSWS0	SLOT LENGTH	WORD LENGTH
0	0	0	0	0	8	8
0	0	1	0	0	12	8
0	0	0	0	1		12
0	1	0	0	0	16	8
0	0	1	0	1		12
0	0	0	1	0		16
0	1	1	0	0	20	8
0	1	0	0	1		12
0	0	1	1	0		16
0	0	0	1	1		20
1	0	0	0	0	24	8
0	1	1	0	1		12
0	1	0	1	0		16
0	0	1	1	1		20
1	1	1	1	0		24
1	1	0	0	0	32	8
1	0	1	0	1		12
1	0	0	1	0		16
0	1	1	1	1		20
1	1	1	1	1		24
0	1	0	1	1	Reserved	
0	1	1	1	0		
1	0	0	0	1		
1	0	0	1	1		
1	0	1	0	0		
1	0	1	1	0		
1	0	1	1	1		
1	1	0	0	1		
1	1	0	1	0		
1	1	0	1	1		
1	1	1	0	0		
1	1	1	0	1		

8.3.2.11 TCR Transmit Frame Sync Length (TFSL) - Bit 15

The TFSL bit selects the length of frame sync to be generated or recognized. If TFSL is cleared, a word-length frame sync is selected. If TFSL is set, a 1-bit clock period frame sync is selected. See Figure 8-7 for examples of frame length selection.

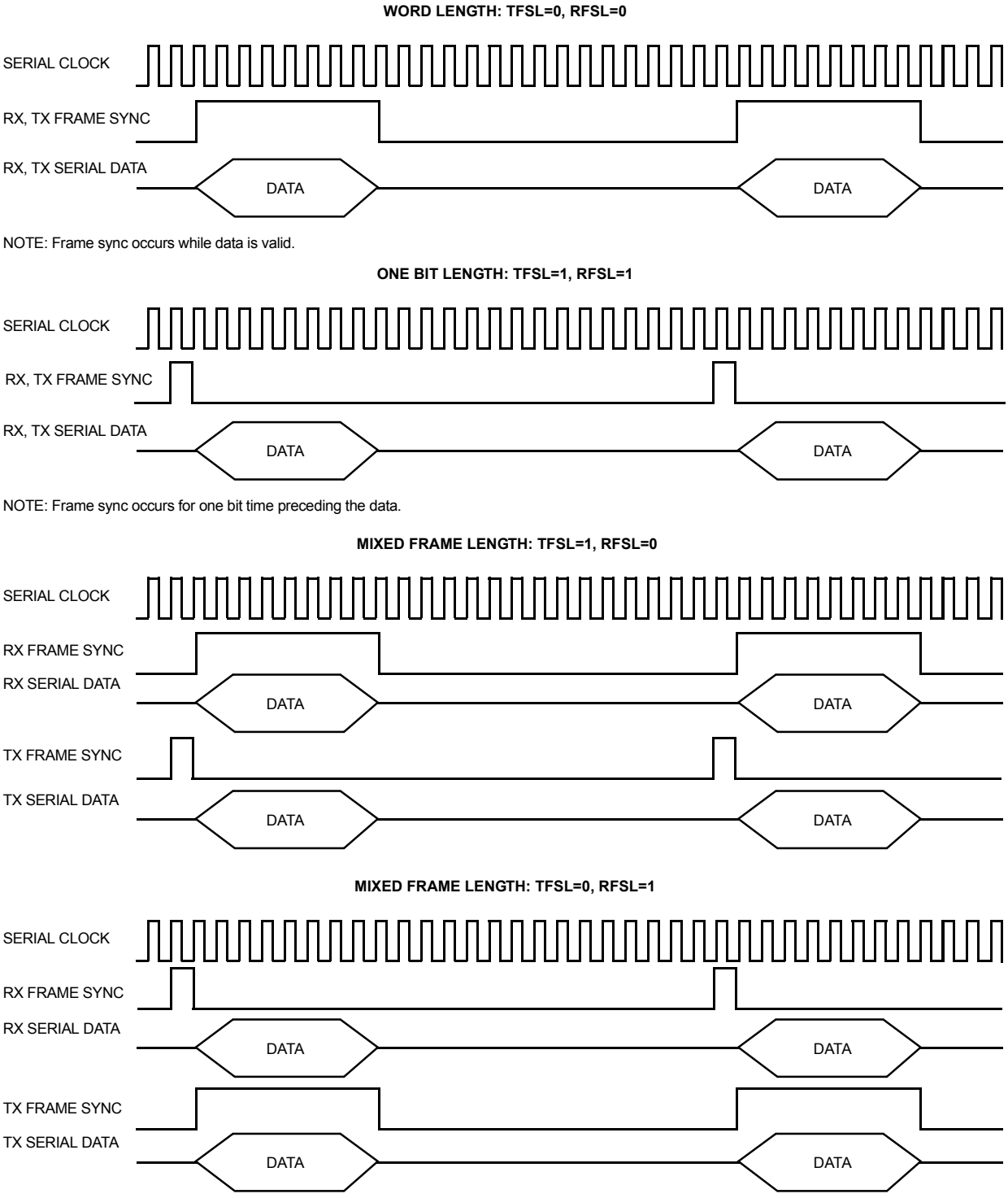


Figure 8-7. Frame Length Selection

8.3.2.12 TCR Transmit Frame Sync Relative Timing (TFSR) - Bit 16

TFSR determines the relative timing of the transmit frame sync signal as referred to the serial data lines, for a word length frame sync only (TFSL=0). When TFSR is cleared the word length frame sync occurs together with the first bit of the data word of the first slot. When TFSR is set the word length frame sync starts one serial clock cycle earlier, i.e., together with the last bit of the previous data word.

8.3.2.13 TCR Transmit Zero Padding Control (PADC) - Bit 17

When PADC is cleared, zero padding is disabled. When PADC is set, zero padding is enabled. PADC, in conjunction with the TWA control bit, determines the way that padding is done for operating modes where the word length is less than the slot length. See the TWA bit description in Section 8.3.2.8, *TCR Transmit Word Alignment Control (TWA) - Bit 7* for more details.

Since the data word is shorter than the slot length, the data word is extended until achieving the slot length, according to the following rule:

1. If the data word is left-aligned (TWA=0), and zero padding is disabled (PADC=0), the last data bit is repeated after the data word has been transmitted. If zero padding is enabled (PADC=1), zeroes are transmitted after the data word has been transmitted.
2. If the data word is right-aligned (TWA=1), and zero padding is disabled (PADC=0), the first data bit is repeated before the transmission of the data word. If zero padding is enabled (PADC=1), zeroes are transmitted before the transmission of the data word.

8.3.2.14 TCR Reserved Bit - Bits 18

This bit is reserved. It reads as zero, and it should be written with zero for future compatibility.

8.3.2.15 TCR Transmit Section Personal Reset (TPR) - Bit 19

The TPR control bit is used to put the transmitter section of the ESAI in the personal reset state. The receiver section is not affected. When TPR is cleared, the transmitter section may operate normally. When TPR is set, the transmitter section enters the personal reset state immediately. When in the personal reset state, the status bits are reset to the same state as after hardware reset. The control bits are not affected by the personal reset state. The transmitter data pins are tri-stated while in the personal reset state; if a stable logic level is desired, the transmitter data pins should be defined as GPIO outputs. The transmitter clock outputs drive zeroes while in the personal reset state. Note that to leave the personal reset state by clearing TPR, the procedure described in Section 8.6, *ESAI Initialization Examples* should be followed.

8.3.2.16 TCR Transmit Exception Interrupt Enable (TEIE) - Bit 20

When TEIE is set, the DSP is interrupted when both TDE and TUE in the SAISR status register are set. When TEIE is cleared, this interrupt is disabled. Reading the SAISR status register followed by writing to all the data registers of the enabled transmitters clears TUE, thus clearing the pending interrupt.

8.3.2.17 TCR Transmit Even Slot Data Interrupt Enable (TEDIE) - Bit 21

The TEDIE control bit is used to enable the transmit even slot data interrupts. If TEDIE is set, the transmit even slot data interrupts are enabled. If TEDIE is cleared, the transmit even slot data interrupts are disabled. A transmit even slot data interrupt request is generated if TEDIE is set and the TEDE status flag in the SAISR status register is set. Even time slots are all even-numbered time slots (0, 2, 4, etc.) when operating in network mode. The zero time slot in the frame is marked by the frame sync signal and is considered to be even. Writing data to all the data registers of the enabled transmitters or to TSR clears the TEDE flag, thus servicing the interrupt.

Transmit interrupts with exception have higher priority than transmit even slot data interrupts, therefore if exception occurs (TUE is set) and TEIE is set, the ESAI requests an ESAI transmit data with exception interrupt from the interrupt controller.

8.3.2.18 TCR Transmit Interrupt Enable (TIE) - Bit 22

The DSP is interrupted when TIE and the TDE flag in the SAISR status register are set. When TIE is cleared, this interrupt is disabled. Writing data to all the data registers of the enabled transmitters or to TSR clears TDE, thus clearing the interrupt.

Transmit interrupts with exception have higher priority than normal transmit data interrupts, therefore if exception occurs (TUE is set) and TEIE is set, the ESAI requests an ESAI transmit data with exception interrupt from the interrupt controller.

8.3.2.19 TCR Transmit Last Slot Interrupt Enable (TLIE) - Bit 23

TLIE enables an interrupt at the beginning of last slot of a frame in network mode. When TLIE is set the DSP is interrupted at the start of the last slot in a frame in network mode regardless of the transmit mask register setting. When TLIE is cleared the transmit last slot interrupt is disabled. TLIE is disabled when TDC[4:0]=$00000 (on-demand mode). The use of the transmit last slot interrupt is described in Section 8.4.3, *ESAI Interrupt Requests*.

8.3.3 ESAI Receive Clock Control Register (RCCR)

The read/write Receive Clock Control Register (RCCR) controls the ESAI receiver clock generator bit and frame sync rates, word length and number of words per frame for the serial data. The RCCR control bits are described in the following paragraphs (see Figure 8-8). The PDRC register provides additional clocking options by allowing the use of EXTAL as the clock source to the ESAI receiver as shown in Table 8-1. (Also see Figure 8-21).

11	10	9	8	7	6	5	4	3	2	1	0
RDC2	RDC1	RDC0	RPSR	RPM7	RPM6	RPM5	RPM4	RPM3	RPM2	RPM1	RPM0

23	22	21	20	19	18	17	16	15	14	13	12
RHCKD	RFSD	RCKD	RHCKP	RFSP	RCKP	RFP3	RFP2	RFP1	RFP0	RDC4	RDC3

Figure 8-8. RCCR Register

Hardware and software reset clear all the bits of the RCCR register. The ESAI RCCR register is located at x:$FFFFB8. The ESAI_1 RCCR register is located at y:$FFFF98.

8.3.3.1 RCCR Receiver Prescale Modulus Select (RPM7–RPM0) - Bits 7–0

The RPM7–RPM0 bits specify the divide ratio of the prescale divider in the ESAI receiver clock generator. A divide ratio from 1 to 256 (RPM[7:0]=$00 to $FF) may be selected. The bit clock output is available at the receiver serial bit clock (SCKR) pin of the DSP. The bit clock output is also available internally for use as the bit clock to shift the receive shift registers. The ESAI receive clock generator functional diagram is shown in Figure 8-3.

8.3.3.2 RCCR Receiver Prescaler Range (RPSR) - Bit 8

The RPSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When RPSR is set, the fixed prescaler is bypassed. When RPSR is cleared, the fixed divide-by-eight prescaler is operational (see Figure 8-3). The maximum internally generated bit clock frequency is Fosc/4, the minimum internally generated bit clock frequency is Fosc/(2 x 8 x 256)=Fosc/4096.

NOTE

Do not use the combination RPSR=1 and RPM7-RPM0=$00, which causes synchronization problems when using the internal DSP clock as source (RHCKD=1 or RCKD=1).

8.3.3.3 RCCR Rx Frame Rate Divider Control (RDC4–RDC0) - Bits 13–9

The RDC4–RDC0 bits control the divide ratio for the programmable frame rate dividers used to generate the receiver frame clocks.

In network mode, this ratio may be interpreted as the number of words per frame minus one. The divide ratio may range from 2 to 32 (RDC[4:0]=00001 to 11111) for network mode. A divide ratio of one (RDC[4:0]=00000) in network mode is a special case (on-demand mode).

In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (RDC[4:0]=00000 to 11111) for normal mode. In normal mode, a divide ratio of one (RDC[4:0]=00000) provides continuous periodic data word transfers. A bit-length frame sync (RFSL=1) must be used in this case.

The ESAI frame sync generator functional diagram is shown in Figure 8-4.

8.3.3.4 RCCR Rx High Frequency Clock Divider (RFP3-RFP0) - Bits 17-14

The RFP3–RFP0 bits control the divide ratio of the receiver high frequency clock to the receiver serial bit clock when the source of the receiver high frequency clock and the bit clock is the internal DSP clock. When the HCKR input is being driven from an external high frequency clock, the RFP3-RFP0 bits specify an additional division ration in the clock divider chain. See Table 8-6 for the specification of the divide ratio. The ESAI high frequency generator functional diagram is shown in Figure 8-3.

Table 8-6. Receiver High Frequency Clock Divider

RFP3-RFP0	Divide Ratio
$0	1
$1	2
$2	3
$3	4
...	...
$F	16

8.3.3.5 RCCR Receiver Clock Polarity (RCKP) - Bit 18

The Receiver Clock Polarity (RCKP) bit controls on which receive bit clock edge the receive data lines are latched in, and the receive frame sync is either clocked out if defined as an output or latched in if defined as an input.

If the RCKP bit is cleared, the receive data lines are latched in on the falling edge of the receive bit clock. The receive frame sync is clocked out on the rising edge of the receive bit clock if it is defined as an output, or it is latched in on the falling edge of the receive bit clock if defined as an input.

If the RCKP bit is set, the receive data lines are latched in on the rising edge of the receive bit clock. The receive frame sync is clocked out on the falling edge of the receive clock if it is defined as an output, or it is latched in on the rising edge of the receive bit clock if defined as an input.

8.3.3.6 RCCR Receiver Frame Sync Polarity (RFSP) - Bit 19

The Receiver Frame Sync Polarity (RFSP) determines the polarity of the receive frame sync signal. When RFSP is cleared the frame sync signal polarity is positive, i.e., the frame start is indicated by a high level on the frame sync pin. When RFSP is set the frame sync signal polarity is negative, i.e., the frame start is indicated by a low level on the frame sync pin.

8.3.3.7 RCCR Receiver High Frequency Clock Polarity (RHCKP) - Bit 20

The Receiver High Frequency Clock Polarity (RHCKP) bit controls on which bit clock edge data and frame sync are clocked out and latched in. If RHCKP is cleared the data and the frame sync are clocked out on the rising edge of the receive high frequency bit clock and the frame sync is latched in on the falling edge of the receive bit clock. If RHCKP is set the falling edge of the receive clock is used to clock the data and frame sync out and the rising edge of the receive clock is used to latch the frame sync in.

8.3.3.8 RCCR Receiver Clock Source Direction (RCKD) - Bit 21

The Receiver Clock Source Direction (RCKD) bit selects the source of the clock signal used to clock the receive shift register in the asynchronous mode (SYN=0) and the IF0/OF0 flag direction in the synchronous mode (SYN=1).

In the asynchronous mode, when RCKD is set, the internal clock source becomes the bit clock for the receive shift registers and word length divider and is the output on the SCKR pin. In the asynchronous mode when RCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCKR pin, and an external clock source may drive this pin.

In the synchronous mode when RCKD is set, the SCKR pin becomes the OF0 output flag. If RCKD is cleared, the SCKR pin becomes the IF0 input flag. See Figure 8-1 and Figure 8-7.

Table 8-7. SCKR Pin Definition Table

Control Bits		SCKR PIN
SYN	RCKD	
0	0	SCKR input
0	1	SCKR output
1	0	IF0
1	1	OF0

8.3.3.9 RCCR Receiver Frame Sync Signal Direction (RFSD) - Bit 22

The Receiver Frame Sync Signal Direction (RFSD) bit selects the source of the receiver frame sync signal when in the asynchronous mode (SYN=0) and the IF1/OF1/Transmitter Buffer Enable flag direction in the synchronous mode (SYN=1).

In the asynchronous mode, when RFSD is set, the internal clock generator becomes the source of the receiver frame sync and is the output on the FSR pin. In the asynchronous mode, when RFSD is cleared, the receiver frame sync source is external; the internal clock generator is disconnected from the FSR pin, and an external clock source may drive this pin.

In the synchronous mode when RFSD is set, the FSR pin becomes the OF1 output flag or the Transmitter Buffer Enable, according to the TEBE control bit. If RFSD is cleared, the FSR pin becomes the IF1 input flag. See Figure 8-1 and Figure 8-8.

Table 8-8. FSR Pin Definition Table

Control Bits			FSR Pin
SYN	TEBE	RFSD	
0	X	0	FSR input
0	X	1	FSR output
1	0	0	IF1
1	0	1	OF1
1	1	0	reserved
1	1	1	Transmitter Buffer Enable

8.3.3.10 RCCR Receiver High Frequency Clock Direction (RHCKD) - Bit 23

The Receiver High Frequency Clock Direction (RHCKD) bit selects the source of the receiver high frequency clock when in the asynchronous mode (SYN=0) and the IF2/OF2 flag direction in the synchronous mode (SYN=1).

In the asynchronous mode, when RHCKD is set, the internal clock generator becomes the source of the receiver high frequency clock and is the output on the HCKR pin. In the asynchronous mode, when RHCKD is cleared, the receiver high frequency clock source is external; the internal clock generator is disconnected from the HCKR pin, and an external clock source may drive this pin.

When RHCKD is cleared, HCKR is an input; when RHCKD is set, HCKR is an output.

In the synchronous mode when RHCKD is set, the HCKR pin becomes the OF2 output flag. If RHCKD is cleared, the HCKR pin becomes the IF2 input flag. See Figure 8-1 and Figure 8-9.

Table 8-9. HCKR Pin Definition Table

Control Bits		HCKR PIN
SYN	RHCKD	
0	0	HCKR input
0	1	HCKR output
1	0	IF2
1	1	OF2

8.3.4 ESAI Receive Control Register (RCR)

The read/write Receive Control Register (RCR) controls the ESAI receiver section. Interrupt enable bits for the receivers are provided in this control register. The receivers are enabled in this register (0,1,2 or 3 receivers can be enabled) if the input data pin is not used by a transmitter. Operating modes are also selected in this register.

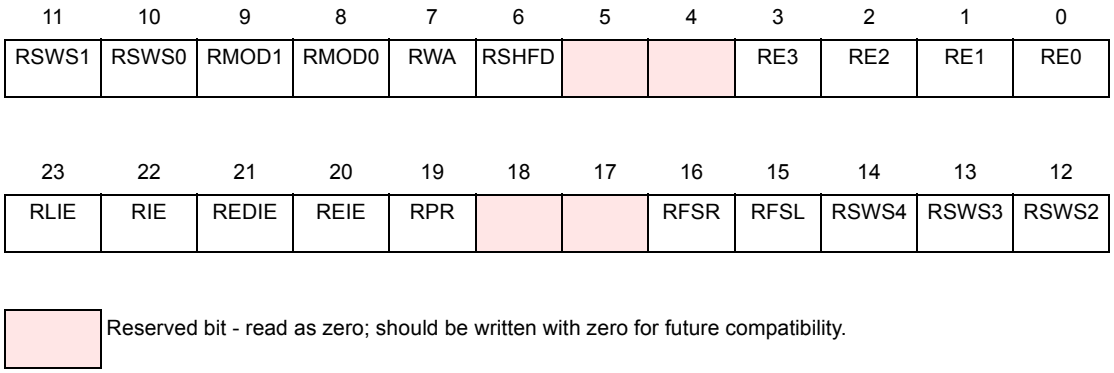


Figure 8-9. RCR Register

Hardware and software reset clear all the bits in the RCR register. The ESAI RCR register is located at x:$FFFFB7. The ESAI_1 RCR register is located at y:$FFFF97.

The ESAI RCR bits are described in the following paragraphs.

8.3.4.1 RCR ESAI Receiver 0 Enable (RE0) - Bit 0

When RE0 is set and TE5 is cleared, the ESAI receiver 0 is enabled and samples data at the SDO5/SDI0 pin. TX5 and RX0 should not be enabled at the same time (RE0=1 and TE5=1). When RE0 is cleared, receiver 0 is disabled by inhibiting data transfer into RX0. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX0 data register.

If RE0 is set while some of the other receivers are already in operation, the first data word received in RX0 will be invalid and must be discarded.

8.3.4.2 RCR ESAI Receiver 1 Enable (RE1) - Bit 1

When RE1 is set and TE4 is cleared, the ESAI receiver 1 is enabled and samples data at the SDO4/SDI1 pin. TX4 and RX1 should not be enabled at the same time (RE1=1 and TE4=1). When RE1 is cleared, receiver 1 is disabled by inhibiting data transfer into RX1. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX1 data register.

If RE1 is set while some of the other receivers are already in operation, the first data word received in RX1 will be invalid and must be discarded.

8.3.4.3 RCR ESAI Receiver 2 Enable (RE2) - Bit 2

When RE2 is set and TE3 is cleared, the ESAI receiver 2 is enabled and samples data at the SDO3/SDI2 pin. TX3 and RX2 should not be enabled at the same time (RE2=1 and TE3=1). When RE2 is cleared, receiver 2 is disabled by inhibiting data transfer into RX2. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX2 data register.

If RE2 is set while some of the other receivers are already in operation, the first data word received in RX2 will be invalid and must be discarded.

8.3.4.4 RCR ESAI Receiver 3 Enable (RE3) - Bit 3

When RE3 is set and TE2 is cleared, the ESAI receiver 3 is enabled and samples data at the SDO2/SDI3 pin. TX2 and RX3 should not be enabled at the same time (RE3=1 and TE2=1). When RE3 is cleared, receiver 3 is disabled by inhibiting data transfer into RX3. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX3 data register.

If RE3 is set while some of the other receivers are already in operation, the first data word received in RX3 will be invalid and must be discarded.

8.3.4.5 RCR Reserved Bits - Bits 5-4, 18-17

These bits are reserved. They read as zero, and they should be written with zero for future compatibility.

8.3.4.6 RCR Receiver Shift Direction (RSHFD) - Bit 6

The RSHFD bit causes the receiver shift registers to shift data in MSB first when RSHFD is cleared or LSB first when RSHFD is set (see Figure 8-13 and Figure 8-14).

8.3.4.7 RCR Receiver Word Alignment Control (RWA) - Bit 7

The Receiver Word Alignment Control (RWA) bit defines the alignment of the data word in relation to the slot. This is relevant for the cases where the word length is shorter than the slot length. If RWA is cleared, the data word is assumed to be left-aligned in the slot frame. If RWA is set, the data word is assumed to be right-aligned in the slot frame.

If the data word is shorter than the slot length, the data bits which are not in the data word field are ignored.

For data word lengths of less than 24 bits, the data word is right-extended with zeroes before being stored in the receive data registers.

8.3.4.8 RCR Receiver Network Mode Control (RMOD1-RMOD0) - Bits 9-8

The RMOD1 and RMOD0 bits are used to define the network mode of the ESAI receivers according to Table 8-10. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot, as shown in Figure 8-6. In network mode, it is possible to transfer a word for every time slot, as shown in Figure 8-6. For more details, see Section 8.4, *Operating Modes*.

In order to comply with AC-97 specifications, RSWS4-RSWS0 should be set to 00011 (20-bit slot, 20-bit word); RFSL and RFSR should be cleared, and RDC4-RDC0 should be set to $0C (13 words in frame).

Table 8-10. ESAI Receive Network Mode Selection

RMOD1	RMOD0	RDC4-RDC0	Receiver Network Mode
0	0	$0-$1F	Normal Mode
0	1	$0	On-Demand Mode
0	1	$1-$1F	Network Mode
1	0	X	Reserved
1	1	$0C	AC97

8.3.4.9 RCR Receiver Slot and Word Select (RSWS4-RSWS0) - Bits 14-10

The RSWS4-RSWS0 bits are used to select the length of the slot and the length of the data words being received via the ESAI. The word length must be equal to or shorter than the slot length. The possible combinations are shown in Table 8-11. See also the ESAI data path programming model in Figure 8-13 and Figure 8-14.

Table 8-11. ESAI Receive Slot and Word Length Selection

RSWS4	RSWS3	RSWS2	RSWS1	RSWS0	SLOT LENGTH	WORD LENGTH
0	0	0	0	0	8	8
0	0	1	0	0	12	8
0	0	0	0	1		12
0	1	0	0	0	16	8
0	0	1	0	1		12
0	0	0	1	0		16
0	1	1	0	0	20	8
0	1	0	0	1		12
0	0	1	1	0		16
0	0	0	1	1		20

Table 8-11. ESAI Receive Slot and Word Length Selection (continued)

RSWS4	RSWS3	RSWS2	RSWS1	RSWS0	SLOT LENGTH	WORD LENGTH
1	0	0	0	0	24	8
0	1	1	0	1		12
0	1	0	1	0		16
0	0	1	1	1		20
1	1	1	1	0		24
1	1	0	0	0	32	8
1	0	1	0	1		12
1	0	0	1	0		16
0	1	1	1	1		20
1	1	1	1	1		24
0	1	0	1	1	Reserved	
0	1	1	1	0		
1	0	0	0	1		
1	0	0	1	1		
1	0	1	0	0		
1	0	1	1	0		
1	0	1	1	1		
1	1	0	0	1		
1	1	0	1	0		
1	1	0	1	1		
1	1	1	0	0		
1	1	1	0	1		

8.3.4.10 RCR Receiver Frame Sync Length (RFSL) - Bit 15

The RFSL bit selects the length of the receive frame sync to be generated or recognized. If RFSL is cleared, a word-length frame sync is selected. If RFSL is set, a 1-bit clock period frame sync is selected. See Figure 8-7 for examples of frame length selection.

8.3.4.11 RCR Receiver Frame Sync Relative Timing (RFSR) - Bit 16

RFSR determines the relative timing of the receive frame sync signal as referred to the serial data lines, for a word length frame sync only. When RFSR is cleared the word length frame sync occurs together with the first bit of the data word of the first slot. When RFSR is set the word length frame sync starts one serial clock cycle earlier, i.e., together with the last bit of the previous data word.

8.3.4.12 RCR Receiver Section Personal Reset (RPR) - Bit 19

The RPR control bit is used to put the receiver section of the ESAI in the personal reset state. The transmitter section is not affected. When RPR is cleared, the receiver section may operate normally. When RPR is set, the receiver section enters the personal reset state immediately. When in the personal reset state, the status bits are reset to the same state as after hardware reset.The control bits are not affected by the personal reset state.The receiver data pins are disconnected while in the personal reset state. Note that to leave the personal reset state by clearing RPR, the procedure described in Section 8.6, *ESAI Initialization Examples* should be followed.

8.3.4.13 RCR Receive Exception Interrupt Enable (REIE) - Bit 20

When REIE is set, the DSP is interrupted when both RDF and ROE in the SAISR status register are set. When REIE is cleared, this interrupt is disabled. Reading the SAISR status register followed by reading the enabled receivers data registers clears ROE, thus clearing the pending interrupt.

8.3.4.14 RCR Receive Even Slot Data Interrupt Enable (REDIE) - Bit 21

The REDIE control bit is used to enable the receive even slot data interrupts. If REDIE is set, the receive even slot data interrupts are enabled. If REDIE is cleared, the receive even slot data interrupts are disabled. A receive even slot data interrupt request is generated if REDIE is set and the REDF status flag in the SAISR status register is set. Even time slots are all even-numbered time slots (0, 2, 4, etc.) when operating in network mode. The zero time slot is marked by the frame sync signal and is considered to be even. Reading all the data registers of the enabled receivers clears the REDF flag, thus servicing the interrupt.

Receive interrupts with exception have higher priority than receive even slot data interrupts, therefore if exception occurs (ROE is set) and REIE is set, the ESAI requests an ESAI receive data with exception interrupt from the interrupt controller.

8.3.4.15 RCR Receive Interrupt Enable (RIE) - Bit 22

The DSP is interrupted when RIE and the RDF flag in the SAISR status register are set. When RIE is cleared, this interrupt is disabled. Reading the receive data registers of the enabled receivers clears RDF, thus clearing the interrupt.

Receive interrupts with exception have higher priority than normal receive data interrupts, therefore if exception occurs (ROE is set) and REIE is set, the ESAI requests an ESAI receive data with exception interrupt from the interrupt controller.

8.3.4.16 RCR Receive Last Slot Interrupt Enable (RLIE) - Bit 23

RLIE enables an interrupt after the last slot of a frame ended in network mode only. When RLIE is set the DSP is interrupted after the last slot in a frame ended regardless of the receive mask register setting. When RLIE is cleared the receive last slot interrupt is disabled. Hardware and software reset clear RLIE. RLIE is disabled when RDC[4:0]=00000 (on-demand mode). The use of the receive last slot interrupt is described in Section 8.4.3, *ESAI Interrupt Requests*.

8.3.5 ESAI Common Control Register (SAICR)

The read/write Common Control Register (SAICR) contains control bits for functions that affect both the receive and transmit sections of the ESAI. See Figure 8-10.

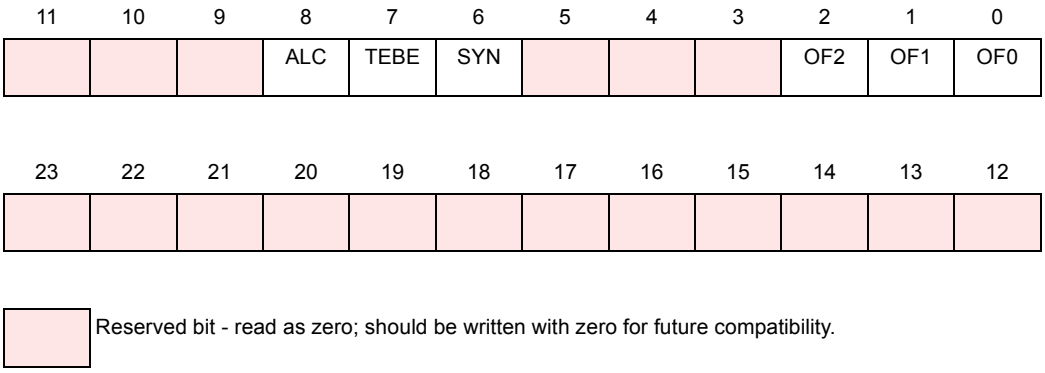


Figure 8-10. SAICR Register

Hardware and software reset clear all the bits in the SAICR register. The ESAI SAICR register is located at x:$FFFFB4. The ESAI_1 SAICR register is located at y:$FFFF94.

8.3.5.1 SAICR Serial Output Flag 0 (OF0) - Bit 0

The Serial Output Flag 0 (OF0) is a data bit used to hold data to be send to the OF0 pin. When the ESAI is in the synchronous clock mode (SYN=1), the SCKR pin is configured as the ESAI flag 0. If the receiver serial clock direction bit (RCKD) is set, the SCKR pin is the output flag OF0, and data present in the OF0 bit is written to the OF0 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.

8.3.5.2 SAICR Serial Output Flag 1 (OF1) - Bit 1

The Serial Output Flag 1 (OF1) is a data bit used to hold data to be send to the OF1 pin. When the ESAI is in the synchronous clock mode (SYN=1), the FSR pin is configured as the ESAI flag 1. If the receiver frame sync direction bit (RFSD) is set and the TEBE bit is cleared, the FSR pin is the output flag OF1, and data present in the OF1 bit is written to the OF1 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.

8.3.5.3 SAICR Serial Output Flag 2 (OF2) - Bit 2

The Serial Output Flag 2 (OF2) is a data bit used to hold data to be send to the OF2 pin. When the ESAI is in the synchronous clock mode (SYN=1), the HCKR pin is configured as the ESAI flag 2. If the receiver high frequency clock direction bit (RHCKD) is set, the HCKR pin is the output flag OF2, and data present in the OF2 bit is written to the OF2 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.

8.3.5.4 SAICR Reserved Bits - Bits 5-3, 23-9

These bits are reserved. They read as zero, and they should be written with zero for future compatibility.

8.3.5.5 SAICR Synchronous Mode Selection (SYN) - Bit 6

The Synchronous Mode Selection (SYN) bit controls whether the receiver and transmitter sections of the ESAI operate synchronously or asynchronously with respect to each other (see Table 8-11). When SYN is cleared, the asynchronous mode is chosen and independent clock and frame sync signals are used for the transmit and receive sections. When SYN is set, the synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals.

When in the synchronous mode (SYN=1), the transmit and receive sections use the transmitter section clock generator as the source of the clock and frame sync for both sections. Also, the receiver clock pins SCKR, FSR and HCKR now operate as I/O flags. See Table 8-7, Table 8-8, and Table 8-9 for the effects of SYN on the receiver clock pins.

8.3.5.6 SAICR Transmit External Buffer Enable (TEBE) - Bit 7

The Transmitter External Buffer Enable (TEBE) bit controls the function of the FSR pin when in the synchronous mode. If the ESAI is configured for operation in the synchronous mode (SYN=1), and TEBE is set while FSR pin is configured as an output (RFSD=1), the FSR pin functions as the transmitter external buffer enable control to enable the use of an external buffers on the transmitter outputs. If TEBE is cleared, the FSR pin functions as the serial I/O flag 1. See Table 8-8 for a summary of the effects of TEBE on the FSR pin.

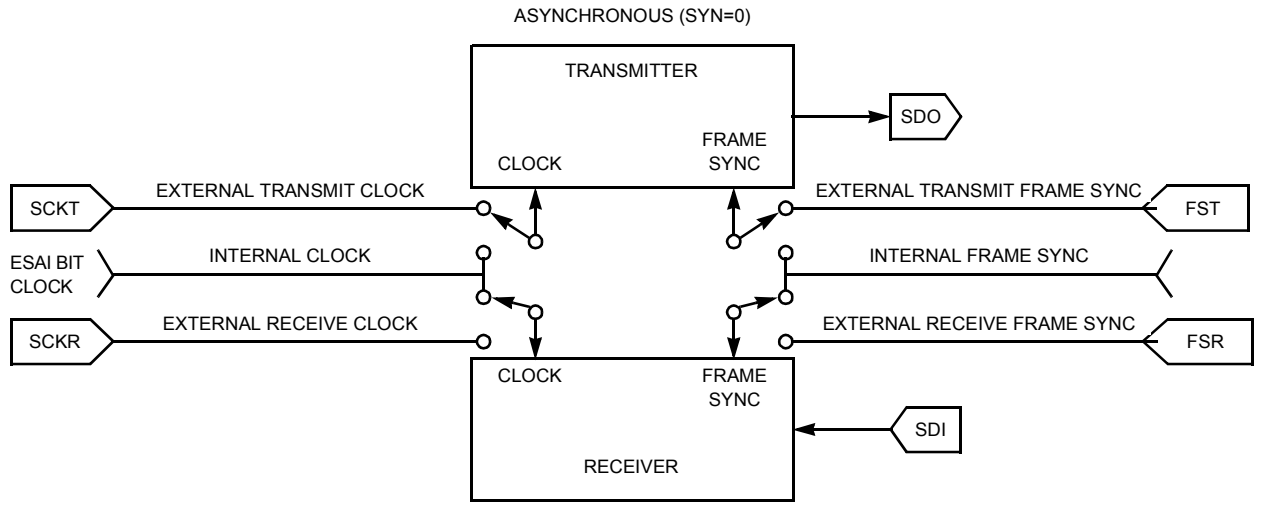
8.3.5.7 SAICR Alignment Control (ALC) - Bit 8

The ESAI is designed for 24-bit fractional data, thus shorter data words are left aligned to the MSB (bit 23). Some applications use 16-bit fractional data. In those cases, shorter data words may be left aligned to bit 15. The Alignment Control (ALC) bit supports these applications.

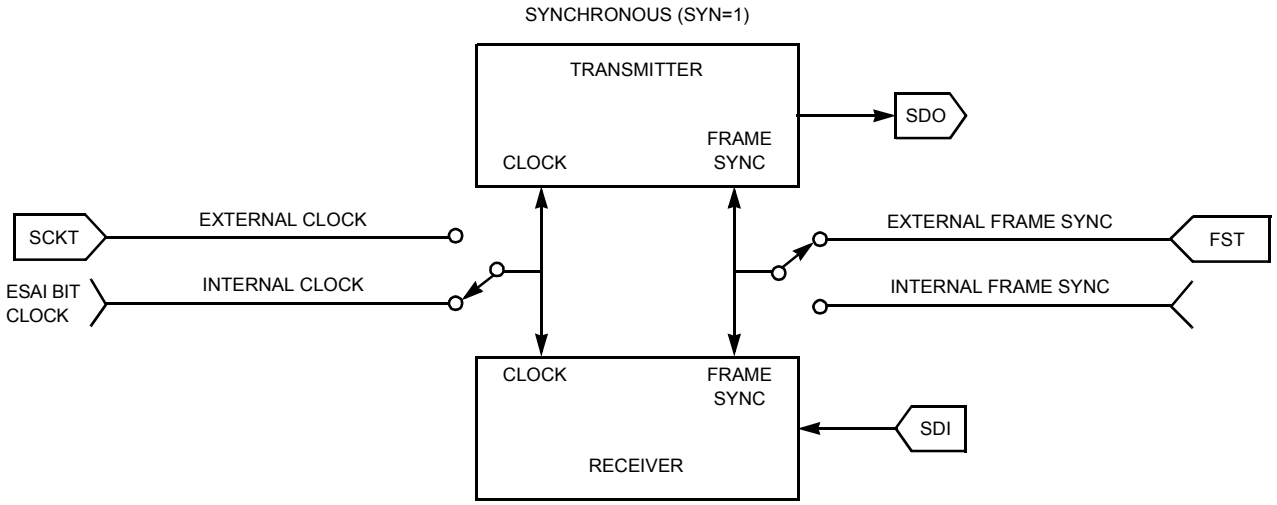
If ALC is set, transmitted and received words are left aligned to bit 15 in the transmit and receive shift registers. If ALC is cleared, transmitted and received word are left aligned to bit 23 in the transmit and receive shift registers.

NOTE

While ALC is set, 20-bit and 24-bit words may not be used, and word length control should specify 8-, 12- or 16-bit words; otherwise, results are unpredictable.



NOTE: Transmitter and receiver may have different clocks and frame syncs.



NOTE: Transmitter and receiver have the same clocks and frame syncs.

Figure 8-11. SAICR SYN Bit Operation

8.3.6 ESAI Status Register (SAISR)

The Status Register (SAISR) is a read-only status register used by the DSP to read the status and serial input flags of the ESAI. See Figure 8-12. The status bits are described in the following paragraphs.

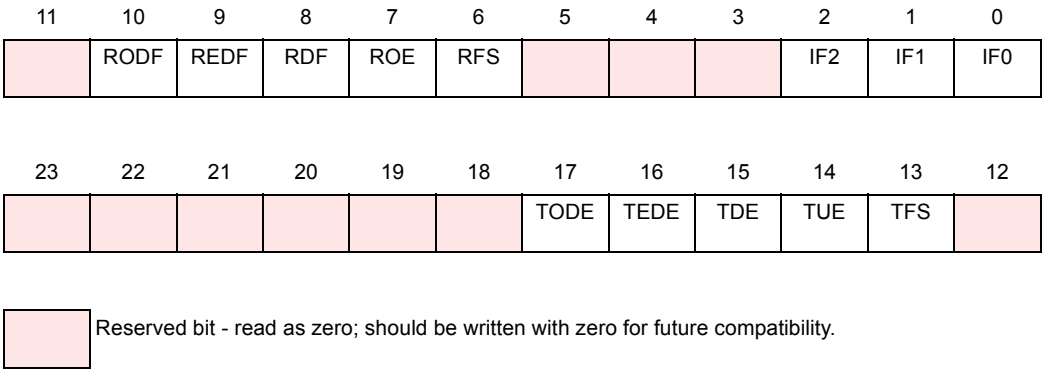


Figure 8-12. SAISR Register

The ESAI SAISR register is located at x:$FFFFB3. The ESAI_1 SAISR register is located at y:$FFFF93.

8.3.6.1 SAISR Serial Input Flag 0 (IF0) - Bit 0

The IF0 bit is enabled only when the SCKR pin is defined as ESAI in the Port Control Register, SYN=1 and RCKD=0, indicating that SCKR is an input flag and the synchronous mode is selected. Data present on the SCKR pin is latched during reception of the first received data bit after frame sync is detected. The IF0 bit is updated with this data when the receiver shift registers are transferred into the receiver data registers. IF0 reads as a zero when it is not enabled. Hardware, software, ESAI individual and STOP reset clear IF0.

8.3.6.2 SAISR Serial Input Flag 1 (IF1) - Bit 1

The IF1 bit is enabled only when the FSR pin is defined as ESAI in the Port Control Register, SYN =1, RFSD=0 and TEBE=0, indicating that FSR is an input flag and the synchronous mode is selected. Data present on the FSR pin is latched during reception of the first received data bit after frame sync is detected. The IF1 bit is updated with this data when the receiver shift registers are transferred into the receiver data registers. IF1 reads as a zero when it is not enabled. Hardware, software, ESAI individual and STOP reset clear IF1.

8.3.6.3 SAISR Serial Input Flag 2 (IF2) - Bit 2

The IF2 bit is enabled only when the HCKR pin is defined as ESAI in the Port Control Register, SYN=1 and RHCKD=0, indicating that HCKR is an input flag and the synchronous mode is selected. Data present on the HCKR pin is latched during reception of the first received data bit after frame sync is detected. The IF2 bit is updated with this data when the receive shift registers are transferred into the receiver data registers. IF2 reads as a zero when it is not enabled. Hardware, software, ESAI individual and STOP reset clear IF2.

8.3.6.4 SAISR Reserved Bits - Bits 5-3, 12-11, 23-18

These bits are reserved for future use. They read as zero.

8.3.6.5 SAISR Receive Frame Sync Flag (RFS) - Bit 6

When set, RFS indicates that a receive frame sync occurred during reception of the words in the receiver data registers. This indicates that the data words are from the first slot in the frame. When RFS is clear and a word is received, it indicates (only in the network mode) that the frame sync did not occur during reception of that word. RFS is cleared by hardware, software, ESAI individual, or STOP reset. RFS is valid only if at least one of the receivers is enabled (REx=1).

NOTE

In normal mode, RFS always reads as a one when reading data because there is only one time slot per frame – the “frame sync” time slot.

8.3.6.6 SAISR Receiver Overrun Error Flag (ROE) - Bit 7

The ROE flag is set when the serial receive shift register of an enabled receiver is full and ready to transfer to its receiver data register (RXx) and the register is already full (RDF=1). If REIE is set, an ESAI receive data with exception (overrun error) interrupt request is issued when ROE is set. Hardware, software, ESAI individual and STOP reset clear ROE. ROE is also cleared by reading the SAISR with ROE set, followed by reading all the enabled receive data registers.

8.3.6.7 SAISR Receive Data Register Full (RDF) - Bit 8

RDF is set when the contents of the receive shift register of an enabled receiver is transferred to the respective receive data register. RDF is cleared when the DSP reads the receive data register of all enabled receivers or cleared by hardware, software, ESAI individual, or STOP reset. If RIE is set, an ESAI receive data interrupt request is issued when RDF is set.

8.3.6.8 SAISR Receive Even-Data Register Full (REDF) - Bit 9

When set, REDF indicates that the received data in the receive data registers of the enabled receivers have arrived during an even time slot when operating in the network mode. Even time slots are all even-numbered slots (0, 2, 4, 6, etc.). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. The zero time slot is considered even. REDF is set when the contents of the receive shift registers are transferred to the receive data registers. REDF is cleared when the DSP reads all the enabled receive data registers or cleared by hardware, software, ESAI individual, or STOP resets. If REDIE is set, an ESAI receive even slot data interrupt request is issued when REDF is set.

8.3.6.9 SAISR Receive Odd-Data Register Full (RODF) - Bit 10

When set, RODF indicates that the received data in the receive data registers of the enabled receivers have arrived during an odd time slot when operating in the network mode. Odd time slots are all odd-numbered slots (1, 3, 5, etc.). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. RODF is set when the contents of the receive shift registers are transferred to the receive data registers. RODF is cleared when the DSP reads all the enabled receive data registers or cleared by hardware, software, ESAI individual, or STOP resets.

8.3.6.10 SAISR Transmit Frame Sync Flag (TFS) - Bit 13

When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. Data written to a transmit data register during the time slot when TFS is set is transmitted (in network mode), if the transmitter is enabled, during the second time slot in the frame. TFS is useful in network mode to identify the start of a frame. TFS is cleared by hardware, software, ESAI individual, or STOP reset. TFS is valid only if at least one transmitter is enabled, i.e., one or more of TE0, TE1, TE2, TE3, TE4 and TE5 are set.

NOTE

In normal mode, TFS always reads as a one when transmitting data because there is only one time slot per frame – the “frame sync” time slot.

8.3.6.11 SAISR Transmit Underrun Error Flag (TUE) - Bit 14

TUE is set when at least one of the enabled serial transmit shift registers is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TX registers that were not written) is retransmitted. If TEIE is set, an ESAI transmit data with exception (underrun error) interrupt request is issued when TUE is set. Hardware, software, ESAI individual and STOP reset clear TUE. TUE is also cleared by reading the SAISR with TUE set, followed by writing to all the enabled transmit data registers or to TSR.

8.3.6.12 SAISR Transmit Data Register Empty (TDE) - Bit 15

TDE is set when the contents of the transmit data register of all the enabled transmitters are transferred to the transmit shift registers; it is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, TDE indicates that data should be written to all the TX registers of the enabled transmitters or to the time slot register (TSR). TDE is cleared when the DSP writes to all the transmit data registers of the enabled transmitters, or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, an ESAI transmit data interrupt request is issued when TDE is set. Hardware, software, ESAI individual and STOP reset clear TDE.

8.3.6.13 SAISR Transmit Even-Data Register Empty (TEDE) - Bit 16

When set, TEDE indicates that the enabled transmitter data registers became empty at the beginning of an even time slot. Even time slots are all even-numbered slots (0, 2, 4, 6, etc.). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. The zero time slot is considered even. This flag is set when the contents of the transmit data register of all the enabled transmitters are transferred to the transmit shift registers; it is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, TEDE indicates that data should be written to all the TX registers of the enabled transmitters or to the time slot register (TSR). TEDE is cleared when the DSP writes to all the transmit data registers of the enabled transmitters, or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, an ESAI transmit data interrupt request is issued when TEDE is set. Hardware, software, ESAI individual and STOP reset clear TEDE.

8.3.6.14 SAISR Transmit Odd-Data Register Empty (TODE) - Bit 17

When set, TODE indicates that the enabled transmitter data registers became empty at the beginning of an odd time slot. Odd time slots are all odd-numbered slots (1, 3, 5, etc.). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. This flag is set when the contents of the transmit data register of all the enabled transmitters are transferred to the transmit shift registers; it is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, TODE indicates that data should be written to all the TX registers of the enabled transmitters or to the time slot register (TSR). TODE is cleared when the DSP writes to all the transmit data registers of the enabled transmitters, or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, an ESAI transmit data interrupt request is issued when TODE is set. Hardware, software, ESAI individual and STOP reset clear TODE.

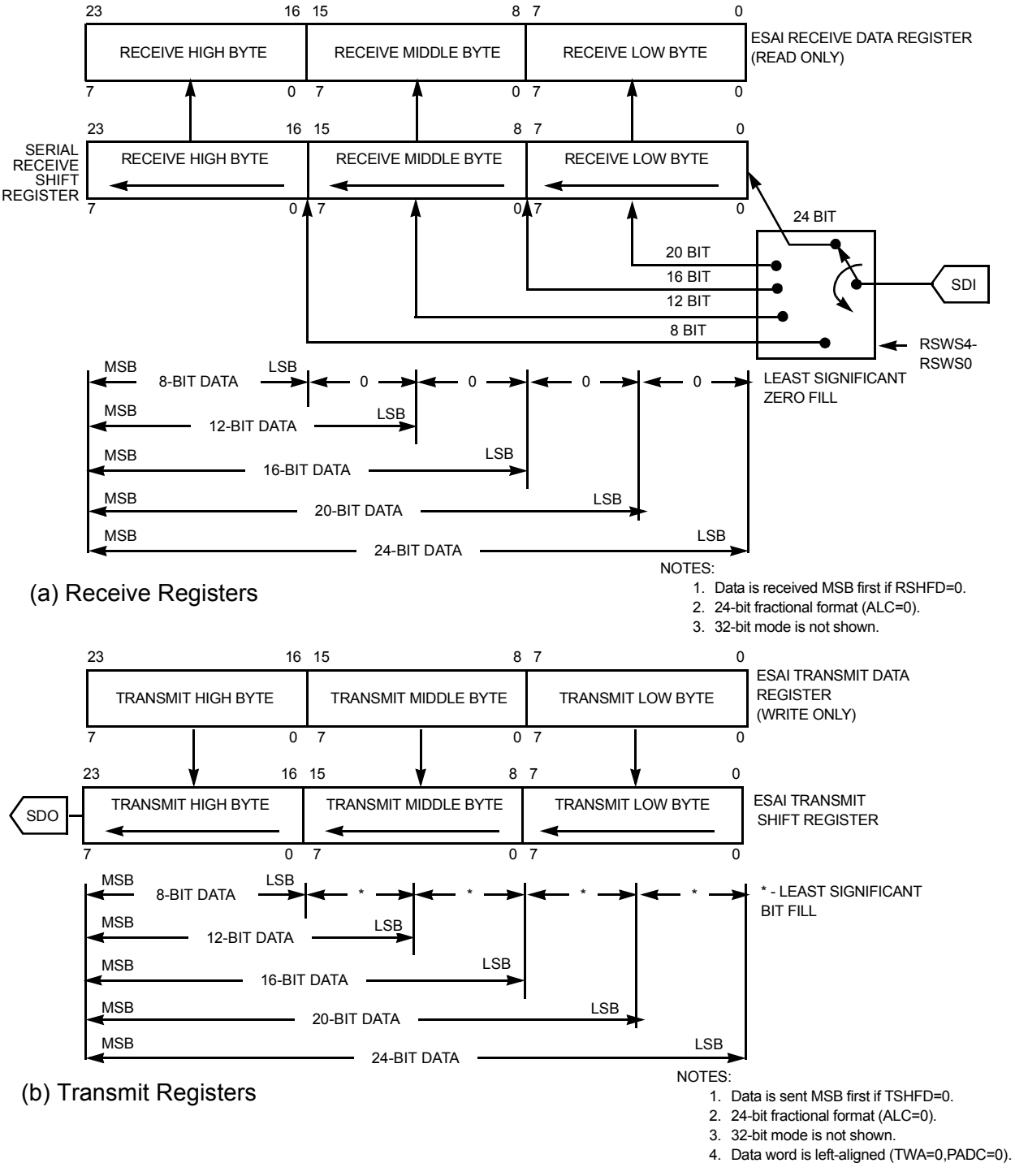
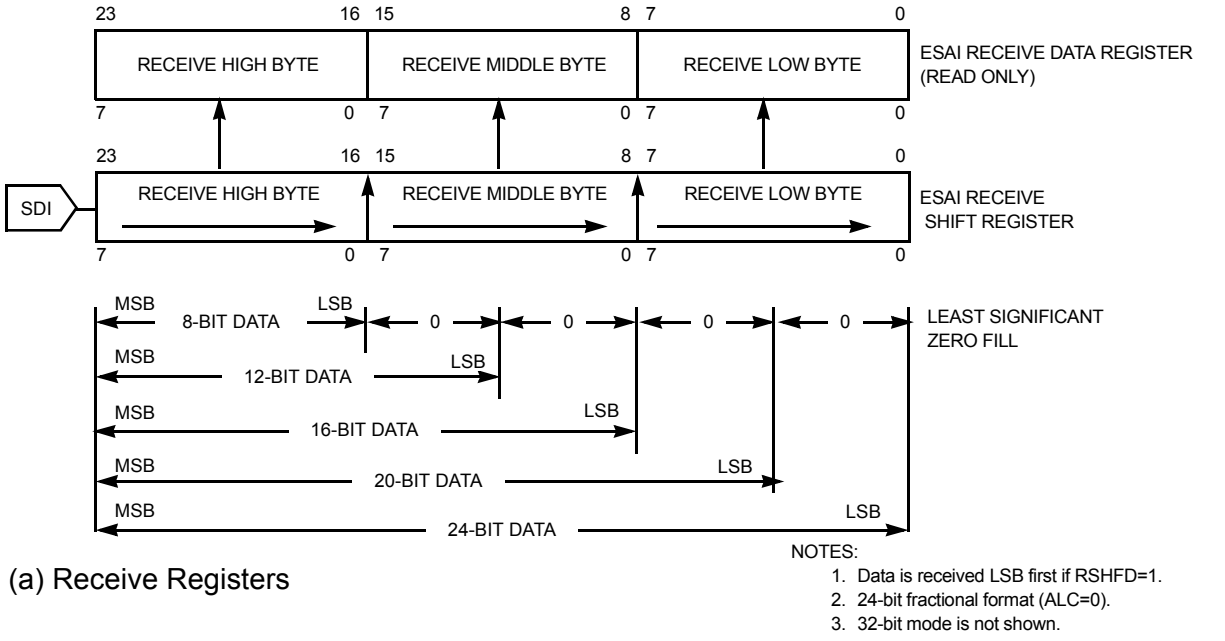
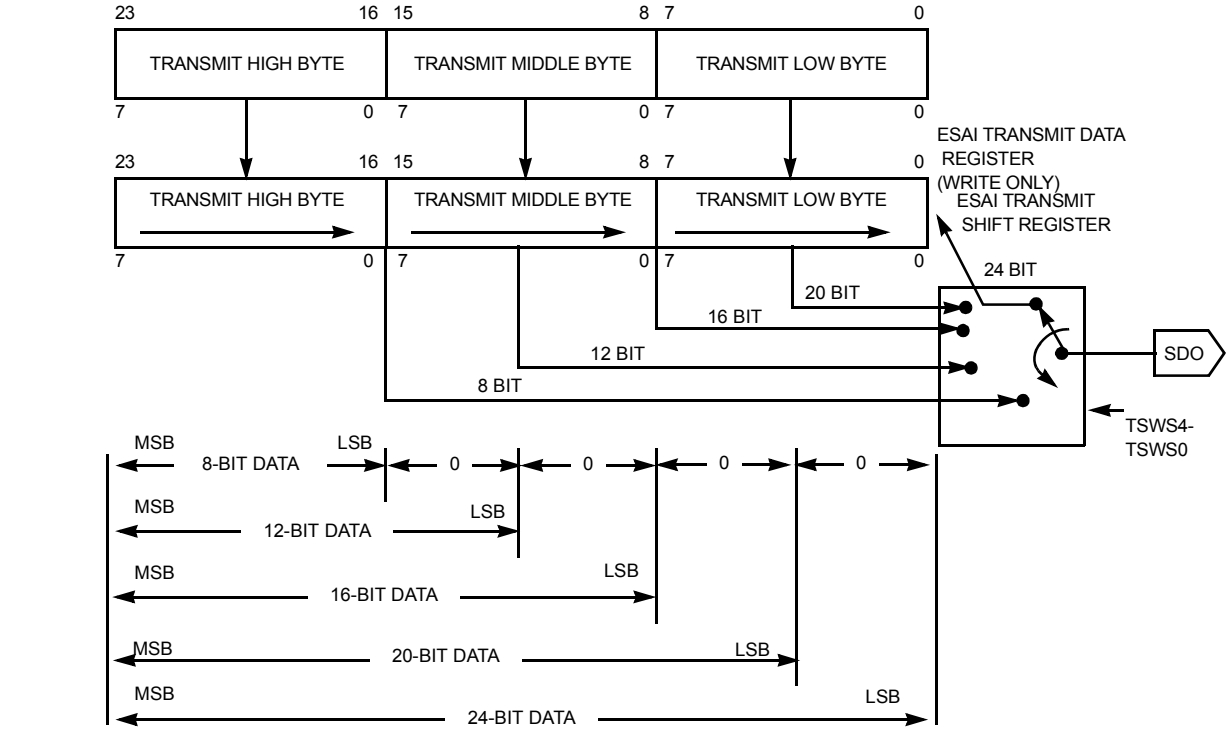


Figure 8-13. ESAI Data Path Programming Model ([R/T]SHFD=0)



(a) Receive Registers

NOTES:
1. Data is received LSB first if RSHFD=1.
2. 24-bit fractional format (ALC=0).
3. 32-bit mode is not shown.



(b) Transmit Registers

NOTES:
1. Data is sent LSB first if TSHFD=1.
2. 24-bit fractional format (ALC=0).
3. 32-bit mode is not shown.
4. Data word is left aligned (TWA=0,PADC=1).

Figure 8-14. ESAI Data Path Programming Model ([R/T]SHFD=1)

8.3.7 ESAI Receive Shift Registers

The receive shift registers (see Figure 8-13 and Figure 8-14) receive the incoming data from the serial receive data pins. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. Data is assumed to be received MSB first if RSHFD=0 and LSB first if RSHFD=1. Data is transferred to the ESAI receive data registers after 8, 12, 16, 20, 24, or 32 serial clock cycles were counted, depending on the slot length control bits in the RCR register.

8.3.8 ESAI Receive Data Registers (RX3, RX2, RX1, RX0)

RX3, RX2, RX1 and RX0 are 24-bit read-only registers that accept data from the receive shift registers when they become full (see Figure 8-13 and Figure 8-14). The data occupies the most significant portion of the receive data registers, according to the ALC control bit setting. The unused bits (least significant portion and 8 most significant bits when ALC=1) read as zeros. The DSP is interrupted whenever RXx becomes full if the associated interrupt is enabled.

8.3.9 ESAI Transmit Shift Registers

The transmit shift registers contain the data being transmitted (see Figure 8-13 and Figure 8-14). Data is shifted out to the serial transmit data pins by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. The number of bits shifted out before the shift registers are considered empty and may be written to again can be 8, 12, 16, 20, 24 or 32 bits (determined by the slot length control bits in the TCR register). Data is shifted out of these registers MSB first if TSHFD=0 and LSB first if TSHFD=1.

8.3.10 ESAI Transmit Data Registers (TX5, TX4, TX3, TX2,TX1,TX0)

TX5, TX4, TX3, TX2, TX1 and TX0 are 24-bit write-only registers. Data to be transmitted is written into these registers and is automatically transferred to the transmit shift registers (see Figure 8-13 and Figure 8-14). The data written (8, 12, 16, 20 or 24 bits) should occupy the most significant portion of the TXx according to the ALC control bit setting. The unused bits (least significant portion and the 8 most significant bits when ALC=1) of the TXx are don't care bits. The DSP is interrupted whenever the TXx becomes empty if the transmit data register empty interrupt has been enabled.

8.3.11 ESAI Time Slot Register (TSR)

The write-only Time Slot Register (TSR) is effectively a null data register that is used when the data is not to be transmitted in the available transmit time slot. The transmit data pins of all the enabled transmitters are in the high-impedance state for the respective time slot where TSR has been written. The Transmitter External Buffer Enable pin (FSR pin when SYN=1, TEBE=1, RFSD=1) disables the external buffers during the slot when the TSR register has been written.

8.3.12 Transmit Slot Mask Registers (TSMA, TSMB)

The Transmit Slot Mask Registers (TSMA and TSMB) are two read/write registers used by the transmitters in network mode to determine for each slot whether to transmit a data word and generate a transmitter empty condition (TDE=1), or to tri-state the transmitter data pins. TSMA and TSMB should each be considered as containing half a 32-bit register TSM. See Figure 8-15 and Figure 8-16. Bit number N in TSM (TS**) is the enable/disable control bit for transmission in slot number N.

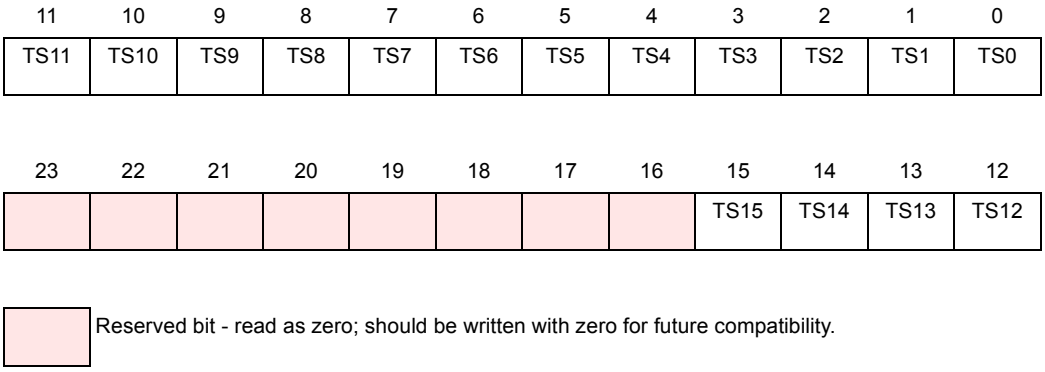


Figure 8-15. TSMA Register

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

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

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

Figure 8-16. TSMB Register

The ESAI TSMA and TSMB registers are located at x:$FFFFB9 and x:$FFFFBA respectively. The ESAI_1 TSMA and TSMB registers are located at y:$FFFF99 and y:$FFFF9A respectively.

When bit number N in TSM is cleared, all the transmit data pins of the enabled transmitters are tri-stated during transmit time slot number N. The data is still transferred from the transmit data registers to the transmit shift registers but neither the TDE nor the TUE flags are set. This means that during a disabled slot, no transmitter empty interrupt is generated. The DSP is interrupted only for enabled slots. Data that is written to the transmit data registers when servicing this request is transmitted in the next enabled transmit time slot.

When bit number N in TSM register is set, the transmit sequence is as usual: data is transferred from the TX registers to the shift registers and transmitted during slot number N, and the TDE flag is set.

Using the slot mask in TSM does not conflict with using TSR. Even if a slot is enabled in TSM, the user may chose to write to TSR instead of writing to the transmit data registers TXx. This causes all the transmit data pins of the enabled transmitters to be tri-stated during the next slot.

Data written to the TSM affects the next frame transmission. The frame being transmitted is not affected by this data and would comply to the last TSM setting. Data read from TSM returns the last written data.

After hardware or software reset, the TSM register is preset to $FFFFFFFF, which means that all 32 possible slots are enabled for data transmission.

NOTE

When operating in normal mode, bit 0 of the mask register must be set, otherwise no output is generated.

8.3.13 Receive Slot Mask Registers (RSMA, RSMB)

The Receive Slot Mask Registers (RSMA and RSMB) are two read/write registers used by the receiver in network mode to determine for each slot whether to receive a data word and generate a receiver full condition (RDF=1), or to ignore the received data. RSMA and RSMB should be considered as each containing half of a 32-bit register RSM. See Table 8-17 and Table 8-18. Bit number N in RSM (RS**) is an enable/disable control bit for receiving data in slot number N.

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

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

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

Figure 8-17. RSMA Register

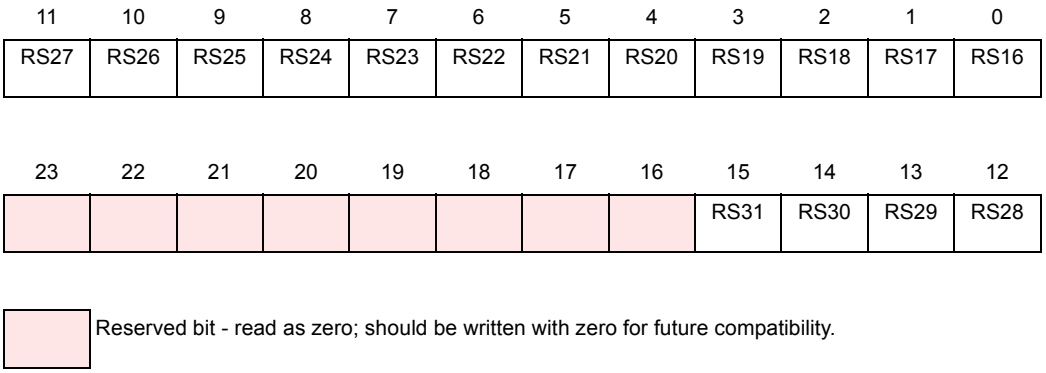


Figure 8-18. RSMB Register

The ESAI RSMA and RSMB registers are located at x:$FFFFBB and x:$FFFFBC respectively. The ESAI_1 RSMA and RSMB registers are located at y:$FFFF9B and y:$FFFF9C respectively.

When bit number N in the RSM register is cleared, the data from the enabled receivers input pins are shifted into their receive shift registers during slot number N. The data is not transferred from the receive shift registers to the receive data registers, and neither the RDF nor the ROE flag is set. This means that during a disabled slot, no receiver full interrupt is generated. The DSP is interrupted only for enabled slots.

When bit number N in the RSM is set, the receive sequence is as usual: data which is shifted into the enabled receivers shift registers is transferred to the receive data registers and the RDF flag is set.

Data written to the RSM affects the next received frame. The frame being received is not affected by this data and would comply to the last RSM setting. Data read from RSM returns the last written data.

After hardware or software reset, the RSM register is preset to $FFFFFFFF, which means that all 32 possible slots are enabled for data reception.

NOTE

When operating in normal mode, bit 0 of the mask register must be set to one, otherwise no input is received.

8.4 Operating Modes

ESAI operating mode are selected by the ESAI control registers (TCCR, TCR, RCCR, RCR and SAICR). The main operating mode are described in the following paragraphs.

8.4.1 ESAI After Reset

Hardware or software reset clears the port control register bits and the port direction control register bits, which configure all ESAI I/O pins as disconnected. The ESAI is in the individual reset state while all ESAI pins are programmed as GPIO or disconnected, and it is active only if at least one of the ESAI I/O pins is programmed as an ESAI pin.

8.4.2 ESAI Initialization

The correct way to initialize the ESAI is as follows:

1. Hardware, software, ESAI individual, or STOP reset.
2. Program ESAI control and time slot registers.
3. Write data to all the enabled transmitters.
4. Configure at least one pin as ESAI pin.

During program execution, all ESAI pins may be defined as GPIO or disconnected, causing the ESAI to stop serial activity and enter the individual reset state. All status bits of the interface are set to their reset state; however, the control bits are not affected. This procedure allows the DSP programmer to reset the ESAI separately from the other internal peripherals. During individual reset, internal DMA accesses to the data registers of the ESAI are not valid and data read is undefined.

The DSP programmer must use an individual ESAI reset when changing the ESAI control registers (except for TEIE, REIE, TLIE, RLIE, TIE, RIE, TE0-TE5, RE0-RE3) to ensure proper operation of the interface.

NOTE

If the ESAI receiver section is already operating with some of the receivers, enabling additional receivers on the fly, i.e., without first putting the ESAI receiver in the personal reset state, by setting their REx control bits will result in erroneous data being received as the first data word for the newly enabled receivers.

8.4.3 ESAI Interrupt Requests

The ESAI can generate eight different interrupt requests (ordered from the highest to the lowest priority)**:**

1. ESAI Receive Data with Exception Status.
   Occurs when the receive exception interrupt is enabled (REIE=1 in the RCR register), at least one of the enabled receive data registers is full (RDF=1) and a receiver overrun error has occurred (ROE=1 in the SAISR register). ROE is cleared by first reading the SAISR and then reading all the enabled receive data registers.
2. ESAI Receive Even Data
   Occurs when the receive even slot data interrupt is enabled (REDIE=1), at least one of the enabled receive data registers is full (RDF=1), the data is from an even slot (REDF=1) and no exception has occurred (ROE=0 or REIE=0).
   Reading all enabled receiver data registers clears RDF and REDF.
3. ESAI Receive Data
   Occurs when the receive interrupt is enabled (RIE=1), at least one of the enabled receive data registers is full (RDF=1), no exception has occurred (ROE=0 or REIE=0) and no even slot interrupt has occurred (REDF=0 or REDIE=0).
   Reading all enabled receiver data registers clears RDF.
4. ESAI Receive Last Slot Interrupt
   Occurs, if enabled (RLIE=1), after the last slot of the frame ended (in network mode only) regardless of the receive mask register setting. The receive last slot interrupt may be used for resetting the receive mask slot register, reconfiguring the DMA channels and reassigning data memory pointers. Using the receive last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame is serviced with the new setting without synchronization problems. Note that the maximum receive last slot interrupt service time should not exceed N-1 ESAI bits service time (where N is the number of bits in a slot).
5. ESAI Transmit Data with Exception Status
   Occurs when the transmit exception interrupt is enabled (TEIE=1), at least one transmit data register of the enabled transmitters is empty (TDE=1) and a transmitter underrun error has occurred (TUE=1). TUE is cleared by first reading the SAISR and then writing to all the enabled transmit data registers, or to the TSR register.
6. ESAI Transmit Last Slot Interrupt
   Occurs, if enabled (TLIE=1), at the start of the last slot of the frame in network mode regardless of the transmit mask register setting. The transmit last slot interrupt may be used for resetting the transmit mask slot register, reconfiguring the DMA channels and reassigning data memory pointers. Using the transmit last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame is serviced with the new setting without synchronization problems. Note that the maximum transmit last slot interrupt service time should not exceed N-1 ESAI bits service time (where N is the number of bits in a slot).
7. ESAI Transmit Even Data
   Occurs when the transmit even slot data interrupt is enabled (TEDIE=1), at least one of the enabled transmit data registers is empty (TDE=1), the slot is an even slot (TEDE=1) and no exception has occurred (TUE=0 or TEIE=0).
   Writing to all the TX registers of the enabled transmitters or to TSR clears this interrupt request.
8. ESAI Transmit Data
   Occurs when the transmit interrupt is enabled (TIE=1), at least one of the enabled transmit data registers is empty (TDE=1), no exception has occurred (TUE=0 or TEIE=0) and no even slot interrupt has occurred (TEDE=0 or TEDIE=0).
   Writing to all the TX registers of the enabled transmitters, or to the TSR clears this interrupt request.

8.4.4 Operating Modes – Normal, Network and On-Demand

The ESAI has three basic operating modes and many data/operation formats.

8.4.4.1 Normal/Network/On-Demand Mode Selection

Selecting between the normal mode and network mode is accomplished by clearing or setting the TMOD0-TMOD1 bits in the TCR register for the transmitter section, as well as in the RMOD0-RMOD1 bits in the RCR register for the receiver section.

For normal mode, the ESAI functions with one data word of I/O per frame (per enabled transmitter or receiver). The normal mode is typically used to transfer data to/from a single device.

For the network mode, 2 to 32 time slots per frame may be selected. During each frame, 0 to 32 data words of I/O may be received/transmitted. In either case, the transfers are periodic. The frame sync signal indicates the first time slot in the frame. Network mode is typically used in time division multiplexed (TDM) networks of codecs, DSPs with multiple words per frame, or multi-channel devices.

Selecting the network mode and setting the frame rate divider to zero (DC=00000) selects the on-demand mode. This special case does not generate a periodic frame sync. A frame sync pulse is generated only when data is available to transmit. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). Therefore, for simplex operation, the synchronous mode could be used; however, for full-duplex operation, the asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into each TX. Although the ESAI is double buffered, only one word can be written to each TX, even if the transmit shift register is empty. The receive and transmit interrupts function as usual using TDE and RDF; however, transmit underruns are impossible for on-demand transmission and are disabled.

8.4.4.2 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of the ESAI may be synchronous or asynchronous, i.e., the transmitter and receiver sections may use common clock and synchronization signals (synchronous operating mode), or they may have their own separate clock and sync signals (asynchronous operating mode). The SYN bit in the SAICR register selects synchronous or asynchronous operation. Since the ESAI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

When SYN is cleared, the ESAI transmitter and receiver clocks and frame sync sources are independent. If SYN is set, the ESAI transmitter and receiver clocks and frame sync come from the transmitter section (either external or internal sources).

Data clock and frame sync signals can be generated internally by the DSP or may be obtained from external sources. If internally generated, the ESAI clock generator is used to derive high frequency clock, bit clock and frame sync signals from the DSP internal system clock.

8.4.4.3 Frame Sync Selection

The frame sync can be either a bit-long or word-long signal. The transmitter frame format is defined by the TFSL bit in the TCR register. The receiver frame format is defined by the RFSL bit in the RCR register.

1. In the word-long frame sync format, the frame sync signal is asserted during the entire word data transfer period. This frame sync length is compatible with Freescale (formerly Motorola) codecs, SPI serial peripherals, serial A/D and D/A converters, shift registers and telecommunication PCM serial I/O.
2. In the bit-long frame sync format, the frame sync signal is asserted for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs and telecommunication PCM serial I/O.

The relative timing of the word length frame sync as referred to the data word is specified by the TFSR bit in the TCR register for the transmitter section and by the RFSR bit in the RCR register for the receive section. The word length frame sync may be generated (or expected) with the first bit of the data word, or with the last bit of the previous word. TFSR and RFSR are ignored when a bit length frame sync is selected.

Polarity of the frame sync signal may be defined as positive (asserted high) or negative (asserted low). The TFSP bit in the TCCR register specifies the polarity of the frame sync for the transmitter section. The RFSP bit in the RCCR register specifies the polarity of the frame sync for the receiver section.

The ESAI receiver looks for a receive frame sync leading edge (trailing edge if RFSP is set) only when the previous frame is completed. If the frame sync goes high before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync or a word length frame sync with RFSR set), the current frame sync is not recognized, and the receiver is internally disabled until the next frame sync. Frames do not have to be adjacent, i.e., a new frame sync does not have to immediately follow the previous frame. Gaps of arbitrary periods can occur between frames. Enabled transmitters are tri-stated during these gaps.

When operating in the synchronous mode (SYN=1), all clocks including the frame sync are generated by the transmitter section.

8.4.4.4 Shift Direction Selection

Some data formats, such as those used by codecs, specify MSB first while other data formats, such as the AES-EBU digital audio interface, specify LSB first. The MSB/LSB first selection is made by programming RSHFD bit in the RCR register for the receiver section and by programming the TSHFD bit in the TCR register for the transmitter section.

8.4.5 Serial I/O Flags

Three ESAI pins (FSR, SCKR and HCKR) are available as serial I/O flags when the ESAI is operating in the synchronous mode (SYN=1). Their operation is controlled by RCKD, RFSD, TEBE bits in the RCR, RCCR and SAICR registers.The output data bits (OF2, OF1 and OF0) and the input data bits (IF2, IF1 and IF0) are double buffered to/from the HCKR, FSR and SCKR pins. Double buffering the flags keeps them in sync with the TX and RX data lines.

Each flag can be separately programmed. Flag 0 (SCKR pin) direction is selected by RCKD, RCKD=1 for output and RCKD=0 for input. Flag 1 (FSR pin) is enabled when the pin is not configured as external transmitter buffer enable (TEBE=0) and its direction is selected by RFSD, RFSD=1 for output and RFSD=0 for input. Flag 2 (HCKR pin) direction is selected by RHCKD, RHCKD=1 for output and RHCKD=0 for input.

When programmed as input flags, the SCKR, FSR and HCKR logic values, respectively, are latched at the same time as the first bit of the receive data word is sampled. Because the input was latched, the signal on the input flag pin (SCKR, FSR or HCKR) can change without affecting the input flag until the first bit of the next receive data word. When the received data words are transferred to the receive data registers, the input flag latched values are then transferred to the IF0, IF1 and IF2 bits in the SAISR register, where they may be read by software.

When programmed as output flags, the SCKR, FSR and HCKR logic values are driven by the contents of the OF0, OF1 and OF2 bits in the SAICR register respectively, and they are driven when the transmit data registers are transferred to the transmit shift registers. The value on SCKR, FSR and HCKR is stable from the time the first bit of the transmit data word is transmitted until the first bit of the next transmit data word is transmitted. Software may change the OF0-OF2 values thus controlling the SCKR, FSR and HCKR pin values for each transmitted word. The normal sequence for setting output flags when transmitting data is as follows: wait for TDE (transmitter empty) to be set; first write the flags, and then write the transmit data to the transmit registers. OF0, OF1 and OF2 are double buffered so that the flag states appear on the pins when the transmit data is transferred to the transmit shift register, i.e., the flags are synchronous with the data.

8.5 GPIO - Pins and Registers

The GPIO functionality of each ESAI port is controlled by three respective registers:

ESAI	ESAI_1
Port C control register (PCRC)	Port E control register (PCRE)
Port C direction register (PRRC)	Port E direction register (PRRE)
Port C data register (PDRC)	Port E data register (PDRE)

8.5.1 Port C (ESAI) GPIO - Pins and Registers

The GPIO functionality of the ESAI port is controlled by three registers: Port C control register (PCRC), Port C direction register (PRRC) and Port C data register (PDRC).

8.5.1.1 Port C Control Register (PCRC)

The read/write 24-bit Port C Control Register (PCRC) in conjunction with the Port C Direction Register (PRRC) controls the functionality of the ESAI GPIO pins. Each of the PC(11:0) bits controls the functionality of the corresponding port pin. See Table 8-12 for the port-pin configurations. Hardware and software reset clear all PCRC bits.

8.5.1.2 Port C Direction Register (PRRC)

The read/write 24-bit Port C Direction Register (PRRC) in conjunction with the Port C Control Register (PCRC) controls the functionality of the ESAI GPIO pins. Table 8-12 describes the port-pin configurations. Hardware and software reset clear all PRRC bits.

Table 8-12. PCRC and PRRC Bits Functionality

PDC[i]	PC[i]	Port Pin[i] Function
0	0	Disconnected
0	1	GPIO input
1	0	GPIO output
1	1	ESAI

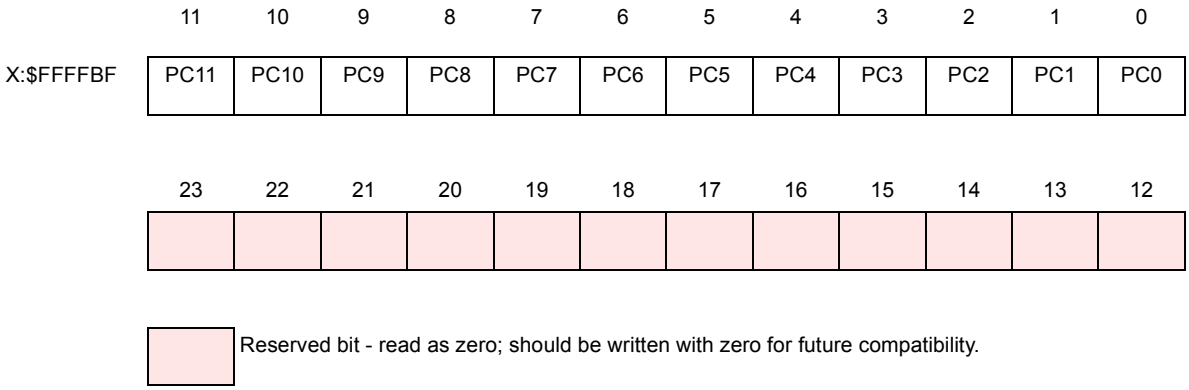


Figure 8-19. PCRC Register

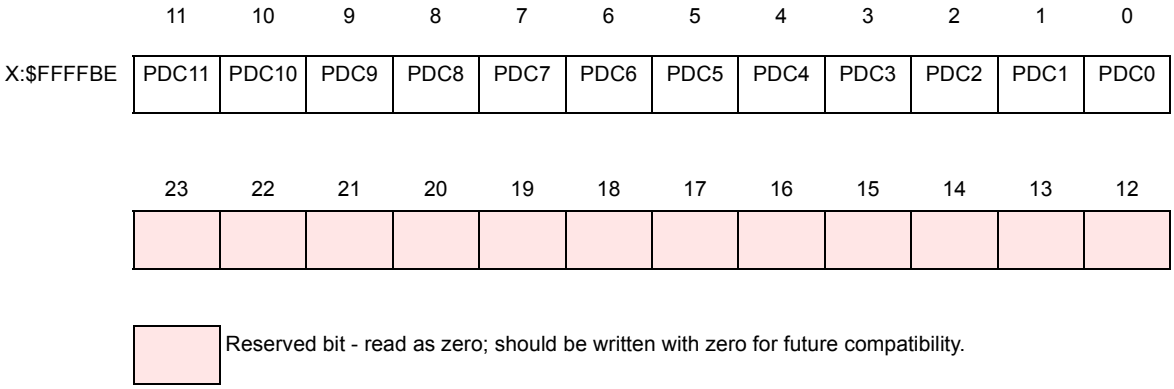


Figure 8-20. PRRC Register

8.5.1.3 Port C Data register (PDRC)

The read/write 24-bit Port C Data Register (see Figure 8-21) is used to read or write data to/from ESAI GPIO pins. Bits PD(11:0) are used to read or write data from/to the corresponding port pins if they are configured as GPIO. If a port pin [i] is configured as a GPIO input, the corresponding PD[i] bit reflects the value present on this pin. If a port pin [i] is configured as a GPIO output, the value written into the corresponding PD[i] bit is reflected on this pin. If a port pin [i] is configured as disconnected, the corresponding PD[i] bit is not reset and contains undefined data.

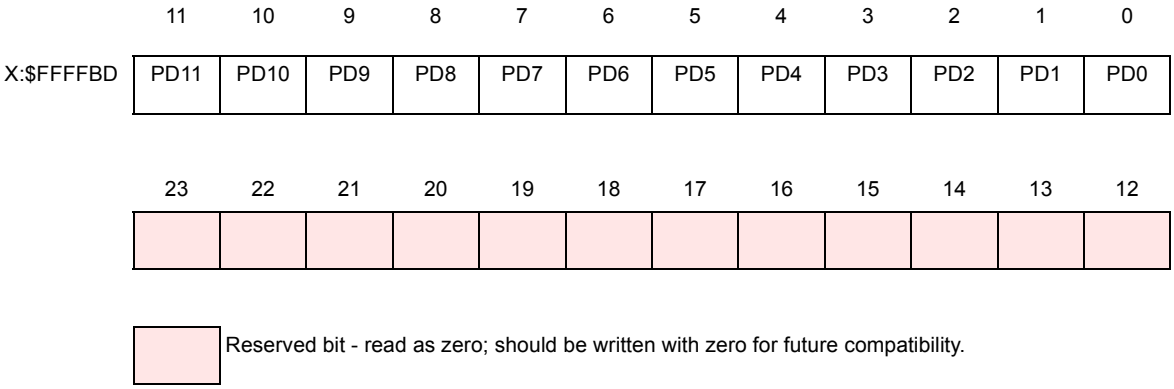


Figure 8-21. PDRC Register

8.5.2 Port E (ESAI_1) GPIO - Pins and Registers

The GPIO functionality of the ESAI_1 port is controlled by three registers: Port E Control register (PCRE), Port E Direction register (PRRE) and Port E Data register (PDRE).

8.5.2.1 Port E Control Register (PCRE)

The read/write 24-bit Port E Control Register (PCRE) in conjunction with the Port E Direction Register (PRRE) controls the functionality of the ESAI_1 GPIO pins. Each of the PE(11:0) bits controls the functionality of the corresponding port pin. See Table 8-12 for the port-pin configurations. Hardware and software reset clear all PCRE bits.

8.5.2.2 Port E Direction Register (PRRE)

The read/write 24-bit Port E Direction Register (PRRE) in conjunction with the Port E Control Register (PCRE) controls the functionality of the ESAI_1 GPIO pins. Table 8-12 describes the port-pin configurations. Hardware and software reset clear all PRRE bits.

Table 8-13. PCRE and PRRE Bits Functionality

PDE[i]	PE[i]	Port Pin[i] Function
0	0	Disconnected
0	1	GPIO input
1	0	GPIO output
1	1	ESAI_1

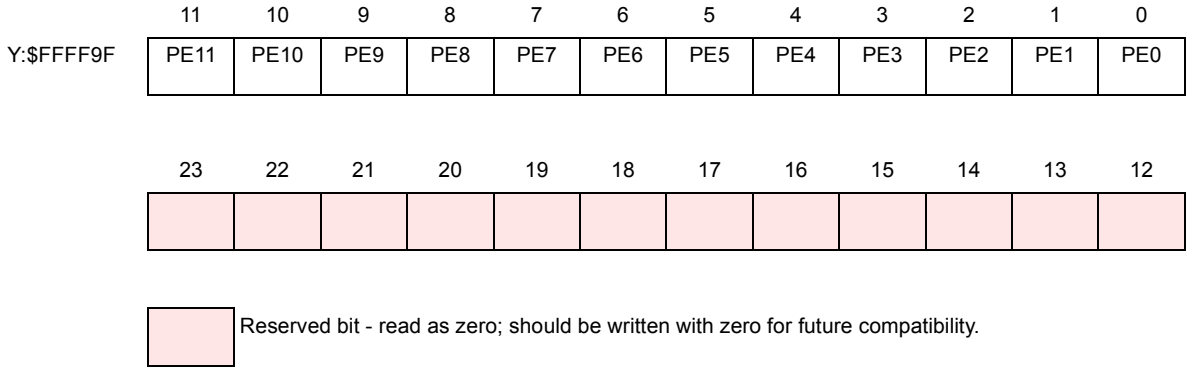


Figure 8-22. PCRE Register

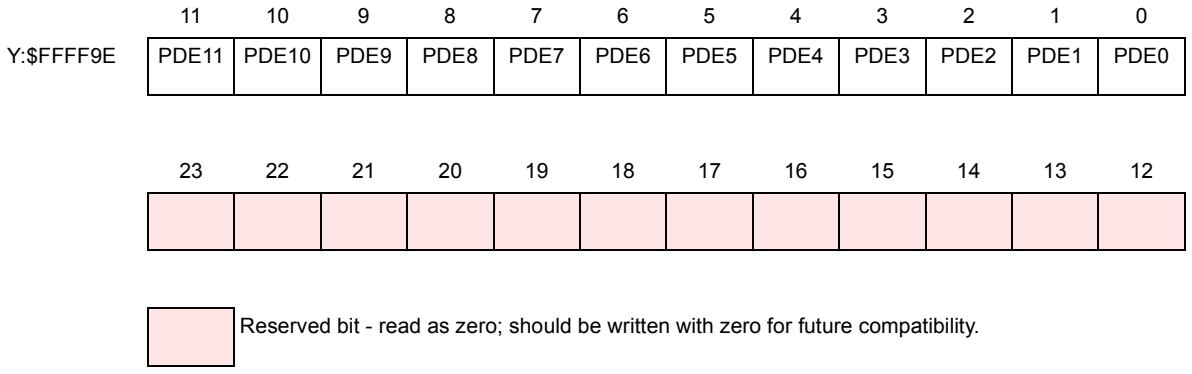


Figure 8-23. PRRE Register

8.5.2.3 Port E Data register (PDRE)

The read/write 24-bit Port E Data Register (see Table 8-24) is used to read or write data to/from ESAI_1 GPIO pins. Bits PD(11:0) are used to read or write data from/to the corresponding port pins if they are configured as GPIO. If a port pin [i] is configured as a GPIO input, the corresponding PD[i] bit will reflect the value present on this pin. If a port pin [i] is configured as a GPIO output, the value written into the corresponding PD[i] bit will be reflected on this pin. If a port pin [i] is configured as disconnected, the corresponding PD[i] bit is not reset and contains undefined data.

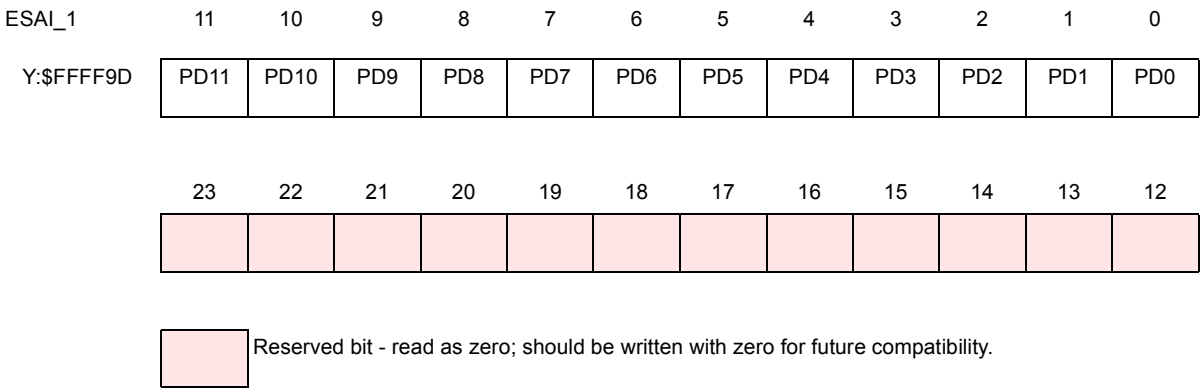


Figure 8-24. PDRE Register

8.6 ESAI Initialization Examples

8.6.1 Initializing the ESAI Using Individual Reset

- The ESAI should be in its individual reset state (PCRC = $000 and PRRC = $000). In the individual reset state, both the transmitter and receiver sections of the ESAI are simultaneously reset. The TPR bit in the TCR register may be used to reset just the transmitter section. The RPR bit in the RCR register may be used to reset just the receiver section.
- Configure the control registers (TCCR, TCR, RCCR, RCR) according to the operating mode, but do not enable transmitters (TE5–TE0 = $0) or receivers (RE3–RE0 = $0). It is possible to set the interrupt enable bits which are in use during the operation (no interrupt occurs).
- Enable the ESAI by setting the PCRC register and PRRC register bits according to pins which are in use during operation.
- Write the first data to be transmitted to the transmitters which are in use during operation.
  This step is needed even if DMA is used to service the transmitters.
- Enable the transmitters and receivers.
- From now on ESAI can be serviced either by polling, interrupts, or DMA.

Operation proceeds as follows:

- For internally generated clock and frame sync, these signals are active immediately after ESAI is enabled (step 3 above).
- Data is received only when one of the receive enable (REx) bits is set and after the occurrence of frame sync signal (either internally or externally generated).
- Data is transmitted only when the transmitter enable (TEx) bit is set and after the occurrence of frame sync signal (either internally or externally generated). The transmitter outputs remain tri-stated after TEx bit is set until the frame sync occurs.

8.6.2 Initializing Just the ESAI Transmitter Section

- It is assumed that the ESAI is operational; that is, at least one pin is defined as an ESAI pin.
- The transmitter section should be in its personal reset state (TPR = 1).
- Configure the control registers TCCR and TCR according to the operating mode, making sure to clear the transmitter enable bits (TE0 - TE5). TPR must remain set.
- Take the transmitter section out of the personal reset state by clearing TPR.
- Write first data to the transmitters which will be used during operation. This step is needed even if DMA is used to service the transmitters.
- Enable the transmitters by setting their TE bits.
- Data is transmitted only when the transmitter enable (TEx) bit is set and after the occurrence of frame sync signal (either internally or externally generated). The transmitter outputs remain tri-stated after TEx bit is set until the frame sync occurs.
- From now on the transmitters are operating and can be serviced either by polling, interrupts, or DMA.

8.6.3 Initializing Just the ESAI Receiver Section

- It is assumed that the ESAI is operational; that is, at least one pin is defined as an ESAI pin.
- The receiver section should be in its personal reset state (RPR = 1).
- Configure the control registers RCCR and RCR according to the operating mode, making sure to clear the receiver enable bits (RE0 - RE3). RPR must remain set.

- Take the receiver section out of the personal reset state by clearing RPR.
- Enable the receivers by setting their RE bits.
- From now on the receivers are operating and can be serviced either by polling, interrupts, or DMA.

Notes

Chapter 9
Triple Timer Module

The timers in the DSP56374 internal triple timer module act as timed pulse generators or as pulse-width modulators. Each of the three timers has a single signal (TIOx) that can function as a GPIO signal or as a timer signal. These three timers can also function as event counters to capture an event or measure the width or period of a signal. Two of the timer pins also have a third functional option. TIO2 can be programmed as a PLL PLOCK pin, and TIO1 can be programmed as a hardware watchdog timer (WDT).

9.1 Overview

The timer module contains a common 21-bit prescaler and three independent and identical general-purpose, 24-bit timer/event counters, each with its own register set. Each of the timers has the following capabilities:

- Uses internal or external clocking
- Interrupts the DSP56374 after a specified number of events (clocks) or signals an external device after counting internal events
- Triggers DMA transfers after a specified number of events (clocks) occurs
- Connects to the external world through one bidirectional signal, designated TIO[0, 1, 2] for timers 0, 1, 2.

When TIO is configured as an input, the timer functions as an external event counter or measures external pulse width/signal period. When TIO is configured as an output, the timer functions as a timer, a watchdog timer, or a pulse-width modulator. When the timer does not use TIO, it can be used as a GPIO signal (also called TIO[0, 1, 2]).

9.1.1 Triple Timer Module Block Diagram

Figure 9-1 shows a block diagram of the triple timer module. This module includes a 24-bit Timer Prescaler Load Register (TPLR), a 24-bit Timer Prescaler Count Register (TPCR), and three timers. Each timer can use the prescaler clock as its clock source.

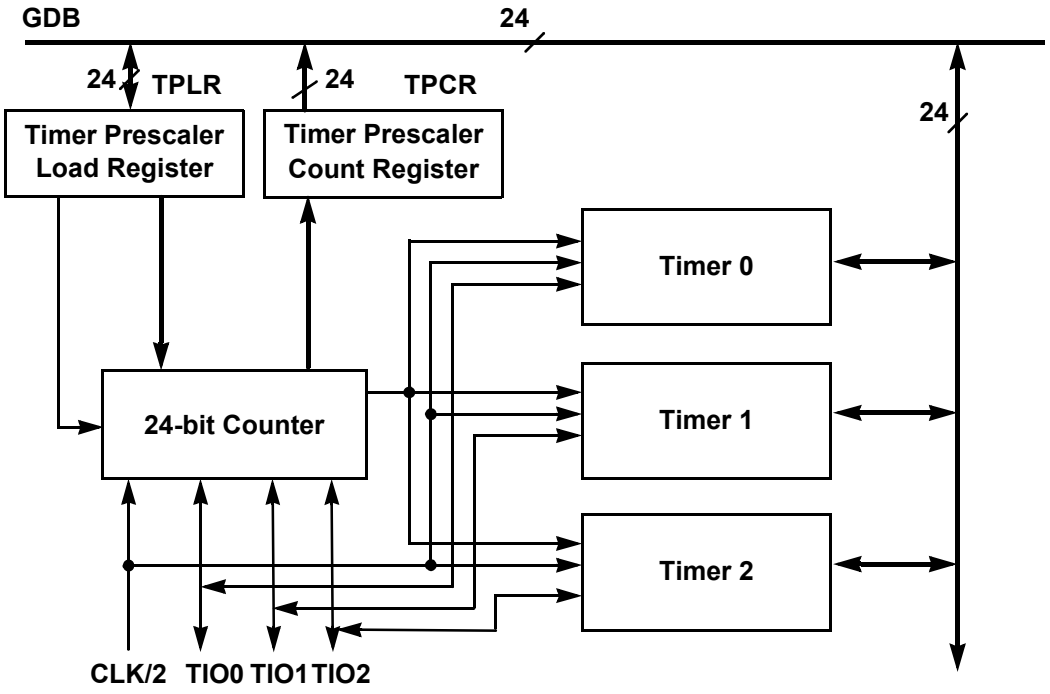


Figure 9-1. Triple Timer Module Block Diagram

9.1.2 Individual Timer Block Diagram

Figure 9-2 shows the structure of an individual timer block. The DSP56374 treats each timer as a memory-mapped peripheral with four registers occupying four 24-bit words in the X data memory space. The three timers are identical in structure and function. Either standard polled or interrupt programming techniques can be used to service the timers. A single, generic timer is discussed in this chapter. Each timer includes the following:

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

The timer mode is controlled by the TC[3–0] bits which are TCSR[7–4]. For a listing of the timer modes and descriptions of their operations, see Section 9.3, *Operating Modes*.

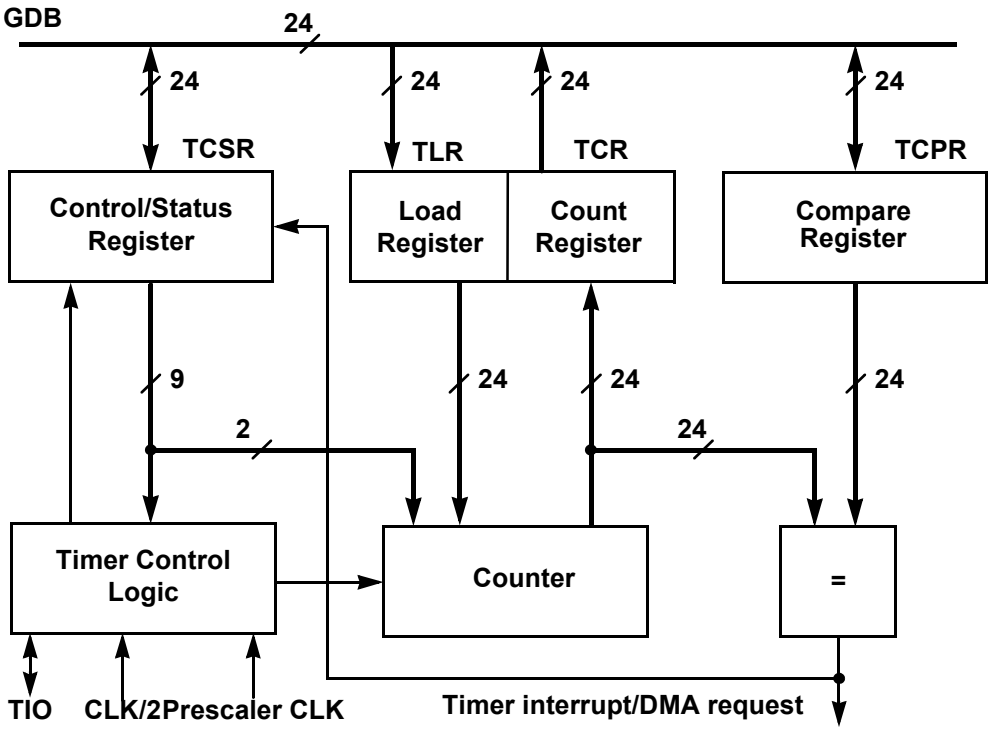


Figure 9-2. Timer Module Block Diagram

9.2 Operation

This section discusses the following timer basics:

- Reset
- Initialization
- Exceptions

9.2.1 Timer After Reset

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

9.2.2 Timer Initialization

To initialize a timer, do the following:

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

9.2.3 Timer Exceptions

Each timer can generate two different exceptions:

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

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

1. Configure the interrupt service routine (ISR):
   a. Load vector base address register — `VBA (b23-8)`
   b. Define I_VEC to be equal to the VBA value (if that is nonzero). If it is defined, I_VEC must be defined for the assembler before the interrupt equate file is included.
   c. Load the exception vector table entry: two-word fast interrupt, or jump/branch to subroutine (long interrupt). — `p:TIM0C`
2. Configure the interrupt trigger:
   a. Enable and prioritize overall peripheral interrupt functionality.
   `IPRP (TOL[1-0])`
   b. Enable a specific peripheral interrupt.
   `TCSR0 (TCIE)`
   c. Unmask interrupts at the global level.
   `SR (I[1-0])`
   d. Configure a peripheral interrupt-generating function.
   `TCSR0 (TC[7-4])`
   e. Enable peripheral and associated signals. — `TCSR0 (TE)`

9.3 Operating Modes

These timers have operating modes that meet a variety of system requirements, as follows:

- Timer
  - GPIO, mode 0: Internal timer interrupt generated by the internal clock
  - Pulse, mode 1: External timer pulse generated by the internal clock
  - Toggle, mode 2: Output timing signal toggled by the internal clock
  - Event counter, mode 3: Internal timer interrupt generated by an external clock
- Measurement
  - Input width, mode 4: Input pulse width measurement
  - Input period, mode 5: Input signal period measurement
  - Capture, mode 6: Capture external signal
- PWM, mode 7: Pulse width modulation
- Watchdog
  - Pulse, mode 9: Output pulse, internal clock
  - Toggle, mode 10: Output toggle, internal clock

Note: To ensure proper operation, the TCSR TC[3–0] bits should be changed only when the timer is disabled (that is, when TCSR[TE] is cleared).

9.3.1 Triple Timer Modes

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

- The TCSR[TE] bit is set to clear the counter and enable the timer. Clearing TCSR[TE] disables the timer.

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

9.3.1.1 Timer GPIO (Mode 0)

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

In Mode 0, the timer generates an internal interrupt when a counter value is reached, if the timer compare interrupt is enabled (see Figure 9-3 and Figure 9-4). When the counter equals the TCPR value, TCSR[TCF] is set and a compare interrupt is generated if the TCSR[TCIE] bit is set. If the TCSR[TRM] bit is set, the counter is reloaded with the TLR value at the next timer clock and the count is resumed. If TCSR[TRM] is cleared, the counter continues to increment on each timer clock signal. This process repeats until the timer is disabled.

Mode 0 (internal clock, no timer output): TRM = 1

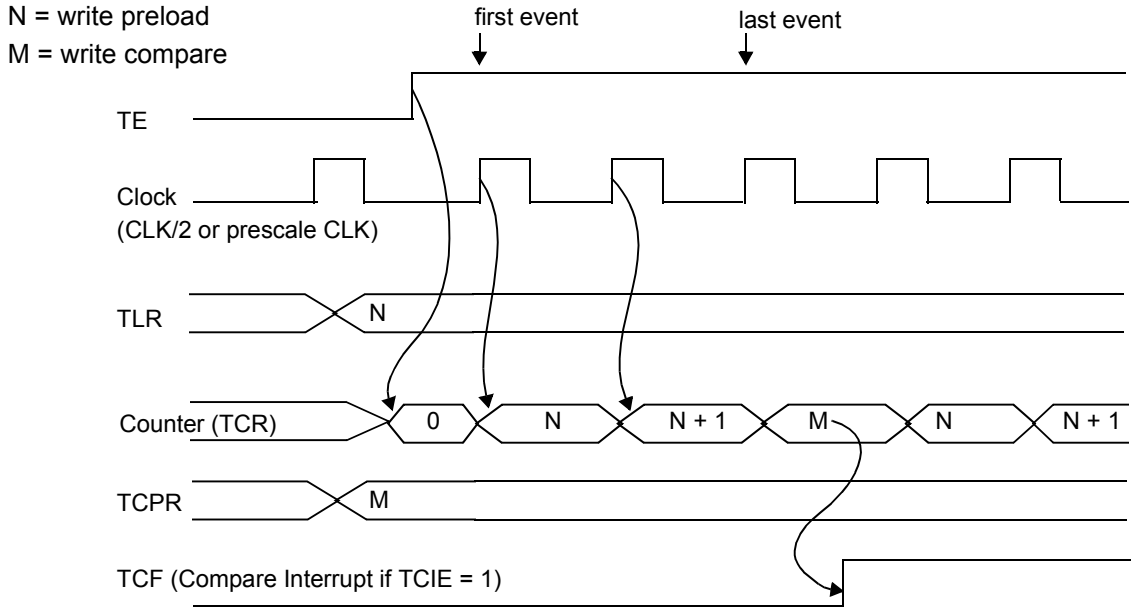


Figure 9-3. Timer Mode (TRM = 1)

Mode 0 (internal clock, no timer output): TRM = 0

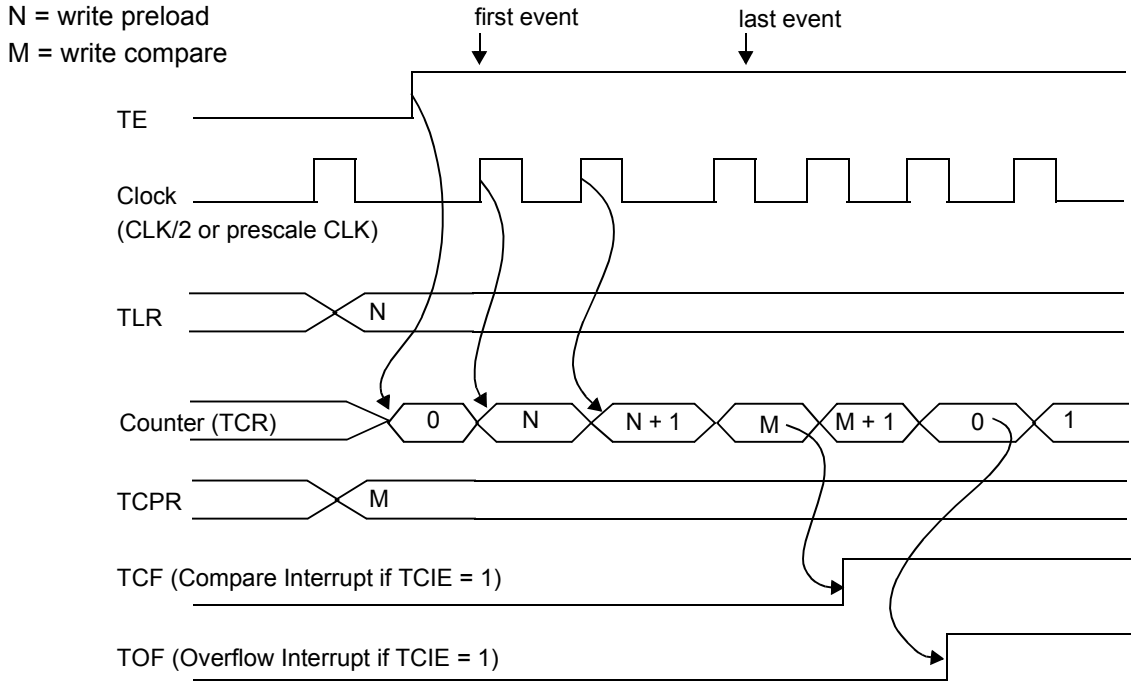


Figure 9-4. Timer Mode (TRM = 0)

9.3.1.2 Timer Pulse (Mode 1)

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

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

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

Mode 1 (internal clock): TRM = 1

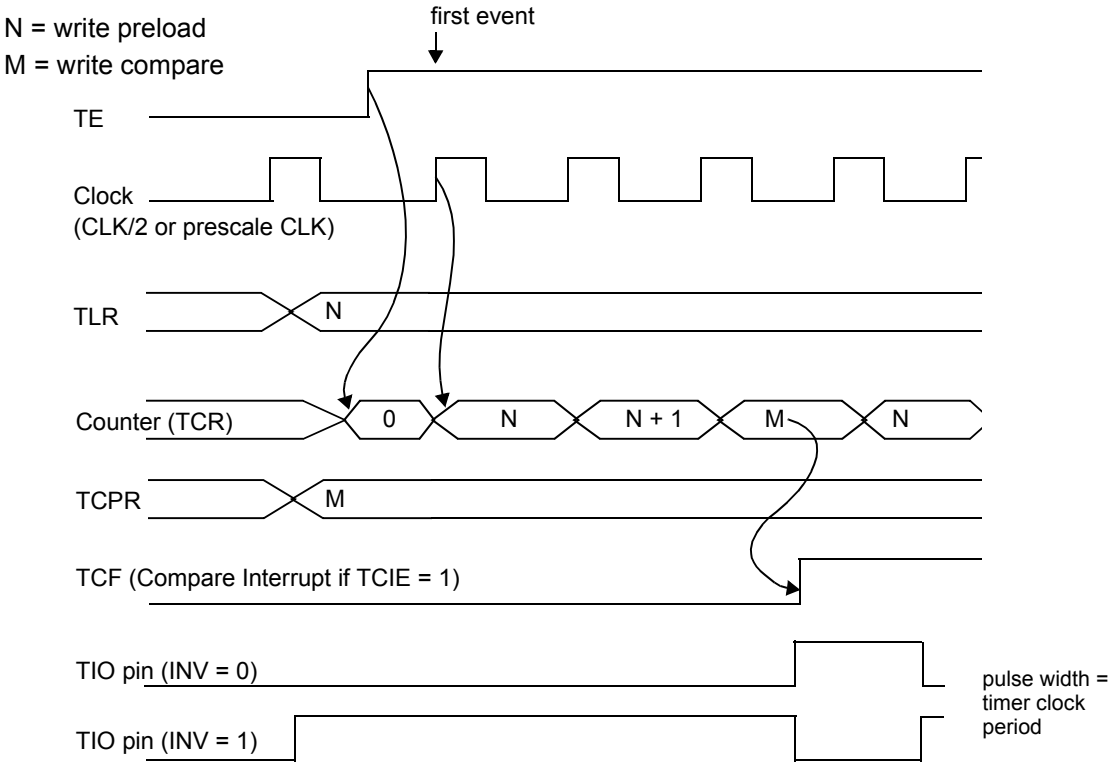


Figure 9-5. Pulse Mode (TRM = 1)

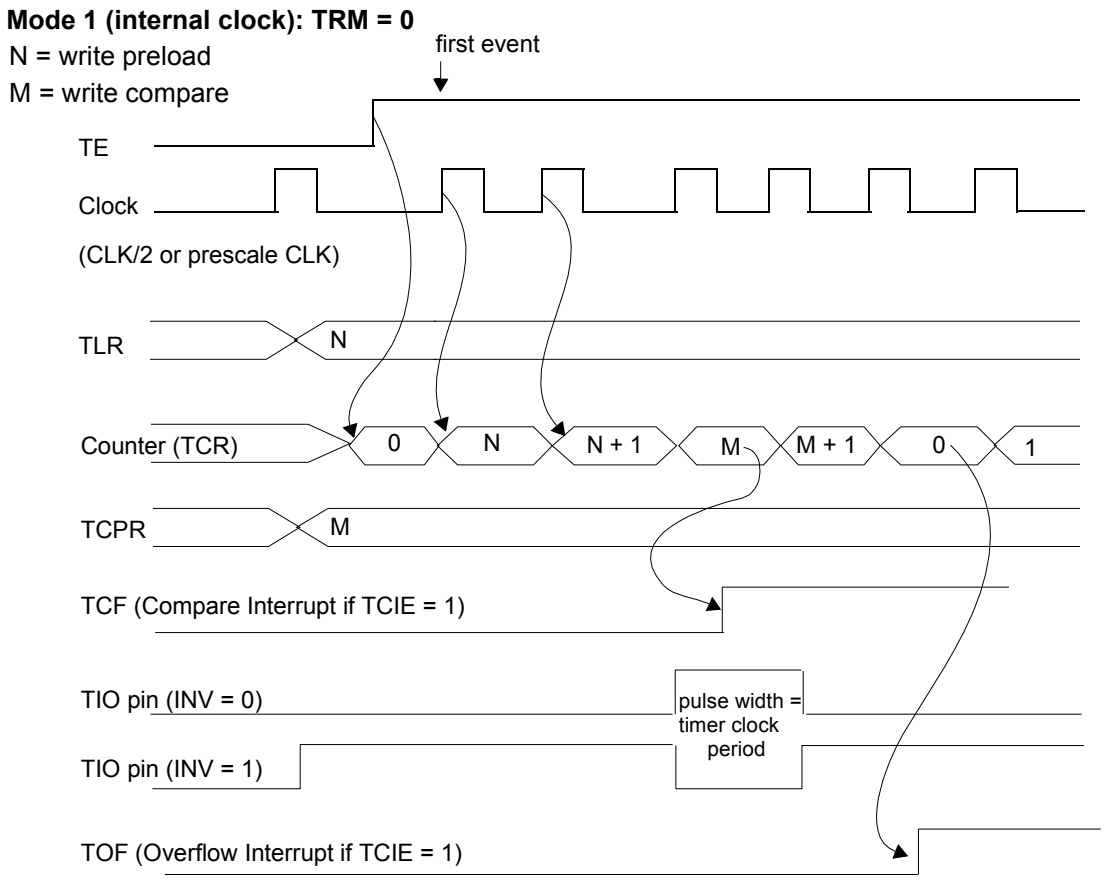


Figure 9-6. Pulse Mode (TRM = 0)

9.3.1.3 Timer Toggle (Mode 2)

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

In Mode 2, the timer periodically toggles the polarity of the TIO signal. When the timer is enabled, the TIO signal is loaded with the value of the TCSR[INV] bit. When the counter value matches the value in the TCPR, the polarity of the TIO output signal is inverted. TCSR[TCF] is set, and a compare interrupt is generated if the TCSR[TCIE] bit is set. If the TCSR[TRM] bit is set, the counter is loaded with the value of the TLR when the next timer clock is received, and the count resumes. If the TRM bit is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is cleared (disabling the timer). The TCPR[TLR] value sets the delay between starting the timer and toggling the TIO signal. To generate output signals with a delay of X clock cycles between toggles, set the TLR value to X/2, and set the TCSR[TRM] bit. This process repeats until the timer is disabled (that is, TCSR[TE] is cleared).

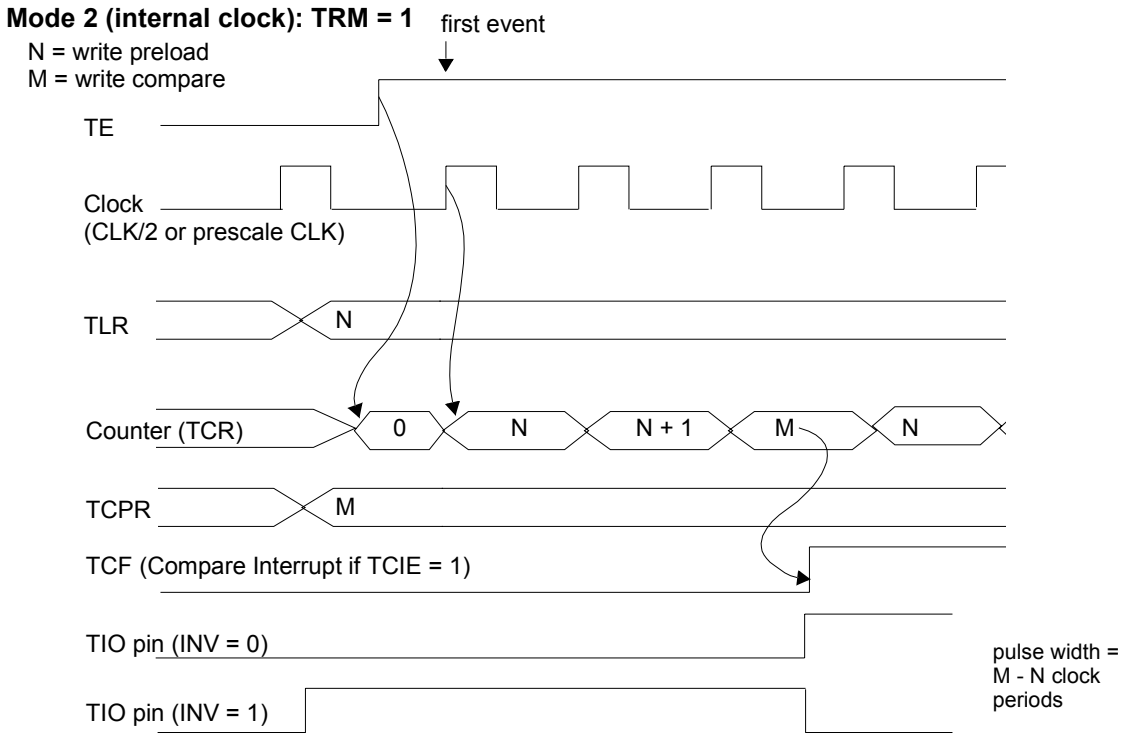


Figure 9-7. Toggle Mode, TRM = 1

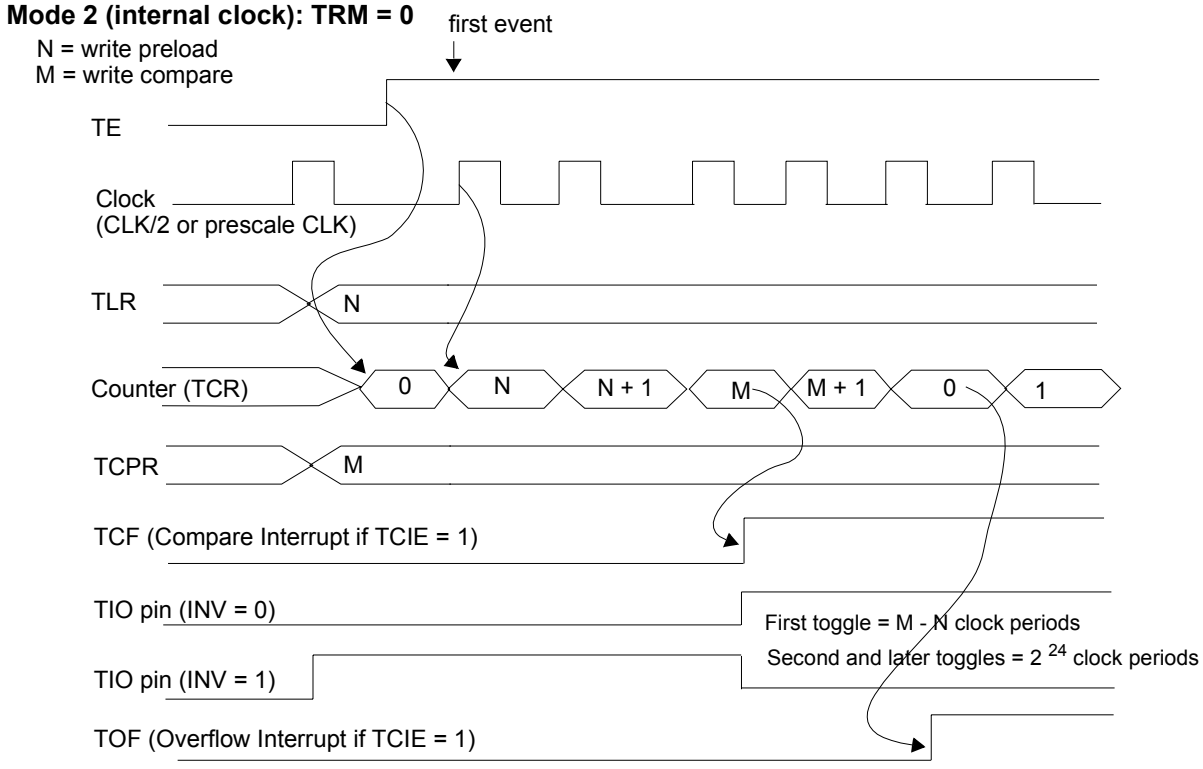


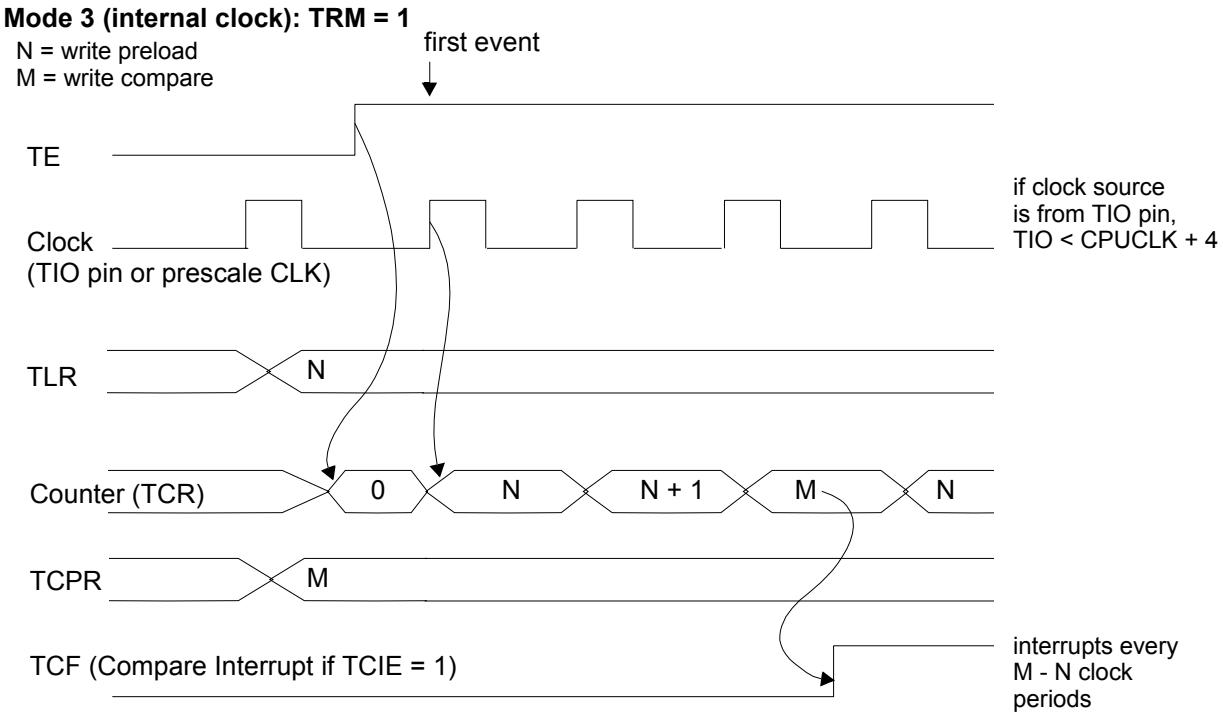
Figure 9-8. Toggle Mode, TRM = 0

9.3.1.4 Timer Event Counter (Mode 3)

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

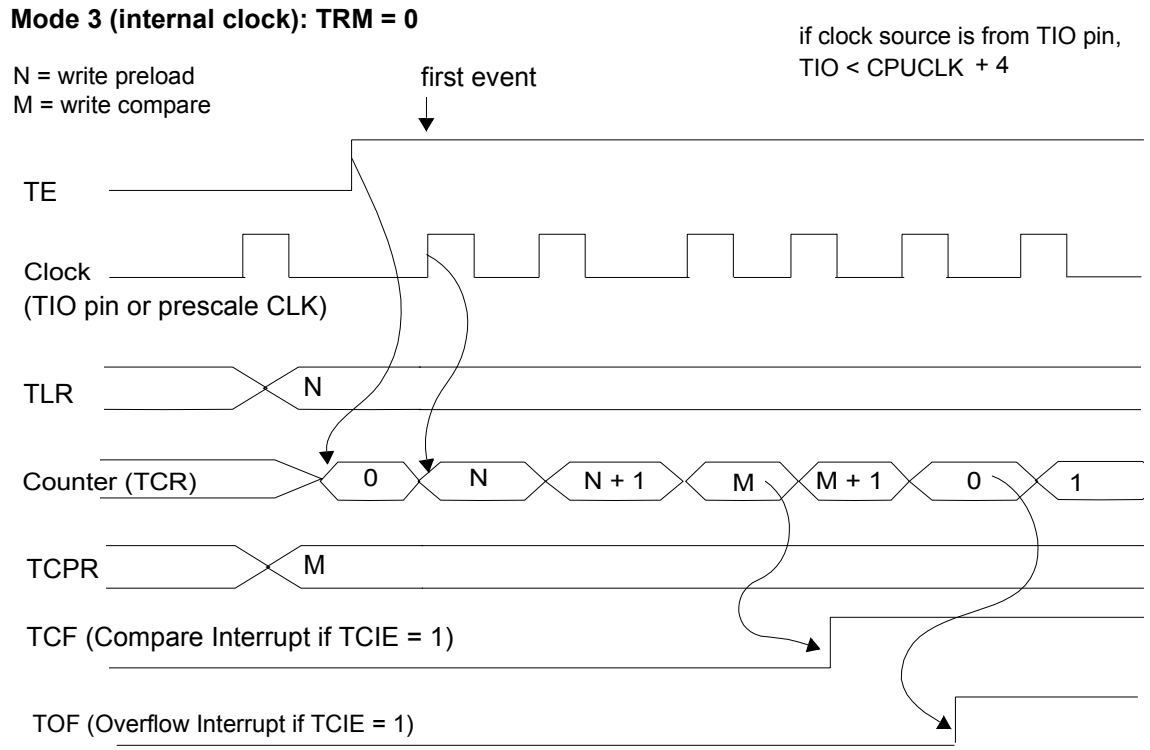
In Mode 3, the timer counts external events and issues an interrupt (if interrupt enable bits are set) when the timer counts a preset number of events. The timer clock signal can be taken from either the TIO input signal or the prescaler clock output. If an external clock is used, it is synchronized internally to the internal clock, and its frequency must be less than the DSP56374 internal operating frequency divided by 4. The value of the TCSR[INV] bit determines whether low-to-high (0 to 1) transitions or high-to-low (1 to 0) transitions increment the counter. If the INV bit is set, high-to-low transitions increment the counter. If the INV bit is cleared, low-to-high transitions increment the counter.

When the counter matches the value contained in the TCPR, TCSR[TCF] is set and a compare interrupt is generated if the TCSR[TCIE] bit is set. If the TCSR[TRM] bit is set, the counter is loaded with the value of the TLR when the next timer clock is received, and the count is resumed. If the TCSR[TRM] bit is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is disabled.



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

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



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

Figure 9-10. Event Counter Mode, TRM = 0

9.3.2 Signal Measurement Modes

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

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

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

9.3.2.1 Measurement Input Width (Mode 4)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	0	0	4	Input width	Measurement	Input	Internal

In Mode 4, the timer counts the number of clocks that occur between opposite edges of an input signal. After the first appropriate transition (as determined by the TCSR[INV] bit) occurs on the TIO input signal, the counter is loaded with the TLR value. If TCSR[INV] is set, the timer starts on the first high-to-low (1 to 0) signal transition on the TIO signal. If the INV bit is cleared, the timer starts on the first low-to-high (that is, 0 to 1) transition on the TIO signal. When the first transition opposite in polarity to the INV bit setting occurs on the TIO signal, the counter stops. TCSR[TCF] is set and a compare interrupt is generated if the TCSR[TCIE] bit is set. The value of the counter (which measures the width of the TIO pulse) is loaded into the TCR, which can be read to determine the external signal pulse width. If the TCSR[TRM] bit is set, the counter is loaded with the TLR value on the first timer clock received following the next valid transition on the TIO input signal, and the count resumes. If TCSR[TRM] is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is disabled.

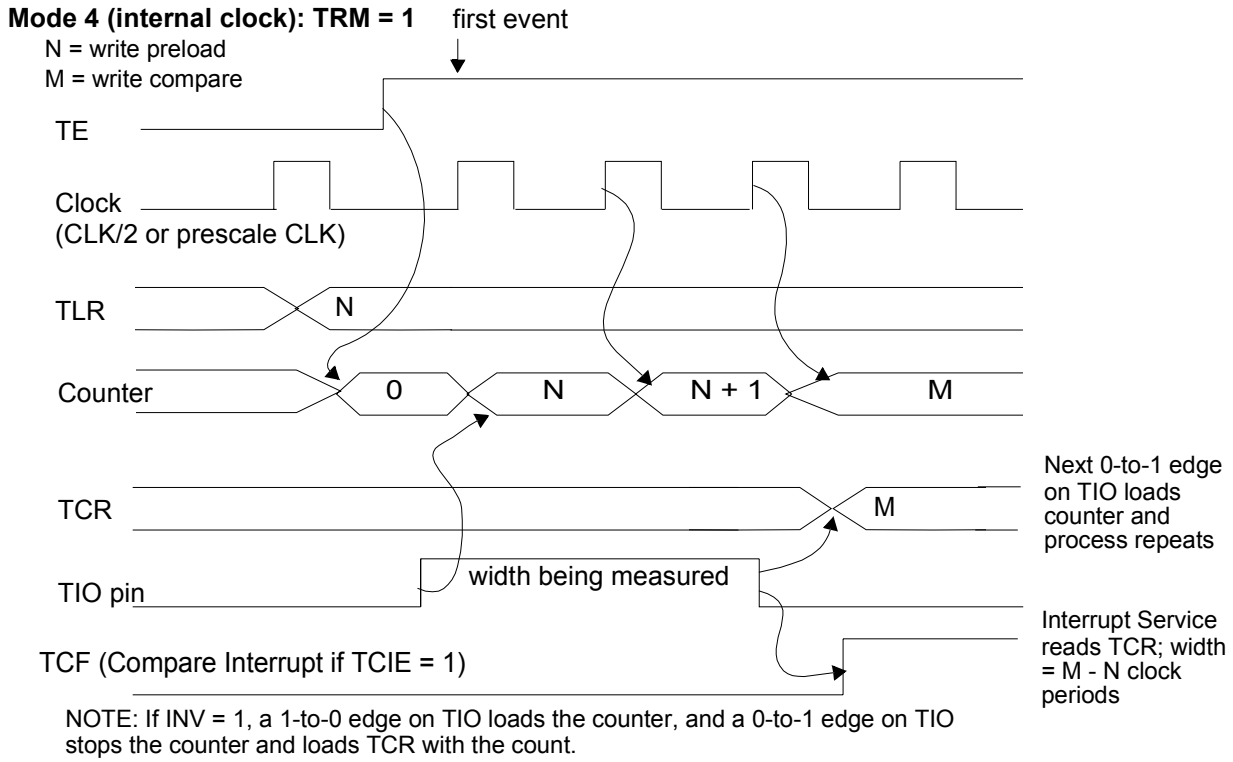


Figure 9-11. Pulse Width Measurement Mode, TRM = 1

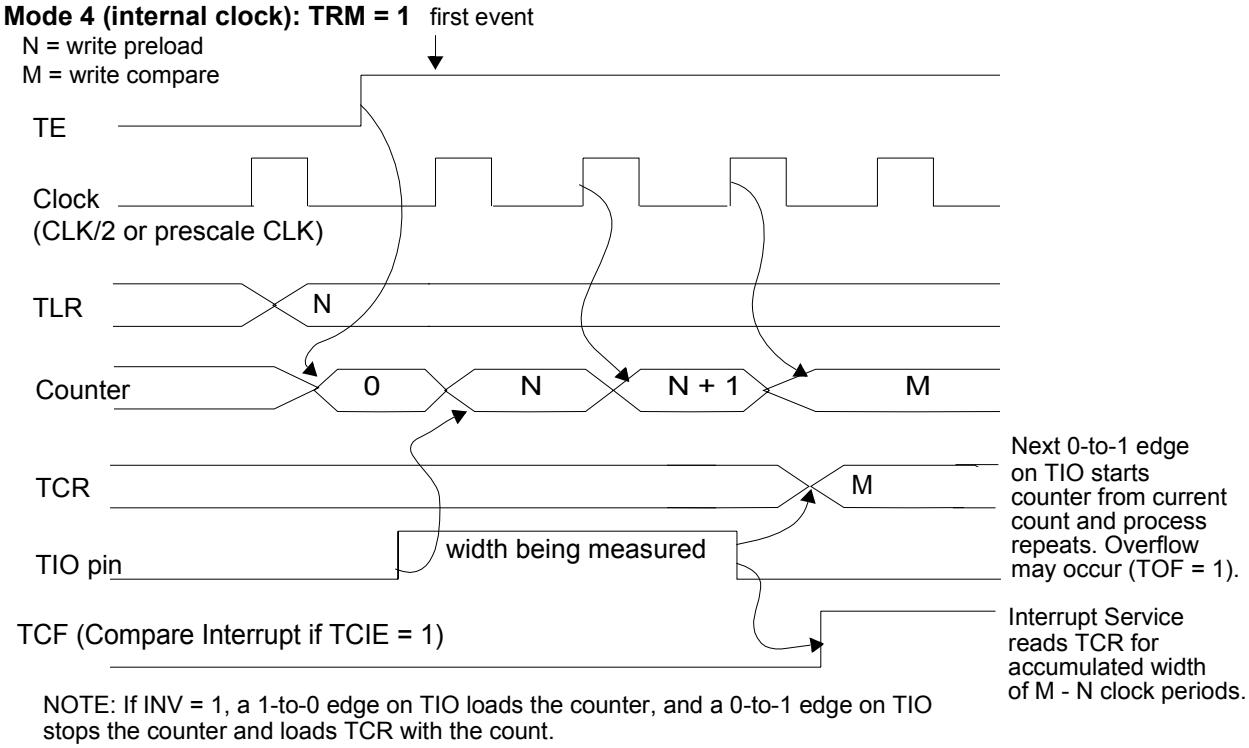


Figure 9-12. Pulse Width Measurement Mode, TRM = 0

9.3.2.2 Measurement Input Period (Mode 5)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	0	1	5	Input period	Measurement	Input	Internal

In Mode 5, the timer counts the period between the reception of signal edges of the same polarity across the TIO signal. The value of the INV bit determines whether the period is measured between consecutive low-to-high (0 to 1) transitions of TIO or between consecutive high-to-low (1 to 0) transitions of TIO. If INV is set, high-to-low signal transitions are selected. If INV is cleared, low-to-high signal transitions are selected. After the first appropriate transition occurs on the TIO input signal, the counter is loaded with the TLR value. On the next signal transition of the same polarity that occurs on TIO, TCSR[TCF] is set, and a compare interrupt is generated if the TCSR[TCIE] bit is set. The contents of the counter load into the TCR. The TCR then contains the value of the time that elapsed between the two signal transitions on the TIO signal. After the second signal transition, if the TCSR[TRM] bit is set, the TCSR[TE] bit is set to clear the counter and enable the timer. The counter is repeatedly loaded and incremented until the timer is disabled. If the TCSR[TRM] bit is cleared, the counter continues to increment until it overflows.

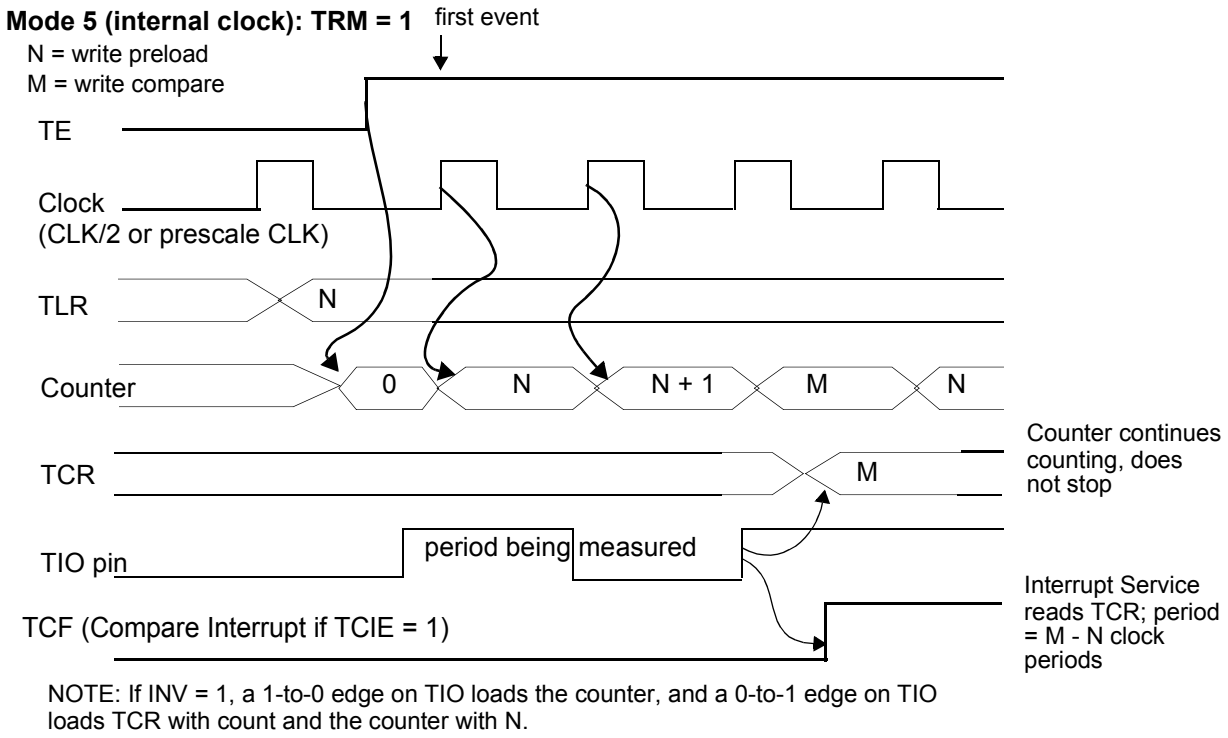


Figure 9-13. Period Measurement Mode, TRM = 1

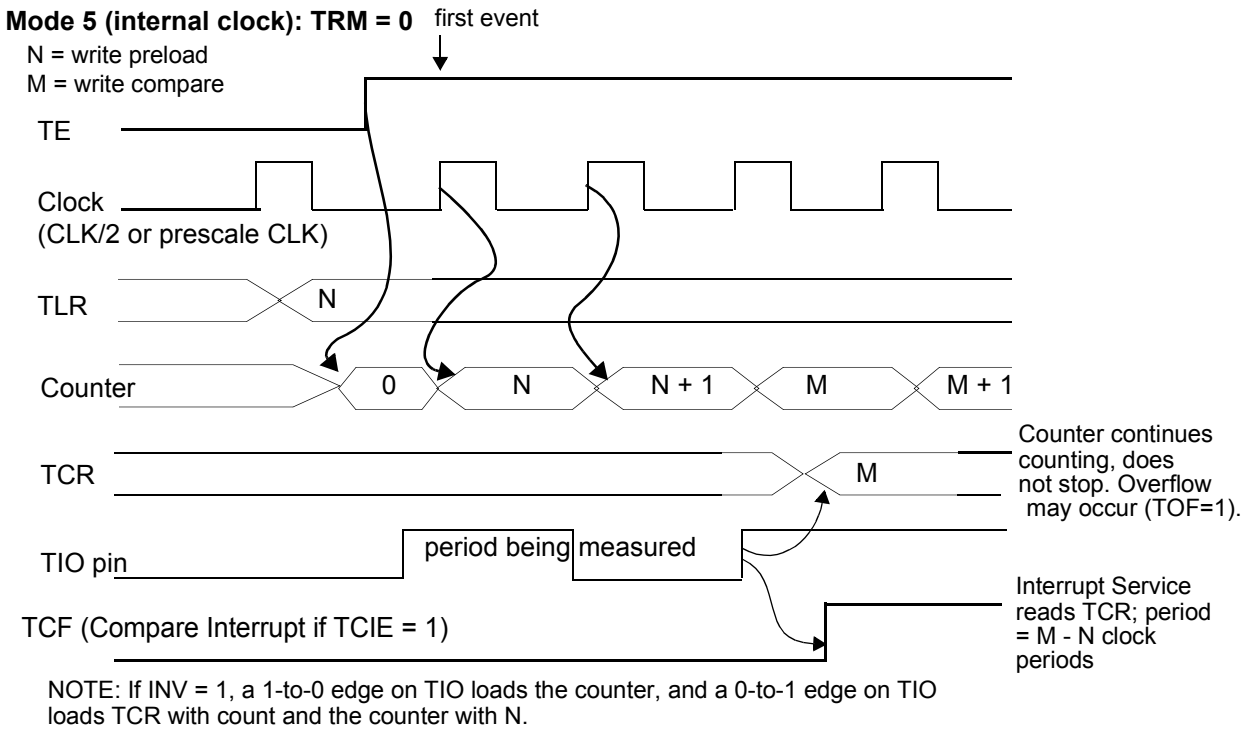
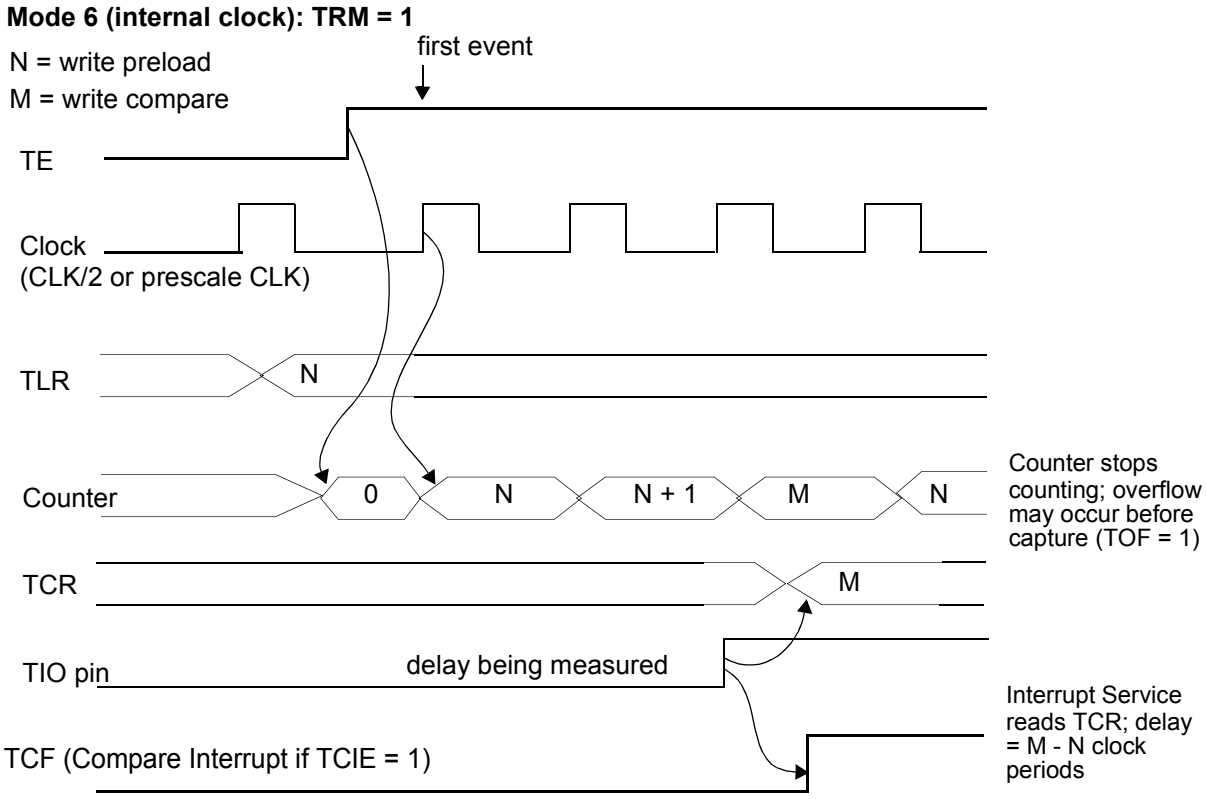


Figure 9-14. Period Measurement Mode, TRM = 0

9.3.2.3 Measurement Capture (Mode 6)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	1	0	6	Capture	Measurement	Input	Internal

In Mode 6, the timer counts the number of clocks that elapse between when the timer starts and when an external signal is received. At the first appropriate transition of the external clock detected on the TIO signal, TCSR[TCF] is set and, if the TCSR[TCIE] bit is set, a compare interrupt is generated. The counter halts. The contents of the counter are loaded into the TCR. The value of the TCR represents the delay between the setting of the TCSR[TE] bit and the detection of the first clock edge signal on the TIO signal. The value of the INV bit determines whether a high-to-low (1 to 0) or low-to-high (0 to 1) transition of the external clock signals the end of the timing period. If the INV bit is set, a high-to-low transition signals the end of the timing period. If INV is cleared, a low-to-high transition signals the end of the timing period.



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

Figure 9-15. Capture Measurement Mode, TRM = 0

9.3.3 Pulse Width Modulation (PWM, Mode 7)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
0	1	1	1	7	Pulse width modulation	PWM	Output	Internal

In Mode 7, the timer generates periodic pulses of a preset width. When the counter equals the value in the TCPR, the TIO output signal is toggled and TCSR[TCF] is set. The contents of the counter are placed into the TCR. If the TCSR[TCIE] bit is set, a compare interrupt is generated. The counter continues to increment on each timer clock.

If counter overflow occurs, the TIO output signal is toggled, TCSR[TOF] is set, and an overflow interrupt is generated if the TCSR[TOIE] bit is set. If the TCSR[TRM] bit is set, the counter is loaded with the TLR value on the next timer clock and the count resumes. If the TCSR[TRM] bit is cleared, the counter continues to increment on each timer clock. This process repeats until the timer is disabled.

When the TCSR[TE] bit is set and the counter starts, the TIO signal assumes the value of INV. On each subsequent toggle of the TIO signal, the polarity of the TIO signal is reversed. For example, if the INV bit is set, the TIO signal generates the following signal: 1010. If the INV bit is cleared, the TIO signal generates the following signal: 0101.

The value of the TLR determines the output period ($FFFFFF – TLR + 1). The timer counter increments the initial TLR value and toggles the TIO signal when the counter value exceeds $FFFFFF. The duty cycle of the TIO signal is determined by the value in the TCPR. When the value in the TLR increments to a value equal to the value in the TCPR, the TIO signal is toggled. The duty cycle is equal to ($FFFFFF – TCPR) divided by ($FFFFFF – TLR + 1). For a 50 percent duty cycle, the value of TCPR is equal to ($FFFFFF + TLR + 1)/2.

NOTE

The value in TCPR must be greater than the value in TLR.

Period = $FFFFFF - TLR + 1
Duty cycle = ($FFFFFF - TCPR)
Ensure that TCPR > TLR for correct functionality

Mode 7 (internal clock): TRM = 1

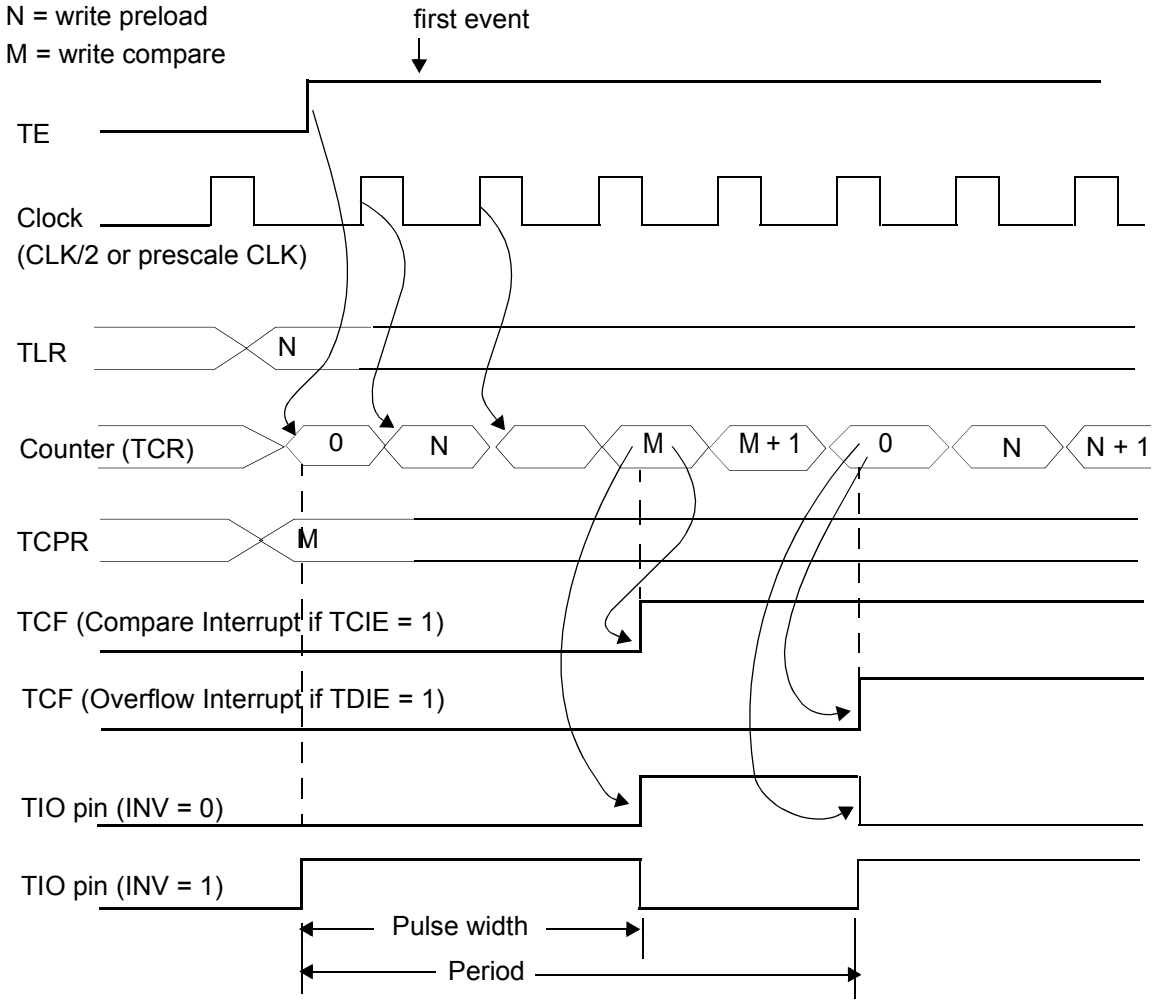
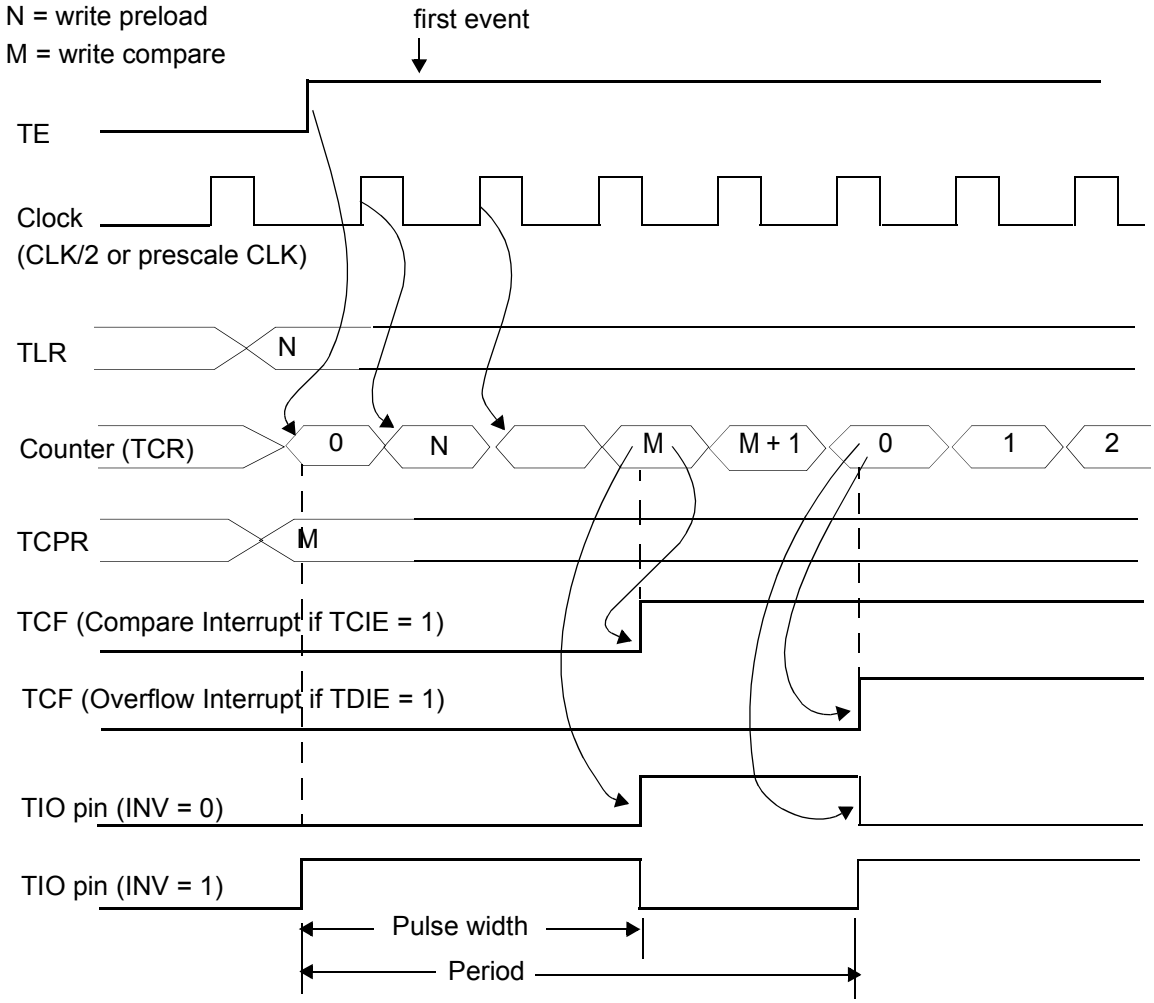


Figure 9-16. Pulse Width Modulation Toggle Mode, TRM = 1

Period = $FFFFFF - TLR + 1
Duty cycle = ($FFFFFF - TCPR)
Ensure that TCPR > TLR for correct functionality

Mode 7 (internal clock): TRM = 0



NOTE: On overflow, TCR is loaded with the value of TLR.

Figure 9-17. Pulse Width Modulation Toggle Mode, TRM = 0

9.3.4 Watchdog Modes

The following watchdog timer modes are provided:

- Watchdog Pulse
- Watchdog Toggle

9.3.4.1 Watchdog Pulse (Mode 9)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
1	0	0	1	9	Pulse	Watchdog	Output	Internal

In Mode 9, the timer generates an external signal at a preset rate. The signal period is equal to the period of one timer clock. After the counter reaches the value in the TCPR, if the TCSR[TRM] bit is set, the counter is loaded with the TLR value on the next timer clock and the count resumes. Therefore TRM = 1 is not useful for watchdog functions. If the TCSR[TRM] bit is cleared, the counter continues to increment on

each subsequent timer clock. This process repeats until the timer is disabled (that is, TCSR[TE] is cleared). If the counter overflows, a pulse is output on the TIO signal with a pulse width equal to the timer clock period. If the INV bit is set, the pulse polarity is high (logical 1). If INV is cleared, the pulse polarity is low (logical 0). The counter reloads when the TLR is written with a new value while the TCSR[TE] bit is set. In Mode 9, internal logic preserves the TIO value and direction for an additional 2.5 internal clock cycles after the hardware $\overline{\text{RESET}}$ signal is asserted. This convention ensures that a valid $\overline{\text{RESET}}$ signal is generated when the TIO signal resets the DSP56374.

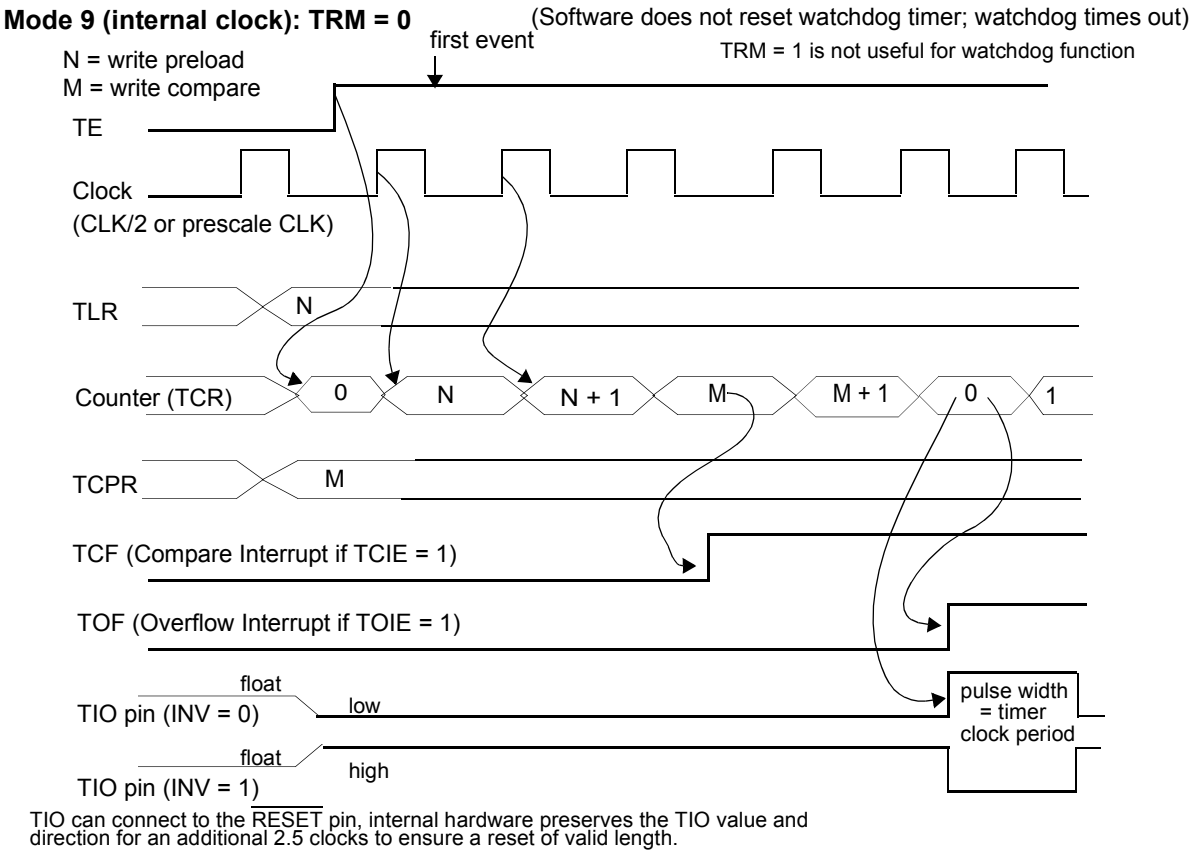
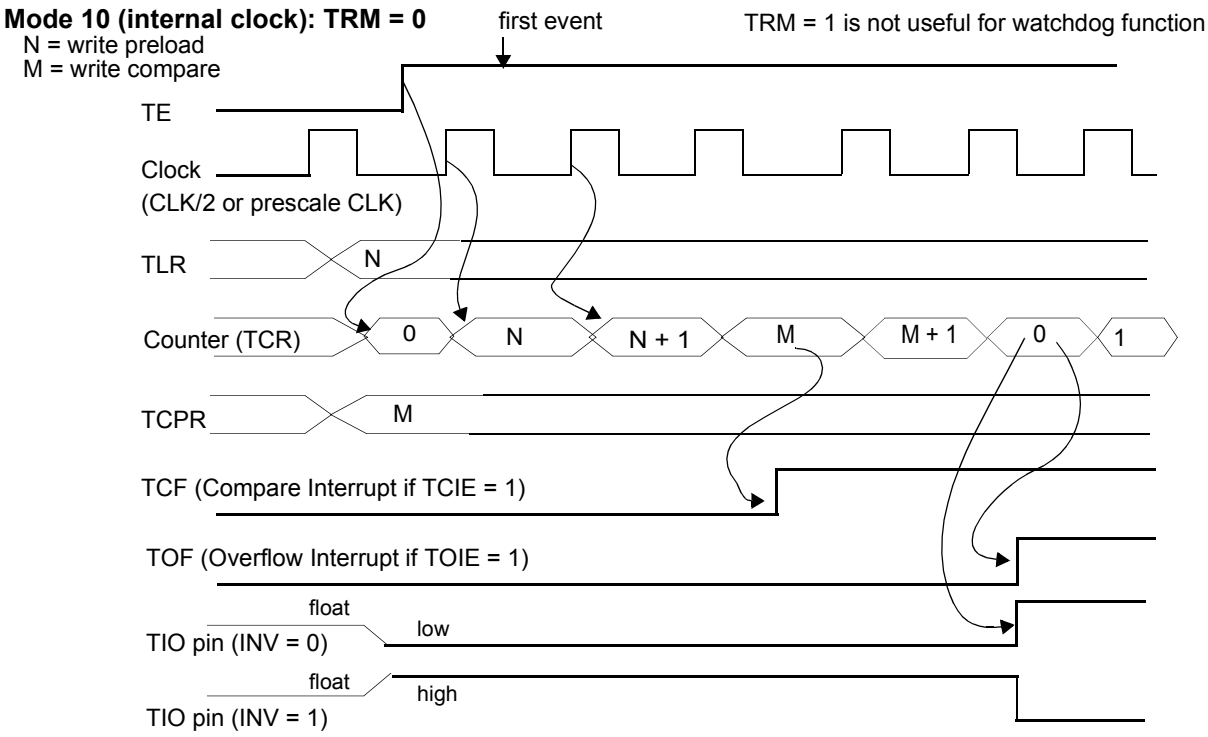


Figure 9-18. Watchdog Pulse Mode

9.3.4.2 Watchdog Toggle (Mode 10)

Bit Settings				Mode Characteristics				
TC3	TC2	TC1	TC0	Mode	Name	Function	TIO	Clock
1	0	1	0	10	Toggle	Watchdog	Output	Internal

In Mode 10, the timer toggles an external signal after a preset period. The TIO signal is set to the value of the INV bit.When the counter equals the value in the TCPR, TCSR[TCF] is set, and a compare interrupt is generated if the TCSR[TCIE] bit is also set. If the TCSR[TRM] bit is set, the counter loads with the TLR value on the next timer clock and the count resumes. Therefore, TRM = 1 is not useful for watchdog functions. If the TCSR[TRM] bit is cleared, the counter continues to increment on each subsequent timer clock. When a counter overflow occurs, the polarity of the TIO output signal is inverted. The counter is reloaded whenever the TLR is written with a new value while the TCSR[TE] bit is set. This process repeats until the timer is disabled. In Mode 10, internal logic preserves the TIO value and direction for an additional 2.5 internal clock cycles after the hardware $\overline{\text{RESET}}$ signal is asserted. This convention ensures that a valid reset signal is generated when the TIO signal resets the DSP56374.



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

Figure 9-19. Watchdog Toggle Mode

9.3.4.3 Reserved Modes

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

9.3.5 Special Cases

The following special cases apply during wait and stop state.

- Timer behavior during wait — Timer clocks are active during the execution of the WAIT instruction and timer activity is undisturbed. If a timer interrupt is generated, the DSP56374 leaves the wait state and services the interrupt.
- Timer behavior during stop — During execution of the STOP instruction, the timer clocks are disabled, timer activity stops, and the TIO signals are disconnected. Any external changes that happen to the TIO signals are ignored when the DSP56374 is in stop state. To ensure correct operation, disable the timers before the DSP56374 is placed in stop state.

9.3.6 DMA Trigger

Each timer can also trigger DMA transfers if a DMA channel is programmed to be triggered by a timer event. The timer issues a DMA trigger on every event in all modes of operation. To ensure that all DMA triggers are serviced, provide for the preceding DMA trigger to be serviced before the DMA channel receives the next trigger.

9.4 Triple Timer Module Programming Model

The timer programmer's model in Figure 9-20 shows the structure of the timer registers.

9.4.1 Prescaler Counter

The prescaler counter is a 21-bit counter that decrements on the rising edge of the prescaler input clock. The counter is enabled when at least one of the three timers is enabled (that is, one or more of the timer enable bits are set) and is using the prescaler output as its source (that is, one or more of the PCE bits are set).

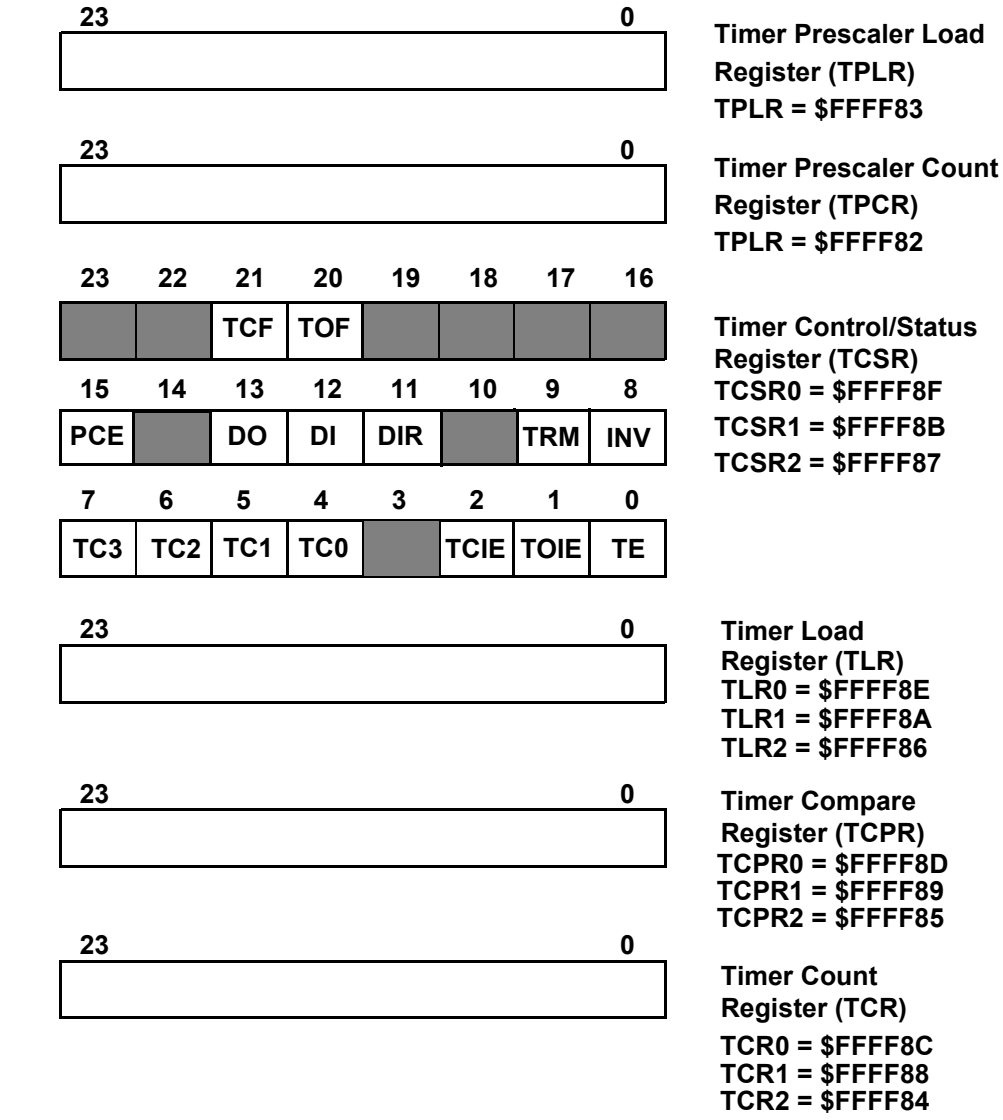


Figure 9-20. Timer Module Programmer's Model

9.4.2 Timer Prescaler Load Register (TPLR)

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

23	22	21	20	19	18	17	16	15	14	13	12
	PS1	PS0	PL20	PL19	PL18	PL17	PL16	PL15	PL14	PL13	PL12
11	10	9	8	7	6	5	4	3	2	1	0
PL11	PL10	PL9	PL8	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0
	— Reserved bit. Read as 0. Write to 0 for future compatibility										

Table 9-1. Timer Prescaler Load Register (TPLR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23		0	Reserved. Write to zero for future compatibility.
22–21	PS[1–0]	0	**Prescaler Source** Control the source of the prescaler clock. The prescaler's use of a TIO signal is not affected by the TCSR settings of the timer of the corresponding TIO signal. If the prescaler source clock is external, the prescaler counter is incremented by signal transitions on the TIO signal. The external clock is internally synchronized to the internal clock. The external clock frequency must be lower than the DSP56374 internal operating frequency divided by 4 (that is, CLK/4). NOTE: To ensure proper operation, change the PS[1–0] bits only when the prescaler counter is disabled. Disable the prescaler counter by clearing TCSR[TE] of each of three timers.

PS1	PS0	Prescaler Clock Source
0	0	Internal CLK/2
0	1	TIO0
1	0	TIO1
1	1	Reserved

20–0	PL[20–0]	0	**Prescaler Preload Value** Contains the prescaler preload value, which is loaded into the prescaler counter when the counter value reaches 0 or the counter switches state from disabled to enabled. If PL[20–0] = N, then the prescaler counts N+1 source clock cycles before generating a prescaler clock pulse. Therefore, the prescaler divide factor = (preload value) + 1.

9.4.3 Timer Prescaler Count Register (TPCR)

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

23	22	21	20	19	18	17	16	15	14	13	12
			PC20	PC19	PC18	PC17	PC16	PC15	PC14	PC13	PC12

11	10	9	8	7	6	5	4	3	2	1	0
PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

Reserved bit; read as 0; write to 0 for future compatibility

Figure 9-21. Timer Prescaler Count Register (TPCR)

Table 9-2. Timer Prescaler Count Register (TPCR) Bit Definitions

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

9.4.4 Timer Control/Status Register (TCSR)

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

23	22	21	20	19	18	17	16	15	14	13	12
		TCF	TOF					PCE		DO	DI
11	10	9	8	7	6	5	4	3	2	1	0
DIR		TRM	INV	TC3	TC2	TC1	TC0		TCIE	TOIE	TE
	Reserved. Read as 0. Write to 0 for future compatibility										

Table 9-3. Timer Control/Status Register (TCSR) Bit Definitions

Bit Number	Bit Name	Reset Value	Description
23–22		0	Reserved. Write to zero for future compatibility.
21	TCF	0	**Timer Compare Flag** Indicate that the event count is complete. In timer, PWM, and watchdog modes, the TCF bit is set after (M – N + 1) events are counted. (M is the value in the compare register and N is the TLR value.) In measurement modes, the TCF bit is set when the measurement completes. Writing a one to the TCF bit clears it. A zero written to the TCF bit has no effect. The bit is also cleared when the timer compare interrupt is serviced. The TCF bit is cleared by a hardware RESET signal, a software RESET instruction, the STOP instruction, or by clearing the TCSR[TE] bit to disable the timer. NOTE: The TOF and TCF bits are cleared by a 1 written to the specific bit. To ensure that only the target bit is cleared, do not use the BSET command. The proper way to clear these bits is to write 1, using a MOVEP instruction, to the flag to be cleared and 0 to the other flag.
20	TOF	0	**Timer Overflow Flag** Indicates that a counter overflow has occurred. This bit is cleared by writing a one to the TOF bit. Writing a zero to TOF has no effect. The bit is also cleared when the timer overflow interrupt is serviced. The TOF bit is cleared by a hardware RESET signal, a software RESET instruction, the STOP instruction, or by clearing the TCSR[TE] bit to disable the timer.
19–16		0	Reserved. Write to zero for future compatibility.
15	PCE	0	**Prescaler Clock Enable** Selects the prescaler clock as the timer source clock. When PCE is cleared, the timer uses either an internal (CLK/2) signal or an external (TIO) signal as its source clock. When PCE is set, the prescaler output is the timer source clock for the counter, regardless of the timer operating mode. To ensure proper operation, the PCE bit is changed only when the timer is disabled. The PS[1–0] bits of the TPLR determine which source clock is used for the prescaler. A timer can be clocked by a prescaler clock that is derived from the TIO of another timer.
14		0	Reserved. Write to zero for future compatibility.
13	DO	0	**Data Output** The source of the TIO value when it is a data output signal. The TIO signal is a data output when the GPIO mode is enabled and DIR is set. A value written to the DO bit is written to the TIO signal. If the INV bit is set, the value of the DO bit is inverted when written to the TIO signal. When the INV bit is cleared, the value of the DO bit is written directly to the TIO signal. When GPIO mode is disabled, writing to the DO bit has no effect.

Table 9-3. Timer Control/Status Register (TCSR) Bit Definitions (continued)

Bit Number	Bit Name	Reset Value	Description
12	DI	0	**Data Input** Reflects the value of the TIO signal. If the INV bit is set, the value of the TIO signal is inverted before it is written to the DI bit. If the INV bit is cleared, the value of the TIO signal is written directly to the DI bit.
11	DIR	0	**Direction** Determines the behavior of the TIO signal when it functions as a GPIO signal. When DIR is set, the TIO signal is an output; when DIR is cleared, the TIO signal is an input. The TIO signal functions as a GPIO signal only when the TC[3–0] bits are cleared. If any of the TC[3–0] bits are set, then the GPIO function is disabled, and the DIR bit has no effect.
10		0	Reserved. Write to zero for future compatibility.
9	TRM	0	**Timer Reload Mode** Controls the counter preload operation. In timer (0–3) and watchdog (9–10) modes, the counter is preloaded with the TLR value after the TCSR[TE] bit is set and the first internal or external clock signal is received. If the TRM bit is set, the counter is reloaded each time after it reaches the value contained by the TCR. In PWM mode (7), the counter is reloaded each time counter overflow occurs. In measurement (4–5) modes, if the TRM and the TCSR[TE] bits are set, the counter is preloaded with the TLR value on each appropriate edge of the input signal. If the TRM bit is cleared, the counter operates as a free running counter and is incremented on each incoming event.
8	INV	0	**Inverter** Affects the polarity definition of the incoming signal on the TIO signal when TIO is programmed as input. It also affects the polarity of the output pulse generated on the TIO signal when TIO is programmed as output. See Table 9-4. The INV bit does not affect the polarity of the prescaler source when the TIO is input to the prescaler. NOTE: The INV bit affects both the timer and GPIO modes of operation. To ensure correct operation, change this bit only when one or both of the following conditions is true: the timer is disabled (the TCSR[TE] bit is cleared). The timer is in GPIO mode.

Table 9-3. Timer Control/Status Register (TCSR) Bit Definitions (continued)

<table>
<tr><th>Bit Number</th><th>Bit Name</th><th>Reset Value</th><th>Description</th></tr>
<tr><td>7–4</td><td>TC[3–0]</td><td>0</td><td>
Timer Control
Control the source of the timer clock, the behavior of the TIO signal, and the Timer mode of operation. Section 9.3, Operating Modes describes the timer operating modes in detail.
NOTE: To ensure proper operation, the TC[3–0] bits should be changed only when the timer is disabled (that is, when the TCSR[TE] bit is cleared)
NOTE: If the clock is external, the counter is incremented by the transitions on the TIO signal. The external clock is internally synchronized to the internal clock, and its frequency should be lower than the internal operating frequency divided by 4 (that is, CLK/4).
<table>
<tr><th colspan="4">Bit Settings</th><th colspan="4">Mode Characteristics</th></tr>
<tr><th>TC3</th><th>TC2</th><th>TC1</th><th>TC0</th><th>Mode Number</th><th>Mode Function</th><th>TIO</th><th>Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Timer and GPIO</td><td>GPIO[1]</td><td>Internal</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>Timer pulse</td><td>Output</td><td>Internal</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td><td>Timer toggle</td><td>Output</td><td>Internal</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>3</td><td>Event counter</td><td>Input</td><td>External</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>4</td><td>Input width measurement</td><td>Input</td><td>Internal</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>5</td><td>Input period measurement</td><td>Input</td><td>Internal</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>6</td><td>Capture event</td><td>Input</td><td>Internal</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>7</td><td>Pulse width modulation</td><td>Output</td><td>Internal</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>8</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>9</td><td>Watchdog pulse</td><td>Output</td><td>Internal</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>10</td><td>Watchdog Toggle</td><td>Output</td><td>Internal</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>11</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>12</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>13</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>14</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>15</td><td>Reserved</td><td>—</td><td>—</td></tr>
<tr><td colspan="8">Note 1: The GPIO function is enabled only if all of the TC[3–0] bits are 0.</td></tr>
</table>
</td></tr>
<tr><td>3</td><td></td><td>0</td><td>Reserved. Write to zero for future compatibility.</td></tr>
</table>

Table 9-3. Timer Control/Status Register (TCSR) Bit Definitions (continued)

Bit Number	Bit Name	Reset Value	Description
2	TCIE	0	**Timer Compare Interrupt Enable** Enables/disables the timer compare interrupts. When set, TCIE enables the compare interrupts. In the timer, pulse width modulation (PWM), or watchdog modes, a compare interrupt is generated after the counter value matches the value of the TCPR. The counter starts counting up from the number loaded from the TLR and if the TCPR value is M, an interrupt occurs after (M – N + 1) events, where N is the value of TLR. When cleared, the TCSR[TCIE] bit disables the compare interrupts.
1	TOIE	0	**Timer Overflow Interrupt Enable** Enables timer overflow interrupts. When set, TOIE enables overflow interrupt generation. The timer counter can hold a maximum value of $FFFFFF. When the counter value is at the maximum value and a new event causes the counter to be incremented to $000000, the timer generates an overflow interrupt. When cleared, the TOIE bit disables overflow interrupt generation.
0	TE	0	**Timer Enable** Enables/disables the timer. When set, TE enables the timer and clears the timer counter. The counter starts counting according to the mode selected by the timer control (TC[3–0]) bit values. When clear, TE bit disables the timer. NOTE: When all three timers are disabled and the signals are not in GPIO mode, all three TIO signals are tri-stated. To prevent undesired spikes on the TIO signals when you switch from tri-state into active state, these signals should be tied to the high or low signal state by pull-up or pull-down resistors.

Table 9-4. Inverter (INV) Bit Operation

Mode	TIO Programmed as Input		TIO Programmed as Output	
	INV = 0	INV = 1	INV = 0	INV = 1
0	GPIO signal on the TIO signal read directly.	GPIO signal on the TIO signal inverted.	Bit written to GPIO put on TIO signal directly.	Bit written to GPIO inverted and put on TIO signal.
1	Counter is incremented on the **rising** edge of the signal from the TIO signal.	Counter is incremented on the **falling** edge of the signal from the TIO signal.	—	—
2	Counter is incremented on the **rising** edge of the signal from the TIO signal.	Counter is incremented on the **falling** edge of the signal from the TIO signal.	Initial output put on TIO signal directly.	Initial output inverted and put on TIO signal.
3	Counter is incremented on the **rising** edge of the signal from the TIO signal.	Counter is incremented on the **falling** edge of the signal from the TIO signal.	—	—
4	Width of the **high** input pulse is measured.	Width of the **low** input pulse is measured.	—	—
5	Period is measured between the **rising** edges of the input signal.	Period is measured between the **falling** edges of the input signal.	—	—

Table 9-4. Inverter (INV) Bit Operation (continued)

Mode	TIO Programmed as Input		TIO Programmed as Output	
	INV = 0	INV = 1	INV = 0	INV = 1
6	Event is captured on the **rising** edge of the signal from the TIO signal.	Event is captured on the **falling** edge of the signal from the TIO signal.	—	—
7	—	—	Pulse generated by the timer has **positive** polarity.	Pulse generated by the timer has **negative** polarity.
9	—	—	Pulse generated by the timer has **positive** polarity.	Pulse generated by the timer has **negative** polarity.
10	—	—	Pulse generated by the timer has **positive** polarity.	Pulse generated by the timer has **negative** polarity.

9.4.5 Timer Load Register (TLR)

The TLR is a 24-bit write-only register. In all modes, the counter is preloaded with the TLR value after the TCSR[TE] bit is set and a first event occurs.

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

9.4.6 Timer Compare Register (TCPR)

The TCPR is a 24-bit read/write register that contains the value to be compared to the counter value. These two values are compared every timer clock after TCSR[TE] is set. When the values match, the timer compare flag bit is set and an interrupt is generated if interrupts are enabled (that is, the timer compare interrupt enable bit in the TCSR is set). The TCPR is ignored in measurement modes.

9.4.7 Timer Count Register (TCR)

The TCR is a 24-bit read-only register. In timer and watchdog modes, the contents of the counter can be read at any time from the TCR register. In measurement modes, the TCR is loaded with the current value of the counter on the appropriate edge of the input signal, and its value can be read to determine the width, period, or delay of the leading edge of the input signal. When the timer is in measurement mode, the TIO signal is used for the input signal.

Notes

Chapter 10
Watchdog Timer Module

10.1 Introduction

The watchdog timer (WDT) is a 16-bit timer used to help software recover from runaway code. The timer is a free-running down-counter used to assert WDT pin on underflow. Software must periodically service the watchdog timer in order to restart the count down and prevent assertion of the WDT pin. Figure 10-1 shows the watchdog block diagram.

10.2 WDT Pin

The watchdog timer pin is muxed with the TIO1 pin. The EN bit of the WCR register determines the operation of this pin (TIO1 or WDT). When this pin is configured as a hardware watchdog timer pin, it is normally pulled high. It is held high during watchdog timer operation until the watchdog timer times out. When the watchdog timer times out this pin is asserted low after a two EXTAL clock cycle delay. Following a reset of the part, this pin will de-assert high after a two EXTAL clock cycle delay. This pin is configured as a WDT pin during hardware reset.

10.3 WDT Operation

The watchdog timer is driven by the DSP's main oscillator (Fosc). Fosc is scaled by a fixed prescaler (/4096) prior to driving the 16-bit counter. The time-out period can be selected by writing to the watchdog modulus register (WMR).

Time-out = 4096 * (WMR + 1) clocks

Example #1:

Fosc = 150 MHz

Time-out = 4096 * ($00FFFF+1) = 268,435,456 clocks

Count down time = (268,435,456 clocks/150,000,000 clocks per second)

Count down time = 1.7896 seconds

Example #2:

Fosc = 100 MHz

Time-out = 4096 * ($006234+1) = 25,141 clocks

Count down time = (25,141 clocks/100,000,000 clocks per second)

Count down time = 251.41 micro seconds

When the counter reaches $000000 the WDT pin is asserted low. The WDT can be serviced by writing to the WSR register as described in Section 10.4.4, *Watchdog Service Register (WSR)*. When serviced, the counter is loaded with the reload value stored in the WMR register and continues counting down from the new value. The following overviews the WDT registers.

he counter has four registers in its programming model:

1. The watchdog control register (WCR) configures the watchdog's operation.
2. The watchdog modulus register (WMR) determines the timer modulus reload value.
3. The watchdog count register (WCNTR) provides visibility to the counter value.
4. The watchdog service register (WSR) requires a service sequence to prevent assertion of the WDT pin.

Note the watchdog timer registers are accessed via the BIU bus and thus require that the BIU be enabled. The BIU may be enabled by writing $01FFFF to the BCR register (x:$FFFFFB).

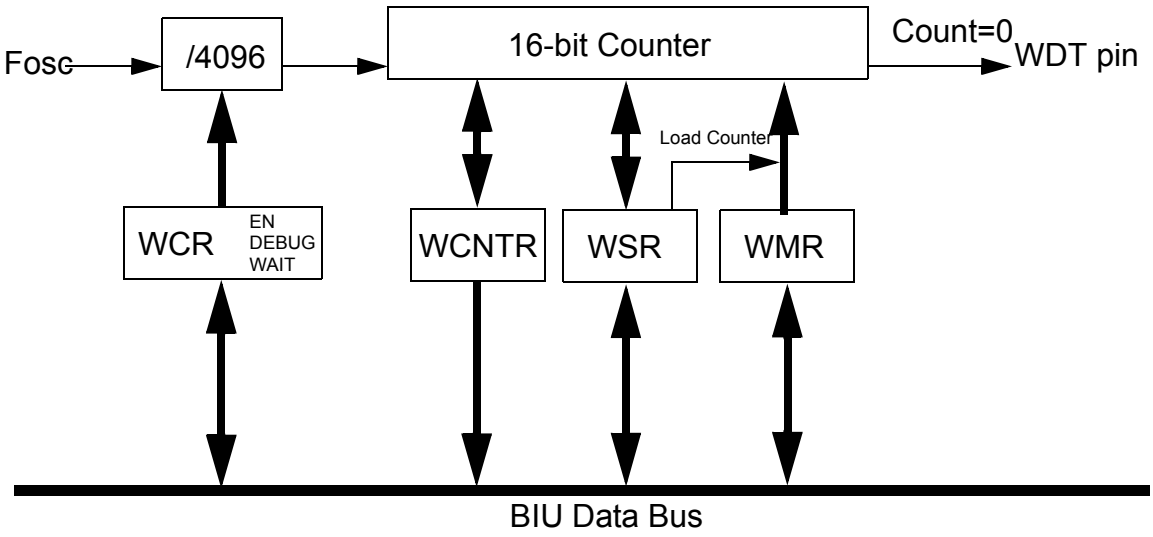


Figure 10-1. Watchdog Timer Block Diagram

10.4 Description of Registers

10.4.1 Watchdog Control Register (WCR)

The Watchdog Control Register is a 16-bit read/write register. It is a write-once register. Once written, it cannot be written again before a hardware reset except in debug mode. This register has only three used bits and the rest are reserved. The WCR Register is located at Y:$FFFFC0.

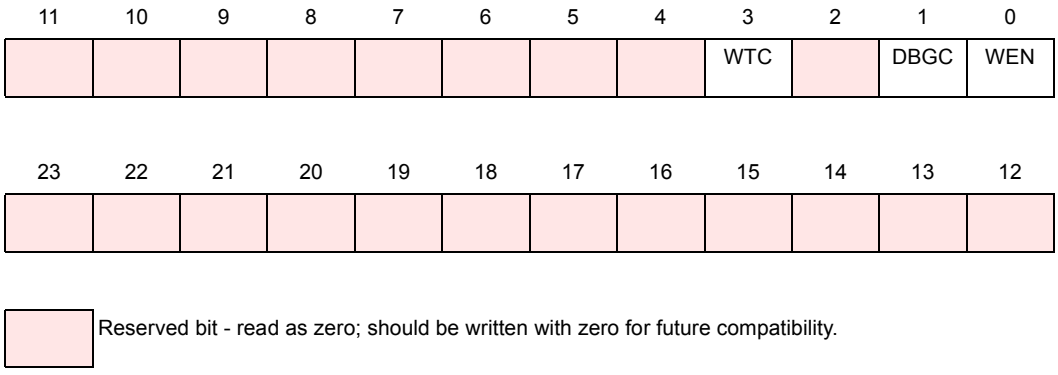


Figure 10-2. WCR Register

1. WAIT mode control bit (WTC)
   0 = Watchdog timer functions normally in WAIT mode.
   1 = Watchdog timer is stopped in WAIT mode.
2. Debug mode control bit (DBGC)
   0 = Watchdog timer functions normally in DEBUG mode.
   1 = Watchdog timer is stopped in DEBUG mode.
3. Watchdog Enable (WEN)
   0 = Watchdog timer is disabled. The TIO1/WDT pin functions as a TIO1 pin.
   1 = Watchdog timer is enabled. The TIO1/WDT pin functions as a WDT pin.

The WCR is reset by a hardware reset only and the reset value is $00000F. The WCR can be updated in the DEBUG mode and it retains the changed value after the DEBUG mode. If WCR has not been written before entering the DEBUG, then writing in DEBUG mode does not affect the WCR capability to be written once in normal mode. When DEBUG mod is exited, timer operation continues from the state it was in before entering debug mode, but any updates made in debug mode remain. If a write-once register is written for the first time in debug mode, the register is still writable when debug mode is exited.

NOTE

Changing the Debug bit from 1 to 0 during debug mode starts the watchdog timer. Changing the DEBUG bit from 0 to 1 during debug mode stops the watchdog timer.

10.4.2 Watchdog Counter & WCNTR Register

The Watchdog Count Register (WCNTR) is a read-only register. Writing to WCNTR has not effect. The WCNTR Register is located at Y:$FFFFC2.

11	10	9	8	7	6	5	4	3	2	1	0
WC11	WC10	WC9	WC8	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0

23	22	21	20	19	18	17	16	15	14	13	12
								WC15	WC14	WC13	WC12

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

Figure 10-3. WCNTR Register

The WCNTR reflects the current value of the counter. The counter is 16-bit down-counter with reset value $FFFF.

1. Reset of counter - The counter is reset asynchronously by a hardware reset.
   Reset value = $FFFF.
2. Load of counter - The counter is synchronously loaded from WMR. A 16-bit load occurs if the watchdog is serviced. On a write to WMR, the corresponding value is updated in the counter.
3. Down counting - The 16-bit counter is decremented every Fosc/4096 clock cycles. When the counter value changes from $0000 to $FFFF, the WDT pin is asserted which can only be cleared by hardware reset.

10.4.3 Watchdog Modulus Register (WMR)

The WMR is a 16-bit read/write register. This is a write-once register.The WMR Register is located at Y:$FFFFC1.

11	10	9	8	7	6	5	4	3	2	1	0
WM11	WM10	WM9	WM8	WM7	WM6	WM5	WM4	WM3	WM2	WM1	WM0

23	22	21	20	19	18	17	16	15	14	13	12
								WM15	WM14	WM13	WM12

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

Figure 10-4. WMR Register

The WMR register contains the modulus value that is reloaded into the watchdog counter by a service sequence. Once written, the WMR is not affected by further writes except in debug mode. The WMR can be written in DEBUG mode even if written-once earlier and it retains the changed value on exiting the DEBUG mode. If WMR has not been written before entering the DEBUG mode, writing in DEBUG mode does not affect the WMR capability to be written once in normal mode.

Writing to WMR immediately loads the new modulus value into the watchdog counter. The new value is also used at all subsequent reloads. Reading the WMR register returns the value in the modulus register.

A hardware reset initializes the WMR to $00FFFF. Notes

10.4.4 Watchdog Service Register (WSR)

The WSR is a 16-bit write register. This register is used to service the Watchdog timer. The WSR Register is located at Y:$FFFFC3.

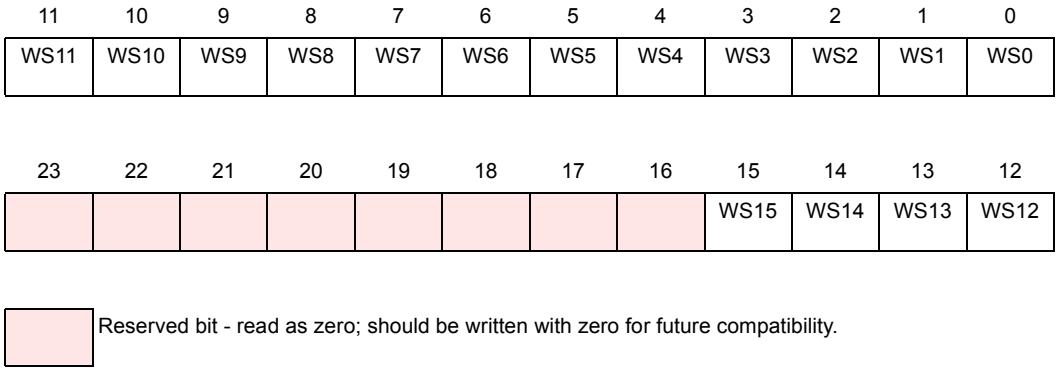


Figure 10-5. WSR Register

When the watchdog timer is enabled, the watchdog timer is serviced by writing $005555 followed by $00AAAA to the watchdog service register (WSR). If the watchdog timer is not serviced before the timeout, the watchdog timer will assert the WDT pin.

Both writes must occur in the order listed before the timeout, but any number of instructions can be executed between the two writes. However, writing any value other than $005555 or $00AAAA to the WSR resets the servicing sequence, requiring both values to be written to keep the watchdog timer from causing the WDT to assert.

10.5 Operation in Different Modes

10.5.1 WAIT Mode

The Watchdog timer function is stopped in wait mode if wcr[3] bit is set. The counter and the prescaler retain their values during wait mode. If wcr[3] is cleared, the timer function is unaffected in wait mode. All register accesses function in the normal fashion, regardless of the value of wcr[3]. Figure 10-1 shows the timer function in wait mode. Since WCR is 0x000F, replacing wait mode by doze or debug mode will make no change in timing diagram.

10.5.2 DEBUG Mode

The Watchdog timer function is stopped in debug mode if wcr[1] bit is set. The counter and the prescaler retain their values during debug mode. If wcr[1] is cleared, the timer function is unaffected in debug mode. In debug mode, the WMR and WCR can be updated like a simple read/write register. The write-once property of these registers do not apply in debug mode. All changes done in debug mode are retained. A write-once register bit that has not previously been written is still writable when debug mode is exited.

10.5.3 STOP MODE

The Fosc is assumed to be stopped in STOP mode. The watchdog timer does not function in stop mode.

Appendix A
Bootstrap Source Code

A.1 DSP56374 Bootstrap Program

```
; BOOTSTRAP CODE FOR DSP56374 Rev. 0 silicon -
; (C) Copyright 2003- Freescale Semiconductor, Inc. (formerly Motorola)
;
;
; Revision 0.0 19 Sep 2003 -
; Modified from 56371Boot.asm:
; Operation mode
;     MD:MC:MB:MA
;  2: 0  0  1  0: Jump to ROM
;  5: 0  1  0  1: Boot via SPI (slave)
;  6: 0  1  1  0: Boot via I2C with spike filter (slave)
;  7: 0  1  1  1: Boot via I2C w/o spike filter (slave)
;  9: 1  0  0  1: Boot via I2C Serial EEPROM with spike filter (master)
;  B: 1  0  1  1: Boot via SPI Serial EEPROM (master)
;  C: 1  1  0  0: Boot via GPIO Serial SPI EEPROM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

         page    132,55,0,0,0

         opt     cex,mex,mu

         section BOOTSTRAP

         XDEF    BootStrap

;;
;;;;;;;;;;;;;;;;;;;; GENERAL EQUATES ;;;;;;;;;;;;;;;;;;;;;;;;
;;
PROMADDR equ    $FF01B0          ; Starting PROM address

MA       EQU     0
MB       EQU     1
MC       EQU     2
MD       EQU     3

;;
;;;;;;;;;;;;;;;;;;;; DSP I/O REGISTERS ;;;;;;;;;;;;;;;;;;;;;;;;
;;
         ifqROM
STRAP_STARTEQU$ff0000
         else
STRAP_STARTEQU$4700
         endif
M_OGDB   EQU     $FFFFFC          ; OnCE GDB Register
OMRSave  EQU$102
M_HRX    EQU     $FFFF94          ; SHI Receive FIFO
M_HCSR   EQU     $FFFF91          ; SHI Control/Status Register
M_HCKR   EQU     $FFFF90          ; SHI Clock Control Register
HRNE     EQU     17               ; SHI FIFO Not Empty flag
HI2C     EQU     1                ; SHI I2C Enable Control Bit
HCKFR    EQU     4                ; SHI I2C Clock Freeze Control Bit
HFM0     EQU     12               ; SHI I2C Filter Mode Bit 0
HFM1     EQU     13               ; SHI I2C Filter Mode Bit 1
```

```
        ORG PH:STRAP_START          ; bootstrap code starts at $ff0000

BootStrap:

START
        movep #$0,X:M_OGDB      ; enable OnCE
        nop                     ; 5 NOP instructions, needed for test procedure
        nop
        nop
        nop
        nop
        clr a #$0,r5            ; clear a and init R5 with 0

        move omr,a
        and #>$f,a
        move #-1,m0
        add #ModeJIT,a
        move a,r0
        move p:(r0),r1
        jmp r1


;=========================================================================
; This is the routine that jumps to the internal Program ROM.
; MD:MC:MB:MA=0010
mode2
        move #>1,a
        move a,y:OMRSave
bootSA:
        move #PROMADDR,r1       ; store starting PROM address in r1
        bra <FINISH

;=========================================================================
; This is the routine that loads from SHI.
; MD:MC:MB:MA=0100 - reserved for SHI
; MD:MC:MB:MA=0101 - Bootstrap from SHI (SPI slave)
; MD:MC:MB:MA=0110 - Bootstrap from SHI (I2C slave,with spike filter HCKFR=0)
; MD:MC:MB:MA=0111 - Bootstrap from SHI (I2C slave, HCKFR=0)


; This is the routine which loads a program through the SHI port.
; The SHI operates in the slave
; mode, with the 10-word FIFO enabled, and with the HREQ pin enabled for
; receive operation. The word size for transfer is 24 bits. The SHI
; operates in the SPI or in the I2C mode, according to the bootstrap mode.
;
; The program is downloaded according to the following rules:
; 1) 3 bytes - Define the program length.
; 2) 3 bytes - Define the address to which to start loading the program to.
; 3) 3n bytes (while n is the program length defined by the first 3 bytes)
; The program words will be stored in contiguous PRAM memory locations starting
; at the specified starting address.
; After storing the program words, program execution starts from the same
; address where loading started.
mode5:
mode6:
mode7:
```

```
         move    #$A9,r1             ; prepare SHI control value in r1
; HEN=1, HI2C=0, HM1-HM0=10, HCKFR=0, HFIFO=1, HMST=0,
; HRQE1-HRQE0=01, HIDLE=0, HBIE=0, HTIE=0, HRIE1-HRIE0=00

         jclr    #MB,omr,shi_loop    ; If MD:MC:MB:MA=0101, select SPI mode

         bset    #HI2C,r1            ; otherwise select I2C mode.
         jset    #MA,omr,no_spikefltr
         bset    #HFM0,x:M_HCKR
         bset    #HFM1,x:M_HCKR
no_spikefltr
shi_loop
         movep   r1,x:M_HCSR         ; enable SHI

         jclr    #HRNE,x:M_HCSR,*    ; wait for no. of words
         movep   x:M_HRX,a0

         jclr    #HRNE,x:M_HCSR,*    ; wait for starting address
         movep   x:M_HRX,r0
         move    r0,r1

         do      a0,_LOOP2
         jclr    #HRNE,x:M_HCSR,*    ; wait for HRX not empty
         movep   x:M_HRX,p:(r0)+     ; store in Program RAM
         nop                         ; req. because of restriction
_LOOP2

         bra     <FINISH


;===============================================================================
; MD:MC:MB:MA = 1100, Boot via GPIO Serial SPI EEPROM
mode9:
         jmp     SerialEEPROMI2C     ; If MD:MC:MB:MA=1001, go to Serial EEPROM
mode11:
         jmp     SerialEEPROMSPI
mode12:
         jmp     SerialEEPROMGPIO
;===============================================================================
; MD:MC:MB:MA = 1010, Burn in mode

        ;jmp     BURN                ; If MD:MC:MB:MA=1010, go to BURN

;=========================================================================
; This is the exit handler that returns execution to normal
; expanded mode and jumps to the RESET vector.

FINISH
         andi #$0,ccr            ; Clear CCR as if RESET to 0.
         jmp (r1)                ; Then go to starting Prog addr.


;===============================================================================
; The following modes are reserved, some of which are used for internal testing
; Operation mode MD:MC:MB:MA=0000 is reserved
; Operation mode MD:MC:MB:MA=0001 is reserved
; Operation mode MD:MC:MB:MA=0011 is reserved
; Operation mode MD:MC:MB:MA=0100 is reserved
; Operation mode MD:MC:MB:MA=1000 is reserved
```

```
; Operation mode MD:MC:MB:MA=1010 is reserved
mode0:
mode1:
mode3:
mode4:
mode8:
mode10:
          bra <*
mode13:
          clr a
          move a,y:OMRSave
          jmp bootSA
mode14:
          move #>2,a
          move a,y:OMRSave
          jmp bootSA
mode15:
          move #>3,a
          move a,y:OMRSave
          jmp bootSA


;================================================================================
; Boot Mode Jump Indirect Table
ModeJIT
         DC mode0,mode1,mode2,mode3
         DC mode4,mode5,mode6,mode7
         DC mode8,mode9,mode10,mode11
         DC mode12,mode13,mode14,mode15
;================================================================================



         include  'SerialBootloader.asm'

;================================================================
; This code fills the unused bootstrap rom locations with their address

 ;               dup STRAP_START+$200-*
 ;               dc *
 ;               endm

;================================================================
; Serial EEPROM Boot Loader


;================================================================
; This code fills the unused bootstrap rom locations with their address

          dup $FF01B0-*
          dc *
          endm

;================================================================

          endsec

          end
```

A.2 Using The Serial EEPROM Boot Mode

There is a boot-up mode included which allows the downloading of code directly from a serial EEPROM. This can be used in applications where there are no microcontrollers available as the DSP itself is used as the master clock source.

Configuration of the MOD pins

To boot up in these modew, the MOD pins should be as follows: -

Mode	MODD	MODC	MODB	MODA	Reset Vector
9	1	0	0	1	$FF0000
11	1	0	1	1	$FF0000
12	1	1	0	0	$FF0000

See table 4.2 for the other boot modes available.

Constraints of EEPROM Used

To use this mode, the following constraints must be followed: -

* An I2C EEPROM must have the device ID 1010

Below are listed some suggestions for suitable devices: -

ST Microelectronics M24128 (128Kbit Serial I2C Bus EEPROM)
ST Microelectronics M24256 (256Kbit Serial I2C Bus EEPROM)
ST Microelectronics M24512 (512Kbit Serial I2C Bus EEPROM)
ST Microelectronics M24M01(1Mbit Serial I2C Bus EEPROM)

Format used for external EEPROM

The DSP expects to receive the data in the following format: -

Assuming the EEPROM address starts at 0: -

EEPROM Address	Byte expected at DSP	Example
0	PstartAddress2	00
1	PstartAddress1	04
2	PstartAddress0	00
3	XstartAddress2	00
4	XstartAddress1	05
5	XstartAddress0	00
6	YstartAddress2	00
7	YstartAddress1	06
8	YstartAddress0	00
9	PdataLength2	67
10	PdataLength1	45
11	PdataLength0	00
12	XdataLength2	21
13	XdataLength1	35
14	XdataLength0	43
15	YdataLength2	56
16	YdataLength1	64
17	YdataLength0	24

```
18                data in P, X and Y order -
19                must make up whole 24 bit words
-
-
-
-                 PstartAddress2 00
-                 PstartAddress1 04
-                 PstartAddress0 00
                  (final word is start address of rogram to run)


The above example would store: $004567 words from P:$400
                               $433521 words from X:$500
                               $246456 words from Y:$600
And will start running from P:$400 once all of the data has been downloaded.
```

Format used in EEPROM

The format that should be used to store the data on the DSP is Intel HEX format, an example and explanation of this is given below: -

```
:10008000AF5F67F0602703E0322CFA92007780C361
:1000900089001C6B7EA7CA9200FE10D2AA00477D81
:0B00A00080FA92006F3600C3A00076CB
:00000001FF
Now look at the top line...
         * The first character (:) indicates the start of a record.
         * The next two characters indicate the record length (10h in this case).
         * The next four characters give the load address (0080h in this case).
         * The next two characters indicate the record type (see below).
         * Then we have the actual data.
         * The last two characters are a checksum (sum of all bytes + checksum = 00).

The last line of the file is special, and will always look like that above.

Record types:
         * 00 - Data record
         * 01 - End of file record
         * 02 - Extended segment address record
         * 03 - Start segment address record
         * 04 - Extended linear address record
         * 05 - Start linear address record
```

Listing of the boot code

Below is the boot code used in this mode: -

```
         TITLE 'SerialBootloader';Name of Program
;BootROM equ      $FFFF00   ;ROM boot area.
HCKR     equ      $FFFF90   ;SHI Clock Control Register.
HCSR     equ      $FFFF91   ;SHI Control/Status Register.
HRX      EQU      $FFFF94   ;SHI Receive Data FIFO
HTX      EQU      $FFFF93   ;SHI Transmit Data Register
         if       qROM
         define   mHPORT    "x"       ;need to change to X for 374
```

```
PCRH     EQU      $FFFF9a   ; Port H Control Register, X
PRRH     EQU      $FFFF99   ; Port H Direction Register, X
PDRH     EQU      $FFFF98   ; Port H Data Register,X
PH3_CLK  EQU      $3        ; SerROM Clock
         else
         define   mHPORT    "y"       ;need to change to X for 374

PCRH     EQU      $FFFF9F

PRRH     EQU      $FFFF9E

PDRH     EQU      $FFFF9D
PH3_CLK  EQU      $5        ; SerROM Clock
         endif
;!!!!!PCRH PRRH PDRH should be defined as comment for 374

PH0_CS   EQU      $0      ; SerROM Chip Select
PH1_SO   EQU      $1      ; SerROM data out
PH2_SI   EQU      $2      ; SerROM data in
;
kPCRH    EQU      (0<<PH0_CS)|(1<<PH1_SO)|(0<<PH2_SI)|(0<<PH3_CLK)
kPRRH    EQU      (1<<PH0_CS)|(0<<PH1_SO)|(1<<PH2_SI)|(1<<PH3_CLK)
kPCRH_SHI EQU     (0<<PH0_CS)
kPRRH_SHI EQU     (1<<PH0_CS)
IN_READ  EQU      $03
         org      ph:
SerialEEPROMSPI:
        ;*** Reset SHI ***
         movep    #$0,X:HCSR

         ;*** Set clock Rate ***
         ; CPOL = 0, CPHA = 0
         ; Set to /8 (i.e., 25MHz/8 = 3125000
         movep    #$0,x:HCKR

         ;*** Set HCSR ***
         brclr    #15,X:HCSR,*
         movep    #$8041,x:HCSR
         movep    #kPCRH_SHI,mHPORT:PCRH
         movep    #kPRRH_SHI,mHPORT:PRRH
         bset     #PH0_CS,mHPORT:PDRH     ; set CS to High
         dor      #256,_delay
         nop
         nop
         nop
         nop
_delay
        ; Send Read Command
         bclr    #PH0_CS,mHPORT:PDRH ; Set CS to low
         ;*** Send Read Command***
         ; Read Command = 3
         movep    #$30000,x:HTX
         brclr    #19,x:HCSR,*
         movep    x:HRX,a1
         ;*** Send Address Key ***
         brclr    #15,x:HCSR,*
         movep    #$0,x:HTX       ; Start from MSB:0
         brclr    #19,x:HCSR,*
         movep    x:HRX,a1
```

```
          brclr    #15,x:HCSR,*
          movep    #$0,x:HTX       ; 0
          brclr    #19,x:HCSR,*
          movep    x:HRX,a1

          brclr    #15,x:HCSR,*
          movep    #$0,x:HTX       ; LSB: 0
          brclr    #19,x:HCSR,*
          movep    x:HRX,a1

          brclr    #15,x:HCSR,*
          movep    #$0,x:HTX

          jsr      GetWordSPI
          move     a1,r0           ; 0: Start Address of P Memory
          jsr      GetWordSPI
          move     a1,r1           ; 1: Start Address of X Memory
          jsr      GetWordSPI
          move     a1,r2           ; 2: Start Address of Y Memory
          jsr      GetWordSPI
          move     a1,y0           ; 3: P Data length
          jsr      GetWordSPI
          move     a1,x1           ; 4: X Data length
          jsr      GetWordSPI
          move     a1,y1           ; 5: Y Data length
          ;*** Get Data of P Memory ***
          do       y0,_endofP
          jsr      GetWordSPI
          move     a1,p:(r0)+
          nop
          nop
          nop
_endofP
         ;*** Get Data of X Memory ***
          do       x1,_endofX
          jsr      GetWordSPI
          move     a1,x:(r1)+
          nop
          nop
          nop
_endofX
         ;*** Get Data of Y Memory ***
          do       y1,_endofY
          jsr      GetWordSPI
          move     a1,y:(r2)+
          nop
          nop
          nop
_endofY
          jsr      GetWordSPI
          move     a1,r0
          jmp      r0

         GLOBAL    GetWordSPI
GetWordSPI:

          brclr    #19,x:HCSR,*
          move     x:HRX,a1
```

```
        tfr     a,b

        brclr   #15,x:HCSR,*
        move    a1,x:HTX


GetByte1SPI:

        brclr#19,x:HCSR,*
        move    x:HRX,a1
        lsr     #8,a
        add     a,b

        brclr#15,x:HCSR,*
        movea1,x:HTX


GetByte0SPI:


        brclr   #19,x:HCSR,*
        move    x:HRX,a1
        lsr     #16,a
        add     a,b

        move    b1,a1

        brclr   #15,x:HCSR,*
        move    a1,x:HTX

        rts

SerialEEPROMI2C:
Start:              ;Start of program

;Need to initialise SHI to operate in I2C master mode first...

; Need to generate the slave address 1010 for ST M24256/M24128 serial EEPROM
; This is sent as 1010 0001 (0001 to signify reading from EEPROM)
; 1 - set up SHI for I2C master mode + program appropriate clock rate
; 2 - Check HTX is empty i.e., HTDE = 1
; 3 - set HIDLE bit
; 4 - write slave address (1010 0001 0000 0000 0000 0000) to HTX register
; 5 - this causes a stop event, start event and the 8 MSBs of data to be TXed
; 6 - slave device should transmit an ack bit on reception of it's device ID
; 7 - slave device should continue to transmit data bytes which the master DSP
;     will acknowledge
; 8 - signal no more data in EEPROM with $55 $55 $55 sequence
; 9 - when master DSP sees $55 $55 $55, it will indicate end of recieve by
;     setting the HIDLE bit

        bsr     SHIReset


       movep    #$FF0000,X:HTX; Generate stop event if error
        rep     #2560
        nop
        rep     #2560
        nop
```

```
        bsr     SHIReset


        ; *** Write slave address to HTX register (will cause HIDLE bit to clear) ***
        ; Address $A1 applicable to ST M24256 / M24128 serial EEPROMs

        movep   #$A00000,X:HTX
        rep     #256
        nop
       ; Send Byte Address 1
        brclr   #15,x:HCSR,*
        movep   #0,x:HTX
        rep     #256
        nop
       ; Send Byte Address 2
        brclr   #15,x:HCSR,*
        movep   #0,x:HTX
        rep     #256
        nop
        brclr   #15,x:HCSR,*
        movep   #$008243,X:HCSR


        rep     #256
        nop

        movep   #$A10000,X:HTX
        ; FORMAT AS FOLLOWS

;       Example, 8 bit EEPROM starting at 0
;
;       0       - PstartAddress2   e.g.  00
;       1       - PstartAddress1         04
;       2       - PstartAddress0         00
;       3       - XstartAddress2         00
;       4       - XstartAddress1         05
;       5       - XstartAddress0         00
;       6       - YstartAddress2         00
;       7       - YstartAddress1         06
;       8       - YstartAddress0         00
;       9       - PdataLength2           67
;       10      - PdataLength1           45
;       11      - PdataLength0           00
;       12      - XdataLength2           21
;       13      - XdataLength1           35
;       14      - XdataLength0           43
;       15      - YdataLength2           56
;       16      - YdataLength1           64
;       17      - YdataLength0           24
;       18      - data in P, X and Y order - must make up whole 24 bit words
;       .
;       .
;       .
;       .
;       .
;       .       - PstartAddress2         00
;       .       - PstartAddress1         04
;       .       - PstartAddress0         00-final word is start address of program to run
```

```
; Example would store $004567 words from P:$400
;                           $433521 words from X:$500
;                           $246456 words from Y:$600
; Will start running from P:$400


         ;'GetWord' will return word in x0

;*** Get Header Data from EEPROM ***

         jsr      GetWord      ;Get P address

         move     a1,r0        ;Pointer to P memory

         jsr      GetWord      ;Get X address

         move     a1,r1        ;Pointer to X memory

         jsr      GetWord      ;Get Y address

         move     a1,r2        ;Pointer to Y memory

         jsr      GetWord      ;Get P length

         move     a1,r4        ;Store P length

         jsr      GetWord      ;Get X length

         move     a1,r5        ;Store X length

         jsr      GetWord      ;Get Y length

         move     a1,r6        ;Store Y length

;************************************

;*** Get data from EPROM ***

         ;Get length of P data

         do       r4,EndofP

         jsr      GetWord

         move     a1,p:(r0)+
         nop
         nop
         nop

EndofP:

         ;Get length of X data

         do       r5,EndofX

         jsr      GetWord

         move     a1,x:(r1)+
         nop
```

```
        nop
        nop

EndofX:

        ;Get length of Y data

        do      r6,EndofY

        jsr     GetWord

        move    a1,y:(r2)+
        nop
        nop
        nop

EndofY:


;*** Get P Start Address ***

        jsr     GetWord  ;Get final word - start address

        movep   #$008243,X:HCSR;Set HIDLE bit high again to terminate data receive

        move    a1,r7    ;P start address

        jmp     r7           ;Start running program

;*******END OF BOOTLOADER********

; *** Get data bytes from HRX and process ***
       GLOBAL   GetWord
GetWord:
        brset   #19,X:HCSR,ReadByte2  ;Read byte received or wait until FIFO full
        bra     GetWord

ReadByte2:
        move    x:HRX,a1            ;Move byte 2 into y1
        asr     #16,a,a
        move    a1,y1               ;Shift 2 MSBs to 2 LSBs
getByte1:
        brset   #19,X:HCSR,ReadByte1  ;Read byte received or wait until FIFO full
        bra     getByte1

ReadByte1:
        move    x:HRX,a1          ;Move byte 1 into y0
        asr     #16,a,a
        move    a1,y0             ;Shift 2 MSBs to 2 LSBs

getByte0:
        brset   #19,X:HCSR,ReadByte0  ;Read byte received or wait until FIFO full
        bra     getByte0

ReadByte0:
        move    x:HRX,a1          ;Move byte 0 into x1
        asr     #16,a,a
        move    a1,x1             ;Shift 2 MSBs to 2 LSBs
```

```
; Routine which takes 3 8-bit values
; from y1,y0,x1 and makes
; up a 24-bit value in b1
; Format is LSB-MSB-USB

MakeWord:
         move     y1,b2              ;Move byte into b2 first
         asr      #8,b,b             ;Then shift right 8 and repeat
         move     y0,b2              ;Move byte into b2 first
         asr      #8,b,b             ;Then shift right 8 and repeat
         move     x1,b2              ;Move byte into b2 first
         asr      #8,b,b             ;Then shift right 8 and repeat
EndofWord:                           ;Result stored in b1
         move     b1,a1              ;Store in a1
         rts                         ;Return from subroutine


;*************************************************************************
SerialEEPROMGPIO:
         movep    #kPCRH,mHPORT:PCRH
         movep   #kPRRH,mHPORT:PRRH
         bset    #PH0_CS,mHPORT:PDRH                ; set CS to High
         bset    #PH3_CLK,mHPORT:PDRH               ; set CLK to High
        ; Send Read Command
         bclr    #PH0_CS,mHPORT:PDRH                ; Set CS to low
         move    #IN_READ,a2
         move    #8,n1
         bsr     TransmitData
        ; Send Address
         clr     a
         move    #24,n1
         bsr     TransmitData
        ; Begin to Read EEPROM
         bsr     ReadData
         move    a1,r0                          ; Get P Start Address
         bsr     ReadData
         move    a1,r1                          ; Get X Start Address
         bsr     ReadData
         move    a1,r2                          ; Get Y Start Address
         bsr     ReadData
         move    a1,y0                          ; Get P Data Length
         bsr     ReadData
         move    a1,x1                          ; Get X Data Length
         bsr     ReadData
         move    a1,y1                          ; Get Y Data Length
         do      y0,_endofPGPIO
         bsr     ReadData
         move    a1,p:(r0)+
         nop
         nop
         nop
_endofPGPIO
         do      x1,_endofXGPIO
         bsr     ReadData
         move    a1,x:(r1)+
         nop
         nop
         nop
_endofXGPIO
```

```
          do       y1,_endofYGPIO
          bsr      ReadData
          move     a1,y:(r2)+
          nop
          nop
          nop
_endofYGPIO
          bsr      ReadData
          move     a1,r0
          jmp      r0
         GLOBAL  TransmitData
TransmitData
          do       n1,_endofTransGPIO
          asl      a
          bcc      _Send0s
_Send1s
          bset     #PH2_SI,mHPORT:PDRH
          bra      _SendClock
_Send0s
          bclr     #PH2_SI,mHPORT:PDRH
_SendClock
          bclr     #PH3_CLK,mHPORT:PDRH
          nop
          nop
          bset     #PH3_CLK,mHPORT:PDRH
_endofTransGPIO
          rts
         GLOBAL    ReadData
ReadData
          bclr     #PH3_CLK,mHPORT:PDRH
          do       #24,_WordRead
          bset     #PH3_CLK,mHPORT:PDRH
          movep    mHPORT:PDRH,x0
          nop
          bclr     #PH3_CLK,mHPORT:PDRH
          bset     #PH1_SO,x0
          rol      a1
          nop
          nop
_WordRead
          rts

SHIReset
          ;*** Reset SHI ***
          movep    #$0,X:HCSR

          ;*** Set clock Rate ***
          ; Set to /64 for max crystal value (i.e., 25MHz/(8*8) = 390625
          ; Enable wide spike filter
          movep    #$003040,X:HCKR

          ;*** Check HTX is empty i.e., HTDE=1 ***
          brclr    #15,X:HCSR,*

          move     #$008243,X:HCSR
          rep      #256
          nop

          rts
```

Appendix B
Equates

```
; Note: Not all peripherals listed are applicable for all 37x derivatives. Some peripherals
; and register bits may be listed that are not applicable to a given 37x derivative.


;*******************************************************************************
;
;
;     EQUATES for DSP56374 interrupts
;
;     Initial update: July 2, 2003
;
;
;
;*******************************************************************************

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

intequ  ident   1,0

    if    @DEF(I_VEC)       ;leave user definition as is.
    else
I_VEC    equ    $0
    endif

;------------------------------------------------------------------------
;
; Non-Maskable interrupts
;
;------------------------------------------------------------------------

I_RESET  EQU  I_VEC+$00   ; Hardware RESET
I_STACK  EQU  I_VEC+$02   ; Stack Error
I_ILL    EQU  I_VEC+$04   ; Illegal Instruction
I_IINST  EQU  I_VEC+$04   ; Illegal Instruction
I_DBG    EQU  I_VEC+$06   ; Debug Request
I_TRAP   EQU  I_VEC+$08   ; Trap
I_NMI    EQU  I_VEC+$0A   ; Non Maskable Interrupt

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

; Interrupt Request Pins

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

I_IRQA   EQU  I_VEC+$10   ; IRQA
I_IRQB   EQU  I_VEC+$12   ; IRQB
I_IRQC   EQU  I_VEC+$14   ; IRQC
I_IRQD   EQU  I_VEC+$16   ; IRQD
;------------------------------------------------------------------------

; DMA Interrupts
```

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

I_DMA0   EQU  I_VEC+$18   ; DMA Channel 0
I_DMA1   EQU  I_VEC+$1A   ; DMA Channel 1
I_DMA2   EQU  I_VEC+$1C   ; DMA Channel 2
I_DMA3   EQU  I_VEC+$1E   ; DMA Channel 3
I_DMA4   EQU  I_VEC+$20   ; DMA Channel 4
I_DMA5   EQU  I_VEC+$22   ; DMA Channel 5

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

; DAX Interrupts

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

I_DAXTUE  EQU  I_VEC+$28   ; DAX Underrun Error
I_DAXBLK  EQU  I_VEC+$2A   ; DAX Block Transferred
I_DAXTD   EQU  I_VEC+$2E   ; DAX Audio Data Empty

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

; ESAI Interrupts

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

I_ESAIRD   EQU  I_VEC+$30   ; ESAI Receive Data
I_ESAIRED  EQU  I_VEC+$32   ; ESAI Receive Even Data
I_ESAIRDE  EQU  I_VEC+$34   ; ESAI Receive Data With Exception Status
I_ESAIRLS  EQU  I_VEC+$36   ; ESAI Receive Last Slot
I_ESAITD   EQU  I_VEC+$38   ; ESAI Transmit Data
I_ESAITED  EQU  I_VEC+$3A   ; ESAI Transmit Even Data
I_ESAITDE  EQU  I_VEC+$3C   ; ESAI Transmit Data With Exception Status
I_ESAITLS  EQU  I_VEC+$3E   ; ESAI Transmit Last Slot

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

; SHI Interrupts

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

I_SHITD    EQU  I_VEC+$40   ; SHI Transmit Data
I_SHITUE   EQU  I_VEC+$42   ; SHI Transmit Underrun Error
I_SHIRNE   EQU  I_VEC+$44   ; SHI Receive FIFO Not Empty
I_SHIRFF   EQU  I_VEC+$48   ; SHI Receive FIFO Full
I_SHIROE   EQU  I_VEC+$4A   ; SHI Receive Overrun Error
I_SHIBER   EQU  I_VEC+$4C   ; SHI Bus Error



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

; Timer Interrupts

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

I_TIM0C  EQU  I_VEC+$54   ; TIMER 0 compare
I_TIM0OF EQU  I_VEC+$56   ; TIMER 0 overflow
I_TIM1C  EQU  I_VEC+$58   ; TIMER 1 compare
I_TIM1OF EQU  I_VEC+$5A   ; TIMER 1 overflow
```

```
I_TIM2C  EQU  I_VEC+$5C   ; TIMER 2 compare
I_TIM2OF EQU  I_VEC+$5E   ; TIMER 2 overflow


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

; EFCOP Interrupts

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

I_EFCOPIBE  EQU  I_VEC+$68   ; EFCOP Input Buffer Empty
I_EFCOPOBF  EQU  I_VEC+$6A   ; EFCOP Output Buffer Full


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

; ESAI_1 Interrupts

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

I_ESAI1RD   EQU  I_VEC+$70   ; ESAI_1 Receive Data
I_ESAI1RED  EQU  I_VEC+$72   ; ESAI_1 Receive Even Data
I_ESAI1RDE  EQU  I_VEC+$74   ; ESAI_1 Receive Data With Exception Status
I_ESAI1RLS  EQU  I_VEC+$76   ; ESAI_1 Receive Last Slot
I_ESAI1TD   EQU  I_VEC+$78   ; ESAI_1 Transmit Data
I_ESAI1TED  EQU  I_VEC+$7A   ; ESAI_1 Transmit Even Data
I_ESAI1TDE  EQU  I_VEC+$7C   ; ESAI_1 Transmit Data With Exception Status
I_ESAI1TLS  EQU  I_VEC+$7E   ; ESAI_1 Transmit Last Slot

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

; INTERRUPT ENDING ADDRESS

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

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

;------------------ end of intequ.asm -----------------------




;*****************************************************************************
;
;     EQUATES for DSP56374 I/O registers and ports
;     Last update: July 2, 2003
;
;*****************************************************************************

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

ioequ   ident   1,0

;-------------------------------------------------------------------------
;
;       EQUATES for I/O Port Programming
;
```

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

;       Register Addresses

M_PCRC   EQU     $FFFFBF         ; X space: Port C Control Register
M_PRRC   EQU     $FFFFBE         ; X space: Port C Direction Register
M_PDRC   EQU     $FFFFBD         ; X space: Port C GPIO Data Register
M_PCRD   EQU     $FFFFD7         ; X space: Port D Control register
M_PRRD   EQU     $FFFFD6         ; X space: Port D Direction Data Register
M_PDRD   EQU     $FFFFD5         ; X space: Port D GPIO Data Register
M_PCRE   EQU     $FFFF9F       ; Y space: Port E Control register
M_PRRE   EQU     $FFFF9E       ; Y space: Port E Direction Data Register
M_PDRE   EQU     $FFFF9D       ; Y space: Port E GPIO Data Register
M_PDRG   EQU     $FFFFF8       ; Y space: Port G Data Register
M_PRRG   EQU     $FFFFF9       ; Y space: Port G Data Direction Register
M_PCRG   EQU     $FFFFFA       ; Y space: Port G Control Register

M_OGDB   EQU     $FFFFFC         ; X space: OnCE GDB Register

;---------------------------------------------------------------------
;
;       EQUATES for Exception Processing
;
;---------------------------------------------------------------------
;       Register Addresses

M_IPRC   EQU     $FFFFFF         ; X space: Interrupt Priority Register Core
M_IPRP   EQU     $FFFFFE         ; X space: Interrupt Priority Register Peripheral

;       Interrupt Priority Register Core (IPRC)

M_IAL    EQU     $7              ; IRQA Mode Mask
M_IAL0   EQU     0               ; IRQA Mode Interrupt Priority Level (low)
M_IAL1   EQU     1               ; IRQA Mode Interrupt Priority Level (high)
M_IAL2   EQU     2               ; IRQA Mode Trigger Mode
M_IBL    EQU     $38             ; IRQB Mode Mask
M_IBL0   EQU     3               ; IRQB Mode Interrupt Priority Level (low)
M_IBL1   EQU     4               ; IRQB Mode Interrupt Priority Level (high)
M_IBL2   EQU     5               ; IRQB Mode Trigger Mode
M_ICL    EQU     $1C0            ; IRQC Mode Mask
M_ICL0   EQU     6               ; IRQC Mode Interrupt Priority Level (low)
M_ICL1   EQU     7               ; IRQC Mode Interrupt Priority Level (high)
M_ICL2   EQU     8               ; IRQC Mode Trigger Mode
M_IDL    EQU     $E00            ; IRQD Mode Mask
M_IDL0   EQU     9               ; IRQD Mode Interrupt Priority Level (low)
M_IDL1   EQU     10              ; IRQD Mode Interrupt Priority Level (high)
M_IDL2   EQU     11              ; IRQD Mode Trigger Mode
M_D0L    EQU     $3000           ; DMA0 Interrupt priority Level Mask
M_D0L0   EQU     12              ; DMA0 Interrupt Priority Level (low)
M_D0L1   EQU     13              ; DMA0 Interrupt Priority Level (high)
M_D1L    EQU     $C000           ; DMA1 Interrupt Priority Level Mask
M_D1L0   EQU     14              ; DMA1 Interrupt Priority Level (low)
M_D1L1   EQU     15              ; DMA1 Interrupt Priority Level (high)
M_D2L    EQU     $30000          ; DMA2 Interrupt priority Level Mask
M_D2L0   EQU     16              ; DMA2 Interrupt Priority Level (low)
M_D2L1   EQU     17              ; DMA2 Interrupt Priority Level (high)
M_D3L    EQU     $C0000          ; DMA3 Interrupt Priority Level Mask
M_D3L0   EQU     18              ; DMA3 Interrupt Priority Level (low)
M_D3L1   EQU     19              ; DMA3 Interrupt Priority Level (high)
```

```
M_D4L    EQU     $300000         ; DMA4 Interrupt priority Level Mask
M_D4L0   EQU     20              ; DMA4 Interrupt Priority Level (low)
M_D4L1   EQU     21              ; DMA4 Interrupt Priority Level (high)
M_D5L    EQU     $C00000         ; DMA5 Interrupt priority Level Mask
M_D5L0   EQU     22              ; DMA5 Interrupt Priority Level (low)
M_D5L1   EQU     23              ; DMA5 Interrupt Priority Level (high)

;       Interrupt Priority Register Peripheral (IPRP)

M_ESL    EQU     $3              ; ESAI Interrupt Priority Level Mask
M_ESL0   EQU     0               ; ESAI Interrupt Priority Level (low)
M_ESL1   EQU     1               ; ESAI Interrupt Priority Level (high)
M_SHL    EQU     $C              ; SHI Interrupt Priority Level Mask
M_SHL0   EQU     2               ; SHI Interrupt Priority Level (low)
M_SHL1   EQU     3               ; SHI Interrupt Priority Level (high)
M_DAL    EQU     $C0             ; DAX Interrupt Priority Level  Mask
M_DAL0   EQU     6               ; DAX Interrupt Priority Level  (low)
M_DAL1   EQU     7               ; DAX Interrupt Priority Level  (high)
M_TAL    EQU     $300            ; Timer Interrupt Priority Level Mask
M_TAL0   EQU     8               ; Timer Interrupt Priority Level (low)
M_TAL1   EQU     9               ; Timer Interrupt Priority Level (high)
M_ES1L   EQU     $C00  ; ESAI_1 Interrupt Priority Level Mask
M_ESL10  EQU     10              ; ESAI_1 Interrupt Priority Level (low)
M_ESL11  EQU     11              ; ESAI_1 Interrupt Priority Level (high)
M_EFC    EQU     $30000  ; EFCOP Interrupt Priority Level Mask
M_EFC0   EQU     16              ; EFCOP Interrupt Priority Level (low)
M_EFC1   EQU     17              ; EFCOP Interrupt Priority Level (high)



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

;       Register Addresses Of DMA

M_DSTR   EQU     $FFFFF4         ; X space: DMA Status Register
M_DOR0   EQU     $FFFFF3         ; X space: DMA Offset Register 0
M_DOR1   EQU     $FFFFF2         ; X space: DMA Offset Register 1
M_DOR2   EQU     $FFFFF1         ; X space: DMA Offset Register 2
M_DOR3   EQU     $FFFFF0         ; X space: DMA Offset Register 3

;       Register Addresses Of DMA0

M_DSR0   EQU     $FFFFEF         ; X space: DMA0 Source Address Register
M_DDR0   EQU     $FFFFEE         ; X space: DMA0 Destination Address Register
M_DCO0   EQU     $FFFFED         ; X space: DMA0 Counter
M_DCR0   EQU     $FFFFEC         ; X space: DMA0 Control Register

;       Register Addresses Of DMA1

M_DSR1   EQU     $FFFFEB         ; X space: DMA1 Source Address Register
M_DDR1   EQU     $FFFFEA         ; X space: DMA1 Destination Address Register
M_DCO1   EQU     $FFFFE9         ; X space: DMA1 Counter
M_DCR1   EQU     $FFFFE8         ; X space: DMA1 Control Register
```

```
;       Register Addresses Of DMA2

M_DSR2   EQU     $FFFFE7         ; X space: DMA2 Source Address Register
M_DDR2   EQU     $FFFFE6         ; X space: DMA2 Destination Address Register
M_DCO2   EQU     $FFFFE5         ; X space: DMA2 Counter
M_DCR2   EQU     $FFFFE4         ; X space: DMA2 Control Register


;       Register Addresses Of DMA3

M_DSR3   EQU     $FFFFE3         ; X space: DMA3 Source Address Register
M_DDR3   EQU     $FFFFE2         ; X space: DMA3 Destination Address Register
M_DCO3   EQU     $FFFFE1         ; X space: DMA3 Counter
M_DCR3   EQU     $FFFFE0         ; X space: DMA3 Control Register
;       Register Addresses Of DMA4

M_DSR4   EQU     $FFFFDF         ; X space: DMA4 Source Address Register
M_DDR4   EQU     $FFFFDE         ; X space: DMA4 Destination Address Register
M_DCO4   EQU     $FFFFDD         ; X space: DMA4 Counter
M_DCR4   EQU     $FFFFDC         ; X space: DMA4 Control Register

;       Register Addresses Of DMA5

M_DSR5   EQU     $FFFFDB         ; X space: DMA5 Source Address Register
M_DDR5   EQU     $FFFFDA         ; X space: DMA5 Destination Address Register
M_DCO5   EQU     $FFFFD9         ; X space: DMA5 Counter
M_DCR5   EQU     $FFFFD8         ; X space: DMA5 Control Register

;       DMA Control Register

M_DSS    EQU     $3              ; DMA Source Space Mask (DSS0-Dss1)
M_DSS0   EQU     0               ; DMA Source Memory space 0
M_DSS1   EQU     1               ; DMA Source Memory space 1
M_DDS    EQU     $C              ; DMA Destination Space Mask (DDS-DDS1)
M_DDS0   EQU     2               ; DMA Destination Memory Space 0
M_DDS1   EQU     3               ; DMA Destination Memory Space 1
M_DAM    EQU     $3f0            ; DMA Address Mode Mask (DAM5-DAM0)
M_DAM0   EQU     4               ; DMA Address Mode 0
M_DAM1   EQU     5               ; DMA Address Mode 1
M_DAM2   EQU     6               ; DMA Address Mode 2
M_DAM3   EQU     7               ; DMA Address Mode 3
M_DAM4   EQU     8               ; DMA Address Mode 4
M_DAM5   EQU     9               ; DMA Address Mode 5
M_D3D    EQU     10              ; DMA Three Dimensional Mode
M_DRS    EQU     $F800           ; DMA Request Source Mask (DRS0-DRS4)
M_DRS0   EQU     11              ;DMA Request Source bit 0
M_DRS1   EQU     12              ;DMA Request Source bit 1
M_DRS2   EQU     13              ;DMA Request Source bit 2
M_DRS3   EQU     14              ;DMA Request Source bit 3
M_DRS4   EQU     15              ;DMA Request Source bit 4
M_DCON   EQU     16              ; DMA Continuous Mode
M_DPR    EQU     $60000          ; DMA Channel Priority
M_DPR0   EQU     17              ; DMA Channel Priority Level (low)
M_DPR1   EQU     18              ; DMA Channel Priority Level (high)
M_DTM    EQU     $380000         ; DMA Transfer Mode Mask (DTM2-DTM0)
M_DTM0   EQU     19              ; DMA Transfer Mode 0
M_DTM1   EQU     20              ; DMA Transfer Mode 1
M_DTM2   EQU     21              ; DMA Transfer Mode 2
M_DIE    EQU     22              ; DMA Interrupt Enable bit
```

```
M_DE     EQU     23              ; DMA Channel Enable bit

;       DMA Status Register

M_DTD    EQU     $3F             ; Channel Transfer Done Status MASK (DTD0-DTD5)
M_DTD0   EQU     0               ; DMA Channel Transfer Done Status 0
M_DTD1   EQU     1               ; DMA Channel Transfer Done Status 1
M_DTD2   EQU     2               ; DMA Channel Transfer Done Status 2
M_DTD3   EQU     3               ; DMA Channel Transfer Done Status 3
M_DTD4   EQU     4               ; DMA Channel Transfer Done Status 4
M_DTD5   EQU     5               ; DMA Channel Transfer Done Status 5
M_DACT   EQU     8               ; DMA Active State
M_DCH    EQU     $E00            ; DMA Active Channel Mask (DCH0-DCH2)
M_DCH0   EQU     9               ; DMA Active Channel 0
M_DCH1   EQU     10              ; DMA Active Channel 1
M_DCH2   EQU     11              ; DMA Active Channel 2



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

;       Register Addresses Of PLL

M_PCTL   EQU     $FFFFFD             ; X space: PLL Control Register
M_PREDIV   EQU 16             ; predivide factor (base)
M_DIVFACT  EQU 8                    ; division factor (base)
M_MULTFACT EQU 0                    ; multiplication factor (base)

;    PLL Control register bits

M_MF     EQU   $FF                  ; Multiplication Factor Bits Mask (MF0-MF11)
M_MF0    EQU   0                    ; Multiplication Factor bit 0
M_MF1    EQU   1                    ; Multiplication Factor bit 1
M_MF2    EQU   2                    ; Multiplication Factor bit 2
M_MF3    EQU   3                    ; Multiplication Factor bit 3
M_MF4    EQU   4                    ; Multiplication Factor bit 4
M_MF5    EQU   5                    ; Multiplication Factor bit 5
M_MF6    EQU   6                    ; Multiplication Factor bit 6
M_MF7    EQU   7                    ; Multiplication Factor bit 7
M_DF     EQU   $700                 ; Division Factor Bits Mask (DF0-DF2)
M_DF0    EQU   8                    ; Division Factor bit 0
M_DF1    EQU   9                    ; Division Factor bit 1
M_DF2    EQU   10                   ; Division Factor bit 2
M_PSTP   EQU   12                   ; p stop
M_PEN    EQU   13                   ; PLL enable
M_OD0    EQU   14                   ; output divide factor [0]
M_OD1    EQU   15                   ; output divide factor [1]
M_PD     EQU   $1F0000              ; PreDivider Factor Bits Mask (PD0-PD3)
M_PD0    EQU   16                   ; PreDivider Factor bit 0
M_PD1    EQU   17                   ; PreDivider Factor bit 1
M_PD2    EQU   18                   ; PreDivider Factor bit 2
M_PD3    EQU   19                   ; PreDivider Factor bit 3
M_PD4    EQU   20                   ; PreDivider Factor bit 4
```

```
;---------------------------------------------------------------------
;
;       EQUATES for Status Register (SR)
;
;---------------------------------------------------------------------

;       control and status bits in SR

M_C       EQU     0               ; Carry
M_V       EQU     1               ; Overflow
M_Z       EQU     2               ; Zero
M_N       EQU     3               ; Negative
M_U       EQU     4               ; Unnormalized
M_E       EQU     5               ; Extension
M_L       EQU     6               ; Limit
M_S       EQU     7               ; Scaling Bit
M_I0      EQU     8               ; Interupt Mask Bit 0
M_I1      EQU     9               ; Interupt Mask Bit 1
M_S0      EQU     10              ; Scaling Mode Bit 0
M_S1      EQU     11              ; Scaling Mode Bit 1
M_SC      EQU     13              ; Sixteen_Bit Compatibility
M_DM      EQU     14              ; Double Precision Multiply
M_LF      EQU     15              ; DO-Loop Flag
M_FV      EQU     16              ; DO-Forever Flag
M_SA      EQU     17              ; Sixteen-Bit Arithmetic
M_CE      EQU     19              ; Instruction Cache Enable
M_SM      EQU     20              ; Arithmetic Saturation
M_RM      EQU     21              ; Rounding Mode
M_CP      EQU     $c00000         ; mask for CORE-DMA priority bits in SR
M_CP0     EQU     22              ; bit 0 of priority bits in SR
M_CP1     EQU     23              ; bit 1 of priority bits in SR

;---------------------------------------------------------------------
;
;       EQUATES for Operating Mode Register (OMR)
;
;---------------------------------------------------------------------

;       control and status bits in OMR

M_MA      EQU     0               ; Operating Mode A
M_MB      EQU     1               ; Operating Mode B
M_MC      EQU     2               ; Operating Mode C
M_MD      EQU     3               ; Operating Mode D
M_SD      EQU     6               ; Stop Delay
M_MS      EQU     7                 ;Memory Switch Mode
```

```
M_CDP    EQU     $300            ; mask for CORE-DMA priority bits in OMR
M_CDP0   EQU     8               ; bit 0 of priority bits in OMR Core DMA
M_CDP1   EQU     9               ; bit 1 of priority bits in OMR Core DMA
M_XYS    EQU     16              ; Stack Extension space select bit in OMR.
M_EUN    EQU     17              ; Extensed stack UNderflow flag in OMR.
M_EOV    EQU     18              ; Extended stack OVerflow flag in OMR.
M_WRP    EQU     19              ; Extended WRaP flag in OMR.
M_SEN    EQU     20              ; Stack Extension Enable bit in OMR.
M_MSW0   EQU     21              ; Memory Switch Mode 0
M_MSW1   EQU     22              ; Memory Switch Mode 1

;---------------------------------------------------------------------
;
;       EQUATES for DAX (SPDIF Tx)
;
;---------------------------------------------------------------------

;       Register Addresses

M_XSTR   EQU     $FFFFD4         ; X space: DAX Status Register (XSTR)
M_XADRB  EQU     $FFFFD3         ; X space: DAX Audio Data Register B (XADRB)
M_XADR   EQU     $FFFFD2         ; X space: DAX Audio Data Register (XADR)
M_XADRA  EQU     $FFFFD2         ; X space: DAX Audio Data Register A (XADRA)
M_XNADR  EQU     $FFFFD1         ; X space: DAX Non-Audio Data Register (XNADR)
M_XCTR   EQU     $FFFFD0         ; X space: DAX Control Register (XCTR)

;       status bits in XSTR

M_XADE   EQU     0               ; DAX Audio Data Register Empty (XADE)
M_XAUR   EQU     1               ; DAX Trasmit Underrun Error Flag (XAUR)
M_XBLK   EQU     2               ; DAX Block Transferred (XBLK)

;       non-audio bits in XNADR

M_XVA    EQU     10              ; DAX Channel A Validity (XVA)
M_XUA    EQU     11              ; DAX Channel A User Data (XUA)
M_XCA    EQU     12              ; DAX Channel A Channel Status (XCA)
M_XVB    EQU     13              ; DAX Channel B Validity (XVB)
M_XUB    EQU     14              ; DAX Channel B User Data (XUB)
M_XCB    EQU     15              ; DAX Channel B Channel Status (XCB)

;       control bits in XCTR

M_XDIE   EQU     0               ; DAX Audio Data Register Empty Interrupt Enable (XDIE)
M_XUIE   EQU     1               ; DAX Underrun Error Interrupt Enable (XUIE)
M_XBIE   EQU     2               ; DAX Block Transferred Interrupt Enable (XBIE)
M_XCS0   EQU     3               ; DAX Clock Input Select 0 (XCS0)
M_XCS1   EQU     4               ; DAX Clock Input Select 1 (XCS1)
M_XSB    EQU     5               ; DAX Start Block (XSB)
```

```
;------------------------------------------------------------------------
;
;       EQUATES for SHI
;
;------------------------------------------------------------------------

;       Register Addresses

M_HRX    EQU     $FFFF94         ; X space: SHI Receive FIFO (HRX)
M_HTX    EQU     $FFFF93         ; X space: SHI Transmit Register (HTX)
M_HSAR   EQU     $FFFF92         ; X space: SHI I2C Slave Address Register (HSAR)
M_HCSR   EQU     $FFFF91         ; X space: SHI Control/Status Register (HCSR)
M_HCKR   EQU     $FFFF90         ; X space: SHI Clock Control Register (HCKR)

;       HSAR bits

M_HA6    EQU     23              ; SHI I2C Slave Address (HA6)
M_HA5    EQU     22              ; SHI I2C Slave Address (HA5)
M_HA4    EQU     21              ; SHI I2C Slave Address (HA4)
M_HA3    EQU     20              ; SHI I2C Slave Address (HA3)
M_HA1    EQU     18              ; SHI I2C Slave Address (HA1)

;      control and status bits in HCSR

M_HBUSY  EQU     22              ; SHI Host Busy (HBUSY)
M_HBER   EQU     21              ; SHI Bus Error (HBER)
M_HROE   EQU     20              ; SHI Receive Overrun Error (HROE)
M_HRFF   EQU     19              ; SHI Receivr FIFO Full (HRFF)
M_HRNE   EQU     17              ; SHI Receive FIFO Not Empty (HRNE)
M_HTDE   EQU     15              ; SHI Host Transmit data Empty (HTDE)
M_HTUE   EQU     14              ; SHI Host Transmit Underrun Error (HTUE)
M_HRIE1  EQU     13              ; SHI Receive Interrupt Enable (HRIE1)
M_HRIE0  EQU     12              ; SHI Receive Interrupt Enable (HRIE0)
M_HTIE   EQU     11              ; SHI Transmit Interrupt Enable (HTIE)
M_HBIE   EQU     10              ; SHI Bus-Error Interrupt Enable (HBIE)
M_HIDLE  EQU     9               ; SHI Idle (HIDLE)
M_HRQE1  EQU     8               ; SHI Host Request Enable (HRQE1)
M_HRQE0  EQU     7               ; SHI Host Request Enable (HRQE0)
M_HMST   EQU     6               ; SHI Master Mode (HMST)
M_HFIFO  EQU     5               ; SHI FIFO Enable Control (HFIFO)
M_HCKFR  EQU     4               ; SHI Clock Freeze (HCKFR)
M_HM1    EQU     3               ; SHI Serial Host Interface Mode (HM1)
M_HM0    EQU     2               ; SHI Serial Host Interface Mode (HM0)
M_HI2C   EQU     1               ; SHI I2c/SPI Selection (HI2C)
M_HEN    EQU     0               ; SHI Host Enable (HEN)

;       control bits in HCKR

M_HFM1   EQU     13              ; SHI Filter Model (HFM1)
M_HFM0   EQU     12              ; SHI Filter Model (HFM0)
M_HDM7   EQU     10               ; SHI Divider Modulus Select (HDM7)
M_HDM6   EQU     9               ; SHI Divider Modulus Select (HDM6)
M_HDM5   EQU     8               ; SHI Divider Modulus Select (HDM5)
M_HDM4   EQU     7               ; SHI Divider Modulus Select (HDM4)
M_HDM3   EQU     6               ; SHI Divider Modulus Select (HDM3)
M_HDM2   EQU     5               ; SHI Divider Modulus Select (HDM2)
M_HDM1   EQU     4               ; SHI Divider Modulus Select (HDM1)
M_HDM0   EQU     3               ; SHI Divider Modulus Select (HDM0)
```

```
M_HRS    EQU     2               ; SHI Prescalar Rate Select (HRS)
M_CPOL   EQU     1               ; SHI Clock Polarity (CPOL)
M_CPHA   EQU     0               ; SHI Clock Phase (CPHA)

;------------------------------------------------------------------------
;
;       EQUATES for ESAI_1 Registers
; register bit equates can be the same as for the ESAI register bit equates.
;
;------------------------------------------------------------------------

;       Register Addresses

M_RSMB_1 EQU    $FFFF9C         ; Y space: ESAI_1 Receive Slot Mask Register B (RSMB_1)
M_RSMA_1 EQU    $FFFF9B         ; Y space: ESAI_1 Receive Slot Mask Register A (RSMA_1)
M_TSMB_1 EQU    $FFFF9A         ; Y space: ESAI_1 Transmit Slot Mask Register B (TSMB_1)
M_TSMA_1 EQU    $FFFF99         ; Y space: ESAI_1 Transmit Slot Mask Register A (TSMA_1)
M_RCCR_1 EQU    $FFFF98         ; Y space: ESAI_1 Receive Clock Control Register (RCCR_1)
M_RCR_1  EQU    $FFFF97         ; Y space: ESAI_1 Receive Control Register (RCR_1)
M_TCCR_1 EQU    $FFFF96         ; Y space: ESAI_1 Transmit Clock Control Register (TCCR_1)
M_TCR_1  EQU    $FFFF95         ; Y space: ESAI_1 Transmit Control Register (TCR_1)
M_SAICR_1 EQU   $FFFF94         ; Y space: ESAI_1 Control Register (SAICR_1)
M_SAISR_1 EQU   $FFFF93         ; Y space: ESAI_1 Status Register (SAISR_1)
M_RX3_1 EQU     $FFFF8B         ; Y space: ESAI_1 Receive Data Register 3 (RX3_1)
M_RX2_1 EQU     $FFFF8A         ; Y space: ESAI_1 Receive Data Register 2 (RX2_1)
M_RX1_1 EQU     $FFFF89         ; Y space: ESAI_1 Receive Data Register 1 (RX1_1)
M_RX0_1 EQU     $FFFF88         ; Y space: ESAI_1 Receive Data Register 0 (RX0_1)
M_TSR_1 EQU     $FFFF86         ; Y space: ESAI_1 Time Slot Register (TSR_1)
M_TX5_1 EQU     $FFFF85         ; Y space: ESAI_1 Transmit Data Register 5 (TX5_1)
M_TX4_1 EQU     $FFFF84         ; Y space: ESAI_1 Transmit Data Register 4 (TX4_1)
M_TX3_1 EQU     $FFFF83         ; Y space: ESAI_1 Transmit Data Register 3 (TX3_1)
M_TX2_1 EQU     $FFFF82         ; Y space: ESAI_1 Transmit Data Register 2 (TX2_1)
M_TX1_1 EQU     $FFFF81         ; Y space: ESAI_1 Transmit Data Register 1 (TX1_1)
M_TX0_1 EQU     $FFFF80         ; Y space: ESAI_1 Transmit Data Register 0 (TX0_1)

;------------------------------------------------------------------------
;
;       EQUATES for ESAI
;
;------------------------------------------------------------------------

;       Register Addresses

M_RSMB   EQU     $FFFFBC        ; X space: ESAI Receive Slot Mask Register B (RSMB)
M_RSMA   EQU     $FFFFBB        ; X space: ESAI Receive Slot Mask Register A (RSMA)
M_TSMB   EQU     $FFFFBA        ; X space: ESAI Transmit Slot Mask Register B (TSMB)
M_TSMA   EQU     $FFFFB9        ; X space: ESAI Transmit Slot Mask Register A (TSMA)
M_RCCR   EQU     $FFFFB8        ; X space: ESAI Receive Clock Control Register (RCCR)
M_RCR    EQU     $FFFFB7        ; X space: ESAI Receive Control Register (RCR)
M_TCCR   EQU     $FFFFB6        ; X space: ESAI Transmit Clock Control Register (TCCR)
M_TCR    EQU     $FFFFB5        ; X space: ESAI Transmit Control Register (TCR)
M_SAICR  EQU     $FFFFB4        ; X space: ESAI Control Register (SAICR)
M_SAISR  EQU     $FFFFB3        ; X space: ESAI Status Register (SAISR)
M_RX3    EQU     $FFFFAB        ; X space: ESAI Receive Data Register 3 (RX3)
M_RX2    EQU     $FFFFAA        ; X space: ESAI Receive Data Register 2 (RX2)
M_RX1    EQU     $FFFFA9        ; X space: ESAI Receive Data Register 1 (RX1)
M_RX0    EQU     $FFFFA8        ; X space: ESAI Receive Data Register 0 (RX0)
M_TSR    EQU     $FFFFA6        ; X space: ESAI Time Slot Register (TSR)
M_TX5    EQU     $FFFFA5        ; X space: ESAI Transmit Data Register 5 (TX5)
```

```
M_TX4    EQU     $FFFFA4         ; X space: ESAI Transmit Data Register 4 (TX4)
M_TX3    EQU     $FFFFA3         ; X space: ESAI Transmit Data Register 3 (TX3)
M_TX2    EQU     $FFFFA2         ; X space: ESAI Transmit Data Register 2 (TX2)
M_TX1    EQU     $FFFFA1         ; X space: ESAI Transmit Data Register 1 (TX1)
M_TX0    EQU     $FFFFA0         ; X space: ESAI Transmit Data Register 0 (TX0)

;       RSMB Register bits

M_RS31   EQU     15              ; ESAI
M_RS30   EQU     14              ; ESAI
M_RS29   EQU     13              ; ESAI
M_RS28   EQU     12              ; ESAI
M_RS27   EQU     11              ; ESAI
M_RS26   EQU     10              ; ESAI
M_RS25   EQU     9               ; ESAI
M_RS24   EQU     8               ; ESAI
M_RS23   EQU     7               ; ESAI
M_RS22   EQU     6               ; ESAI
M_RS21   EQU     5               ; ESAI
M_RS20   EQU     4               ; ESAI
M_RS19   EQU     3               ; ESAI
M_RS18   EQU     2               ; ESAI
M_RS17   EQU     1               ; ESAI
M_RS16   EQU     0               ; ESAI

;       RSMA Register bits

M_RS15   EQU     15              ; ESAI
M_RS14   EQU     14              ; ESAI
M_RS13   EQU     13              ; ESAI
M_RS12   EQU     12              ; ESAI
M_RS11   EQU     11              ; ESAI
M_RS10   EQU     10              ; ESAI
M_RS9    EQU     9               ; ESAI
M_RS8    EQU     8               ; ESAI
M_RS7    EQU     7               ; ESAI
M_RS6    EQU     6               ; ESAI
M_RS5    EQU     5               ; ESAI
M_RS4    EQU     4               ; ESAI
M_RS3    EQU     3               ; ESAI
M_RS2    EQU     2               ; ESAI
M_RS1    EQU     1               ; ESAI
M_RS0    EQU     0               ; ESAI

;       TSMB Register bits

M_TS31   EQU     15              ; ESAI
M_TS30   EQU     14              ; ESAI
M_TS29   EQU     13              ; ESAI
M_TS28   EQU     12              ; ESAI
M_TS27   EQU     11              ; ESAI
M_TS26   EQU     10              ; ESAI
M_TS25   EQU     9               ; ESAI
M_TS24   EQU     8               ; ESAI
M_TS23   EQU     7               ; ESAI
M_TS22   EQU     6               ; ESAI
M_TS21   EQU     5               ; ESAI
M_TS20   EQU     4               ; ESAI
M_TS19   EQU     3               ; ESAI
```

```
M_TS18   EQU     2               ; ESAI
M_TS17   EQU     1               ; ESAI
M_TS16   EQU     0               ; ESAI

;       TSMA Register bits

M_TS15   EQU     15              ; ESAI
M_TS14   EQU     14              ; ESAI
M_TS13   EQU     13              ; ESAI
M_TS12   EQU     12              ; ESAI
M_TS11   EQU     11              ; ESAI
M_TS10   EQU     10              ; ESAI
M_TS9    EQU     9               ; ESAI
M_TS8    EQU     8               ; ESAI
M_TS7    EQU     7               ; ESAI
M_TS6    EQU     6               ; ESAI
M_TS5    EQU     5               ; ESAI
M_TS4    EQU     4               ; ESAI
M_TS3    EQU     3               ; ESAI
M_TS2    EQU     2               ; ESAI
M_TS1    EQU     1               ; ESAI
M_TS0    EQU     0               ; ESAI

;       RCCR Register bits

M_RHCKD  EQU     23              ; ESAI
M_RFSD   EQU     22              ; ESAI
M_RCKD   EQU     21              ; ESAI
M_RHCKP   EQU     20              ;ESAI
M_RFSP   EQU     19              ; ESAI
M_RCKP   EQU     18               ;ESAI
M_RFP     EQU      $3C000        ;ESAI MASK
M_RFP3   EQU     17              ; ESAI
M_RFP2   EQU     16              ; ESAI
M_RFP1   EQU     15              ; ESAI
M_RFP0   EQU     14              ; ESAI
M_RDC     EQU     $3E00         ;ESAI MASK
M_RDC4   EQU     13              ; ESAI
M_RDC3   EQU     12              ; ESAI
M_RDC2   EQU     11              ; ESAI
M_RDC1   EQU     10              ; ESAI
M_RDC0   EQU     9              ; ESAI
M_RPSR   EQU     8               ; ESAI
M_RPM     EQU    $FF
M_RPM7   EQU     7               ; ESAI
M_RPM6   EQU     6               ; ESAI
M_RPM5   EQU     5               ; ESAI
M_RPM4   EQU     4               ; ESAI
M_RPM3   EQU     3               ; ESAI
M_RPM2   EQU     2               ; ESAI
M_RPM1   EQU     1               ; ESAI
M_RPM0   EQU     0               ; ESAI

;       RCR Register bits

M_RLIE   EQU     23              ; ESAI
M_RIE    EQU     22              ; ESAI
M_REDIE  EQU     21              ; ESAI
M_REIE   EQU     20              ; ESAI
```

```
M_RPR     EQU       19         ;          ESAI
M_RFSR   EQU     16              ; ESAI
M_RFSL   EQU     15              ; ESAI
M_RSWS    EQU       $7C00        ;ESAI MASK
M_RSWS4   EQU     14              ; ESAI
M_RSWS3   EQU     13              ; ESAI
M_RSWS2   EQU     12              ; ESAI
M_RSWS1   EQU     11              ; ESAI
M_RSWS0   EQU     10              ; ESAI
M_RMOD      EQU       $300
M_RMOD1  EQU     9               ; ESAI
M_RMOD0  EQU     8               ; ESAI
M_RWA   EQU     7               ; ESAI
M_RSHFD  EQU     6               ; ESAI
M_RE      EQU      $F
M_RE3    EQU     3               ; ESAI
M_RE2    EQU     2               ; ESAI
M_RE1    EQU     1               ; ESAI
M_RE0    EQU     0               ; ESAI

;       TCCR Register bits

M_THCKD  EQU     23              ; ESAI
M_TFSD   EQU     22              ; ESAI
M_TCKD   EQU     21              ; ESAI
M_THCKP   EQU     20               ;ESAI
M_TFSP   EQU     19              ; ESAI
M_TCKP   EQU     18              ; ESAI
M_TFP    EQU      $3C000
M_TFP3   EQU     17              ; ESAI
M_TFP2   EQU     16              ; ESAI
M_TFP1   EQU     15              ; ESAI
M_TFP0   EQU     14              ; ESAI
M_TDC     EQU     $3E00       ;
M_TDC4   EQU     13              ; ESAI
M_TDC3   EQU     12              ; ESAI
M_TDC2   EQU     11              ; ESAI
M_TDC1   EQU     10              ; ESAI
M_TDC0   EQU     9              ; ESAI
M_TPSR   EQU     8               ; ESAI
M_TPM     EQU    $FF            ;
M_TPM7   EQU     7               ; ESAI
M_TPM6   EQU     6               ; ESAI
M_TPM5   EQU     5               ; ESAI
M_TPM4   EQU     4               ; ESAI
M_TPM3   EQU     3               ; ESAI
M_TPM2   EQU     2               ; ESAI
M_TPM1   EQU     1               ; ESAI
M_TPM0   EQU     0               ; ESAI

;       TCR Register bits

M_TLIE   EQU     23              ; ESAI
M_TIE    EQU     22              ; ESAI
M_TEDIE  EQU     21              ; ESAI
M_TEIE   EQU     20              ; ESAI
M_TPR     EQU       19         ; ESAI
M_PADC    EQU       17         ; ESAI
M_TFSR   EQU     16              ; ESAI
```

```
M_TFSL   EQU     15              ; ESAI
M_TSWS   EQU    $7C00
M_TSWS4   EQU     14              ; ESAI
M_TSWS3   EQU     13              ; ESAI
M_TSWS2   EQU     12              ; ESAI
M_TSWS1   EQU     11              ; ESAI
M_TSWS0   EQU     10              ; ESAI
M_TMOD    EQU      $300
M_TMOD1  EQU     9               ; ESAI
M_TMOD0  EQU     8               ; ESAI
M_TWA   EQU     7               ; ESAI
M_TSHFD  EQU     6               ; ESAI
M_TEM       EQU     $3F
M_TE5    EQU     5               ; ESAI
M_TE4    EQU     4               ; ESAI
M_TE3    EQU     3               ; ESAI
M_TE2    EQU     2               ; ESAI
M_TE1    EQU     1               ; ESAI
M_TE0    EQU     0               ; ESAI

;       control bits of SAICR

M_ALC   EQU   8               ;ESAI
M_TEBE   EQU     7               ; ESAI
M_SYN    EQU     6               ; ESAI
M_OF2    EQU     2               ; ESAI
M_OF1    EQU     1               ; ESAI
M_OF0    EQU     0               ; ESAI

;       status bits of SAISR

M_TODE   EQU     17              ; ESAI
M_TEDE   EQU     16              ; ESAI
M_TDE    EQU     15              ; ESAI
M_TUE    EQU     14              ; ESAI
M_TFS    EQU     13              ; ESAI
M_RODF   EQU     10              ; ESAI
M_REDF   EQU     9               ; ESAI
M_RDF    EQU     8               ; ESAI
M_ROE    EQU     7               ; ESAI
M_RFS    EQU     6               ; ESAI
M_IF2    EQU     2               ; ESAI
M_IF1    EQU     1               ; ESAI
M_IF0    EQU     0               ; ESAI



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

;       Register Addresses Of TIMER0

M_TCSR0  EQU     $FFFF8F         ; X space: TIMER0 Control/Status Register
M_TLR0   EQU     $FFFF8E     ; X space: TIMER0 Load Reg
M_TCPR0  EQU     $FFFF8D         ; X space: TIMER0 Compare Register
M_TCR0   EQU     $FFFF8C     ; X space: TIMER0 Count Register
```

```
;       Register Addresses Of TIMER1

M_TCSR1  EQU     $FFFF8B         ; X space: TIMER1 Control/Status Register
M_TLR1   EQU     $FFFF8A     ; X space: TIMER1 Load Reg
M_TCPR1  EQU     $FFFF89         ; X space: TIMER1 Compare Register
M_TCR1   EQU     $FFFF88     ; X space: TIMER1 Count Register

;       Register Addresses Of TIMER2

M_TCSR2  EQU     $FFFF87         ; X space: TIMER2 Control/Status Register
M_TLR2   EQU     $FFFF86     ; X space: TIMER2 Load Reg
M_TCPR2  EQU     $FFFF85         ; X space: TIMER2 Compare Register
M_TCR2   EQU     $FFFF84     ; X space: TIMER2 Count Register

M_TPLR   EQU     $FFFF83     ; X space: TIMER Prescaler Load Register
M_TPCR   EQU     $FFFF82     ; X space: TIMER Prescalar Count Register

;       Timer Control/Status Register Bit Flags

M_TE     EQU     0               ; Timer Enable
M_TOIE   EQU     1               ; Timer Overflow Interrupt Enable
M_TCIE   EQU     2               ; Timer Compare Interrupt Enable
M_TC     EQU     $F0             ; Timer Control Mask (TC0-TC3)
M_TC0    EQU     4               ; Timer Control 0 - Timer Control Bits
M_TC1    EQU     5               ; Timer Control 1
M_TC2    EQU     6               ; Timer Control 2
M_TC3    EQU     7               ; Timer Control 3
M_INV    EQU     8               ; Inverter Bit
M_TRM    EQU     9               ; Timer Restart Mode
M_DIR    EQU     11              ; Direction Bit
M_DI     EQU     12              ; Data Input
M_DO     EQU     13              ; Data Output
M_PCE    EQU     15      ; Prescaled Clock Enable
M_TOF    EQU     20              ; Timer Overflow Flag
M_TCF    EQU     21              ; Timer Compare Flag

;       Timer Prescaler Register Bit Flags

M_PS     EQU  $600000     ; Prescaler Source Mask
M_PS0    EQU  21
M_PS1    EQU  22
```

```
;---------------------------------------------------------------------
;
;       EQUATES for EFCOP
;
;---------------------------------------------------------------------

M_EFCOP EQU   $FFFFB0
M_FDIR  EQU   M_EFCOP+$0      ; Y space: EFCOP Data Input Register
M_FDOR  EQU   M_EFCOP+$1      ; Y space: EFCOP Data Output Register
```

```
M_FKIR  EQU   M_EFCOP+$2       ; Y space: EFCOP K-Constant Input Register
M_FCNT  EQU   M_EFCOP+$3       ; Y space: EFCOP Filter Count Register
M_FCSR  EQU   M_EFCOP+$4       ; Y space: EFCOP Control Status Register
M_FACR  EQU   M_EFCOP+$5       ; Y space: EFCOP ALU Control Register
M_FDBA  EQU   M_EFCOP+$6       ; Y space: EFCOP Data Base Address
M_FCBA  EQU   M_EFCOP+$7       ; Y space: EFCOP Coefficient Base Address
M_FDCH  EQU   M_EFCOP+$8       ; Y space: EFCOP Decimation/Channel Count Register
M_FL    EQU   $30000           ; EFCOP interrupt mask

;------------------------------------------------------------------------
;
;       EQUATES for Patch Module
;
;------------------------------------------------------------------------

M_PATCH EQU   $FFFFA0
M_PATCHA0  EQU   M_PATCH+$0      ; Y space: Patch Address 0 Register
M_PATCHA1  EQU   M_PATCH+$1      ; Y space: Patch Address 1 Register
M_PATCHA2  EQU   M_PATCH+$2      ; Y space: Patch Address 2 Register
M_PATCHA3  EQU   M_PATCH+$3      ; Y space: Patch Address 3 Register
M_PATCHA4  EQU   M_PATCH+$4      ; Y space: Patch Address 4 Register
M_PATCHA5  EQU   M_PATCH+$5      ; Y space: Patch Address 5 Register
M_PATCHA6  EQU   M_PATCH+$6      ; Y space: Patch Address 6 Register
M_PATCHA7  EQU   M_PATCH+$7      ; Y space: Patch Address 7 Register
M_PATCHI0  EQU   M_PATCH+$8      ; Y space: Patch Instruction 0 Register
M_PATCHI1  EQU   M_PATCH+$9      ; Y space: Patch Instruction 0 Register
M_PATCHI2  EQU   M_PATCH+$A      ; Y space: Patch Instruction 0 Register
M_PATCHI3  EQU   M_PATCH+$B      ; Y space: Patch Instruction 0 Register
M_PATCHI4  EQU   M_PATCH+$C      ; Y space: Patch Instruction 0 Register
M_PATCHI5  EQU   M_PATCH+$D      ; Y space: Patch Instruction 0 Register
M_PATCHI6  EQU   M_PATCH+$E      ; Y space: Patch Instruction 0 Register
M_PATCHI7  EQU   M_PATCH+$F      ; Y space: Patch Instruction 0 Register

;------------------------------------------------------------------------
;
;       EQUATES for Watchdog Timer (WDT)
;
;------------------------------------------------------------------------
M_WCR      EQU   $FFFFC0
M_WCR      EQU       M_WCR+$0   ; Y space: Watchdog Timer Control Register
M_WMR      EQU       M_WCR+$1   ; Y space: Watchdog Timer Modulus Register
M_WCNTR    EQU       M_WCR+$2   ; Y space: Watchdog Timer Counter Register
M_WSR      EQU       M_WCR+$3   ; Y space: Watchdog Timer Service Register

;       Watchdog Timer Control Register Bit Flags

M_WEN EQU     0                ; Watchdog Timer Enable
M_DBGC EQU     1               ; Watchdog Timer DEBUG control
M_WTC EQU     3                ; Watchdog Timer WAIT control

;       Watchdog Timer Modulus Register Bit Flags

M_WM0 EQU 0                    ; Watchdog Timer Modulus bit 0 mask
M_WM1 EQU 1                    ; Watchdog Timer Modulus bit 1 mask
M_WM2 EQU 2                    ; Watchdog Timer Modulus bit 2 mask
M_WM3 EQU 3                    ; Watchdog Timer Modulus bit 3 mask
M_WM4 EQU 4                    ; Watchdog Timer Modulus bit 4 mask
M_WM5 EQU 5                    ; Watchdog Timer Modulus bit 5 mask
M_WM6 EQU 6                    ; Watchdog Timer Modulus bit 6 mask
```

Equates

```
M_WM7 EQU 7                     ; Watchdog Timer Modulus bit 7 mask
M_WM8 EQU 8                     ; Watchdog Timer Modulus bit 8 mask
M_WM9 EQU 9                     ; Watchdog Timer Modulus bit 9 mask
M_WM10 EQU 10                   ; Watchdog Timer Modulus bit 10 mask
M_WM11 EQU 11                   ; Watchdog Timer Modulus bit 11 mask
M_WM12 EQU 12                   ; Watchdog Timer Modulus bit 12 mask
M_WM13 EQU 13                   ; Watchdog Timer Modulus bit 13 mask
M_WM14 EQU 14                   ; Watchdog Timer Modulus bit 14 mask
M_WM15 EQU 15                   ; Watchdog Timer Modulus bit 15 mask

;       Watchdog Timer Counter Register Bit Flags

M_WC0 EQU 0                     ; Watchdog Timer Counter bit 0 mask
M_WC1 EQU 1                     ; Watchdog Timer Counter bit 1 mask
M_WC2 EQU 2                     ; Watchdog Timer Counter bit 2 mask
M_WC3 EQU 3                     ; Watchdog Timer Counter bit 3 mask
M_WC4 EQU 4                     ; Watchdog Timer Counter bit 4 mask
M_WC5 EQU 5                     ; Watchdog Timer Counter bit 5 mask
M_WC6 EQU 6                     ; Watchdog Timer Counter bit 6 mask
M_WC7 EQU 7                     ; Watchdog Timer Counter bit 7 mask
M_WC8 EQU 8                     ; Watchdog Timer Counter bit 8 mask
M_WC9 EQU 9                     ; Watchdog Timer Counter bit 9 mask
M_WC10 EQU 10                   ; Watchdog Timer Counter bit 10 mask
M_WC11 EQU 11                   ; Watchdog Timer Counter bit 11 mask
M_WC12 EQU 12                   ; Watchdog Timer Counter bit 12 mask
M_WC13 EQU 13                   ; Watchdog Timer Counter bit 13 mask
M_WC14 EQU 14                   ; Watchdog Timer Counter bit 14 mask
M_WC15 EQU 15                   ; Watchdog Timer Counter bit 15 mask

;       Watchdog Timer Service Register Bit Flags

M_WS0 EQU 0                     ; Watchdog Timer Service bit 0 mask
M_WS1 EQU 1                     ; Watchdog Timer Service bit 1 mask
M_WS2 EQU 2                     ; Watchdog Timer Service bit 2 mask
M_WS3 EQU 3                     ; Watchdog Timer Service bit 3 mask
M_WS4 EQU 4                     ; Watchdog Timer Service bit 4 mask
M_WS5 EQU 5                     ; Watchdog Timer Service bit 5 mask
M_WS6 EQU 6                     ; Watchdog Timer Service bit 6 mask
M_WS7 EQU 7                     ; Watchdog Timer Service bit 7 mask
M_WS8 EQU 8                     ; Watchdog Timer Service bit 8 mask
M_WS9 EQU 9                     ; Watchdog Timer Service bit 9 mask
M_WS10 EQU 10                   ; Watchdog Timer Service bit 10 mask
M_WS11 EQU 11                   ; Watchdog Timer Service bit 11 mask
M_WS12 EQU 12                   ; Watchdog Timer Service bit 12 mask
M_WS13 EQU 13                   ; Watchdog Timer Service bit 13 mask
M_WS14 EQU 14                   ; Watchdog Timer Service bit 14 mask
M_WS15 EQU 15                   ; Watchdog Timer Service bit 15 mask


;------------------ end of ioequ.asm ------------------------
```

Appendix C
Programmer's Reference

C.1 Introduction

This section has been compiled as a reference for programmers. It contains a table showing the addresses of all the DSPs memory-mapped peripherals, an interrupt address table, an interrupt exception priority table, and programming sheets for the major programmable registers on the DSP.

C.1.1 Peripheral Addresses

Figure C-1. lists the memory addresses of all on-chip peripherals.

C.1.2 Interrupt Addresses

Figure C-2. lists the interrupt starting addresses and sources.

C.1.3 Interrupt Priorities

Figure C-3. lists the priorities of specific interrupts within interrupt priority levels.

C.1.4 Programming Sheets

The remaining figures describe major programmable registers on the DSP56374.

C.1.5 Internal I/O Memory Map

Table C-1. Internal I/O Memory Map (X Memory)

Peripheral	Address	Register Name
IPR	X:$FFFFFF	INTERRUPT PRIORITY REGISTER CORE (IPR-C)
	X:$FFFFFE	INTERRUPT PRIORITY REGISTER PERIPHERAL (IPR-P)
PLL	X:$FFFFFD	PLL CONTROL REGISTER (PCTL)
ONCE	X:$FFFFFC	ONCE GDB REGISTER (OGDB)
BIU	X:$FFFFFB	BCR
	X:$FFFFFA	RESERVED
	X:$FFFFF9	RESERVED
	X:$FFFFF8	RESERVED
	X:$FFFFF7	RESERVED
	X:$FFFFF6	RESERVED
	X:$FFFFF5	ID Register (IDR)
DMA	X:$FFFFF4	DMA STATUS REGISTER (DSTR)
	X:$FFFFF3	DMA OFFSET REGISTER 0 (DOR0)
	X:$FFFFF2	DMA OFFSET REGISTER 1 (DOR1)
	X:$FFFFF1	DMA OFFSET REGISTER 2 (DOR2)
	X:$FFFFF0	DMA OFFSET REGISTER 3 (DOR3)
DMA0	X:$FFFFEF	DMA SOURCE ADDRESS REGISTER (DSR0)
	X:$FFFFEE	DMA DESTINATION ADDRESS REGISTER (DDR0)
	X:$FFFFED	DMA COUNTER (DCO0)
	X:$FFFFEC	DMA CONTROL REGISTER (DCR0)
DMA1	X:$FFFFEB	DMA SOURCE ADDRESS REGISTER (DSR1)
	X:$FFFFEA	DMA DESTINATION ADDRESS REGISTER (DDR1)
	X:$FFFFE9	DMA COUNTER (DCO1)
	X:$FFFFE8	DMA CONTROL REGISTER (DCR1)

Table C-1. Internal I/O Memory Map (X Memory) (continued)

Peripheral	Address	Register Name
DMA2	X:$FFFFE7	DMA SOURCE ADDRESS REGISTER (DSR2)
	X:$FFFFE6	DMA DESTINATION ADDRESS REGISTER (DDR2)
	X:$FFFFE5	DMA COUNTER (DCO2)
	X:$FFFFE4	DMA CONTROL REGISTER (DCR2)
DMA3	X:$FFFFE3	DMA SOURCE ADDRESS REGISTER (DSR3)
	X:$FFFFE2	DMA DESTINATION ADDRESS REGISTER (DDR3)
	X:$FFFFE1	DMA COUNTER (DCO3)
	X:$FFFFE0	DMA CONTROL REGISTER (DCR3)
DMA4	X:$FFFFDF	DMA SOURCE ADDRESS REGISTER (DSR4)
	X:$FFFFDE	DMA DESTINATION ADDRESS REGISTER (DDR4)
	X:$FFFFDD	DMA COUNTER (DCO4)
	X:$FFFFDC	DMA CONTROL REGISTER (DCR4)
DMA5	X:$FFFFDB	DMA SOURCE ADDRESS REGISTER (DSR5)
	X:$FFFFDA	DMA DESTINATION ADDRESS REGISTER (DDR5)
	X:$FFFFD9	DMA COUNTER (DCO5)
	X:$FFFFD8	DMA CONTROL REGISTER (DCR5)
RESERVED	X:$FFFFD7	RESERVED
	X:$FFFFD6	RESERVED
	X:$FFFFD5	RESERVED
	X:$FFFFD4	RESERVED
	X:$FFFFD3	RESERVED
	X:$FFFFD2	RESERVED
	X:$FFFFD1	RESERVED
	X:$FFFFD0	RESERVED
	X:$FFFFCF	RESERVED
	X:$FFFFCE	RESERVED
	X:$FFFFCD	RESERVED
	X:$FFFFCC	RESERVED
	X:$FFFFCB	RESERVED
	X:$FFFFCA	RESERVED
	X:$FFFFC9	RESERVED
	X:$FFFFC8	RESERVED
	X:$FFFFC7	RESERVED
	X:$FFFFC6	RESERVED
	X:$FFFFC5	RESERVED
	X:$FFFFC4	RESERVED
	X:$FFFFC3	RESERVED
	X:$FFFFC2	RESERVED
	X:$FFFFC1	RESERVED
	X:$FFFFC0	RESERVED
PORT C	X:$FFFFBF	PORT C CONTROL REGISTER (PCRC)
	X:$FFFFBE	PORT C DIRECTION REGISTER (PRRC)
	X:$FFFFBD	PORT C GPIO DATA REGISTER (PDRC)

Table C-1. Internal I/O Memory Map (X Memory) (continued)

Peripheral	Address	Register Name
ESAI	X:$FFFFBC	ESAI RECEIVE SLOT MASK REGISTER B (RSMB)
	X:$FFFFBB	ESAI RECEIVE SLOT MASK REGISTER A (RSMA)
	X:$FFFFBA	ESAI TRANSMIT SLOT MASK REGISTER B (TSMB)
	X:$FFFFB9	ESAI TRANSMIT SLOT MASK REGISTER A (TSMA)
	X:$FFFFB8	ESAI RECEIVE CLOCK CONTROL REGISTER (RCCR)
	X:$FFFFB7	ESAI RECEIVE CONTROL REGISTER (RCR)
	X:$FFFFB6	ESAI TRANSMIT CLOCK CONTROL REGISTER (TCCR)
	X:$FFFFB5	ESAI TRANSMIT CONTROL REGISTER (TCR)
	X:$FFFFB4	ESAI COMMON CONTROL REGISTER (SAICR)
	X:$FFFFB3	ESAI STATUS REGISTER (SAISR)
	X:$FFFFB2	RESERVED
	X:$FFFFB1	RESERVED
	X:$FFFFB0	RESERVED
	X:$FFFFAF	RESERVED
	X:$FFFFAE	RESERVED
	X:$FFFFAD	RESERVED
	X:$FFFFAC	RESERVED
	X:$FFFFAB	ESAI RECEIVE DATA REGISTER 3 (RX3)
	X:$FFFFAA	ESAI RECEIVE DATA REGISTER 2 (RX2)
	X:$FFFFA9	ESAI RECEIVE DATA REGISTER 1 (RX1)
	X:$FFFFA8	ESAI RECEIVE DATA REGISTER 0 (RX0)
	X:$FFFFA7	RESERVED
	X:$FFFFA6	ESAI TIME SLOT REGISTER (TSR)
	X:$FFFFA5	ESAI TRANSMIT DATA REGISTER 5 (TX5)
	X:$FFFFA4	ESAI TRANSMIT DATA REGISTER 4 (TX4)
	X:$FFFFA3	ESAI TRANSMIT DATA REGISTER 3 (TX3)
	X:$FFFFA2	ESAI TRANSMIT DATA REGISTER 2 (TX2)
	X:$FFFFA1	ESAI TRANSMIT DATA REGISTER 1 (TX1)
	X:$FFFFA0	ESAI TRANSMIT DATA REGISTER 0 (TX0)
	X:$FFFF9F	RESERVED
	X:$FFFF9E	RESERVED
	X:$FFFF9D	RESERVED
	X:$FFFF9C	RESERVED
	X:$FFFF9B	RESERVED
Port H	X:$FFFF9A	PORT H CONTROL REGISTER (PCRH)
	X:$FFFF99	PORT H DIRECTION REGISTER (PRRH)
	X:$FFFF98	PORT H GPIO DATA REGISTER (PDRH)
SHI	X:$FFFF97	RESERVED
	X:$FFFF96	RESERVED
	X:$FFFF95	RESERVED
	X:$FFFF94	SHI RECEIVE FIFO (HRX)
	X:$FFFF93	SHI TRANSMIT REGISTER (HTX)
	X:$FFFF92	SHI I^2C SLAVE ADDRESS REGISTER (HSAR)
	X:$FFFF91	SHI CONTROL/STATUS REGISTER (HCSR)
	X:$FFFF90	SHI CLOCK CONTROL REGISTER (HCKR)

Table C-1. Internal I/O Memory Map (X Memory) (continued)

Peripheral	Address	Register Name
TRIPLE TIMER	X:$FFFF8F	TIMER 0 CONTROL/STATUS REGISTER (TCSR0)
	X:$FFFF8E	TIMER 0 LOAD REGISTER (TLR0)
	X:$FFFF8D	TIMER 0 COMPARE REGISTER (TCPR0)
	X:$FFFF8C	TIMER 0 COUNT REGISTER (TCR0)
	X:$FFFF8B	TIMER 1 CONTROL/STATUS REGISTER (TCSR1)
	X:$FFFF8A	TIMER 1 LOAD REGISTER (TLR1)
	X:$FFFF89	TIMER 1 COMPARE REGISTER (TCPR1)
	X:$FFFF88	TIMER 1 COUNT REGISTER (TCR1)
	X:$FFFF87	TIMER 2 CONTROL/STATUS REGISTER (TCSR2)
	X:$FFFF86	TIMER 2 LOAD REGISTER (TLR2)
	X:$FFFF85	TIMER 2 COMPARE REGISTER (TCPR2)
	X:$FFFF84	TIMER 2 COUNT REGISTER (TCR2)
	X:$FFFF83	TIMER PRESCALER LOAD REGISTER (TPLR)
	X:$FFFF82	TIMER PRESCALER COUNT REGISTER (TPCR)
	X:$FFFF81	RESERVED
	X:$FFFF80	RESERVED

Table C-2. Internal I/O Memory Map (Y Memory)

Peripheral	Address	Register Name
RESERVED	Y:$FFFFFF	RESERVED
	Y:$FFFFFE	RESERVED
	Y:$FFFFFD	RESERVED
	Y:$FFFFFC	RESERVED
	Y:$FFFFFB	RESERVED
Port G	Y:$FFFFFA	Port G CONTROL REGISTER (PCRG)
	Y:$FFFFF9	Port G DIRECTION REGISTER (PRRG)
	Y:$FFFFF8	Port G GPIO DATA REGISTER (PDRG)
RESERVED	Y:$FFFFF7	RESERVED
	Y:$FFFFF6	RESERVED
	Y:$FFFFF5	RESERVED
	Y:$FFFFF4	RESERVED
	Y:$FFFFF3	RESERVED
	Y:$FFFFF2	RESERVED
	Y:$FFFFF1	RESERVED
	Y:$FFFFF0	RESERVED
RESERVED	Y:$FFFFEF	RESERVED
	Y:$FFFFEE	RESERVED
	Y:$FFFFED	RESERVED
	Y:$FFFFEC	RESERVED
	Y:$FFFFEB	RESERVED
	Y:$FFFFEA	RESERVED
	Y:$FFFFE9	RESERVED
	Y:$FFFFE8	RESERVED
	Y:$FFFFE7	RESERVED
	Y:$FFFFE6	RESERVED
	Y:$FFFFE5	RESERVED

Table C-2. Internal I/O Memory Map (Y Memory) (continued)

Peripheral	Address	Register Name
	Y:$FFFFE4	RESERVED
	Y:$FFFFE3	RESERVED
	Y:$FFFFE2	RESERVED
	Y:$FFFFE1	RESERVED
	Y:$FFFFE0	RESERVED
	Y:$FFFFDF	RESERVED
	Y:$FFFFDE	RESERVED
	Y:$FFFFDD	RESERVED
	Y:$FFFFDC	RESERVED
	Y:$FFFFDB	RESERVED
	Y:$FFFFDA	RESERVED
	Y:$FFFFD9	RESERVED
	Y:$FFFFD8	RESERVED
	Y:$FFFFD7	RESERVED
	Y:$FFFFD6	RESERVED
	Y:$FFFFD5	RESERVED
	Y:$FFFFD4	RESERVED
	Y:$FFFFD3	RESERVED
	Y:$FFFFD2	RESERVED
	Y:$FFFFD1	RESERVED
	Y:$FFFFD0	RESERVED
	Y:$FFFFCF	RESERVED
	Y:$FFFFCE	RESERVED
	Y:$FFFFCD	RESERVED
	Y:$FFFFCC	RESERVED
	Y:$FFFFCB	RESERVED
	Y:$FFFFCA	RESERVED
	Y:$FFFFC9	RESERVED
	Y:$FFFFC8	RESERVED
	Y:$FFFFC7	RESERVED
	Y:$FFFFC6	RESERVED
	Y:$FFFFC5	RESERVED
	Y:$FFFFC4	RESERVED
Watchdog Timer	Y:$FFFFC3	Watchdog Service Register (WSR)
	Y:$FFFFC2	Watchdog Count Register (WCNTR)
	Y:$FFFFC1	Watchdog Modulus Register (WMR)
	Y:$FFFFC0	Watchdog Control Register (WCR)
RESERVED	Y:$FFFFBF	RESERVED
	Y:$FFFFBE	RESERVED
	Y:$FFFFBD	RESERVED
	Y:$FFFFBC	RESERVED
	Y:$FFFFBB	RESERVED
	Y:$FFFFBA	RESERVED
	Y:$FFFFB9	RESERVED
	Y:$FFFFB8	RESERVED
	Y:$FFFFB7	RESERVED

Table C-2. Internal I/O Memory Map (Y Memory) (continued)

Peripheral	Address	Register Name
	Y:$FFFFB6	RESERVED
	Y:$FFFFB5	RESERVED
	Y:$FFFFB4	RESERVED
	Y:$FFFFB3	RESERVED
	Y:$FFFFB2	RESERVED
	Y:$FFFFB1	RESERVED
	Y:$FFFFB0	RESERVED
Patch Module	Y:$FFFFAF	Patch Instruction 7
	Y:$FFFFAE	Patch Instruction 6
	Y:$FFFFAD	Patch Instruction 5
	Y:$FFFFAC	Patch Instruction 4
	Y:$FFFFAB	Patch Instruction 3
	Y:$FFFFAA	Patch Instruction 2
	Y:$FFFFA9	Patch Instruction 1
	Y:$FFFFA8	Patch Instruction 0
	Y:$FFFFA7	Patch Address 7
	Y:$FFFFA6	Patch Address 6
	Y:$FFFFA5	Patch Address 5
	Y:$FFFFA4	Patch Address 4
	Y:$FFFFA3	Patch Address 3
	Y:$FFFFA2	Patch Address 2
	Y:$FFFFA1	Patch Address 1
	Y:$FFFFA0	Patch Address 0
PORT E	Y:$FFFF9F	PORT E CONTROL REGISTER (PCRE)
	Y:$FFFF9E	PORT E DIRECTION REGISTER(PRRE)
	Y:$FFFF9D	PORT E GPIO DATA REGISTER(PDRE)

Table C-2. Internal I/O Memory Map (Y Memory) (continued)

Peripheral	Address	Register Name
ESAI_1	Y:$FFFF9C	ESAI_1 RECEIVE SLOT MASK REGISTER B (RSMB_1)
	Y:$FFFF9B	ESAI_1 RECEIVE SLOT MASK REGISTER A (RSMA_1)
	Y:$FFFF9A	ESAI_1 TRANSMIT SLOT MASK REGISTER B (TSMB_1)
	Y:$FFFF99	ESAI_1 TRANSMIT SLOT MASK REGISTER A (TSMA_1)
	Y:$FFFF98	ESAI_1 RECEIVE CLOCK CONTROL REGISTER (RCCR_1)
	Y:$FFFF97	ESAI_1 RECEIVE CONTROL REGISTER (RCR_1)
	Y:$FFFF96	ESAI_1 TRANSMIT CLOCK CONTROL REGISTER (TCCR_1)
	Y:$FFFF95	ESAI_1 TRANSMIT CONTROL REGISTER (TCR_1)
	Y:$FFFF94	ESAI_1 COMMON CONTROL REGISTER (SAICR_1)
	Y:$FFFF93	ESAI_1 STATUS REGISTER (SAISR_1)
	Y:$FFFF92	RESERVED
	Y:$FFFF91	RESERVED
	Y:$FFFF90	RESERVED
	Y:$FFFF8F	RESERVED
	Y:$FFFF8E	RESERVED
	Y:$FFFF8D	RESERVED
	Y:$FFFF8C	RESERVED
	Y:$FFFF8B	ESAI_1 RECEIVE DATA REGISTER 3 (RX3_1)
	Y:$FFFF8A	ESAI_1 RECEIVE DATA REGISTER 2 (RX2_1)
	Y:$FFFF89	ESAI_1 RECEIVE DATA REGISTER 1 (RX1_1)
	Y:$FFFF88	ESAI_1 RECEIVE DATA REGISTER 0 (RX0_1)
	Y:$FFFF87	RESERVED
	Y:$FFFF86	ESAI_1 TIME SLOT REGISTER (TSR_1)
	Y:$FFFF85	ESAI_1 TRANSMIT DATA REGISTER 5 (TX5_1)
	Y:$FFFF84	ESAI_1 TRANSMIT DATA REGISTER 4 (TX4_1)
	Y:$FFFF83	ESAI_1 TRANSMIT DATA REGISTER 3 (TX3_1)
	Y:$FFFF82	ESAI_1 TRANSMIT DATA REGISTER 2 (TX2_1)
	Y:$FFFF81	ESAI_1 TRANSMIT DATA REGISTER 1 (TX1_1)
	Y:$FFFF80	ESAI_1 TRANSMIT DATA REGISTER 0 (TX0_1)

C.1.6 Interrupt Vector Addresses

Table C-3. DSP56374 Interrupt Vectors

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$00	3	Hardware $\overline{\text{RESET}}$
VBA:$02	3	Stack Error
VBA:$04	3	Illegal Instruction
VBA:$06	3	Debug Request Interrupt
VBA:$08	3	Trap
VBA:$0A	3	Non-maskable Interrupt ($\overline{\text{NMI}}$)
VBA:$0C	3	Reserved For Future Level-3 Interrupt Source
VBA:$0E	3	Reserved For Future Level-3 Interrupt Source
VBA:$10	0 - 2	IRQA

Table C-3. DSP56374 Interrupt Vectors (continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$12	0 - 2	IRQB
VBA:$14	0 - 2	IRQC
VBA:$16	0 - 2	IRQD
VBA:$18	0 - 2	DMA Channel 0
VBA:$1A	0 - 2	DMA Channel 1
VBA:$1C	0 - 2	DMA Channel 2
VBA:$1E	0 - 2	DMA Channel 3
VBA:$20	0 - 2	DMA Channel 4
VBA:$22	0 - 2	DMA Channel 5
VBA:$24	0 - 2	RESERVED
VBA:$26	0 - 2	RESERVED
VBA:$28	0 - 2	RESERVED
VBA:$2A	0 - 2	RESERVED
VBA:$2C	0 - 2	RESERVED
VBA:$2E	0 - 2	RESERVED
VBA:$30	0 - 2	ESAI Receive Data
VBA:$32	0 - 2	ESAI Receive Even Data
VBA:$34	0 - 2	ESAI Receive Data With Exception Status
VBA:$36	0 - 2	ESAI Receive Last Slot
VBA:$38	0 - 2	ESAI Transmit Data
VBA:$3A	0 - 2	ESAI Transmit Even Data
VBA:$3C	0 - 2	ESAI Transmit Data with Exception Status
VBA:$3E	0 - 2	ESAI Transmit Last Slot
VBA:$40	0 - 2	SHI Transmit Data
VBA:$42	0 - 2	SHI Transmit Underrun Error
VBA:$44	0 - 2	SHI Receive FIFO Not Empty
VBA:$46	0 - 2	RESERVED
VBA:$48	0 - 2	SHI Receive FIFO Full
VBA:$4A	0 - 2	SHI Receive Overrun Error
VBA:$4C	0 - 2	SHI Bus Error
VBA:$4E	0 - 2	RESERVED
VBA:$50	0 - 2	RESERVED
VBA:$52	0 - 2	RESERVED
VBA:$54	0 - 2	TIMER0 Compare
VBA:$56	0 - 2	TIMER0 Overflow
VBA:$58	0 - 2	TIMER1 Compare

Table C-3. DSP56374 Interrupt Vectors (continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$5A	0 - 2	TIMER1 Overflow
VBA:$5C	0 - 2	TIMER2 Compare
VBA:$5E	0 - 2	TIMER2 Overflow
VBA:$60	0 - 2	RESERVED
VBA:$62	0 - 2	RESERVED
VBA:$64	0 - 2	RESERVED
VBA:$66	0 - 2	RESERVED
VBA:$68	0 - 2	RESERVED
VBA:$6A	0 - 2	RESERVED
VBA:$6C	0 - 2	RESERVED
VBA:$6E	0 - 2	RESERVED
VBA:$70	0 - 2	ESAI_1 Receive Data
VBA:$72	0 - 2	ESAI_1 Receive Even Data
VBA:$74	0 - 2	ESAI_1 Receive Data With Exception Status
VBA:$76	0 - 2	ESAI_1 Receive Last Slot
VBA:$78	0 - 2	ESAI_1 Transmit Data
VBA:$7A	0 - 2	ESAI_1 Transmit Even Data
VBA:$7C	0 - 2	ESAI_1 Transmit Data with Exception Status
VBA:$7E	0 - 2	ESAI_1 Transmit Last Slot
VBA:$80	0 - 2	RESERVED
VBA:$82	0 - 2	RESERVED
VBA:$84	0 - 2	RESERVED
VBA:$86	0 - 2	RESERVED
VBA:$88	0 - 2	RESERVED
VBA:$8A	0 - 2	RESERVED
VBA:$8C	0 - 2	RESERVED
VBA:$8E	0 - 2	RESERVED
VBA:$90	0 - 2	RESERVED
VBA:$92	0 - 2	RESERVED
VBA:$94	0 - 2	RESERVED
VBA:$96	0 - 2	RESERVED
VBA:$98	0 - 2	RESERVED
VBA:$9A	0 - 2	RESERVED
VBA:$9C	0 - 2	RESERVED
VBA:$9E	0 - 2	RESERVED
VBA:$A0	0 - 2	RESERVED

Table C-3. DSP56374 Interrupt Vectors (continued)

Interrupt Starting Address	Interrupt Priority Level Range	Interrupt Source
VBA:$A2	0 - 2	RESERVED
VBA:$A4	0 - 2	RESERVED
VBA:$A6	0 - 2	RESERVED
VBA:$A8	0 - 2	RESERVED
VBA:$AA	0 - 2	RESERVED
VBA:$AC	0 - 2	RESERVED
VBA:$AE	0 - 2	RESERVED
VBA:$B0	0 - 2	RESERVED
VBA:$B2	0 - 2	RESERVED
VBA:$B4	0 - 2	RESERVED
VBA:$B6	0 - 2	RESERVED
VBA:$B8	0 - 2	RESERVED
VBA:$BA	0 - 2	RESERVED
:	:	:
VBA:$FE	0 - 2	RESERVED

C.2 Interrupt Source Priorities (within an IPL)

Table C-4. Interrupt Sources Priorities Within an IPL

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

Table C-4. Interrupt Sources Priorities Within an IPL (continued)

Priority	Interrupt Source
	DMA Channel 4 Interrupt
	DMA Channel 5 Interrupt
	ESAI Receive Data with Exception Status
	ESAI Receive Even Data
	ESAI Receive Data
	ESAI Receive Last Slot
	ESAI Transmit Data with Exception Status
	ESAI Transmit Last Slot
	ESAI Transmit Even Data
	ESAI Transmit Data
	SHI Bus Error
	SHI Receive Overrun Error
	SHI Transmit Underrun Error
	SHI Receive FIFO Full
	SHI Transmit Data
	SHI Receive FIFO Not Empty
	TIMER0 Overflow Interrupt
	TIMER0 Compare Interrupt
	TIMER1 Overflow Interrupt
	TIMER1 Compare Interrupt
	TIMER2 Overflow Interrupt
	TIMER2 Compare Interrupt
	ESAI_1 Receive Data with Exception Status
	ESAI_1 Receive Even Data
	ESAI_1 Receive Data
	ESAI_1 Receive Last Slot
	ESAI_1 Transmit Data with Exception Status
	ESAI_1 Transmit Last Slot
	ESAI_1 Transmit Even Data
Lowest	ESAI_1 Transmit Data

C.3 Programming Sheets

The worksheets shown on the following pages contain listings of major programmable registers for the DSP56374. The programming sheets are grouped into the following order:

- Central Processor
- Serial Host Interface (SHI)
- Two Enhanced Serial Audio Interfaces (ESAI and ESAI_1)
- Timer/Event Controller (TEC)

- Hardware Watchdog Timer (WDT)
- GPIO (Ports C - H)

Each sheet provides room to write in the value of each bit and the hexadecimal value for each register. Programmers can photocopy these sheets and reuse them for each application development project.

For details on the instruction set of the DSP56300 family chips, see the *DSP56300 Family Manual*.

Application: _____ Date: _____

_____ Programmer: _____

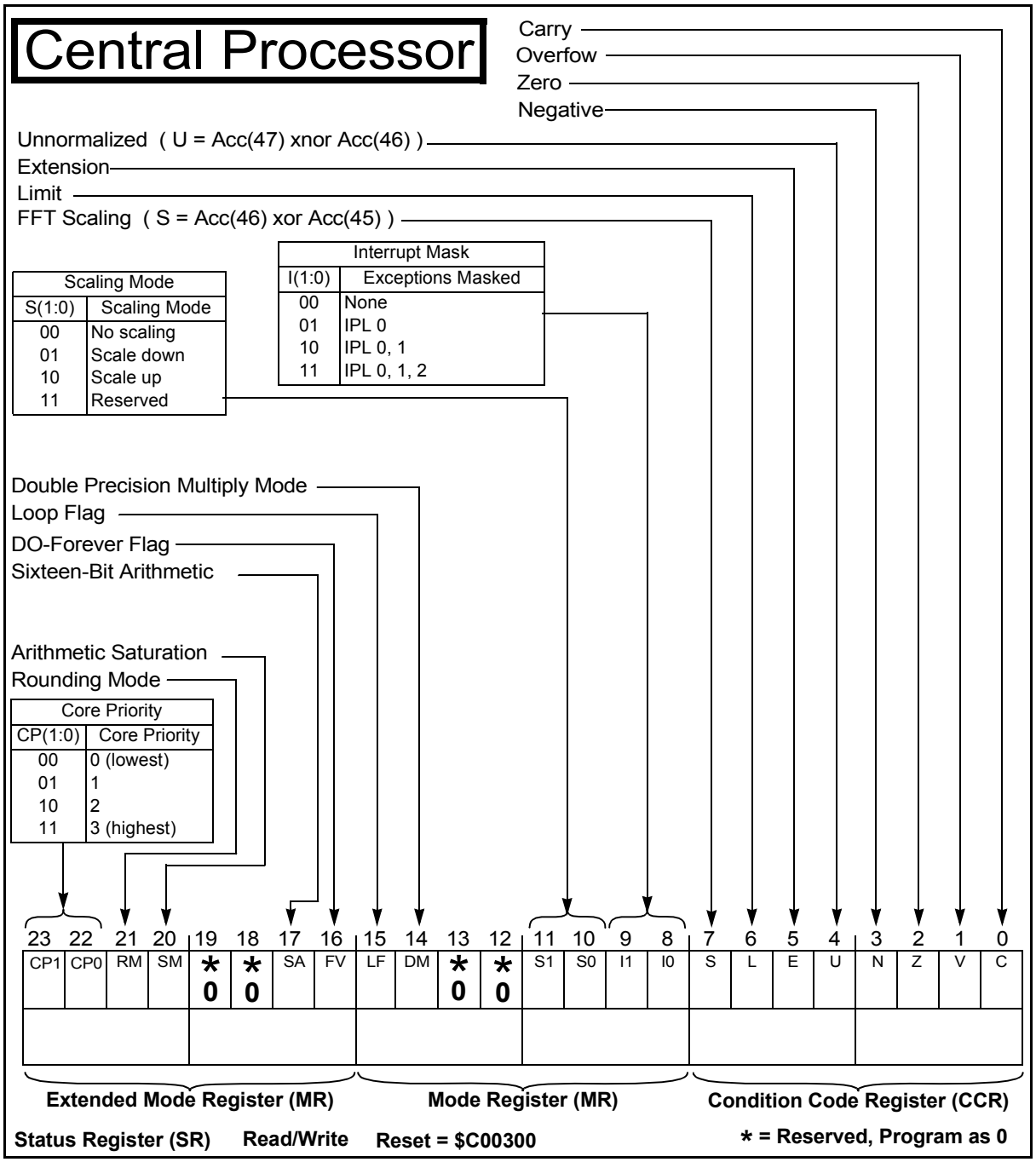


Figure C-1. Status Register (SR)

Application: _____ Date: _____

_____ Programmer: _____

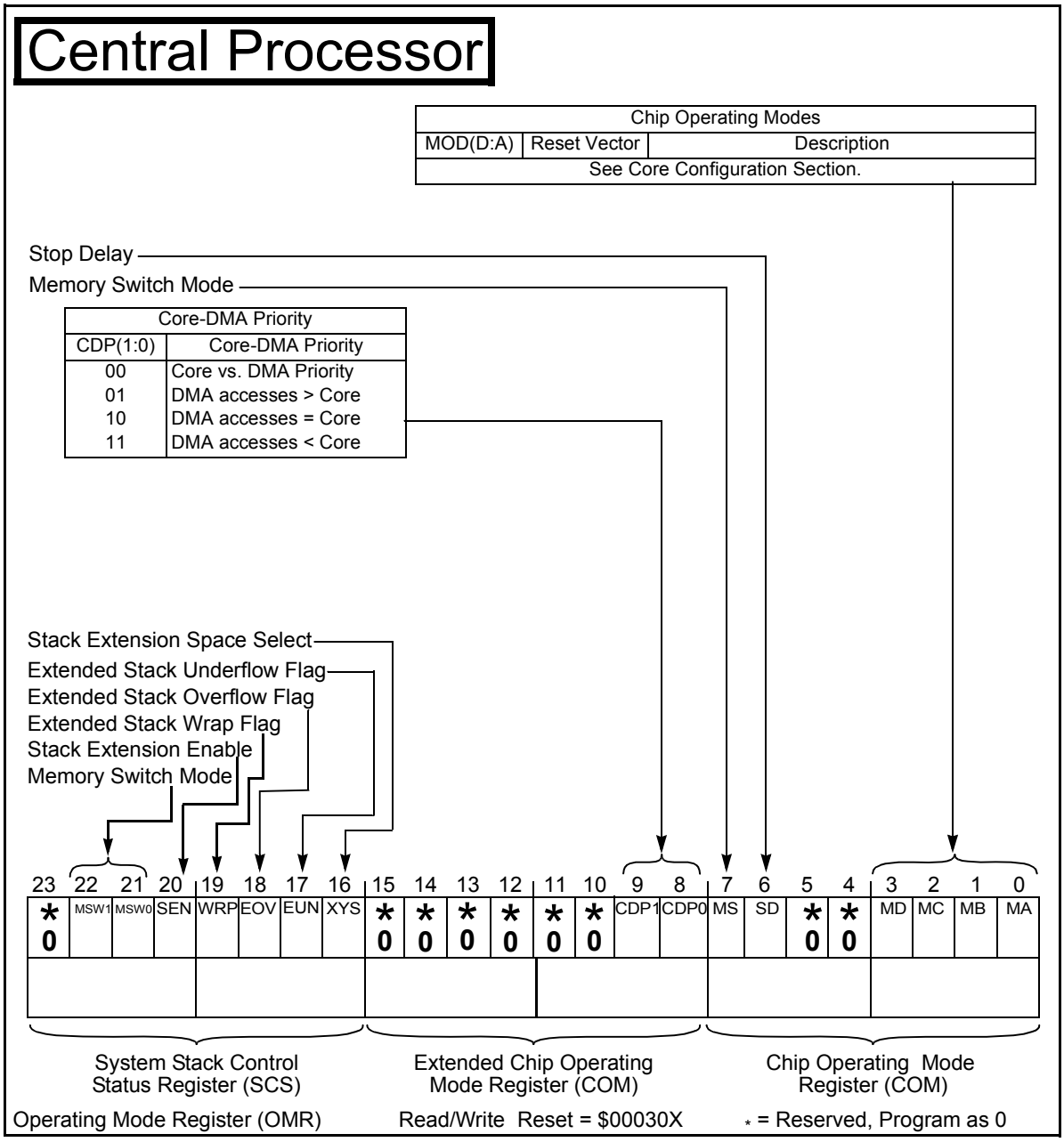


Figure C-2. Operating Mode Register (OMR)

Application: _____ Date: _____

_____ Programmer: _____

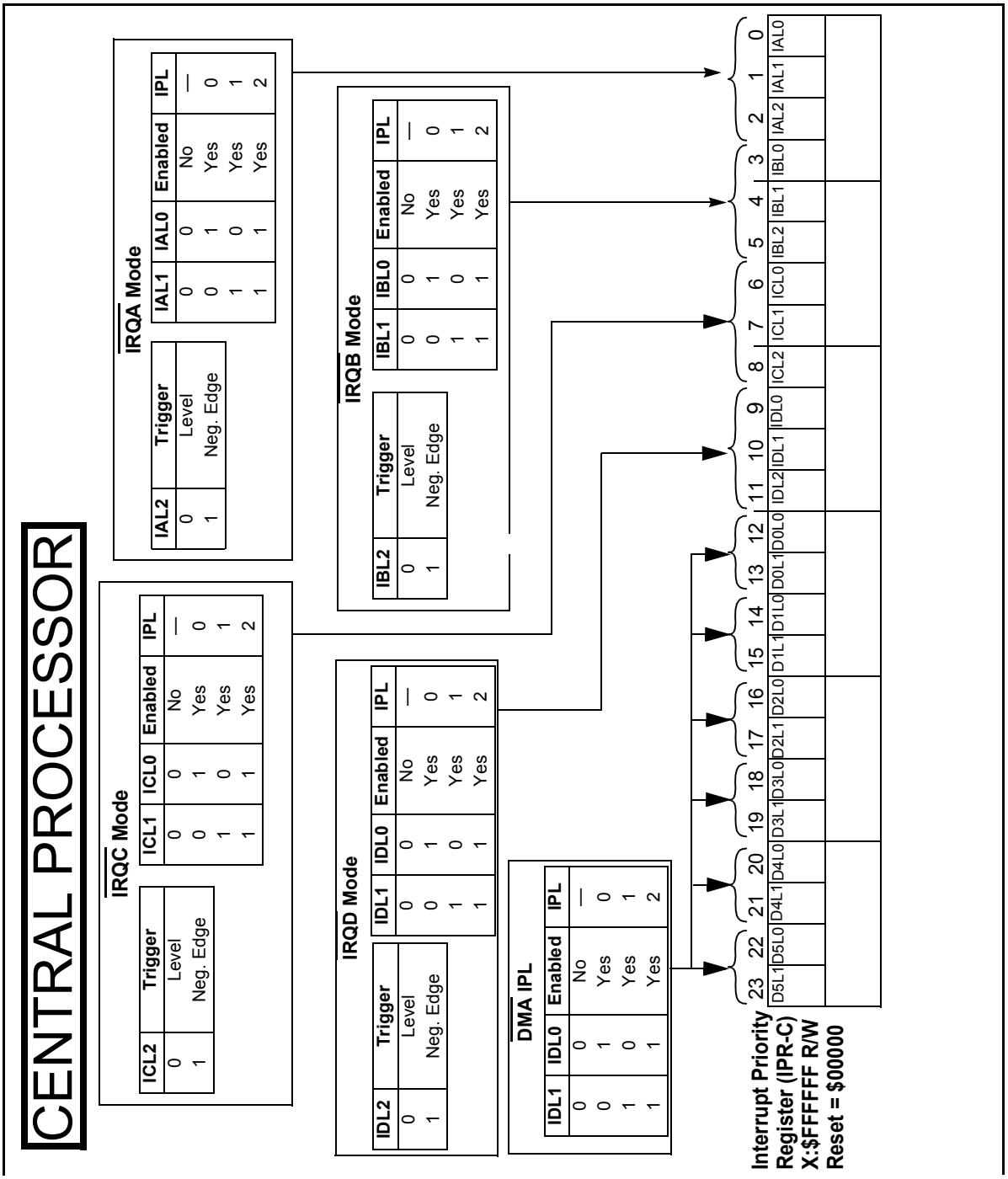


Figure C-3. Interrupt Priority Register–Core (IPR–C)

Application: _____ Date: _____

_____ Programmer: _____

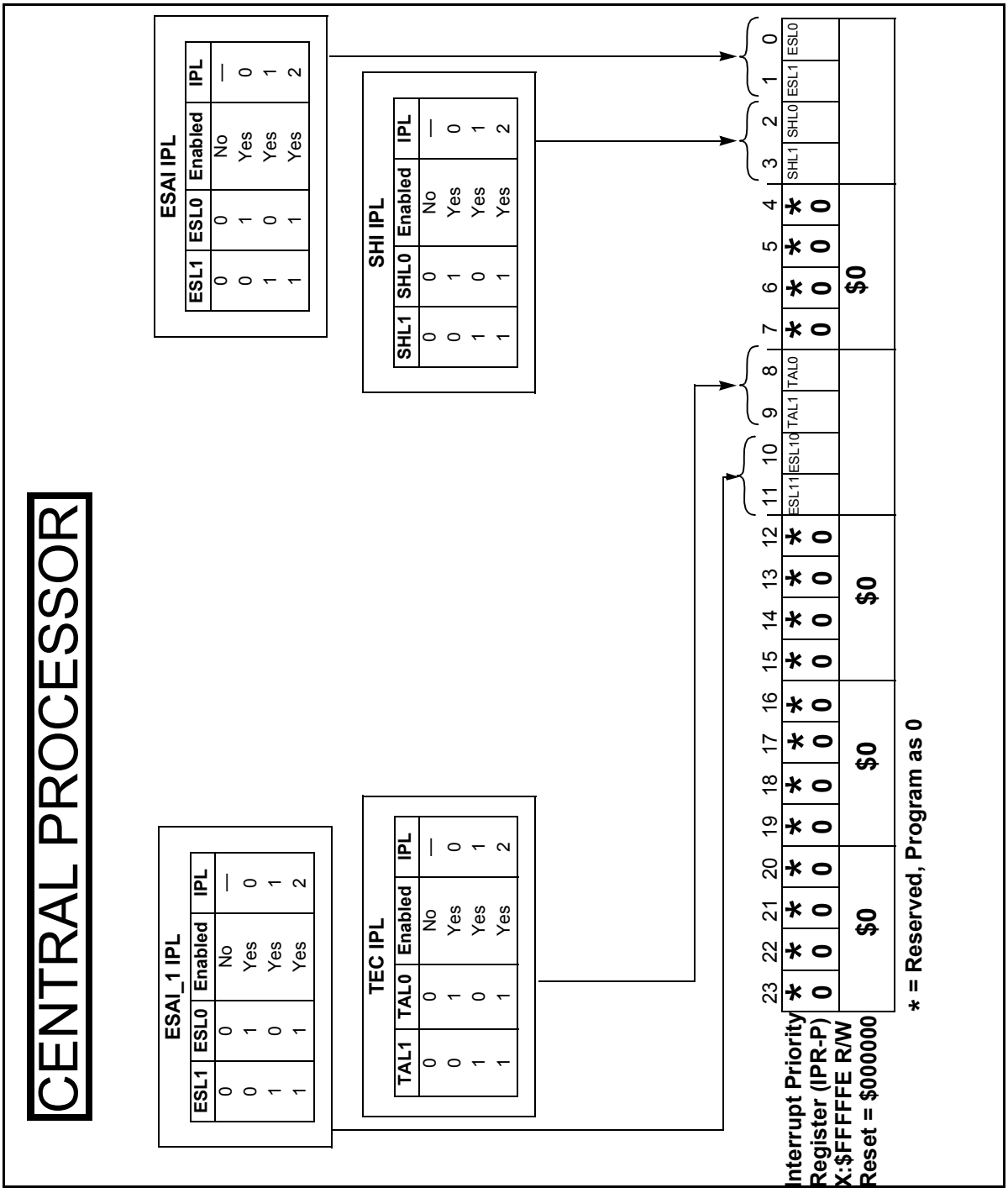
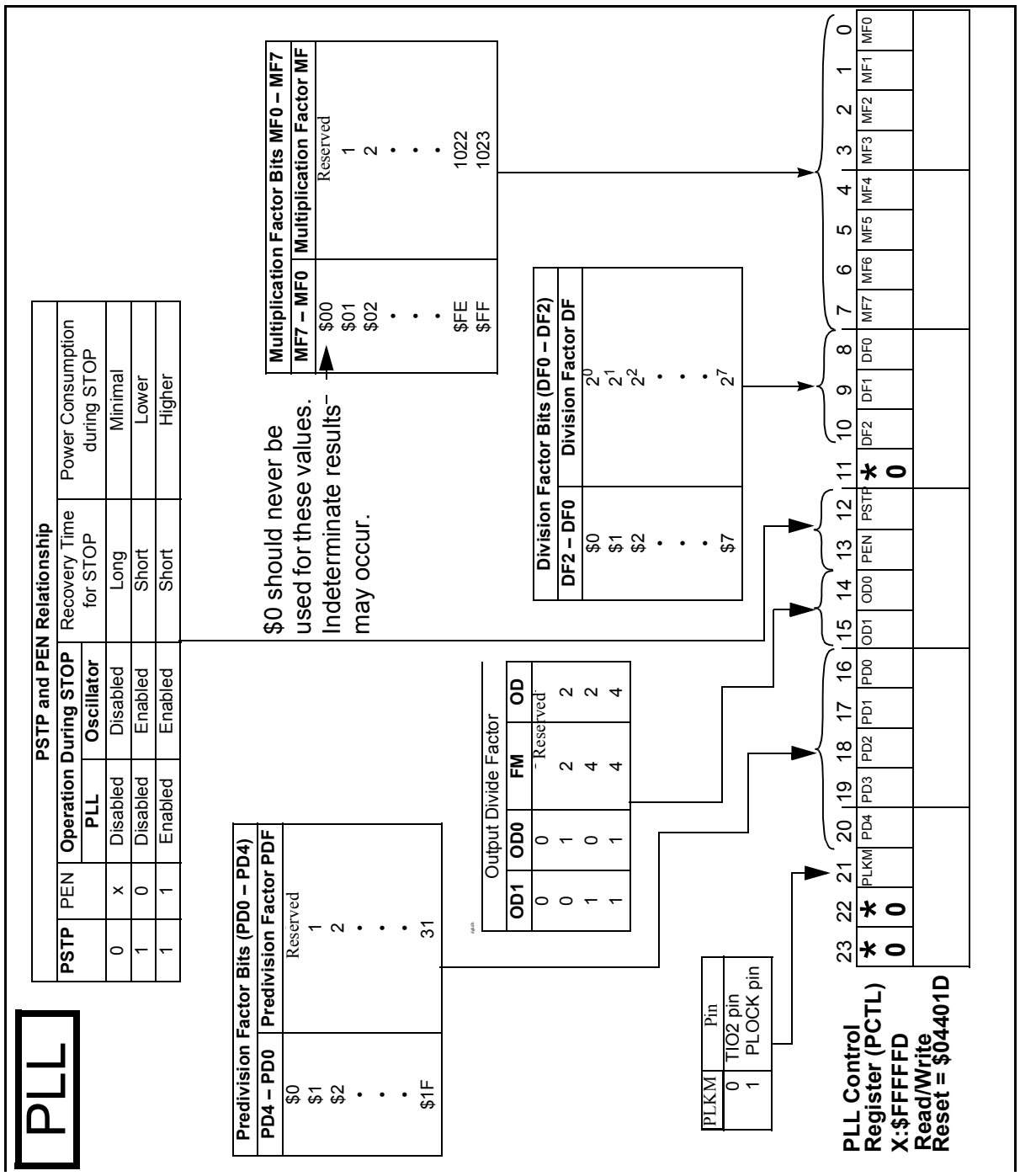


Figure C-4. Interrupt Priority Register – Peripherals (IPR–P)

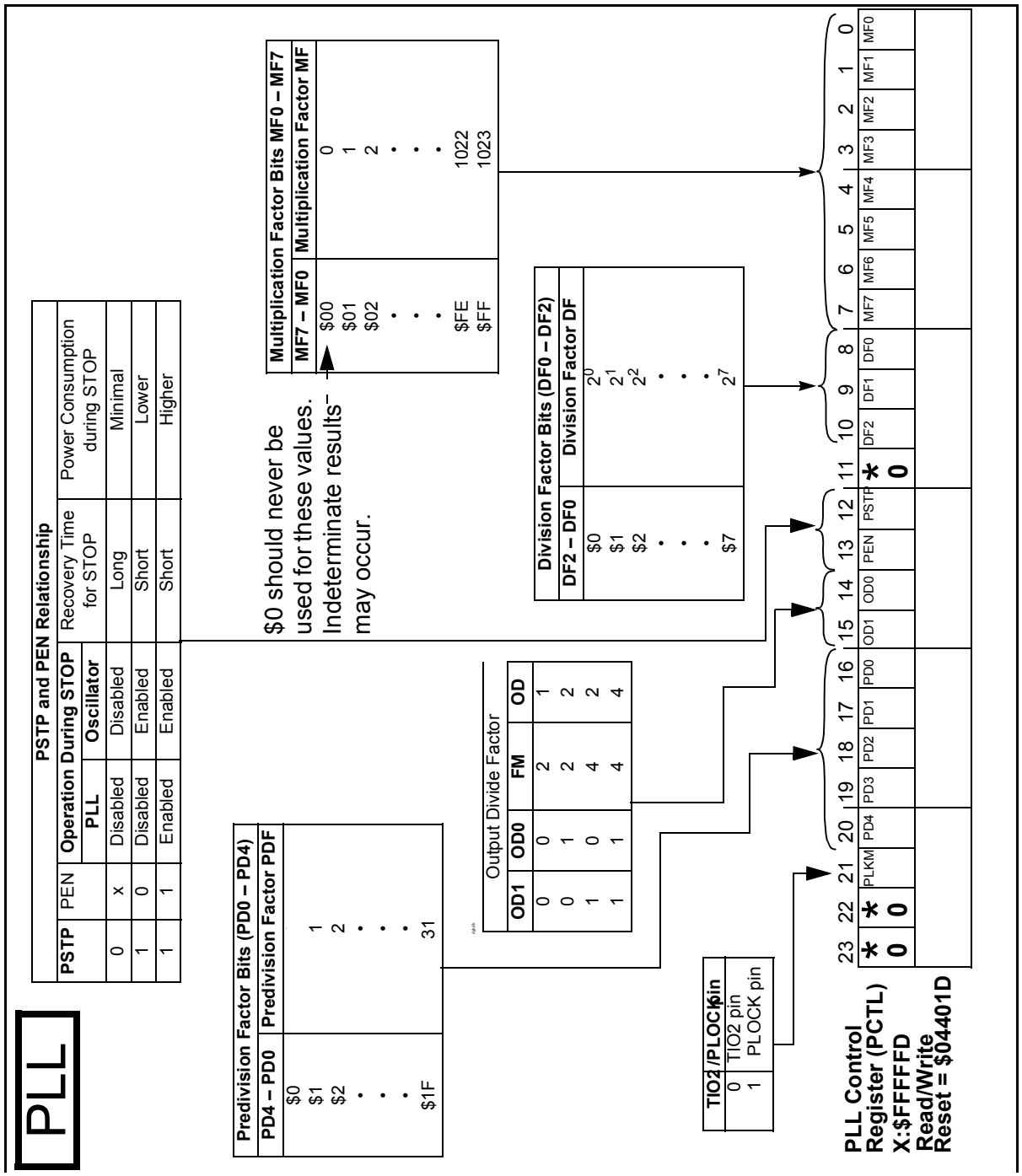
Application: _____ Date: _____

_____ Programmer: _____



Application: _____ Date: _____

_____ Programmer: _____



Application:_____ Date:_____

_____ Programmer:_____

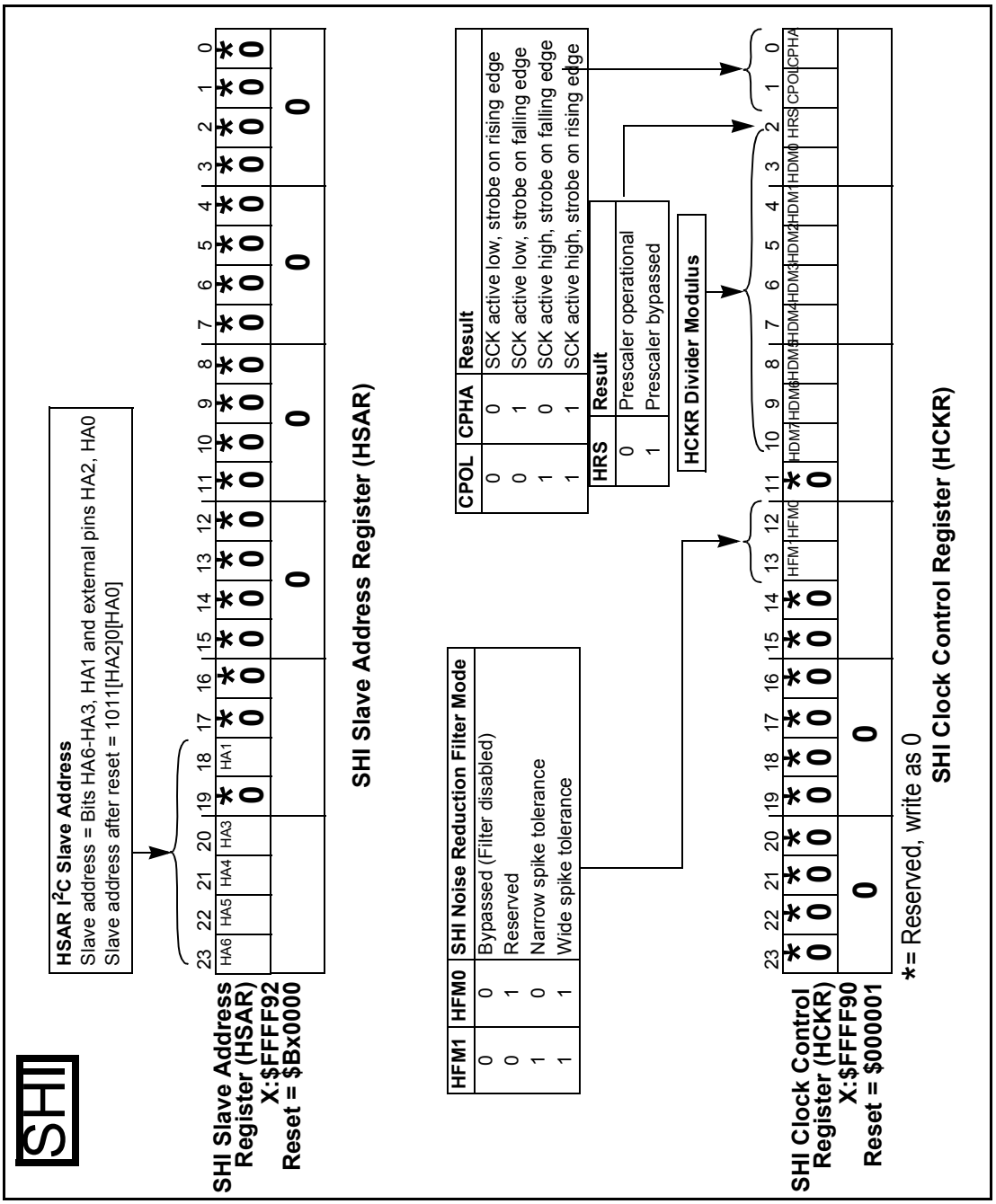


Figure C-5. SHI Slave Address and Clock Control Registers

Application:_____ Date:_____

_____ Programmer:_____

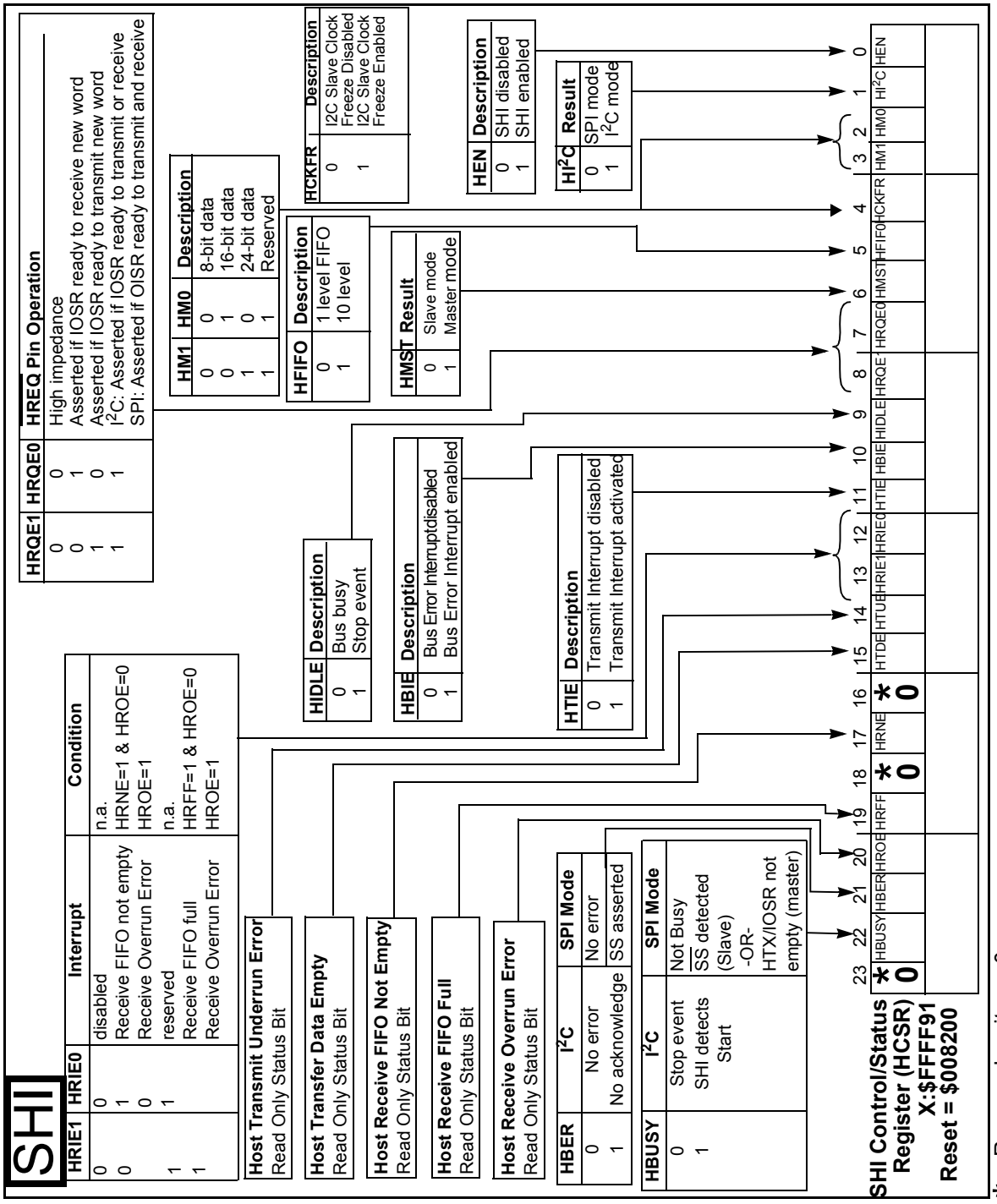


Figure C-6. SHI Host Control/Status Register

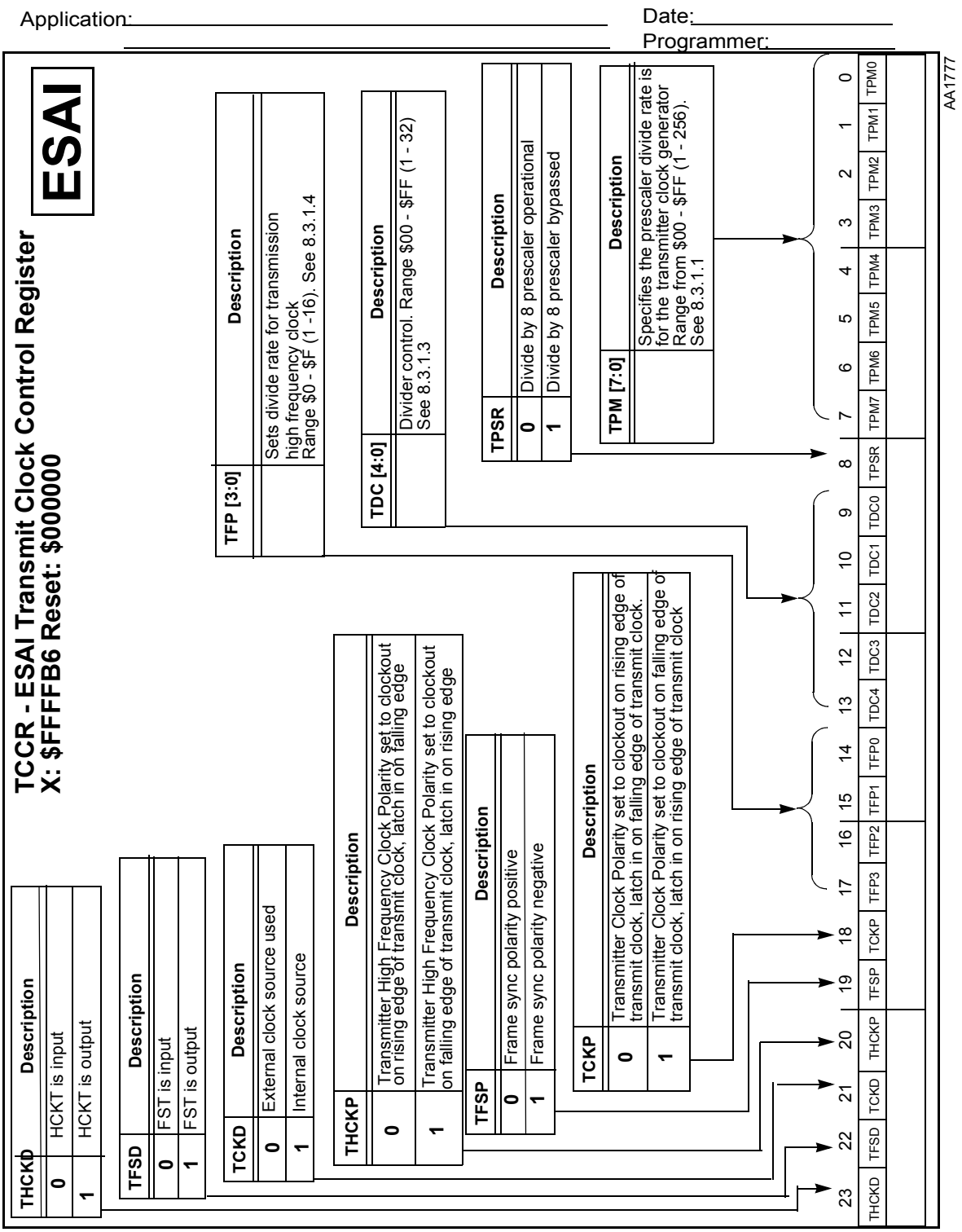


Figure C-7. ESAI Transmit Clock Control Register

Application: _____ Date: _____

Programmer: _____

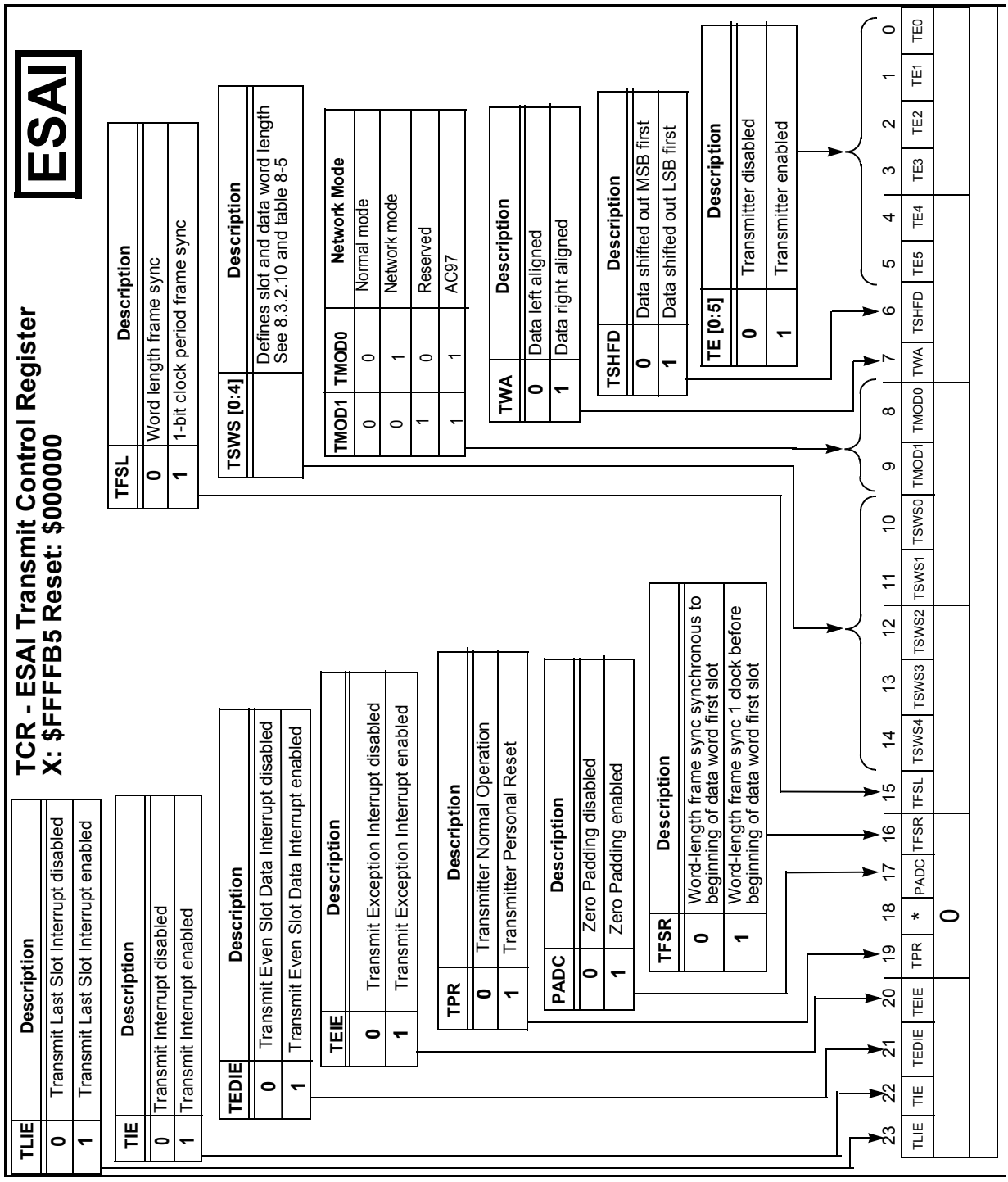


Figure C-8. ESAI Transmit Control Register

Application:_____ Date:_____

Programmer:_____

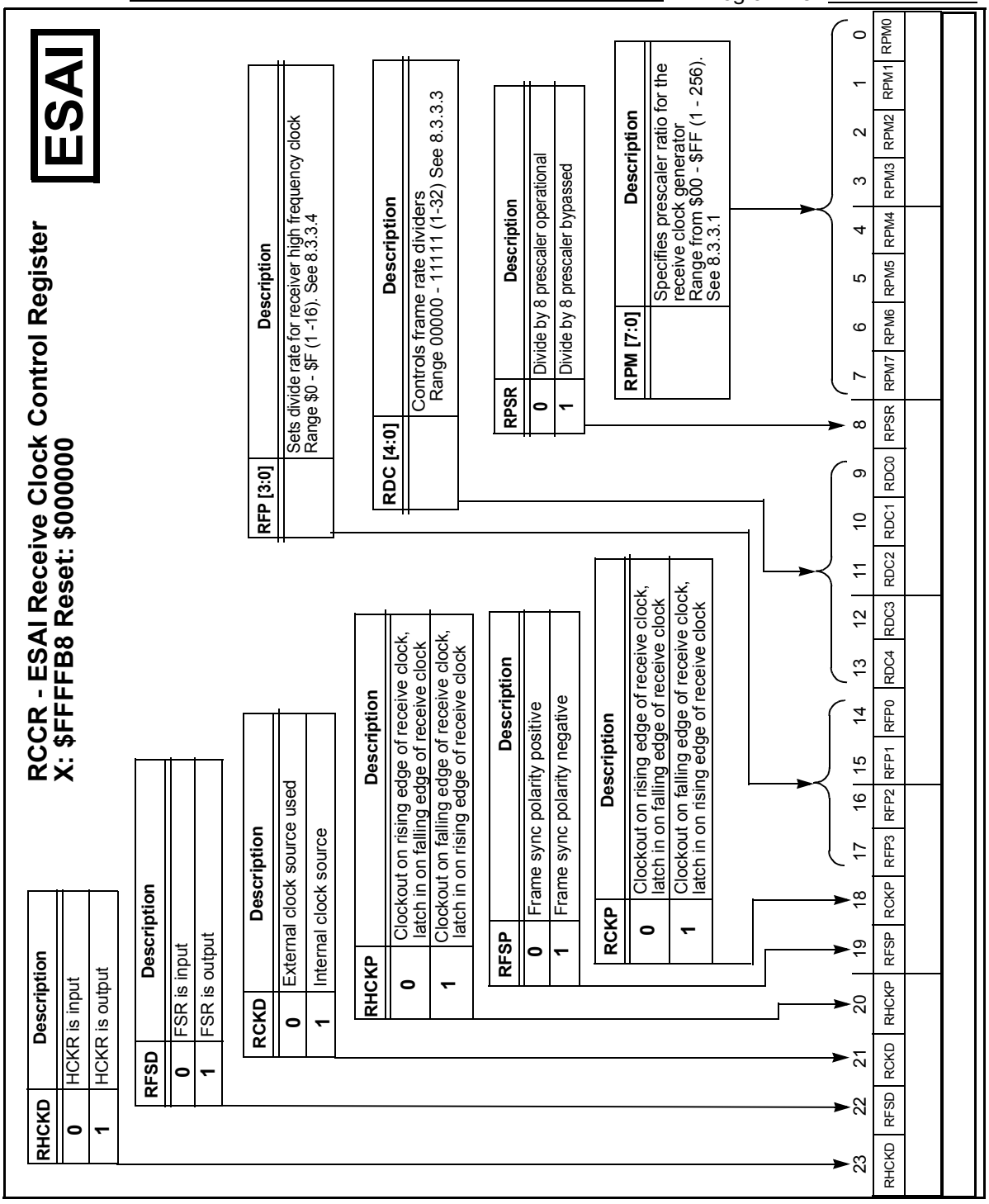


Figure C-9. ESAI Receive Clock Control Register

Application: _____ Date: _____

Programmer: _____

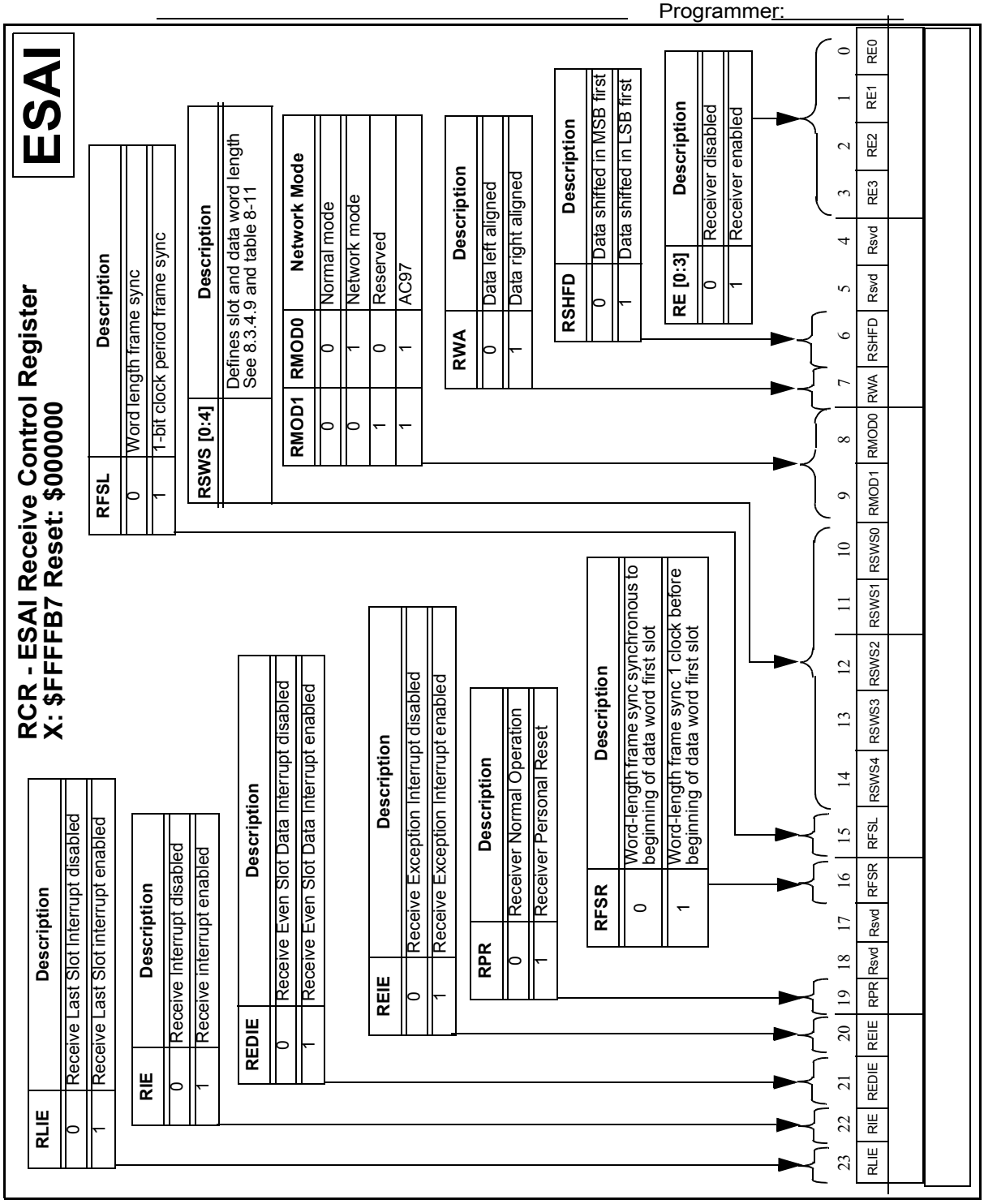


Figure C-10. ESAI Receive Control Register

Application: _____ Date: _____

_____ Programmer: _____

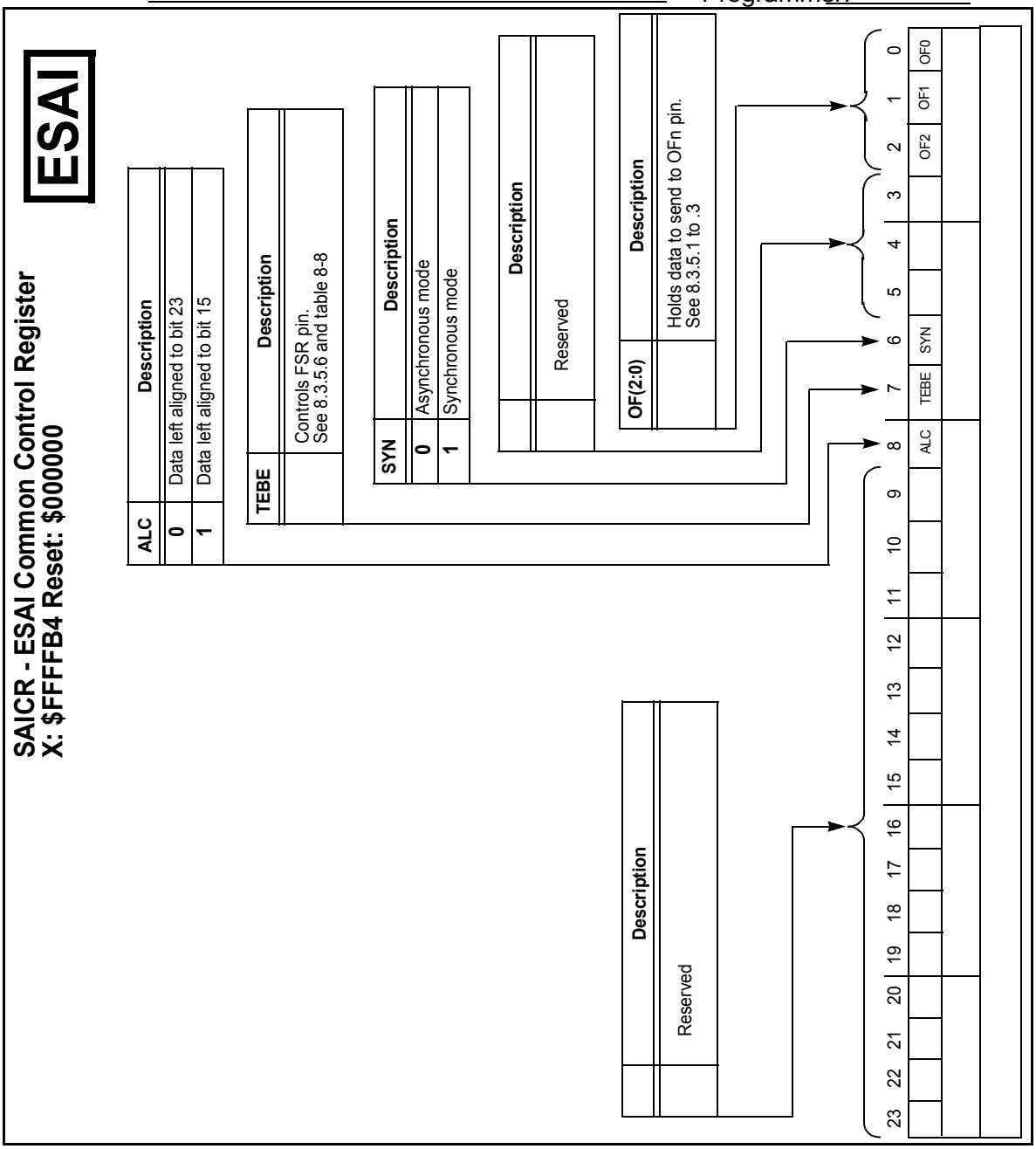


Figure C-11. ESAI Common Control Register

Application: _____ Date: _____

Programmer: _____

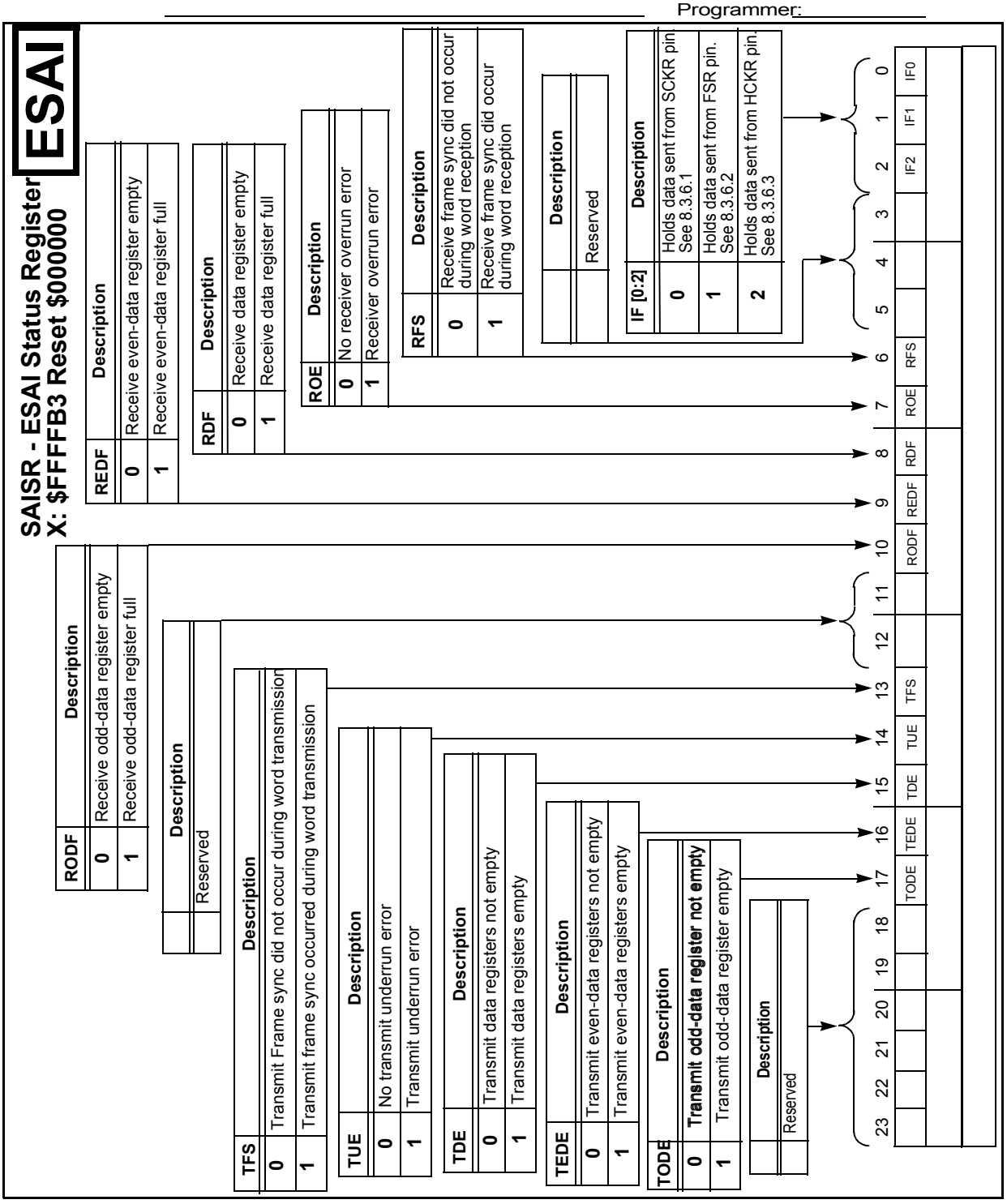


Figure C-12. ESAI Status Register

Application: Date:

Programmer:

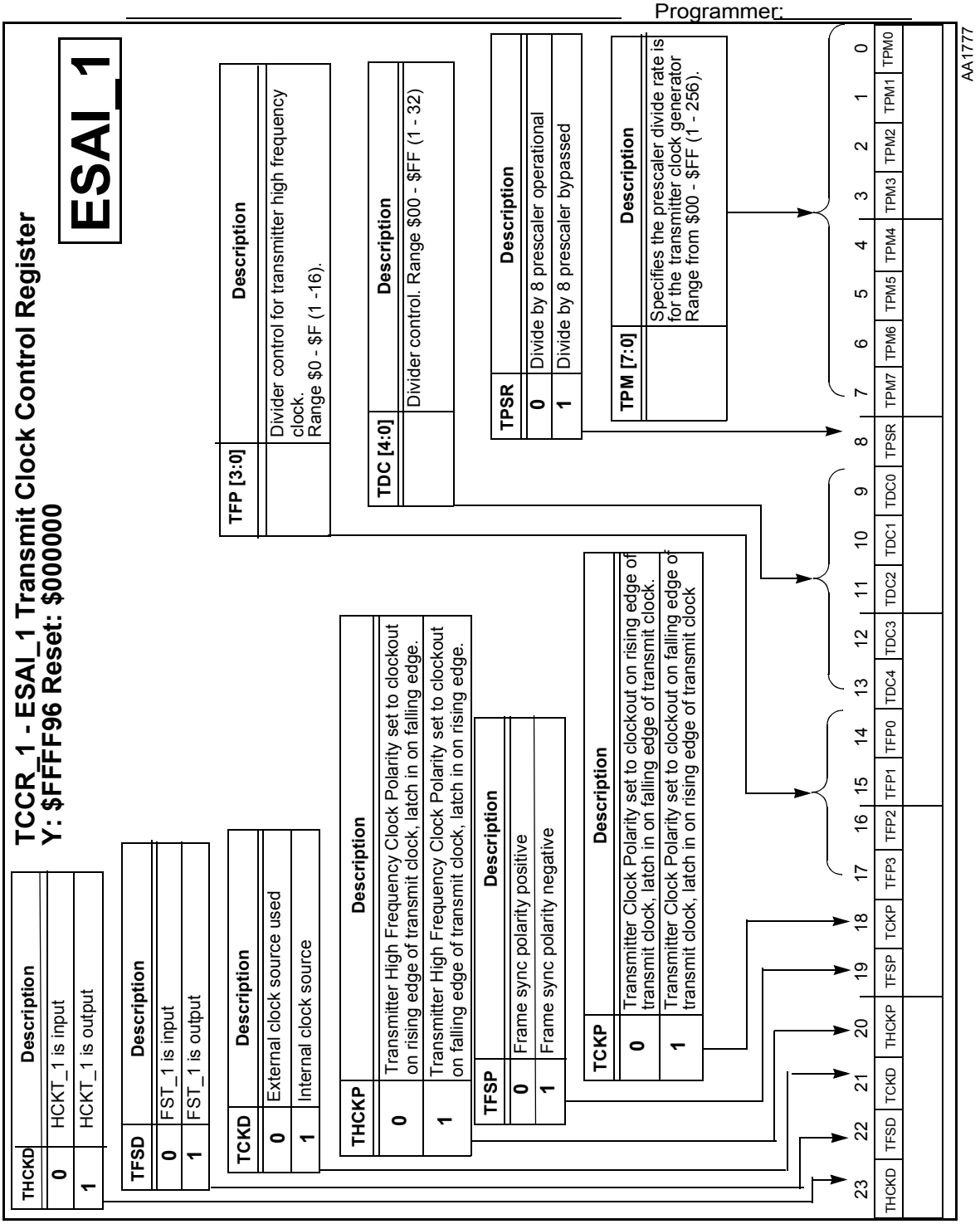


Figure C-13. ESAI_1 Transmit Clock Control Register

Application: _____ Date: _____

_____ Programmer: _____

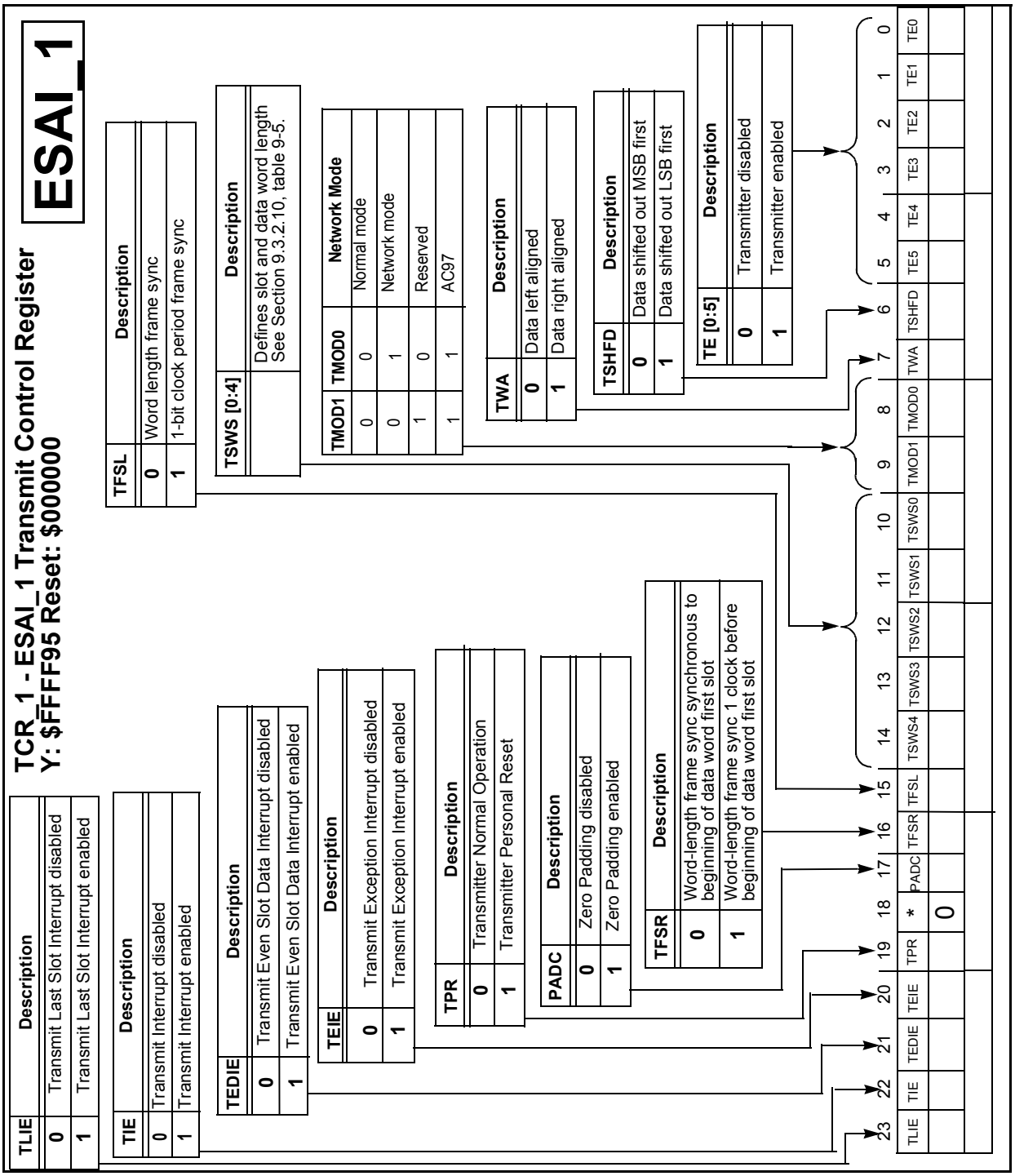


Figure C-14. ESAI_1 Transmit Control Register

Application:_____ Date:_____

Programmer:_____

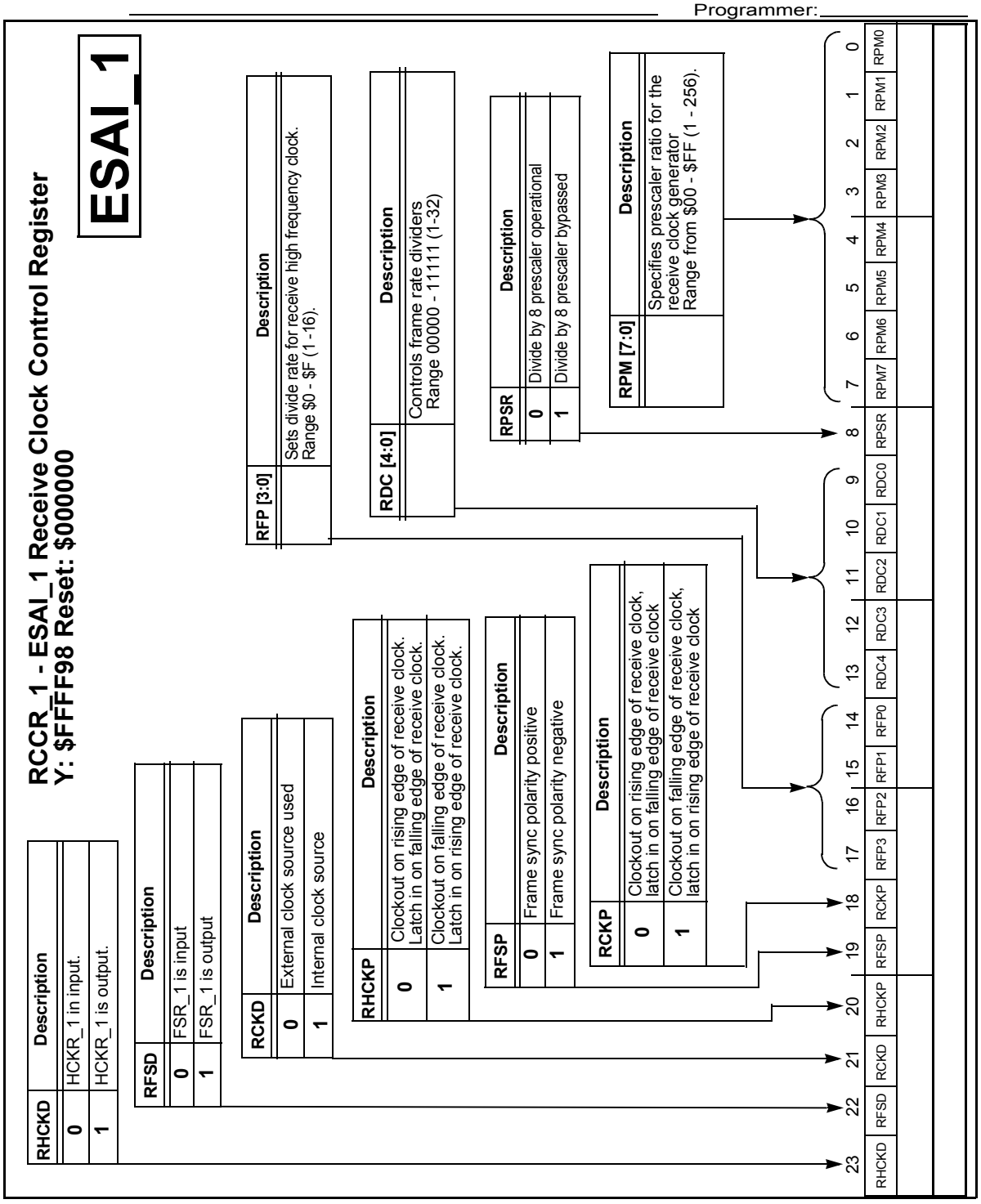


Figure C-15. ESAI_1 Receive Clock Control Register

Application: _____ Date: _____

Programmer: _____

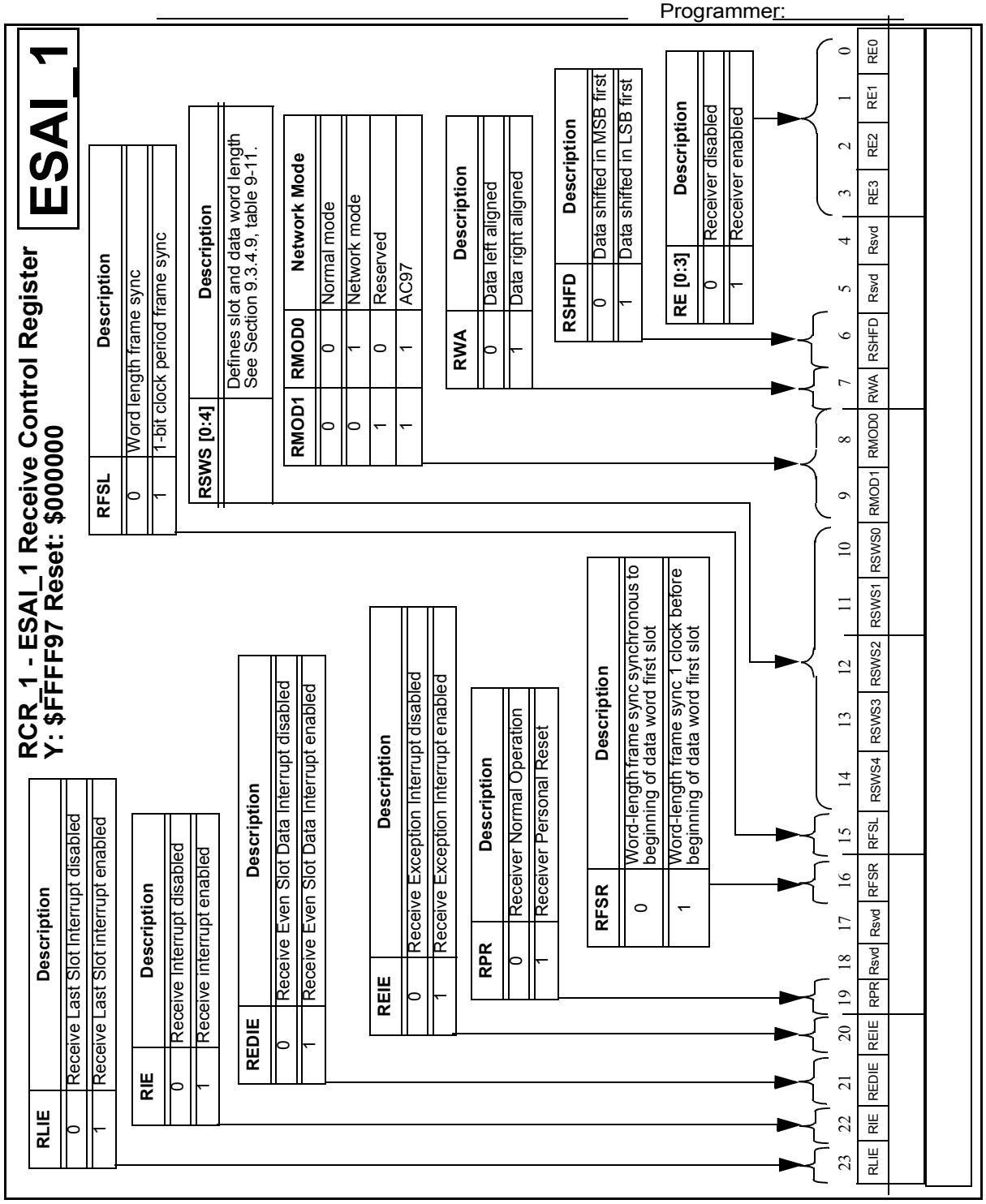


Figure C-16. ESAI_1 Receive Control Register

Application: _____ Date: _____

Programmer: _____

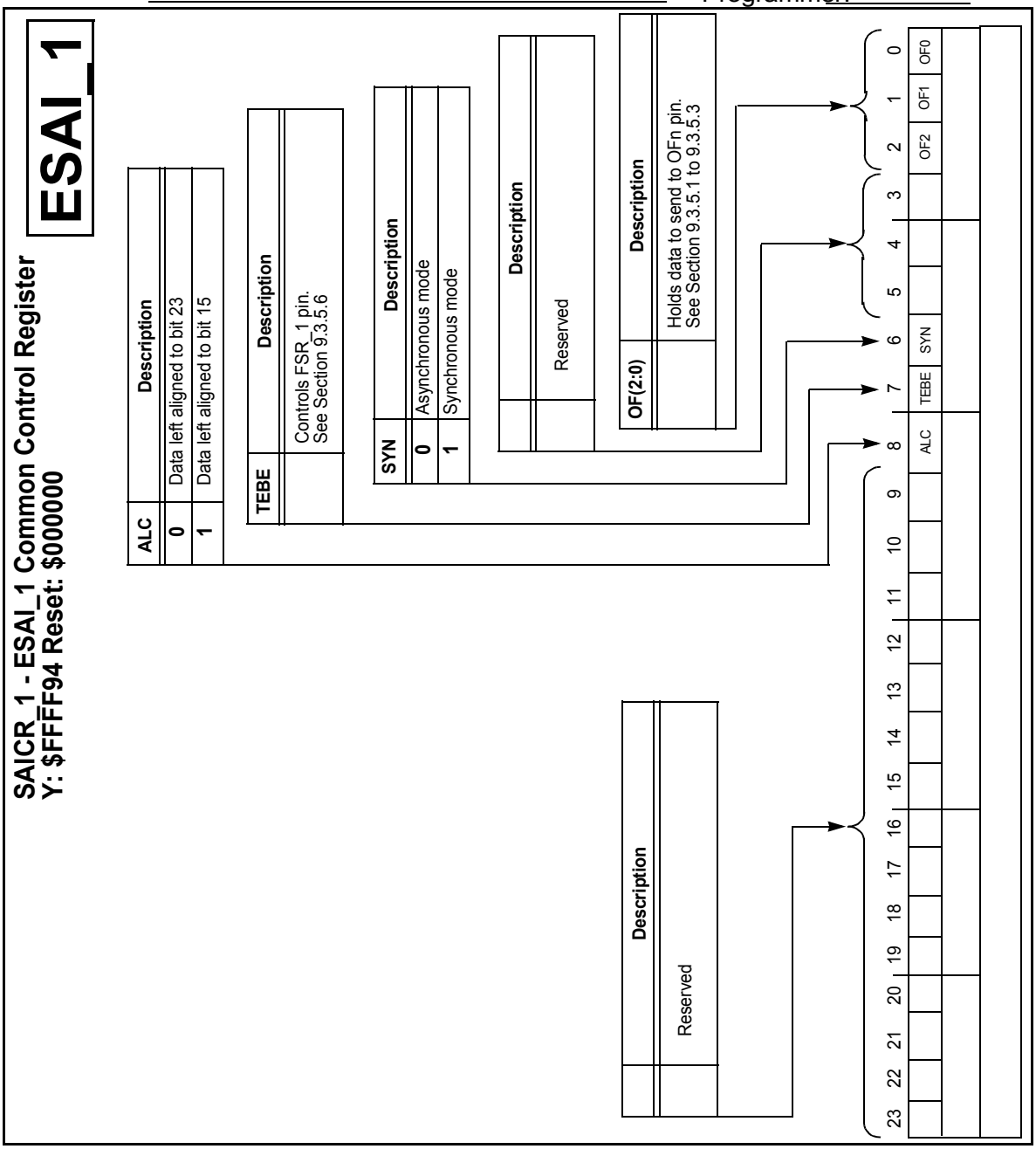


Figure C-17. ESAI_1 Common Control Register

Application: _____ Date: _____

Programmer: _____

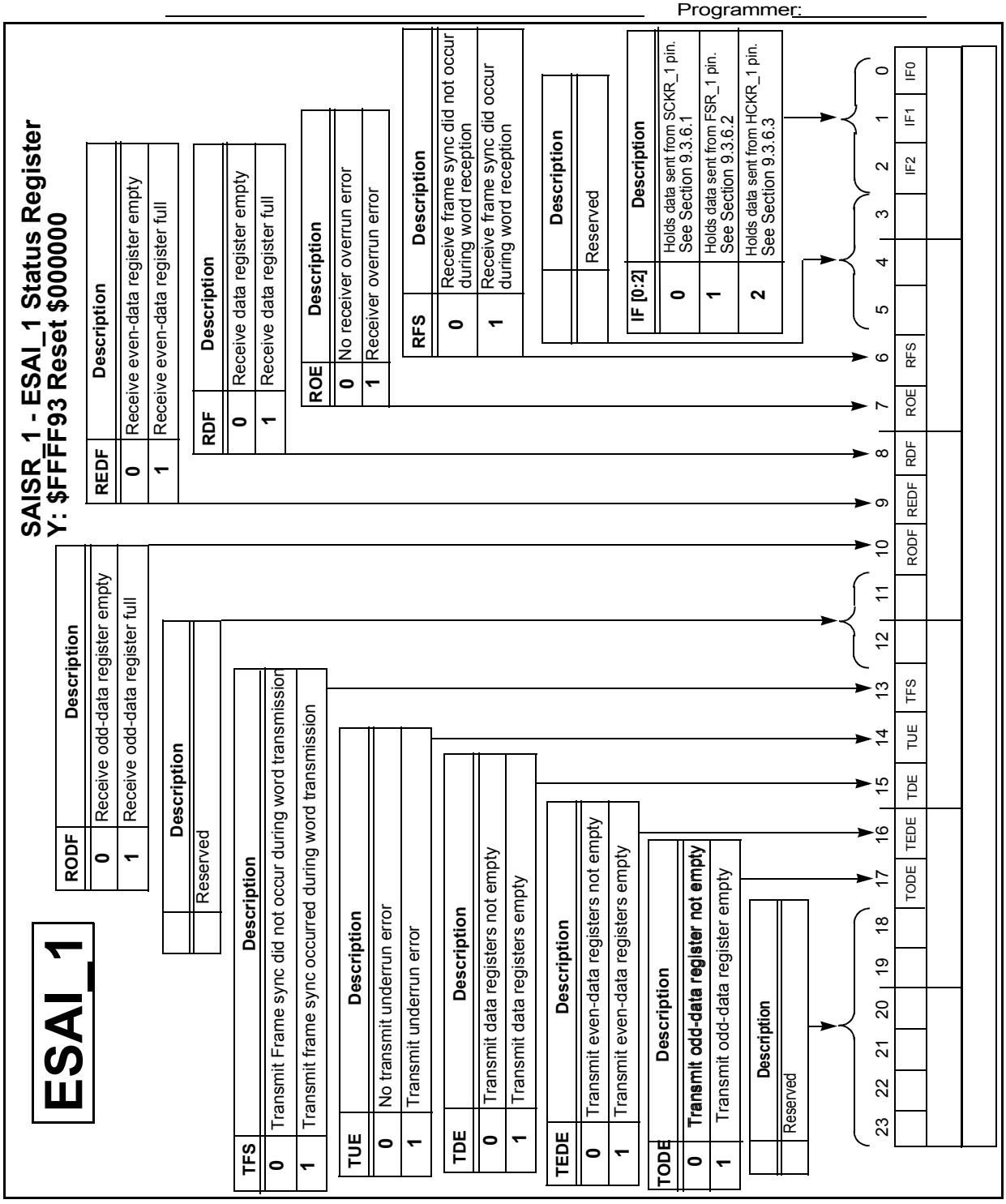


Figure C-18. ESAI_1 Status Register

Application:_____ Date:_____

_____ Programmer:_____

TEC

PS (1:0)	Prescaler Clock Source
00	Internal CLK/2
01	TIO0
10	Reserved
11	Reserved

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	PS1	PS0	Prescaler Preload Value (PL [0:20])																				

Timer Prescaler Load Register
X:$FFFF83 Read/Write
Reset = $000000

* = Reserved, Program as 0

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	Current Value of Prescaler Counter (PC [0:20])																				

Timer Prescaler Count Register
X:$FFFF82 Read Only
Reset = $000000

*= Reserved, Program as 0

Figure C-19. Timer Prescaler Load and Prescaler Count Registers (TPLR, TPCR)

Application: _____ Date: _____

_____ Programmer: _____

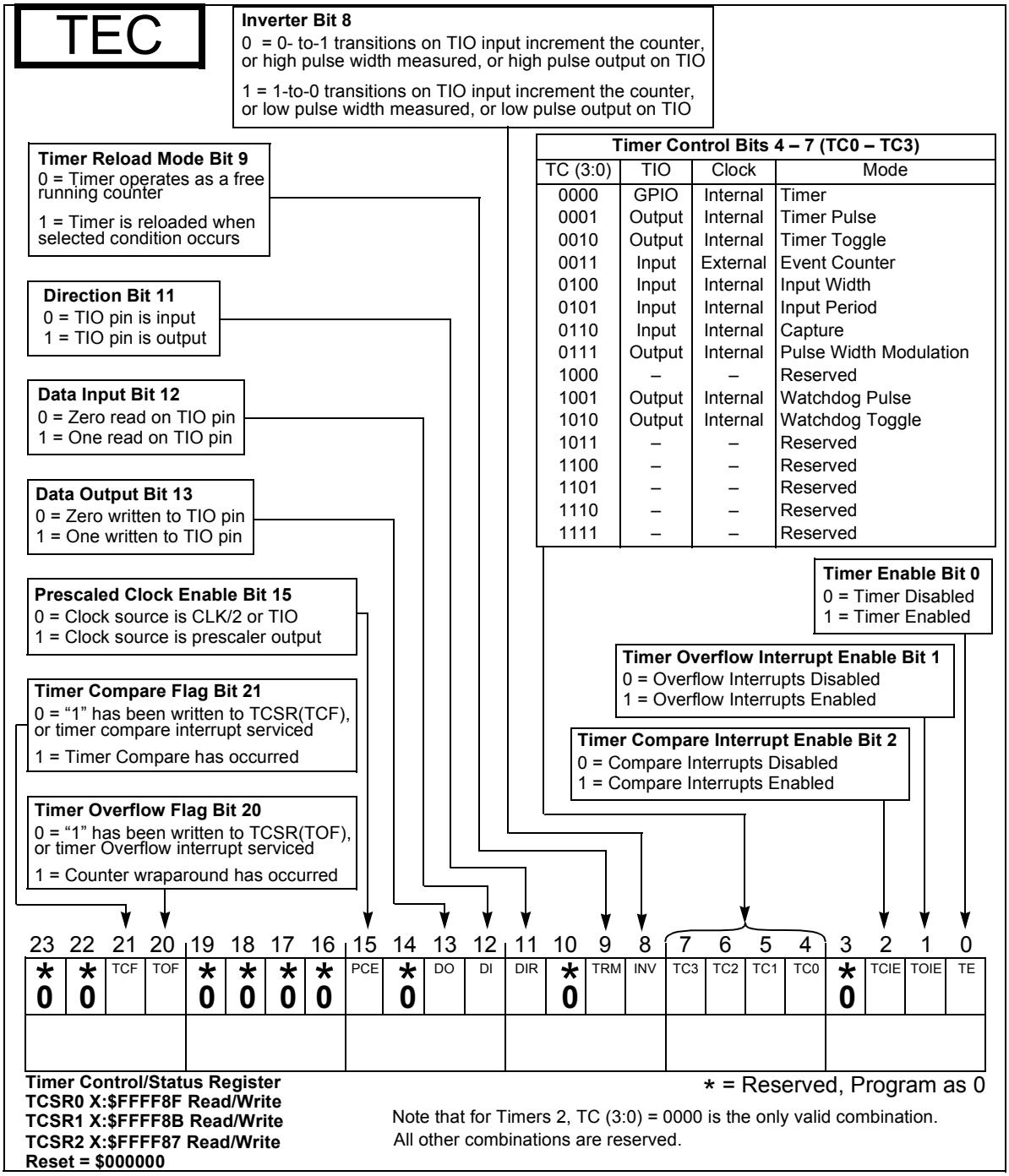


Figure C-20. Timer Control/Status Register

Application: _____ Date: _____

_____ Programmer: _____

TEC

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timer Reload Value																							

Timer Load Register
TLR0 X:$FFFF8E Write Only
TLR1 X:$FFFF8A Write Only
TLR2 X:$FFFF86 Write Only
Reset = $XXXXXX

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value Compared to Counter Value																							

Timer Compare Register
TCPR0 X:$FFFF8D Read/Write
TCPR1 X:$FFFF89 Read/Write
TCPR2 X:$FFFF85 Read/Write
Reset = $XXXXXX

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timer Count Value																							

Timer Count Register
TCR0 X:$FFFF8C Read Only
TCR1 X:$FFFF88 Read Only
TCR2 X:$FFFF84 Read Only
Reset = $000000

Figure C-21. Timer Load, Compare and Count Registers

Application: _____ Date: _____

_____ Programmer: _____

WDT

Watchdog Timer (WDT)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	WTC	* 0	DBGC	WEN

Watchdog Control Register (WCR) Y:$FFFFC0 * = Reserved, Program as 0
Write Once Reset = $00000F

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	WM15	WM14	WM13	WM12	WM11	WM10	WM9	WM8	WM7	WM6	WM5	WM4	WM3	WM2	WM1	WM0

Watchdog Modulus Register (WMR) Y:$FFFFC1 * = Reserved, Program as 0
Read/Write Reset = $00FFFF

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	WC15	WC14	WC13	WC12	WC11	WC10	WC9	WC8	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0

Watchdog Counter Register (WCNTR) Y:$FFFFC2 * = Reserved, Program as 0
Read Only Reset = $00FFFF

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	WS15	WS14	WS13	WS12	WS11	WS10	WS9	WS8	WS7	WS6	WS5	WS4	WS3	WS2	WS1	WS0

Watchdog Service Register (WSR) Y:$FFFFC3 * = Reserved, Program as 0
Write Only Reset = Undefined

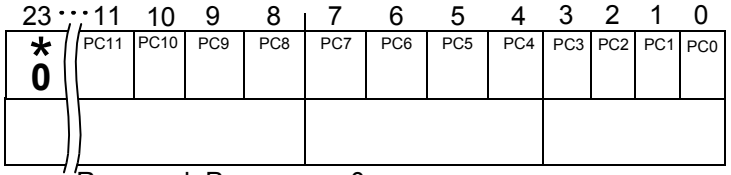
Application: _____ Date: _____

_____ Programmer: _____

GPIO

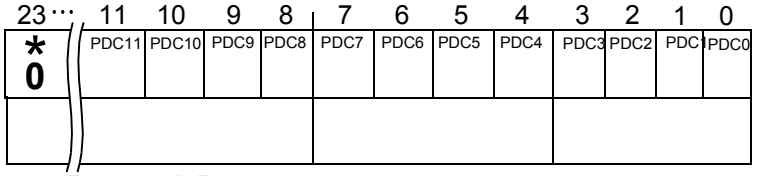
Port C (ESAI)

Port C Control Register (PCRC)
X:$FFFFBF
Read/Write
Reset = $0



* = Reserved, Program as 0

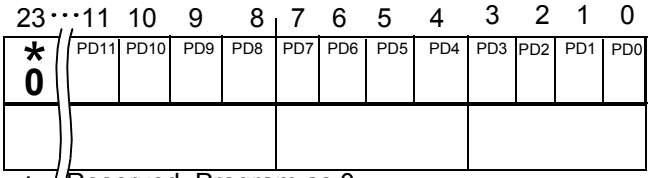
Port C Direction Register (PRRC)
X:$FFFFBE
Read/Write
Reset = $0



* = Reserved, Program as 0

PCn = 0 & PDCn = 0 -> Port pin PCn disconnected
PCn = 1 & PDCn = 0 -> Port pin PCn configured as input
PCn = 0 & PDCn = 1 -> Port pin PCn configured as output
PCn = 1 & PDCn = 1 -> Port pin configured as ESAI

Port C GPIO Data Register (PDRC)
X:$FFFFBD
Read/Write
Reset = undefined



* = Reserved, Program as 0

If port pin n is GPIO input, then PDn reflects the value on port pin n
if port pin n is GPIO output, then value written to PDn is reflected on port pin n

Figure C-22. GPIO Port C

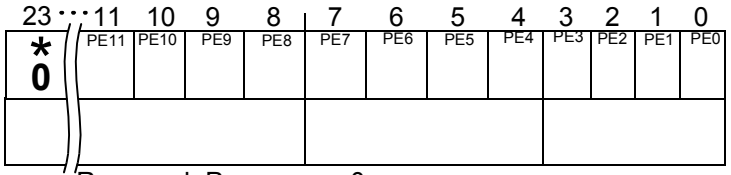
Application: _____ Date: _____

_____ Programmer: _____

GPIO

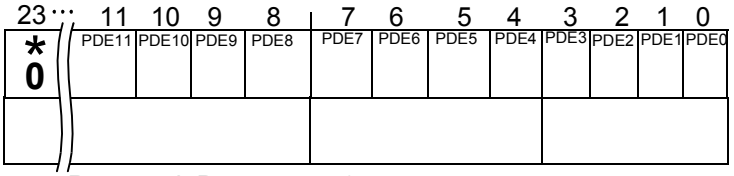
Port E (ESAI_1)

Port E Control Register (PCRE)
Y:$FFFF9F
Read/Write
Reset = $0



* = Reserved, Program as 0

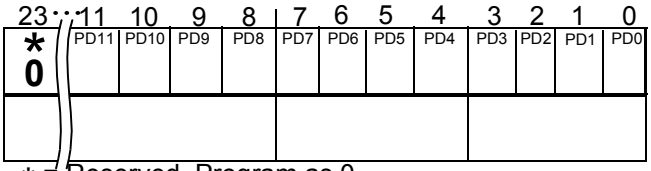
Port E Direction Register (PRRE)
Y:$FFFF9E
Read/Write
Reset = $0



* = Reserved, Program as 0

PEn = 0 & PDEn = 0 -> Port pin PEn disconnected
PEn = 1 & PDEn = 0 -> Port pin PEn configured as input
PEn = 0 & PDEn = 1 -> Port pin PEn configured as output
PEn = 1 & PDEn = 1 -> Port pin configured as ESAI_1

Port E GPIO Data Register (PDRE)
Y:$FFFF9D
Read/Write
Reset = undefined



* = Reserved, Program as 0

If port pin n is GPIO input, then PDn reflects the value on port pin n
if port pin n is GPIO output, then value written to PDn is reflected on port pin n

Figure C-23. GPIO Port E

Application: _____ Date: _____

_____ Programmer: _____

GPIO

Port G (GPIO)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	PG14	PG13	PG12	PG11	PG10	PG9	PG8	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0

Port G Control Register (PCRG) Y:$FFFFFA * = Reserved, Program as 0
Read/Write Reset = $0

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	PDG14	PDG13	PDG12	PDG11	PDG10	PDG9	PDG8	PDG7	PDG6	PDG5	PDG4	PDG3	PDG2	PDG1	PDG0

Port G Direction Register (PRRG) Y:$FFFFF9 * = Reserved, Program as 0
Read/Write Reset = $0

PGn = 0 & PDGn = 0 -> Port pin PGn disconnected
PGn = 1 & PDGn = 0 -> Port pin PGn configured as input
PGn = 0 & PDGn = 1 -> Port pin PGn configured as output
PGn = 1 & PDGn = 1 -> Port pin PGn configured as open drain output

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETI1	ETO1	ERI1	ERO1	ETI0	ETO0	ERI0	ERO0	* 0	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0

Port G GPIO Data Register (PDRG) Y:$FFFFF8 * = Reserved, Program as 0
Read/Write Reset = undefined

If port pin n is GPIO input, then PDn reflects the value on port pin n
if port pin n is GPIO output, then value written to PDn is reflected on port pin n

Figure C-24. GPIO Port G

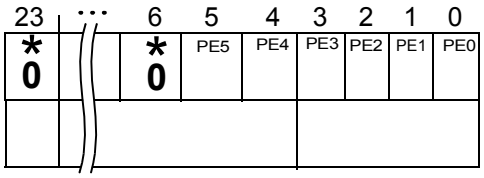
Application: _____ Date: _____

_____ Programmer: _____

GPIO

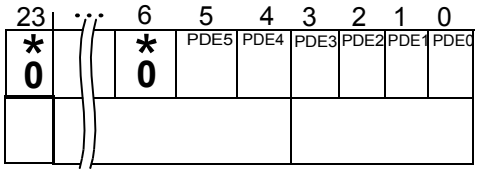
Port H

Port E Control Register
(PCRH)
X:$FFFF9A
Read/Write
Reset = $0



* = Reserved, Program as 0

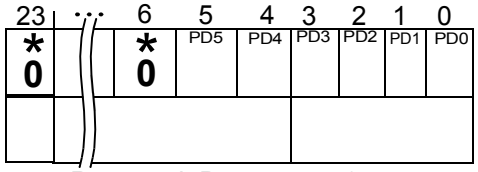
Port E Direction Register
(PRRH)
X:$FFFF99
Read/Write
Reset = $0



* = Reserved, Program as 0

PHn = 0 & PDHn = 0 -> Port pin PHn disconnected
PHn = 1 & PDHn = 0 -> Port pin PHn configured as input
PHn = 0 & PDHn = 1 -> Port pin PHn configured as output
PHn = 1 & PDHn = 1 -> Port pin configured as Respective Peripheral

Port H GPIO Data Register
(PDRH)
Y:$FFFF98
Read/Write
Reset = undefined



* = Reserved, Program as 0

If port pin n is GPIO input, then PDn reflects the value on port pin n
if port pin n is GPIO output, then value written to PDn is reflected on port pin n

Figure C-25. GPIO Port H

Appendix D
BSDL

D.1 52-pin BSDL

-- F R E E S C A L E (F O R M E R L Y M O T O R O L A) A D V T J T A G S O F T W A R E

-- BSDL File Generated: Mon Nov 24 14:48:49 2003

--

-- Revision History:

--

entity DSP56374 is

generic (PHYSICAL_PIN_MAP : string := "DSP56374_52PIN");

```
port (      tck:              in          bit;
            tms:              in          bit;
            tdi:              in          bit;
            tdo:              out         bit;
       moda_irqa:             inout       bit;
       modb_irqb:             inout       bit;
       modc_irqc:             inout       bit;
       modd_irqd:             inout       bit;
        sck_scl:              inout       bit;
       miso_sda:              inout       bit;
       mosi_ha0:              inout       bit;
         ss_ha2:              in          bit;
           hreq:              inout       bit;
       hckr_pc2:              inout       bit;
       hckt_pc5:              inout       bit;
        fsr_pc1:              inout       bit;
        fst_pc4:              inout       bit;
       sckr_pc0:              inout       bit;
       sckt_pc3:              inout       bit;
   sdo5_sdi0_pc6:             inout       bit;
   sdo4_sdi1_pc7:             inout       bit;
   sdo3_sdi2_pc8:             inout       bit;
   sdo2_sdi3_pc9:             inout       bit;
     sdo1_pc10:               inout       bit;
     sdo0_pc11:               inout       bit;
       tio0_pb0:              inout       bit;
       tio1_pb1:              inout       bit;
       tio2_pb2:              inout       bit;
           scan:              in          bit;
          extal:              in          bit;
      pinit_nmi:              in          bit;
        reset_b:              in          bit;
           xtal:              linkage     bit;
         io_vdd:              linkage     bit_vector(2 downto 0);
         io_gnd:              linkage     bit_vector(1 downto 0);
       core_vdd:              linkage     bit_vector(3 downto 0);
       core_gnd:              linkage     bit_vector(3 downto 0);
       plla_vdd:              linkage     bit;
       plla_gnd:              linkage     bit;
       pllp_vdd:              linkage     bit;
```

```
        pllp_gnd:           linkage         bit;
        plld_vdd:           linkage         bit;
        plld_gnd:           linkage         bit);

    use STD_1149_1_2001.all;

    attribute COMPONENT_CONFORMANCE of DSP56374 : entity is "STD_1149_1_2001";

    attribute PIN_MAP of DSP56374 : entity is PHYSICAL_PIN_MAP;

    constant DSP56374_52PIN : PIN_MAP_STRING :=
    "io_vdd:        (40, 14, 1), " &
    "moda_irqa:       2, " &
    "modb_irqb:       3, " &
    "modc_irqc:       4, " &
    "modd_irqd:       5, " &
    "core_vdd:      (47, 35, 22, 6), " &
    "core_gnd:      (46, 34, 23, 7), " &
    "hreq:           8, " &
    "ss_ha2:         9, " &
    "sck_scl:       10, " &
    "miso_sda:       11, " &
    "mosi_ha0:       12, " &
    "io_gnd:        (52, 13), " &
    "tdo:           15, " &
    "tdi:           16, " &
    "tms:           17, " &
    "tck:           18, " &
    "tio0_pb0:       19, " &
    "tio1_pb1:       20, " &
    "tio2_pb2:       21, " &
    "reset_b:       24, " &
    "plla_vdd:       25, " &
    "plla_gnd:       26, " &
    "pllp_vdd:       27, " &
    "pllp_gnd:       28, " &
    "plld_gnd:       29, " &
    "plld_vdd:       30, " &
    "extal:         31, " &
    "xtal:          32, " &
    "pinit_nmi:      33, " &
    "sdo0_pc11:       36, " &
    "sdo1_pc10:       37, " &
    "sdo2_sdi3_pc9:   38, " &
    "sdo3_sdi2_pc8:   39, " &
    "scan:          41, " &
    "hckt_pc5:       42, " &
    "hckr_pc2:       43, " &
    "sckt_pc3:       44, " &
    "sckr_pc0:       45, " &
    "fst_pc4:       48, " &
    "fsr_pc1:       49, " &
    "sdo4_sdi1_pc7:   50, " &
    "sdo5_sdi0_pc6:   51 ";

    attribute TAP_SCAN_IN    of  tdi : signal is true;
    attribute TAP_SCAN_OUT   of  tdo : signal is true;
```

```
attribute TAP_SCAN_MODE  of  tms : signal is true;
attribute TAP_SCAN_CLOCK of  tck : signal is (20.0e6, BOTH);

attribute COMPLIANCE_PATTERNS of DSP56374 : entity is
  "(scan) (0)";

attribute INSTRUCTION_LENGTH of DSP56374 : entity is 4;

attribute INSTRUCTION_OPCODE of DSP56374 : entity is
  "EXTEST                 (0000)," &
  "IDCODE                 (0010)," &
  "CLAMP                  (0101)," &
  "HIGHZ                  (0100)," &
  "SAMPLE                 (0001)," &
  "PRELOAD                (0001)," &
  "PLL_COMMAND            (0011)," &
  "ENABLE_ONCE            (0110)," &
  "DEBUG_REQUEST          (0111)," &
  "MBIST_COMMAND          (1000)," &
  "MBIST_ACCESS           (1010)," &
  "SHI_FILTER_TEST        (1101)," &
  "BYPASS                 (1111)";

attribute INSTRUCTION_CAPTURE of DSP56374 : entity is "0001";
attribute INSTRUCTION_PRIVATE of DSP56374 : entity is
  "PLL_COMMAND, MBIST_COMMAND, MBIST_ACCESS, SHI_FILTER_TEST ";

attribute IDCODE_REGISTER   of DSP56374 : entity is
  "00000001110000000011000000011101";

attribute REGISTER_ACCESS of DSP56374 : entity is
  "ONCE[8]       (ENABLE_ONCE," &
  "              DEBUG_REQUEST)";

attribute BOUNDARY_LENGTH of DSP56374 : entity is 104;

attribute BOUNDARY_REGISTER of DSP56374 : entity is
-- num   cell   port    func         safe [ccell  dis  rslt]
  "0    (BC_2, *,             internal,     1)," &
  "1    (BC_2, *,             internal,     1)," &
  "2    (BC_2, *,             internal,     1)," &
  "3    (BC_2, *,             internal,     1)," &
  "4    (BC_2, *,             internal,     1)," &
  "5    (BC_2, *,             internal,     1)," &
  "6    (BC_2, *,             internal,     1)," &
  "7    (BC_2, *,             internal,     1)," &
  "8    (BC_7, mosi_ha0,      bidir,        X,     9,   1,   Z)," &
  "9    (BC_2, *,             control,      1)," &
  "10   (BC_7, miso_sda,      bidir,        X,     11,   1,   Z)," &
  "11   (BC_2, *,             control,      1)," &
  "12   (BC_7, sck_scl,       bidir,        X,     13,   1,   Z)," &
  "13   (BC_2, *,             control,      1)," &
  "14   (BC_2, ss_ha2,        input,        X)," &
  "15   (BC_7, hreq,          bidir,        X,     16,   1,   Z)," &
  "16   (BC_2, *,             control,      1)," &
  "17   (BC_2, *,             internal,     1)," &
  "18   (BC_2, *,             internal,     1)," &
```

```
    "19   (BC_2, *,            internal,     1)," &
  -- num   cell  port   func        safe [ccell  dis  rslt]
    "20   (BC_2, *,            internal,     1)," &
    "21   (BC_2, *,            internal,     1)," &
    "22   (BC_2, *,            internal,     1)," &
    "23   (BC_7, modd_irqd,    bidir,        X,    24,   1,   Z)," &
    "24   (BC_2, *,            control,      1)," &
    "25   (BC_7, modc_irqc,    bidir,        X,    26,   1,   Z)," &
    "26   (BC_2, *,            control,      1)," &
    "27   (BC_2, *,            internal,     1)," &
    "28   (BC_2, *,            internal,     1)," &
    "29   (BC_2, *,            internal,     1)," &
    "30   (BC_2, *,            internal,     1)," &
    "31   (BC_7, modb_irqb,    bidir,        X,    32,   1,   Z)," &
    "32   (BC_2, *,            control,      1)," &
    "33   (BC_7, moda_irqa,    bidir,        X,    34,   1,   Z)," &
    "34   (BC_2, *,            control,      1)," &
    "35   (BC_7, sdo5_sdi0_pc6, bidir,       X,    36,   1,   Z)," &
    "36   (BC_2, *,            control,      1)," &
    "37   (BC_7, sdo4_sdi1_pc7, bidir,       X,    38,   1,   Z)," &
    "38   (BC_2, *,            control,      1)," &
    "39   (BC_2, *,            internal,     1)," &
  -- num   cell  port   func        safe [ccell  dis  rslt]
    "40   (BC_2, *,            internal,     1)," &
    "41   (BC_7, fsr_pc1,      bidir,        X,    42,   1,   Z)," &
    "42   (BC_2, *,            control,      1)," &
    "43   (BC_7, fst_pc4,      bidir,        X,    44,   1,   Z)," &
    "44   (BC_2, *,            control,      1)," &
    "45   (BC_2, *,            internal,     1)," &
    "46   (BC_2, *,            internal,     1)," &
    "47   (BC_2, *,            internal,     1)," &
    "48   (BC_2, *,            internal,     1)," &
    "49   (BC_2, *,            internal,     1)," &
    "50   (BC_2, *,            internal,     1)," &
    "51   (BC_7, sckr_pc0,     bidir,        X,    52,   1,   Z)," &
    "52   (BC_2, *,            control,      1)," &
    "53   (BC_7, sckt_pc3,     bidir,        X,    54,   1,   Z)," &
    "54   (BC_2, *,            control,      1)," &
    "55   (BC_2, *,            internal,     1)," &
    "56   (BC_2, *,            internal,     1)," &
    "57   (BC_2, *,            internal,     1)," &
    "58   (BC_2, *,            internal,     1)," &
    "59   (BC_7, hckr_pc2,     bidir,        X,    60,   1,   Z)," &
  -- num   cell  port   func        safe [ccell  dis  rslt]
    "60   (BC_2, *,            control,      1)," &
    "61   (BC_7, hckt_pc5,     bidir,        X,    62,   1,   Z)," &
    "62   (BC_2, *,            control,      1)," &
    "63   (BC_2, *,            internal,     1)," &
    "64   (BC_2, *,            internal,     1)," &
    "65   (BC_2, *,            internal,     1)," &
    "66   (BC_2, *,            internal,     1)," &
    "67   (BC_2, *,            internal,     1)," &
    "68   (BC_2, *,            internal,     1)," &
    "69   (BC_7, sdo3_sdi2_pc8, bidir,       X,    70,   1,   Z)," &
    "70   (BC_2, *,            control,      1)," &
    "71   (BC_7, sdo2_sdi3_pc9, bidir,       X,    72,   1,   Z)," &
    "72   (BC_2, *,            control,      1)," &
```

```
        "73   (BC_7, sdo1_pc10,     bidir,       X,    74,   1,   Z)," &
        "74   (BC_2, *,             control,     1)," &
        "75   (BC_7, sdo0_pc11,     bidir,       X,    76,   1,   Z)," &
        "76   (BC_2, *,             control,     1)," &
        "77   (BC_2, *,             internal,    1)," &
        "78   (BC_2, *,             internal,    1)," &
        "79   (BC_2, *,             internal,    1)," &
      -- num   cell   port   func        safe [ccell  dis  rslt]
        "80   (BC_2, *,             internal,    1)," &
        "81   (BC_2, *,             internal,    1)," &
        "82   (BC_2, *,             internal,    1)," &
        "83   (BC_2, *,             internal,    1)," &
        "84   (BC_2, *,             internal,    1)," &
        "85   (BC_2, pinit_nmi,     input,       X)," &
        "86   (BC_4, extal,         clock,       X)," &
        "87   (BC_2, *,             internal,    1)," &
        "88   (BC_2, *,             internal,    1)," &
        "89   (BC_2, *,             internal,    1)," &
        "90   (BC_2, *,             internal,    1)," &
        "91   (BC_2, *,             internal,    1)," &
        "92   (BC_2, *,             internal,    1)," &
        "93   (BC_2, reset_b,       input,       X)," &
        "94   (BC_2, *,             internal,    1)," &
        "95   (BC_2, *,             internal,    1)," &
        "96   (BC_7, tio2_pb2,      bidir,       X,    97,   1,   Z)," &
        "97   (BC_2, *,             control,     1)," &
        "98   (BC_7, tio1_pb1,      bidir,       X,    99,   1,   Z)," &
        "99   (BC_2, *,             control,     1)," &
      -- num   cell   port   func        safe [ccell  dis  rslt]
        "100  (BC_7, tio0_pb0,      bidir,       X,    101,  1,   Z)," &
        "101  (BC_2, *,             control,     1)," &
        "102  (BC_2, *,             internal,    1)," &
        "103  (BC_2, *,             internal,    1)";

end DSP56374;
```

D.2 80-pin BSDL

```
-- F R E E S C A L E  (F O R M E R L Y  M O T O R O L A)  A D V T  J T A G  S O F T W A R E
-- BSDL File Generated: Mon Nov 24 14:46:27 2003
--
-- Revision History:
--


entity DSP56374 is
        generic (PHYSICAL_PIN_MAP : string := "DSP56374_80PIN");

        port (      tck:            in          bit;
                    tms:            in          bit;
                    tdi:            in          bit;
                    tdo:            out         bit;
               moda_irqa:           inout       bit;
               modb_irqb:           inout       bit;
               modc_irqc:           inout       bit;
               modd_irqd:           inout       bit;
                gpio_pg0:           inout       bit;
                gpio_pg1:           inout       bit;
                gpio_pg2:           inout       bit;
                gpio_pg3:           inout       bit;
                gpio_pg4:           inout       bit;
                gpio_pg5:           inout       bit;
                gpio_pg6:           inout       bit;
                gpio_pg7:           inout       bit;
                gpio_pg8:           inout       bit;
                gpio_pg9:           inout       bit;
               gpio_pg10:           inout       bit;
               gpio_pg11:           inout       bit;
               gpio_pg12:           inout       bit;
               gpio_pg13:           inout       bit;
               gpio_pg14:           inout       bit;
                 sck_scl:           inout       bit;
                miso_sda:           inout       bit;
                mosi_ha0:           inout       bit;
                  ss_ha2:           in          bit;
                    hreq:           inout       bit;
                hckr_pc2:           inout       bit;
                hckt_pc5:           inout       bit;
                 fsr_pc1:           inout       bit;
                 fst_pc4:           inout       bit;
                sckr_pc0:           inout       bit;
                sckt_pc3:           inout       bit;
           sdo5_sdi0_pc6:           inout       bit;
           sdo4_sdi1_pc7:           inout       bit;
           sdo3_sdi2_pc8:           inout       bit;
           sdo2_sdi3_pc9:           inout       bit;
              sdo1_pc10:            inout       bit;
              sdo0_pc11:            inout       bit;
                hckr_pe2:           inout       bit;
                hckt_pe5:           inout       bit;
                 fsr_pe1:           inout       bit;
                 fst_pe4:           inout       bit;
```

```
        sckr_pe0:           inout           bit;
        sckt_pe3:           inout           bit;
    sdo5_sdi0_pe6:          inout           bit;
    sdo4_sdi1_pe7:          inout           bit;
    sdo3_sdi2_pe8:          inout           bit;
    sdo2_sdi3_pe9:          inout           bit;
      sdo1_pe10:            inout           bit;
      sdo0_pe11:            inout           bit;
       tio0_pb0:            inout           bit;
       tio1_pb1:            inout           bit;
       tio2_pb2:            inout           bit;
          scan:             in              bit;
         extal:             in              bit;
     pinit_nmi:             in              bit;
       reset_b:             in              bit;
          xtal:             linkage         bit;
        io_vdd:             linkage         bit_vector(3 downto 0);
        io_gnd:             linkage         bit_vector(1 downto 0);
      core_vdd:             linkage         bit_vector(3 downto 0);
      core_gnd:             linkage         bit_vector(3 downto 0);
       plla_vdd:            linkage         bit;
       plla_gnd:            linkage         bit;
       pllp_vdd:            linkage         bit;
       pllp_gnd:            linkage         bit;
       plld_vdd:            linkage         bit;
       plld_gnd:            linkage         bit);

use STD_1149_1_2001.all;

attribute COMPONENT_CONFORMANCE of DSP56374 : entity is "STD_1149_1_2001";

attribute PIN_MAP of DSP56374 : entity is PHYSICAL_PIN_MAP;

constant DSP56374_80PIN : PIN_MAP_STRING :=
"io_vdd:        (61, 47, 21, 1), " &
"moda_irqa:       2, " &
"modb_irqb:       3, " &
"gpio_pg13:       4, " &
"gpio_pg12:       5, " &
"modc_irqc:       6, " &
"modd_irqd:       7, " &
"gpio_pg11:       8, " &
"core_vdd:      (72, 52, 32, 9), " &
"core_gnd:      (71, 51, 33, 10), " &
"gpio_pg10:       11, " &
"gpio_pg9:        12, " &
"hreq:          13, " &
"ss_ha2:          14, " &
"sck_scl:         15, " &
"miso_sda:        16, " &
"mosi_ha0:        17, " &
"gpio_pg8:        18, " &
"gpio_pg7:        19, " &
"io_gnd:        (80, 20), " &
"gpio_pg6:        22, " &
"gpio_pg5:        23, " &
"tdo:           24, " &
```

```
"tdi:            25, " &
"tms:             26, " &
"tck:            27, " &
"gpio_pg4:       28, " &
"tio0_pb0:       29, " &
"tio1_pb1:       30, " &
"tio2_pb2:       31, " &
"gpio_pg3:       34, " &
"reset_b:        35, " &
"gpio_pg2:       36, " &
"gpio_pg1:       37, " &
"gpio_pg0:       38, " &
"plla_vdd:       39, " &
"plla_gnd:       40, " &
"pllp_vdd:       41, " &
"pllp_gnd:       42, " &
"plld_gnd:       43, " &
"plld_vdd:       44, " &
"extal:          45, " &
"xtal:           46, " &
"pinit_nmi:      48, " &
"sdo0_pe11:      49, " &
"sdo1_pe10:      50, " &
"sdo2_sdi3_pe9:  53, " &
"sdo3_sdi2_pe8:  54, " &
"sdo0_pc11:      55, " &
"sdo1_pc10:      56, " &
"sdo2_sdi3_pc9:  57, " &
"sdo3_sdi2_pc8:  58, " &
"sdo4_sdi1_pe7:  59, " &
"sdo5_sdi0_pe6:  60, " &
"scan:           62, " &
"hckt_pe5:       63, " &
"hckt_pc5:       64, " &
"hckr_pc2:       65, " &
"hckr_pe2:       66, " &
"sckt_pe3:       67, " &
"sckt_pc3:       68, " &
"sckr_pc0:       69, " &
"sckr_pe0:       70, " &
"gpio_pg14:      73, " &
"fst_pe4:        74, " &
"fst_pc4:        75, " &
"fsr_pc1:        76, " &
"fsr_pe1:        77, " &
"sdo4_sdi1_pc7:  78, " &
"sdo5_sdi0_pc6:  79 ";

attribute TAP_SCAN_IN    of  tdi : signal is true;
attribute TAP_SCAN_OUT   of  tdo : signal is true;
attribute TAP_SCAN_MODE  of  tms : signal is true;
attribute TAP_SCAN_CLOCK of  tck : signal is (20.0e6, BOTH);

attribute COMPLIANCE_PATTERNS of DSP56374 : entity is
  "(scan) (0)";

attribute INSTRUCTION_LENGTH of DSP56374 : entity is 4;
```

```
attribute INSTRUCTION_OPCODE of DSP56374 : entity is
  "EXTEST                 (0000)," &
  "IDCODE                 (0010)," &
  "CLAMP                  (0101)," &
  "HIGHZ                  (0100)," &
  "SAMPLE                 (0001)," &
  "PRELOAD                (0001)," &
  "PLL_COMMAND            (0011)," &
  "ENABLE_ONCE            (0110)," &
  "DEBUG_REQUEST          (0111)," &
  "MBIST_COMMAND          (1000)," &
  "MBIST_ACCESS           (1010)," &
  "SHI_FILTER_TEST        (1101)," &
  "BYPASS                 (1111)";

attribute INSTRUCTION_CAPTURE of DSP56374 : entity is "0001";
attribute INSTRUCTION_PRIVATE of DSP56374 : entity is
  "PLL_COMMAND, MBIST_COMMAND, MBIST_ACCESS, SHI_FILTER_TEST ";

attribute IDCODE_REGISTER   of DSP56374 : entity is
  "00000001110000000011000000011101";

attribute REGISTER_ACCESS of DSP56374 : entity is
  "ONCE[8]       (ENABLE_ONCE," &
  "              DEBUG_REQUEST)";

attribute BOUNDARY_LENGTH of DSP56374 : entity is 104;

attribute BOUNDARY_REGISTER of DSP56374 : entity is
-- num   cell   port    func        safe [ccell  dis  rslt]
  "0    (BC_7, gpio_pg5,      bidir,       X,     1,   1,   Z)," &
  "1    (BC_2, *,             control,     1)," &
  "2    (BC_7, gpio_pg6,      bidir,       X,     3,   1,   Z)," &
  "3    (BC_2, *,             control,     1)," &
  "4    (BC_7, gpio_pg7,      bidir,       X,     5,   1,   Z)," &
  "5    (BC_2, *,             control,     1)," &
  "6    (BC_7, gpio_pg8,      bidir,       X,     7,   1,   Z)," &
  "7    (BC_2, *,             control,     1)," &
  "8    (BC_7, mosi_ha0,      bidir,       X,     9,   1,   Z)," &
  "9    (BC_2, *,             control,     1)," &
  "10   (BC_7, miso_sda,      bidir,       X,     11,   1,   Z)," &
  "11   (BC_2, *,             control,     1)," &
  "12   (BC_7, sck_scl,       bidir,       X,     13,   1,   Z)," &
  "13   (BC_2, *,             control,     1)," &
  "14   (BC_2, ss_ha2,        input,       X)," &
  "15   (BC_7, hreq,          bidir,       X,     16,   1,   Z)," &
  "16   (BC_2, *,             control,     1)," &
  "17   (BC_7, gpio_pg9,      bidir,       X,     18,   1,   Z)," &
  "18   (BC_2, *,             control,     1)," &
  "19   (BC_7, gpio_pg10,     bidir,       X,     20,   1,   Z)," &
-- num   cell   port    func        safe [ccell  dis  rslt]
  "20   (BC_2, *,             control,     1)," &
  "21   (BC_7, gpio_pg11,     bidir,       X,     22,   1,   Z)," &
  "22   (BC_2, *,             control,     1)," &
  "23   (BC_7, modd_irqd,     bidir,       X,     24,   1,   Z)," &
  "24   (BC_2, *,             control,     1)," &
```

```
    "25   (BC_7, modc_irqc,     bidir,       X,    26,  1,  Z)," &
    "26   (BC_2, *,             control,     1)," &
    "27   (BC_7, gpio_pg12,     bidir,       X,    28,  1,  Z)," &
    "28   (BC_2, *,             control,     1)," &
    "29   (BC_7, gpio_pg13,     bidir,       X,    30,  1,  Z)," &
    "30   (BC_2, *,             control,     1)," &
    "31   (BC_7, modb_irqb,     bidir,       X,    32,  1,  Z)," &
    "32   (BC_2, *,             control,     1)," &
    "33   (BC_7, moda_irqa,     bidir,       X,    34,  1,  Z)," &
    "34   (BC_2, *,             control,     1)," &
    "35   (BC_7, sdo5_sdi0_pc6, bidir,       X,    36,  1,  Z)," &
    "36   (BC_2, *,             control,     1)," &
    "37   (BC_7, sdo4_sdi1_pc7, bidir,       X,    38,  1,  Z)," &
    "38   (BC_2, *,             control,     1)," &
    "39   (BC_7, fsr_pe1,       bidir,       X,    40,  1,  Z)," &
  -- num   cell  port   func       safe [ccell  dis  rslt]
    "40   (BC_2, *,             control,     1)," &
    "41   (BC_7, fsr_pc1,       bidir,       X,    42,  1,  Z)," &
    "42   (BC_2, *,             control,     1)," &
    "43   (BC_7, fst_pc4,       bidir,       X,    44,  1,  Z)," &
    "44   (BC_2, *,             control,     1)," &
    "45   (BC_7, fst_pe4,       bidir,       X,    46,  1,  Z)," &
    "46   (BC_2, *,             control,     1)," &
    "47   (BC_7, gpio_pg14,     bidir,       X,    48,  1,  Z)," &
    "48   (BC_2, *,             control,     1)," &
    "49   (BC_7, sckr_pe0,      bidir,       X,    50,  1,  Z)," &
    "50   (BC_2, *,             control,     1)," &
    "51   (BC_7, sckr_pc0,      bidir,       X,    52,  1,  Z)," &
    "52   (BC_2, *,             control,     1)," &
    "53   (BC_7, sckt_pc3,      bidir,       X,    54,  1,  Z)," &
    "54   (BC_2, *,             control,     1)," &
    "55   (BC_7, sckt_pe3,      bidir,       X,    56,  1,  Z)," &
    "56   (BC_2, *,             control,     1)," &
    "57   (BC_7, hckr_pe2,      bidir,       X,    58,  1,  Z)," &
    "58   (BC_2, *,             control,     1)," &
    "59   (BC_7, hckr_pc2,      bidir,       X,    60,  1,  Z)," &
  -- num   cell  port   func       safe [ccell  dis  rslt]
    "60   (BC_2, *,             control,     1)," &
    "61   (BC_7, hckt_pc5,      bidir,       X,    62,  1,  Z)," &
    "62   (BC_2, *,             control,     1)," &
    "63   (BC_7, hckt_pe5,      bidir,       X,    64,  1,  Z)," &
    "64   (BC_2, *,             control,     1)," &
    "65   (BC_7, sdo5_sdi0_pe6, bidir,       X,    66,  1,  Z)," &
    "66   (BC_2, *,             control,     1)," &
    "67   (BC_7, sdo4_sdi1_pe7, bidir,       X,    68,  1,  Z)," &
    "68   (BC_2, *,             control,     1)," &
    "69   (BC_7, sdo3_sdi2_pc8, bidir,       X,    70,  1,  Z)," &
    "70   (BC_2, *,             control,     1)," &
    "71   (BC_7, sdo2_sdi3_pc9, bidir,       X,    72,  1,  Z)," &
    "72   (BC_2, *,             control,     1)," &
    "73   (BC_7, sdo1_pc10,     bidir,       X,    74,  1,  Z)," &
    "74   (BC_2, *,             control,     1)," &
    "75   (BC_7, sdo0_pc11,     bidir,       X,    76,  1,  Z)," &
    "76   (BC_2, *,             control,     1)," &
    "77   (BC_7, sdo3_sdi2_pe8, bidir,       X,    78,  1,  Z)," &
    "78   (BC_2, *,             control,     1)," &
    "79   (BC_7, sdo2_sdi3_pe9, bidir,       X,    80,  1,  Z)," &
```

```
        -- num   cell  port   func        safe [ccell  dis  rslt]
          "80   (BC_2, *,           control,     1)," &
          "81   (BC_7, sdo1_pe10,    bidir,       X,    82,  1,  Z)," &
          "82   (BC_2, *,           control,     1)," &
          "83   (BC_7, sdo0_pe11,    bidir,       X,    84,  1,  Z)," &
          "84   (BC_2, *,           control,     1)," &
          "85   (BC_2, pinit_nmi,    input,       X)," &
          "86   (BC_4, extal,        clock,       X)," &
          "87   (BC_7, gpio_pg0,     bidir,       X,    88,  1,  Z)," &
          "88   (BC_2, *,           control,     1)," &
          "89   (BC_7, gpio_pg1,     bidir,       X,    90,  1,  Z)," &
          "90   (BC_2, *,           control,     1)," &
          "91   (BC_7, gpio_pg2,     bidir,       X,    92,  1,  Z)," &
          "92   (BC_2, *,           control,     1)," &
          "93   (BC_2, reset_b,      input,       X)," &
          "94   (BC_7, gpio_pg3,     bidir,       X,    95,  1,  Z)," &
          "95   (BC_2, *,           control,     1)," &
          "96   (BC_7, tio2_pb2,     bidir,       X,    97,  1,  Z)," &
          "97   (BC_2, *,           control,     1)," &
          "98   (BC_7, tio1_pb1,     bidir,       X,    99,  1,  Z)," &
          "99   (BC_2, *,           control,     1)," &
        -- num   cell  port   func        safe [ccell  dis  rslt]
          "100  (BC_7, tio0_pb0,     bidir,       X,    101,  1,  Z)," &
          "101  (BC_2, *,           control,     1)," &
          "102  (BC_7, gpio_pg4,     bidir,       X,    103,  1,  Z)," &
          "103  (BC_2, *,           control,     1)";

end DSP56374;
```

Notes

Index

Numerics

5 V tolerance 1

A

adder
- modulo 4
- offset 4
- reverse-carry 4

address bus 1
Address Generation Unit 4
addressing modes 4
AGU 4

B

barrel shifter 3
block diagram
- Clock Generator 6
- Phase Locked Loop (PLL) 2
- PLL clock generator 1

bus control 1
buses
- internal 4

C

Central Processing Unit (CPU) i
charge pump loop filter 2
CLKGEN 5
Clock 4
clock 1
Clock divider 9
Clock Generator (CLKGEN) 5, 1, 6
clock input frequency division 2, 3
CPHA and CPOL (HCKR Clock Phase and Polarity Controls) 5

D

data ALU 3
- registers 3

data bus 1
Data Input (DI) bit 22
Data Output (DO) bit 21
DAX 1, 16
Digital Audio Transmitter 1, 16
Direct Memory Access (DMA)
- triggered by timer 18

Direction (DIR) bit 22
Divide Factor (DF) 5
DMA 5
DO loop 4
DSP56300 core 2
DSP56300 Family Manual i, 2
DSP56303 Technical Data i

E

Enhanced Serial Audio Interface 8, 12
Enhanced Synchronous Audio Interface 1
ESAI 1, 8, 12
ESAI block diagram 1
ESSI0 (GPIO) 1
ESSI1 (GPIO) 1, 3
EXTAL 2

F

Frequency Divider 3
frequency multiplication 3
frequency predivider 2
functional signal groups 1

G

Global Data Bus 5
GPIO 6
GPIO (ESSI0, Port C) 1
GPIO (ESSI1, Port D) 1, 3
GPIO (Timer) 4
Ground 3
ground 1

H

HA1, HA3-HA6 (HSAR I^2C Slave Address) 5
hardware stack 4
HBER (HCSR Bus Error) 11
HBIE (HCSR Bus Error Interrupt Enable) 9
HBUSY (HCSR Host Busy) 11
HCKR (SHI Clock Control Register) 5
HCSR
- Receive Interrupt Enable Bits 10
- SHI Control/Status Register 7

HDI08 1
HDM0-HDM5 (HCKR Divider Modulus Select) 7
HEN (HCSR SHI Enable) 7
HFIFO (HCSR FIFO Enable Control) 8
HFM0-HFM1 (HCKR Filter Mode) 7
HI^{2}C (HCSR Serial Host Interface I^2C/SPI Selection) 8
HIDLE (HCSR Idle) 9
HM0-HM1 (HCSR Serial Host Interface Mode) 8
HMST (HCSR Master Mode) 8
Host
- Receive Data FIFO (HRX) 5
- Receive Data FIFO—DSP Side 5
- Transmit Data Register (HTX) 4
- Transmit Data Register—DSP Side 4

Host Interface 1
HREQ Function In SHI Slave Modes 9
HRFF (HCSR Host Receive FIFO Full) 10
HRIE0-HRIE1 (HCSR Receive Interrupt Enable) 10
HRNE (HCSR Host Receive FIFO Not Empty) 10
HROE (HCSR Host Receive Overrun Error) 11
HRQE0-HRQE1 (HCSR Host Request Enable) 9
HTDE (HCSR Host Transmit Data Empty) 10
HTIE (HCSR Transmit Interrupt Enable) 9
HTUE (HCSR Host Transmit Underrun Error) 10

I

I^2C 7, 1, 11
- Bit Transfer 12
- Bus Protocol For Host Read Cycle 13
- Bus Protocol For Host Write Cycle 13
- Data Transfer Formats 13
- Master Mode 15
- Protocol for Host Write Cycle 13
- Receive Data In Master Mode 16

Receive Data In Slave Mode 15
Slave Mode 14
Start and Stop Events 12
Transmit Data In Master Mode 16
Transmit Data In Slave Mode 15
I2C Bus Acknowledgment 12
I2C Mode 1
initializing the timer 2
Inter Integrated Circuit Bus 7, 1
internal buses 4
Internal Exception Priorities
SHI 4
interrupt 4
interrupt and mode control 1, 4, 5
interrupt control 4, 5
Interrupt Service Routine (ISR) 3
Interrupt Vectors
SHI 4
Inverter (INV) bit 22, 24

J

JTAG 19
JTAG/OnCE port 1

L

LA register 4
LC register 4
Locked state, PLL 2
Loop Address register (LA) 4
Loop Counter register (LC) 4
Low-Power Divider (LPD) 6

M

MAC 3
Manual Conventions iii
memory
on-chip 5
MF (Multiplication Factor) 3, 10
mode control 4, 5
modulo adder 4
Multiplication Factor 3, 10
multiplier-accumulator (MAC) 3

O

offset adder 4
OMR register 4
OnCE module 19
on-chip memory 5
operating mode 2
Operating Mode Register (OMR) 4

P

PAB 5
PAG 4
PC register 4
PCU 4
PDB 5
PDC 4
Peripheral I/O Expansion Bus 4
Phase Detector (PD) 2
Phase Locked Loop (PLL). See PLL
PIC 4
PINIT 1
PLL 5, 1, 4
clock generator 1
Control (PCTL) register 7
Bit Definitions 8
Division Factor (DF) bit 9
Multiplication Factor (MF) bits 10
PLL Enable (PEN) bit 9
PLL Stop State (PSTP) bit 9
Predivider Factor (PD) bit 8
Control Elements in its circuitry
clock input division 3
frequency multiplication 3
control mechanisms 1
charge pump loop filter 2
frequency predivider 2
phase detector 2
Division Factor 3
PCTL Multiplication Factor 3
PCTL Predivider Factor (PDF) bits 3
Port A 1
Port B 1
Port C 1, 8, 12, 1
Port D 16, 1, 3
power 1
Prescaler Clock Enable (PCE) bit 21
prescaler counter 18
Prescaler Counter Value (PC) bits 20
Prescaler Preload Value (PL) bits 20
Prescaler Source (PS) bits 20
Program Address Bus (PAB) 5
Program Address Generator (PAG) 4
Program Control Unit (PCU) 4
Program Counter register (PC) 4
Program Data Bus (PDB) 5
Program Decode Controller (PDC) 4
Program Interrupt Controller (PIC) 4
Program Memory Expansion Bus 4
Programming Model
SHI—DSP Side 3
SHI—Host Side 3
programming model
timer 18

R

$\overline{\text{RESET}}$ 5
reverse-carry adder 4

S

SC register 4
Serial Host Interface 1, 6
Serial Host Interface (SHI) 7, 1
Serial Host Interface—See Section 5
Serial Peripheral Interface Bus 7, 1
setting timer operating mode 2
SHI 7, 1, 6, 1
Block Diagram 2
Clock Control Register—DSP Side 5
Clock Generator 2
Control/Status Register—DSP Side 7
Data Size 8
Exception Priorities 4
HCKR

Clock Phase and Polarity Controls 5
Divider Modulus Select 7
Prescaler Rate Select 6
HCKR Filter Mode 7
HCSR
Bus Error Interrupt Enable 9
FIFO Enable Control 8
Host Request Enable 9
Idle 9
Master Mode 8
Serial Host Interface I^2C/SPI Selection 8
Serial Host Interface Mode 8
SHI Enable 7
Host Receive Data FIFO—DSP Side 5
Host Transmit Data Register—DSP Side 4
HREQ
Function In SHI Slave Modes 9
HSAR
I^2C Slave Address 5
Slave Address Register 5
I/O Shift Register 4
Input/Output Shift Register—Host Side 4
Internal Architecture 1
Internal Interrupt Priorities 4
Interrupt Vectors 4
Introduction 1
Operation During Stop 17
Programming Considerations 13
Programming Model 2
Programming Model—DSP Side 3
Programming Model—Host Side 3
Slave Address Register—DSP Side 5
SHI Noise Reduction Filter Mode 7
signal groupings 1
signals 1
Size register (SZ) 4
SP 4
SPI 7, 1
HCSR
Bus Error 11
Host Busy 11
Host Receive FIFO Full 10
Host Receive FIFO Not Empty 10
Host Receive Overrun Error 11
Host Transmit Data Empty 10
Host Transmit Underrun Error 10
Receive Interrupt Enable 9, 10
Master Mode 14
Slave Mode 13
SPI Data-To-Clock Timing 6
SPI Data-To-Clock Timing Diagram 6
SPI Mode 1
SR register 4
SS 4
Stack Counter register (SC) 4
Stack Pointer (SP) 4
Status Register (SR) 4
System Stack (SS) 4
SZ register 4

T

Timer 1
timer
after Reset 2
enabling 2
exception 3
Compare 3
Overflow 3
initialization 2
module
timer block diagram 1
operating modes 3
Capture (mode 6) 3, 10, 13, 14
Event Counter (mode 3) 3, 9
GPIO (mode 0) 3, 4
Input Period (mode 5) 3, 10, 12
Input Width (mode 4) 3, 10
overview 3
Pulse (mode 1) 3, 5
Pulse Width Modulation (PWM) (mode 7) 3, 10, 14
reserved 18
setting 2
signal measurement modes 10
Toggle (mode 2) 3, 7
watchdog modes 16
Watchdog Pulse (mode 9) 3, 16
Watchdog Toggle (mode 10) 3, 16
prescaler counter 18
programming model 18
special cases 18
timer compare interrupts 24
Timer Compare Register (TCPR) 25
Timer Control/Status Register (TCSR) 21
Data Input (DI) 22
Data Output (DO) 21
Direction (DIR) 22
Inverter (INV) 22, 24
Prescaler Clock Enable (PCE) 21
Timer Compare Flag (TCF) 21
Timer Compare Interrupt Enable (TCIE) 24
Timer Control (TC) 23
Timer Enable (TE) 24
Timer Overflow Flag (TOF) 21
Timer Overflow Interrupt Enable (TOIE) 24
Timer Reload Mode (TRM) 22
Timer Count Register (TCR) 25
Timer Load Registers (TLR) 25
Timer Prescaler Count Register (TPCR) 20
Prescaler Counter Value (PC) 20
Timer Prescaler Load Register (TPLR) 19
bit definitions 20
Prescaler Preload Value (PL) 20
Prescaler Source (PS) 20
Timer (GPIO) 4
Timer Compare Flag (TCF) bit 21
Timer Compare Interrupt Enable (TCIE) bit 24
Timer Compare Register (TCPR) 2, 3, 25
Timer Control (TC) bits 23
Timer Control/Status Register (TCSR) 2, 21
bit definitions 21
Data Input (DI) 22
Data Output (DO) 21
Direction (DIR) 22
Inverter (INV) 22, 24

Prescaler Clock Enable (PCE) 21
Timer Compare Flag (TCF) 21
Timer Compare Interrupt Enable (TCIE) 24
Timer Control (TC) 23
Timer Enable (TE) 24
Timer Overflow Flag (TOF) 21
Timer Overflow Interrupt Enable (TOIE) 24
Timer Reload Mode (TRM) 22
Timer Count Register (TCR) 25
Timer Enable (TE) bit 24
Timer Load Registers (TLR) 2, 25
Timer module
architecture 1
Timer Overflow Flag (TOF) bit 21
Timer Overflow Interrupt Enable (TOIE) bit 24
Timer Prescaler Count Register (TPCR) 20
bit definitions 20
Prescaler Counter Value (PC) 20
Timer Prescaler Load Register (TPLR) 2, 19
bit definitions 20
Prescaler Preload Value (PL) 20
Prescaler Source (PS) 20
Timer Reload Mode (TRM) bit 22
Transmitter High Frequency Clock Divider 9

V

VBA register 4
Vector Base Address register (VBA) 4
Voltage Controlled Oscillator. See VCO

X

X Memory Address Bus (XAB) 5
X Memory Data Bus (XDB) 5
X Memory Expansion Bus 5
XAB 5
XDB 5

Y

Y Memory Address Bus (YAB) 5
Y Memory Data Bus (YDB) 5
Y Memory Expansion Bus 5
YAB 5
YDB 5