

# **DSP56166**

## **16-BIT DIGITAL SIGNAL PROCESSOR USER'S MANUAL**



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



**MOTOROLA**

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
<b>SECTION 1</b>		
<b>DSP56166 OVERVIEW</b>		
1.1	INTRODUCTION .....	1-3
1.2	DSP5616 CORE BLOCK DIAGRAM DESCRIPTION .....	1-4
1.2.1.	Data Buses .....	1-4
1.2.2.	Address Buses .....	1-5
1.2.3.	Data ALU .....	1-6
1.2.4.	Address Generation Unit (AGU) .....	1-9
1.2.5.	Program Control Unit (PCU) .....	1-10
1.2.5.1.	Interrupt Priority Structure .....	1-10
1.2.5.2.	Interrupt Priority Levels (IPL) .....	1-13
1.2.5.3.	Exception Priorities within an IPL .....	1-13
1.3	MEMORY ORGANIZATION .....	1-13
1.4	EXTERNAL BUS, I/Os and ON-CHIP PERIPHERALS .....	1-14
1.4.1.	Memory Expansion Port (Port A) .....	1-15
1.4.2.	General Purpose I/O (Port B, Port C) .....	1-15
1.4.3.	RSSI0 and RSSI1 .....	1-16
1.4.4.	Timer .....	1-16
1.4.5.	Host Interface (HI) .....	1-16
1.5	OnCE .....	1-17
1.6	PROGRAMMING MODEL .....	1-17
1.6.1.	Data ALU .....	1-17
1.6.1.1.	Data ALU Input Registers (X1, X0, Y1, Y0) .....	1-18
1.6.1.2.	Data ALU Accumulator Registers (A2, A1, A0, B2, B1, B0) .....	1-18
1.6.2.	Address Generation Unit .....	1-20
1.6.2.1.	Address Register File (R0-R3) .....	1-20
1.6.2.2.	Offset Register File (N0-N3) .....	1-20
1.6.2.3.	Modifier Register File (M0-M3) .....	1-20
1.6.3.	Program Control Unit .....	1-20
1.6.3.1.	Program Counter (PC) .....	1-20
1.6.3.2.	Status Register (SR) .....	1-21
1.6.3.3.	Loop Counter (LC) .....	1-22
1.6.3.4.	Loop Address Register (LA) .....	1-22
1.6.3.5.	System Stack (SS) .....	1-22
1.6.3.6.	Stack Pointer (SP) .....	1-23

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
1.6.3.7.	Operating Mode Register (OMR) .....	1-23
1.7	INSTRUCTION SET SUMMARY .....	1-26
1.7.1.	Instruction Groups .....	1-26
1.7.1.1.	Arithmetic Instructions .....	1-27
1.7.1.2.	Logical Instructions .....	1-28
1.7.1.3.	Bit Field Manipulation Instructions .....	1-28
1.7.1.4.	Loop Instructions .....	1-29
1.7.1.5.	Move Instructions .....	1-29
1.7.1.6.	Program Control Instructions .....	1-30
1.7.2.	Instruction Formats .....	1-30
1.7.3.	Addressing Modes .....	1-31
1.7.4.	Address Arithmetic .....	1-33
1.7.4.1.	Linear Modifier .....	1-33
1.7.4.2.	Reverse Carry Modifier .....	1-33
1.7.4.3.	Modulo Modifier .....	1-34

## SECTION 2 DSP56166 PIN DESCRIPTIONS

2.1	INTRODUCTION .....	2-3
2.2	ADDRESS AND DATA BUS (32 PINS) .....	2-3
2.3	BUS CONTROL (10 PINS) .....	2-3
2.4	INTERRUPT AND MODE CONTROL (4 PINS) .....	2-9
2.5	POWER, GROUND, AND CLOCK (30 PINS) .....	2-10
2.6	HOST INTERFACE (15 PINS) .....	2-11
2.7	16-BIT TIMER (2 PINS) .....	2-12
2.8	REDUCED SYNCHRONOUS SERIAL INTERFACES (RSSI0 AND RSSI1) AND PORT C (8 PINS) .....	2-13
2.9	ON-CHIP EMULATION (4 PINS) .....	2-14
2.10	ON-CHIP CODEC (7 PINS) .....	2-15

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
---------------------	-------	----------------

---

### SECTION 3 OPERATING MODES AND MEMORY CONFIGURATION

3.1	INTRODUCTION .....	3-3
3.2	DSP56166 RAM BASED DESCRIPTION .....	3-3
3.2.1.	X Data Memory .....	3-3
3.2.2.	Program Memory .....	3-5
3.2.3.	Bootstrap ROM .....	3-5
3.2.4.	RAM Based DSP56166 Operating Modes .....	3-5
3.2.4.1.	Bootstrap Mode (Mode 0). .....	3-6
3.2.4.2.	Bootstrap Mode (Mode 1). .....	3-6
3.2.4.3.	Normal Expanded Mode (Mode 2). .....	3-7
3.2.4.4.	Development Mode (Mode 3). .....	3-7
3.2.5.	Bootstrap Mode .....	3-7
3.2.5.1.	Bootstrap ROM .....	3-7
3.2.5.2.	Bootstrap Control Logic .....	3-7
3.2.5.3.	Bootstrap Program .....	3-8
3.3	DSP56166 ROM BASED DESCRIPTION .....	3-10
3.3.1.	X Data Memory .....	3-10
3.3.2.	Program Memory .....	3-11
3.3.3.	ROM Based DSP56166 Operating Modes .....	3-12
3.3.3.1.	Single-chip Mode (Mode 0). .....	3-12
3.3.3.2.	Single-chip Mode (Mode 1). .....	3-12
3.3.3.3.	Normal Expanded Mode (Mode 2). .....	3-13
3.3.3.4.	Development Mode (Mode 3). .....	3-13

### SECTION 4 DSP56166 I/O INTERFACE

4.1	INTRODUCTION .....	4-3
4.2	I/O PORT SET-UP AND PROGRAMMING .....	4-3
4.2.1.	Port Registers .....	4-6
4.2.1.1.	Bus Control Registers (BCR and BCR2) .....	4-6
4.2.1.2.	Port B and Port C Registers .....	4-7

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
<b>SECTION 5</b>		
<b>HOST INTERFACE</b>		
5.1	INTRODUCTION .....	5-3
5.2	HOST INTERFACE PROGRAMMING MODEL .....	5-5
5.3	HOST TRANSMIT DATA REGISTER (HTX) .....	5-5
5.4	RECEIVE BYTE REGISTERS (RXH, RXL) .....	5-5
5.5	TRANSMIT BYTE REGISTERS (TXH, TXL) .....	5-5
5.6	HOST RECEIVE DATA REGISTER (HRX) .....	5-6
5.7	COMMAND VECTOR REGISTER (CVR) .....	5-7
5.7.1.	CVR Host Vector (HV) Bits 0 through 4 .....	5-7
5.7.2.	CVR Reserved Bits – Bits 5 and 6 .....	5-7
5.7.3.	CVR Host Command Bit (HC) Bit 7 .....	5-9
5.8	HOST CONTROL REGISTER (HCR) .....	5-9
5.8.1.	HCR Host Receive Interrupt Enable (HRIE) Bit 0 .....	5-10
5.8.2.	HCR Host Transmit Interrupt Enable (HTIE) Bit 1 .....	5-10
5.8.3.	HCR Host Command Interrupt Enable (HCIE) Bit 2 .....	5-10
5.8.4.	HCR Host Flag 2 (HF2) Bit 3 .....	5-10
5.8.5.	HCR Host Flag 3 (HF3) Bit 4 .....	5-10
5.8.6.	HCR Reserved Control – Bits 5, 6, and 7 .....	5-11
5.9	HOST STATUS REGISTER (HSR) .....	5-11
5.9.1.	HSR Host Receive Data Full (HRDF) Bit 0 .....	5-11
5.9.2.	HSR Host Transmit Data Empty (HTDE) Bit 1 .....	5-11
5.9.3.	HSR Host Command Pending (HCP) Bit 2 .....	5-11
5.9.4.	HSR Host Flag 0 (HF0) Bit 3 .....	5-12
5.9.5.	HSR Host Flag 1 (HF1) Bit 4 .....	5-12
5.9.6.	HSR Reserved Status – Bits 5 and 6 .....	5-12
5.9.7.	HSR DMA Status (DMA) Bit 7 .....	5-12
5.10	INTERRUPT CONTROL REGISTER (ICR) .....	5-12
5.10.1.	ICR Receive Request Enable (RREQ) Bit 0 .....	5-12
5.10.2.	ICR Transmit Request Enable (TREQ) Bit 1 .....	5-13
5.10.3.	ICR Reserved bit – Bit 2 .....	5-13
5.10.4.	ICR Host Flag 0 (HF0) Bit 3 .....	5-13
5.10.5.	ICR Host Flag 1 (HF1) Bit 4 .....	5-14
5.10.6.	ICR Host Mode Control (HM1, HM0) Bits 5 and 6 .....	5-14
5.10.7.	ICR Initialize Bit (INIT) Bit 7 .....	5-15
5.11	INTERRUPT STATUS REGISTER (ISR) .....	5-15

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
5.11.1.	ISR Receive Data Register Full (RXDF) Bit 0 . . . . .	5-16
5.11.2.	ISR Transmit Data Register Empty (TXDE) Bit 1 . . . . .	5-16
5.11.3.	ISR Transmitter Ready (TRDY) Bit 2 . . . . .	5-16
5.11.4.	ISR Host Flag 2 (HF2) Bit 3 . . . . .	5-17
5.11.5.	ISR Host Flag 3 (HF3) Bit 4 . . . . .	5-17
5.11.6.	ISR (Reserved Status) Bit 5 . . . . .	5-17
5.11.7.	ISR DMA Status (DMA) Bit 6 . . . . .	5-17
5.11.8.	ISR Host Request (HREQ) Bit 7 . . . . .	5-17
5.12	INTERRUPT VECTOR REGISTER (IVR) . . . . .	5-17
5.13	IVR HOST INTERFACE INTERRUPTS . . . . .	5-18
5.14	DMA MODE OPERATION . . . . .	5-18
5.14.1.	Host to DSP – Host Interface Action . . . . .	5-19
5.14.2.	Host to DSP – Host Processor Procedure . . . . .	5-19
5.14.3.	DSP to Host Interface Action . . . . .	5-20
5.14.4.	DSP to Host – Host Processor Procedure . . . . .	5-21
5.15	HOST PORT USAGE – GENERAL CONSIDERATIONS . . . . .	5-21
5.15.1.	Host Programmer Considerations . . . . .	5-21
5.15.2.	DSP Programmer Considerations . . . . .	5-23

## SECTION 6 DSP56166 ON-CHIP SIGMA/DELTA CODEC

6.1	GENERAL DESCRIPTION . . . . .	6-3
6.2	CODEC BLOCK DIAGRAM . . . . .	6-4
6.3	ANALOG I/O DEFINITION . . . . .	6-5
6.4	INTERFACE WITH THE DSP5616 CORE . . . . .	6-6
6.4.1.	Interface Definition . . . . .	6-6
6.4.2.	On-chip Codec Programming Model . . . . .	6-7
6.4.3.	Codec Receive Register CRX . . . . .	6-8
6.4.4.	Codec Transmit Register CTX . . . . .	6-8
6.4.5.	Codec Control Register CCR0 . . . . .	6-8
6.4.5.1.	CCR0 Input Divider Bits (ED5-ED0) Bits 0-5 . . . . .	6-8
6.4.5.2.	CCR0 Codec Ratio Select Bits (CRS5-0) Bits 13-8 . . . . .	6-8
6.4.5.3.	CCR0 Reserved Bits 6-7 and 14-15 . . . . .	6-11
6.4.6.	Codec Control Register CCR1 . . . . .	6-11

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
6.4.6.1.	CCR1 Audio Level Control Bits (VC3-VC0) Bits 0-3 . . . . .	6-11
6.4.6.2.	CCR1 Codec Loop Back Bit (CLB) Bit 7 . . . . .	6-12
6.4.6.3.	CCR1 Clock Select Bit (CLS) Bit 8 . . . . .	6-12
6.4.6.4.	CCR1 Mute Bit (MUT) Bit 10 . . . . .	6-12
6.4.6.5.	CCR1 Microphone Gain Select Bits (MGS1-0) Bits 11 and 12 . .	6-12
6.4.6.6.	CCR1 Input Select Bit (INS) Bit 13 . . . . .	6-13
6.4.6.7.	CCR1 Codec Enable Bit (COE) Bit 14 . . . . .	6-13
6.4.6.8.	CCR1 Codec Interrupt Enable Bit (COIE) Bit 15 . . . . .	6-13
6.4.6.9.	CCR1 Reserved Bits 4,5,6, and 9 . . . . .	6-13
6.4.7.	Codec Status Register COSR . . . . .	6-13
6.4.7.1.	COSR Codec Transmit Underrun Error FLag Bit (CTUE) Bit 0 .	6-13
6.4.7.2.	COSR Codec Receive Overrun Error Flag Bit (CROE) Bit 1 . . .	6-14
6.4.7.3.	COSR Codec Transmit Data Empty Bit (CTDE) Bit 2 . . . . .	6-14
6.4.7.4.	COSR Codec Receive Data Full Bit (CRDF) Bit 3 . . . . .	6-14
6.4.7.5.	COSR Reserved Bits 4-15 . . . . .	6-14
6.5	ON-CHIP CODEC GAIN AND FREQUENCY RESPONSE ANALYSIS .	6-16
6.5.1.	DC Gain and Frequency Response of the A/D Section . . . . .	6-16
6.5.2.	DC Gain and Frequency Response of the D/A Section . . . . .	6-21
6.5.2.1.	D/A Second Order Digital Interpolation Comb Filter . . . . .	6-21
6.5.2.2.	D/A Digital Modulator . . . . .	6-24
6.5.2.3.	D/A Butterworth Analog Low Pass Filter . . . . .	6-24
6.5.2.4.	Overall Frequency Response of the D/A Section . . . . .	6-25

## SECTION 7 16-BIT TIMER AND EVENT COUNTER

7.1	INTRODUCTION . . . . .	7-3
7.2	TIMER ARCHITECTURE . . . . .	7-3
7.3	TIMER COUNT REGISTER (TCTR) . . . . .	7-3
7.4	TIMER PRELOAD REGISTER (TPR) . . . . .	7-4
7.5	TIMER COMPARE REGISTER (TCPR) . . . . .	7-5
7.6	TIMER CONTROL REGISTER (TCR) . . . . .	7-6
7.6.1.	TCR Decrement Ratio (DC7-DC0) Bits 0-7 . . . . .	7-6
7.6.2.	TCR Event Select (ES) Bit 8 . . . . .	7-6
7.6.3.	TCR Overflow Interrupt Enable (OIE) Bit 9 . . . . .	7-6
7.6.4.	TCR Compare Interrupt Enable (CIE) Bit 10 . . . . .	7-7

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
7.6.5.	TCR Timer Output Enable (TO2-TO0) Bits 11-13 .....	7-7
7.6.6.	TCR Inverter Bit (INV) Bit 14 .....	7-7
7.6.7.	TCR Timer Enable (TE) Bit 15 .....	7-8
7.7	TIMER RESOLUTION .....	7-8
7.8	EVENT COUNTER TIMER DIAGRAMS .....	7-8

## SECTION 8 REDUCED SSI (RSSI0 and RSSI1)

8.1	INTRODUCTION .....	8-3
8.2	RSSI OPERATING MODES .....	8-3
8.3	RSSI CLOCK AND FRAME SYNC GENERATION .....	8-3
8.4	RSSI DATA AND CONTROL PINS .....	8-4
8.4.1.	Serial Transmit Data Pin — STD .....	8-6
8.4.2.	Serial Receive Data Pin — SRD .....	8-6
8.4.3.	Serial Clock — SCK .....	8-6
8.4.4.	Serial Frame Sync — SFS .....	8-6
8.5	RSSI RESET AND INITIALIZATION PROCEDURE .....	8-7
8.6	RSSI INTERFACE PROGRAMMING MODEL .....	8-8
8.7	RSSI TRANSMIT SHIFT REGISTER .....	8-10
8.8	RSSI TRANSMIT DATA REGISTER (TX) .....	8-10
8.9	RSSI RECEIVE SHIFT REGISTER .....	8-10
8.10	RSSI RECEIVE DATA REGISTER (RX) .....	8-10
8.11	RSSI CONTROL REGISTER A (CRA) .....	8-11
8.11.1.	CRA Prescale Modulus Select (PM7-PM0) Bits 0-7 .....	8-11
8.11.2.	CRA Frame Rate Divider Control (DC2-DC0) Bits 8-10 .....	8-12
8.11.3.	CRA Word Length Control (WL0, WL1) Bits 13 and 14 .....	8-13
8.11.4.	CRA Prescaler Range (PSR) Bit 15 .....	8-13
8.12	RSSI CONTROL REGISTER B (CRB) .....	8-13
8.12.1.	CRB Early/Late Frame Sync Bit (ELFS) Bit 2 .....	8-14
8.12.2.	CRB Gated Clock (GCK) Bit 3 .....	8-14
8.12.3.	CRB Frame Sync Direction (FSD) Bit 4 .....	8-14
8.12.4.	CRB Clock Source Direction (SCKD) Bit 5 .....	8-14
8.12.5.	CRB Clock Polarity Bit (SCKP) Bit 6 .....	8-15
8.12.6.	CRB MSB/LSB Position Bit (SHFD) Bit 7 .....	8-15
8.12.7.	CRB Frame Sync Length (FSL) Bit 8 .....	8-15



## Table of Contents (Continued)

Paragraph Number	Title	Page Number
8.12.8.	CRB Frame Sync Invert (FSI) Bit 9 .....	8-15
8.12.9.	CRB RSSI Enable Bit (SSIEN) Bit 10 .....	8-15
8.12.10.	CRB RSSI Mode Select (MOD) Bit 11 .....	8-15
8.12.11.	CRB RSSI Transmit Enable (TE) Bit 12 .....	8-15
8.12.12.	CRB RSSI Receive Enable (RE) Bit 13 .....	8-16
8.12.13.	CRB RSSI Transmit Interrupt Enable (TIE) Bit 14 .....	8-16
8.12.14.	CRB RSSI Receive Interrupt Enable (RIE) Bit 15 .....	8-17
8.13	RSSI STATUS REGISTER .....	8-17
8.13.1.	RSSISR Transmit/Receive Frame Sync (TRFS) Bit 3 .....	8-17
8.13.2.	RSSISR Transmitter Underrun Error (TUE) Bit 4 .....	8-18
8.13.3.	RSSISR Receiver Overrun Error (ROE) Bit 5 .....	8-18
8.13.4.	RSSISR Transmit Data Register Empty (TDE) Bit 6 .....	8-18
8.13.5.	RSSISR Receive Data Register Full (RDF) Bit 7 .....	8-19
8.14	TIME SLOT REGISTER — TSR .....	8-19
8.15	NORMAL AND NETWORK OPERATING MODES .....	8-19
8.15.1.	Normal Mode Transmit .....	8-19
8.15.2.	Normal Mode Receive .....	8-20
8.15.3.	Network Mode .....	8-20
8.15.3.1.	Network Mode Transmit .....	8-21
8.15.3.2.	Network Mode Receive .....	8-22

## SECTION 9 DSP56166 ON-CHIP PLL

9.1	INTRODUCTION .....	9-3
9.2	ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PCR0 .....	9-4
9.2.1.	PCR0 Feedback Divider Bits (YD7-YD0) Bits 0-7 .....	9-4
9.2.2.	PCR0 Input Divider Bits (ID3-ID0) Bits 8-11 .....	9-5
9.2.3.	PCR0 Power Divider Bits (PD3-PD0) Bits 12-15 .....	9-5
9.3	ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PCR1 .....	9-5
9.3.1.	PCR1 Reserved Bits — Bits 0-9 .....	9-5
9.3.2.	PCR1 CLKO Select Bits (CS1-CS0) Bits 10 and 11 .....	9-5
9.3.3.	PCR1 Phase Select Bit (PS) Bit 12 .....	9-6
9.3.4.	PCR1 PLL Power Down Bit (PLLD) Bit 13 .....	9-6
9.3.5.	PCR1 PLL Enable Bit (PLLE) Bit 14 .....	9-6
9.3.6.	PCR1 Voltage Controlled Oscillator Lock Bit (LOCK) Bit 15 .....	9-7

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
---------------------	-------	----------------

---

### APPENDIX A DSP56166 RAM BOOTSTRAP MODES

A.1.	INTRODUCTION.....	A-3
A.2.	BOOTSTRAP ROM.....	A-3
A.2.1.	Bootstrap Control Logic.....	A-3
A.2.2.	Bootstrap Firmware Program.....	A-4

### APPENDIX B DSP56166 APPLICATION EXAMPLES

### APPENDIX C DSP56166 PROGRAMMING SHEETS

C.1.	ADDRESSES.....	C-3
C.2.	INSTRUCTIONS.....	C-5
C.3.	CORE.....	C-18
C.4.	PLL.....	C-22
C.5.	TIMER.....	C-24
C.6.	CODEC.....	C-25
C.7.	GPI/O.....	C-27
C.8.	HOST.....	C-29
C.9.	RSSI.....	C-33

---

## Table of Contents (Continued)

Paragraph Number	Title	Page Number
---------------------	-------	----------------

---

# LIST of FIGURES

Figure Number	Title	Page Number
<b>SECTION 1</b>		
<b>DSP56166 OVERVIEW</b>		
1-1.	DSP56100 Family Product Literature . . . . .	1-3
1-2.	DSP56166 RAM and ROM Based Functional Block Diagram . . . . .	1-4
1-3.	DSP56166 RAM based Block Diagram . . . . .	1-6
1-4.	DSP5616 Core Block Diagram . . . . .	1-7
1-5.	Data ALU Architecture Block Diagram . . . . .	1-8
1-6.	AGU Block Diagram . . . . .	1-9
1-7.	Interrupt Priority Register IPR and IPR2 . . . . .	1-12
1-8.	Programing Model . . . . .	1-19
1-9.	Status Register Format . . . . .	1-21
1-10.	Operating Mode Register (OMR) . . . . .	1-24
<b>SECTION 2</b>		
<b>DSP56166 PIN DESCRIPTIONS</b>		
2-1.	Bus Operations . . . . .	2-6
2-2.	$\overline{\text{TA}}$ Controlled Accesses . . . . .	2-7
<b>SECTION 3</b>		
<b>OPERATING MODES AND MEMORY CONFIGURATION</b>		
3-1.	External Peripheral Bus Control Register (BCR2) . . . . .	3-3
3-2.	DSP56166 RAM based Memory Map . . . . .	3-4
3-3.	DSP56166 ROM Based Memory Map . . . . .	3-11

---

## List of Figures (Continued)

Figure Number	Title	Page Number
------------------	-------	----------------

---

### SECTION 4 DSP56166 I/O INTERFACE

4-1.	DSP56166 Input/Output Block Diagram . . . . .	4-4
4-2.	DSP56166 I/O and On-Chip Peripheral Memory Map . . . . .	4-5
4-3.	DSP56166 I/O Port B and C Programming Models . . . . .	4-8

### SECTION 5 HOST INTERFACE

5-1.	Host Interface Block Diagram . . . . .	5-4
5-2.	Host Interface - DSP Programming Model . . . . .	5-6
5-3.	Host Interface - Host Processor Programming Model . . . . .	5-8

### SECTION 6 DSP56166 ON-CHIP SIGMA/DELTA CODEC

6-1.	DSP56166 On-chip $\Sigma\Delta$ Functional Diagram . . . . .	6-3
6-2.	DSP56166 Codec General Block Diagram . . . . .	6-5
6-3.	DSP56166 Codec Analog Input and Output Diagram . . . . .	6-6
6-4.	On-chip Codec Programming Model . . . . .	6-7
6-5.	On-Chip Codec Programming Model Summary . . . . .	6-15
6-6.	A/D Comb Filter Transfer Function . . . . .	6-16
6-7.	Log Magnitude Frequency Response of the A/D Comb Filter for F=2.048 MHz and D=128 . . . . .	6-17
6-8.	Log Magnitude Frequency Response of the A/D Comb Filter in the Band 0-4KHz for F=2.048 MHz and D=128 . . . . .	6-18
6-9.	IIR Decimation and A/D Section Log Magnitude Frequency Response for F=2.048 MHz and D=128 . . . . .	6-20
6-10.	Log Magnitude Frequency Responses of the Three Sections of the D/A F=2048 MHz and D=128 . . . . .	6-21
6-11.	D/A Comb Filter Transfer Function . . . . .	6-22
6-12.	Log Magnitude Frequency Response of the D/A Comb Filter for F=2.048MHz and D=128 . . . . .	6-23
6-13.	Analog Low-pass Filter Transfer Function . . . . .	6-24

---

## List of Figures (Continued)

Figure Number	Title	Page Number
6-14.	Log Magnitude Frequency Response of the D/A Analog Low-pass Filter for F=2.048MHz . . . . .	6-25
6-15.	Log Magnitude Frequency Response of the D/A Section for F=2.048 MHz and D=128 . . . . .	6-26
6-16.	Log Magnitude Frequency Response of the D/A Section for F=2.048 MHz and D=128 . . . . .	6-27
6-17.	IIR Interpolation and D/A Section Log Magnitude Frequency Response for F=2.048 MHz and D=128 . . . . .	6-29

### SECTION 7 16-BIT TIMER AND EVENT COUNTER

7-1.	16-bit Timer General Block Diagram . . . . .	7-4
7-2.	Timer Programming Model . . . . .	7-5
7-3.	Timer Control Register . . . . .	7-6
7-4.	Standard Timer Operation with Overflow Interrupt . . . . .	7-9
7-5.	Standard Timer Disable . . . . .	7-9
7-6.	Write to the Count Register After Writing to the Preload Register When the Timer is Disabled . . . . .	7-10
7-7.	Timer Disable After a Write to the Count Register . . . . .	7-10
7-8.	Write to the Count Register when the Timer is Enabled . . . . .	7-11
7-9.	Write to DC7-DC0 when the Timer is Enabled . . . . .	7-12
7-10.	Standard Timer Operation with Compare Interrupt . . . . .	7-13

### SECTION 8 REDUCED SSI (RSSI0 and RSSI1)

8-1.	RSSI External Continuous Clock . . . . .	8-4
8-2.	RSSI Internal Gated Clock . . . . .	8-4
8-3.	RSSI External Gated Clock . . . . .	8-5
8-4.	RSSI Clock Generator Functional Block Diagram . . . . .	8-5
8-5.	RSSI Frame Sync Generator Functional Block Diagram . . . . .	8-6
8-6.	Serial Clock and Frame Sync Timing . . . . .	8-7
8-7.	RSSI Programming Model . . . . .	8-9
8-8.	Normal Mode Timing . . . . .	8-20
8-9.	Network Mode Timing (Gated Clock) . . . . .	8-23
8-10.	Network Mode Timing (Continuous Clock) . . . . .	8-24

---

## List of Figures (Continued)

Figure Number	Title	Page Number
------------------	-------	----------------

---

### SECTION 9 DSP56166 ON-CHIP PLL

9-1.	On-Chip Frequency Synthesizer Programming Model Summary. . . . .	9-8
------	--	-----

### APPENDIX A DSP56166 RAM BOOTSTRAP MODES

A-1.	DSP56166 Bootstrap Program Listing . . . . .	A-5
------	--	-----

### APPENDIX B DSP56166 APPLICATION EXAMPLES

B-1.	No Glue Logic, Low Cost Memory Port Bootstrap — Mode 0. . . . .	B-2
B-2.	DSP56166 Host Bootstrap Example — Mode 1 . . . . .	B-2
B-3.	32K Words of External Program ROM — Mode 2 . . . . .	B-3
B-4.	Reset Circuit . . . . .	B-4
B-5.	Reset Circuit Using 555 Timer . . . . .	B-4

### APPENDIX C DSP56166 PROGRAMMING SHEETS

# LIST of TABLES

Table Number	Title	Page Number
<b>SECTION 1</b>		
<b>DSP56166 OVERVIEW</b>		
1-1	DSP56166 Feature List .....	1-5
1-2	Interrupt Sources .....	1-11
1-3	Status Register (SR) Interrupt Mask Bits .....	1-11
1-4	Interrupt Priority Level Bits .....	1-12
1-5	External Interrupt Trigger Mode Bits .....	1-12
1-6	Interrupt Priority Level Bits for IRQC .....	1-13
1-7	Exception Priorities within an IPL .....	1-14
1-8	Stack Pointer Values .....	1-23
1-9	Operating Mode Summary — DSP56166 RAM Based Part .....	1-24
1-10	Operating Mode Summary — DSP56166 ROM Based Part .....	1-25
1-11	Actions of the Saturation Mode (SA=1) .....	1-26
1-12	DSP5616 Addressing Modes .....	1-32
<b>SECTION 2</b>		
<b>DSP56166 PIN DESCRIPTIONS</b>		
2-1	Functional Group Pin Allocations .....	2-3
<b>SECTION 3</b>		
<b>OPERATING MODES</b>		
<b>AND MEMORY CONFIGURATION</b>		
3-1	Operating Mode Summary	
	Program RAM Part .....	3-6
3-2	Data Mapping for External Bus Bootstrap .....	3-9
3-3	DSP56166 ROM Based Operating Modes .....	3-12



---

## List of Tables (Continued)

Table Number	Title	Page Number
-----------------	-------	----------------

---

### SECTION 4 DSP56166 I/O INTERFACE

#### SECTION 5 HOST INTERFACE

5-1	Host Interface Interrupt Structure . . . . .	5-9
5-2	HREQ Pin Definition - Interrupt Mode . . . . .	5-13
5-3	HREQ Pin Definition - DMA Mode . . . . .	5-13
5-4	Host Mode (HM1, HM0) Bit Definition . . . . .	5-14
5-5	INIT Execution Definition . . . . .	5-15

#### SECTION 6 DSP56166 ON-CHIP SIGMA/DELTA CODEC

6-1	On-chip Codec Main Features . . . . .	6-4
6-2	Decimation/Interpolation Ratio Control . . . . .	6-9
6-3	Audio Level Control . . . . .	6-11
6-4	Audio Level Control with DSP Filter Gain . . . . .	6-12
6-5	Microphone Gain Control . . . . .	6-13
6-6	Example of a Four Biquads IIR Decimation and Compensation Filter . . . . .	6-19
6-7	Example of a Four Biquad IIR Interpolation and Compensation Filter . . . . .	6-28

#### SECTION 7 16-BIT TIMER AND EVENT COUNTER

7-1	TOUT Pin Function . . . . .	7-7
7-2	Timer Range and Resolution . . . . .	7-8

---

## List of Tables (Continued)

Table Number	Title	Page Number
-----------------	-------	----------------

---

### SECTION 8 REDUCED SSI (RSSI0 and RSSI1)

8-1	RSSI Operating Modes .....	8-3
8-2	RSSI bit clock as a function of Fosc and PM7-PM0 (PSR=0) .....	8-12
8-3	RSSI Data Word Lengths .....	8-13

### SECTION 9 DSP56166 ON-CHIP PLL

9-1	CLKOUT Pin Control .....	9-6
9-2	PLL Operations .....	9-7

### APPENDIX A DSP56166 RAM BOOTSTRAP MODES

### APPENDIX B DSP56166 APPLICATION EXAMPLES

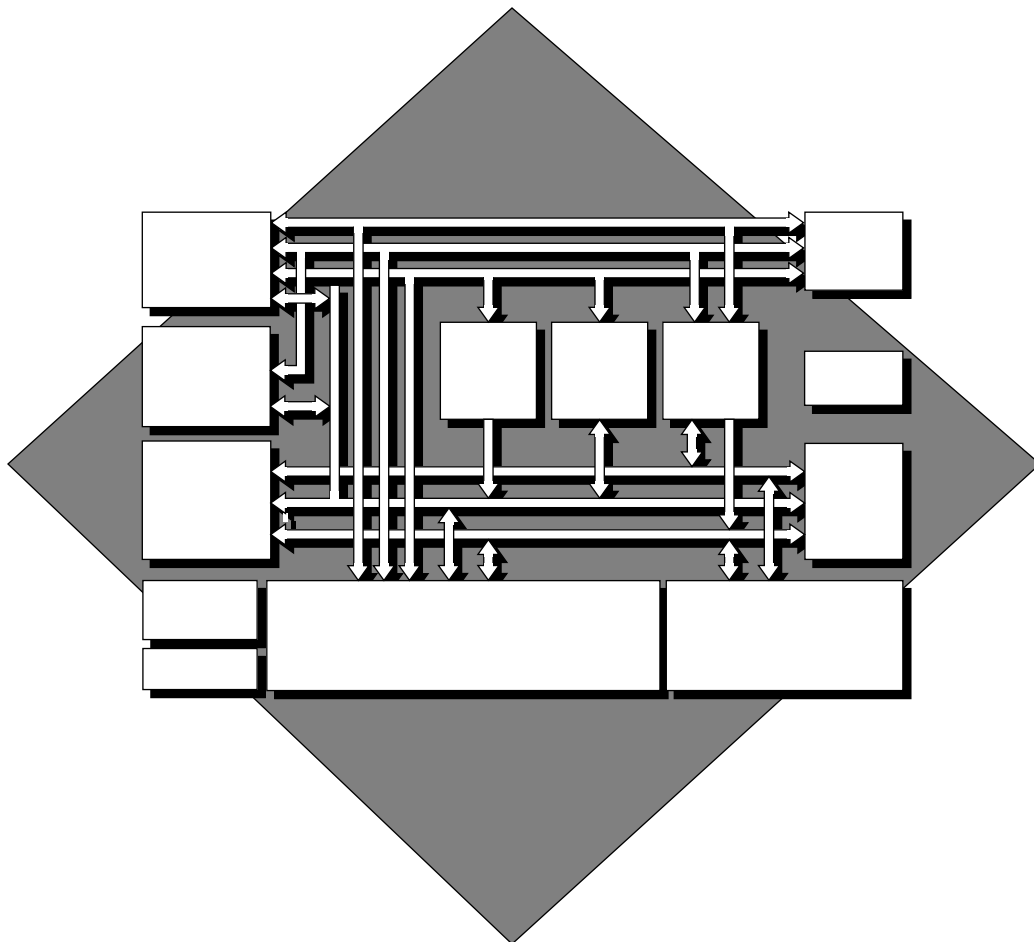
### APPENDIX C DSP56166 PROGRAMMING SHEETS

List of Tables (Continued)		
Table Number	Title	Page Number

---

## SECTION 1

# DSP56166 OVERVIEW



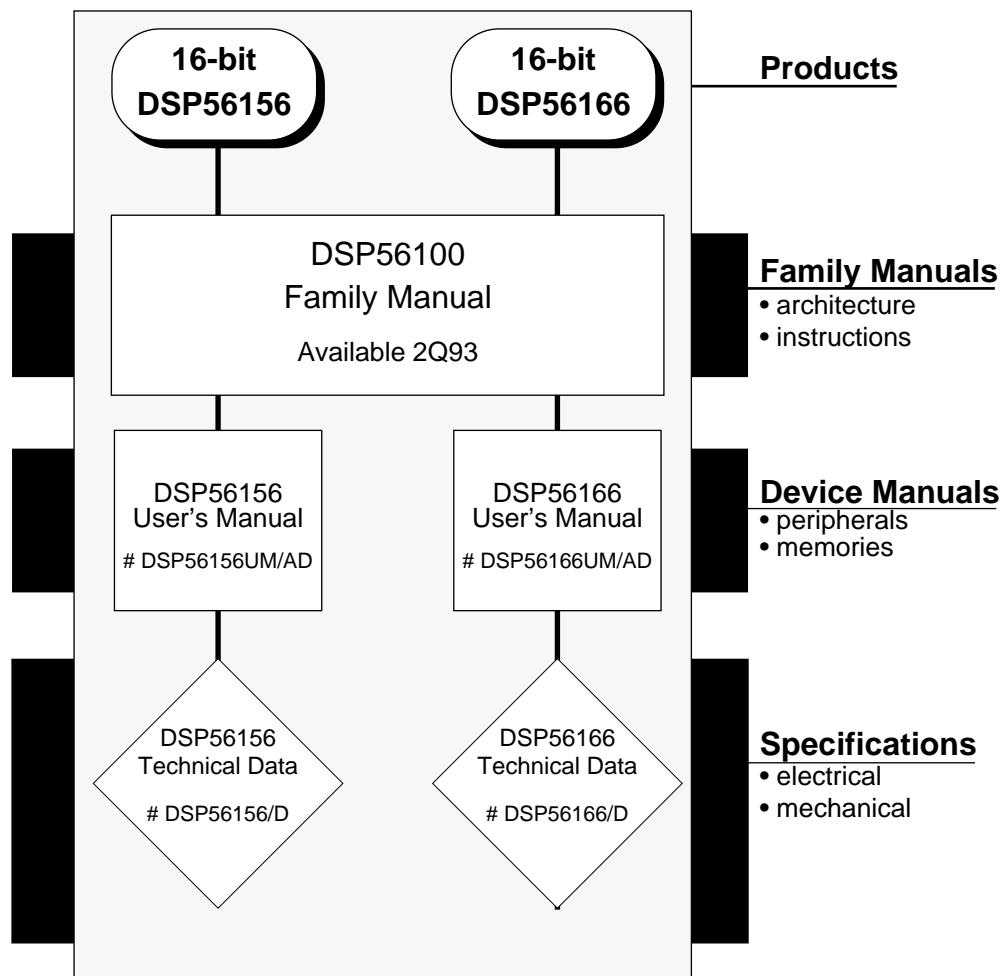
# SECTION CONTENTS

---

1.1	INTRODUCTION .....	1-3
1.2	DSP5616 CORE BLOCK DIAGRAM DESCRIPTION .....	1-4
1.2.1.	Data Buses .....	1-4
1.2.2.	Address Buses .....	1-5
1.2.3.	Data ALU .....	1-6
1.2.4.	Address Generation Unit (AGU) .....	1-9
1.2.5.	Program Control Unit (PCU) .....	1-10
1.3	MEMORY ORGANIZATION .....	1-13
1.4	EXTERNAL BUS, I/Os and ON-CHIP PERIPHERALS .....	1-14
1.4.1.	Memory Expansion Port (Port A) .....	1-15
1.4.2.	General Purpose I/O (Port B, Port C) .....	1-15
1.4.3.	RSSI0 and RSSI1 .....	1-16
1.4.4.	Timer .....	1-16
1.4.5.	Host Interface (HI) .....	1-16
1.5	OnCE .....	1-17
1.6	PROGRAMMING MODEL .....	1-17
1.6.1.	Data ALU .....	1-17
1.6.2.	Address Generation Unit .....	1-20
1.6.3.	Program Control Unit .....	1-20
1.7	INSTRUCTION SET SUMMARY .....	1-26
1.7.1.	Instruction Groups .....	1-26
1.7.2.	Instruction Formats .....	1-30
1.7.3.	Addressing Modes .....	1-31
1.7.4.	Address Arithmetic .....	1-33

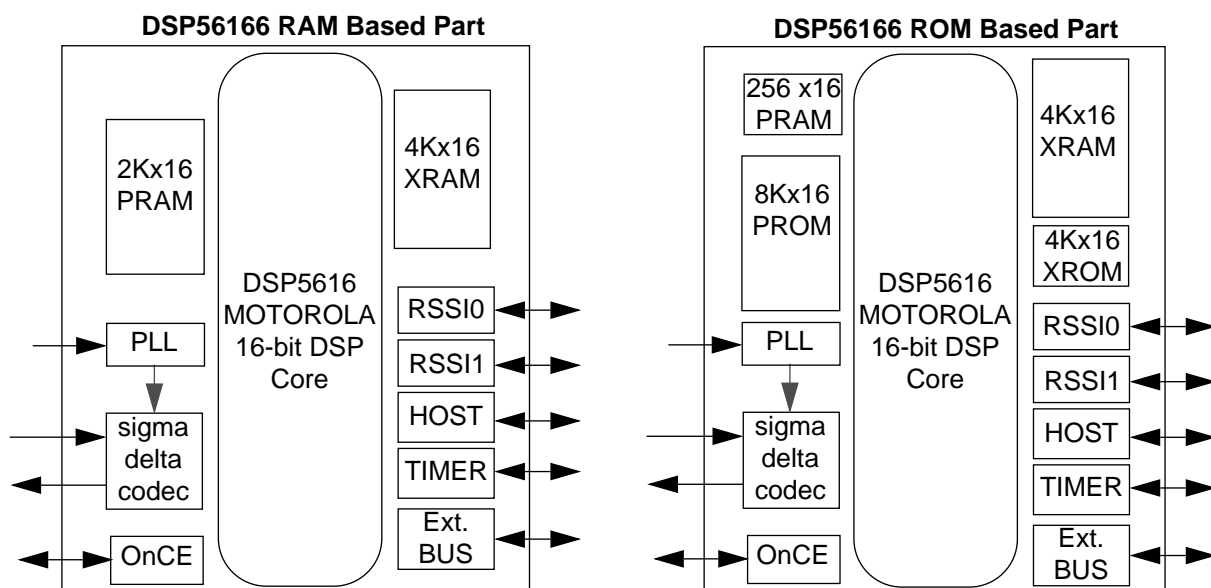
## 1.1 INTRODUCTION

This manual is intended to be used with the DSP56100 Family Manual (see Figure 1-1). The DSP56100 Family Manual provides a description of the components of the DSP5616 core that are common to all DSP56100 family processors and includes a detailed description of the basic DSP56100 family instruction set. The DSP56166 User's Manual provides a brief overview of the core processor and a detailed descriptions of the memory and peripherals that are specific to the DSP56166.



**Figure 1-1 DSP56100 Family Product Literature**

A general block diagram of the DSP56166 is shown in Figure 1-2. It is available as a RAM based or ROM based part (see Section 3.3 for information on the ROM based part). The DSP56166 is optimized for applications such as medium to low bit rate speech encoding but can also be used in many other types of applications.



**Figure 1-2 DSP56166 RAM and ROM Based Functional Block Diagram**

Table 1-1 is a list of the DSP56166 primary features. The core features are common to any product using the DSP5616 Core. Figure 1-3 provides a more detailed block diagram of the DSP56166 RAM based part.

## 1.2 DSP56166 CORE BLOCK DIAGRAM DESCRIPTION

The heart of the DSP56166 architecture is a 16-bit multiple-bus core processor called the DSP5616 which was designed specifically for real-time digital signal processing (DSP). The overall core architecture is presented here and can be seen in Figure 1-4. For a detailed description of the core processor, see the **DSP56100 Family User's Manual**.

### 1.2.1 Data Buses

Data movement on the chip occurs over three bidirectional 16-bit buses: the X Data Bus (XDB), the Program Data Bus (PDB), and the Global Data Bus (GDB). Data transfer between the Data ALU and the X Data Memory occurs over the XDB when one memory access is performed or over the XDB and the GDB when two simultaneous memory reads are performed. All other data transfers occur over the Global Data Bus. Instruction word pre-fetches take place in parallel over the PDB. The bus structure supports general register to register, register to memory, memory to register, and memory to memory data movement and can transfer up to three 16-bit words in the same instruction cycle.

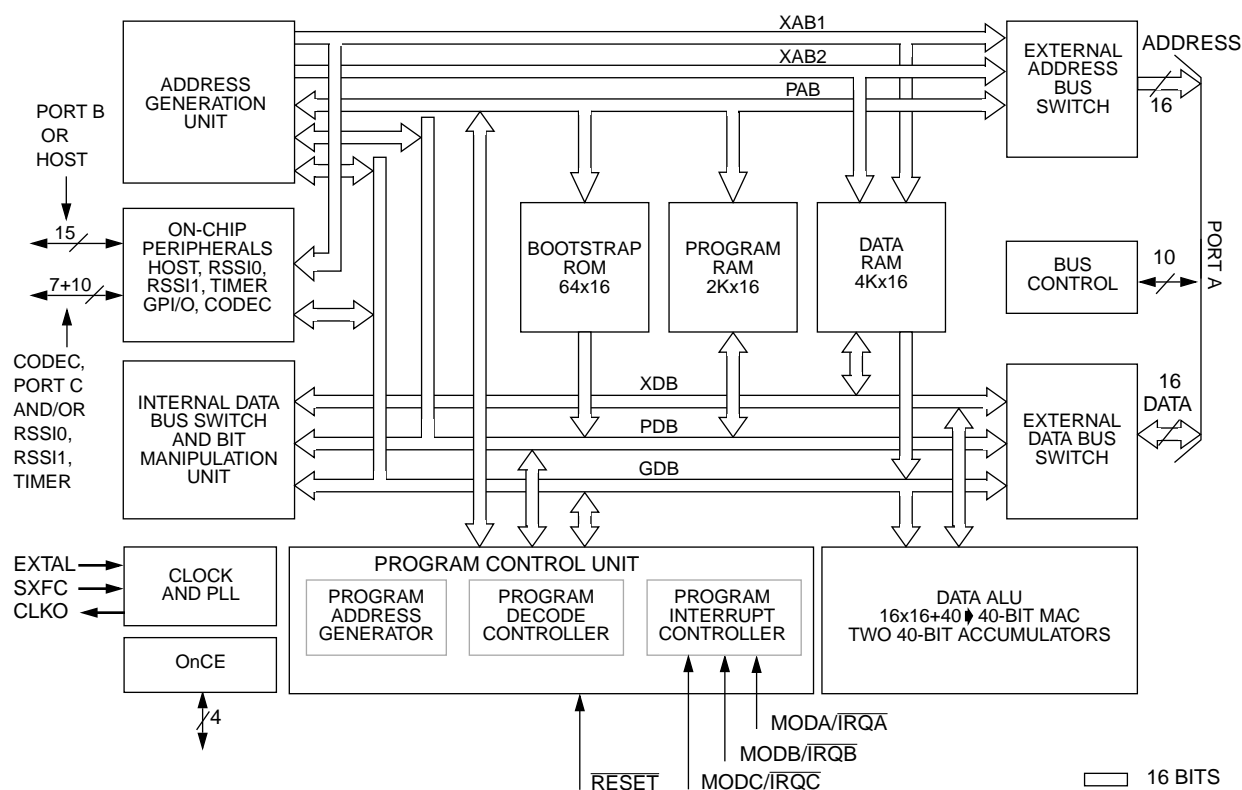
**Table 1-1 DSP56166 Feature List**

<b>DSP5616 Core Features</b>	<b>DSP56166 On-chip Resources</b>
<ul style="list-style-type: none"> <li>Up to 30 Million Instructions per Second (MIPS) at 60 MHz.— 33.3 ns instruction cycle</li> <li>Single-cycle 16 x 16-bit parallel multiply-accumulate</li> <li>2 x 40-bit accumulators with extension byte</li> <li>Fractional and integer arithmetic with support for multiprecision arithmetic</li> <li>Highly parallel instruction set with unique DSP addressing modes</li> <li>Nested hardware DO loops including infinite loops</li> <li>Two instruction LMS adaptive filter loop</li> <li>Fast auto-return interrupts</li> <li>Three external interrupt request pins</li> <li>Three 16-bit internal data buses and three 16-bit internal address buses</li> <li>Programmable access time on the external bus</li> <li>On-chip peripheral registers memory mapped in data memory space</li> <li>Off-chip peripheral space with programmable access time memory mapped in data memory space</li> <li>Low power wait and stop modes</li> <li>On-Chip Emulation (OnCE) for unobtrusive, processor speed independent debugging</li> <li>Operating frequency down to DC</li> <li>5V single power supply</li> <li>Low power (HCMOS)</li> </ul>	<p><u>DSP56166 RAM Based Part:</u>            4K x 16 on-chip data RAM            2K x16 on-chip program RAM            One bootstrap ROM            Bootstrap loading from external byte wide PROM, Host Interface, or Reduced Synchronous Serial Interface 0 (RSSI0)</p> <p><u>DSP56166 ROM Based Part:</u>            4K x 16 on-chip data RAM            4K x16 on-chip data ROM            256 x 16 on-chip program RAM            8Kx16 on-chip program ROM</p> <p><u>DSP56166 RAM based and ROM Based Part:</u>            One external 16-bit address bus            One external 16-bit data bus            On-chip <math>\Sigma\Delta</math> voice band codec (A/D-D/A)                — Internal voltage reference (2/5 of positive power supply)                — No off-chip components required            25 general purpose I/O pins            On-chip, programmable PLL            Byte-wide Host Interface with DMA support            Two independent reduced synchronous serial interfaces            One 16-bit timer            112 pin quad flat pack packaging</p>

## 1.2.2 Address Buses

Addresses are specified for internal X Data Memory on two unidirectional 16-bit buses — X Address Bus One (XAB1) and X Address Bus Two (XAB2). Program memory addresses are specified on the PAB. External memory spaces are addressed via a single 16-bit, unidirectional address bus driven by a three input multiplexer that can select the XAB1, XAB2, or PAB. One instruction cycle is needed for each external memory access. There is no speed penalty if only one external memory space is accessed in an instruction and if no wait states are inserted in the external bus cycle. If two or three external memory spaces are accessed in a single instruction, there will be a one or two instruction cycle execution delay, respectively, or more if wait states are inserted on the external bus. A bus arbitrator controls external accesses, making it transparent to the user. See the DSP56100 Family User's Manual for additional information.



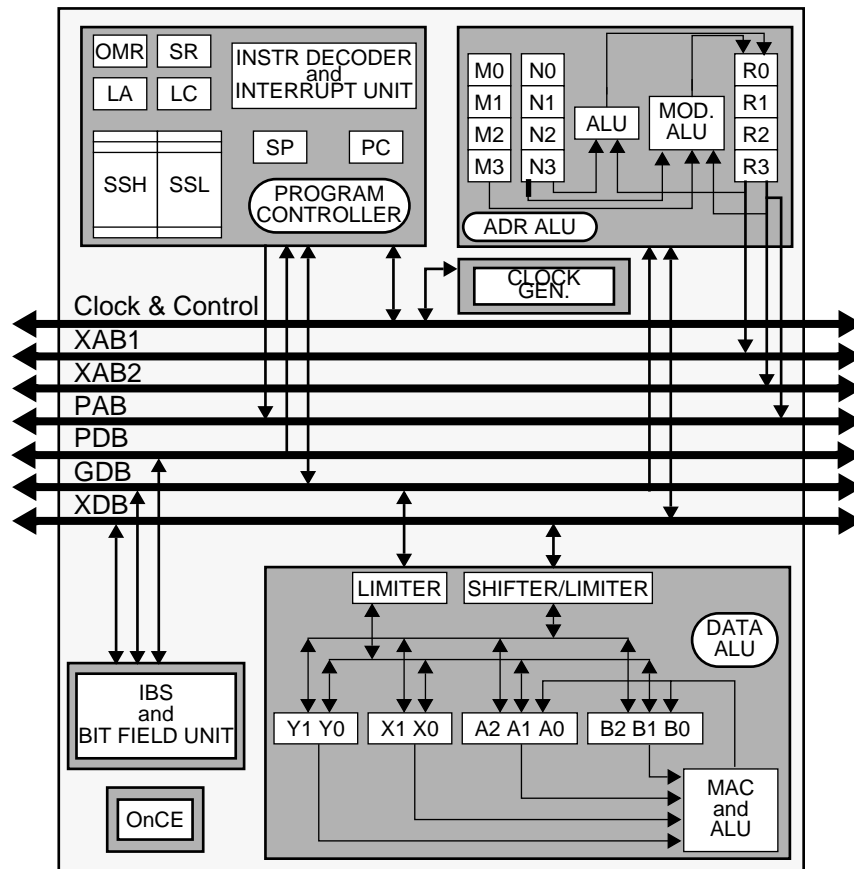


**Figure 1-3 DSP56166 RAM Based Part Block Diagram**

## 1.2.3 Data ALU

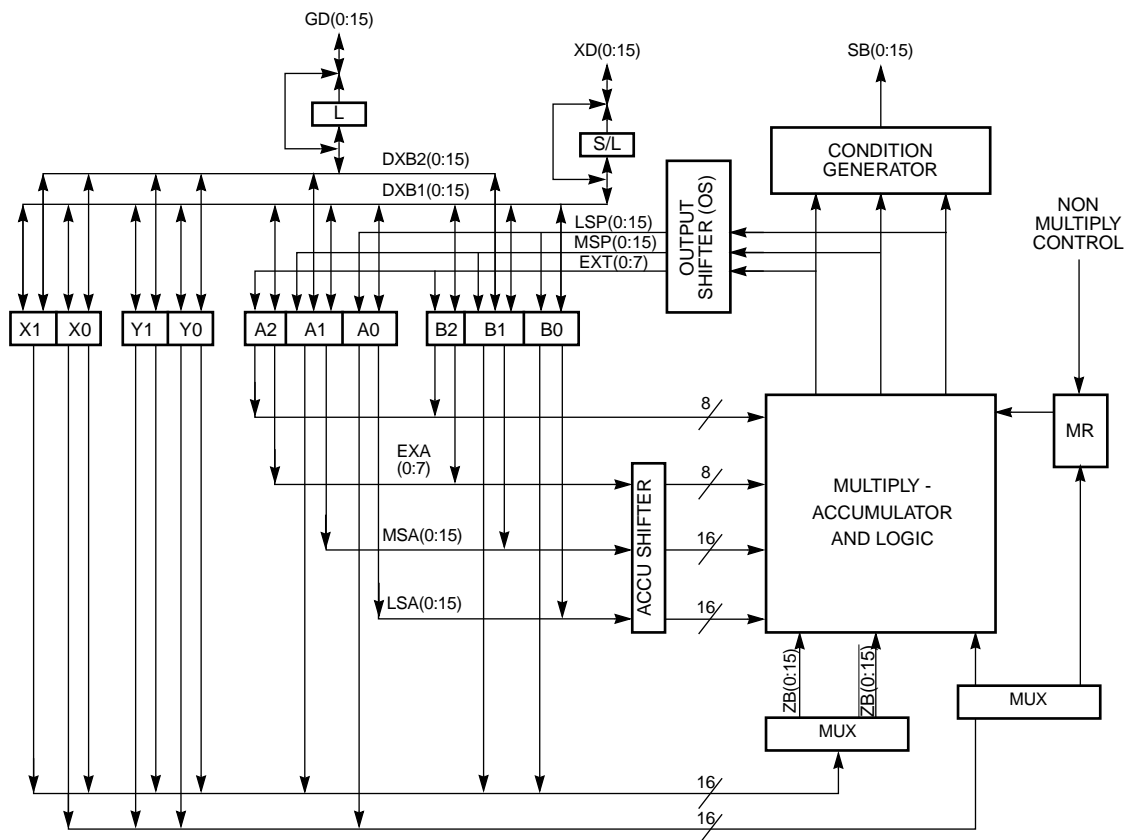
The Data ALU performs all arithmetic and logical operations on data operands and consists of:

- four 16-bit input registers,
- two 32-bit accumulator registers,
- two 8-bit accumulator extension registers,
- an accumulator shifter,
- an output shifter,
- one data bus shifter/limiter,
- one data bus limiter,
- and a parallel, single cycle, non-pipelined Multiply-Accumulator (MAC) unit.



**Figure 1-4 DSP5616 Core Block Diagram**

Data ALU registers may be read or written via the XDB and GDB as 16-bit operands (see Figure 1-5). The Data ALU is capable of multiplication, multiply-accumulate with positive or negative accumulation, addition, subtraction, shifting, and logical operations in one instruction cycle. Data ALU arithmetic operations generally use fractional two's complement arithmetic. Some signed/unsigned and integer operations are also available. Data ALU source operands may be 16, 32, or 40 bits and may originate from input registers and/or accumulators. Data ALU results are always stored in one of the accumulators. The upper 16-bits of an accumulator can be used as a multiplier input. Arithmetic operations always have a 40-bit result and logical operations are performed on 16-bit operands yielding 16-bit results in one of the two accumulators.



**Figure 1-5 Data ALU Architecture Block Diagram**

The DSP56166 supports the two's complement representation of binary numbers. Unsigned numbers are only supported by the multiply and multiply-accumulate instruction. For fractional arithmetic, the 31-bit product is added to the 40-bit contents of either the A or B accumulator. The 40-bit sum is stored back in the same accumulator. This multiply/accumulate is a single cycle operation (no pipeline). Integer operations always generate a 16-bit result located in the accumulator MSP (A1 or B1). Full precision integer operations are possible using the IMPY or IMAC instructions. Saturation arithmetic is provided to selectively limit overflow when reading a data ALU accumulator register.

The DSP56166 implements two types of rounding: convergent rounding and two's complement rounding. The type of rounding is selected by the status register rounding bit (R bit).

The logic unit in the MAC array performs the logical operations AND, OR, EOR, and NOT on data ALU registers. The logic unit is 16 bits wide and operates on data in the MSP portion of the accumulator. The LSP and EXT portions of the accumulator are not affected. See the DSP56100 Family User's Manual for additional information.

### 1.2.4 Address Generation Unit (AGU)

The AGU performs all address storage and effective address calculations necessary to address data operands in memory (see Figure 1-6). This unit operates in parallel with other chip resources to minimize address generation overhead. The AGU can implement three types of arithmetic: linear, modulo, and reverse carry. The Address ALU contains four address registers (R0-R3), four offset registers (N0-N3), and four modifier registers (M0-M3). The address registers are 16-bit registers which may contain addresses or data. Each address register may be output to the PAB and XAB1. R3 may be output to XAB2 when R0, R1, or R2 are output to XAB1. The modifier and offset registers are 16-bit registers which are normally used to control updating of the address registers. Any register can also be used as a general purpose register for storage of 16-bit data.

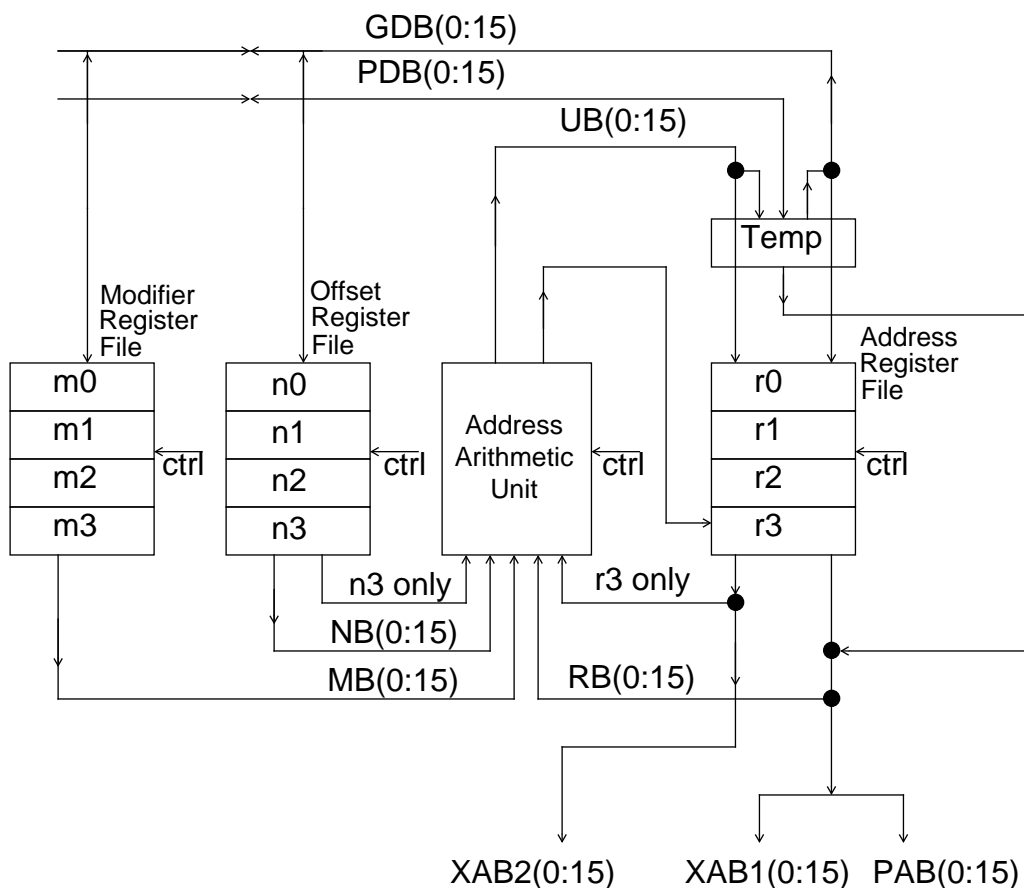


Figure 1-6 AGU Block Diagram

AGU registers may be read or written by the GDB as 16-bit operands. The AGU can generate two 16-bit addresses every instruction cycle: one for either the XAB1 or PAB and one for XAB2. The ALU can directly address 65536 locations on the XAB1 and 65536 locations on the XAB2. See the DSP56100 Family User's Manual for additional information.

### **1.2.5 Program Control Unit (PCU)**

The PCU performs instruction fetch, instruction decoding, hardware REP, DO loop control, and exception processing. Two interrupt priority registers (IPR and IPR2) are used to program the priority level of the interrupts. They have control bits for the three external interrupt pins and each of the on-chip peripherals (see Figure 1-7, Table 1-4, Table 1-5 and Table 1-6).

There are 63 interrupts available and one reserved on the DSP56166. Table 1-2 shows each of these interrupts with their respective starting address and Interrupt Priority Level (IPL). The four level three interrupts are not maskable and if two or more are simultaneously issued, their priority is (1) Hardware Reset, (2) Illegal Instruction, (3) Stack Error, and (4) the SWI instruction. The reserved interrupt is not available for use.

The PCU contains five directly addressable registers in addition to the program counter (PC). These are the loop address (LA), loop counter (LC), status register (SR), operating mode register (OMR), and stack pointer (SP). The PC also contains a 15 level, 32-bit wide system stack memory. The 16-bit PC can address 65,536 locations in program memory space. See the DSP56100 Family User's Manual for additional information.

#### **1.2.5.1 Interrupt Priority Structure**

Four levels of interrupt priority are provided. Interrupt priority levels (IPLs) numbered 0, 1, and 2, are maskable with level 0 as the lowest level. Level 3 (the highest level), is non-maskable. The only level 3 interrupts are Reset, Illegal Instruction, Stack Error and SWI. The interrupt mask bits (I1, I0) in the status register reflect the current processor priority level and indicate the interrupt priority level needed for an interrupt source to interrupt the processor (see Table 1-3). Interrupts are inhibited for all priority levels less than the current processor priority level. However, level 3 interrupts are not maskable and therefore can always interrupt the processor.

Table 1-2 Interrupt Sources

Interrupt Starting Address	IPL	Interrupt Source
\$0000	3	Hardware RESET
\$0002	3	Illegal Instruction
\$0004	3	Stack Error
\$0006	3	Reserved
\$0008	3	SWI
\$000A	0-2	IRQA
\$000C	0-2	IRQB
\$000E	0-2	IRQC
\$0010	0-2	RSSI0 Receive Data with Exception Status
\$0012	0-2	RSSI0 Receive Data
\$0014	0-2	RSSI0 Transmit Data with Exception Status
\$0016	0-2	RSSI0 Transmit Data
\$0018	0-2	RSSI1 Receive Data with Exception Status
\$001A	0-2	RSSI1 Receive Data
\$001C	0-2	RSSI1 Transmit Data with Exception Status
\$001E	0-2	RSSI1 Transmit Data
\$0020	0-2	Timer Overflow
\$0022	0-2	Timer Compare
\$0024	0-2	Host DMA Receive Data
\$0026	0-2	Host DMA Transmit Data
\$0028	0-2	Host Receive Data
\$002A	0-2	Host Transmit Data
\$002C	0-2	Host Command (default)
\$002E	0-2	Codec Receive/Transmit
\$0030	0-2	Available for Host Command
\$0032	0-2	Available for Host Command
\$0034	0-2	Available for Host Command
.	.	.
.	.	.
.	.	.
\$007E	0-2	Available for Host Command

Table 1-3 Status Register (SR) Interrupt Mask Bits

I1	I0	Exceptions Permitted	Exceptions Masked
0	0	IPL 0,1,2,3	None
0	1	IPL 1,2,3	IPL0
1	0	IPL 2,3	IPL 0,1
1	1	IPL 3	IPL 0,1,2

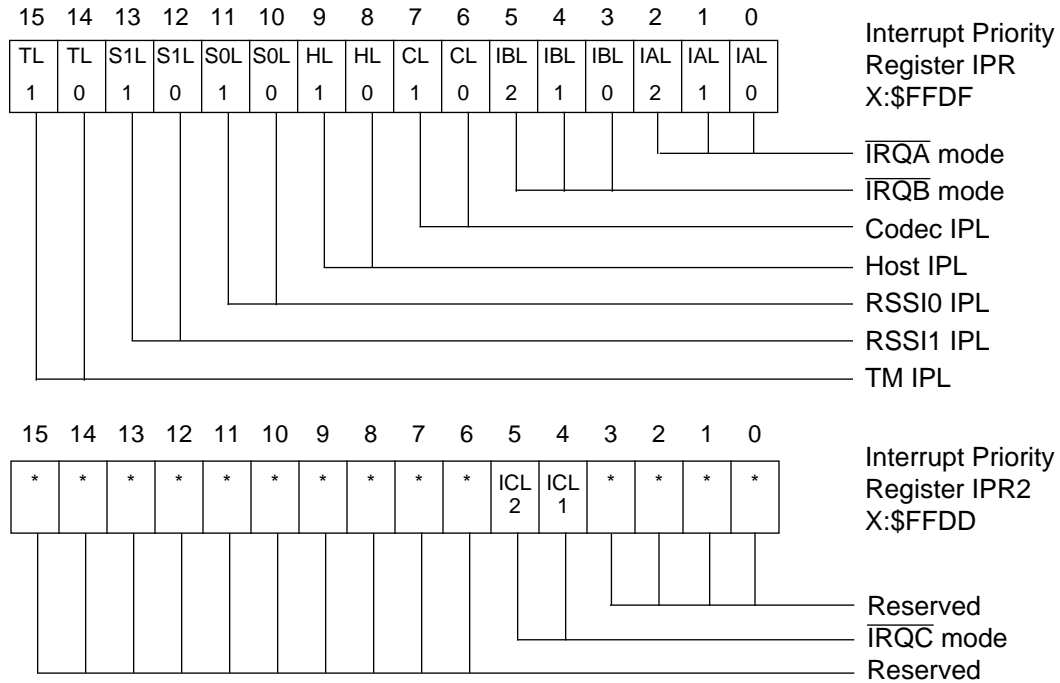


Figure 1-7 Interrupt Priority Register IPR and IPR2

Table 1-4 Interrupt Priority Level Bits

xxL1	xxL0	Enabled	IPL
0	0	No	-
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

Table 1-5 External Interrupt Trigger Mode Bits

IxL2	Trigger Mode
0	Level
1	Negative Edge

**Table 1-6 Interrupt Priority Level Bits for IRQC**

ICL2	ICL1	Enabled	IPL
0	0	Not enabled	-
0	1	Level	1
1	0	Not Enabled	-
1	1	Neg. Edge	1

## 1.2.5.2 Interrupt Priority Levels (IPL)

The interrupt priority level for each on-chip peripheral device and for each external interrupt source ( $\overline{\text{IRQA}}$ ,  $\overline{\text{IRQB}}$ ,  $\overline{\text{IRQC}}$ ) can be programmed under software control. Each on-chip or external peripheral device can be programmed to one of the three maskable priority levels (IPL 0, 1, or 2). Interrupt priority levels are set by writing to the Interrupt Priority Registers IPR and IPR2 shown in Figure 1-7. These read/write registers specify the interrupt priority level for each of the interrupting devices (Codec, Host, RSSIs, Timer,  $\overline{\text{IRQA}}$ ,  $\overline{\text{IRQB}}$ ,  $\overline{\text{IRQC}}$ ). In addition, this register specifies the trigger mode of both external interrupt sources  $\overline{\text{IRQA}}$ ,  $\overline{\text{IRQB}}$  and it is used to enable or disable the individual external interrupts. This register is cleared on  $\overline{\text{RESET}}$ . Table 1-4 defines the interrupt priority level bits. Table 1-5 defines the external interrupt trigger mode bits for  $\overline{\text{IRQA}}$ ,  $\overline{\text{IRQB}}$ . Table 1-6 defines the interrupt priority level for the third external interrupt  $\overline{\text{IRQC}}$ .

## 1.2.5.3 Exception Priorities within an IPL

If more than one exception is pending when an instruction is executed, the interrupt with the highest priority level is serviced first. When multiple interrupt requests with the same IPL are pending, a second fixed priority structure within that IPL determines which interrupt is serviced. The fixed priority of interrupts within an IPL and the interrupt enable bits for all interrupts are shown in Table 1-7. The interrupt enable bits for the Host, RSSIs, and TM are located in the control registers associated with their respective on-chip peripherals.

## 1.3 MEMORY ORGANIZATION

Two independent memory space configurations (X data and P program), are described in section 3. These memory spaces are configured by control bits in the Operating Mode Register, OMR. MA and MB control the program memory map and EX controls the data memory map.



Table 1-7 Exception Priorities within an IPL

Priority	Exception	Enabled by	IP Reg. Bit No.	Control Register Address
<b>Level 3 (Non-maskable)</b>				
Highest	Hardware RESET	—	—	—
	Illegal Instruction Interrupt	—	—	—
	Stack Error	—	—	—
Lowest	SWI	—	—	—
<b>Level 0, 1, 2 (Maskable)</b>				
Highest	IRQA (External Interrupt)	IRQA mode bits	0, 1	X:\$FFDF
	IRQB (External Interrupt)	IRQB mode bits	3, 4	X:\$FFDF
	IRQC (External Interrupt)	IRQC mode bits	IPR2 4,5	X:\$FFDD
	Host Command Interrupt	HCIE	2	X:\$FFC4
	Host/DMA RX Data Interrupt	HRIE	0	X:\$FFC4
	Host/DMA TX Data Interrupt	HTIE	1	X:\$FFC4
	RSSI0 RX Data with Exception Status	RIE	15	X:\$FFD1
	RSSI0 RX Data	RIE	15	X:\$FFD1
	RSSI0 TX Data with Exception Status	TIE	14	X:\$FFD1
	RSSI0 TX Data	TIE	14	X:\$FFD1
	RSSI1 RX Data with Exception Status	RIE	15	X:\$FFD9
	RSSI1 RX Data	RIE	15	X:\$FFD9
	RSSI1 TX Data with Exception Status	TIE	14	X:\$FFD9
	RSSI1 TX Data	TIE	14	X:\$FFD9
	Timer Overflow Interrupt	OIE	9	X:\$FFEC
Lowest	Timer Compare Interrupt	CIE	10	X:\$FFEC

#### 1.4 EXTERNAL BUS, I/Os and ON-CHIP PERIPHERALS

Five on-chip peripherals are provided on the DSP56166: an 8-bit parallel Host MPU/DMA Interface, a 16-bit timer, two Reduced Synchronous Serial Interfaces (RSSI1 and RSSI0), and a sigma-delta codec (see Figure 1-2). The DSP56166 provides 16 pins for an external

address bus and 16 pins for an external data bus. These pins are grouped to form the Port A bus interface. The DSP56166 also provides 25 programmable I/O pins. These pins may be used as general purpose I/O pins or allocated to an on-chip peripheral. They are separate from the DSP56166 codec, address buses, and data buses and are grouped as two I/O ports (B and C).

Port B is a 15-pin I/O interface that may be used as a general purpose I/O or as a Host MPU/DMA Interface. The Host MPU/DMA Interface provides a dedicated 8-bit parallel port to a host microprocessor or DMA controller and can provide debugging facilities via host exceptions.

Port C is a 10-pin I/O interface that may be used as a general purpose I/O or as a Timer and two identical Reduced Synchronous Serial Interfaces.

The sigma-delta codec has seven dedicated pins which are not available as general purpose I/O. Both analog to digital (A/D) and digital to analog (D/A) converters are provided on-chip. The final decimation, antialiasing and compensation filters for the A/D and interpolation, reconstruction, and compensation filters for the D/A converter are implemented in software on the DSP providing the user considerable flexibility in the codec filter characteristics.

#### **1.4.1 Memory Expansion Port (Port A)**

The DSP56166 expansion port is designed to synchronously interface over a common 16-bit data bus with a wide variety of memory and peripheral devices such as high speed static RAMs, slower memory devices, DSPs, and MPUs in master/slave configurations. This capability is possible because the external bus cycle time is programmable. The expansion bus timing is controlled by a two Bus Control Registers (BCR & BCR2). The bus control registers control the timing of the bus interface signals and data lines. Each of the two memory spaces, X data and Program data, has its own 5 control bits in the BCR which can be programmed for up to 31 wait states (one wait state is equal to a clock period, or equivalently, one-half of an instruction cycle). The access time on the external peripheral memory space can be controlled with 5 control bits in BCR2. In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces or external peripherals.

#### **1.4.2 General Purpose I/O (Port B, Port C)**

Each Port B and C pin may be programmed as a general purpose I/O pin or as a dedicated on-chip peripheral pin under software control. A 10-bit Port Control Register, PCC, is associated with Port C and allows each port pin to be programmed individually for one of these two functions. The port control register associated with Port B, PBC, contains only

one bit which programs all 15 pins. Also associated with each general purpose port is a data direction register which programs each pin as an input or output, and a data register for data I/O.

### **1.4.3 RSSI0 and RSSI1**

The DSP56166 provides two identical Reduced Synchronous Serial Interfaces (RSSI's). They are extremely flexible, full-duplex serial interfaces which allow the DSP56166 to communicate with a variety of serial devices. These interfaces include one or more industry standard codecs, other DSPs, microprocessors, and peripherals. The RSSIx interface consists of a transmitter and a receiver section and a transmit/receive RSSI clock generator. Each of the following characteristics of the RSSI can be independently defined: the number of bits per word, the protocol or mode, and the clock. The transmit and receive sections are synchronous. Three modes of operation are available: Normal, Network, and Gated. The Normal Mode is typically used to interface with devices on a regular or periodic basis. In this mode, the RSSI functions with one data word of I/O per frame. The Network Mode provides time slots in addition to a bit clock and a frame synchronization pulse. The RSSI functions with 2 to 8 words of I/O per frame in the Network Mode. This mode is typically used in star or ring Time Division Multiplex (TDM) networks with other DSP56166s and/or codecs. The clock can be programmed to be continuous or gated. The RSSI supports a subset of the Motorola SPI interface. The RSSI requires three to four pins depending on the operating mode selected.

### **1.4.4 Timer**

The Timer is a general purpose 16-bit timer/event counter with internal or external clocking which can be used to interrupt the DSP or to signal an external device at periodic intervals, after counting internal events or after counting external events. A Timer Input pin (TIN) can be used as an event counter input and a Timer Output pin (TOUT) can be used for timer pulse or timer clock generation.

The timer includes three 16-bit registers: the Timer Count Register (TCTR), the Timer Preload Register (TPR), and the Timer Compare Register (TCPR). An additional Timer Control Register (TCR) controls the timer operations.

A decrement register, programmed by the control register, is not available to the user. All other registers are memory mapped read/write registers.

### **1.4.5 Host Interface (HI)**

The HI is a byte-wide parallel slave port which may be connected directly to the data bus of a host processor. The host processor may be any of a number of popular microcomputers or microprocessors, another DSP, or DMA hardware. The HI is composed of an 8-bit bidirectional data bus and 7 control lines to control data transfers. The HI appears as

a memory mapped peripheral, occupying 8 bytes in the host processor's address space and three words in the DSP processor's address space. Separate transmit and receive data registers are double-buffered to allow the DSP56166 and host processor to efficiently transfer data at high speed. Host processor communication with the HI registers is accomplished using standard host processor instructions and addressing modes. Host processors may use byte move instructions to communicate with the HI registers. The host registers are addressed so that 8-bit MC6801-type host processors can use 16-bit load (LDD) and store (STD) instructions for data transfers. The 16-bit MC68000/10 host processor can address the HI using the special MOVEP instruction for word (16-bit) or long word (32-bit) transfers. The 32-bit MC68020 host processor can use its dynamic bus sizing feature to address the HI using standard MOVE word (16-bit), long word (32-bit) or quad word (64-bit) instructions.

One of the most innovative features of the HI is the Host Command feature. With this feature, the host processor can issue vectored exception requests to the DSP56166. The host may select any one of 63 DSP56166 exception routines to be executed by writing a Vector Address Register in the HI. This flexibility allows the host programmer to execute up to 63 functions preprogrammed in the DSP56166.

## **1.5 OnCE**

OnCE provides hardware/software emulation and debug on the DSP56166 and a means of interacting with the DSP56166 and any memory mapped peripherals non-intrusively so that a user may examine registers, memory, or peripherals. To achieve this, special circuits and dedicated pins on the DSP are used to avoid sacrificing any user accessible on-chip resource. A key feature of the special OnCE pins is to allow the user to insert the DSP56166 into his target system yet retain debug control, especially in the case of devices specified without an external bus. The need for a costly cable which brings out all processor pins on a traditional emulator system is eliminated.

## **1.6 PROGRAMMING MODEL**

The DSP56166 is based on the DSP5616 core. The programmer can view the DSP5616 core architecture as three execution units operating in parallel. The three execution units are the Data ALU, Address Generation Unit, and Program Control Unit. The programming model appears like that of a conventional MPU. The programming model is shown in Figure 1-8 and is described in the following paragraphs.

### **1.6.1 Data ALU**

The data ALU features four 16-bit input/output data registers which can be concatenated to handle 32-bit data, two 40-bit accumulators, automatic scaling, and saturation arithmetic.

#### 1.6.1.1 Data ALU Input Registers (X1, X0, Y1, Y0)

X1, X0, Y1, and Y0 are 16-bit latches which serve as input pipeline registers for the data ALU. Each register may be read or written by the XDB. X0, X1, and Y0 may be written over the GDB. They may be treated as four independent 16-bit registers or as two 32-bit registers called X and Y which are developed by the concatenation of X1:X0 and Y1:Y0 respectively. X1 is the most significant word in X and Y1 is the most significant word in Y.

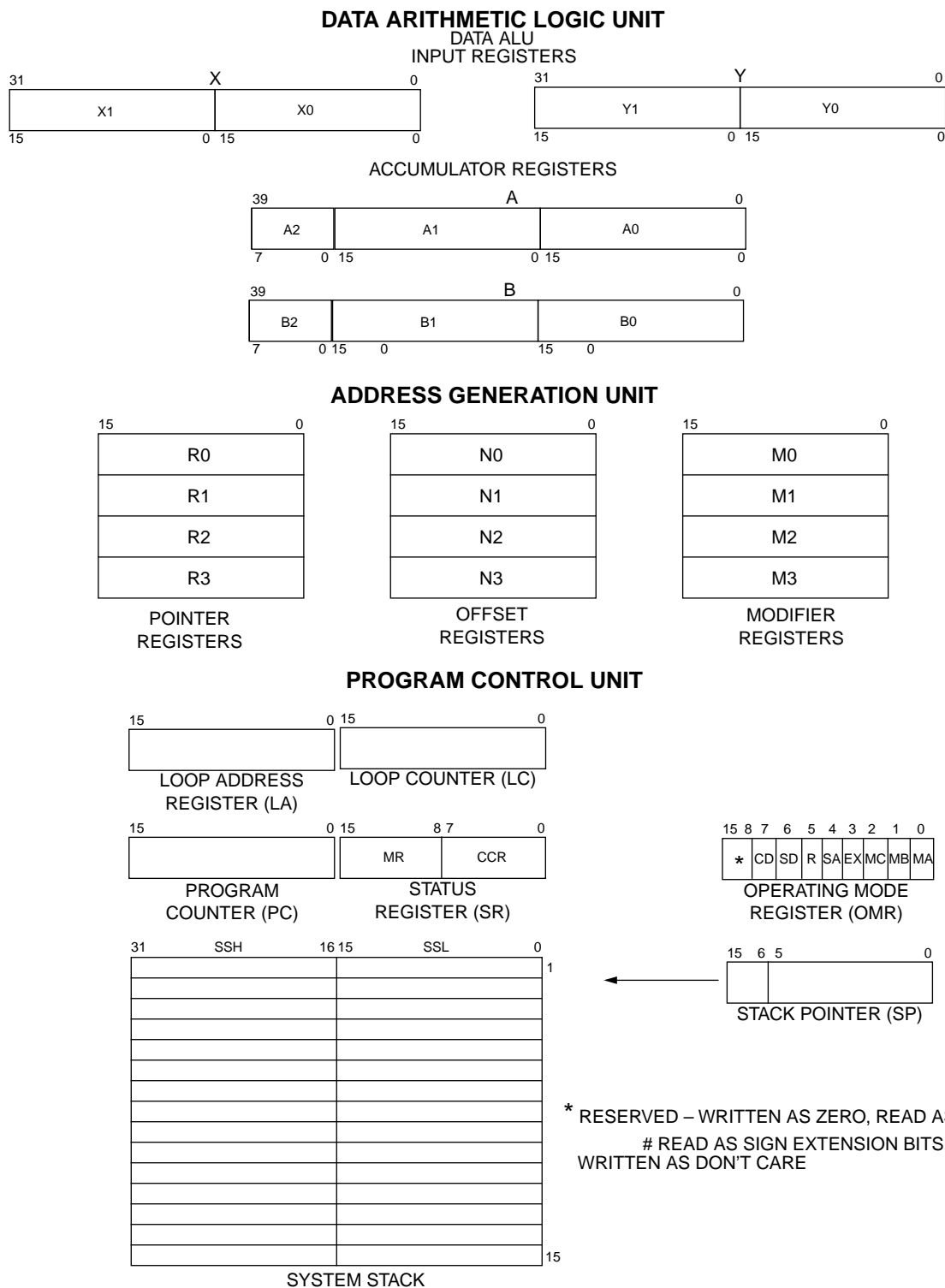
These Data ALU input registers are used as source operands for most data ALU operations and allow new operands to be loaded for the next instruction while the register contents are being used by the current instruction.

#### 1.6.1.2 Data ALU Accumulator Registers (A2, A1, A0, B2, B1, B0)

A1, A0, B1, and B0 are 16-bit latches that serve as data ALU accumulator registers. A2 and B2 are 8-bit latches that serve as accumulator extension registers. Each register may be read or written by the XDB as a word operand. A1 and B1 may be written by the GDB. When A2 or B2 is read, the register contents occupy the low-order portion (bits 7-0) of the word; the high-order portion (bits 16-8) is sign-extended. When A2 or B2 is written, the register receives the low-order portion of the word; the high-order portion is not used. Automatic sign extension of the 40-bit accumulators is provided when the A or B register is written with a smaller size operand. If the A or B register is written with a 16-bit value, then the least significant 16 bits are set to zero.

It is also possible to saturate the accumulator on a 32-bit value automatically after every accumulation. Overflow protection is performed after the contents of the accumulator have been shifted according to the scaling mode defined in the status register. When limiting occurs, the L bit flag in the status register is set and latched.

# PROGRAMMING MODEL



**Figure 1-8 Programing Model**

### **1.6.2 Address Generation Unit**

The programmer's model for the address generation unit consists of three banks of register files — pointer register files, offset register files, and modifier register files. These provide all the registers necessary to generate address register indirect effective addresses.

#### **1.6.2.1 Address Register File (R0-R3)**

The address register file consists of four, sixteen-bit registers. The file contains the address registers R0-R3 which usually contain addresses used as pointers to memory. Each register may be read or written by the Global Data Bus. Each address register may be used as an input to the modulo arithmetic unit for a register update calculation. Each register may be written by the GDB or by the output of the modulo arithmetic unit.

#### **1.6.2.2 Offset Register File (N0-N3)**

The offset register file consists of four, sixteen-bit registers. The file contains the offset registers N0-N3 and usually contains offset values used to update address pointers. Each offset register may be read or written by the Global Data Bus. Each offset register is used as an input to the modulo arithmetic unit when the same number address register is read and used as an input to the modulo arithmetic unit.

#### **1.6.2.3 Modifier Register File (M0-M3)**

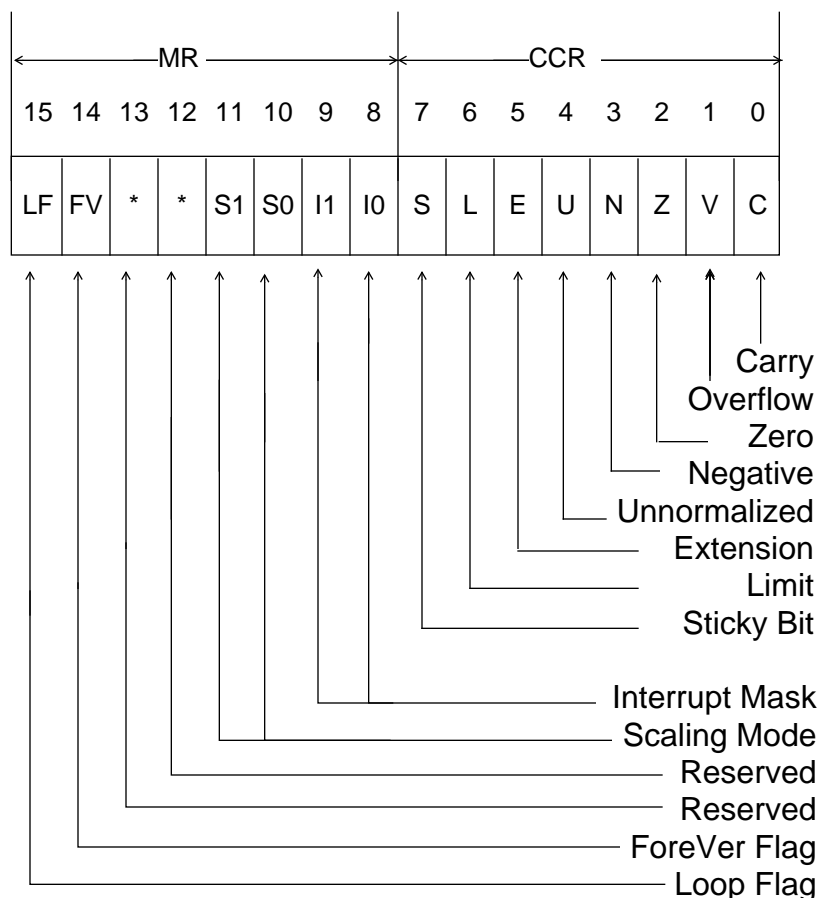
The modifier register file consists of four, 16-bit registers. The file contains the modifier registers M0-M3 and usually specifies the type of arithmetic used to modify an address register during address register update calculations — linear, modulo, or reverse carry. Each modifier register may be used as an input to the modulo arithmetic unit or written by the Global Data Bus. Each modifier register is read when the same number address register is read and used as an input to the modulo arithmetic unit. Each modifier register is preset to \$FFFF during a processor reset. Note that when R3 is used for the second read of a dual read operation, only linear arithmetic is available on this address register.

### **1.6.3 Program Control Unit**

The program control unit features loop address and loop counter registers which are dedicated to supporting the hardware DO loop instruction in addition to the standard program flow control resources such as a program counter, status register, and system stack. With the exception of the program counter, all registers are read/write to facilitate system debug.

#### **1.6.3.1 Program Counter (PC)**

This 16-bit register contains the address of the next location to be fetched from program memory space. The PC may point to instructions, data operands, or addresses of oper-



**Figure 1-9 Status Register Format**

ands. References to this register are always inherent and are implied by most instructions. This special purpose address register is stacked when program looping is initiated, when a branch or a jump to subroutine is performed, and when interrupts occur (except for fast interrupts).

### 1.6.3.2 Status Register (SR)

The SR is a 16-bit register consisting of an 8-bit Mode Register (MR) and an 8-bit Condition Code Register (CCR) — see Figure 1-9. The MR register is the high-order 8 bits of the SR; the CCR register is the low-order 8 bits.

The MR bits are only affected by processor reset, exception processing, the DO, ENDDO, RTI, and SWI instructions and by instructions which directly reference the MR register (e.g., MOVE, ANDI, ORI). During processor reset, the interrupt mask bits of the mode register will be set, the scaling mode bits, loop flag, and the forever flag will be cleared. The



CCR is a special purpose control register which defines the current user state of the processor at any given time. The CCR bits are affected by data ALU operations, one address ALU operation (CHKAAU), bit field manipulation instructions, parallel move operations, and by instructions which directly reference the CCR register. The CCR bits are not affected by data transfers over XDB except when data limiting occurs while reading the A or B accumulators or by the conditions described for the Sticky Bit, Bit 7. During processor reset, all CCR bits are cleared. The SR register is stacked when program looping is initialized, when a jump or branch to subroutine (JScc, BSc, JSR, BSR) is performed, and when long interrupts occur.

#### **1.6.3.3 Loop Counter (LC)**

The LC is a special 16-bit counter used to specify the number of times to repeat a hardware program loop. This register is stacked by a DO instruction and unstacked by end of loop processing or by execution of a BRKcc or an ENDDO instruction. When the end of a hardware program loop is reached, the contents of the loop counter register are tested for one. If the LC is one, the program loop is terminated and the LC register is loaded with the previous LC contents stored on the stack. If the LC is not one, it is decremented by one and the program loop is repeated. The LC may be read under program control. This allows the number of times a loop has been executed to be determined during execution. Note that if LC=0 during execution of the DO instruction, the loop will not be executed and the program will continue with the instruction immediately after the loop end of expression. LC is also used by the REP instruction.

#### **1.6.3.4 Loop Address Register (LA)**

The LA indicates the location of the last instruction word in a program DO loop. This register is stacked by a DO instruction and unstacked by end of loop processing or by execution of an ENDDO or BRKcc instruction. When the instruction word at the address contained in this register is fetched, the content of LC is checked. If it is not one, the LC is decremented and the next instruction is taken from the address at the top of the system stack. Otherwise; the PC is incremented, the loop flag is restored (pulled from stack), the stack pointer is decremented by two, the LA and LC registers are pulled from the stack and restored, and instruction execution continues normally. The LA register is a read/write register written into by a DO instruction.

#### **1.6.3.5 System Stack (SS)**

The SS is a separate internal RAM, 15 locations “deep”, and divided into two 16-bit wide banks: High (SSH) and Low (SSL). SSH stores the PC and LA contents; SSL stores the LC and SR contents. The PC and SR registers are pushed on the stack for subroutine calls and long interrupts. These registers are pulled from the stack for subroutine returns

using the RTS instruction (PC only) and for interrupt returns that use the RTI instruction. The system stack is also used for storing the address of the beginning instruction of a hardware program loop as well as the SR, LA, and LC register contents just prior to the start of the loop. This allows nesting of DO loops.

Up to 15 long interrupts, 7 DO loops, or 15 JSRs or combinations of these can be accommodated by the stack. Care must be taken when approaching the stack limit. When the Stack limit is exceeded, the data to be stacked will be lost and a non-maskable Stack Error interrupt will occur. The stack error interrupt occurs after the stack limits have been exceeded.

#### 1.6.3.6 Stack Pointer (SP)

The SP is a 6-bit register that indicates the location of the top of the SS and the stack status (underflow, empty, full, and overflow conditions – see Table 1-8). The SP is referenced implicitly by some instructions (DO, REP, JSR, RTI, etc.) or directly by the MOVEC instruction. Note that the stack pointer register is implemented as a 6-bit counter which addresses (selects) a fifteen location stack with its four least significant bits.

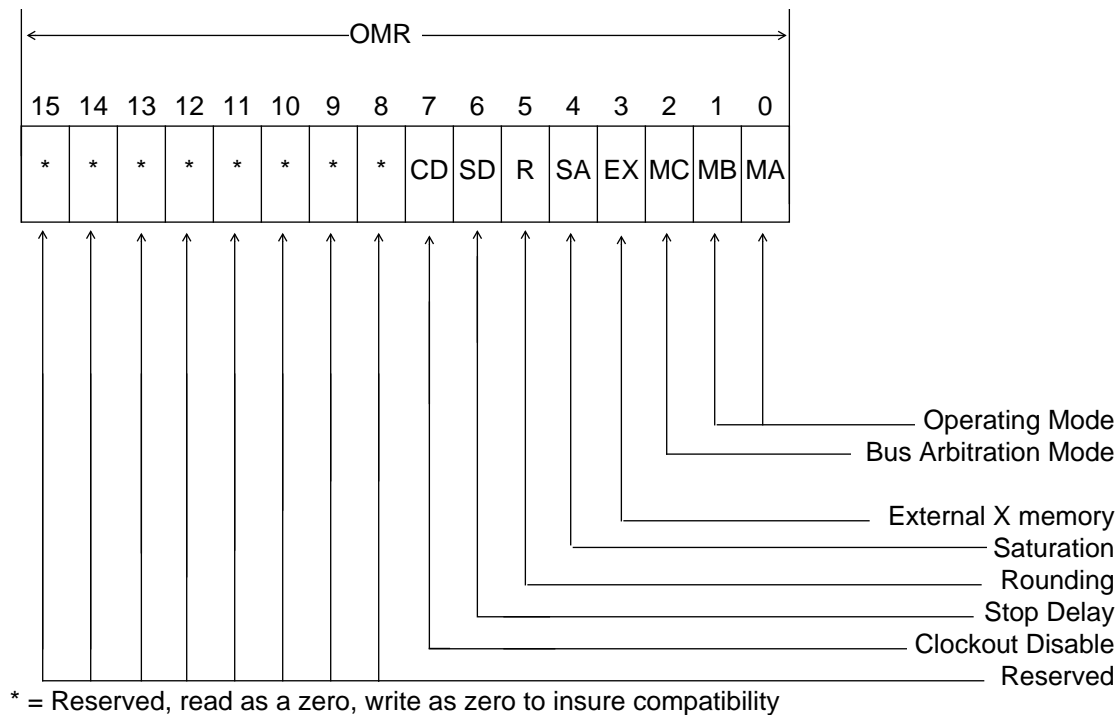
**Table 1-8 Stack Pointer Values**

UF	SE	P3	P2	P1	P0	CAUSE
1	1	1	1	1	0	← Stack Underflow condition after double pull.
1	1	1	1	1	1	← Stack Underflow condition.
0	0	0	0	0	0	← Stack Empty (reset). Pull causes underflow.
0	0	0	0	0	1	← stack location 1.
...						
...						
0	0	1	1	1	0	← Stack location 14.
0	0	1	1	1	1	← Stack location 15 (stack full). Push causes overflow.
0	1	0	0	0	0	← Stack Overflow condition.
0	1	0	0	0	1	← Stack Overflow condition after double push.

#### 1.6.3.7 Operating Mode Register (OMR)

The OMR is a 16-bit register which defines the current processor operating mode. The OMR bits are only affected by processor reset and by instructions which directly reference the OMR.

The operating mode register format is shown in Figure 1-10.



**Figure 1-10 Operating Mode Register (OMR)**

During processor reset, the chip operating mode bits will be loaded from the external Mode Select pins. Table 1-9 summarizes the operating modes for the 56166 RAM based part and Table 1-10 does the same for the ROM based part.

**Table 1-9 Operating Mode Summary — DSP56166 RAM Based Part**

Operating Mode	M B	M A	Description
Special Bootstrap 1	0	0	Bootstrap from an external byte-wide memory located at P:\$C000.
Special Bootstrap 2	0	1	Bootstrap from the host port or RSSI0
Normal Expanded	1	0	Internal PRAM enabled; external reset at P:\$E000
Development Expanded	1	1	Internal program RAM disabled; external reset at P:\$0000.

**Table 1-10 Operating Mode Summary — DSP56166 ROM Based Part**

MB	MA	Chip Operating Mode	Reset Vector	Program Memory Configuration
0	0	Single Chip	Internal Pmem P:\$0	Internal Pmem Enabled
0	1	Single Chip	Internal Pmem P:\$100	Internal Pmem Enabled
1	0	Normal Expanded	External Pmem P:\$E000	Internal Pmem Enabled
1	1	Development	External Pmem P:\$0	Internal Pmem Disabled

\*: the address will be the first location of the internal PROM.

The Mode C bit (MC) selects the initial bus operating mode. When set, the external bus is programmed in the master mode ( $\overline{BR}$  output and  $\overline{BG}$  input) and when cleared, the bus is programmed in the slave mode ( $\overline{BR}$  input and  $\overline{BG}$  output).

The EX bit, when set, will force all accesses to the X data memory to be external accesses, preventing access to on-chip data memory and on-chip peripheral space. This allows a continuous memory map when using 64K of external data memory. When the EX bit is set, the memory mapped I/O registers, both on and off-chip, can only be accessed during fast interrupt. During normal operation and during a long interrupt, the EX bit must be explicitly cleared by the user's code before the on-chip data memory and the memory mapped I/O registers can be read or written. The on-chip peripherals, however, will still be active and interrupts will still be recognized. Special care should be taken with this bit in a nested interrupt environment since it is not in the Status Register (SR) and thus is not stacked with each new interrupt. Note that OnCE cannot read the content of on-chip X memories, registers and peripherals when the EX bit is set, but it can still write to them.

The Saturation bit (SA), when set, selects automatic saturation on 32 bits for the results from the MAC array back to the accumulator. This saturation is done by a special saturation circuit inside the MAC unit. The purpose of this bit is to provide a saturation mode for 16-bit algorithms which do not recognize or cannot take advantage of the extension accumulator. The saturation logic operates by checking three bits of the 40-bit result: two bits of the extension byte (exp[7] and exp[0]) and one bit on the MSP (msp[15]). The result obtained in the accumulator when SA = 1 is shown in Table 1-11.

The Rounding bit (R) selects between convergent rounding and two's complement rounding. When set, two's complement rounding (always round up) is used.

The Stop Delay bit (SD) is used to select the delay that the DSP needs to exit the STOP mode.

**Table 1-11 Actions of the Saturation Mode (SA=1)**

exp[7]	exp[0]	msp[15]	result in accumulator
0	0	0	unchanged
0	0	1	\$00 7FFF FFFF
0	1	0	\$00 7FFF FFFF
0	1	1	\$00 7FFF FFFF
1	0	0	\$FF 8000 0000
1	0	1	\$FF 8000 0000
1	1	0	\$FF 8000 0000
1	1	1	unchanged

When the Clock out Disable bit (CD) is cleared in the OMR, the CLKO pin provides a clock to external circuitry. Setting the CD bit will disable this signal one instruction cycle after the bit has been set.

**Note:** When a bit of the OMR is changed by an instruction, a delay of one instruction cycle is necessary before the new mode comes into effect.

## 1.7 INSTRUCTION SET SUMMARY

As indicated by the programming model, the DSP architecture can be viewed as three functional units operating in parallel (Data ALU, AGU, and PCU). The goal of the instruction set is to keep each of these units busy each instruction cycle. This achieves maximum throughput and minimum use of program memory.

This section introduces the DSP instruction set and instruction format. The complete range of instruction capabilities combined with the flexible addressing modes provide a very powerful assembly language for digital signal processing algorithms. The instruction set has also been designed to allow efficient coding for future high-level DSP language compilers. Execution time is enhanced by the hardware looping capabilities.

### 1.7.1 Instruction Groups

The instruction set is divided into the following groups:

- Arithmetic
- Logical
- Bit Field Manipulation
- Loop
- Move
- Program Control

Each instruction group is described in the following sections. Detailed information on each instruction is given in Appendix A of the DSP56100 Family User's Manual.

### 1.7.1.1 Arithmetic Instructions

The arithmetic instructions perform all of the arithmetic operations within the Data ALU. They may affect all of the condition code register bits. Arithmetic instructions are register-based (register direct addressing modes used for operands) so that the Data ALU operation indicated by the instruction does not use the XDB or the GDB. Optional data transfers may be specified with most arithmetic instructions. This allows for parallel data movement over the XDB and over the GDB during a Data ALU operation. This allows new data to be prefetched for use in subsequent instructions and allows results calculated by previous instructions to be stored. These instructions execute in one instruction cycle. The following are the arithmetic instructions.

ABS	Absolute Value
ADC	Add Long with Carry*
ADD	Add †
ASL	Arithmetic Shift Left
ASL4	4-Bit Arithmetic Shift Left*
ASR	Arithmetic Shift Right
ASR4	4-Bit Arithmetic Shift Right*
ASR16	16-Bit Arithmetic Shift Right*
CHKAU	Update the V, Z, N flags according to the address calculation result*
CLR	Clear an Accumulator
CLR24	Clear 24 MSBs of an Accumulator
CMP	Compare
CMPM	Compare Magnitude
DEC	Decrement Accumulator
DEC24	Decrement upper 24 bits of Accumulator
DIV	Divide Iteration*
DMAC	Double (Multi) precision oriented MAC*
EXT	Sign Extend Accumulator from bit 31*
IMAC	Integer Multiply-Accumulate*
IMPY	Integer Multiply*
INC	Increment Accumulator
INC24	Increment 24 MSBs of Accumulator
MAC	Signed Multiply-Accumulate †
MACR	Signed Multiply-Accumulate and Round †
MPY	Signed Multiply †
MPYR	Signed Multiply and Round †
MPY(su,uu)	Mixed Mode Multiply*

MAC(su,uu)	Mixed Mode Multiply-Accumulate*
NEG	Negate
NEGC	Negate with Borrow*
NORM	Normalize*
RND	Round
SBC	Subtract Long with Carry
SUB	Subtract †
SUBL	Shift Left and Subtract
SWAP	Swap MSP and LSP of an Accumulator*
Tcc	Transfer Conditionally*
TFR	Transfer Data ALU Register (Accumulator as destination) †
TFR2	Transfer Accumulator (32-bit Data Alu register as destination)*
TST	Test an accumulator
TST2	Test an ALU data register*
ZERO	Zero Extend Accumulator from bit 31*

\*These instructions do not allow parallel data moves.

† These instructions allow a dual read parallel move.

### 1.7.1.2 Logical Instructions

The logical instructions perform all of the logical operations within the Data ALU. They may affect all of the condition code register bits. Logical instructions are register-based as are the arithmetic instructions above. Optional data transfers may be specified with most logical instructions. With the exceptions of ANDI and ORI instructions, the destination of all logical instructions is A1 or B1. These instructions execute in one instruction cycle. The following are the logical instructions.

AND	Logical AND
ANDI	AND Immediate Program Controller Register*
EOR	Logical Exclusive OR
LSL	Logical Shift Left
LSR	Logical Shift Right
NOT	Logical Complement
OR	Logical Inclusive OR
ORI	OR Immediate Program Controller Register*
ROL	Rotate Left
ROR	Rotate Right

\*These instructions do not allow parallel data moves.

### 1.7.1.3 Bit Field Manipulation Instructions

This group tests the state of any set of bits within a byte in a memory location or a register and then sets, clears, or inverts bits in this byte. Bit fields which can be tested include the upper byte and the lower byte in a 16-bit value. The carry bit of the condition code register will contain the result of the bit test for each instruction. These instructions are read-modify-write and require two instruction cycles. Parallel data moves are not allowed with any of these instructions. The following are the bit field manipulation instructions.

BFTSTL	Bit Field Test Low
BFTSTH	Bit Field Test High
BFCLR	Bit Field Test and Clear
BFSET	Bit Field Test and Set
BFCHG	Bit Field Test and Change

### 1.7.1.4 Loop Instructions

The loop instructions control hardware looping by initiating a program loop and setting up looping parameters, or by “cleaning” up the system stack when terminating a loop. Initialization includes saving registers used by a program loop (LA and LC) on the system stack so that program loops can be nested. The address of the first instruction in a program loop is also saved to allow no-overhead looping. The end address of the DO loop is specified as PC relative. Parallel data moves are not allowed with any of these instructions. The following are the loop instructions.

DO	Start Hardware Loop
DO FOREVER	Hardware Loop Forever
ENDDO	Disable Current Loop and Unstack Parameters
BRKcc	Conditional Exit from Hardware Loop

### 1.7.1.5 Move Instructions

The move instructions perform data movement over the XDB and over the GDB. Move instructions do not affect the condition code register except for the limit bit, L, if limiting is performed or the Sticky Bit, S, when reading a Data ALU accumulator register. AGU instructions are also included among the following move instructions. These instructions do not allow optional data transfers.

LEA	Load Effective Address
MOVE	Move Data with or without Register Transfer – TFR(3)
MOVE(C)	Move Control Register
MOVE(I)	Move Immediate Short
MOVE(M)	Move Program Memory



MOVE(P)	Move Peripheral Data
MOVE(S)	Move Absolute Short

### 1.7.1.6 Program Control Instructions

The program control instructions include branches, jumps, conditional branches, jumps and other instructions which affect the PC and system stack. Program control instructions may affect the condition code register bits as specified in the instruction. Parallel data moves are not allowed with any of these instructions. The following are the program control instructions.

Bcc	Branch Conditionally (PC relative)
BSR	Branch to Subroutine (PC relative)
BRA	Branch (PC relative)
BScc	Branch to Subroutine Conditionally (PC relative)
DEBUG	Enter Debug Mode
DEBUGcc	Enter Debug Mode Conditionally
Jcc	Jump Conditionally
JMP	Jump
JSR	Jump to Subroutine
JScc	Jump to Subroutine Conditionally
NOP	No Operation
REP	Repeat Next Instruction
REPcc	Repeat Next Instruction Conditionally
RESET	Reset Peripheral Devices
RTI	Return from Interrupt
RTS	Return from Subroutine
STOP	Stop Processing (low power stand-by)
SWI	Software Interrupt
WAIT	Wait for Interrupt (low power stand-by)

### 1.7.2 Instruction Formats

Instructions are one or two words in length. The instruction and its length are specified by the first word of the instruction. The next word may contain information about the instruction itself or about an operand for the instruction. The assembly language source code for a typical one word instruction is shown below. The source code is organized into four columns.

Opcode	Operands	X Bus Data	G Bus Data
MAC	X0,Y0,A	X:(R0)+,X0	X:(R3)+,Y0

The Opcode column indicates the Data ALU, AGU, or PCU operation to be performed. The Operands column specifies the operands to be used by the opcode. The X Bus Data and G Bus Data columns specify optional data transfers over the X Bus, the G bus, and the addressing modes to be used. The Opcode column must always be included in the source code.

The DSP offers parallel processing using the Data ALU, AGU, and PCU. For the instruction word above, the DSP will perform the designated ALU operation (Data ALU), up to two data transfers specified with address register updates (AGU), and will also decode the next instruction and fetch an instruction from program memory (PCU), all in one instruction cycle. When an instruction is more than one word in length, an additional instruction execution cycle is required. Most instructions involving the Data ALU are register-based (all operands are in Data ALU registers) and allow the programmer to keep each parallel processing unit busy. An instruction which is memory-oriented (such as a bit field manipulation instruction) or that causes a control flow change (such as a branch/jump) prevents the use of parallel processing resources during its execution. See the DSP56100 Family User's Manual for additional information.

## 1.7.3 Addressing Modes

The addressing modes are grouped into three categories — register direct, address register indirect, and special. These addressing modes are summarized in Table 1-12. All address calculations are performed in the address generation unit to minimize execution time. Addressing modes specify whether the operand(s) is(are) in a register, memory, or encoded in the instruction as immediate data.

The register direct addressing mode can be subclassified according to the specific register addressed. The data registers include X1, X0, Y1, Y0, X, Y, A2, A1, A0, B2, B1, B0, A, and B. The control registers include SR, OMR, SP, SSH, SSL, LA, LC, CCR, and MR.

Address register indirect modes use an address register, Rn, to point to locations in memory. The content of Rn is the effective address (ea) except in the indexed by offset mode where the ea is Rn+Nn, or in the indexed by short displacement where the ea is Rn+a short immediate constant. Address register indirect modes use a modifier register, Mn, to specify the type of arithmetic to be used to update Rn. If a mode using an offset is specified, an offset register, Nn, is also used for the update. The Nn and Mn registers are assigned to the Rn with the same n. Thus, the assigned register sets are R0;N0;M0, R1;N1;M1, R2;N2;M2, and R3;N3;M3.

**Table 1-12 DSP56166 Addressing Modes**

Addressing Mode	Uses Mn Modifier	Operand Reference						
		S	C	D	A	P	X	XX
Register Direct								
Data or Control Register	No	X	X	X				
Address Register Rn	No				X			
Address Modifier Register Mn	No				X			
Address Offset Register Nn	No				X			
Address Register Indirect								
No Update	No					X	X	
Postincrement by 1	Yes*					X	X	X
Postdecrement by 1	Yes					X	X	
Postincrement by Offset Nn	Yes*					X	X	X
Indexed by Offset Nn	Yes						X	
Predecrement by 1	Yes						X	
PC Relative								
Long Displacement	No		X			X		
Short Displacement	No		X			X		
Address Register	No		X		X	X		
Special								
Upper word of accumulator	No						X	
Immediate Data	No					X		
Immediate Short Data	No					X		
Absolute Address	No					X	X	
Absolute Short Address	No					X	X	
Short Jump Address	No					X		
I/O Short Address	No						X	
Implicit	No	X	X			X		
Indexed by short displacement	No						X	
Where:								
S = System Stack Reference								
P = Program Memory Reference								
C =Program Controller Register Reference								
X = X Memory Reference								
D = Data ALU Register Reference								
XX = Double X Memory Read								
A = Address ALU Register Reference								
*Note: M3 is not used for updating R3 in the second read in the X memory								

This structure is unique and extremely powerful in general, and particularly powerful in setting up DSP oriented data structures. Two sets of address registers can be used by the

instruction set: one set for the first memory operation and one set for a second memory operation. Note that M3 is not used for updating R3 in the second read in the X memory.

The special addressing modes include immediate and absolute modes as well as implied references to the PC, system stack, and program memory. In addition, it is possible to use the upper word (MSP) of an accumulator as an address.

#### 1.7.4 Address Arithmetic

The DSP56166 Address Generation Unit supports linear, modulo, and bit-reversed address arithmetic for all address register indirect modes. The address modifiers, Mn, determine the type of arithmetic used to update addresses. Address modifiers allow the creation of data structures in memory for FIFOs (queues), delay lines, circular buffers, stacks, and bit-reversed FFT buffers. Data is manipulated by updating address registers (pointers) rather than moving large blocks of data. The contents of the address modifier register, Mn, defines the type of address arithmetic to be performed for addressing mode calculations, and for the case of modulo arithmetic, the content of Mn also specifies the modulus. Each address register Rn has its own modifier register Mn associated with it.

##### 1.7.4.1 Linear Modifier

Address modification is performed using normal 16-bit (modulo 65,536) two's complement linear arithmetic. A 16-bit offset Nn, or immediate data (+1, -1, or a displacement value) may be used in the address calculations. The range of values may be considered as signed (Nn from -32,768 to +32,767) or unsigned (Nn from 0 to +65,536).

##### 1.7.4.2 Reverse Carry Modifier

The address modification is performed by propagating the carry in the reverse direction, i.e., from the MSB to the LSB. If the (Rn)+Nn addressing mode is used with this address modifier, and Nn contains the value  $2^{K-1}$  (a power of two), then postincrementing by Nn is equivalent to bit-reversing the K LSBs of Rn, incrementing Rn by 1, and bit-reversing the K LSBs of Rn again. This address modification is useful for  $2^K$  point FFT addressing. The range of values for Nn is 0 to +32,767 which allows bit-reversed addressing for FFTs up to 65,536 points.

As an example, consider a 1024 point FFT with real and imaginary data stored in memory. Then Nn would contain the value 512 and postincrementing by +N would generate the address sequence 0, 512, 256, 768, 128, 640, ... This is the scrambled FFT data order for sequential frequency points from 0 to  $2\pi$ . For proper operation, the reverse carry modifier restricts the base address of the bit reversed data buffer to an integer multiple of  $2^K$ , such as 1024, 2048, 3072, etc. The use of addressing modes other than postincrement by Nn is possible but may not provide a useful result.

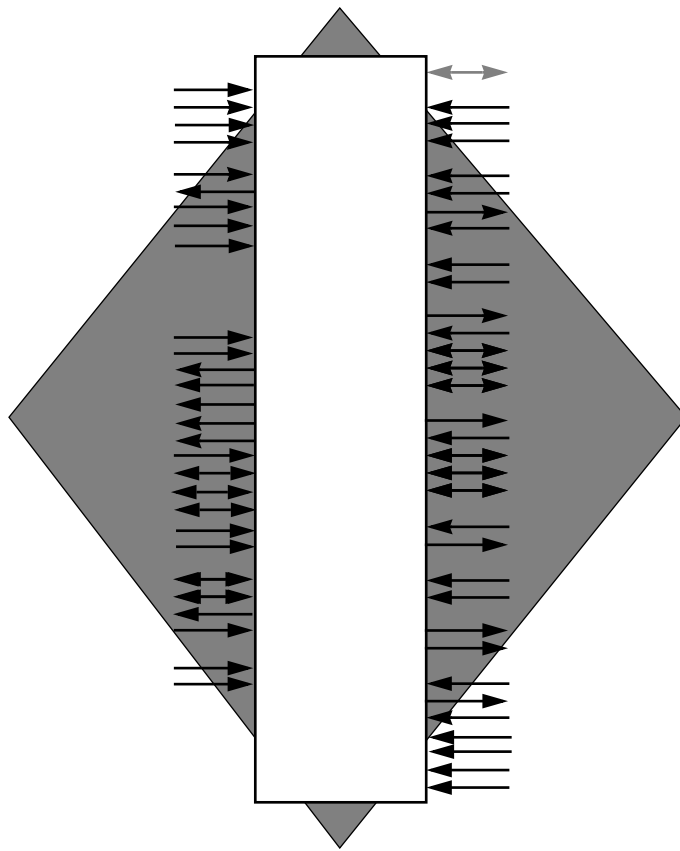
### 1.7.4.3 Modulo Modifier

The modulo arithmetic unit can update one address register,  $R_n$ , during one instruction cycle and is capable of performing linear, reverse carry, and modulo arithmetic. The contents of the selected modifier register specifies the type of arithmetic required in an address register update calculation. There is no modulo capability for an  $R3$  update for dual reads.

---

## SECTION 2

# DSP56166 PIN DESCRIPTIONS



## SECTION CONTENTS

---

2.1	INTRODUCTION .....	2-3
2.2	ADDRESS AND DATA BUS (32 PINS) .....	2-3
2.3	BUS CONTROL (10 PINS) .....	2-3
2.4	INTERRUPT AND MODE CONTROL (4 PINS) .....	2-9
2.5	POWER, GROUND, AND CLOCK (30 PINS) .....	2-10
2.6	HOST INTERFACE (15 PINS) .....	2-11
2.7	16-BIT TIMER (2 PINS) .....	2-12
2.8	REDUCED SYNCHRONOUS SERIAL INTERFACES (RSSI0 AND RSSI1) AND PORT C (8 PINS) .....	2-13
2.9	ON-CHIP EMULATION (4 PINS) .....	2-14
2.10	ON-CHIP CODEC (7 PINS) .....	2-15

## 2.1 INTRODUCTION

The DSP56166 pinout is shown in Figure 2-1. The input and output signals on the chip are organized into the 13 functional groups shown in Table 2-1. Figure 2-2 illustrates the relative timing for the bus signals. See the timing descriptions in the technical data sheet for more information.

**Table 2-1 Functional Group Pin Allocations**

Functional Group	Number of Pins
Address and Data Buses	32
Bus Control	10
Interrupt and Mode Control	4
Clock and PLL	3
Host Interface or PIO	15
Timer Interface or PIO	2
SSI Interfaces or PIO	8
On-chip CODEC	7
On-chip emulation (OnCE)	4
Power (Vdd)	10
Ground (Vss)	15
APower (Vdda)	1
AGround (Vssa)	1
Total	112

## 2.2 ADDRESS AND DATA BUS (32 PINS)

**A0-A15 (Address Bus) — three state, active high outputs.** A0-A15 change in  $t_0$ , and specify the address for external program and data memory accesses. If there is no external bus activity, A0-A15 remain at their previous values. A0-A15 are three-stated during hardware reset.

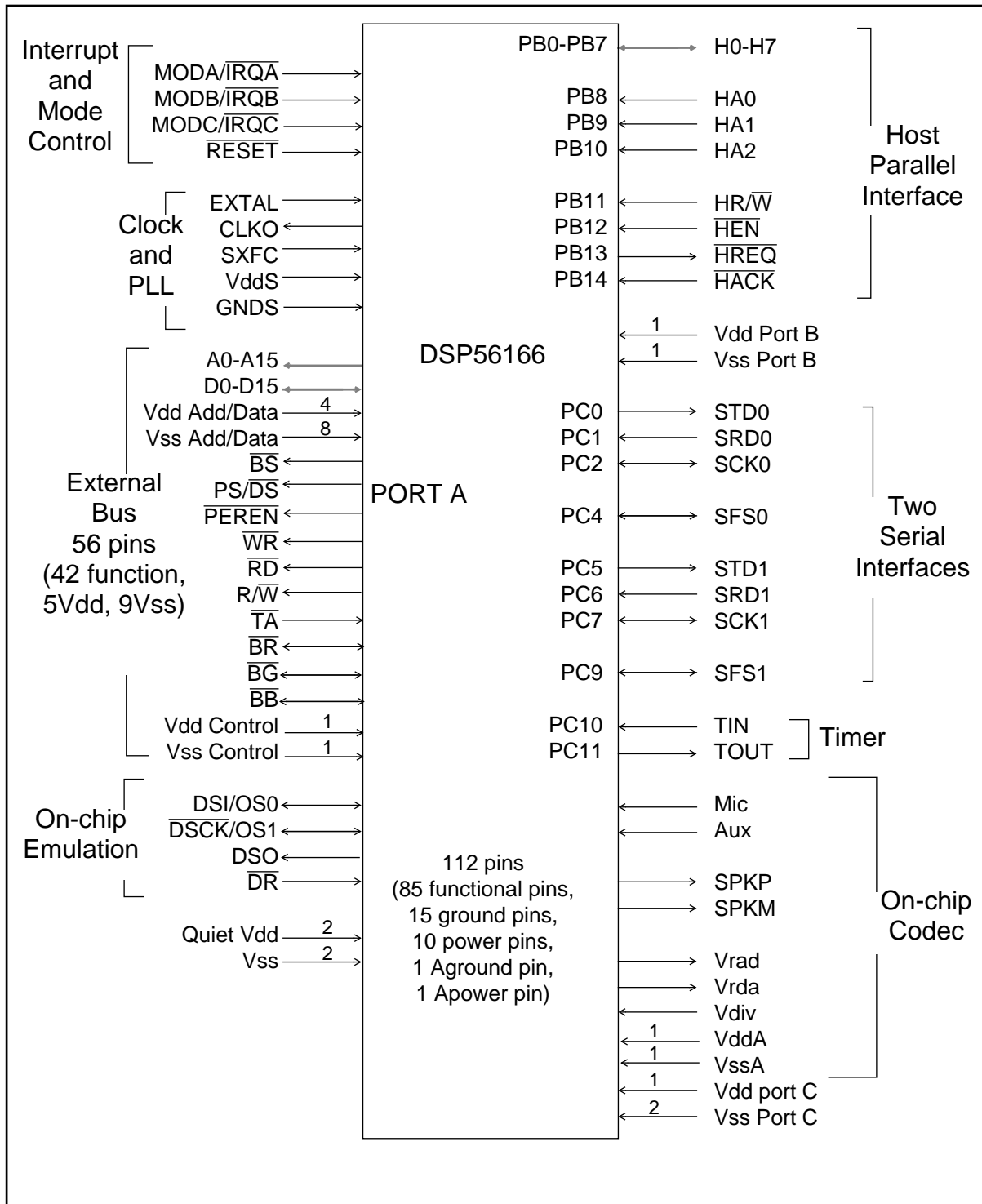
**D0-D15 (Data Bus) — three state, active high, bidirectional input/outputs.** Read data is sampled in by the trailing edge of  $t_2$ , while write data output is enabled by the leading edge of  $t_2$  and three-stated at the leading edge of  $t_0$ . If there is no external bus activity, D0-D15 are three-stated. D0-D15 are also three-stated during hardware reset.

## 2.3 BUS CONTROL (10 PINS)

**PS/ $\overline{DS}$  (Program/Data Memory Select) — three state active low output.** This output is asserted only when external data memory is referenced. PS/ $\overline{DS}$  timing is the same as the A0-A15 address lines. PS/ $\overline{DS}$  is high for program memory access and is low for data memory access. If the external bus is not used during an instruction cycle ( $t_0$ ,  $t_1$ ,  $t_2$ ,  $t_3$ ), PS/ $\overline{DS}$  goes high in  $t_0$ . PS/ $\overline{DS}$  is in the high impedance state during hardware reset.



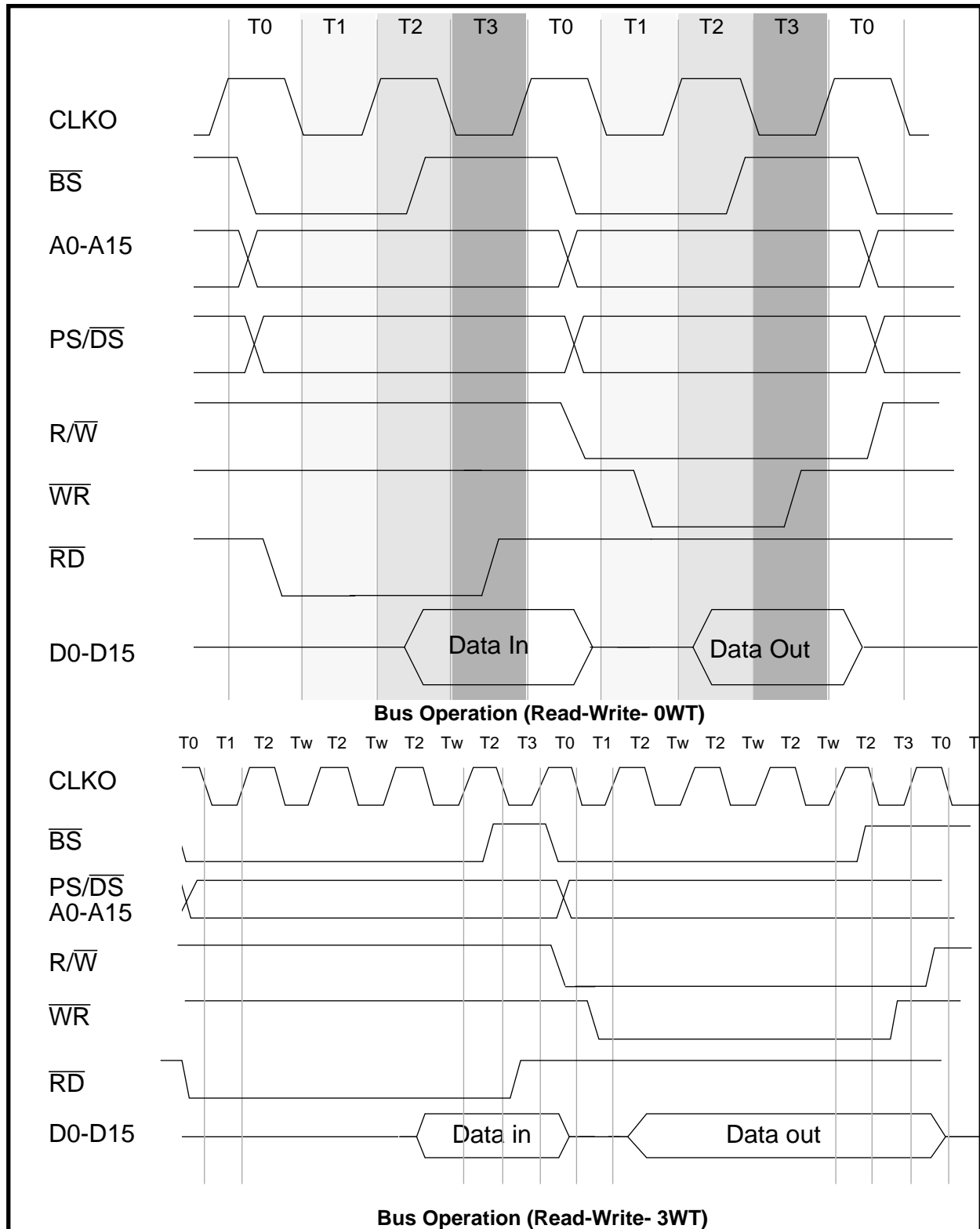
## BUS CONTROL (10 PINS)



**Figure 2-1. DSP56166 Pinout**

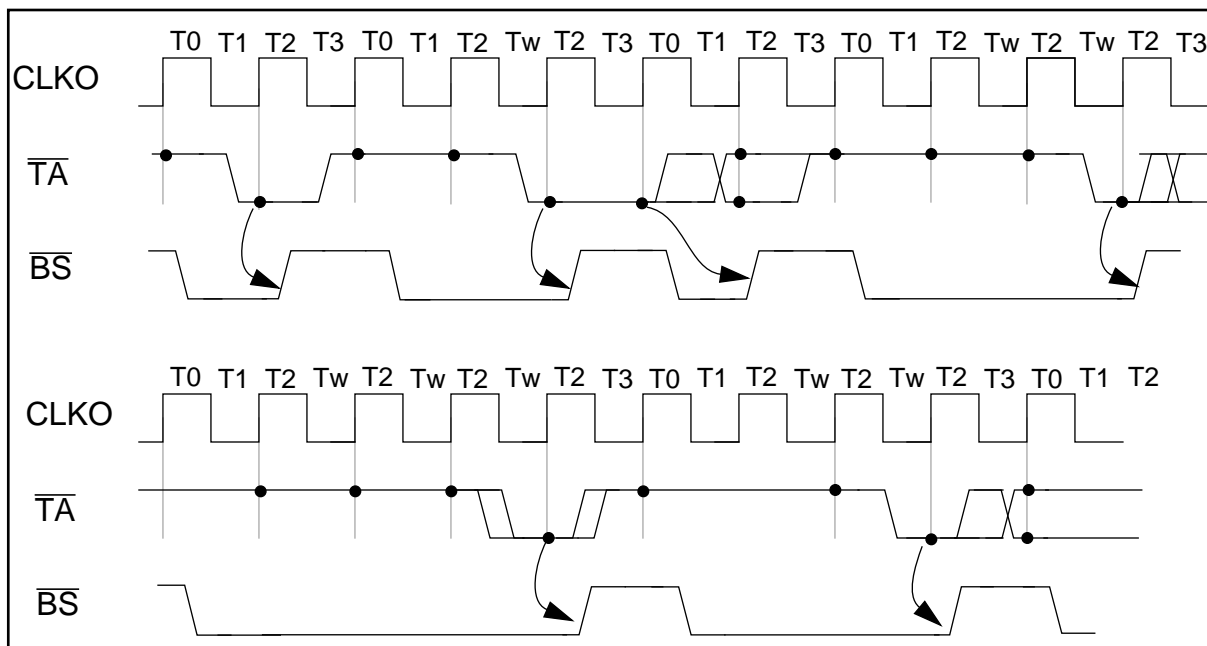
- $\overline{\text{PEREN}}$  (Peripheral Enable) — three state active low output.** This output is asserted only when external peripheral space of the data memory is referenced (any address between X:\$FF00 and X:\$FF7F).  $\overline{\text{PEREN}}$  timing is the same as the A0-A15 address lines; it is asserted and deasserted during t0.  $\overline{\text{PEREN}}$  is high for any program memory access and for data memory access not in the space X:\$FF00 - X:\$FF7F.  $\overline{\text{PEREN}}$  is in the high impedance state during hardware reset.
- $\text{R}/\overline{\text{W}}$  (Read/Write) — three state, active low output.** Timing is the same as the address lines, providing an “early write” signal.  $\text{R}/\overline{\text{W}}$  (which changes to t0) is high for a read access and is low for a write access. If the external bus is not used during an instruction cycle (t0, t1, t2, and t3),  $\text{R}/\overline{\text{W}}$  goes high in t0.  $\text{R}/\overline{\text{W}}$  is three-stated during hardware reset.
- $\overline{\text{WR}}$  (Write Enable) — three state, active low output.** This output is asserted during external memory write cycles. When  $\overline{\text{WR}}$  is asserted low in t1, the data bus pins D0-D15 become outputs and the DSP puts data on the bus during the leading edge of t2. When  $\overline{\text{WR}}$  is deasserted high in t3, the external data has been latched inside the external device. When  $\overline{\text{WR}}$  is asserted, it qualifies the A0-A15, PS/ $\overline{\text{DS}}$  pins.  $\overline{\text{WR}}$  is three-stated during hardware reset or when the DSP is not bus master.  $\overline{\text{WR}}$  can be connected directly to the  $\overline{\text{WE}}$  pin of a static RAM.
- $\overline{\text{RD}}$  (Read Enable) — three state, active low output.** This output is asserted during external memory read cycles. When  $\overline{\text{RD}}$  is asserted low late t0/early t1, the data bus pins D0-D15 become inputs and an external device is enabled onto the data bus. When  $\overline{\text{RD}}$  is deasserted high in t3, the external data has been latched inside the DSP. When  $\overline{\text{RD}}$  is asserted, it qualifies the A0-A15 and PS/ $\overline{\text{DS}}$  pins.  $\overline{\text{RD}}$  is three-stated during hardware reset or when the DSP is not bus master.  $\overline{\text{RD}}$  can be connected directly to the  $\overline{\text{OE}}$  pin of a static RAM or ROM.
- $\overline{\text{BS}}$  (Bus Strobe) — active low output.** This output is asserted low at the start of a bus cycle (during t0) and deasserted high at the end of the bus cycle (during t2). This pin provides an “early bus start” signal which can be used as an address latch and as an “early bus end” signal which can be used by an external bus controller.  $\overline{\text{BS}}$  is three-stated during hardware reset.

## BUS CONTROL (10 PINS)



**Figure 2-2 Bus Operations**

**$\overline{\text{TA}}$  (Transfer Acknowledge) — active low input.** If there is no external bus activity, the  $\overline{\text{TA}}$  input is ignored by the DSP. When there is external bus cycle activity,  $\overline{\text{TA}}$  can be used to insert wait states in the external bus cycle.  $\overline{\text{TA}}$  is sampled on the leading edge of the clock. Any number of wait states from 1 to infinity may be inserted by using  $\overline{\text{TA}}$ . If  $\overline{\text{TA}}$  is sampled high on the leading edge of the clock beginning the bus cycle, the bus cycle will end 2T after  $\overline{\text{TA}}$  has been sampled low on a leading edge of the clock; if the Bus Control Register (BCR) value does not program more wait states. The number of wait states is determined by the  $\overline{\text{TA}}$  input or by the BCR, whichever is longer.  $\overline{\text{TA}}$  is still sampled during the leading edge of the clock when wait states are controlled by the BCR value. In that case,  $\overline{\text{TA}}$  will have to be sampled low during the leading edge of the last period of the bus cycle programmed by the BCR (2T before the end of the bus cycle programmed by the BCR) in order not to add any wait states.  $\overline{\text{TA}}$  should always be deasserted high during t3 to be sampled high by the leading edge of t0. If  $\overline{\text{TA}}$  is sampled low (asserted) at the leading edge of the t0 beginning the bus cycle, and if no wait states are specified in the BCR register, zero wait states will be inserted in the external bus cycle regardless the status of  $\overline{\text{TA}}$  during the leading edge of t2.



**Figure 2-3  $\overline{\text{TA}}$  Controlled Accesses**

**$\overline{\text{BR}}$  (Bus Request) — active low output when in the master mode, active low input when in the slave mode.** After power-on reset, this pin is an input (slave mode). In this mode,  $\overline{\text{BR}}$  allows another device such as a processor or DMA

controller to become master of the DSP external data bus D0-D15 and the external address bus A0-A15. The DSP asserts  $\overline{BG}$  a few T states after the  $\overline{BR}$  input is asserted. The DSP bus controller will release control of the external data bus D0-D15, address bus A0-A15 and bus control pins  $\overline{PEREN}$ ,  $PS/\overline{DS}$ ,  $\overline{RD}$ ,  $\overline{WR}$ , and  $R/\overline{W}$  at the earliest time possible consistent with proper synchronization. These pins will then be placed in the high impedance state and the  $\overline{BB}$  pin will be deasserted. The DSP will continue executing instructions only if internal program and data memory resources are being accessed. If the DSP requests the external bus while the  $\overline{BR}$  input pin is asserted, the DSP bus controller inserts wait states until the external bus becomes available ( $\overline{BR}$  and  $\overline{BB}$  deasserted). Note that interrupts are not serviced when a DSP instruction is waiting for the bus controller. Note also that  $\overline{BR}$  is prevented from interrupting the execution of a read/ modify/ write instruction.

If the master bit in the OMR register is set, this pin becomes an output (Master Mode). In this mode, the DSP is not the external bus master and has to assert  $\overline{BR}$  to request the bus mastership. The DSP bus controller will insert wait states until  $\overline{BG}$  input is asserted and will then begin normal bus accesses after the rising of the clock which sampled  $\overline{BB}$  high. The  $\overline{BR}$  output signal will remain asserted until the DSP no longer needs the bus. In this mode, the Request Hold bit (RH) of the BCR allows  $\overline{BR}$  to be asserted under software control.

During external accesses caused by an instruction executed out of external program memory,  $\overline{BR}$  remains asserted low for consecutive external X memory accesses and continues toggling for consecutive external P memory accesses unless the Request Hold bit (RH) is set inside the BCR.

In the master mode,  $\overline{BR}$  can also be used for non arbitration purposes: if  $\overline{BG}$  is always asserted,  $\overline{BR}$  is asserted in  $t_0$  of every external bus access. It can then be used as a chip select to turn an external memory device off and on between internal and external bus accesses.  $\overline{BR}$  timing is, in that case, similar to A0-A15,  $R/\overline{W}$ , and  $PS/\overline{DS}$ ; it is asserted and deasserted during  $t_0$ .

**$\overline{BG}$  (Bus Grant) — active low input when in the master mode, active low output when in the slave mode.** This pin is an output after power on reset if the slave is selected and is asserted to acknowledge an external bus request. It indicates that the DSP will release control of the external address bus A0-A15, data bus D0-D15, and bus control pins when  $\overline{BB}$  is deasserted. The  $\overline{BG}$  output is asserted in response to a  $\overline{BR}$  input. When the  $\overline{BG}$  output is asserted and  $\overline{BB}$  is deasserted, the external address bus A0-A15, data bus D0-D15 and bus con-

control pins are in the high impedance state.  $\overline{BG}$  assertion may occur in the middle of an instruction which requires more than one external bus cycle for execution. Note that  $\overline{BG}$  assertion will not occur during indivisible read-modify-write instructions (BFSET, BFCLR, BFCHG). When  $\overline{BR}$  is deasserted and  $\overline{BB}$  pin is sampled high then  $\overline{BG}$  is deasserted and the DSP regains control of the external address bus, data bus, and bus control pins.

This pin becomes an input if the master bit in the OMR register is set (Master Mode). It is asserted by an external processor when the DSP may become the bus master. The DSP can start normal external memory access after the  $\overline{BB}$  pin has been deasserted by the previous bus master. When  $\overline{BG}$  is deasserted, the DSP will release the bus as soon as the current transfer is completed. The state of  $\overline{BG}$  may be tested by testing the BS bit in the Bus Control Register.

$\overline{BG}$  is ignored during hardware reset.

**$\overline{BB}$  (Bus Busy) — active low input when not a bus master, active low output when a bus master.** This pin is asserted by the DSP when it becomes the bus master and it performs an external access. It is deasserted when the DSP releases bus mastership.  $\overline{BB}$  becomes an input when the DSP is no longer the bus master.

### 2.4 INTERRUPT AND MODE CONTROL (4 PINS)

**MODA/ $\overline{IRQA}$  (Mode Select A/External Interrupt Request A)** — This input has two functions: (1) to select the initial chip operating mode and (2), after synchronization, to allow an external device to request a DSP interrupt. MODA is read and internally latched in the DSP when the processor exits the reset state. MODA and MODB select the initial chip operating mode. Several clock cycles after leaving the reset state, the MODA pin changes to external interrupt request  $\overline{IRQA}$ . The chip operating mode can be changed by software after reset. The  $\overline{IRQA}$  input is a synchronized external interrupt request which indicates that an external device is requesting service. It may be programmed to be level sensitive or negative edge triggered. If level sensitive triggering is selected, an external pull up resistor is required for wired-OR operation. If the processor is in the stop state and  $\overline{IRQA}$  is asserted, the processor will exit the stop state.

**MODB/ $\overline{IRQB}$  (Mode Select B/External Interrupt Request B)** — This input has two functions: (1) to select the initial chip operating mode and (2), after internal synchronization, to allow an external device to request a DSP interrupt. MODB is read and internally latched in the DSP when the processor exits the reset state.

MODA and MODB select the initial chip operating mode. Several clock cycles after leaving the reset state, the MODB pin changes to external interrupt request  $\overline{\text{IRQB}}$ . After reset, the chip operating mode can be changed by software. The  $\overline{\text{IRQB}}$  input is an external interrupt request which indicates that an external device is requesting service. It may be programmed to be level sensitive or negative edge triggered. If level sensitive triggering is selected, an external pull up resistor is required for wired-OR operation.

**MODC/ $\overline{\text{IRQC}}$  (Mode Select C/External Interrupt Request C)** — This input has two functions: (1) to select the initial bus operating mode and (2), after internal synchronization, to allow an external device to request a DSP interrupt. MODC is read and internally latched in the DSP when the processor exits the reset state. When tied high, the external bus is programmed in the master mode ( $\overline{\text{BR}}$  output and  $\overline{\text{BG}}$  input) and when tied low, the bus is programmed in the slave mode ( $\overline{\text{BR}}$  input and  $\overline{\text{BG}}$  output). After reset, the bus operating mode can be changed by software writing the MC bit of the OMR register. Several clock cycles after leaving the reset state, the MODC pin changes to the external interrupt request  $\overline{\text{IRQC}}$ . The  $\overline{\text{IRQC}}$  input is an external interrupt request which indicates that an external device is requesting service. It may be programmed to be level sensitive or negative edge triggered. If level sensitive triggering is selected, an external pull up resistor is required for wired-OR operation.

**$\overline{\text{RESET}}$  (Reset)** — This input is a direct hardware reset on the processor. When  $\overline{\text{RESET}}$  is asserted low, the DSP is initialized and placed in the reset state. A Schmitt trigger input is used for noise immunity. When the  $\overline{\text{RESET}}$  pin is deasserted, the initial chip operating mode is latched from the MODA and MODB pins. The internal reset signal should be deasserted synchronous with the internal clocks.

## 2.5 POWER, GROUND, AND CLOCK (30 PINS)

**VCC (8) (Power)** — power pins.

**VSS (15) (Ground)** — ground pins.

**VDDS (Synthesizer Power)** — This pin supplies a quiet power source to the VCO to provide greater frequency stability.

**GNDS (Synthesizer Ground)** — This pin supplies a quiet ground to the VCO to provide greater frequency stability.

**EXTAL (External Clock/Crystal Input)** — This input should be connected to an external clock or to an external oscillator. After being squared, the input clock can be

selected to directly provide the clock to the DSP core. In that case, it is divided by two into the core to produce a four phase instruction cycle clock, the minimum instruction time being two input clock periods. This input clock can also be selected as input clock for the on-chip codec and the on-chip PLL.

- CLKO (Clock Output)** — This pin outputs a buffered clock signal. By programming two bits (CS1-CS0) inside the PLL control register PLCR, the user can select between outputting a squared version of the signal applied to EXTAL, a squared version of the signal applied to EXTAL divided by 2, and a delayed version of the DSP core master clock. The clock frequency on this pin can be disabled by setting the Clockout Disable bit (CD; bit 7) of the Operating Mode Register (OMR).
- SXFC (External Filter Capacitor)** — This pin is used to add an external capacitor to the filter circuit of the phase locked loop. The capacitor should be connected between SXFC and VDDS.
- VDDA (Power Supply Input)** — This pin is the positive analog supply input. It should be connected to Vdd when the codec is not used.
- VSSA (Analog Ground)** — This pin is the analog ground return. It should be connected to Vss when the codec is not used.

## 2.6 HOST INTERFACE (15 PINS)

- H0-H7 (Host Data Bus)** — This bidirectional data bus is used to transfer data between the host processor and the DSP. This bus is an input unless enabled by a host processor read. H0-H7 may be programmed as general purpose parallel I/O pins called PB0-PB7 when the Host Interface (HI) is not being used.
- HA0-2 (Host Address 0-2)** — These inputs provide the address selection for each HI register and are stable when  $\overline{H\overline{EN}}$  is asserted. HA0-HA2 may be programmed as general purpose I/O pins called PB8-PB10 when the HI is not being used.
- HR/ $\overline{W}$  (Host Read/Write)** — This input selects the direction of data transfer for each host processor access. If HR/ $\overline{W}$  is high and  $\overline{H\overline{EN}}$  is asserted, H0-H7 are outputs and DSP data is transferred to the host processor. If HR/ $\overline{W}$  is low and  $\overline{H\overline{EN}}$  is asserted, H0-H7 are inputs and host data is transferred to the DSP. HR/ $\overline{W}$  is stable when  $\overline{H\overline{EN}}$  is asserted. HR/ $\overline{W}$  may be programmed as a general purpose I/O pin called PB11 when the HI is not being used.
- $\overline{H\overline{EN}}$  (Host Enable)** — This input enables a data transfer on the host data bus. When  $\overline{H\overline{EN}}$  is asserted and HR/ $\overline{W}$  is high, H0-H7 becomes an output and DSP data



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

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

**$\overline{HACK}$  (Host Acknowledge)** — This input has two functions: (1) to provide a Host Acknowledge signal for DMA transfers or (2) to control handshaking and to provide a Host Interrupt Acknowledge compatible with MC68000 family processors. If programmed as a Host Acknowledge signal,  $\overline{HACK}$  may be used as a data strobe for HI DMA data transfers. If programmed as an MC68000 Host Interrupt Acknowledge,  $\overline{HACK}$  is used to enable the HI Interrupt Vector Register (IVR) onto the Host Data Bus H0-H7 if the Host Request  $\overline{HREQ}$  output is asserted. In this case, all other HI control pins are ignored and the HI state is not affected.  $\overline{HACK}$  may be programmed as a general purpose I/O pin called PB14 when the HI is not being used.

## 2.7 16-BIT TIMER (2 PINS)

**TIN (Timer Input)** — This input receives external pulses to be counted by the on-chip 16-bit timer when external clocking is selected. The pulses are internally synchronized to the DSP core internal clock. TIN may be programmed as a general purpose I/O pin called PC10 when the external event function is not being used.

**TOUT (Timer Output)** — This output generates pulses, toggles on a timer overflow event, or toggles on a compare event. TOUT may be programmed as a general purpose I/O pin called PC11 when disabled by the timer out enable bits (TO2-TO0).

## 2.8 REDUCED SYNCHRONOUS SERIAL INTERFACES (RSSI0 AND RSSI1) AND PORT C (8 PINS)

**STD0/PC0 (RSSI0 Transmit Data)** — This output pin transmits serial data from the RSSI0 Transmit Shift Register. STD0 may be programmed as a general purpose I/O pin called PC0 when the RSSI0 STD0 function is not being used.

**SRD0/PC1 (RSSI0 Receive Data)** — This input pin receives serial data and transfers the data to the RSSI0 Receive Shift Register. SRD0 may be programmed as a general purpose I/O pin called PC1 when the RSSI0 SRD0 function is not being used.

**SCK0/PC2 (RSSI0 Serial Clock)** — This bidirectional pin provides the serial bit rate clock for the RSSI0 interface. The clock signal can be continuous or gated and is used by both the transmitter and receiver. SCK0 may be programmed as a general purpose I/O pin called PC2 when the RSSI0 interface is not being used.

**SFS0/PC4 (Serial Frame Sync 0)** — This bidirectional pin is used by the RSSI0 serial interface as frame sync I/O or flag I/O. The SFS0 is used by both the transmitter and receiver to synchronize data transfer and can be an input or an output. SFS0 may be programmed as a general purpose I/O pin called PC4 when the RSSI0 is not using this pin.

**STD1/PC5 (RSSI1 Transmit Data)** — This output pin transmits serial data from the RSSI1 Transmit Shift Register. STD1 may be programmed as a general purpose I/O pin called PC5 when the RSSI1 STD1 function is not being used.

**SRD1/PC6 (RSSI1 Receive Data)** — This input pin receives serial data and transfers the data to the RSSI1 Receive Shift Register. SRD1 may be programmed as a general purpose I/O pin called PC6 when the RSSI1 SRD function is not being used.

**SCK1/PC7 (RSSI1 Serial Clock)** — This bidirectional pin provides the serial bit rate clock for the RSSI1 interface. The clock signal can be continuous or gated and is used by both the transmitter and receiver. SCK1 may be programmed as a general purpose I/O pin called PC7 when the RSSI1 interface is not being used.

**SFS1/PC9 (Serial Frame Sync 1)** — This bidirectional pin is used by the RSSI1 serial interface as frame sync I/O or flag I/O. The SFS1 is used by both the transmitter and receiver to synchronize data transfer and can be an input or an output.

SFS1 may be programmed as a general purpose I/O pin called PC9 when the RSSI1 is not using this pin.

## 2.9 ON-CHIP EMULATION (4 PINS)

**DSI/OS0 (Debug Serial Input/Chip Status 0)** — The DSI/OS0 pin, when an input, is the pin through which serial data or commands are provided to the OnCE controller. The data received on the DSI pin will be recognized only when the DSP has entered the debug mode of operation. Data must have valid TTL logic levels before the serial clock falling edge. Data is always shifted into the OnCE serial port most significant bit (MSB) first. When the DSP is not in the debug mode, the DSI/OS0 pin provides information about the chip status if it is an output and used in conjunction with the OS1 pin.

**$\overline{\text{DSCK}}$ /OS1 (Debug Serial Clock/Chip Status 1)** — The  $\overline{\text{DSCK}}$ /OS1 pin, when an input, is the pin through which the serial clock is supplied to the OnCE. The serial clock provides pulses required to shift data into and out of the OnCE serial port. Data is clocked into the OnCE on the falling edge and is clocked out of the OnCE serial port on the rising edge. When the DSP is not in the debug mode, the  $\overline{\text{DSCK}}$ /OS1 pin provides information about the chip status if it is an output and used in conjunction with the OS0 pin.

**DSO (Debug Serial Output)** — The debug serial output provides the data contained in one of the OnCE controller registers as specified by the last command received from the command controller. When idle, this pin is high. When the requested data is available, the DSO line will be asserted (negative true logic) for four T cycles (one instruction cycle) to indicate that the serial shift register is ready to receive clocks in order to deliver the data. When the chip enters the debug mode due to an external debug request ( $\overline{\text{DR}}$ ), an internal software debug request (DEBUG), a hardware breakpoint occurrence, or a trace/step occurrence, this line will be asserted for three T cycles to indicate that the chip has entered the debug mode and is waiting for commands. Data is always shifted out the OnCE serial port most significant bit (MSB) first.

**$\overline{\text{DR}}$  (Debug Request Input)** — The debug request input provides a means of entering the debug mode of operation. This pin, when asserted (negative true logic), will cause the DSP to finish the instruction being executed, enter the debug mode, and wait for commands to be entered from the debug serial input line.

**2.10 ON-CHIP CODEC (7 PINS)**

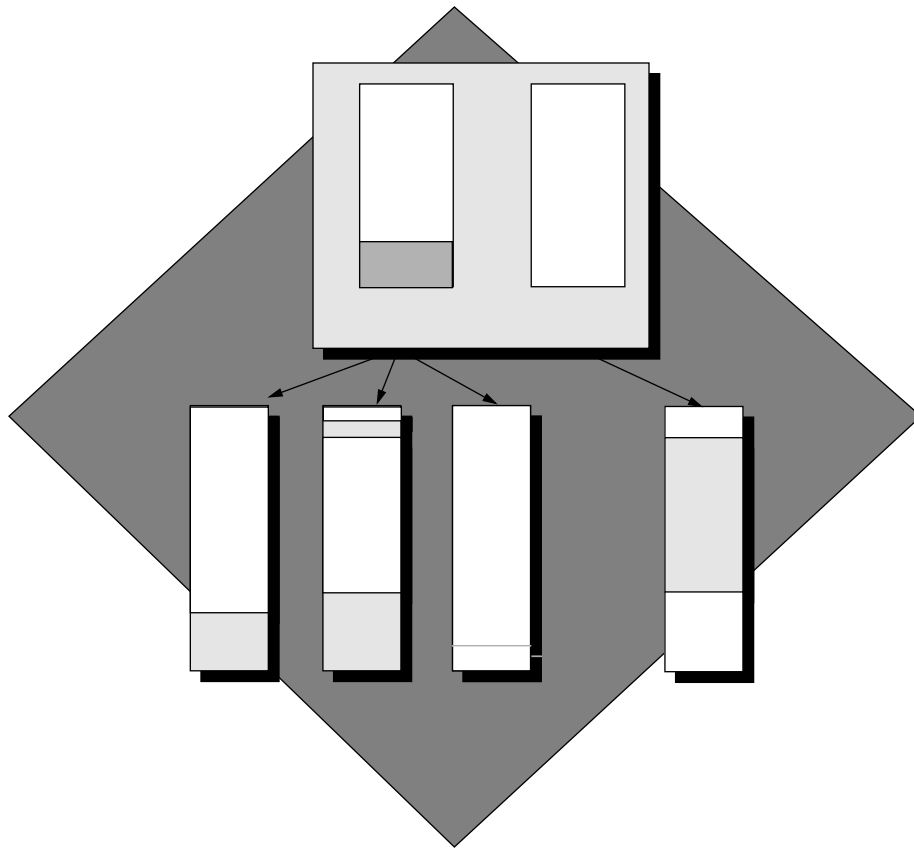
- AUX**     **(Auxiliary Input)** — This pin is selected as the analog input to the A/D converter when the INS bit is set in the codec control register COCR. This pin should be left floating when the codec is not used.
- MIC**     **(Microphone Input)** — This pin is selected as the analog input to the A/D converter when the INS bit is cleared in the codec control register COCR. This pin should be left floating when the codec is not used.
- SPKP**    **(Speaker Positive Output)** — This pin is the positive analog output from the on-chip D/A converter. This pin should be left floating when the codec is not used.
- SPKM**    **(Speaker Negative Output)** — This pin is the negative analog output from the on-chip D/A converter. This pin should be left floating when the codec is not used.
- VRAD**    **(Voltage Reference Output for the A/D)** — This pin is the output of the op-amp buffer in the A/D sections reference voltage generator. It has a value of  $(2/5) V_{DDA}$ . This voltage is used as the analog ground internal to the block. This pin should always be connected to ground through two capacitors, even when the codec is not used.
- VRDA**    **(Voltage Reference Output for the D/A)** — This pin is the output of the op-amp buffer in the D/A sections reference voltage generator. It has a value of  $(2/5) V_{DDA}$ . This voltage is used as the analog ground internal to the block. This pin should always be connected to the ground through two capacitors, even when the codec is not used.
- VDIV**    **(Voltage Division Output)** — This pin is the input to the op-amp buffer in the reference voltage generator. It is connected to a resistor divider network located within the codec block which provides a voltage equal to  $2/5 V_{DDA}$ . This pin should be left floating when the codec is not used.



---

## SECTION 3

# OPERATING MODES AND MEMORY CONFIGURATION



## SECTION CONTENTS

---

3.1	INTRODUCTION .....	3-3
3.2	DSP56166 RAM BASED DESCRIPTION .....	3-3
3.2.1.	X Data Memory .....	3-3
3.2.2.	Program Memory .....	3-5
3.2.3.	Bootstrap ROM .....	3-5
3.2.4.	RAM Based DSP56166 Operating Modes .....	3-5
3.2.4.1.	Bootstrap Mode (Mode 0). ....	3-6
3.2.4.2.	Bootstrap Mode (Mode 1). ....	3-6
3.2.4.3.	Normal Expanded Mode (Mode 2). ....	3-7
3.2.4.4.	Development Mode (Mode 3). ....	3-7
3.2.5.	Bootstrap Mode .....	3-7
3.2.5.1.	Bootstrap ROM .....	3-7
3.2.5.2.	Bootstrap Control Logic .....	3-7
3.2.5.3.	Bootstrap Program .....	3-8
3.3	DSP56166 ROM BASED DESCRIPTION .....	3-10
3.3.1.	X Data Memory .....	3-10
3.3.2.	Program Memory .....	3-11
3.3.3.	ROM Based DSP56166 Operating Modes .....	3-12
3.3.3.1.	Single-chip Mode (Mode 0). ....	3-12
3.3.3.2.	Single-chip Mode (Mode 1). ....	3-12
3.3.3.3.	Normal Expanded Mode (Mode 2). ....	3-13
3.3.4.	Development Mode (Mode 3). ....	3-13

### 3.1 INTRODUCTION

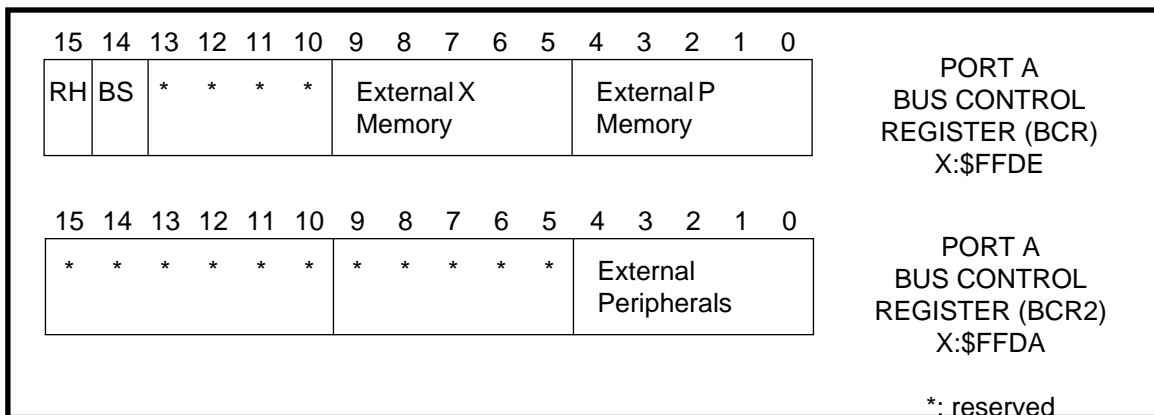
The DSP56166 is available in two different on-chip memory configurations: a RAM based part and a ROM based part. This section describes in detail the on-chip memories and the operating modes of the two versions.

### 3.2 DSP56166 RAM BASED DESCRIPTION

The RAM based DSP56166 uses **RAM** for the on-chip **Program Memory** and for the on-chip **Data Memory**. The two independent memory spaces, X data and program, are shown in Figure 3-2. The memory spaces are configured by control bits in the operating mode register (OMR). The operating mode control bits (MA and MB) in the OMR control the program memory map and select the reset vector address. Both the program and data memories can be expanded off-chip.

#### 3.2.1 X Data Memory

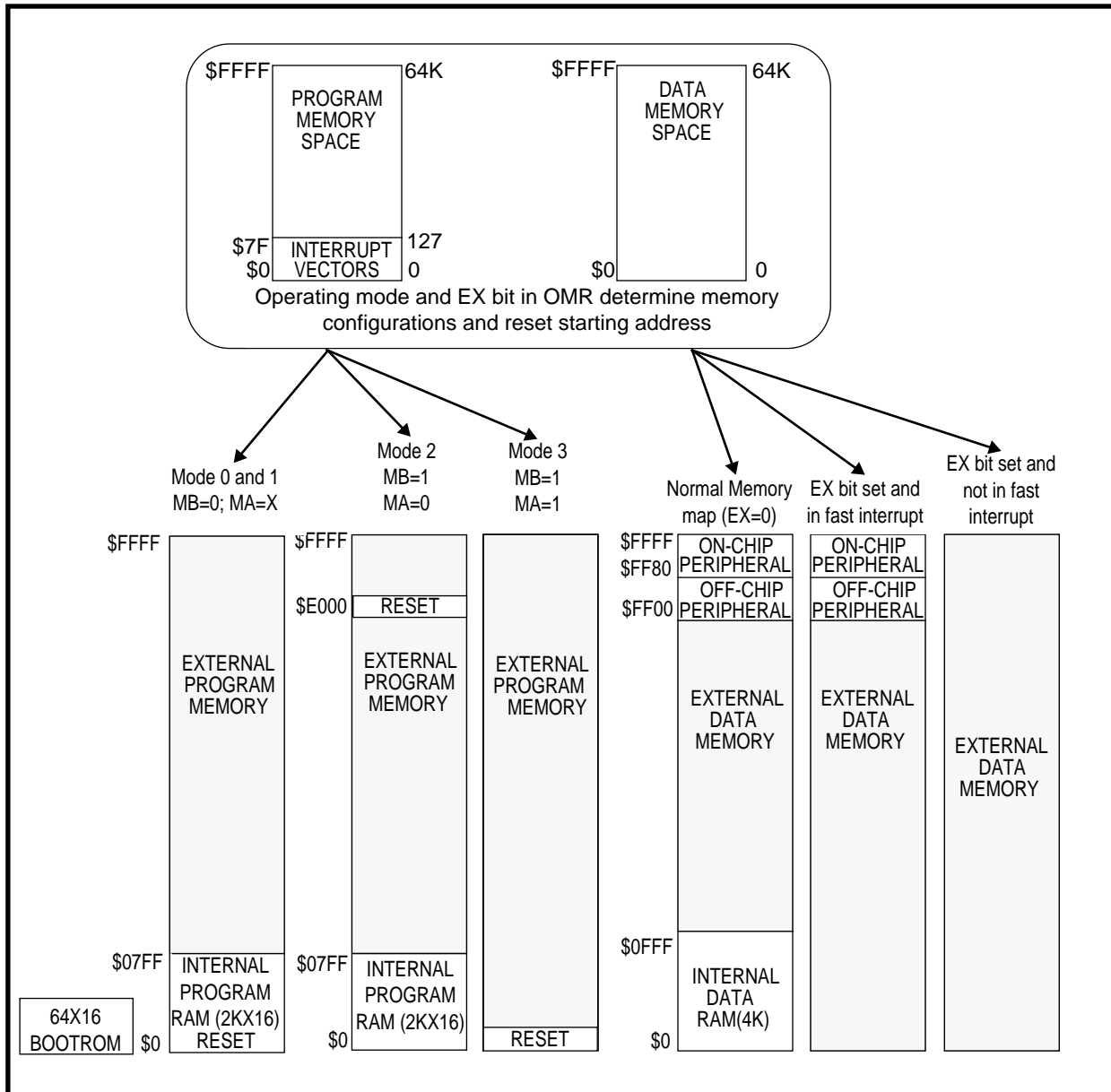
The DSP56166 has 4096 words of on-chip data RAM: 128 data memory locations are reserved for on-chip peripheral registers (X:\$FF80-FFFF) and 128 additional locations (X:\$FF00-FF7F) are reserved for off-chip peripheral accesses. The external bus access time on this external peripheral space is controlled by 5 bits of an additional bus control register, BCR2, located at X:\$FFDA (see Figure 3-1). Between 0 and 31 software programmable wait states can be generated. A special pin,  $\overline{\text{PEREN}}$ , is asserted low during accesses to the memory mapped external peripheral registers.



**Figure 3-1 External Peripheral Bus Control Register (BCR2)**

The X memory may be expanded off-chip for a total of 65,280 (65,536-256) addressable locations. The external data memory bus access time is controlled by 5 bits of an additional bus control register (BCR, located at X:\$FFDE). This register is described in Figure 3-1.





**Figure 3-2 DSP56166 RAM based Memory Map**

The External X memory bit (EXT bit 3) of the OMR determines the mapping of the X memory as shown in Figure 3-2. Setting this bit completely disables the on-chip data memory and enables the full 64 K external memory map. This bit is ignored by the fast interrupt process so that the on-chip peripheral space and off-chip peripheral space can be accessed during fast interrupts. During long interrupts, this bit, if set before the interrupt, can be cleared by the user program inside the interrupt routine in order to access the on-chip peripheral space and the on-chip memory; it can be set again before the RTI instruction.

When the EX bit is set, external data memory access is only controlled by the BCR except during fast interrupts, where BCR2 controls external accesses to the off-chip peripheral space.

### **3.2.2 Program Memory**

The RAM based DSP56166 has 2048 words of on-chip program RAM. The first 128 locations of program memory are reserved for interrupt vectors. The program memory may be expanded off-chip for a total of 65,536 addressable locations. The external data memory bus access time is controlled by 5 bits of the bus control register BCR located at X:\$FFDE. This register is described in Figure 3-1.

### **3.2.3 Bootstrap ROM**

The Bootstrap ROM is a 64 location by 16-bit factory programmed ROM which is used only in the bootstrap modes, Operating Modes 0 and 1, during which the on-chip program RAM is defined as write-only. The bootstrap ROM is not accessible by the user and is disabled in normal operating modes. Refer to **APPENDIX A BOOTSTRAP OPERATING MODE — OPERATING MODE 0 or 1** for a full description of the bootstrap feature of the DSP56166.

The bootstrap program can load from any one of three different sources. Selection of which one of the three is made by reading the mode pins and, if necessary, bit-15 of P:\$C000 from the external data bus.

If MB:MA = 00 (Mode 0) then the bootstrap program will load from an external byte-wide memory. This bootstrap program will load 4,096 bytes from the external P: memory space beginning at location P:\$C000 (bits 0-7). These will be packed into 2,048 16-bit words and stored in contiguous internal program RAM memory locations starting at P:\$0000. The byte-wide data will be packed into the 16-bit memory least significant byte first.

If MB:MA = 01 (Mode 1), the bootstrap program will read bit-15 of P:\$C000 from the external data bus. If bit-15 = 0, the bootstrap program will load 4,096 bytes through the host port (the host processor can terminate down-loading early by setting HF0=0). If bit-15 = 1, the bootstrap program will load through RSSI0. Data is packed into program RAM least significant byte of P:\$0000 first.

### **3.2.4 RAM Based DSP56166 Operating Modes**

The DSP operating modes determine the memory maps for program and data memories and the start-up procedure when the DSP leaves the reset state. The MODA, MODB, and MODC pins are sampled as the DSP leaves the reset state and the initial operating mode of the DSP is set accordingly. After the reset state is exited, the MODA and MODB pins become general-purpose interrupt pins,  $\overline{\text{IRQA}}$  and  $\overline{\text{IRQB}}$ . One of three initial operating modes is selected: single chip, normal expanded, or development. chip operating modes

**Table 3-1 Operating Mode Summary  
Program RAM Part**

Operating Mode	M B	M A	Description
Special Bootstrap 1	0	0	Bootstrap from an external byte-wide memory located at P:\$C000. Reset at P:\$0000
Special Bootstrap 2	0	1	Bootstrap from the Host port (P:\$C000 bit 15=0) or RSSI0 (P:\$C000 bit 15=0) Reset at P:\$0000
Normal Expanded	1	0	Internal PRAM enabled; External reset at P:\$E000
Development Expanded	1	1	Internal program memory disabled; External reset at P:\$0000.

can be changed by writing the operating mode bits (MB, MA) in the OMR. Changing operating modes does not reset the DSP. It is desirable to disable interrupts immediately before changing the OMR to prevent an interrupt from going to the wrong memory location. Also, one no-operation (NOP) instruction should be included after changing the OMR to allow for remapping to occur.

#### **3.2.4.1 Bootstrap Mode (Mode 0).**

Mode 0 is one of two single-chip modes which have all internal program memories enabled (see Figure 3-2). This mode can be entered by either grounding both mode pins and resetting the chip or by writing to the OMR and changing the MA and MB bits. When the operating mode is first changed to Mode 0, the DSP56166 executes a bootstrap program which loads program memory from a byte wide memory located at P:\$C000 (see Table 3-1). Section 3.2.5.2 describes the bootstrap operation. The memory maps for Mode 0 and Mode 1 are identical. The only difference between the two modes is the location of the reset vector in program memory. The reset vector location in Mode 0 is P:\$0000. The reset vector location in Mode 2 is P:\$E000 (external memory).

#### **3.2.4.2 Bootstrap Mode (Mode 1).**

Mode 1 is one of two single-chip modes which have all internal program and data RAM memories enabled (see Figure 3-2). This mode can be entered by either grounding the MB pin and pulling the MA pin high or by writing to the OMR and changing the MA and MB bits (see Table 3-1). When the operating mode is first changed to Mode 1, the DSP56166 executes a bootstrap program which loads program memory from either the

host port or the RSSI0 depending on whether bit 15 of location P:C000 is a zero (host port) or a one (RSSI0). Section 3.2.5.2 describes the bootstrap operation. The memory maps for Mode 0 and Mode 1 are identical. The memory maps for Mode 1 and Mode 2 are very similar. The memory map difference between Mode 0 and Mode 2 is the location of the reset vector in program memory. The reset vector location in Mode 0 is P:\$0000. The reset vector location in Mode 2 is P:\$E000.

#### **3.2.4.3 Normal Expanded Mode (Mode 2).**

The normal expanded mode (Mode 2) has the same memory map as Mode 0 and Mode 1 (see Figure 3-2). The difference is that entering Mode 2 does not cause the bootstrap program to be executed and the reset vectors to external program memory location P:\$C000. This mode can be entered by either grounding the MA pin and pulling the MB pin high or by writing to the OMR and changing the MA and MB bits (see Table 3-1).

#### **3.2.4.4 Development Mode (Mode 3).**

The development mode is similar to the normal expanded mode except that internal program memory is disabled (see Figure 3-2). All references to program memory space are directed to external program memory, which is accessed on the external data bus. This mode can be entered by either pulling the MA and MB pins high or by writing to the OMR and changing the MA and MB bits (see Table 3-1). The reset vector location in Mode3 is external program memory location P:\$0000.

### **3.2.5 Bootstrap Mode**

The bootstrap feature consists of a special on-chip bootstrap ROM containing a bootstrap program and a bootstrap control logic. The bootstrap feature is only available on the program RAM part. It is not available on the program ROM part. Appendix A describes the contents of the boot ROM.

#### **3.2.5.1 Bootstrap ROM**

This 64-word on-chip ROM is factory programmed to perform the actual bootstrap operation from the memory expansion port (Port A), from the Host Interface, or from the Reduced Synchronous Serial Interface RSSI0. No access is provided to the bootstrap ROM other than through the bootstrap process. Control logic will disable the bootstrap ROM during normal operations.

#### **3.2.5.2 Bootstrap Control Logic**

The bootstrap mode control logic is activated when the DSP is placed in one of the bootstrap modes, Mode 0 or Mode 1. The control logic maps the bootstrap ROM into program

memory space until the bootstrap program changes operating modes when the bootstrap load is completed.

When the DSP exits the reset state in Mode 0 or 1, the following actions occur:

1. The control logic maps the bootstrap ROM into the internal DSP program memory space starting at location P:\$0000. All program fetches during the bootstrap operation are from the bootstrap ROM.
2. The control logic forces the entire internal program RAM space to be write-only memory during the bootstrap loading process. All write operations during the bootstrap program execution are to the PRAM.
3. Program execution begins at location \$0000 in the bootstrap ROM. The bootstrap ROM program performs the load of the internal program RAM (PRAM) through either the memory expansion port from a byte-wide external memory, through the Host Interface, or through the Reduced Synchronous Serial Interface RSSI0.
4. Upon completing the program RAM load, the bootstrap program terminates the bootstrap operation by entering Operating Mode 2 (writing to the OMR) and by branching to the internal program RAM location P:\$0000. During the execution of the branch to P:\$0000, the bootstrap ROM is disabled and fetches from the PRAM are re-enabled.

The bootstrap mode may also be selected by setting the OMR bits for Operating Mode 0 or 1. This initiates a timed operation to map the bootstrap ROM into the program address space after a delay to allow execution of a single instruction and a jump to P:\$0000 to start executing the bootstrap program. This technique allows the user to reboot the internal PRAM (with a different program if desired).

### **3.2.5.3 Bootstrap Program**

The bootstrap ROM contains the bootstrap firmware program that performs initial loading of the DSP's internal program RAM (see Appendix A for a listing of the bootstrap code). The program is written in DSP5616 core assembly language. It contains three separate methods of initializing the PRAM: loading from a byte-wide memory starting at location P:\$C000, loading through the Host Interface, or loading through the Reduced Synchronous Serial Interface RSSI0.

When Mode 0 is selected, the external bus version of the bootstrap is executed. The data contents of the external byte-wide memory must be organized as shown in Table 3-2.

**Table 3-2 Data Mapping for External Bus Bootstrap**

Address of External Byte-wide Memory	Contents Loaded to Internal PRAM at:
P:\$C000	P:\$0000 low byte
P:\$C001	P:\$0000 high byte
*	*
*	*
P:\$CFFE	P:\$07FF low byte
P:\$CFFF	P:\$07FF high byte

When Mode 1 is selected, the bootstrap is performed through the Host port or the RSSI0 depending on the level of the most significant bit of P:\$C000.

If Bit 15 of P:\$C000 is zero (a pull-down resistor can be used in some applications), the host port bootstrap is selected. Typically a host processor will be connected to the 16-bit DSP Host Interface and a host microprocessor will write the Host Interface registers TXH and TXL with the desired contents of PRAM from locations P:\$0000 to P:\$07FF. If less than 2048 words are to be loaded into the PRAM, the host programmer can terminate the bootstrap process by setting HF0=1 in the Host Interface.

If bit 15 of P:\$C000 is set (a pull-up resistor can be used in some applications), the bootstrap is performed through the Reduced Synchronous Serial Interface RSSI0. The bootstrap program sets up the RSSI0 in 8 bit mode, external clock, and synchronous mode.

### 3.3 DSP56166 ROM BASED DESCRIPTION

The RAM DSP56166 uses a **combination of RAM and ROM** for the on-chip **Program Memory** and for the on-chip **Data Memory**. The two independent memory spaces, X data, and program, are shown in Figure 3-3. The memory spaces are configured by control bits in the operating mode register (OMR). The operating mode control bits (MA and MB) in the OMR control the program memory map and select the reset vector address. Both the program and data memories can be expanded off-chip.

#### 3.3.1 X Data Memory

DSP56166 has 4096 words of on-chip data RAM and 4096 words of on-chip data ROM. 128 data memory locations are reserved for on-chip peripheral registers (X:\$FF80-FFFF) and 128 additional locations (X:\$FF00-FF7F) are reserved for off-chip peripheral accesses. The external bus access time on this external peripheral space is controlled by 5 bits of an additional bus control register BCR2 located at X:\$FFDA. This register is described in Figure 3-1. Between 0 and 31 software programmable wait states can be generated. A special pin,  $\overline{\text{PEREN}}$ , is asserted low during accesses to the memory mapped external peripheral registers. The X memory may be expanded off-chip for a total of 65,280 (65,536-256) addressable locations.

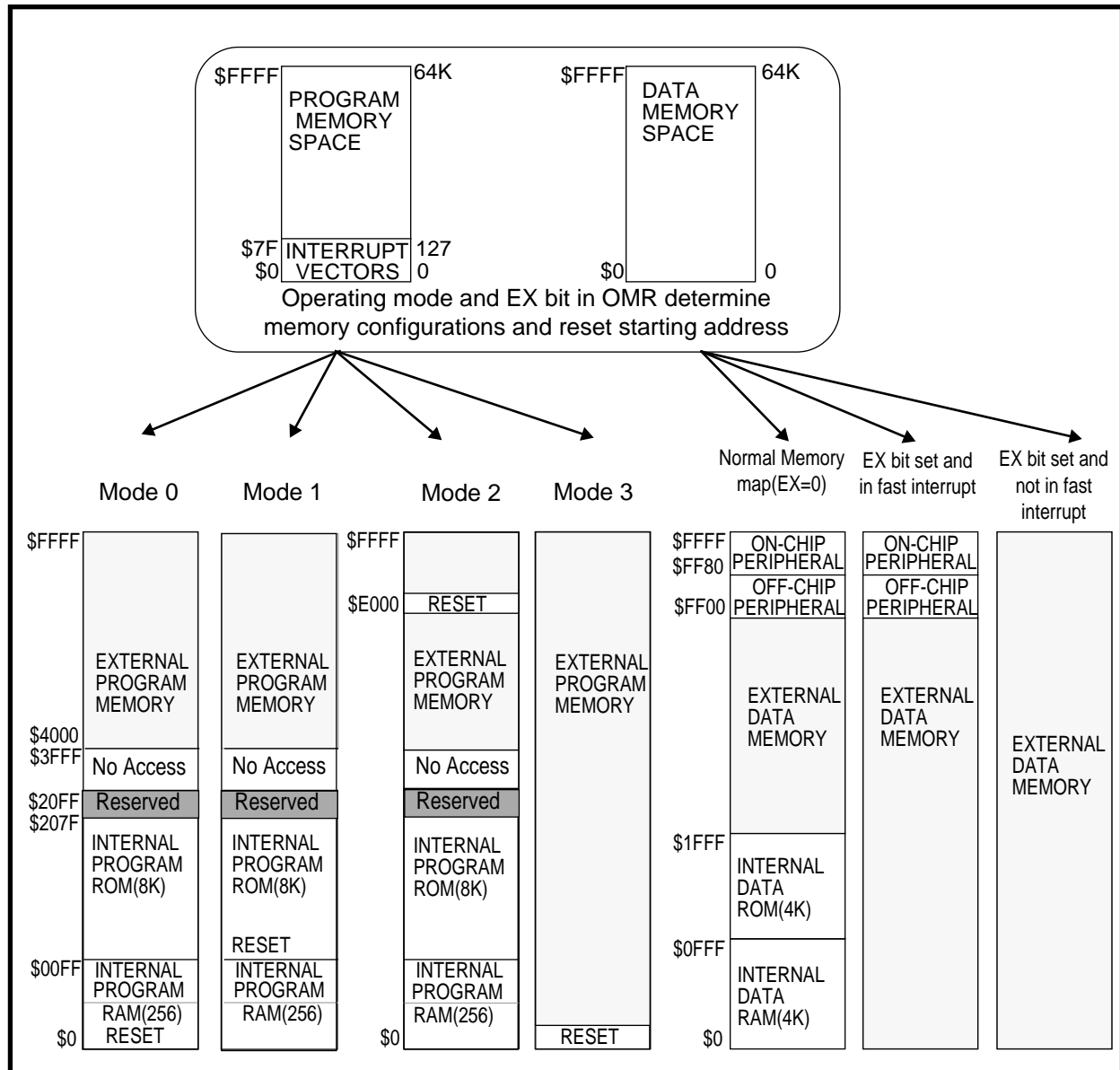
The External X memory bit (EXT bit-3) of the OMR determines the mapping of the X memory as shown on Figure 3-3. Setting this bit completely disables the on-chip data memory and enables a full 64 K external memory map. This bit is ignored by the fast interrupt process so that the on-chip peripheral space and off-chip peripheral space can be accessed during fast interrupts. During long interrupts, this bit, if set before the interrupt, can be cleared by the user program inside the interrupt routine in order to access the on-chip peripheral space and the on-chip memory; it can be set again before the RTI instruction.

When the EX bit is set, the external access on the data memory is controlled by the BCR only. Except during fast interrupts, where BCR2 will control external accesses to the off-chip peripheral space.

### NOTICE

**The on-chip XROM on the ROM based part is only connected to the XAB1 address bus.** Therefore, the data located in this ROM is only accessible by the **first** read during **dual** parallel read instructions.

During development using the **RAM** based part, the data to be mapped into the on-chip XROM of the **ROM** based part **should not** be accessed with a **second** read during a **dual** parallel read instruction.



**Figure 3-3 DSP56166 ROM Based Memory Map**

## 3.3.2 Program Memory

DSP56166 has 8192 words of on-chip program ROM and 256 words of on-chip program RAM. Since the DSP5616 core specifies on-chip program memory by blocks of power of two words, there is a non-accessible hole in the program memory between addresses P:\$20FF and P:\$3FFF (P:8447-P:16383).

External program memory will start at location P:\$4000 (P:16384) for mode 0,1 and 2 like shown on Figure 3-3.

When mode 3 is selected, the complete 64 K words of program memory are external.



**Note:** The last 128 locations of the on-chip program ROM are reserved and are not available for use (P:\$2080-\$20FF).

### 3.3.3 ROM Based DSP56166 Operating Modes

The DSP operating modes determine the memory maps for program and data memories and the start-up procedure when the DSP leaves the reset state. The MODA, MODB, and MODC pins are sampled as the DSP leaves the reset state and the initial operating mode of the DSP is set accordingly. After the reset state is exited, the MODA and MODB pins become general-purpose interrupt pins,  $\overline{\text{IRQA}}$ , and  $\overline{\text{IRQB}}$ . One of three initial operating modes is selected: single chip, normal expanded, or development. Chip operating modes can be changed by writing the operating mode bits (MB, MA) in the OMR. Changing operating modes does not reset the DSP. It is desirable to disable interrupts immediately before changing the OMR to prevent an interrupt from going to the wrong memory location. Also, one no-operation (NOP) instruction should be included after changing the OMR to allow for remapping to occur.

**Note:** Since the on-chip X data ROM is connected to the XAB1 address bus, the data located in this ROM is only accessible by the **first** read during **dual** parallel read instructions.

**Table 3-3 DSP56166 ROM Based Operating Modes**

MB	MA	Chip Operating Mode	Reset Vector	Program Memory Configuration
0	0	Single Chip	Internal PRAM P:\$0	Internal Pmem Enabled
0	1	Single Chip	Internal PROM P:\$100	Internal Pmem Enabled
1	0	Normal Expanded	External Pmem P:\$E000	Internal Pmem Enabled
1	1	Development	External Pmem P:\$0	Internal Pmem Disabled

#### 3.3.3.1 Single-chip Mode (Mode 0).

Mode 0 is one of two single-chip modes which have all internal program and data memories enabled (see Figure 3-3). This mode can be entered by either grounding both mode pins before resetting the chip or by writing to the OMR and changing the MA and MB bits. The memory maps for Mode 0 and Mode 1 are identical. The only difference between the two modes is the location of the reset vector in program memory. The reset vector location in Mode 0 is P:\$0000 in the internal PRAM; whereas, the reset vector location in Mode 1 is P:\$100 in the internal PROM.

#### 3.3.3.2 Single-chip Mode (Mode 1).

Mode 1 is one of two single-chip modes which have all internal program and data memories enabled (see Figure 3-3). This mode can be entered by either grounding the MB pin and pulling the MA pin high before resetting the chip, or by writing to the OMR and chang-

ing the MA and MB bits. The memory maps for Mode 0 and 1 are identical. The only difference between these two modes is the reset vector location in program memory. The reset vector location in Mode 0 is P:\$0000 in internal PRAM; whereas, the reset vector location in Mode 1 is P:\$100 in internal PROM.

#### **3.3.3.3 Normal Expanded Mode (Mode 2).**

The normal expanded mode (Mode 2) can be entered by either pulling the MB pin high and grounding the MA pin before resetting the chip, or by writing to the OMR and changing the MA and MB bits. The memory maps for Mode 0,1, and 2 are identical. The only difference between the three modes is the location of the reset vector in program memory. The reset vector location in Mode 2 is located in the external program memory space at location P:\$E000.

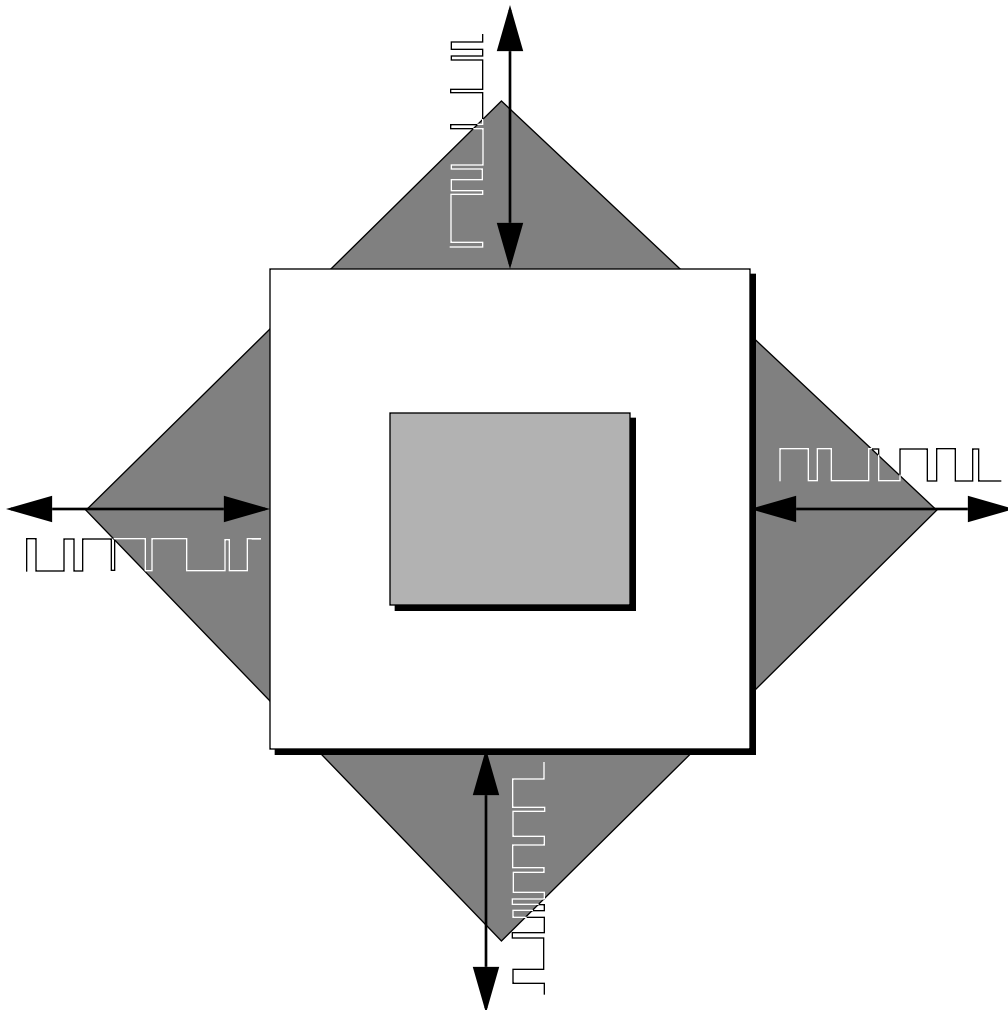
#### **3.3.3.4 Development Mode (Mode 3).**

The development mode is similar to the normal expanded mode except that internal program memory is disabled (see Figure 3-3). All references to program memory space are directed to external program memory, which is accessed on the external data bus. This mode can be entered by either pulling the MA and MB pins high or by writing to the OMR and changing the MA and MB bits (see Table 3-3). DSP56166 ROM based chips with bad or obsolete internal program ROM code can be used with external program memory in the development mode. Reset vectors to external program memory location P:\$0000.



## SECTION 4

# DSP56166 I/O INTERFACE



## SECTION CONTENTS

---

4.1	INTRODUCTION .....	4-3
4.2	I/O PORT SET-UP AND PROGRAMMING .....	4-3
4.2.1.	Port Registers .....	4-6
4.2.1.1.	Bus Control Registers (BCR and BCR2) .....	4-6
4.2.1.2.	Port B and Port C Registers .....	4-7

## **4.1 INTRODUCTION**

The DSP56166 provides 16 pins for an external address bus, 16 pins for an external data bus, and 10 pins for bus control. These pins are grouped to form the Port A bus interface. The DSP56166 also provides 25 programmable I/O pins. These pins may be used as general purpose I/O pins or allocated to on-chip peripherals. Four digital on-chip peripherals are provided on the DSP56166: an 8 bit parallel Host MPU/DMA Interface, a 16-bit timer, and two Reduced Synchronous Serial Interfaces (RSSI0 and RSSI1). These 25 pins are separate from the DSP56166 address and data buses and are grouped as two I/O ports (B and C). Figure 4-1 shows the I/O block diagram.

Port B is a 15-bit I/O interface which may be used either as general purpose I/O pins or as Host MPU/DMA Interface pins. The Host MPU/DMA Interface provides a dedicated 8-bit parallel port to a host microprocessor or DMA controller and can provide debugging facilities via host exceptions.

Port C is a 10-bit I/O interface which may be used as general purpose I/O pins or as Timer and Serial Interface pins. The 16-bit timer can generate periodic interrupts based on a multiple of the internal or external clock. The two Reduced Synchronous Serial Interfaces, RSSI0 and RSSI1, are identical. They provide high speed synchronous serial data communication capability between the DSP56166 and other serial devices. Support for TDM network configurations allows communication among up to eight devices.

These I/O interfaces are intended to minimize system chip count and “glue” logic in many DSP applications. Each I/O interface has its own control, status, and data registers and is treated as memory-mapped I/O by the DSP56166 (see Figure 4-2 and Figure 4-3). Each interface has several dedicated interrupt vector addresses and control bits to enable/disable interrupts. This minimizes the overhead associated with servicing the device since each interrupt source may have its own service routine.

## **4.2 I/O PORT SET-UP AND PROGRAMMING**

Port A Bus Control Registers BCR and BCR2, located respectively at X:\$FFDE and X:\$FFDA, may be programmed to insert wait states in a bus cycle during external data and program memory accesses for BCR and during external peripheral accesses for BCR2. Five bits are available in each control register for each type of external memory access. Each 5 bit field can specify up to 31 wait states. On processor reset, these five bits for both P and X memories are preset to all ones so that 31 wait states are inserted allowing slow, inexpensive memory to be used immediately after reset. All other Port Control Register bits are cleared on processor reset; i.e., reset sets the BCR to \$03FF and BCR2 to \$001F.

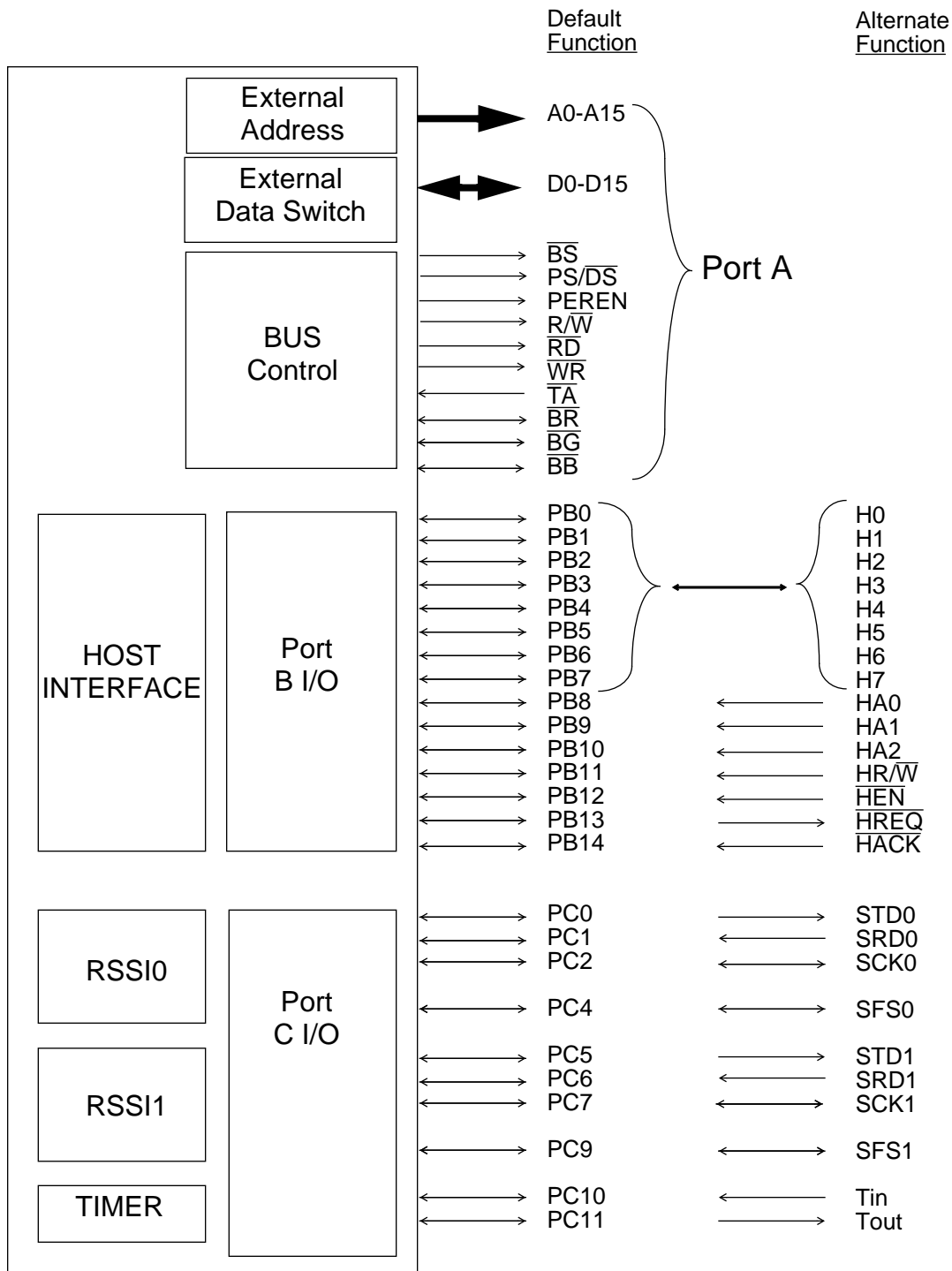


Figure 4-1 DSP56166 Input/Output Block Diagram

## I/O PORT SET-UP AND PROGRAMMING

\$FFFF	Reserved for on-chip emulation	\$FFDF	IPR: Interrupt Priority Register
\$FFFE		\$FFDE	BCR: Bus Control Register
\$FFFD		\$FFDD	IPR2: Interrupt Priority Register 2
\$FFFC		\$FFDC	PCR1
\$FFFB		\$FFDB	PCR0
\$FFFA		\$FFDA	BCR2: Bus Control Register 2
\$FFF9	TX/RX RSSI1 TX/RX Registers	\$FFD9	CRB-RSSI1 Control Register B
\$FFF8	SR/TSR RSSI1 Status Register	\$FFD8	CRA-RSSI1 Control Register A
\$FFF7		\$FFD7	
\$FFF6		\$FFD6	
\$FFF5		\$FFD5	
\$FFF4		\$FFD4	
\$FFF3		\$FFD3	
\$FFF2		\$FFD2	
\$FFF1	TX/RX RSSI0 TX/RX Registers	\$FFD1	CRB-RSSI0 Control Register B
\$FFF0	SR/TSR RSSI0 Status Register	\$FFD0	CRA-RSSI0 Control Register A
\$FFEF	Timer Preload Register(TPR)	\$FFCF	
\$FFEE	Timer Compare Register(TCPR)	\$FFCE	
\$FFED	Timer Count Register(TCTR)	\$FFCD	
\$FFEC	Timer Control Register(TCR)	\$FFCC	
\$FFEB		\$FFCB	
\$FFEA		\$FFCA	
\$FFE9	<u>CRX/CTX</u>	\$FFC9	reserved
\$FFE8	<u>COSR</u>	\$FFC8	CCR1
\$FFE7		\$FFC7	CCR0
\$FFE6		\$FFC6	
\$FFE5	HTX/HRX: Host TX/RX Register	\$FFC5	
\$FFE4	HSR: Host Status Register	\$FFC4	HCR: Host Control Register
\$FFE3	Port C Data Register (PCD)	\$FFC3	Port C Data Direction Register
\$FFE2	Port B Data Register (PBD)	\$FFC2	Port B Data Direction Register
\$FFE1		\$FFC1	Port C Control Register (PCC)
\$FFE0		\$FFC0	Port B Control Register (PBC)

**Figure 4-2 DSP56166 I/O and On-Chip Peripheral Memory Map**



Ports B and C pins may be programmed under software control as general purpose I/O pins or as dedicated on-chip peripheral pins. A Port Control Register is associated with each port which allows the port pins to be selected for one of these two functions. All port B pins are collectively configured as general purpose I/O pins if the corresponding Port Control Register bit is cleared and all are configured as HI pins if the corresponding Port Control Register bit is set. In contrast, each Port C pin is independently configured as a general purpose I/O pin if the corresponding Port Control Register bit is cleared and is configured as an RSSI pin or Timer pin if the corresponding Port Control Register bit is set. If a port pin is selected as a general purpose I/O pin, the direction of that pin is determined by a corresponding control bit in the Port Data Direction Register. The port pin is configured as an input if the corresponding Data Direction Register bit is cleared and is configured as an output if the corresponding Data Direction Register bit is set. All Port Control Register bits and Data Direction Register bits are cleared on processor reset, configuring all port pins as general purpose input pins. If the port pin is selected as an on-chip peripheral pin, the corresponding data direction bit is ignored and the direction of that pin is determined by the operating mode of the on-chip peripheral.

A port pin configured as a general purpose I/O pin is accessed through an associated Port Data Register B or C. Data written to the Port Data Register is stored in an output latch. If the port pin is configured as an output, the output latch data is driven out on the port pin. When the Port Data Register is read, the logic value on the output port pin is read. If the port pin is configured as an input, data written to the Port Data Register is still stored in the output latch but is not gated to the port pin. When the Port Data Register is read, the state of the port pin is read. That is, reading the port data register will reflect the state of the pins regardless of how they were configured.

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

#### **4.2.1 Port Registers**

Ports A, B and C are controlled by programmable registers. Port A is controlled by the Bus Control Registers which control memory wait states. Ports B and C each have registers that select the peripheral to be available and control that peripheral. See Figure 4-3.

##### **4.2.1.1 Bus Control Registers (BCR and BCR2)**

Port A Bus Control Registers (BCR and BCR2) are 16-bit read/write registers. They can be programmed to insert wait states in a bus cycle during external memory accesses and external peripheral accesses.

In the BCR, 5 bit wait control fields specify between 0 and 31 wait states for an external X memory and P memory access. Wait state fields are set to \$1F during hardware reset.

**Bit 15 of the BCR**, the Bus Request Hold bit, (RH), can be used for direct software control of the  $\overline{\text{BR}}$  pin. When this bit is set, the  $\overline{\text{BR}}$  pin is asserted even though the DSP does not need the bus. If RH is cleared, the  $\overline{\text{BR}}$  pin will only be asserted if an external access is being attempted or pending. RH is cleared by hardware reset.

**Bit 14 of the BCR**, the Bus State status bit (BS), is set if the DSP is currently the bus master. If the DSP is not the bus master, BS is cleared. In the slave mode, the BS bit is set when the  $\overline{\text{BG}}$  output pin is high and cleared when  $\overline{\text{BG}}$  is low. In the master mode, BS is cleared when the  $\overline{\text{BG}}$  input pin is high and set when both pins ( $\overline{\text{BG}}$  and BB) are low. This bit is set by hardware reset.

128 locations of the external data space (X:\$FF00-FF7F) are reserved for off-chip peripheral accesses. The external bus access time on this external peripheral space is controlled by 5 bits of BCR2 located at X:\$FFDA. Between 0 and 31 software programmable wait states can be generated. A special pin,  $\overline{\text{PEREN}}$ , is asserted low during accesses to the memory mapped external peripheral registers. Wait state fields are set to \$1F during hardware reset.

#### 4.2.1.2 Port B and Port C Registers

Port B consists of three read/write registers — a 1-bit Port B Control Register (PBC), a 15-bit Port B Data Direction Register (PBDDR), and a 15-bit Port B Data Register (PBD).

Port C consists of three read/write registers — a 12-bit Port C Control Register (PCC), a 12-bit Port C Data Direction Register (PCDDR), and a 12 bit Port C Data Register (PCD). Only ten bits in these three registers have pins and are available for GPIO. Bits 3 and 8 are not connected to pins. These registers are shown in Figure 4-3. All registers are read/write. Bit manipulation instructions can be used to access individual bits.

## I/O PORT SET-UP AND PROGRAMMING

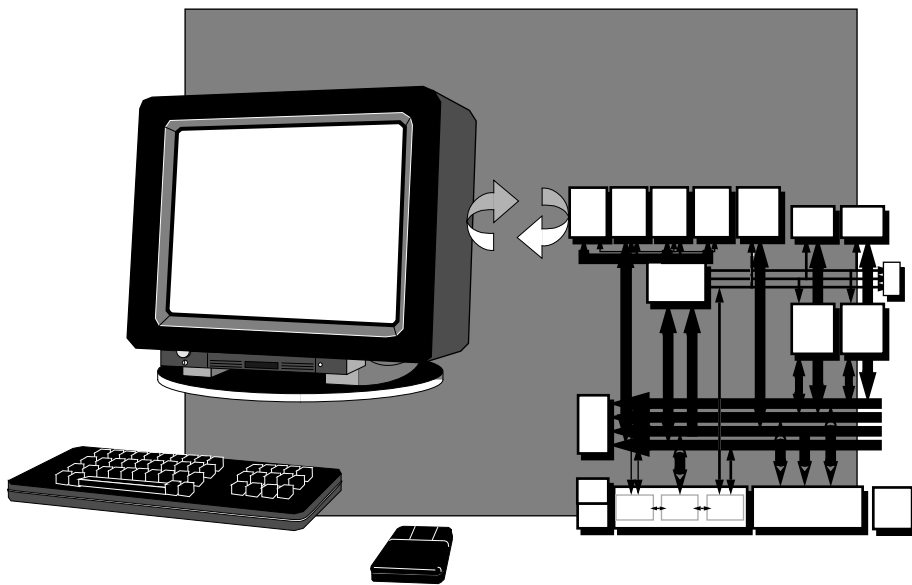
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RH	BS	**	**	**	**	External X Memory					External P Memory					
																PORT A BUS CONTROL REGISTER (BCR) X:\$FFDE
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
**						**					External Peripherals					
																PORT A BUS CONTROL REGISTER (BCR2) X:\$FFDA
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	BC 0	
																PORT B CONTROL REGISTER (PBC) X:\$FFC0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
**	DB 14	DB 13	DB 12	DB 11	DB 10	DB 9	DB 8	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0	
																PORT B DATA DIRECTION REGISTER (PBDDR) X:\$FFC2
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
**	PB 14	PB 13	PB 12	PB 11	PB 10	PB 9	PB 8	PB 7	PB 6	PB 5	PB 4	PB 3	PB 2	PB 1	PB 0	
																PORT B DATA REGISTER (PBD) X:\$FFE2
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
**	**	**	**	CC 11	CC 10	CC 9	**	CC 7	CC 6	CC 5	CC 4	**	CC 2	CC 1	CC 0	
																PORT C CONTROL REGISTER (PCC) X:\$FFC1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
**	**	**	**	DC 11	DC 10	DC 9	**	DC 7	DC 6	DC 5	DC 4	**	DC 2	DC 1	DC 0	
																PORT C DATA DIRECTION REGISTER (PCDDR) X:\$FFC3
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
**	**	**	**	PC 11	PC 10	PC 9	**	PC 7	PC 6	PC 5	PC 4	**	PC 2	PC 1	PC 0	
																PORT C DATA REGISTER (PCD) X:\$FFE3
																**.. reserved

\*\* : reserved

**Figure 4-3 DSP56166 I/O Port B and C Programming Models**

## SECTION 5

# HOST INTERFACE



## SECTION CONTENTS

---

5.1	INTRODUCTION .....	5-3
5.2	HOST INTERFACE PROGRAMMING MODEL .....	5-5
5.3	HOST TRANSMIT DATA REGISTER (HTX) .....	5-5
5.4	RECEIVE BYTE REGISTERS (RXH, RXL) .....	5-5
5.5	TRANSMIT BYTE REGISTERS (TXH, TXL) .....	5-5
5.6	HOST RECEIVE DATA REGISTER (HRX) .....	5-6
5.7	COMMAND VECTOR REGISTER (CVR) .....	5-7
5.8	HOST CONTROL REGISTER (HCR) .....	5-9
5.9	HOST STATUS REGISTER (HSR) .....	5-11
5.10	INTERRUPT CONTROL REGISTER (ICR) .....	5-12
5.11	INTERRUPT STATUS REGISTER (ISR) .....	5-15
5.12	INTERRUPT VECTOR REGISTER (IVR) .....	5-17
5.13	IVR HOST INTERFACE INTERRUPTS .....	5-18
5.14	DMA MODE OPERATION .....	5-18
5.15	HOST PORT USAGE – GENERAL CONSIDERATIONS .....	5-21

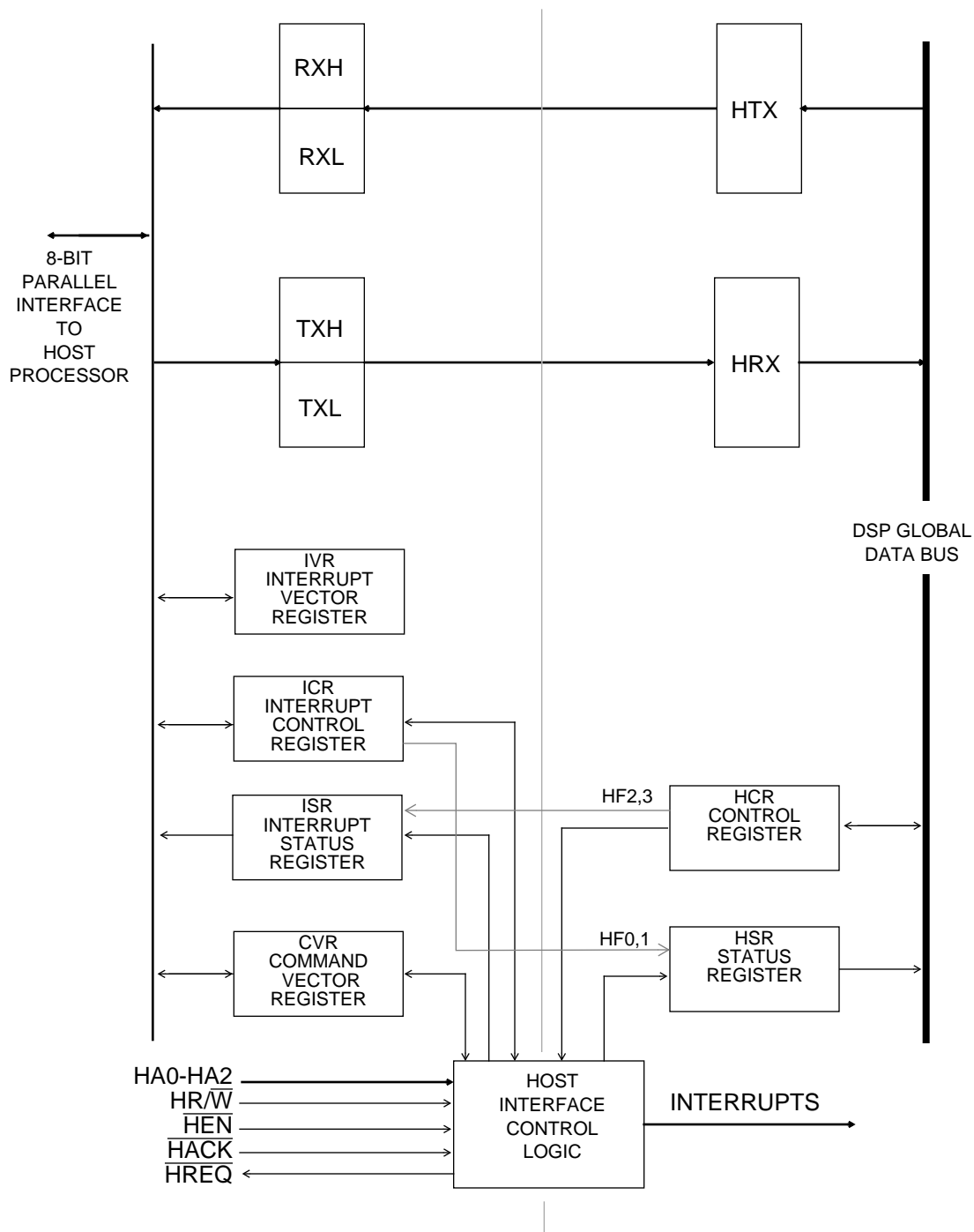
## 5.1. INTRODUCTION

The Host Interface (HI) is a byte-wide parallel slave port which may be connected directly to the data bus of a host processor. The host processor may be any of a number of popular microcomputers or microprocessors, another DSP, or DMA hardware. The DSP56166 has an 8-bit bidirectional data bus H0-H7 (PB0-PB7) and 7 control lines  $\overline{HR}/\overline{W}$ ,  $\overline{HEN}$ ,  $\overline{HREQ}$ , HA0-HA2, and  $\overline{HACK}$  (PB8-PB14) to control data transfers. The HI pin functions are described in Section 2. The HI appears as a memory mapped peripheral, occupying 8 bytes in the host processor's address space and three words in the DSP processor's address space. Figure 5-1 shows the HI block diagram. Separate transmit and receive data registers are double-buffered to allow the DSP56166 and host processor to efficiently transfer data at high speed. Host processor communication with the HI registers is accomplished using standard host processor instructions and addressing modes. Host processors may use byte move instructions to communicate with the HI registers. The host registers are addressed so that 8-bit MC6801-type host processors can use 16-bit load (LDD) and store (STD) instructions for data transfers. The 16-bit MC68000/10 host processor can address the HI using the special MOVEP instruction for word (16-bit) or long word (32-bit) transfers. The 32-bit MC68020 host processor can use its dynamic bus sizing feature to address the HI using standard MOVE word (16-bit) or long word (32-bit) instructions.

Handshake flags are provided for polled or interrupt-driven data transfers. The DSP56166 interrupt response is sufficiently fast that most host microprocessors can load or store data at their maximum programmed I/O (non-DMA) instruction rate without testing the handshake flags for each transfer. If the full handshake is not needed, the host processor can treat the DSP56166 as fast memory and data can be transferred between the host and DSP56166 at the fastest host processor rate. DMA hardware may be used with the external Host Request and Host Acknowledge pins to transfer data at the maximum DSP56166 interrupt rate.

The host processor can also issue vectored exception requests to the DSP56166 with the host command feature. The host may select any of the 64 DSP exception routines to be executed by writing a vector address register. This flexibility allows the host programmer to execute a wide variety of preprogrammed functions inside the DSP56166. Host exceptions can allow the host processor to read or write DSP56166 registers, data memory, or program memory locations and perform control and debugging operations if exception routines are implemented in the DSP to do these tasks.

The DSP5616 core views the HI as a memory mapped peripheral occupying three 16-bit words in data memory space. The DSP56166 may access the HI as a normal memory-mapped peripheral using standard polled or interrupt programming techniques.



**Figure 5-1 Host Interface Block Diagram**

## **5.2. HOST INTERFACE PROGRAMMING MODEL**

The HI has two programming models — one for the DSP56166 programmer and one for the host processor programmer. In most cases, the notation used in this manual reflects the DSP56166 perspective. The Host Interface — DSP56166 Programming Model is shown in Figure 5-2. The programming model register names on the DSP CPU side of the HI begin with the letter “H”. The Host Interface — Host Processor Programming Model is shown in Figure 5-3. The HI Interrupt Structure is shown in Table 5-2.

## **5.3. HOST TRANSMIT DATA REGISTER (HTX)**

The Host Transmit Register (HTX) is used for DSP to host processor data transfers. The HTX register is viewed as a 16-bit write-only register by the DSP. Writing the HTX register clears HTDE. The DSP may program the HTIE bit to cause a Host Transmit Data interrupt when HTDE is set. The HTX register is transferred as 16-bit data to the receive byte registers RXH:RXL if both the HTDE bit and the Receive Data Full, RXDF, status bit are cleared. This transfer operation sets RXDF and HTDE.

## **5.4. RECEIVE BYTE REGISTERS (RXH, RXL)**

The receive byte registers are viewed as two 8-bit read-only registers by the host processor called Receive High (RXH) and Receive Low (RXL). These two registers receive data from the high byte and low byte respectively of the Host Transmit Data register HTX and are selected by three external Host Address inputs HA2, HA1, and HA0 during a host processor read operation or by an on-chip address counter in DMA operations. The receive byte registers (at least RXL) contain valid data when the Receive Data Register Full RXDF bit is set. The host processor may program the RREQ bit to assert the external Host Request  $\overline{\text{HREQ}}$  pin when RXDF is set. This informs the host processor or DMA controller that the receive byte registers are full. These registers may be read in any order to transfer 8- or 16-bit data. However, reading RXL clears the Receive Data Full RXDF bit. Because reading RXL clears the RXDF status bit, RXL is normally the last register read during a 16-bit data transfer.

## **5.5. TRANSMIT BYTE REGISTERS (TXH, TXL)**

The transmit byte registers are viewed by the host processor as two 8-bit write-only registers called Transmit High (TXH) and Transmit Low (TXL). These two registers send data to the high byte and low byte respectively of the Host Receive Data register (HRX) and are selected by three external Host Address inputs HA2, HA1, and HA0 during a host processor write operation. Data may be written into the transmit byte registers when the Transmit Data Register Empty TXDE bit is set. The host processor may program the TREQ bit to assert the external Host Request  $\overline{\text{HREQ}}$  pin when TXDE is set. This informs



## HOST RECEIVE DATA REGISTER (HRX)

the host processor or DMA controller that the transmit byte registers are empty. These registers may be written in any order to transfer 8- or 16-bit data. However, writing the Transmit Low register TXL clears the TXDE bit. Because writing the TXL register clears the TXDE status bit, TXL is normally the last register written during a 16-bit data transfer. The transmit byte registers TXH:TXL are transferred as 16-bit data to the Host Receive Data Register HRX when both TXDE bit and the Host Receive Data Full, HRDF, bit are cleared. This transfer operation sets TXDE and HRDF

7	6	5	4	3	2	1	0	READ/WRITE HOST CONTROL REGISTER (HCR); ADDRESS X:\$FFC4	
*	*	*	HF3	HF2	HCIE	HTIE	HRIE		
7	6	5	4	3	2	1	0	READ-ONLY HOST STATUS REGISTER (HSR); ADDRESS X:\$FFE4	
DMA	*	*	HF1	HF0	HCP	HTDE	HRDF		
15			8	7				0	READ-ONLY HOST RECEIVE DATA REGISTER (HRX); ADDRESS X:\$FFE5
HIGH BYTE				LOW BYTE					
15			8	7				0	WRITE-ONLY HOST TRANSMIT DATA REGISTER (HTX); ADDRESS X:\$FFE5
HIGH BYTE				LOW BYTE					
ADDR(HEX)			DSP READ		DSP WRITE			HOST INTERFACE DSP ADDRESS MAP	
X:\$FFC4 X:\$FFE4 X:\$FFE5			HCR (8 BIT) HSR (8 BIT) HRX (16 BIT)		HCR (8 BIT) READ ONLY HTX (16 BIT)				

**Figure 5-2 Host Interface — DSP Programming Model**

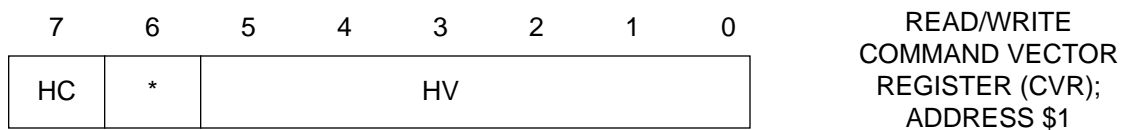
### 5.6. HOST RECEIVE DATA REGISTER (HRX)

The Host Receive Data register (HRX) is used for host processor to DSP data transfers. The HRX register is viewed as a 16-bit read-only register by the DSP. The HRX register is loaded with 16-bit data from the Transmit Data Registers TXH: TXL when both the

Transmit Data Register Empty, TXDE, and Host Receive Data Full, HRDF, bits are cleared. This transfer operation sets TXDE and HRDF. The HRX register contains valid data when the HRDF bit is set. Reading HRX clears HRDF. The DSP may program the HRIE bit to cause a Host Receive Data interrupt when HRDF is set.

### 5.7. COMMAND VECTOR REGISTER (CVR)

The CVR is used by the host to request vectored exception service from the DSP of any exception routine in the DSP. The Host Command feature is independent of any of the data transfer mechanisms in the HI but can be used to initialize the DSP for data transfer by triggering the appropriate preprogrammed software routine.



#### 5.7.1 CVR Host Vector (HV) Bits 0 through 4

The 5-bit Host Vector (HV) selects the host command exception address to access the host command exception routine. When the Host Command Exception is recognized by the DSP interrupt control logic, the starting address of the exception taken is  $2 \times HV$ . This allows the host processor to provide the exception starting address for the Host Command Exception. The host processor can select any of the 64 possible exception routine starting addresses in the DSP by writing the exception routine starting address (divided by 2) into HV. This means that the host processor can force any of the existing exception handlers (SSI, TIMER, IRQA, IRQB, etc.) and can use any of the reserved or otherwise unused starting addresses provided they are pre-programmed in the DSP. HV is set to \$16 (vector location \$002C) by DSP reset.

**CAUTION:**

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

#### 5.7.2 CVR Reserved Bits – Bits 5 and 6

Reserved bits are unused and are read by the host as zeros. Reserved bits should be written as zero for future compatibility.

## COMMAND VECTOR REGISTER (CVR)

7	6	5	4	3	2	1	0	READ/WRITE INTERRUPT CONTROL REGISTER (ICR); ADDRESS \$0	
INIT	HM1	HM0	HF1	HF0	*	TREQ	RREQ		
7	6	5	4	3	2	1	0	READ/WRITE COMMAND VECTOR REGISTER (CVR); ADDRESS \$1	
HC	*	HV							
7	6	5	4	3	2	1	0	READ-ONLY INTERRUPT STATUS REGISTER (ISR); ADDRESS \$2	
HREQ	DMA	*	HF3	HF2	TRDY	TXDE	RXDF		
7	6	5	4	3	2	1	0	READ/WRITE INTERRUPT VECTOR REGISTER (IVR); ADDRESS \$3	
IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0		
15				8	7			0	READ-ONLY RECEIVE BYTE REGISTERS; ADDRESSES \$6,7
RXH (ADDR \$6) HIGH BYTE				RXL (ADDR \$7) LOW BYTE					
15				8	7			0	WRITE-ONLY TRANSMIT BYTE REGISTERS; ADDRESSES \$6,7
TXH (ADDR \$6) HIGH BYTE				TXL (ADDR \$7) LOW BYTE					
ADDR (BIN)			HOST READ		HOST WRITE		HOST INTERFACE HOST PROCESSOR ADDRESS MAP		
HA2	HA1	HA0							
0	0	0	ICR(8 BIT)	ICR (8 BIT)					
0	0	1	CVR (8 BIT)	CVR (8 BIT)					
0	1	0	ISR (8 BIT)	not used					
0	1	1	IVR (8 BIT)	IVR (8 BIT)					
1	0	0	read zeros	not used					
1	0	1	reserved	reserved					
1	1	0	RXH (8 BIT)	TXH (8 BIT)					
1	1	1	RXL (8 BIT)	TXL (8 BIT)					

**Figure 5-3 Host Interface — Host Processor Programming Model**

**DSP INTERRUPT STRUCTURE**

INTERRUPT SOURCE	STATUS	MASK	EXCEPTION STARTING ADDRESS
RECEIVE DATA FULL TRANSMIT DATA EMPTY HOST COMMAND	HRDF HTDE HCP	HRIE HTIE HCIE	\$0028 \$002A 2*HV(\$0000-003E)

**HOST PROCESSOR  $\overline{\text{HREQ}}$  STRUCTURE**

$\overline{\text{HREQ}}$ SOURCE	STATUS	MASK
RECEIVE DATA FULL TRANSMIT DATA EMPTY	RXDF TXDE	RREQ TREQ

**Table 5-1 Host Interface Interrupt Structure**

### 5.7.3 CVR Host Command Bit (HC) Bit 7

The Host Command bit (HC) is used by the host to handshake the execution of host command exceptions. Normally the host processor sets HC=1 to request the host command exception from the DSP. When the host command exception is taken by the DSP, the HC bit is cleared by the HI hardware. The host processor can read the state of HC to determine when execution of the host command has started. The host processor may elect to clear the HC bit, cancelling the Host Command Exception request at any time before it is recognized by the DSP.

**CAUTION:**

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

Setting HC causes HCP (Host Command Pending) to be set in the HSR register. The host can write HC and HV in the same write cycle if desired. HC is cleared by DSP reset.

### 5.8. HOST CONTROL REGISTER (HCR)

The Host Control Register (HCR) is an 8-bit read/write control register used by the DSP to control the HI interrupts and flags. HCR cannot be accessed by the host processor. The HCR register occupies the low order byte of the internal data bus — the high order portion

is zero filled. HCR is a read/write register which can be accessed using bit manipulation instructions on control register bits. Any reserved bits are read as zeros and should be programmed as zeros for future compatibility. The contents of HCR are cleared on DSP reset. The control bits are described in the following paragraphs.

7	6	5	4	3	2	1	0	READ/WRITE HOST CONTROL REGISTER (HCR); ADDRESS X:\$FFC4
*	*	*	HF3	HF2	HCIE	HTIE	HRIE	

### 5.8.1 HCR Host Receive Interrupt Enable (HRIE) Bit 0

The Host Receive Interrupt Enable (HRIE) bit is used to enable a DSP interrupt when the Host Receive Data Full (HRDF) status bit in the Host Status register (HSR) is set. When HRIE is cleared, HRDF interrupts are disabled. When HRIE is set, a Host Receive Data interrupt request will occur if HRDF is set.

### 5.8.2 HCR Host Transmit Interrupt Enable (HTIE) Bit 1

The Host Transmit Interrupt Enable (HTIE) bit is used to enable a DSP interrupt when the Host Transmit Data Empty (HTDE) status bit in the Host Status Register (HSR) is set. When HTIE is cleared, HTDE interrupts are disabled. When HTIE is set, a Host Transmit Data interrupt request will occur if HTDE is set.

### 5.8.3 HCR Host Command Interrupt Enable (HCIE) Bit 2

The Host Command Interrupt Enable (HCIE) bit is used to enable a vectored DSP interrupt when the Host Command Pending (HCP) status bit in the Host Status Register (HSR) is set. When HCIE is cleared, HCP interrupts are disabled. When HCIE is set, a Host Command interrupt request will occur if HCP is set. The starting address of this interrupt is determined by the Host Vector (HV).

### 5.8.4 HCR Host Flag 2 (HF2) Bit 3

The Host Flag 2 (HF2) bit is used as a general purpose flag for DSP to host processor communication. Changing HF2 will change the Host Flag 2 (HF2) bit of the Interrupt Status Register ISR on the host processor side of the host interface. HF2 may be set or cleared by the DSP.

### 5.8.5 HCR Host Flag 3 (HF3) Bit 4

The Host Flag 3 (HF3) bit is used as a general purpose flag for DSP to host processor communication. Changing HF3 will change the Host Flag 3 (HF3) bit of the Interrupt Status Register ISR on the host processor side of the host interface. HF3 may be set or cleared by the DSP.

### 5.8.6 HCR Reserved Control – Bits 5, 6, and 7

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

## 5.9. HOST STATUS REGISTER (HSR)

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

7	6	5	4	3	2	1	0	
DMA	*	*	HF1	HF0	HCP	HTDE	HRDF	READ-ONLY HOST STATUS REGISTER (HSR) ADDRESS X:\$FFE4

### 5.9.1 HSR Host Receive Data Full (HRDF) Bit 0

The Host Receive Data Full (HRDF) bit indicates that the Host Receive Data register (HRX) contains data from the host processor. HRDF is set when data is transferred from the TXH:TXL registers to the HRX register. HRDF is cleared when the Receive Data register HRX is read by the DSP. HRDF can also be cleared by the host processor using the Initialize function. HRDF is also cleared by a DSP reset. This bit is typically used for polling operations.

### 5.9.2 HSR Host Transmit Data Empty (HTDE) Bit 1

The Host Transmit Data Empty (HTDE) bit indicates that the Host Transmit Data register (HTX) is empty and can be written by the DSP. HTDE is set when the HTX register is transferred to the RXH:RXL registers. HTDE is cleared when the Transmit Data register HTX is written by the DSP. HTDE can also be set by the host processor using the Initialize function. HTDE is also set by a DSP reset. This bit is typically used for polling operations.

### 5.9.3 HSR Host Command Pending (HCP) Bit 2

The Host Command Pending (HCP) bit indicates that the host processor has set the HC bit and that a Host Command Interrupt is pending. The HCP bit reflects the status of the HC bit in the Command Vector Register (CVR) on the host processor side of the host interface. HC and HCP are cleared by the DSP exception hardware when the exception is taken. The host processor can clear HC which also clears HCP. The HCP is cleared by DSP reset. This bit is typically used for polling operations.

#### 5.9.4 HSR Host Flag 0 (HF0) Bit 3

The Host Flag 0 (HF0) bit indicates the state of Host Flag 0 (HF0) in the Interrupt Control Register ICR on the host processor side of the host interface. HF0 can only be changed by the host processor. HF0 is cleared by a DSP reset.

#### 5.9.5 HSR Host Flag 1 (HF1) Bit 4

The Host Flag 1 (HF1) bit indicates the state of Host Flag 1 (HF1) in the Interrupt Control Register ICR on the host processor side of the host interface. HF1 can only be changed by the host processor. HF1 is cleared by a DSP reset.

#### 5.9.6 HSR Reserved Status – Bits 5 and 6

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

#### 5.9.7 HSR DMA Status (DMA) Bit 7

The DMA status bit (DMA) indicates that the host processor has enabled the DMA mode of the HI by setting HM1 or HM0 to a one. When the DMA status bit is a zero, it indicates that the DMA mode is disabled by the Host Mode bits HM0 and HM1 (both are cleared) in the Interrupt Control Register ICR and no DMA operations are pending. When the DMA status bit is set, the DMA mode is enabled by the Host Mode bits HM0 and HM1. The channel not in use (i.e., the transmit channel or receive channel) can be used for polled or interrupt operation by the DSP. DMA is cleared by a DSP reset.

### 5.10. INTERRUPT CONTROL REGISTER (ICR)

The Interrupt Control Register (ICR) is an 8-bit read/write control register used by the host processor to control the HI interrupts and flags. ICR cannot be accessed by the DSP. ICR is a read/write register which can be accessed using bit manipulation instructions on control register bits. The control bits are described in the following paragraphs.

7	6	5	4	3	2	1	0	READ/WRITE INTERRUPT CONTROL REGISTER (ICR) ADDRESS \$0
INIT	HM1	HM0	HF1	HF0	*	TREQ	RREQ	

#### 5.10.1 ICR Receive Request Enable (RREQ) Bit 0

The Receive Request enable (RREQ) bit is used to control the  $\overline{\text{HREQ}}$  pin for host receive data transfers. In the Interrupt Mode (DMA off), RREQ is used to enable interrupt requests via the external Host Request  $\overline{\text{HREQ}}$  pin when the Receive Data Register Full (RXDF) status bit in the Interrupt Status register (ISR) is set. When RREQ is cleared, RXDF interrupts are disabled. When RREQ is set, the external Host Request  $\overline{\text{HREQ}}$  pin will be asserted if RXDF is set.

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

### 5.10.2 ICR Transmit Request Enable (TREQ) Bit 1

The Transmit Request enable (TREQ) bit is used to control the  $\overline{\text{HREQ}}$  pin for host transmit data transfers. In the Interrupt Mode (DMA off), TREQ is used to enable interrupt requests via the external Host Request  $\overline{\text{HREQ}}$  pin when the Transmit Data Register Empty (TXDE) status bit in the Interrupt Status register (ISR) is set. When TREQ is cleared, TXDE interrupts are disabled. When TREQ is set, the external Host Request  $\overline{\text{HREQ}}$  pin will be asserted if TXDE is set.

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

Table 5-2 and Table 5-3 summarize the effect of RREQ and TREQ on the  $\overline{\text{HREQ}}$  pin. TREQ is cleared by DSP reset.

### 5.10.3 ICR Reserved bit – Bit 2

This bit is reserved and unused. It reads as a logic zero. Reserved bits should be written as zero for future compatibility.

**Table 5-2 HREQ Pin Definition - Interrupt Mode**

TREQ	RREQ	$\overline{\text{HREQ}}$ Pin
0	0	No Interrupts (Polling)
0	1	RXDF Request (Interrupt)
1	0	TXDE Request (Interrupt)
1	1	RXDF and TXDE Request (Interrupt)

**Table 5-3 HREQ Pin Definition - DMA Mode**

TREQ	RREQ	$\overline{\text{HREQ}}$ Pin
0	0	DMA Transfers Disabled
0	1	DSP→HOST Request (RX)
1	0	HOST→DSP Request (TX)
1	1	Undefined (Illegal)

### 5.10.4 ICR Host Flag 0 (HF0) Bit 3

The Host Flag 0 (HF0) bit is used as a general purpose flag for host processor to DSP communication. HF0 may be set or cleared by the host processor and cannot be changed by the DSP. Changing HF0 also changes the Host Flag bit 0 (HF0) of the Host Status register HSR on the DSP side of the HI. HF0 is cleared by DSP reset.



#### 5.10.5 ICR Host Flag 1 (HF1) Bit 4

The Host Flag 1 (HF1) bit is used as a general purpose flag for host processor to DSP communication. HF1 may be set or cleared by the host processor and cannot be changed by the DSP. Changing HF1 also changes the Host Flag bit 1 (HF1) of the Host Status register HSR on the DSP side of the HI. HF1 is cleared by a DSP reset.

#### 5.10.6 ICR Host Mode Control (HM1, HM0) Bits 5 and 6

The Host Mode control bits HM0 and HM1 select the transfer mode of the HI. HM1 and HM0 enable the DMA mode of operation or interrupt (non-DMA) mode of operation.

When the DMA mode is enabled, the  $\overline{\text{HREQ}}$  pin is used as a DMA transfer request output to a DMA controller and the  $\overline{\text{HACK}}$  pin is used as a DMA Transfer Acknowledge input from a DMA controller. The DMA Control bits HM0 and HM1 select the size of the DMA word to be transferred as shown in Table 5-4. The direction of the DMA transfer is selected by the TREQ and RREQ bits.

**Table 5-4 Host Mode (HM1, HM0) Bit Definition**

HM1	HM0	Mode
0	0	Interrupt Mode (DMA off)
0	1	Illegal
1	0	DMA mode; 16-bit
1	1	DMA mode; 8-bit

When both HM1 and HM0 are cleared, the DMA mode is disabled and the TREQ and RREQ control bits are used for host processor interrupting via the external Host Request  $\overline{\text{HREQ}}$  output pin. In the interrupt mode, the Host Acknowledge  $\overline{\text{HACK}}$  input pin is used for the MC68000 family vectored Interrupt Acknowledge input.

When HM1 or HM0 are set, the DMA mode is enabled and the  $\overline{\text{HREQ}}$  pin is not available for host processor interrupts. When the DMA mode is enabled, the TREQ and RREQ bits select the direction of DMA transfers; the Host Acknowledge  $\overline{\text{HACK}}$  input pin is used as a DMA Transfer Acknowledge input. If the DMA direction is from DSP to Host, the contents of the selected register are enabled onto the host data bus when  $\overline{\text{HACK}}$  is asserted. If the DMA direction is from Host to DSP, the selected register is written to TXH or TXL from the host data bus when  $\overline{\text{HACK}}$  is asserted. The size of the DMA word to be transferred is determined by the DMA control bits HM0 and HM1. The HI register selected during a DMA transfer is determined by a 2-bit address counter which is preloaded with the value in HM1 and HM0. The address counter substitutes for the Host Address bits HA1 and HA0 of the HI during a DMA transfer. The Host Address bit HA2 is forced to one during each DMA transfer. The address counter can be initialized with the INIT bit feature. After each DMA transfer on the host data bus, the address counter is incremented to the next register.

When the address counter reaches the highest register (RXL or TXL), the address counter is not incremented but is loaded with the value in HM1 and HM0. This allows 8- or 16-bit data to be transferred in a circular fashion and eliminates the need for the DMA controller to supply the Host Address HA2, HA1, and HA0 pins. For 16-bit data transfers, the DSP interrupt rate is reduced by a factor of two from the Host Request rate.

HM1 and HM0 are cleared by DSP reset.

## 5.10.7 ICR Initialize Bit (INIT) Bit 7

The INIT bit is used by the host to force initialization of the HI hardware. This may or may not be necessary, depending on the software design of the interface. The type of initialization done depends on the state of TREQ and RREQ. The INIT command is designed to conveniently convert into the desired data transfer mode after the INIT is completed. The commands are described below and in Table 5-5. The host sets INIT which causes the HI hardware to execute the command. The interface hardware clears INIT when the command is complete. INIT is cleared by DSP reset.

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

**Table 5-5 INIT Execution Definition**

TREQ	RREQ	After INIT Execution — Interrupt Mode (HM1=0, HM0=0)
0	0	INIT=0; "address counter = 00"
0	1	INIT=0; RXDF=0; HTDE=1; "address counter = 00"
1	0	INIT=0; TXDE=1; HRDF=0; "address counter = 00"
1	1	INIT=0; RXDF=0; HTDE=1; TXDE=1; HRDF=0; "address counter = 00"

**Table 5-5 INIT Execution Definition (Continued)**

TREQ	RREQ	After INIT Execution — DMA Mode (HM1 or HM0 = 1)
0	0	INIT=0; address counter = HM1,HM0
0	1	INIT=0; RXDF=0; HTDE=1; address counter = HM1,HM0
1	0	INIT=0; TXDE=1; HRDF=0; address counter = HM1,HM0
1	1	Undefined (illegal)

## 5.11. INTERRUPT STATUS REGISTER (ISR)

The Interrupt Status register (ISR) is an 8-bit read-only status register used by the host processor to interrogate the status and flags of the HI. The ISR can not be accessed by the DSP. The status bits are described in the following paragraphs.

7	6	5	4	3	2	1	0	READ-ONLY INTERRUPT STATUS REGISTER (ISR) ADDRESS \$2
HREQ	DMA	*	HF3	HF2	TRDY	TXDE	RXDF	

#### 5.11.1 ISR Receive Data Register Full (RXDF) Bit 0

The Receive Data Register Full (RXDF) bit indicates that both the receive byte registers, RXH and RXL, contain data from the DSP and may be read by the host processor. RXDF is set when the contents of the Host Transmit Data Register HTX is transferred to the receive byte registers RXH:RXL. RXDF is cleared when the Receive Data Low (RXL) register is read by the host processor. RXL is normally the last byte of the receive byte registers to be read by the host processor. RXDF can be cleared by the host processor using the Initialize function. RXDF is cleared by a DSP reset. RXDF may be used to assert the external Host Request  $\overline{\text{HREQ}}$  pin if the Receive Request enable RREQ bit is set. RXDF provides valid status regardless of whether the RXDF interrupt is enabled or not so that polling techniques may be used by the host processor.

#### 5.11.2 ISR Transmit Data Register Empty (TXDE) Bit 1

The Transmit Data Register Empty (TXDE) bit indicates that the transmit byte registers TXH and TXL are both empty and can be written by the host processor. TXDE is set when the content of the transmit byte registers TXH:TXL are transferred to the Host Receive Data Register (HRX). TXDE is cleared when the Transmit Byte Low (TXL) register is written by the host processor. TXL is normally the last byte of the transmit byte registers to be written by the host processor. TXDE can be set by the host processor using the Initialize feature. TXDE is set by a DSP reset. TXDE may be used to assert the external Host Request  $\overline{\text{HREQ}}$  pin if the Transmit Request Enable TREQ bit is set. TXDE provides valid status regardless of whether the TXDE interrupt is enabled or not so that polling techniques may be used by the host.

#### 5.11.3 ISR Transmitter Ready (TRDY) Bit 2

The Transmitter Ready (TRDY) status bit indicates that both the transmit byte registers and Host Receive Data register are empty, i.e., the channel from the host processor through the HI to the DSP CPU is clear. By testing TRDY, the host processor programmer can be assured that the first word received by the DSP will be the first word the host processor transmits.

$\text{TRDY} = \text{TXDE} \wedge \text{HRDF}$

The DSP reset will set TRDY.

#### **5.11.4 ISR Host Flag 2 (HF2) Bit 3**

The Host Flag 2 (HF2) bit indicates the state of Host Flag 2 (HF2) in the Host Control Register (HCR). HF2 can only be changed by the DSP. HF2 is cleared by a DSP reset.

#### **5.11.5 ISR Host Flag 3 (HF3) Bit 4**

The Host Flag 3 (HF3) bit indicates the state of Host Flag 3 (HF3) in the Host Control Register HCR. HF3 can only be changed by the DSP. HF3 is cleared by a DSP reset.

#### **5.11.6 ISR (Reserved Status) Bit 5**

This status bit is reserved for future expansion and will read as zero during host processor read operations. Reserved bits should be written as zero for future compatibility.

#### **5.11.7 ISR DMA Status (DMA) Bit 6**

The DMA status bit (DMA) indicates that the host processor has enabled the DMA mode of the HI (HM1 or HM0 =1). When the DMA status bit is clear, it indicates that the DMA mode is disabled by the Host Mode bits (HM0 and HM1) in the Interrupt Control Register ICR and no DMA operations are pending. When DMA is set, it indicates that the DMA mode is enabled and the host processor should not use the active DMA channel (RXH:RXL or TXH:TXL depending on DMA direction) to avoid conflicts with the DMA data transfers.

#### **5.11.8 ISR Host Request (HREQ) Bit 7**

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

### **5.12. INTERRUPT VECTOR REGISTER (IVR)**

The Interrupt Vector Register (IVR) is an 8-bit read/write register which contains the exception vector number for use with MC68000 processor family vectored interrupts. This register is accessible only to the host processor. The contents of the IVR register are placed on the host data bus, H0-H7, when  $\overline{\text{HREQ}}$  and  $\overline{\text{HACK}}$  pins both are asserted and the DMA mode is disabled. The content of this register is initialized to \$0F by a DSP reset. This corresponds to the un-initialized exception vector in the MC68000 family.

7	6	5	4	3	2	1	0	READ/WRITE INTERRUPT CONTROL REGISTER (IVR); ADDRESS \$3
IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0	

## 5.13. IVR HOST INTERFACE INTERRUPTS

The HI may request interrupt service from either the DSP or host processor. The DSP interrupts are internal and do not require the use of an external interrupt pin. The DSP acknowledges host interrupts by jumping to the appropriate interrupt service routine. The DSP interrupt service routine must read or write the appropriate HI register (e.g. clearing HRDF or HTDE) to clear the interrupt. In the case of Host Command interrupts, the interrupt acknowledge from the program controller will clear the pending interrupt condition.

The host processor interrupts are external and use the Host Request  $\overline{\text{HREQ}}$  pin.  $\overline{\text{HREQ}}$  is normally connected to the host processor maskable interrupt (IRQ) input. The host processor acknowledges host interrupts by executing an interrupt service routine. The MC68000 processor family will assert the  $\overline{\text{HACK}}$  pin to read the exception vector number from the Interrupt Vector Register (IVR) of the HI. The most significant bit (HREQ) of the Interrupt Status Register (ISR) may be tested to determine if the DSP is the interrupting device and the two least significant bits (RXDF and TXDE) may be tested to determine the interrupt source. The host processor interrupt service routine must read or write the appropriate HI register to clear the interrupt.

## 5.14. DMA MODE OPERATION

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

### 5.14.1 Host to DSP – Host Interface Action

The following procedure outlines the steps that the HI hardware takes to transfer DMA

data from the host data bus to DSP memory.

1. Assert the Host Request  $\overline{\text{HREQ}}$  output pin when the transmit byte registers TXH:TXL are empty (this always occurs in host to DSP DMA mode when TXDE=1).
2. Write the selected transmit byte register from the host data bus when the  $\overline{\text{HACK}}$  input pin is asserted by the DMA controller. Deassert the  $\overline{\text{HREQ}}$  pin.
3. If the highest register address has not been reached (i.e., TXDE=1), postincrement the DMA address counter to select the next register. Wait until  $\overline{\text{HACK}}$  is deasserted then go to step 1.
4. If the highest register address has been reached (i.e., TXDE=0), load the DMA address counter with the value in HM1 and HM0 and transfer the transmit byte registers TXH:TXL to the Host Receive Data Register HRX when HRDF=0. This will set HRDF=1. Wait until  $\overline{\text{HACK}}$  is deasserted then go to step 1.

**Notes:**

1. The DSP to host data transfers can occur normally in the channel not used for DMA except that the host must use polling and not interrupts.
2. The transfer of data from the TXH:TXL register to the HRX register automatically loads the DMA address counter from the HM1 and HM0 bits in the DMA host to DSP mode.

The host exception is triggered when HRDF=1. The host exception routine must read the Host Receive Data Register HRX to clear HRDF. The transfer from step 4 to step 1 will automatically occur if TXDE=1. Note that the execution of the host exception on HRDF=1 condition will occur after the transfer to step 1 and is independent of the handshake since it is only dependent on HRDF=1.

**5.14.2 Host to DSP – Host Processor Procedure**

The following procedure outlines the typical steps that the host processor must take to set-up and terminate a host to DSP DMA transfer.

1. Setup the external DMA controller source address, direction, byte count, and other control registers. Enable the DMA controller channel.
2. Set TXDE and clear HRDF. This can be done with the appropriate Initialize function. The host must also initialize the DMA counter in the HI using the initialize feature.
3. The DSP's destination pointer used in the DMA exception handler (an address register for example) must be initialized and HRIE must be set to enable the HRDF interrupt. This could be done with a separate Host Command exception routine in the DSP.  $\overline{\text{HREQ}}$  output pin will be asserted immediately by the DSP hardware which begins the DMA transfer.

4. Perform other tasks until interrupted by the DMA controller DMA complete interrupt. The DSP Interrupt Control Register (ICR), the Interrupt Status Register (ISR), and RXH:RXL may be accessed at any time by the host processor (using HA0-HA2,  $\overline{\text{HR}}/\overline{\text{W}}$ , and  $\overline{\text{H}}\overline{\text{EN}}$ ) but the transmit byte registers (TXH:TXL) may not be accessed until the DMA mode is disabled.
5. Terminate the DMA controller channel to disable DMA transfers.
6. Terminate the DSP HI DMA mode by clearing the HM1 and HM0 bits and clearing TREQ in the Interrupt Control Register (ICR).

#### **5.14.3 DSP to Host Interface Action**

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

1. The transmit exception will be triggered when HTIE=1 and HTDE=1. The exception routine software will write the data word into HTX.
2. Transfer the HTX register to the receive byte registers RXH:RXL when they are empty (RXDF=0). This will automatically occur. Load the DMA address counter from HM1 and HM0. This action will set HTDE=1 and trigger another DSP transmit exception to write HTX (i.e., HTDE=0).
3. Assert the Host Request  $\overline{\text{HREQ}}$  pin when the receive byte registers are full.
4. Enable the selected Receive Byte register on the host data bus when  $\overline{\text{HACK}}$  is asserted. Deassert the Host Request  $\overline{\text{HREQ}}$  pin.
5. If the highest register address has not been reached (i.e., RXDF=1), postincrement the DMA address counter to select the next register. Wait until  $\overline{\text{HACK}}$  is deasserted, then go to step 3.
6. If the highest register address has been reached (i.e., RXDF=0), wait until  $\overline{\text{HACK}}$  is deasserted then go to step 2. The DSP transmit exception must have written HTX (i.e., HTDE=0) before Step 2 will be executed.

#### **Notes:**

1. The HOST→ DSP data transfers can occur normally in the channel not used for DMA except that the host must use polling and not interrupts.
2. The transfer of data from the HTX register to the RXH:RXL registers automatically loads the DMA address counter from the HM1 and HM0 bits when in DMA DSP to Host mode.

#### **5.14.4 DSP to Host – Host Processor Procedure**

The following procedure outlines the typical steps that the host processor must take to set-up and terminate a DSP to host DMA transfer.

1. Setup the DMA controller destination address, direction, byte count, and other control registers. Enable the DMA controller channel.
2. Set HTDE and clear RXDF. This can be done with the appropriate INIT function.
3. The DSP's source pointer used in the DMA exception handler (an address register for example) must be initialized and HTIE must be set to enable the DSP host transmit interrupt. This could be done by the host with a Host Command exception routine. The DSP host transmit exception will be activated immediately by DSP hardware which begins the DMA transfer.
4. Perform other tasks until interrupted by the DMA controller DMA complete interrupt. The DSP Interrupt Control Register (ICR), the Interrupt Status Register (ISR), and TXH:TXL may be accessed at any time by the host processor (using HA0-HA2,  $\overline{\text{HR}}/\overline{\text{W}}$ , and  $\overline{\text{H\overline{EN}}}$ ) but the receive byte registers (RXH and RXL) may not be accessed until the DMA mode is disabled.
5. Terminate the DMA controller channel to disable DMA transfers.
6. Terminate the DSP HI DMA mode by clearing the HM1 and HM0 bits and clearing RREQ in the Interrupt Control Register (ICR).

#### **5.15. HOST PORT USAGE – GENERAL CONSIDERATIONS**

Careful synchronization is required when reading multi-bit registers that are written by another asynchronous system. This is a common problem when two asynchronous systems are connected. The situation exists in the host port. However, if the port is used in the way it was designed, proper operation is guaranteed. The considerations for proper operation are discussed below.

##### **5.15.1 Host Programmer Considerations**

1. Unsynchronized Reading of receive byte registers.

When reading receive byte registers, RXH or RXL, the host programmer should use interrupts or poll the RXDF flag which indicates that data is available. This guarantees that the data in the receive byte registers will be stable.

2. Overwriting transmit byte registers.



The host programmer should not write to the transmit byte registers, TXH or TXL, unless the TXDE bit is set, indicating that the transmit byte registers are empty. This guarantees that the DSP will read stable data when it reads the HRX register.

### 3. Synchronization of Status Bits from DSP to Host.

HC, HREQ, DMA, HF3, HF2, TRDY, TXDE, and RXDF status bits are set or cleared from inside the DSP and read by the host processor. The host can read these status bits very quickly without regard to the clock rate used by the DSP but there is a chance that the state of the bit could be changing during the read operation. This is generally not a system problem since the bit will be read correctly in the next pass of any host polling routine. However, if the host holds the  $\overline{\text{H\!E\!N}}$  input pin for the minimum assert time plus 1.5 Ccyc, the status data is guaranteed to be stable. The 1.5 Ccyc is first used to synchronize the  $\overline{\text{H\!E\!N}}$  signal and then to block internal updates of the status bits. There is no other minimum  $\overline{\text{H\!E\!N}}$  assert time relationship to the DSP clocks. There is a minimum  $\overline{\text{H\!E\!N}}$  deassert time of 1.5 Ccyc so that the blocking latch can be deasserted to allow updates if the host is in a tight polling loop. This only applies to reading status bits.

The only potential system problem with the uncertainty of reading any status bits by the host is HF3 and HF2 as an encoded pair. For example, if the DSP changes HF3 and HF2 from “00” to “11”, there is a very small probability that the host could read the bits during the transition and receive “01” or “10” instead of “11”. If the combination of HF3 and HF2 has significance, the host would potentially read the wrong combination.

Solutions:

- a. Read the bits twice and check for consensus.
- b. Assert  $\overline{\text{H\!E\!N}}$  access for  $\overline{\text{H\!E\!N}}$  + 1.5 Ccyc so that status bit transitions are stabilized.

### 4. Overwriting the Host Vector

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

### 5. Cancelling a pending Host Command Exception

The host processor may elect to clear the HC bit to cancel the Host Command Exception request at any time before it is recognized by the DSP. The DSP may execute the host exception after the HC bit is cleared: (1) because the host does not know exactly when the exception will be recognized, (2) because of synchronization, and (3) because of exception processing pipelining. As a result, the HV must not be changed at

the same time the HC bit is cleared. In this way, if the exception was taken, the vector will be known.

### 5.15.2 DSP Programmer Considerations

1. Reading HF1 and HF0 as an encoded pair.

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

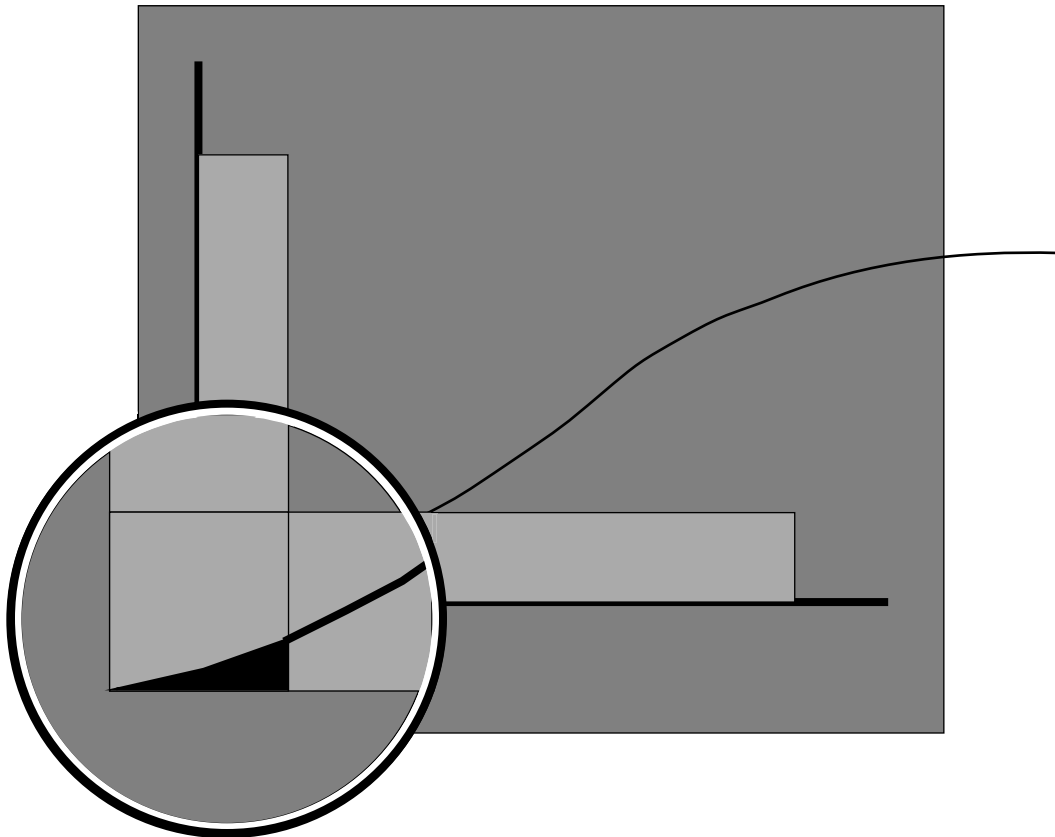
The only system problem with reading status is HF1 and HF0 if they are encoded as a pair, e.g. the four combinations 00, 01, 10, and 11 each have significance. This is because there is a very small probability that the DSP will read the status bits that were synchronized during transition. The solution to this potential problem is to read the bits twice for consensus.



---

## SECTION 6

# DSP56166 ON-CHIP SIGMA/DELTA CODEC



## SECTION CONTENTS

---

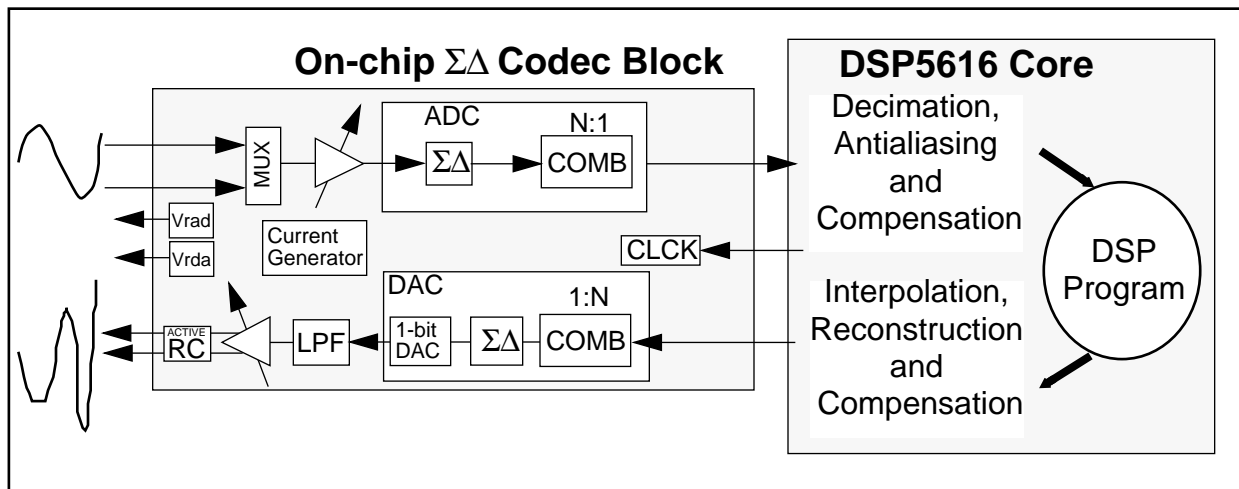
6.1	GENERAL DESCRIPTION .....	6-3
6.2	CODEC BLOCK DIAGRAM .....	6-4
6.3	ANALOG I/O DEFINITION .....	6-5
6.4	INTERFACE WITH THE DSP5616 CORE .....	6-6
6.4.1.	Interface Definition .....	6-6
6.4.2.	On-chip Codec Programming Model .....	6-7
6.4.3.	Codec Receive Register CRX .....	6-8
6.4.4.	Codec Transmit Register CTX .....	6-8
6.4.5.	Codec Control Register CCR0 .....	6-8
6.4.6.	Codec Control Register CCR1 .....	6-11
6.4.7.	Codec Status Register COSR .....	6-13
6.5	ON-CHIP CODEC GAIN AND FREQUENCY RESPONSE ANALYSIS .....	6-16
6.5.1.	DC Gain and Frequency Response of the A/D Section .....	6-16
6.5.2.	DC Gain and Frequency Response of the D/A Section .....	6-21

## 6.1 INTRODUCTION

This section describes the DSP56166 Sigma-Delta ( $\Sigma\Delta$ ) over sampled voice band codec block. It discusses the general block diagram of the A/D and D/A sections, the handshake between the DSP5616 core and the codec, as well as the last decimation antialiasing filter and first interpolation reconstruction filter performed in software by the DSP5616 core.

## 6.2 GENERAL DESCRIPTION

The  $\Sigma\Delta$  oversampled voice band codec block is built using HCMOS technology and utilizes switched capacitor technology in some circuits. The codec contains one A/D converter and one D/A converter. It also contains two reference voltage generators, a current bias generator, and a master clock circuit (see Figure 6-1).



**Figure 6-1 DSP56166 On-chip  $\Sigma\Delta$  Functional Diagram**

The A/D converter consists of an analog  $\Sigma\Delta$  modulator with selectable input gain, a digital low-pass comb filter, and a parallel bus interface. The analog modulator is a second-order  $\Sigma\Delta$  loop implemented using fully differential CMOS switched capacitor circuitry. The analog modulator input is selectable from one of two pins. The analog modulator output is the input to a third-order digital comb filter which provides low-pass filtering and decimation. The final 16-bit result is output through a parallel interface to the global data bus of the DSP5616 core.

The D/A converter consists of a second-order comb interpolating filter, a digital  $\Sigma\Delta$  modulator, a 1-bit DAC, a two-pole Butterworth low pass filter, a selectable attenuator, and an active RC low pass output stage. The interpolator takes in 16-bit two's complement numbers from the DSP5616 core and upsamples them to a high frequency. The modulator

changes these 16-bit words into a 1-bit stream. The 1-bit DAC converts this 1-bit stream into  $\pm 2$  volt differential analog levels. The output of the DAC is filtered by the switched capacitor Butterworth filter which attenuates the out-of-band shaped modulator noise. The selectable attenuator provides volume control in increments of 5 dB per step. The active RC low pass filter provides reconstruction filtering and driver capabilities.

This  $\Sigma\Delta$  codec block has been designed for maximum flexibility. The user can select any value between 65 and 128 as the decimation (interpolation) ratio for the A/D (D/A) converters. Operating at its nominal sampling rate of 2 MHz, the A/D converter provides a 16-bit digital output with more than 60 dB S/(N+D) for input signals. The D/A converter nominally takes in a 16-bit word at a 16 KHz rate, and has a fully differential analog output which provides more than of 60 dB S/(N+D). Table 6-1 summarizes the main features of the codec block.

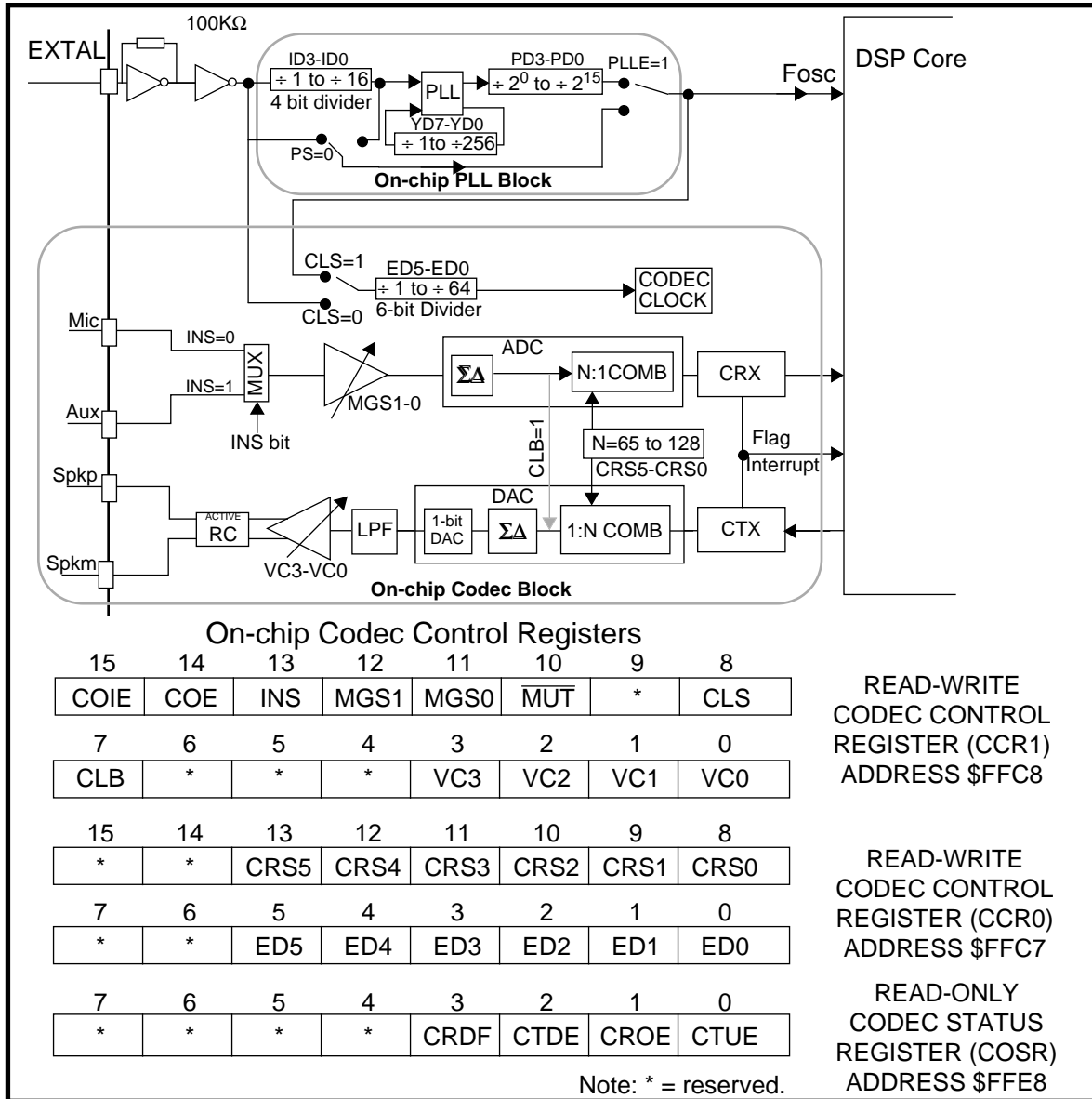
**Table 6-1 On-chip Codec Main Features**

- 16-bit resolution
- Dynamic Range of 80 dB
- More than 60 dB S/(N+D)
- Operates at clock rates between 100 KHz and 3 MHz
- No off-chip component required
- Internal voltage reference (2/5 of positive power supply)
- Low power HCMOS

The last decimation filtering stage of the A/D section as well as the first interpolation filtering stage of the D/A section is implemented by software in the 16-bit DSP core in order to reduce the codec cell die area.

### 6.3 CODEC BLOCK DIAGRAM

A general block diagram of the DSP56166 codec and its programming model can be seen in Figure 6-2.



**Figure 6-2 DSP56166 Codec General Block Diagram**

## 6.4 ANALOG I/O DEFINITION

This section describes the Motorola DSP56166 Codec analog input and output characteristics (see Figure 6-3).

There are two analog inputs, MIC and AUX. Selection between MIC or AUX is made via one control bit (INS bit) and can be changed any time desired. The electrical specifications of the two pins are identical.



The analog output consists of a fully differential driver stage, with each output having an operating range of  $V_{rda} = 1.0 \text{ V}_p$ . The output op-amp is capable of driving a load of  $1 \text{ k}\Omega$  in series with  $50 \text{ nF}$  between the differential outputs.

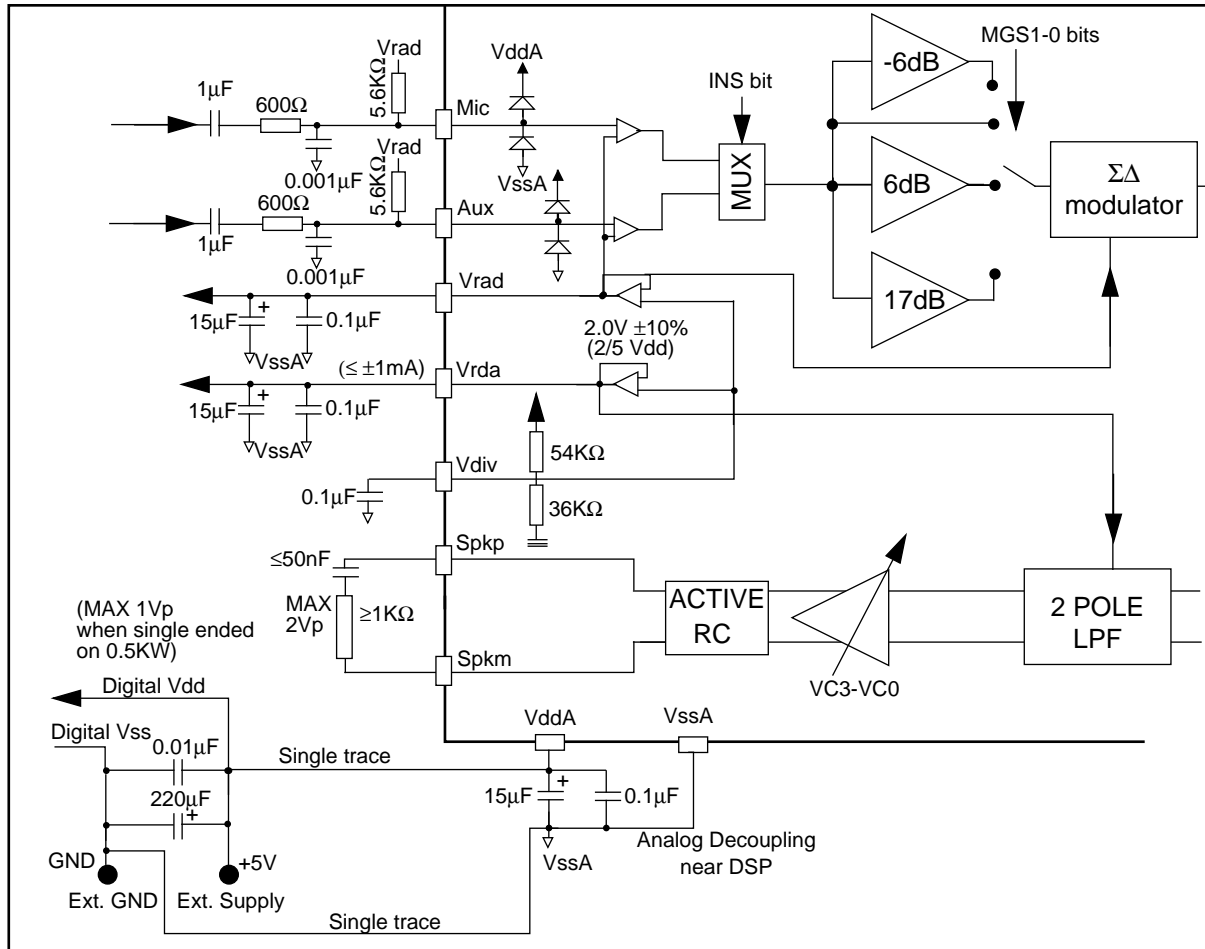


Figure 6-3 DSP56166 Codec Analog Input and Output Diagram

The reference voltages for the A/D and D/A converters are generated by two on-chip voltage references. Current bias for the opamps in the analog blocks are set by the on-chip current bias generator.

## 6.5 INTERFACE WITH THE DSP5616 CORE

This section discusses the use of each bit in the codec control registers.

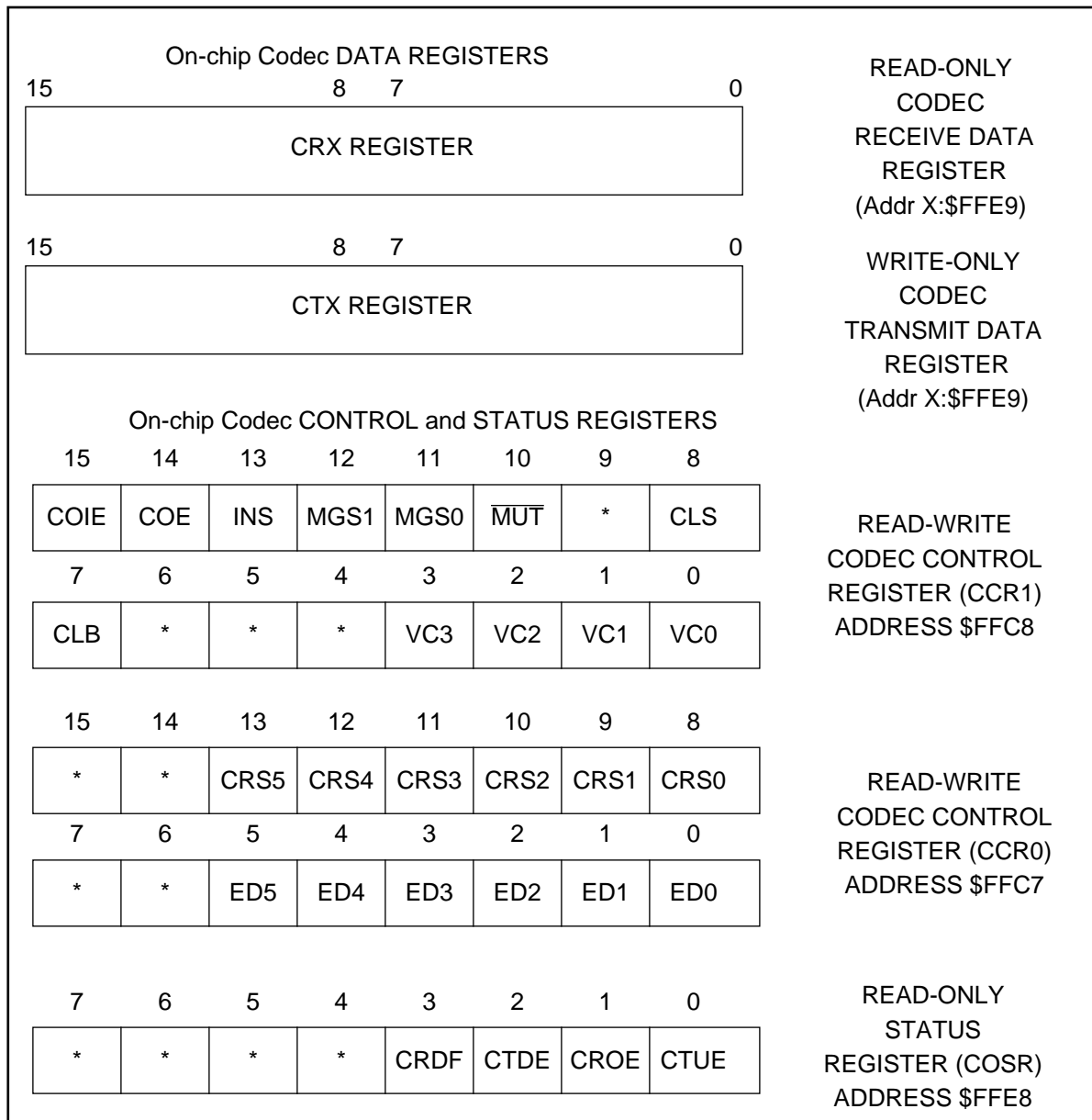
### 6.5.1 Interface Definition

The  $\Sigma\Delta$  Codec is seen from the core as a memory mapped on-chip peripheral. Data memory locations are dedicated for the receive data register, transmit data register, status register, and control registers. One interrupt vector is assigned to the  $\Sigma\Delta$  Codec.

The A/D section (receive) and the D/A section (transmit) are synchronous; that is, a common interrupt vector is used by the two sections to notify the DSP core that an input sample is to be read and/or that an output sample is to be written.

### 6.5.2 On-chip Codec Programming Model

Figure 6-4 shows the memory mapped registers used with the on-chip codec. There are five registers mapped into four memory locations.



\* - reserved bits, read as zero, should be written with zero for future compatibility.

**Figure 6-4 On-chip Codec Programming Model**

## 6.5.3 Codec Receive Register CRX

The CRX Codec Receive register is used for A/D to DSP core data transfers. The CRX register is viewed as a 16-bit read-only register by the DSP core. The CRX register is loaded with 16-bit data from the A/D section comb filter output. This transfer operation sets the CRDF bit in the codec status register COSR. Reading CRX clears CRDF. CRDF assertion will generate an interrupt if the COIE bit is set by the user.

## 6.5.4 Codec Transmit Register CTX

The CTX Codec Transmit register is used for DSP core to D/A data transfers. The CTX register is viewed as a 16-bit write-only register by the DSP core. Writing the CTX register clears the CTDE bit in the codec status register COSR. CTDE assertion will generate an interrupt if the COIE bit is set by the user.

## 6.5.5 Codec Control Register CCR0

The Codec Control register CCR0 is a 16-bit read/write register used to direct the on-chip codec operation. The CCR0 controls the clocking scheme and decimation/interpolation ratio of the  $\Sigma\Delta$  codec. The CCR0 bits are described in the following sections.

All of the CCR0 bits are cleared by DSP hardware and software reset.

### 6.5.5.1 CCR0 Input Divider Bits (ED5-ED0) Bits 0-5

The six input divider bits are used to divide the input clock to the codec by any number between 1 and 64. If ED is the value contained in the six bits, the input clock is divided by ED+1.

Care should be taken to remain in the Codec operating range between 100KHz and 3 MHz.

**Note:** When the CLS bit is set in the CCR1 register (PLL output selected as input clock), the value programmed in the ED divider should be greater than 1 (minimum 2) for proper codec operation.

### 6.5.5.2 CCR0 Codec Ratio Select Bits (CRS5-0) Bits 13-8

The Codec Ratio Select bits are used by the DSP core to program the decimation and interpolation ratio of the codec comb filter. The ratio values available are given in Figure 6-13.

Table 6-2 Decimation/Interpolation Ratio Control

CRS5-CRS0	Decimation Interpolation Ratio Rate	A/D Comb Filter DC Gain		D/A Comb Filter DC Gain	
\$00	65	$4(65^3)/2^{21}$	-5.617 dB	65/128	-5.886 dB
\$01	66	$4(66^3)/2^{21}$	-5.219 dB	66/128	-5.753 dB
\$02	67	$4(67^3)/2^{21}$	-4.827 dB	67/128	-5.622 dB
\$03	68	$4(68^3)/2^{21}$	-4.441 dB	68/128	-5.494 dB
\$04	69	$4(69^3)/2^{21}$	-4.060 dB	69/128	-5.367 dB
\$05	70	$4(70^3)/2^{21}$	-3.686 dB	70/128	-5.242 dB
\$06	71	$4(71^3)/2^{21}$	-3.316 dB	71/128	-5.119 dB
\$07	72	$4(72^3)/2^{21}$	-2.951 dB	72/128	-4.998 dB
\$08	73	$4(73^3)/2^{21}$	-2.592 dB	73/128	-4.877 dB
\$09	74	$4(74^3)/2^{21}$	-2.237 dB	74/128	-4.759 dB
\$0A	75	$4(75^3)/2^{21}$	-1.887 dB	75/128	-4.643 dB
\$0B	76	$4(76^3)/2^{21}$	-1.542 dB	76/128	-4.527 dB
\$0C	77	$4(77^3)/2^{21}$	-1.202 dB	77/128	-4.414 dB
\$0D	78	$4(78^3)/2^{21}$	-0.866 dB	78/128	-4.302 dB
\$0E	79	$4(79^3)/2^{21}$	-0.534 dB	79/128	-4.192 dB
\$0F	80	$4(80^3)/2^{21}$	-0.206 dB	80/128	-4.082 dB
\$10	81	$4(81^3)/2^{21}$	0.118 dB	81/128	-3.974 dB
\$11	82	$4(82^3)/2^{21}$	0.437 dB	82/128	-3.868 dB
\$12	83	$4(83^3)/2^{21}$	0.753 dB	83/128	-3.763 dB
\$13	84	$4(84^3)/2^{21}$	1.065 dB	84/128	-3.659 dB
\$14	85	$4(85^3)/2^{21}$	1.374 dB	85/128	-3.558 dB
\$15	86	$4(86^3)/2^{21}$	1.678 dB	86/128	-3.454 dB
\$16	87	$4(87^3)/2^{21}$	1.980 dB	87/128	-3.354 dB
\$17	88	$4(88^3)/2^{21}$	2.277 dB	88/128	-3.254 dB
\$08	72	$125^3/2^{21}$	-0.618 dB	125/128	-0.206 dB
\$09	73	1	0 dB	1	0 dB
\$0A	74	$2(105^3)/2^{21}$	0.859 dB	105/128	-1.720 dB
\$0B	75	$2(81^3)/2^{21}$	0.118 dB	81/128	-3.974 dB
\$0C	76	$125^3/2^{21}$	-0.618 dB	125/128	-0.206 dB
\$0D	77	1	0 dB	1	0 dB
\$0E	78	$2(105^3)/2^{21}$	0.859 dB	105/128	-1.720 dB
\$0F	79	$2(81^3)/2^{21}$	0.118 dB	81/128	-3.974 dB

Table 6-2 Decimation/Interpolation Ratio Control - continued

CRS5-CRS0	Decimation Interpolation Ratio Rate	A/D Comb Filter DC Gain		D/A Comb Filter DC Gain	
\$18	89	$2(89^3)/2^{21}$	-3.449 dB	89/128	-3.156 dB
\$19	90	$2(90^3)/2^{21}$	-3.157 dB	90/128	-3.059 dB
\$1A	91	$2(91^3)/2^{21}$	-2.869 dB	91/128	-2.963 dB
\$1B	92	$2(92^3)/2^{21}$	-2.584 dB	92/128	-2.868 dB
\$1C	93	$2(93^3)/2^{21}$	-2.303 dB	93/128	-2.774 dB
\$1D	94	$2(94^3)/2^{21}$	-2.024 dB	94/128	-2.682 dB
\$1E	95	$2(95^3)/2^{21}$	-1.748 dB	95/128	-2.590 dB
\$1F	96	$2(96^3)/2^{21}$	-1.476 dB	96/128	-2.499 dB
\$20	97	$2(97^3)/2^{21}$	-1.206 dB	97/128	-2.409 dB
\$21	98	$2(98^3)/2^{21}$	-0.938 dB	98/128	-2.320 dB
\$22	99	$2(99^3)/2^{21}$	-0.674 dB	99/128	-2.231 dB
\$23	100	$2(100^3)/2^{21}$	-0.412 dB	100/128	-2.144 dB
\$24	101	$2(101^3)/2^{21}$	-0.153 dB	101/128	-2.058 dB
\$25	102	$2(102^3)/2^{21}$	0.104 dB	102/128	-1.972 dB
\$26	103	$2(103^3)/2^{21}$	0.358 dB	103/128	-1.887 dB
\$27	104	$2(104^3)/2^{21}$	0.610 dB	104/128	-1.804 dB
\$28	105	$2(105^3)/2^{21}$	0.859 dB	105/128	-1.720 dB
\$29	106	$2(106^3)/2^{21}$	1.106 dB	106/128	-1.638 dB
\$2A	107	$2(107^3)/2^{21}$	1.351 dB	107/128	-1.556 dB
\$2B	108	$2(108^3)/2^{21}$	1.593 dB	108/128	-1.476 dB
\$2C	109	$2(109^3)/2^{21}$	1.833 dB	109/128	-1.396 dB
\$2D	110	$2(110^3)/2^{21}$	2.072 dB	110/128	-1.316 dB
\$2E	111	$2(111^3)/2^{21}$	2.307 dB	111/128	-1.238 dB
\$2F	112	$2(112^3)/2^{21}$	2.541 dB	112/128	-1.160 dB
\$30	113	$(113^3)/2^{21}$	-3.248 dB	113/128	-1.083 dB
\$31	114	$(114^3)/2^{21}$	-3.018 dB	114/128	-1.006 dB
\$32	115	$(115^3)/2^{21}$	-2.791 dB	115/128	-0.930 dB
\$33	116	$(116^3)/2^{21}$	-2.565 dB	116/128	-0.855 dB
\$34	117	$(117^3)/2^{21}$	-2.341 dB	117/128	-0.780 dB
\$35	118	$(118^3)/2^{21}$	-2.120 dB	118/128	-0.707 dB
\$36	119	$(119^3)/2^{21}$	-1.820 dB	119/128	-0.633 dB
\$37	120	$(120^3)/2^{21}$	-1.682 dB	120/128	-0.561 dB
\$38	121	$(121^3)/2^{21}$	-1.465 dB	121/128	-0.488 dB
\$39	122	$(122^3)/2^{21}$	-1.251 dB	122/128	-0.417 dB
\$3A	123	$(123^3)/2^{21}$	-1.039 dB	123/128	-0.346 dB
\$3B	124	$(124^3)/2^{21}$	-0.827 dB	124/128	-0.258 dB
\$3C	125	$(125^3)/2^{21}$	-0.618 dB	125/128	-0.206 dB
\$3D	126	$(126^3)/2^{21}$	-0.410 dB	126/128	-0.137 dB
\$3E	127	$(127^3)/2^{21}$	-0.204 dB	127/128	-0.068 dB
\$3F	128	1	0 dB	1	0 dB

As shown in Figure 6-13, the value selected as decimation and interpolation ratio also affects the DC gain of the comb filter in the A/D and D/A sections. The global DC gain of the A/D and D/A sections is discussed in detail in Section 6.6.

### 6.5.5.3 CCR0 Reserved Bits 6-7 and 14-15

These bits are reserved. They should be written as zero by the user program to insure future compatibility.

### 6.5.6 Codec Control Register CCR1

The Codec Control Register CCR1 is a 16-bit read/write register used to direct the operation of the on-chip codec. The CCR1 controls the receive and transmit audio gains, the codec clocking source, the selection of the analog input, the muting of the analog output, the codec power down, the codec enable, and the codec interrupt enable. The CCR1 bits are described in the following sections.

All of the CCR1 bits are cleared by DSP hardware and software reset.

#### 6.5.6.1 CCR1 Audio Level Control Bits (VC3-VC0) Bits 0-3

Audio gain control is employed in the last stage of the on-chip codec D/A section. Bits VC0-VC3 control the volume between -15 dB and 40 dB as shown in Figure 6-13.

**Table 6-3 Audio Level Control**

VC3	VC2	VC1	VC0	Relative Level in dB
0	0	0	0	-15
0	0	0	1	-10
0	0	1	0	-5
0	0	1	1	0
0	1	0	0	5
0	1	0	1	11
0	1	1	0	17
0	1	1	1	23
1	0	0	0	5
1	0	0	1	11
1	0	1	0	17
1	0	1	1	23
1	1	0	0	29
1	1	0	1	35
1	1	1	0	35
1	1	1	1	40

A 16-bit full scale positive value of \$7FFF written to the D/A should produce a voltage level equal to  $V_{rda} = +1V_p$  singled-ended and  $V_{rda} = +2V_p$  differential at the output when the volume control bits VC3-VC0 are set to the level of 0 dB and when the DC gain of the D/A comb filter is unity (decimation ratio of 128).

The digital reconstruction-interpolation filter performed by the DSP core can also be used to control the output audio level in conjunction with the four VC3-VC0 bits. The gain of this filter can be adjusted in order to modify the relative level and the step between levels. Table 6-4 gives an example where the gain of the interpolation digital filter is adjusted in order to provide output volume control between -15dB and +40dB in 5dB steps.

Table 6-4 Audio Level Control with DSP Filter Gain

VC3	VC2	VC1	VC0	Relative Level dB	Digital Filter Gain	Final Output Level
0	0	0	0	-15	0	-15
0	0	0	1	-10	0	-10
0	0	1	0	-5	0	-5
0	0	1	1	0	0	0
0	1	0	0	5	0	5
0	1	0	1	11	-1	10
0	1	1	0	17	-2	15
0	1	1	1	23	-3	20
1	0	0	0	5	0	5
1	0	0	1	11	-1	10
1	0	1	0	17	-2	15
1	0	1	1	23	-3	20
1	1	0	0	29	-4	25
1	1	0	1	35	-5	30
1	1	1	0	35	0	35
1	1	1	1	40	0	40

#### 6.5.6.2 CCR1 Codec Loop Back Bit (CLB) Bit 7

The Codec Loop Back Bit selects the input to the analog back end of the DAC. This bit is cleared for normal DAC operation. When CLB is set, the output of the A/D  $\Sigma\Delta$  analog modulator is used as input to the analog back end of the DAC, as shown in Figure 6-2.

#### 6.5.6.3 CCR1 Clock Select Bit (CLS) Bit 8

This bit is used to select the source of the codec clock. When the CLS bit is cleared, the squared version of Fext is selected as the codec clock input. When this bit is set, the codec input clock is derived from the PLL output, as shown in Figure 6-2.

**Note:** When CLS is set, the value programmed in the ED divider should be greater than one (minimum two) for proper codec operation.

#### 6.5.6.4 CCR1 Mute Bit ( $\overline{\text{MUT}}$ ) Bit 10

The mute bit is used to mute the output signal. When the  $\overline{\text{MUT}}$  bit is cleared, the output signal is muted. When the  $\overline{\text{MUT}}$  bit is set, the output signal is not muted.

#### 6.5.6.5 CCR1 Microphone Gain Select Bits (MGS1-0) Bits 11 and 12

The Microphone Gain Select Bits are used by the DSP core to program the analog input gain. The values are given in Table 6-5. The analog modulator is guaranteed to be linear up to 3 dB below full scale saturation values. The full scale saturation analog value results in a maximum digital A/D output (\$7FFF) when the A/D comb filter has a DC unity gain.

**Table 6-5 Microphone Gain Control**

MGS1	MGS0	Gain		Modulator full scale	Full scale linearity
		Actual	dB		
0	0	0.5	-6	2 Vp	1.414 Vp
0	1	1	0	1 Vp	0.707 Vp
1	0	2	6	500 mVp	354 mVp
1	1	7.07	17	141 mVp	100 mVp

#### 6.5.6.6 CCR1 Input Select Bit (INS) Bit 13

The input select bit is used by the DSP to select between the two inputs — MIC and AUX. When INS is cleared, MIC is selected and when INS is set, the AUX input is selected.

#### 6.5.6.7 CCR1 Codec Enable Bit (COE) Bit 14

The Codec Enable Bit enables the on-chip codec section. When this bit is cleared, the section is disabled and put in the power down mode. Setting the bit wakes-up and enables the on-chip codec section.

#### 6.5.6.8 CCR1 Codec Interrupt Enable Bit (COIE) Bit 15

The Codec Interrupt Enable Bit enables the on-chip codec interrupt. When this bit is cleared the interrupt is disabled. Setting the bit enables the interrupt.

#### 6.5.6.9 CCR1 Reserved Bits 4,5,6, and 9

These bits are reserved. They should be written as zero by the user program to insure future compatibility.

### 6.5.7 Codec Status Register COSR

The Codec Status Register COSR is an 8-bit read-only status register used by the DSP to interrogate the status and flags of the on-chip codec. The status bits and flag bits are described in the following paragraphs.

#### 6.5.7.1 COSR Codec Transmit Underrun Error FFlag Bit (CTUE) Bit 0

The Codec Transmit Underflow Error Flag Bit is set when a sample has to be transmitted to the codec section while the DSP has not yet written to the CTX transmit register (underrun error). In this case, the previous sample written to the CTX register is re-transmitted to the D/A section.

Hardware or software reset and the STOP instruction clear CTUE. CTUE is also cleared by reading the COSR with CTUE set followed by writing CTX. Clearing the COE bit in the Codec Control Register CCR1 does not affect CTUE.



#### 6.5.7.2 COSR Codec Receive Overrun Error Flag Bit (CROE) Bit 1

The Codec Receive Overrun Error Flag bit is set when a new sample is received from the codec section while the previous received sample in the CRX receive register has not been read by the DSP (overrun error). In this case, the previous received sample is overwritten in the CRX register.

Hardware or software reset and the STOP instruction clear CROE. CROE is also cleared by reading the COSR with CROE set followed by reading CRX. Clearing the COE bit in the Codec Control Register CCR1 does not affect CTUE.

#### 6.5.7.3 COSR Codec Transmit Data Empty Bit (CTDE) Bit 2

The Codec Transmit Data Empty (CTDE) bit indicates that the D/A Transmit register CTX is empty and can be written by the DSP. CTDE is set when the CTX register is transferred to the D/A comb filter input. CTDE is cleared when the CTX register is written by the DSP. CTDE is also set entering the codec power down mode (COE cleared) and by a DSP reset (Hardware  $\overline{\text{RESET}}$  or RESET instruction) and the STOP instruction.

#### 6.5.7.4 COSR Codec Receive Data Full Bit (CRDF) Bit 3

The Codec Receive Data Full (CRDF) bit indicates that the A/D Data Receive register CRX contains data from the codec A/D section. CRDF is set when data is transferred from the A/D comb filter output to the CRX register. CRDF is cleared when the CRX Register is read by the DSP. CRDF is also cleared entering the Codec power down mode (COE cleared), by a DSP reset (Hardware  $\overline{\text{RESET}}$  or RESET instruction) and the STOP instruction.

#### 6.5.7.5 COSR Reserved Bits 4-15

These bits are reserved. They will be read as zeros by the user program.

On-chip Codec Status Registers (COSR X:\$FFE8)															
		7	6	5	4	3	2	1	0						
		*	*	*	*	CRDF	CTDE	CROE	CTUE						
CRDF (read-only)	0	Data From A/D not received in CRX													
	1	Data From A/D received in CRX													
CTDE (read-only)	0	Data in CTX has not been transferred to D/A													
	1	CTX empty													
CROE (read-only)	0	No Receive Overrun Error													
	1	Receive Overrun Error													
CTUE (read-only)	0	No Transmit Underrun Error													
	1	Transmit Underrun Error													

On-chip Codec Control (CCR1) X:\$FFC8															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

COIE	0	Codec interrupt disabled														
	1	Codec interrupt enabled														
COE	0	Codec disabled and put in power down														
	1	Codec enabled														
INS	0	MIC pin selected as A/D input														
	1	AUX pin selected as A/D input														
MGS1-0 Gain Select	00	MIC or AUX amplifier gain of -6dB								10	MIC or AUX amplifier gain of 6dB					
	01	MIC or AUX amplifier gain of 0dB								11	MIC or AUX amplifier gain of 17dB					
MUT	0	Speaker output muted														
	1	Speaker output active														
CLS	0	Squared Fext selected as codec input clock														
	1	PLL block output selected as codec input clock														
CLB	0	Normal Codec operation														
	1	The input of the D/A digital $\Sigma\Delta$ modulator is the output of the A/D $\Sigma\Delta$ modulator														
VC3VC0 D/A output Gain in dB	\$0	-15 dB Output volume gain								\$8	5 dB Output volume gain					
	\$1	-10 dB Output volume gain								\$9	11 dB Output volume gain					
	\$2	-5 dB Output volume gain								\$A	17 dB Output volume gain					
	\$3	0 dB Output volume gain								\$B	23 dB Output volume gain					
	\$4	5 dB Output volume gain								\$C	29 dB Output volume gain					
	\$5	11 dB Output volume gain								\$D	35 dB Output volume gain					
	\$6	17 dB Output volume gain								\$E	35 dB Output volume gain					
	\$7	23 dB Output volume gain								\$F	40 dB Output volume gain					

On-chip Codec Control (CCR0) X:\$FFC7															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CRS0-5	\$0-\$3F	Continuous ratio from 65 to 128 (see Figure 6-13)				ED0-5	\$0-\$3F	Codec Clock Division from 1 to 64			
--------	----------	--	--	--	--	-------	----------	--------------------------------------	--	--	--

**Figure 6-5 On-Chip Codec Programming Model Summary**

## 6.6 ON-CHIP CODEC GAIN AND FREQUENCY RESPONSE ANALYSIS

This section discusses the DC gain and the frequency response of the A/D and D/A blocks as a function of the decimation and interpolation ratios.

### 6.6.1 DC Gain and Frequency Response of the A/D Section

The DC gain and the frequency response of the A/D comb filter depends on the decimation rates selected by programming the six bits CRS5-CRS0 in the codec control register CCR0. The decimation rate can take any values between 65 and 128. A negative DC gain is introduced by scaling inside the comb filter in order to avoid any signal clipping caused by a positive DC gain of the comb filter itself. The A/D comb filter gains are given in Figure 6-13.

The digital output level out of the A/D section can be normalized to 0 dB or to any other value by changing the gain of the last decimation/antialiasing filter performed by the DSP core program. In addition to filtering and decimating, this last digital filter can also compensate the frequency response of the A/D comb filter.

The comb filter is a linear phase filter constructed as a cascade of digital integrators and differentiators which realizes the transfer function and frequency response of Figure 6-13.

$$H(z) = \frac{c}{128^3} \left[ \frac{1 - z^{-D}}{1 - z^{-1}} \right]^3$$

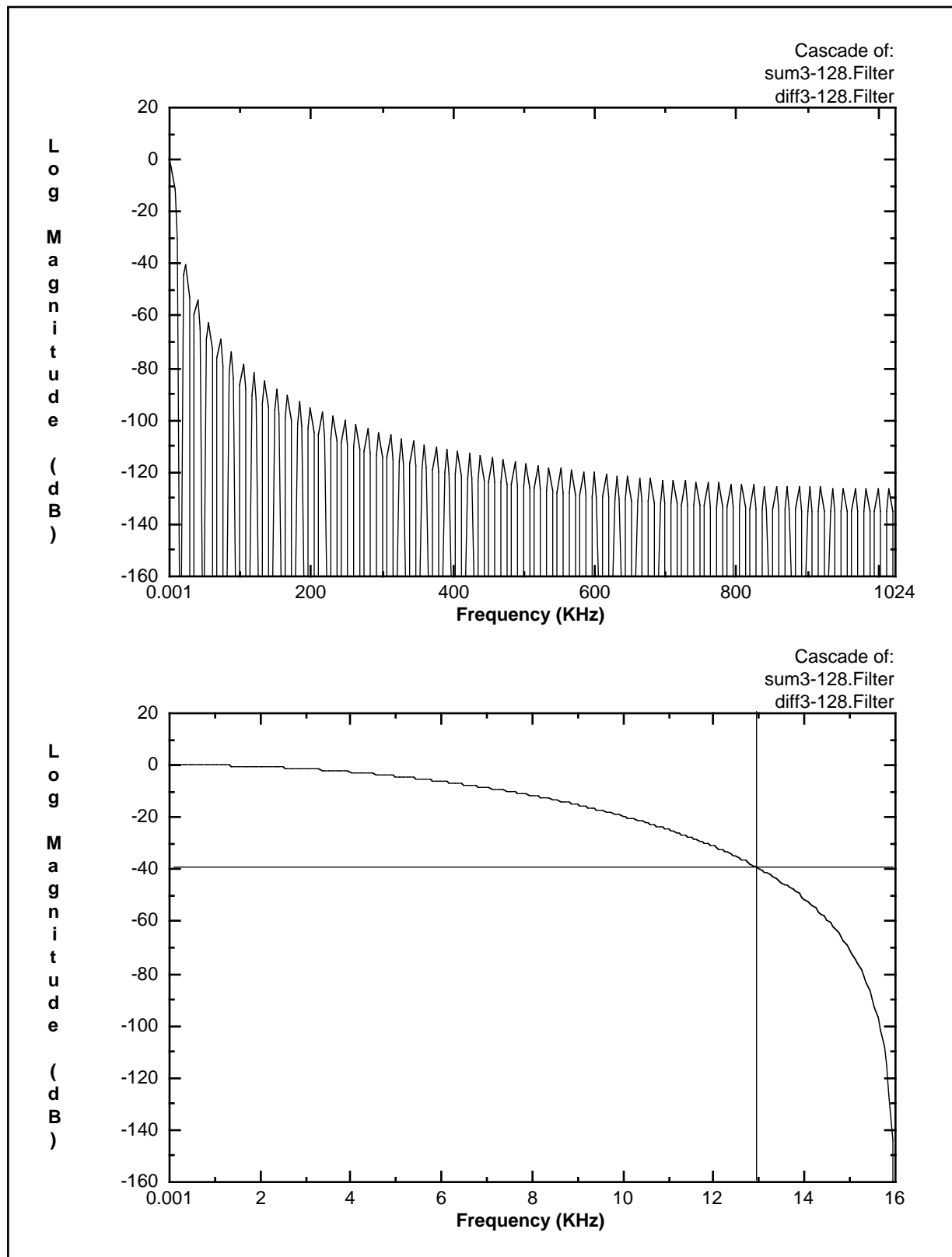
$$F(f) = \frac{c}{128^3} \left( \frac{\sin\left(\frac{2\pi D}{2F} \times f\right)}{\sin\left(\frac{2\pi}{2F} \times f\right)} \right)^3$$

$c=4$  for  $D=[65,88]$   
 $c=2$  for  $D=[89,112]$   
 $c=1$  for  $D=[112,128]$   
 $D$ : decimation ratio  
 $F$ :  $\Sigma\Delta$  modulator clock

**Figure 6-6 A/D Comb Filter Transfer Function**

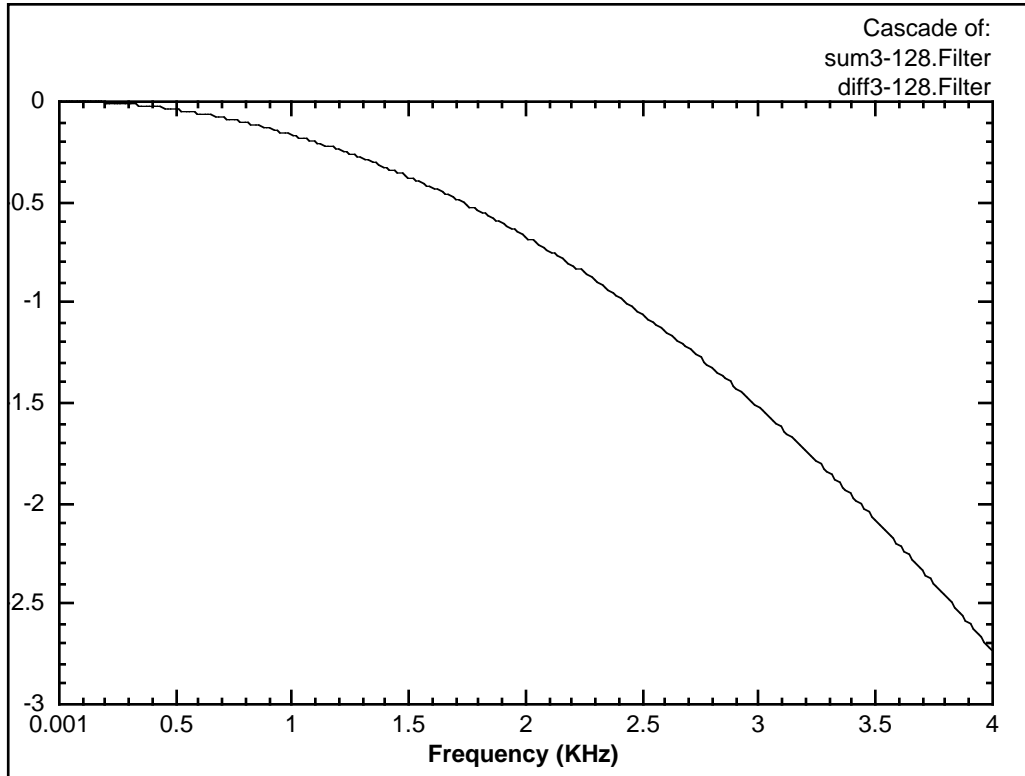
The cubic power is due to the order. The comb filter is ideal for decimation applications because it has zeros at the final output clock rate and all its multiples.

Figure 6-7 and Figure 6-8 show an example of the log magnitude response of the A/D comb filter using a master clock of 2.048 MHz and a decimation ratio of  $D=128$ .



**Figure 6-7 Log Magnitude Frequency Response of the A/D Comb Filter for F=2.048 MHz and D=128**

These figures show the frequency response in the 0-1.024 MHz band and also the frequency response after decimation in the 0-16KHz and 0-4KHz bands.



**Figure 6-8 Log Magnitude Frequency Response of the A/D Comb Filter in the Band 0-4KHz for F=2.048 MHz and D=128**

The output of the decimating comb filter is a 2's complement 16-bit word at the decimated rate which goes to the DSP core. It is hard limited to the range \$8000-\$7FFF to prevent roll over error. Figure 6-8 gives the frequency response of the A/D section in the 0-4KHZ band. It can be seen that this response is not flat in the band. The 3 dB drop-off can be compensated by the decimation filter inside the DSP core. The transfer function of the A/D comb is given on Figure 6-13

It is equal to:

$$F(f) = \frac{c}{128^3} \left( \frac{\sin\left(\frac{2\pi D}{2F} \times f\right)}{\sin\left(\frac{2\pi}{2F} \times f\right)} \right)^3$$

In the present example, D=128, c=1 and F=2.048 MHz, which gives:

$$F(f) = \frac{1}{(128)^3} \left( \frac{\sin\left(\frac{2\pi \times 128}{2 \times 2.048} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 2.048} \times f\right)} \right)^3$$

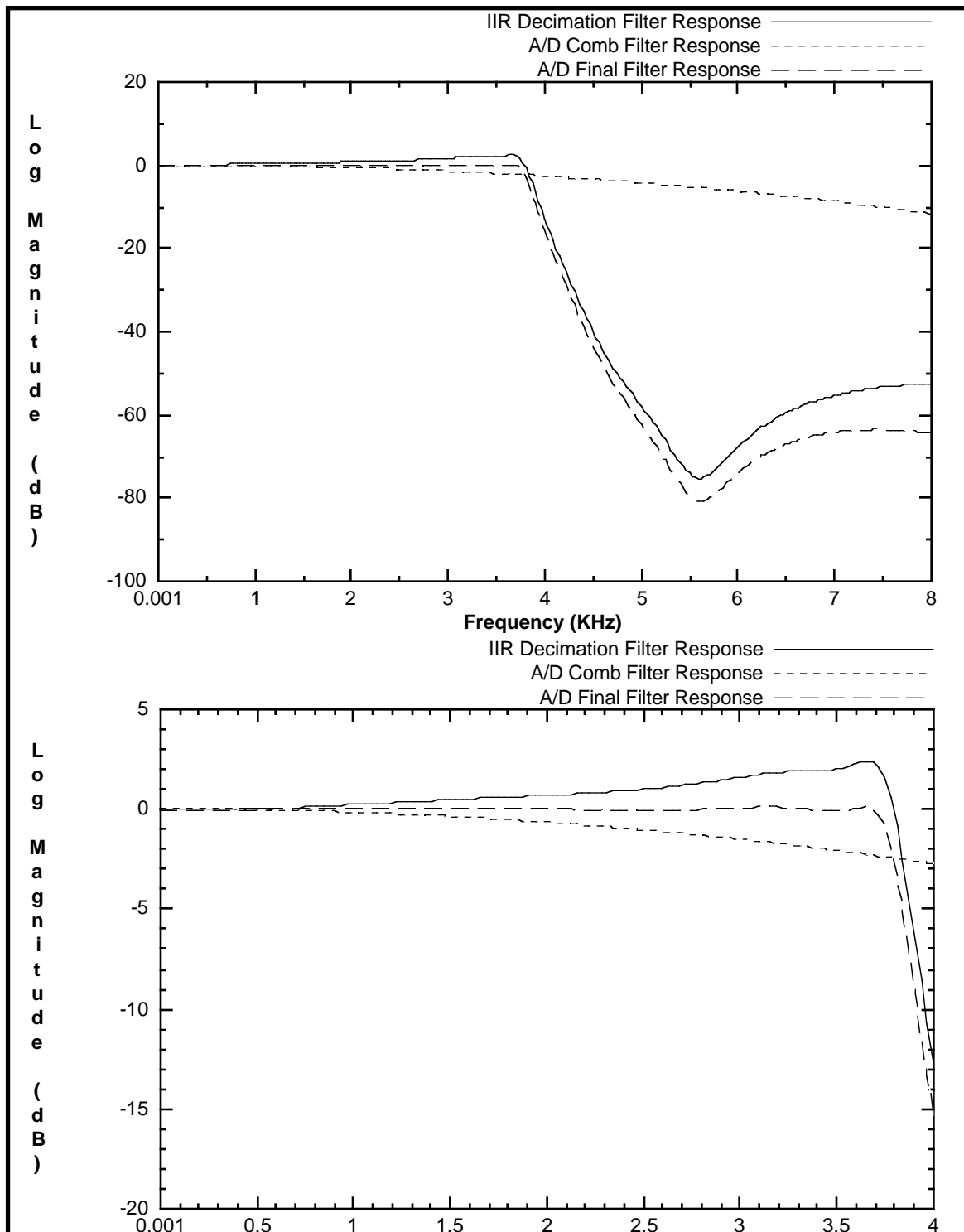
The transfer function of the decimation filter will have to be shaped by 1/F(f) in order to flatten the response of the A/D section.

Table 6-6 gives a 4 biquad IIR low pass filter who's transfer function has been shaped accordingly.

Figure 6-9 shows the frequency response of the filter and the effects on the overall response of the D/A section.

**Table 6-6 Example of a Four Biquad IIR Decimation and Compensation Filter**

; Source filter file: "adcomp128.IIR.Filter"			
_filter_type	equ	BIQUAD_FILTER_TYPE	
_NSTAGES	equ	4	; number of stages
	dc	\$02b2	; gain = 0.04211689868/2
	; Biquad stage no. 1		
	dc	-\$1237	; d2_1 = 0.284627003/2
	dc	\$e93c	; d1_1 = -0.3557032662/2
	dc	\$316a	; n2_1 = 0.7720730807/2
	dc	\$4186	; n1_1 = 1.023796768/2
	; Biquad stage no. 2		
	dc	-\$28f8	; d2_2 = 0.6401634264/2
	dc	\$e8df	; d1_2 = -0.3613604173/2
	dc	\$0d0c	; n2_2 = 0.2038857781/2
	dc	\$12dc	; n1_2 = 0.2946510536/2
	; Biquad stage no. 3		
	dc	-\$0c66	; d2_3 = 0.1937047354/2
	dc	\$d60d	; d1_3 = -0.6554721197/2
	dc	\$3454	; n2_3 = 0.8176134701/2
	dc	\$4328	; n1_3 = 1.049344022/2
	; Biquad stage no. 4		
	dc	-\$3951	; d2_4 = 0.8955455145/2
	dc	\$f4e3	; d1_4 = -0.1736521771/2
	dc	\$2ed3	; n2_4 = 0.7316442217/2
	dc	\$1b24	; n1_4 = 0.42404578/2
<b>NOTE:</b> This filter, as well as all the figures representing filter responses, has been generated using ZOLA Technologies, Inc., DSP Designer™ software package.			



**Figure 6-9 IIR Decimation and A/D Section Log Magnitude Frequency Response for F=2.048 MHz and D=128**

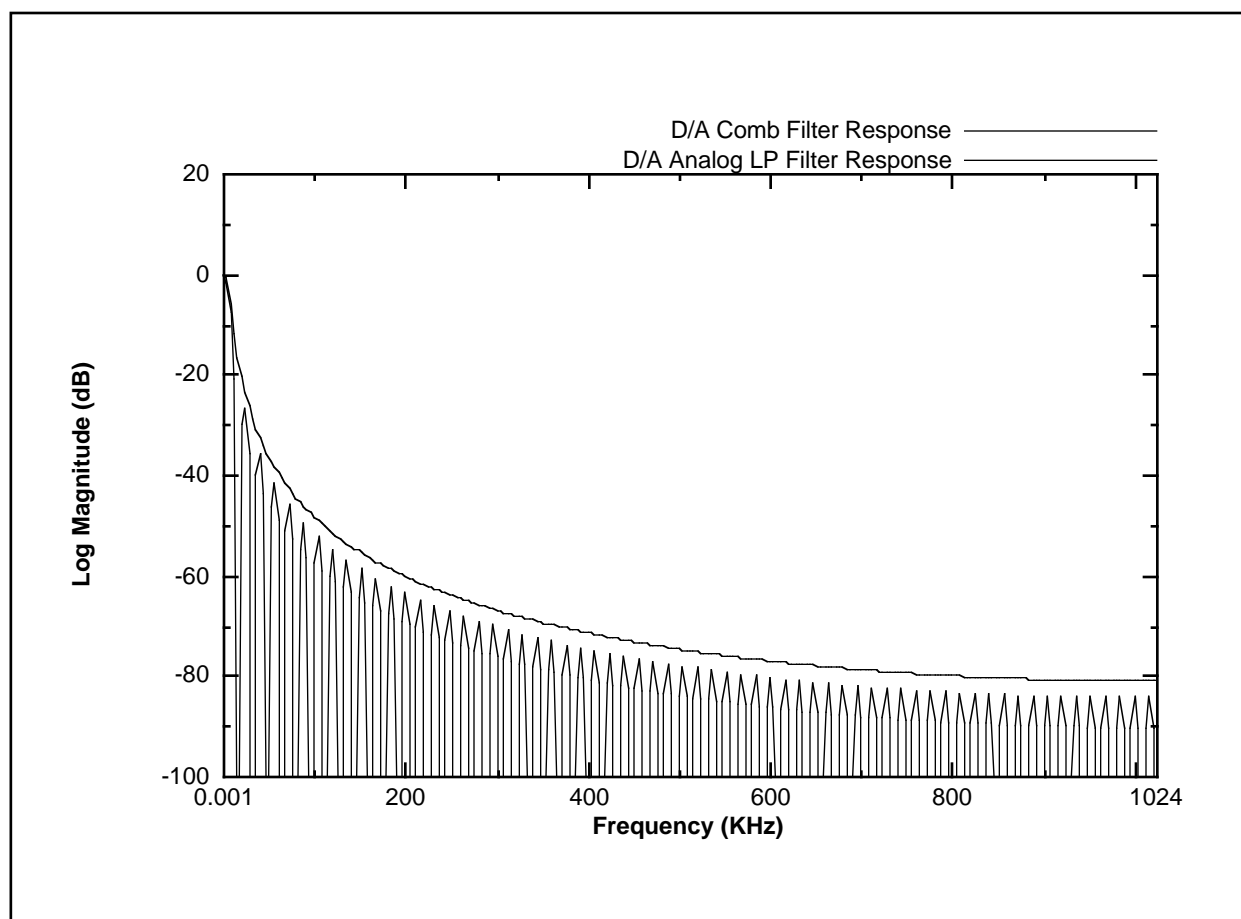
### 6.6.2 DC Gain and Frequency Response of the D/A Section

The DC gain and the frequency response of the D/A section depend on the interpolation rates selected by programming the six bits CRS5-CRS0 in the codec control register CCR0. The interpolation rate can take any value between 65 and 128.

The D/A comb filter gains are given in Figure 6-13.

The D/A section of the on-chip codec is composed of three subsections which are described below.

The frequency responses of the different sections of the D/A are illustrated in Figure 6-10.



**Figure 6-10 Log Magnitude Frequency Responses of the Three Sections of the D/A F=2048 MHz and D=128**

#### 6.6.2.1 D/A Second Order Digital Interpolation Comb Filter

The parallel interface of the D/A section receives a 2's complement 16-bit word from the DSP core at the initial data rate. The interpolating comb filter is a second order digital



comb filter. It is a linear phase filter that provides interpolation and anti-imaging on the input to the digital modulator. The comb filter is constructed as a sample and hold stage followed by a cascade of a digital differentiator and integrator which realizes the transfer function and the frequency response given in Figure 6-11.

$$H(z) = \frac{1}{128D} \left[ \frac{1 - z^{-D}}{1 - z^{-1}} \right]^2$$

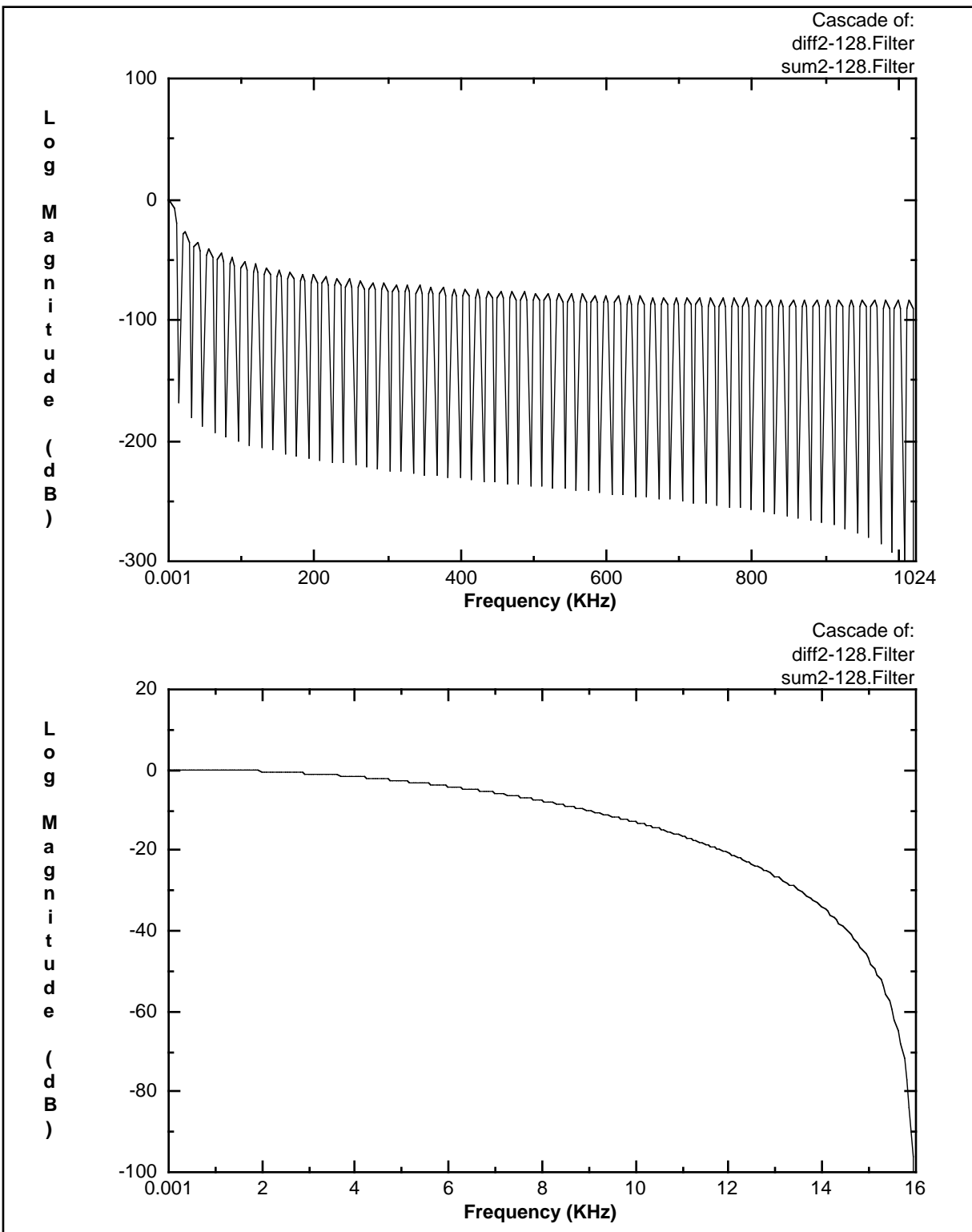
$$F(f) = \frac{1}{128D} \left( \frac{\sin\left(\frac{2\pi D}{2F} \times f\right)}{\sin\left(\frac{2\pi}{2F} \times f\right)} \right)^2$$

D: interpolation ratio  
F: comb filter integrator clock

**Figure 6-11 D/A Comb Filter Transfer Function**

The power of two is due to the order of the filter. The comb filter is ideal for interpolation applications because it has zeros at the initial data rate and all its multiples.

Figure 6-12 shows an example of the log magnitude response of the D/A digital comb filter using a master clock of 2.048 MHz and a decimation ratio of D=128. The figure shows the frequency response of the comb filter in the 0-1024KHz band and also after decimation (0-16KHz).



**Figure 6-12 Log Magnitude Frequency Response of the D/A Comb Filter for F=2.048MHz and D=128**

### 6.6.2.2 D/A Digital Modulator

The digital modulator is a second-order sigma-delta loop. It is made up of two digital integrators, a digital comparator, and a hard limit circuit. The output of the modulator is a 1-bit stream which contains the input signal plus a large amount of quantization noise which is shaped away from the baseband and towards the high frequencies. This 1-bit stream is then filtered by the analog low pass filter.

When the Codec Loop Back bit (CLB) of CCR1 is set, the input for the digital modulator is the output of the analog  $\Sigma\Delta$  modulator of the A/D section.

### 6.6.2.3 D/A Butterworth Analog Low Pass Filter

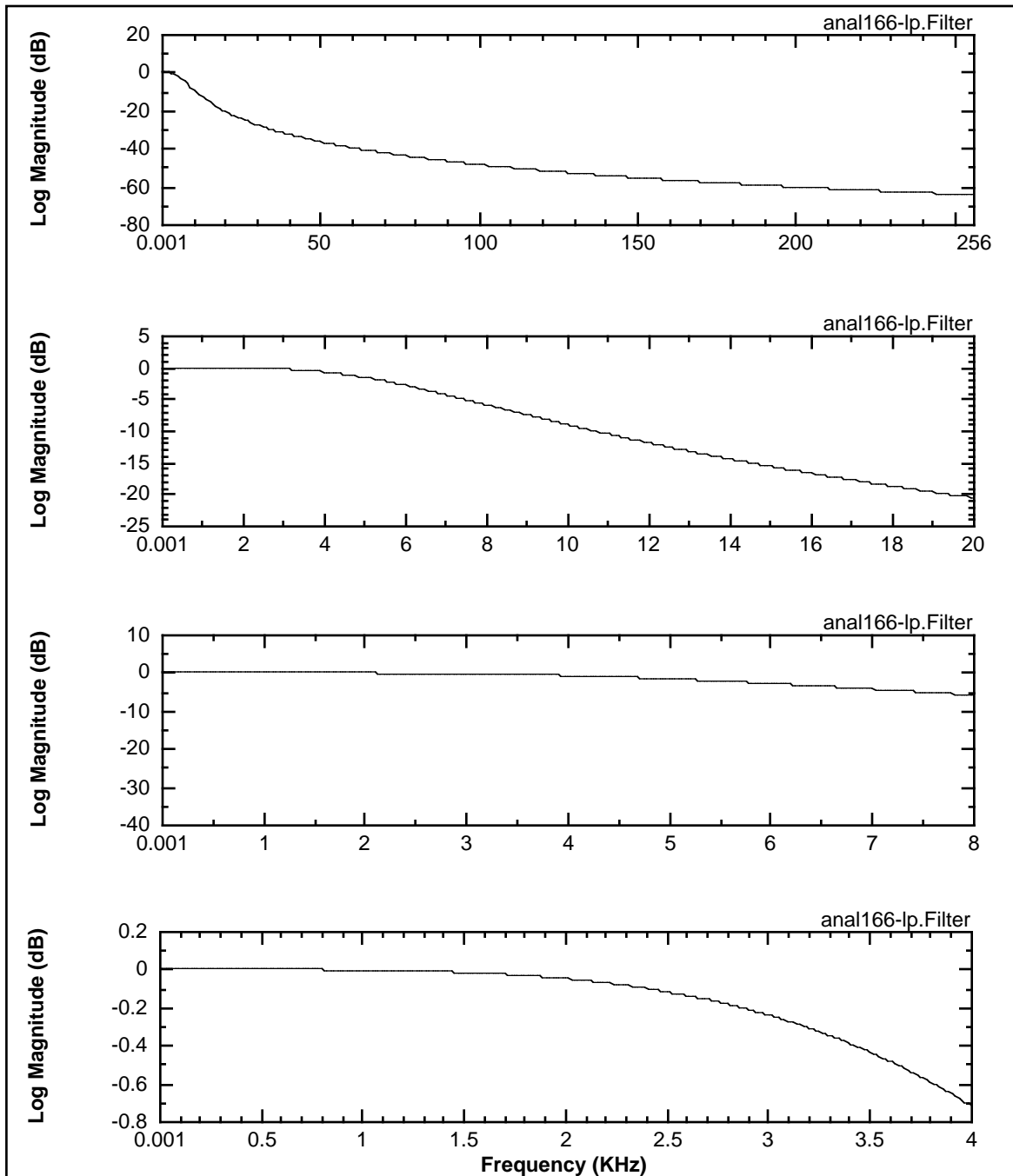
The analog low-pass filter is a two-pole Butterworth switched capacitor filter. The purpose of this filter is to attenuate the out-of-band quantization noise and the images created by the upsampling.

Figure 6-13 gives the transfer function of the filter:

$$H(z) = \frac{(0.0003506002)}{(1 - 1.973345z^{-1} + 0.9736956z^{-2})}$$

**Figure 6-13 Analog Low-pass Filter Transfer Function**

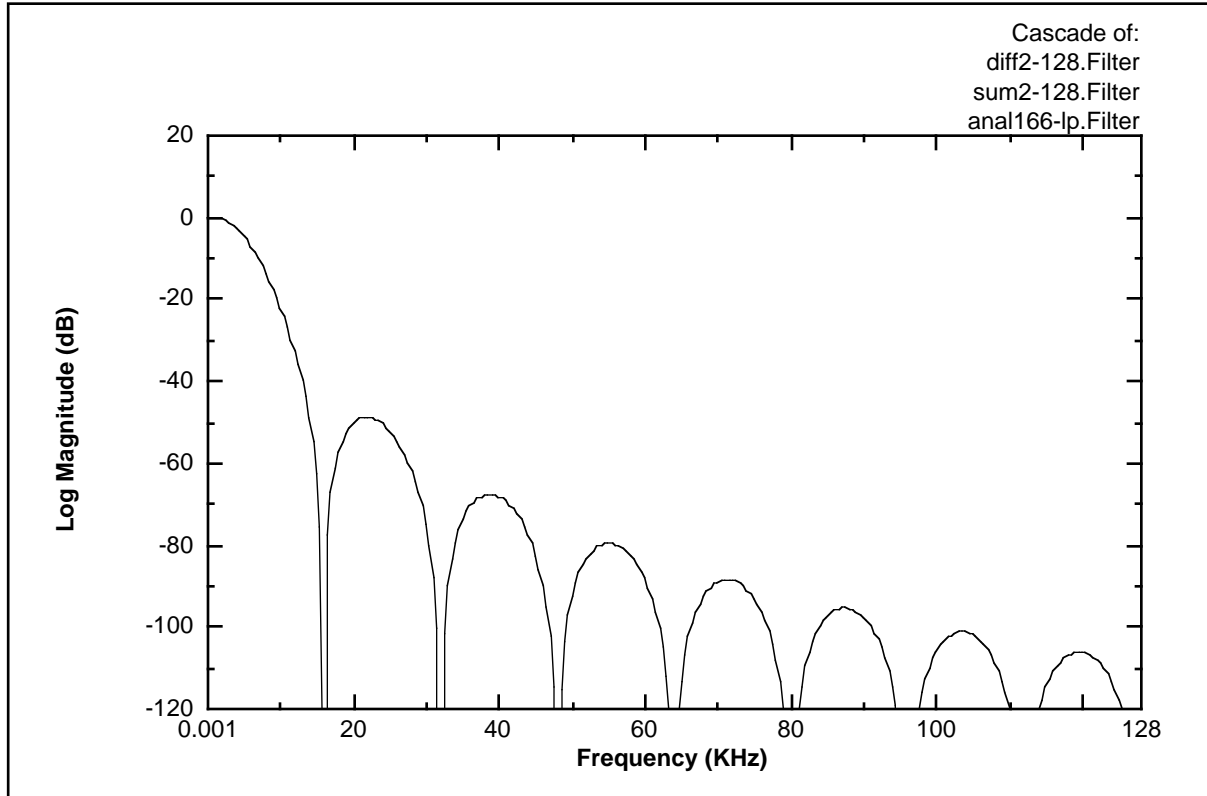
Figure 6-14 shows the log magnitude response of the D/A analog low-pass filter. The figures show the frequency response of the filter in the 0-256KHz, 0-20KHz, 0-8KHz, and 0-4KHz bands.



**Figure 6-14 Log Magnitude Frequency Response of the D/A Analog Low-pass Filter for  $F=2.048\text{MHz}$**

## 6.6.2.4 Overall Frequency Response of the D/A Section

Figure 6-15 shows the D/A frequency response in the 0-128KHz band. Figure 6-16 shows the overall frequency response of the D/A section in the 0-16KHz and 0-4KHz bands.



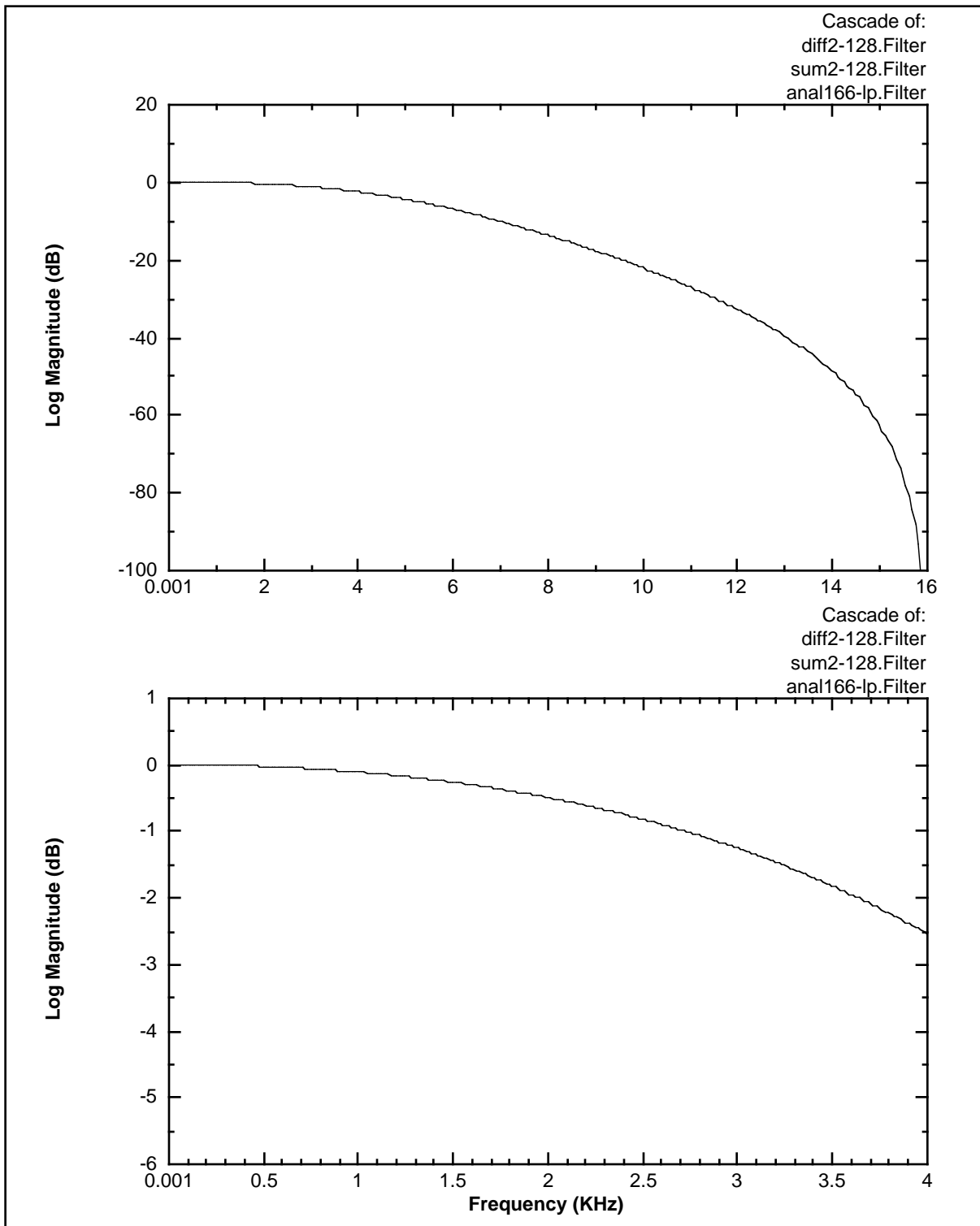
**Figure 6-15 Log Magnitude Frequency Response of the D/A Section for F=2.048 MHz and D=128**

It can be noted in Figure 6-16b that the frequency response is not flat in the 0-4KHz band. The interpolation filter performed by the DSP core can compensate for this droop in amplitude caused by the D/A hardware section.

The digital second order D/A comb filter is the only filter that needs to be compensated. Its transfer function is given in Figure 6-13.

It is equal to:

$$F(f) = \frac{1}{128D} \left( \frac{\sin\left(\frac{2\pi D}{2F} \times f\right)}{\sin\left(\frac{2\pi}{2F} \times f\right)} \right)^2$$



**Figure 6-16 Log Magnitude Frequency Response of the D/A Section for F=2.048 MHz and D=128**

In the present example, D=128 and F=2048 MHz, which gives:

$$F(f) = \frac{1}{(128)^2} \left( \frac{\sin\left(\frac{2\pi \times 128}{2 \times 2048} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 2048} \times f\right)} \right)^2$$

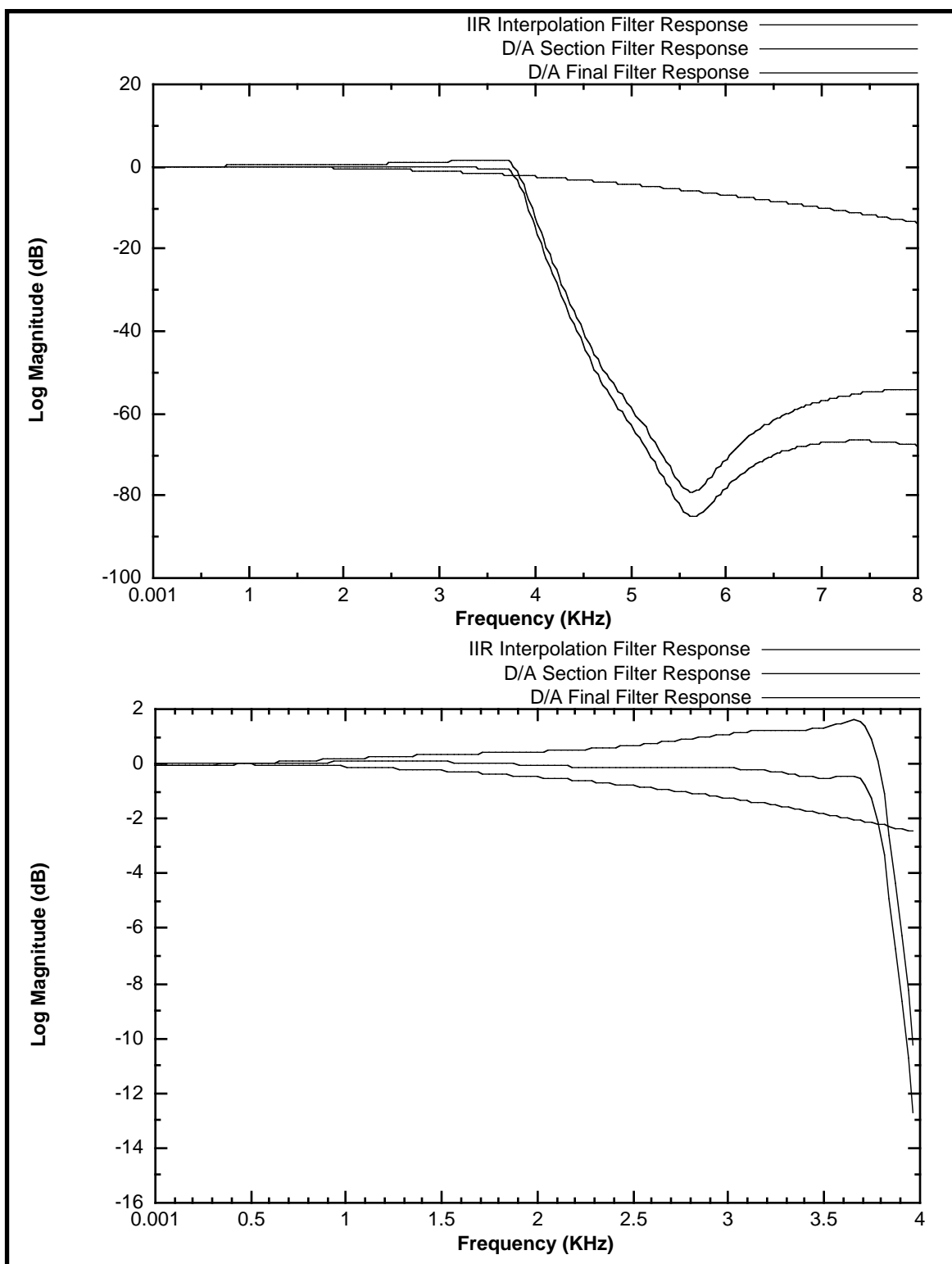
The transfer function of the interpolation filter will have to be shaped by 1/F(f) in order to flatten the response of the D/A section.

Table 6-7 gives a 4 biquad IIR low pass filter who's transfer function has been shaped accordingly.

Figure 6-17 shows the frequency response of the filter and the effects on the overall response of the D/A section.

**Table 6-7 Example of a Four Biquad IIR Interpolation and Compensation Filter**

; Source filter file: "comp1282nd.IIR.Filter"		
_filter_type	equ	BIQUAD_FILTER_TYPE
_NSTAGES	equ	4 ; number of stages
	dc	\$0270 ; gain = 0.03807147513/2
	; Biquad stage no. 1	
	dc	-\$18de ; d2_1 = 0.3885309315/2
	dc	\$e920 ; d1_1 = -0.3574372286/2
	dc	\$329e ; n2_1 = 0.7908895273/2
	dc	\$43d3 ; n1_1 = 1.059754603/2
	; Biquad stage no. 2	
	dc	-\$26b9 ; d2_2 = 0.6050537737/2
	dc	\$ec15 ; d1_2 = -0.3112185033/2
	dc	\$1014 ; n2_2 = 0.2512476586/2
	dc	\$1146 ; n1_2 = 0.2698979411/2
	; Biquad stage no. 3	
	dc	-\$0e57 ; d2_3 = 0.2240856408/2
	dc	\$cf53 ; d1_3 = -0.7605800752/2
	dc	\$3603 ; n2_3 = 0.8439490258/2
	dc	\$4614 ; n1_3 = 1.094956692/2
	; Biquad stage no. 4	
	dc	-\$38d6 ; d2_4 = 0.8880624617/2
	dc	\$f59a ; d1_4 = -0.162456827/2
	dc	\$2f25 ; n2_4 = 0.7366593399/2
	dc	\$1c8f ; n1_4 = 0.4462262322/2
<b>NOTE:</b> This filter, as well as all the figures representing filter responses, has been generated using ZOLA Technologies, Inc., DSP Designer™ software package.		



**Figure 6-17 IIR Interpolation and D/A Section Log Magnitude Frequency Response for F=2.048 MHz and D=128**

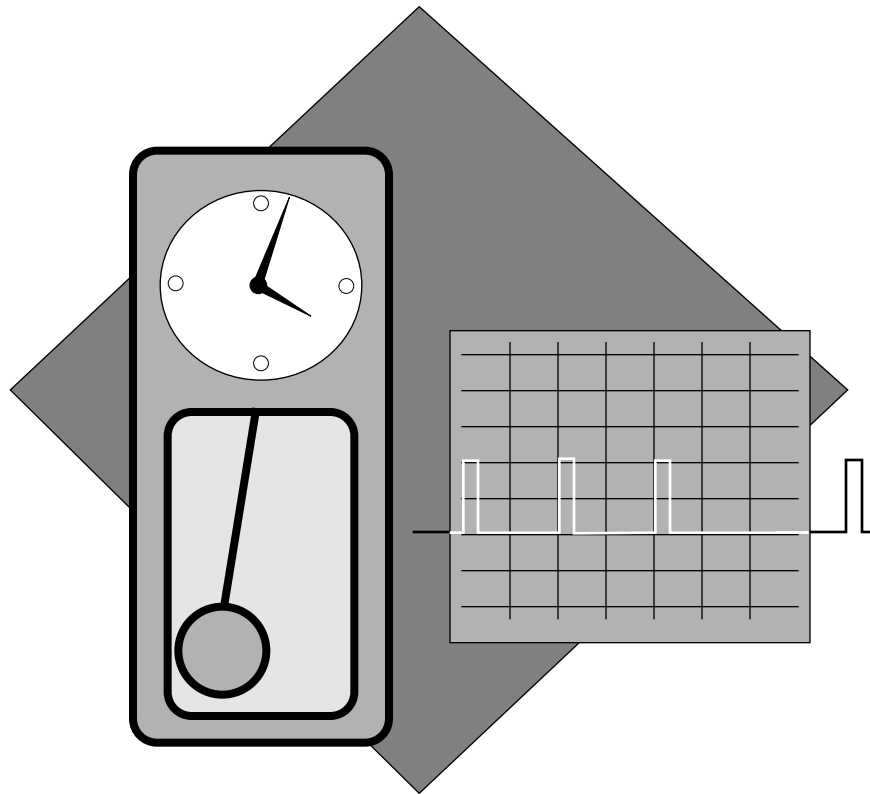




---

## SECTION 7

# 16-BIT TIMER AND EVENT COUNTER



## SECTION CONTENTS

---

7.1	INTRODUCTION .....	7-3
7.2	TIMER ARCHITECTURE .....	7-3
7.3	TIMER COUNT REGISTER (TCTR) .....	7-3
7.4	TIMER PRELOAD REGISTER (TPR) .....	7-4
7.5	TIMER COMPARE REGISTER (TCPR) .....	7-5
7.6	TIMER CONTROL REGISTER (TCR) .....	7-6
7.6.1.	TCR Decrement Ratio (DC7-DC0) Bits 0-7 .....	7-6
7.6.2.	TCR Event Select (ES) Bit 8 .....	7-6
7.6.3.	TCR Overflow Interrupt Enable (OIE) Bit 9 .....	7-6
7.6.4.	TCR Compare Interrupt Enable (CIE) Bit 10 .....	7-7
7.6.5.	TCR Timer Output Enable (TO2-TO0) Bits 11-13 .....	7-7
7.6.6.	TCR Inverter Bit (INV) Bit 14 .....	7-7
7.6.7.	TCR Timer Enable (TE) Bit 15 .....	7-8
7.7	TIMER RESOLUTION .....	7-8
7.8	FUNCTIONAL DESCRIPTION OF THE TIMER .....	7-8

## **7.1 INTRODUCTION**

This section describes a general purpose 16-bit timer/event counter with either internal clocking to count internal events or external clocking to count external events. This timer/event counter can be used to either interrupt the DSP or to signal an external device at periodic intervals. A Timer Input pin (TIN) can be used as an event counter input and a Timer Output pin (TOUT) can be used as a timer pulse or for timer clock generation.

## **7.2 TIMER ARCHITECTURE**

Figure 7-1 shows the general block diagram of the timer. It includes three 16-bit registers: the Timer Count Register (TCTR), the Timer Preload Register (TPR), and the Timer Compare Register (TCPR). An additional Timer Control Register (TCR) controls the timer operations.

A decrement register, programmed by the control register, is not available to the user. All other registers are read/write registers memory mapped as shown in Figure 7-2.

## **7.3 TIMER COUNT REGISTER (TCTR)**

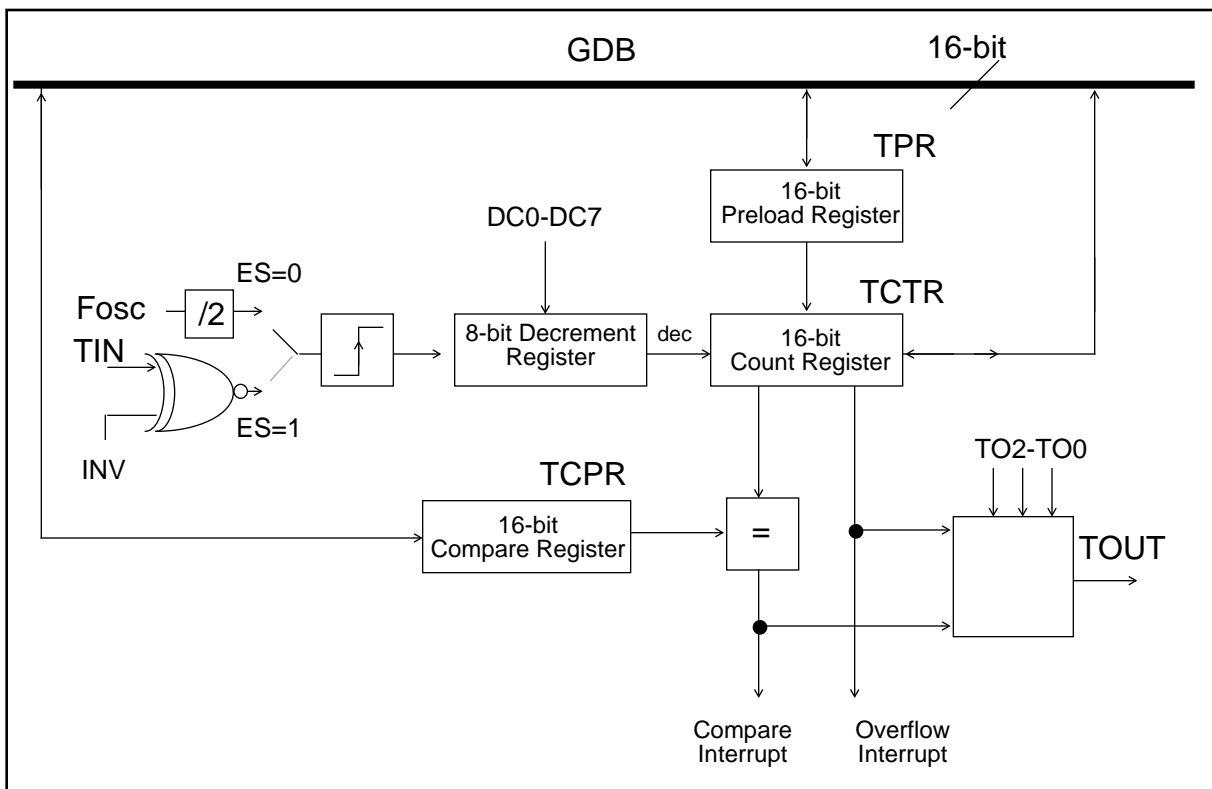
When the timer is enabled ( $TE=1$ ), the 16-bit timer count register is decremented by one after the decrement register has reached the value zero. On the next event after the count register reaches the value zero, an overflow interrupt will be generated if the Overflow Interrupt Enable bit (OIE) is set in the timer control register. Also, the state of the TOUT pin can then be affected according to the mode selected by the Timer Out Enable bits (TO2-TO0) of the timer control register. If  $n$  is the value stored in the count register when the timer is enabled, the overflow interrupt occurs after  $(n+1)*(DC+1)$  input events,  $DC$  being the preset value of the decrement register. After reaching zero, the count register is reloaded with the contents of the preload register or with a direct value if a direct write to the count register had been executed after the last timer count register reload.

On the next event after the count register reaches the value of the compare register, a compare interrupt is generated if the Compare Interrupt Enable bit (CIE) is set in the timer control register. The state of the TOUT pin may also be affected according to the mode selected by the Timer Out Enable bits (TO2-TO0) of the timer control register.

The user program can write a new value into the count register anytime. If the timer is enabled ( $TE=1$ ) during the write to the count register, this new value is written to the TCTR on the next count register decrement (next event after the decrement register reaches zero). If the timer is disabled ( $TE=0$ ) during the write, the value is immediately written to the count register and will not be overwritten by the value stored in the preload register when the timer gets enabled ( $TE=1$ ). In that case, the value stored or written in the preload reg-

ister will be loaded into the count register on the next event after it reaches zero, unless another write to the count register is performed in between. Refer to Section 7.8, Functional Description of the Timer, for more details.

The count register is initialized to zero on hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).



**Figure 7-1 16-bit Timer General Block Diagram**

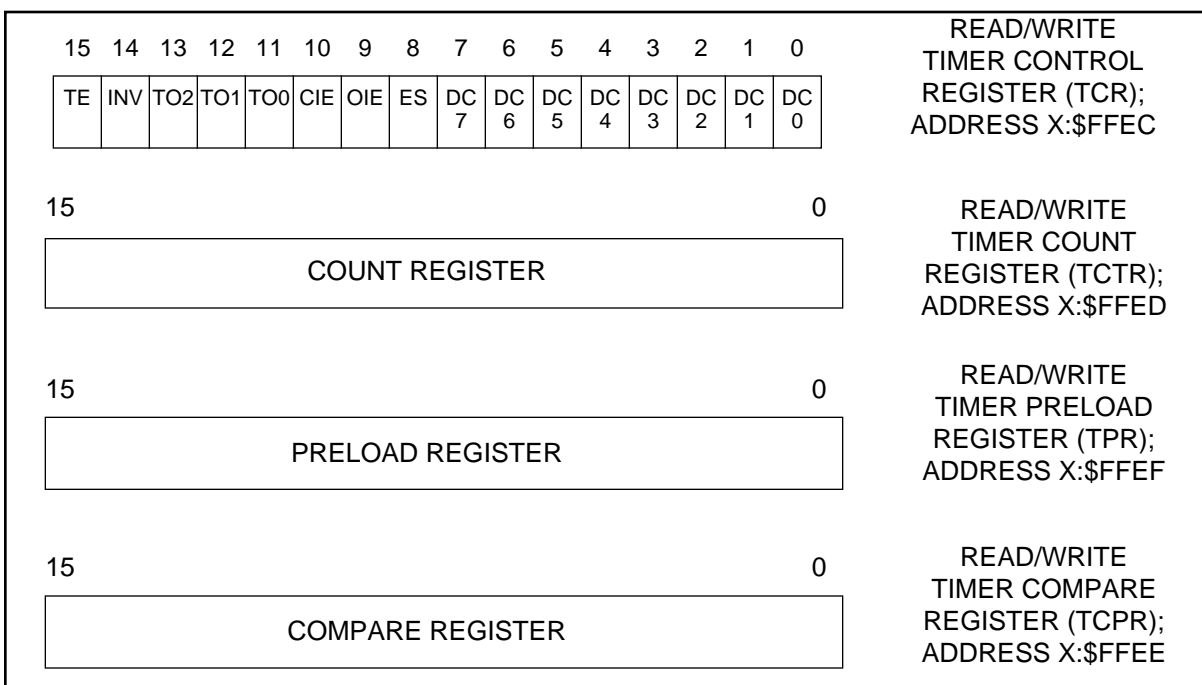
## 7.4 TIMER PRELOAD REGISTER (TPR)

The preload register is a 16-bit read/write register which typically contains the value to be reloaded inside the count register when the timer is enabled and when the timer count register (TCTR) has been decremented to zero.

If the timer is enabled (TE=1) when the user program writes a new value inside the preload register (TPR), this new value is transferred to the count register the next time the count register is loaded (after it reaches zero), unless a direct write to the count register is performed while the TCTR is zero.

If the timer is disabled (TE=0) when the user program writes a new value inside the preload register, this new value transfers immediately into the count register unless a direct write to the TCTR has already been performed.

The preload register is initialized to zero by hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).



**Figure 7-2 Timer Programming Model**

## 7.5 TIMER COMPARE REGISTER (TCPR)

The output compare register is a 16-bit read/write register used to program an action when the value of the count register reaches the value contained in the compare register. The value in the compare register is compared against the value of the count register on every instruction cycle. At the next event after the compare matches, an interrupt is generated if interrupts are enabled (CIE=1) and the state of the TOUT pin changes according to the mode selected by the Timer Out Enable bits (TO2-TO0) of the timer control register. This is useful for providing a pulse width modulated timer output.

The compare register is initialized to zero by hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

## 7.6 TIMER CONTROL REGISTER (TCR)

The timer control register is a 16-bit read/write register that contains the control bits for the timer. The control bits are defined in the following paragraphs.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TE	INV	TO2	TO1	TO0	CIE	OIE	ES	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0	READ/WRITE TIMER CONTROL REGISTER (TCR) ADDRESS X:\$FFEC

**Figure 7-3 Timer Control Register**

### 7.6.1 TCR Decrement Ratio (DC7-DC0) Bits 0-7

DC7-DC0 are eight clock divider bits that are used to preset an 8-bit counter which is decremented at the input clock rate. If DC7-DC0 = n, n+1 clock cycles will be counted before decrementing the count register, i.e., the decrement register acts as a prescaler. The 8-bit decrement register is not accessible to the user.

If the timer is disabled (TE=0) when a new value is written to this field, the decrement register will start decrementing with this initial value when the timer is enabled (TE=1). If the timer is enabled (TE=1) when a new value is written to this field, the decrement register will be reloaded with this value after it has reached the value zero. DC7-DC0 are reset to zero on hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

### 7.6.2 TCR Event Select (ES) Bit 8

The event select bit (ES) selects the source of the timer clock. If ES is cleared, Fosc/2 is selected as input of the decrement register. If ES is set, an external signal coming from the TIN pin is used as input to the decrement register. The external signal is synchronized to the internal clock and should be lower than the maximum internal frequency Fosc/4. ES is cleared by hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

### 7.6.3 TCR Overflow Interrupt Enable (OIE) Bit 9

The overflow interrupt has precedence over the compare interrupt at the same priority level. A compare interrupt will remain pending until all pending overflow interrupts are serviced. When the Overflow Interrupt Enable bit (OIE) is set, the DSP will be interrupted at the next event after the count register reaches zero. When the OIE bit is cleared, this interrupt is disabled. OIE bit is cleared on hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

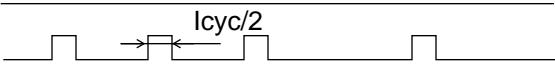


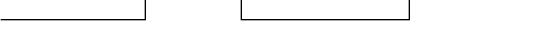

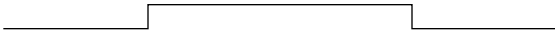
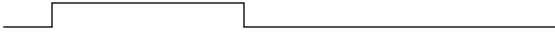
#### 7.6.4 TCR Compare Interrupt Enable (CIE) Bit 10

When the Compare Interrupt Enable bit (CIE) is set, the DSP will be interrupted at the next event after the count register reaches the value contained in the compare register. When the CIE bit is cleared, this interrupt is disabled. The CIE bit is cleared on hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

#### 7.6.5 TCR Timer Output Enable (TO2-TO0) Bits 11-13

The three timer output enable bits (TO2-TO0) are used to program the function of the timer output pin (TOUT). Table 7-1 shows the relationship between the value of TO2-TO0 and the function of the TOUT pin. These bits are cleared on hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

**Table 7-1 TOUT Pin Function**

TO2	TO1	TO0	Function of TOUT	Signal on TOUT
0	0	0	TOUT disabled	
0	0	1	Compare/Overflow pulse	
0	1	0	Overflow pulse	
0	1	1	Compare pulse	
1	0	0	Overflow/Compare toggle	
1	0	1	Compare/Overflow toggle	
1	1	0	Overflow toggle	
1	1	1	Compare toggle	

**Note:** If one of the toggle modes is selected and TE is written as zero while the TOUT pin is either high or low, the pin remains in the same state. If the TO2 bit is written as zero with the TE bit, the pin will remain high and will go low when the timer is re-enabled. Writing the TO2 bit as zero before writing the TE bit as zero will clear the TOUT pin, i.e., in the non-toggle modes, TOUT is normally low.

#### 7.6.6 TCR Inverter Bit (INV) Bit 14

When the inverter bit INV is set, the external signal coming in the TIN pin is inverted before entering the 8-bit decrement register. All 1 to 0 transitions of the TIN pin will then decrement the decrement register. When the INV bit is cleared, the external signal on TIN is not



inverted and the decrement register is decremented on all 0 to 1 transitions. INV is cleared on hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

### 7.6.7 TCR Timer Enable (TE) Bit 15

The TE bit is used to enable or disable the timer. Setting the TE bit will enable the timer. The decrement register will start decrementing from its preset value each time an event comes in. Clearing the TE bit will disable the timer. The decrement register will be preset to the value contained in bits DC7-DC0 of the control register and the count register will be loaded with the value of the preload register. However, if a direct write to the count register has happened since the last count register reload, the value written will be loaded into the count register instead of the preload value. TE is cleared by hardware  $\overline{\text{RESET}}$  and software reset (RESET instruction).

## 7.7 TIMER RESOLUTION

Table 7-2 shows the range of timer interrupt rates (overflow interrupt using internal event,  $F_{osc}/2$ ) that are provided by the 16-bit count register, the 16-bit preload register, and the 8-bit decrement register.

**Table 7-2 Timer Range and Resolution**

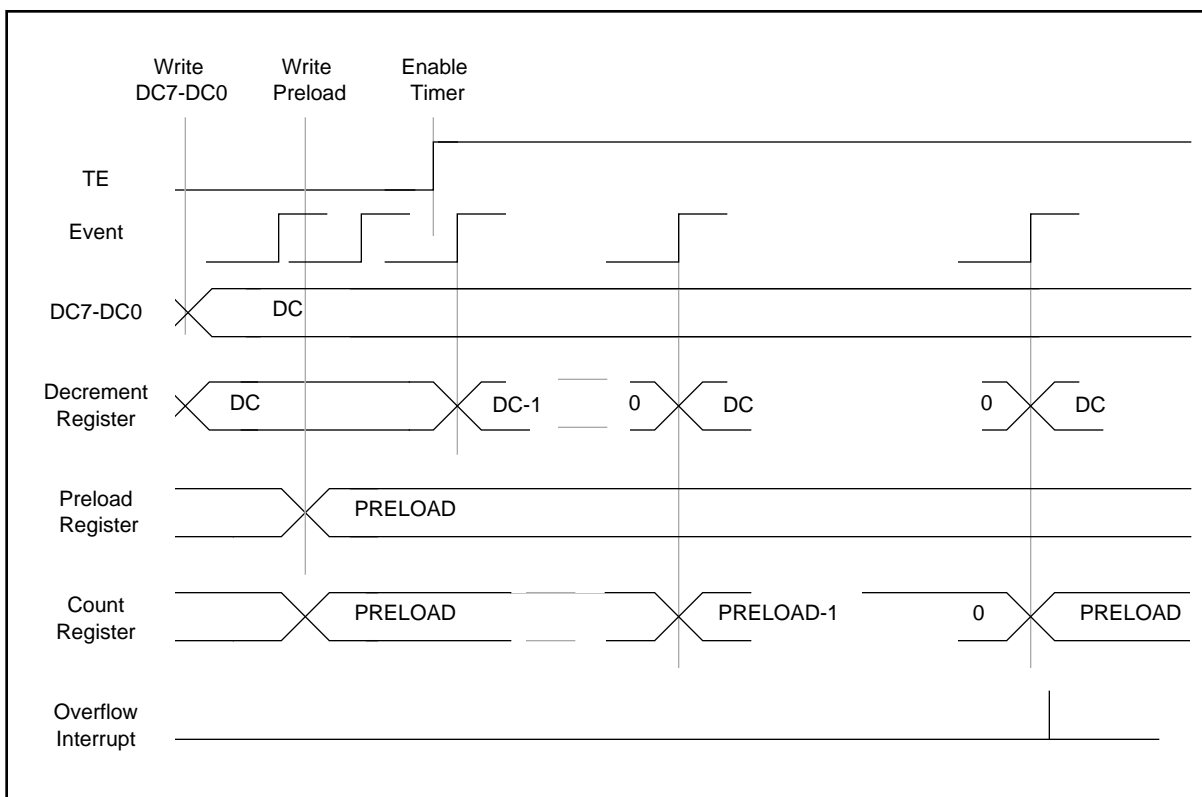
ICycle Time	DC7-DC0 value	Interrupt Rate (Preload = $2^{16}$ )	Resolution (Preload=0)
33 ns (60MHz-30 MIPS)	0 255	2.162 ms 553.5 ms	33 ns 8.4 $\mu$ s
51 ns (39 MHz-19.5MIPS))	0 255	3.342 ms 855.6 ms	51 ns 13.06 $\mu$ s
74 ns (27 MHz-13.5MIPS)	0 255	4.85 ms 1.242 s	74 ns 18.94 $\mu$ s

The overflow interrupt occurs every  $(\text{PRELOAD}+1) \cdot (\text{DC7-DC0}+1)$  input clock cycles.

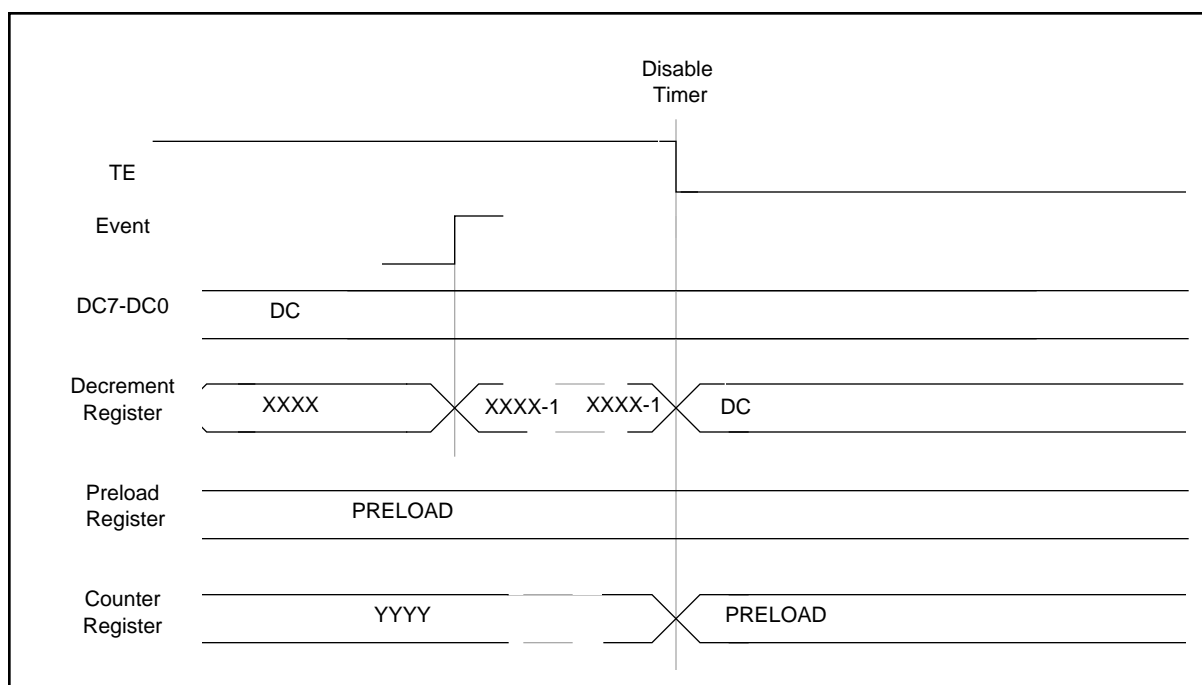
## 7.8 EVENT COUNTER TIMER DIAGRAMS

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

## EVENT COUNTER TIMER DIAGRAMS

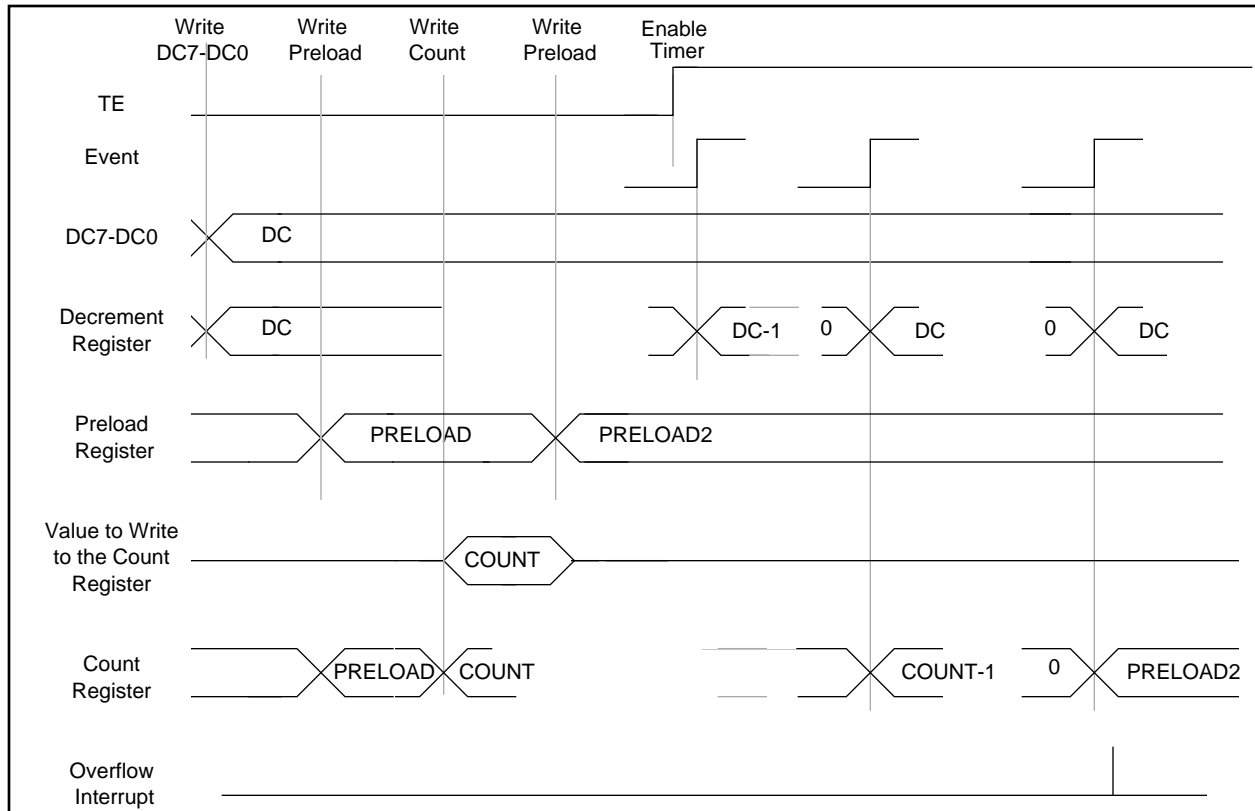


**Figure 7-4 Standard Timer Operation with Overflow Interrupt**

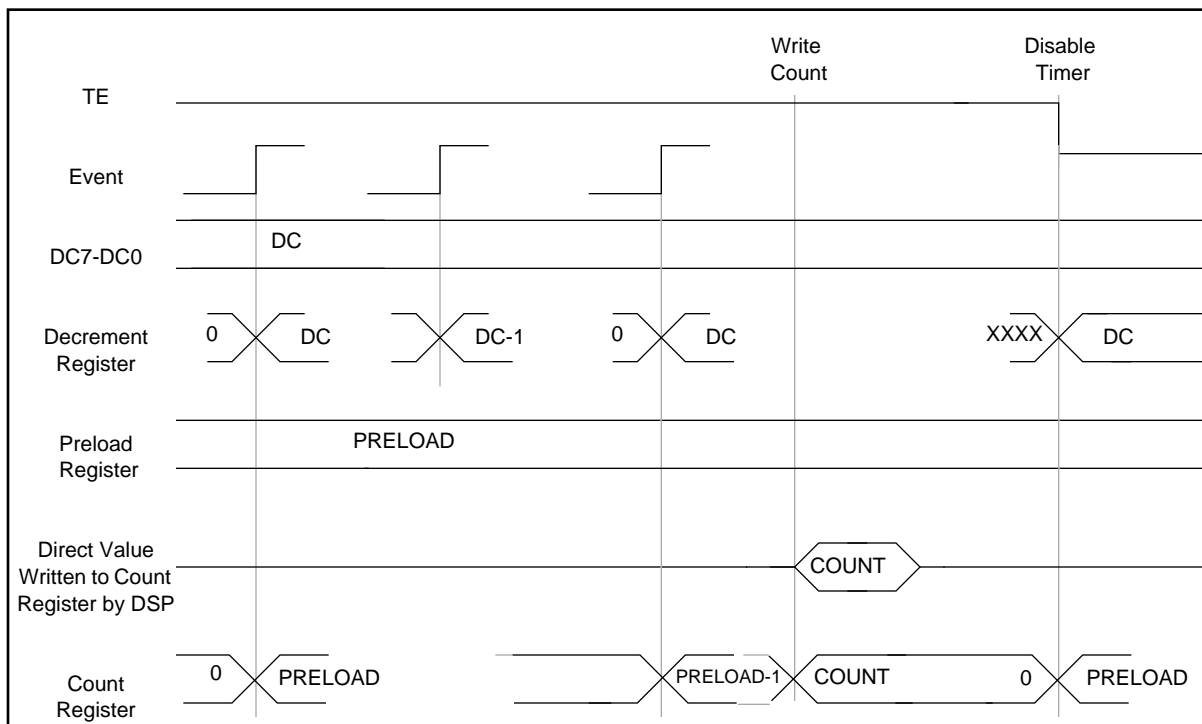


**Figure 7-5 Standard Timer Disable**

## EVENT COUNTER TIMER DIAGRAMS

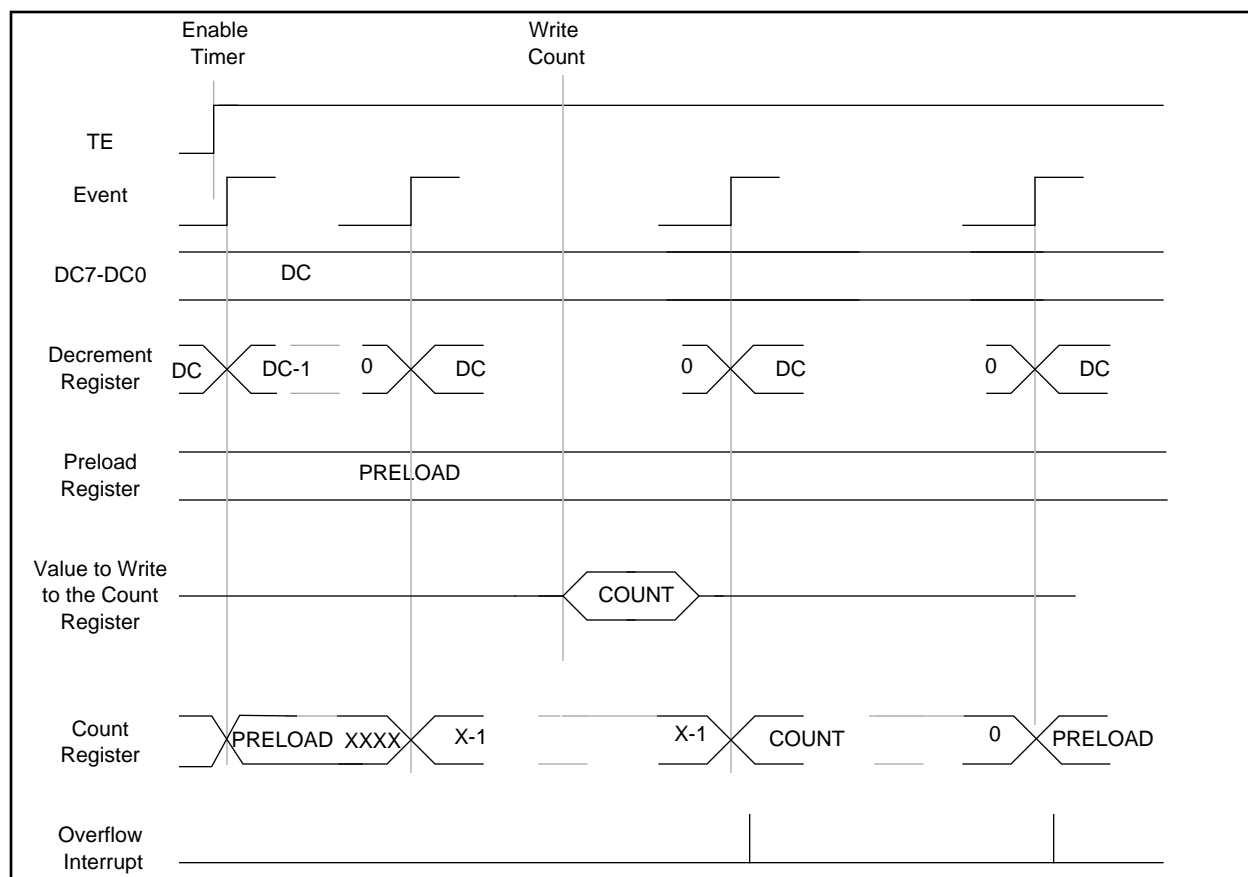


**Figure 7-6**  
**Write to the Count Register After Writing to the Preload Register**  
**When the Timer is Disabled**

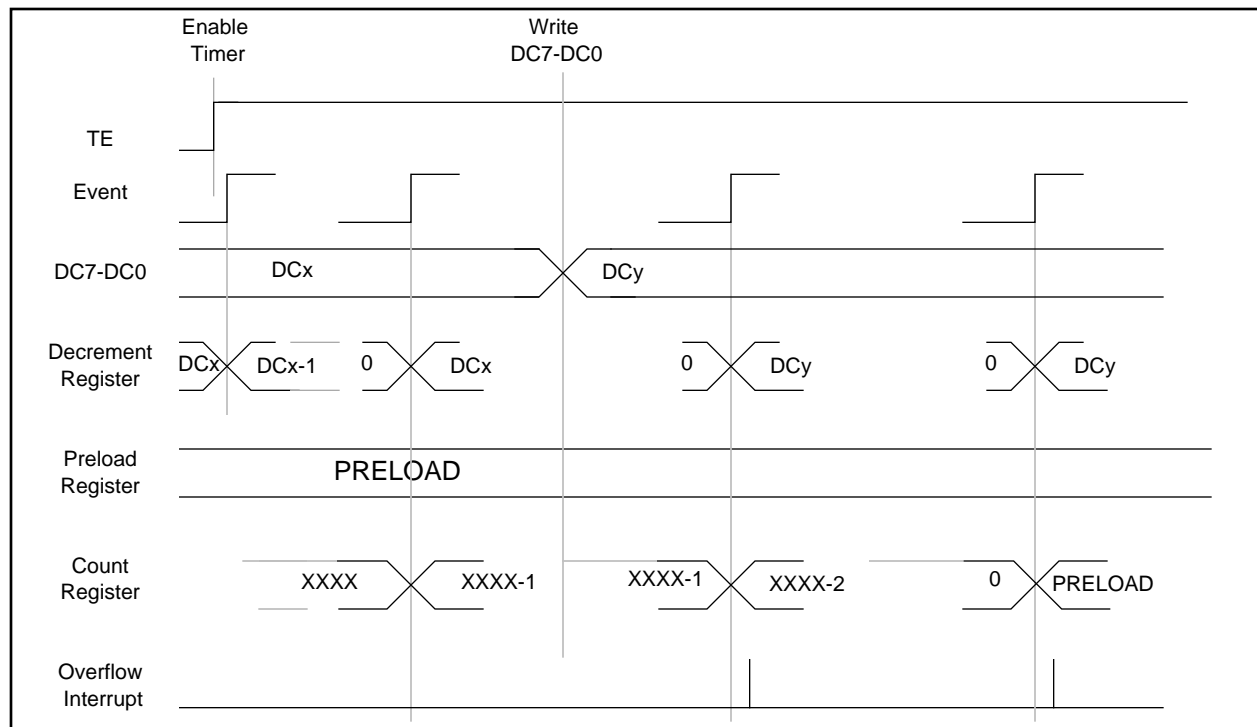


**Figure 7-7** **Timer Disable After a Write to the Count Register**

## EVENT COUNTER TIMER DIAGRAMS

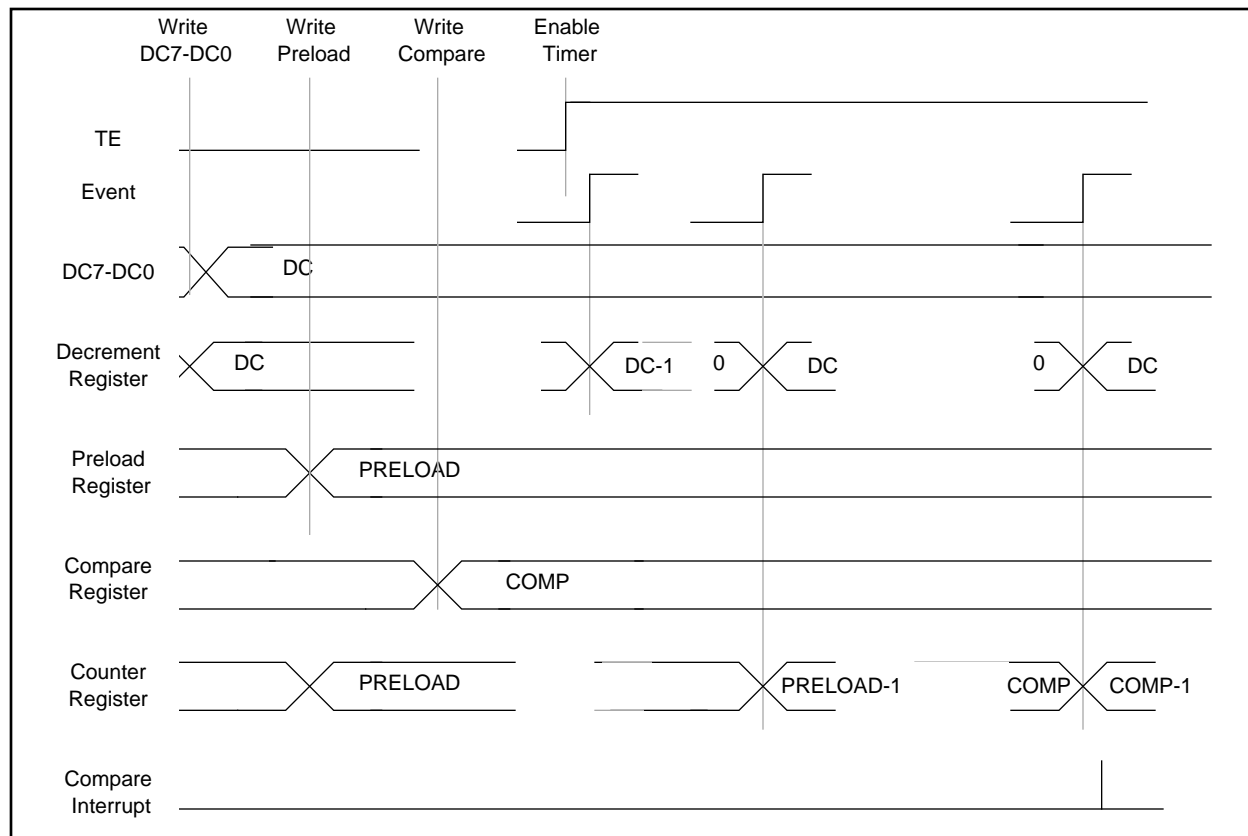


**Figure 7-8 Write to the Count Register when the Timer is Enabled**



**Figure 7-9 Write to DC7-DC0 when the Timer is Enabled**

## EVENT COUNTER TIMER DIAGRAMS

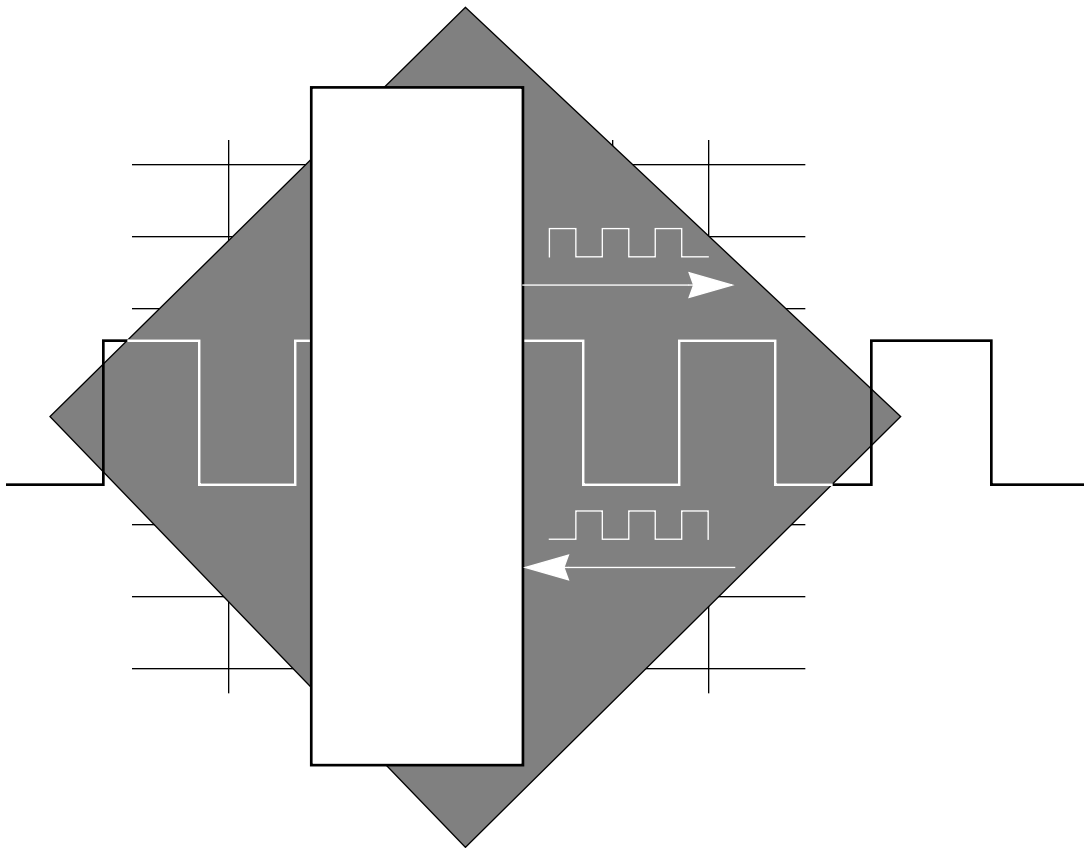


**Figure 7-10 Standard Timer Operation with Compare Interrupt**



## SECTION 8

### REDUCED SSI (RSSI0 and RSSI1)





## SECTION CONTENTS

---

8.1	INTRODUCTION .....	8-3
8.2	RSSI OPERATING MODES .....	8-3
8.3	RSSI CLOCK AND FRAME SYNC GENERATION .....	8-3
8.4	RSSI DATA AND CONTROL PINS .....	8-4
8.5	RSSI RESET AND INITIALIZATION PROCEDURE .....	8-7
8.6	RSSI INTERFACE PROGRAMMING MODEL .....	8-8
8.7	RSSI TRANSMIT SHIFT REGISTER .....	8-10
8.8	RSSI TRANSMIT DATA REGISTER (TX) .....	8-10
8.9	RSSI RECEIVE SHIFT REGISTER .....	8-10
8.10	RSSI RECEIVE DATA REGISTER (RX) .....	8-10
8.11	RSSI CONTROL REGISTER A (CRA) .....	8-11
8.12	RSSI CONTROL REGISTER B (CRB) .....	8-13
8.13	RSSI STATUS REGISTER .....	8-17
8.14	TIME SLOT REGISTER — TSR .....	8-19
8.15	NORMAL AND NETWORK OPERATING MODES .....	8-19

## 8.1 INTRODUCTION

The Reduced Synchronous Serial Interface (RSSI) is a full duplex serial port which allows the DSP to communicate with a variety of serial devices including one or more industry standard codecs, other DSPs, microprocessors, and peripherals. The RSSI interface consists of independent transmitter and receiver sections with a common RSSI clock generator and frame synchronization.

## 8.2 RSSI OPERATING MODES

The RSSI has several basic operating modes. These modes can be programmed by several bits in the RSSI control registers. Table 8-1. lists these operating modes and some of the typical applications in which they may be used:

**Table 8-1. RSSI Operating Modes**

<b>TX, RX Sections</b>	<b>Serial Clock</b>	<b>Protocol</b>	<b>Typical Applications</b>
Synchronous	Continuous	Normal	Multiple Synchronous Codecs
Synchronous	Continuous	Network	TDM Codec or DSP Networks
Synchronous	Gated	Normal	SPI-Type Devices; DSP to MCU
Synchronous	Gated	Network	DSP to SPI peripherals

The transmit and receive sections of this interface are synchronous; that is, the transmitter and the receiver use a common clock and frame synchronization signal. Continuous or gated mode may be selected. For continuous mode, the clock is continuously running and for gated mode the clock is only functioning during transmission. Normal or network protocol may also be selected. For normal protocol, the RSSI functions with one data word of I/O per frame. For network protocol, any number from two to eight data words of I/O may be used per frame. These distinctions result in the basic operating modes which allow the RSSI to communicate with a wide variety of devices.

## 8.3 RSSI CLOCK AND FRAME SYNC GENERATION

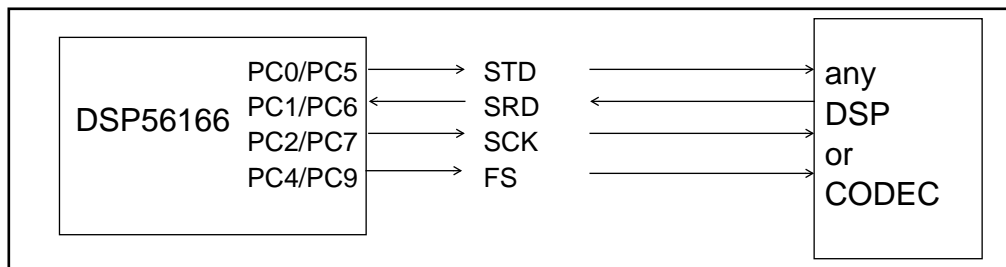
Data clock and frame sync signals can be generated internally by the DSP or may be obtained from external sources. If internally generated, the RSSI clock generator is used to derive bit clock and frame sync signals from the DSP internal system clock. The RSSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation. For gated clock mode, the data clock will be valid only when data is to be transmitted, otherwise the clock pin will be three-stated. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation. For gated mode, no frame sync signal is used.

## 8.4 RSSI DATA AND CONTROL PINS

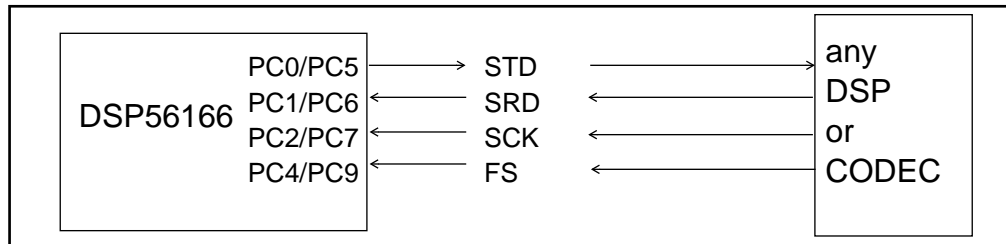
The RSSI has four dedicated I/O pins:

- Transmit data STDx (PC0 for RSSI0 and PC5 for RSSI1)
- Receive data SRDx (PC1 for RSSI0 and PC6 for RSSI1)
- Serial clock SCKx (PC2 for RSSI0 and PC7 for RSSI1)
- Serial frame sync SFS (PC4 for RSSI0 and PC9 for RSSI1)

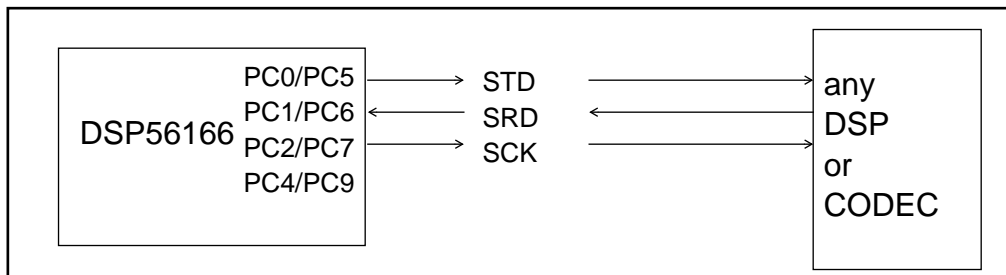
Figure 8-1 through Figure 8-4 show the main RSSI configurations and the following paragraphs describe the uses of these pins for each of the RSSI operating modes.



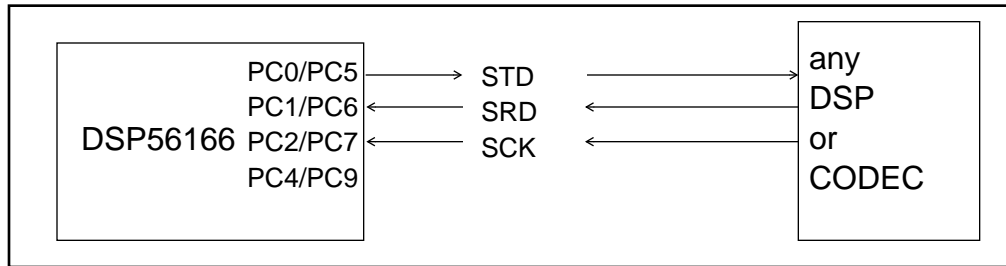
**Figure 8-1 RSSI Internal Continuous Clock**



**Figure 8-2 RSSI External Continuous Clock**

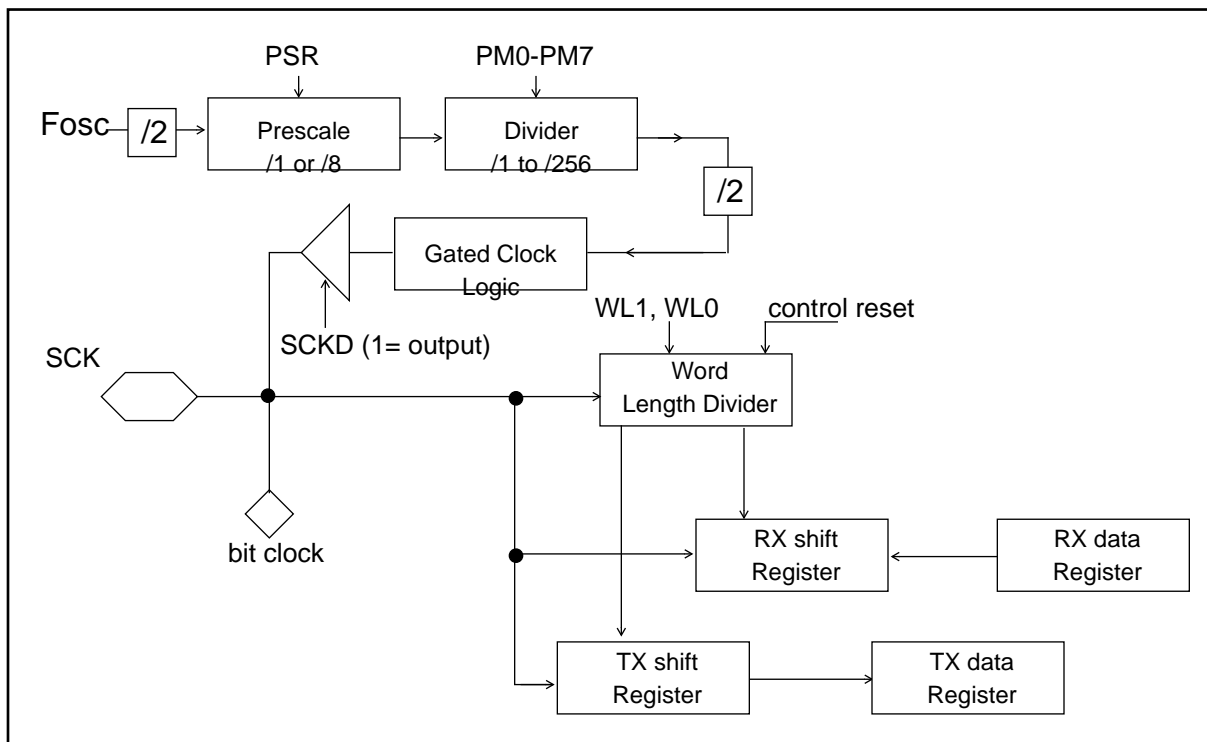


**Figure 8-3 RSSI Internal Gated Clock**



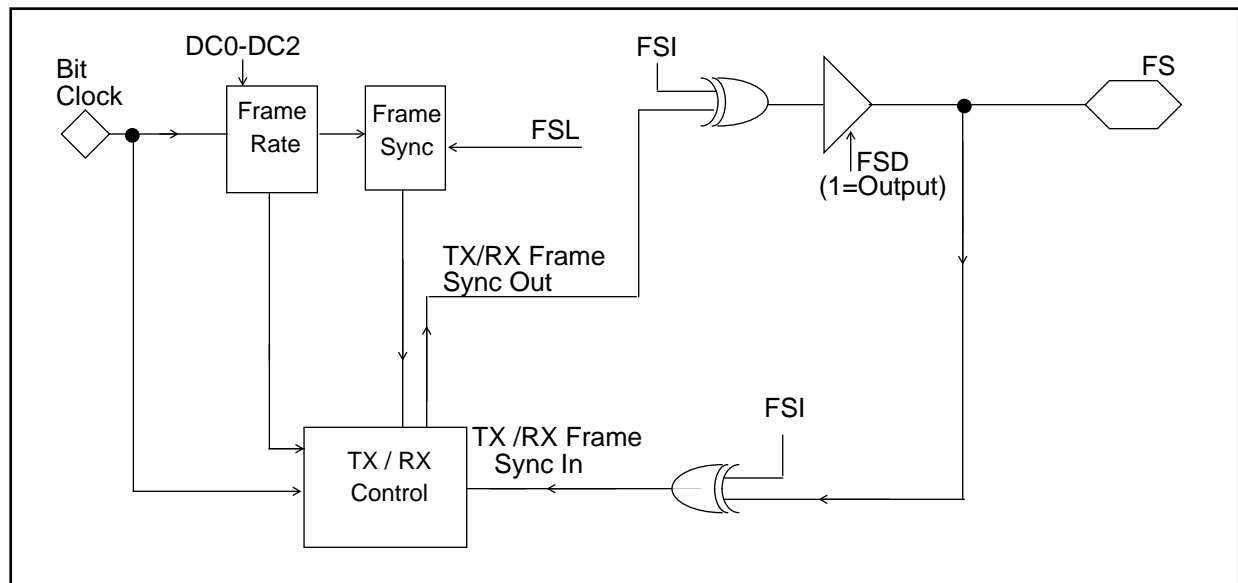
**Figure 8-4 RSSI External Gated Clock**

Figure 8-5 shows the internal clock path connections in block diagram form. The serial bit clock can be internal or external depending on the SCKD bit in the control register.



**Figure 8-5 RSSI Clock Generator Functional Block Diagram**

Figure 8-6 shows frame sync generation. When internally generated, both receive and transmit frame sync are generated from the word clock and are defined by the frame rate divider (DC2-DC0) bit and the word length (WL1-WL0) bits of CRA.



**Figure 8-6 RSSI Frame Sync Generator Functional Block Diagram**

#### 8.4.1 Serial Transmit Data Pin — STD

The Serial Transmit Data Pin (STD) is used to transmit data from the Serial Transmit Shift Register. STD is an output when data is being transmitted and is three-stated between data word transmissions and on the trailing edge of the bit clock after the last bit of a word is transmitted.

#### 8.4.2 Serial Receive Data Pin — SRD

The Serial Receive Data Pin (SRD) is used to bring serial data into the Receive Data Shift Register.

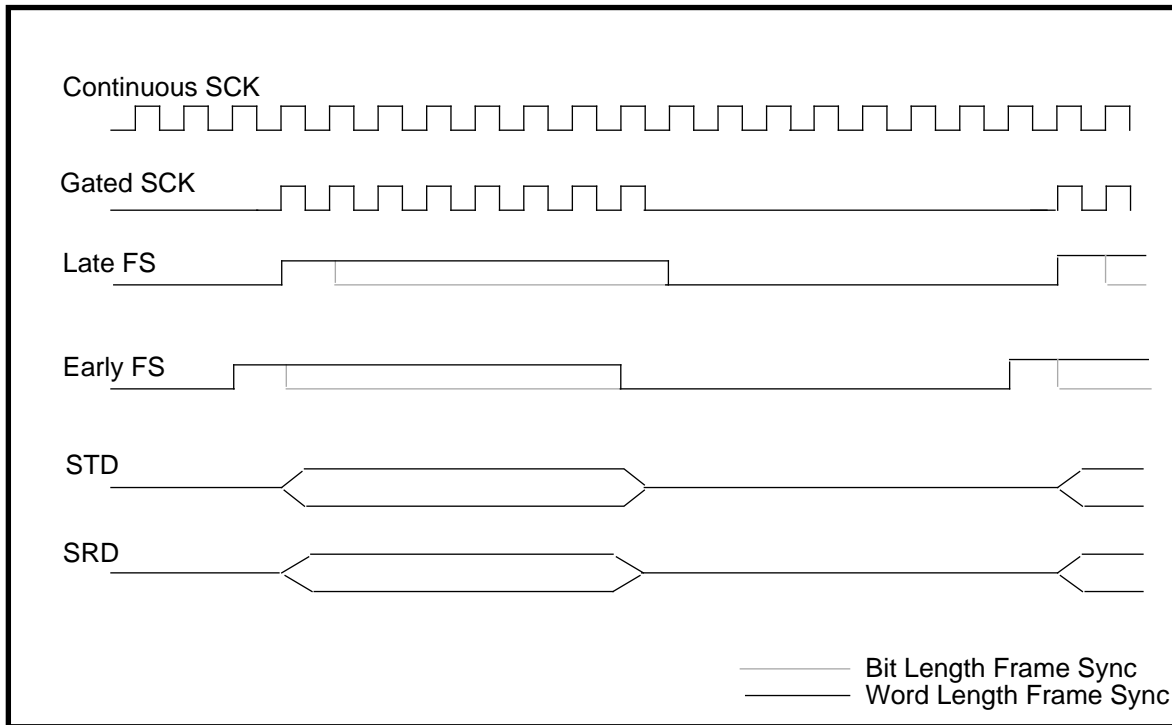
#### 8.4.3 Serial Clock — SCK

The Serial Clock (SCK) pin can be used as either an input or an output. This clock signal is used by both the transmitter and receiver and can be either continuous or gated. During gated mode, SCK is valid only during the transmission of data, otherwise it is three-stated.

#### 8.4.4 Serial Frame Sync — SFS

The Serial Frame Sync (FS) pin can be used as either an input or an output. The frame sync is used by both the transmitter and receiver to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. In the gated clock mode, frame sync signals are not used.

The pin signals are shown in Figure 8-7. Continuous and gated clock signals are shown as well as the bit length and word length frame sync signals.



**Figure 8-7 Serial Clock and Frame Sync Timing**

## 8.5 RSSI RESET AND INITIALIZATION PROCEDURE

The RSSI is affected by three types of reset:

- DSP Reset** The DSP hardware reset is generated by asserting the  $\overline{\text{RESET}}$  pin or the software reset is generated by executing the RESET instruction. The DSP hardware or software reset clears the SSI enable bit (SSIEN) in control register B which disables the RSSI. All other status and control bits in the RSSI are affected as described below.
- RSSI Reset** The RSSI reset is generated when the SSI enable bit (SSIEN) in control register B is cleared. The RSSI status bits are preset to the same state produced by the DSP reset. The RSSI control bits are unaffected. The RSSI reset is useful for selective reset of the RSSI interface without changing the present RSSI control bits and without affecting the other peripherals.

**STOP Reset** The STOP reset is caused by executing the STOP instruction. During the stop state no clocks are active in the chip. The RSSI status bits are preset to the same state produced by the DSP reset. The RSSI control bits are unaffected.

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

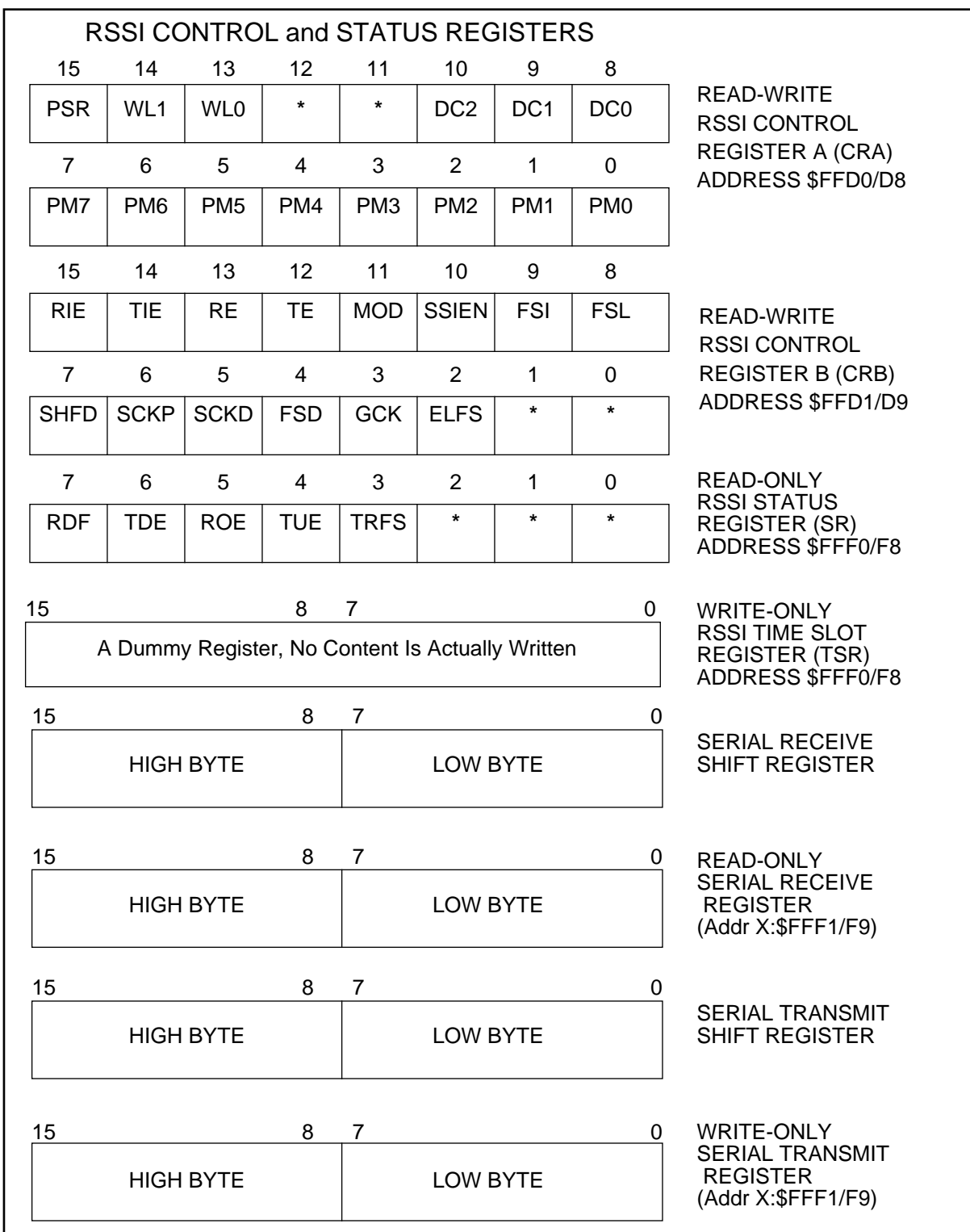
1. DSP reset or RSSI reset.
2. Program RSSI control registers.
3. Set the SSI enable bit (SSIEN) in control register B.

The DSP programmer should use the DSP or RSSI reset before changing the MOD, FSI, FSL, SHFD, SCKP, SCKD, FSD, GCK, ELFS, WL0, or WL1 control bits to ensure proper operation of the RSSI interface. That is, these control bits should not be changed during RSSI operation.

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

### 8.6 RSSI INTERFACE PROGRAMMING MODEL

The registers comprising the RSSI interface are shown in Figure 8-8. Note that the Codec device labels the MSB as bit 0, whereas the DSP labels the LSB as bit 0. Therefore, when using a standard Codec, the DSP MSB (or Codec bit 0) is shifted out first, and the MSB of CRB should be cleared.



\*\* - Reserved bits, read as zero, should be written with zero for future compatibility.

**Figure 8-8 RSSI Programming Model**



## 8.7 RSSI TRANSMIT SHIFT REGISTER

The Transmit Shift Register (TSR) is a 16 bit shift register that contains the data being transmitted. When a continuous clock is used, data is shifted out to the serial transmit data STD pin by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the serial transmit data STD pin by the selected (internal/external) gated clock. The Word Length control bits (WL1-WL0), in RSSI Control Register A, determine the number of bits to be shifted out of the TSR before it is considered empty and may be written to again. This word length can be 8, 12, or 16 bits. The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is always shifted out of this register with the most significant bit (MSB) first when the SHFD bit of the control register B is cleared. If this bit is set, the least significant bit (LSB) is shifted out first.

## 8.8 RSSI TRANSMIT DATA REGISTER (TX)

The Transmit Data Register is a 16-bit write-only register. Data to be transmitted is written into this register and is automatically transferred to the transmit shift register. The data written should occupy the most significant portion of the transmit data register. The unused bits (least significant portion) of the transmit data register are don't care bits. The DSP is interrupted whenever the transmit data register becomes empty provided that the transmit data register empty interrupt has been enabled.

**Note:** When early frame sync is selected (ELFS=1), if data is written into TX in the time between the frame sync and transmission of the first bit, the data will not be transmitted. TDE and TUE will be set when the first bit is transmitted.

## 8.9 RSSI RECEIVE SHIFT REGISTER

The Receive Shift Register is a 16 bit shift register that receives incoming data from the serial receive data SRD pin. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock. Data is assumed to be received most significant bit (MSB) first if the SHFD bit of CRB is cleared. If this bit is set, the data is received least significant bit (LSB) first. Data is transferred to the RSSI Receive Data Register after 8, 12, or 16 bits have been shifted in depending on the Word Length control bits (WL1-WL0) in RSSI Control Register A.

## 8.10 RSSI RECEIVE DATA REGISTER (RX)

The RSSI Receive Data Register is a 16 bit read-only register that accepts data from the Receive Shift Register as it becomes full. The data read will occupy the most significant portion of the receive data register. The unused bits (least significant portion) will read as

zeros. The DSP is interrupted whenever the receive data register becomes full if the associated interrupt is enabled.

## 8.11 RSSI CONTROL REGISTER A (CRA)

The RSSI Control Register A (CRA) is one of two 16 bit read/write control registers used to direct the operation of the Synchronous Serial Interface. The CRA controls the RSSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The DSP reset clears all CRA bits. RSSI reset and STOP reset do not affect the CRA bits. The CRA control bits are described in the following paragraphs.

15	14	13	12	11	10	9	8	READ-WRITE RSSI CONTROL REGISTER A (CRA) ADDRESS \$FFD0/D8
PSR	WL1	WL0	*	*	DC2	DC1	DC0	
7	6	5	4	3	2	1	0	
PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0	

\*unused bits should be set to zero

### 8.11.1 CRA Prescale Modulus Select (PM7-PM0) Bits 0-7

The Prescale Modulus Select bits (PM7 through PM0) specify the divide ratio of the prescale divider in the RSSI clock generator. This prescaler is used only in internal clock mode to divide the internal clock of the core. A divide ratio from 1 to 256 (PM=\$00 to \$FF) may be selected. The bit clock output is available at the clock SCK. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. Careful choice of the crystal oscillator frequency and the prescaler modulus will allow the telecommunication industry standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz to be generated. For example, a 24.576 MHz clock frequency may be used to generate the standard 2.048 MHz and 1.536 MHz rates, and a 24.704 MHz clock frequency may be used to generate the standard 1.544 MHz rate. Table 8-2. gives examples of PM7-PM0 values that can be used in order to generate different bit clocks.

**Table 8-2. RSSI bit clock as a function of Fosc and PM7-PM0 (PSR=0)**

Fosc (MHz)	Max bit Clock (MHz)	PM7-PM0 Values for different SCK				
		2.048Mhz	1.544Mhz	1.536Mhz	128Khz	64Khz
16.384	4.096	1	—	—	31(\$1F)	63(\$3F)
18.432	4.608	—	—	2	35(\$23)	71(\$47)
20.480	5.12	—	—	—	39(\$27)	79(\$4F)
26.624	6.656	—	—	—	51(\$33)	103(\$67)
24.576	6.144	2	—	3	47(\$2F)	95(\$5F)
24.704	6.176	—	3	—	—	—
32.768	8.192	3	—	—	63(\$3F)	127(\$7F)
36.864	9.216	—	—	5	71(\$47)	143(8F)
49.152	12.288	5	—	7	95(\$5F)	191(\$BF)
49.408	12.352	—	7	—	—	—

The bit clock on the RSSI can be calculated from the Fosc value using the following equation:

$$SCK = Fosc \div [(4 \times [7PSR+1] \times (PM+1))]$$

#### 8.11.2 CRA Frame Rate Divider Control (DC2-DC0) Bits 8-10

The Frame Rate Divider Control bits (DC2, DC1, and DC0) control the divide ratio for the programmable frame rate dividers. It operates on the word clock.

In normal mode, this ratio determines the word transfer rate. In network mode this ratio may be interpreted as the number of words per frame. The divide ratio may range from 1 to 8 (DC = 000 to 111) for normal mode and 2 to 8 (DC = 001 to 111) for network mode.

#### Examples:

##### In 8 bit word normal mode:

DC2-DC0= 1, PM7-PM0=9, PSR=1, Fosc = 40.96MHz would give a bit clock of  $40.96\text{Mhz} \div [8 \times 4 \times 10] = 128 \text{ kHz}$ . The 8-bit word rate being equal to two, the sampling rate (FS rate) would then be  $128 \text{ kHz} \div [2 \times 8] = 8\text{kHz}$ .

##### In 8 bit word network mode:

DC2-DC0= 6, PM7-PM0=80, PSR=0, Fosc = 62.22MHz would give a bit clock of  $62.22\text{Mhz} \div [4 \times 81] = 0.192 \text{ MHz}$  for a 7 slot TDM multiplex of 8-bit words. The sampling rate for every word (FS rate) would then be  $0.192 \text{ MHz} \div [3 \times 8] = 8\text{kHz}$ .

## 8.11.3 CRA Word Length Control (WL0, WL1) Bits 13 and 14

The Word Length Control bits (WL1-WL0) are used to select the length of the data words being transferred via the RSSI. Word lengths of 8, 12 or 16 bits may be selected as shown in Table 8-3.

**Table 8-3. RSSI Data Word Lengths**

WL1	WL0	Number of bits/word
0	0	8
0	1	(Reserved)
1	0	12
1	1	16

These bits control the Word Length Divider shown in the RSSI Clock Generator. The WL control bits also control the frame sync pulse length when FSL=0.

## 8.11.4 CRA Prescaler Range (PSR) Bit 15

The Prescaler Range (PSR) controls a fixed divide-by-eight prescaler in series with the variable prescaler. It is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When PSR is cleared, the fixed prescaler is bypassed. When PSR is set, the fixed divide-by-eight prescaler is operational. This allows a 128kHz master clock to be generated for Motorola MC1440X series codecs. The maximum internally generated bit clock frequency is  $F_{osc}/4$  and the minimum internally generated bit clock frequency is  $F_{osc}/(4 \times 8 \times 256)$ .

## 8.12 RSSI CONTROL REGISTER B (CRB)

The RSSI Control Register B (CRB) is one of two 16 bit read/write control registers used to direct the operation of the RSSI. The RSSI reset is controlled by a bit in the CRB. CRB controls the direction of the bit clock pin SCK and the frame sync pin FS. Interrupt enable bits for each data register interrupt are provided in this control register. RSSI operating modes are also selected in this register. The DSP reset clears all CRB bits. However, RSSI reset and STOP reset do not affect the CRB bits. The RSSI Control Register B bits are described in the following paragraphs.

15	14	13	12	11	10	9	8	READ-WRITE RSSI1 CONTROL REGISTER B (CRB) ADDRESS \$FFD1/D9
RIE	TIE	RE	TE	MOD	SSIEN	FSI	FSL	
7	6	5	4	3	2	1	0	
SHFD	SCKP	SCKD	FSD	GCK	ELFS	*	*	

\*Unused bits should be set to zero.

## 8.12.1 CRB Early/Late Frame Sync Bit (ELFS) Bit 2

The early/late frame sync bit (ELFS) controls when the frame sync is initiated for the transmit and receive sections. When ELFS is cleared, the frame sync is initiated as the first bit of data is transmitted and/or received. When ELFS is set, the frame sync is initiated one bit before the data is transmitted and/or received. The frame sync is disabled after one bit for bit length frame sync and after one word for word length frame sync.

## 8.12.2 CRB Gated Clock (GCK) Bit 3

The gated clock bit (GCK) selects the type of clock signal used to clock the Transmit Shift Register and the Receive Shift Register. When GCK is cleared, the clock signal is a continuous clock. When GCK and SCKD are set, the clock signal is an internally gated clock. The internally gated clock will run only when the transmitter is enabled (TE=1) during a valid time slot. When GCK is set and SCKD is cleared, the clock signal is an externally gated clock. The gated clock signal is useful when interfacing with microcontrollers.

## 8.12.3 CRB Frame Sync Direction (FSD) Bit 4

The Frame Sync Direction bit (FSD) controls the direction of the FS pin for the transmit and receive sections. When FSD is cleared, the FS pin is an input, meaning that the frame sync is supplied from an external source. When FSD is set, the FS pin is an output, meaning that the frame sync is generated internally.

## 8.12.4 CRB Clock Source Direction (SCKD) Bit 5

The Clock Source Direction bit (SCKD) selects the source of the clock signal used to clock the Transmit Shift Register and the Receive Shift Register. When SCKD is set, the clock source is internal and is the bit clock output of the RSSI clock generator. This clock will also appear at the SCK pin when SCKD is set. When SCKD is cleared, (1) the clock source is external, (2) the internal clock generator is disconnected from the SCK pin and (3) an external clock source may drive this pin to clock the Transmit Shift Register and the Receive Shift Register.

**8.12.5 CRB Clock Polarity Bit (SCKP) Bit 6**

The clock polarity bit controls which bit clock edge is used to clock out data and latched in data. If SCKP=0, the data is clocked out on the rising edge of the bit clock and latched in on the falling edge of the clock. If SCKP = 1, the falling edge of the clock is used to clock the data out and the rising edge of the clock is used to latch the data in.

**8.12.6 CRB MSB/LSB Position Bit (SHFD) Bit 7**

The SHFD bit controls whether the MSB or LSB is transmitted and received first. If SHFD = 0, the data is transmitted and received MSB first. If SHFD = 1, the LSB is transmitted and received first.

**8.12.7 CRB Frame Sync Length (FSL) Bit 8**

The Frame Sync Length (FSL) bit selects the length of the frame sync signal to be generated or recognized. If FSL=1, then a one clock bit long frame sync is selected. If FSL=0, a one word long frame sync is selected. The length this frame sync is the same as the length of the data word selected by WL0 and WL1.

**8.12.8 CRB Frame Sync Invert (FSI) Bit 9**

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

**8.12.9 CRB RSSI Enable Bit (SSIEN) Bit 10**

The RSSI enable (SSIEN) bit enables and disables the RSSI. If SSIEN=1, the RSSI is enabled, which causes an output frame sync to be generated (FSD=1) or causes the RSSI to wait for the input frame sync (FSD=0). If SSIEN=0, the RSSI is disabled. When disabled, the output clock and frame sync will be three-stated, the status register bits will be preset to the same state produced by the DSP reset, and the control register bits will not be affected.

**8.12.10 CRB RSSI Mode Select (MOD) Bit 11**

The Mode select bit (MOD) selects the operational mode of the RSSI. When MOD is cleared, the normal mode is selected. When MOD is set, the network mode is selected.

**8.12.11 CRB RSSI Transmit Enable (TE) Bit 12**

The RSSI Transmit Enable (TE) bit enables the transfer of TX to the Transmit Shift Register and also enables the internal gated clock. When TE is set and a word boundary is detected, the transmit portion of the RSSI is enabled. When TE is cleared, the transmitter will continue to transmit the data currently in the RSSI Transmit Shift Register and then disable the transmitter. The serial output is three-stated and any data present in

TX will not be transmitted, i.e. data can be written to TX with TE cleared, TDE will be cleared but data will not be transferred to the Transmit Shift Register. If TE is disabled and then re-enabled during the same transmitted word, then the data will continue to be transmitted. If TE is re-enabled during a different word slot, then data will not be transmitted until the next word boundary. **The normal transmit enable sequence for transmit is to write data to TX or to TSR before setting TE. The normal transmit disable sequence is to clear TE and TIE after TDE=1.**

When an internal gated clock is being used, the gated clock will run during valid word slots if the TE bit is set. If TE is disabled, the transmitter will continue to transmit the data currently in the RSSI Transmit Shift Register and then the clock will stop. When TE is re-enabled, the gated clock will start immediately and will run during any valid word slots.

**Note:** The function of disabling and enabling TE is different from the DSP56156 SSI.

#### 8.12.12 CRB RSSI Receive Enable (RE) Bit 13

When the RSSI Receive Enable (RE) bit is set, the receive portion of the RSSI is enabled. When this bit is cleared, the receiver will be disabled by inhibiting data transfer into RX. If data is being received while this bit is cleared, the rest of the word will not be shifted in and transferred to the RSSI Receive Data Register. If RE is re-enabled during a word slot before the second to last bit then that word will be received.

**Note:** The function of disabling and enabling RE is different from the DSP56156 SSI.

#### 8.12.13 CRB RSSI Transmit Interrupt Enable (TIE) Bit 14

When the RSSI Transmit Interrupt Enable bit (TIE) is set, the program controller will be interrupted when the RSSI Transmit Data Register Empty flag (TDE) in the RSSI Status Register is set. When TIE is cleared, this interrupt is disabled. However, the TDE bit will always indicate the transmit data register empty condition even when the transmitter is disabled by the TE bit. Writing data to the TX or TSR register will clear TDE thus clearing the interrupt.

There are two transmit data interrupts (these have separate interrupt vectors):

1. Transmit data with exception status.  
This interrupt is generated on the following condition:  
TIE=1 and TDE=1 and TUE=1
2. Transmit data without exceptions.  
This interrupt is generated on the following condition:  
TIE=1 and TDE=1 and TUE=0

**8.12.14 CRB RSSI Receive Interrupt Enable (RIE) Bit 15**

When the RSSI Receive Interrupt Enable bit (RIE) is set, the program controller will be interrupted when the RSSI Receive Data Register Full flag (RDF) in the RSSI Status Register is set. If RIE is cleared, this interrupt is disabled. However, the RDF bit still indicates the receive data register full condition. Reading the receive data register will clear RDF and thus clear the pending interrupt.

There are two receive data interrupts which have separate interrupt vectors:

1. Receive Data with exception status — This interrupt is generated on the following condition:

RIE=1 and RDF=1 and ROE=1

2. Receive Data without exceptions — This interrupt is generated on the following condition:

RIE=1 and RDF=1 and ROE=0

**8.13 RSSI STATUS REGISTER**

The RSSI Status Register (SR) is a 16-bit read only status register used by the DSP to interrogate the status and serial input flags of the Synchronous Serial Interface. The status bits are described in the following paragraphs.

7	6	5	4	3	2	1	0	READ-ONLY RSSI STATUS REGISTER (SR) ADDRESS \$FFF0/F8
RDF	TDE	ROE	TUE	TRFS	*	*	*	

\*Unused bits should be set to zero.

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

**8.13.1 RSSISR Transmit/Receive Frame Sync (TRFS) Bit 3**

When set, the Transmit/Receive Frame Sync flag (TRFS) indicates that a frame sync occurred during transmission of the last word written to the Tx register or receiving of the next word into the Rx register. Data written to the transmit data register during the time slot when TRFS is set will be transmitted during the second time slot (network mode) or in the next first time slot (normal mode). In network mode, TRFS is set during transmission or receiving of the first slot of the frame. It will then be cleared when starting transmission or receiving of the next slot.

TRFS is cleared by DSP, RSSI or STOP reset.



**Note:** This bit functions differently than the DSP56156 SSI TFS/RFS bits.

#### 8.13.2 RSSISR Transmitter Underrun Error (TUE) Bit 4

The Transmitter Underrun Error (TUE) flag is set when the Serial Transmit Shift Register is empty (no data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data will be re-transmitted.

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

TUE does not cause any interrupts; however, TUE does cause a change in the interrupt vector used for transmit interrupts so that a different interrupt handler may be used for a transmit underrun condition. If a transmit interrupt occurs with TUE set, the Transmit Data With Exception Status interrupt will be generated. If a transmit interrupt occurs with TUE clear, the Transmit Data Without Errors interrupt will be generated.

TUE is cleared by DSP, RSSI, or STOP reset. TUE is cleared by reading the SR with TUE set followed by writing TX or TSR.

#### 8.13.3 RSSISR Receiver Overrun Error (ROE) Bit 5

The Receiver Overrun Error flag (ROE) is set when the serial receive shift register is filled and ready to transfer to the receiver data register (RX) and the RX is already full (i.e. RDF=1). The Receiver Shift Register is not transferred to RX. ROE does not cause any interrupts; however, ROE does cause a change in the interrupt vector used so that a different interrupt handler may be used for a receive overrun condition. If a receive interrupt occurs with ROE set, the Receive Data With Exception Status interrupt will be generated and if a receive interrupt occurs with ROE clear, the Receive Data Without Errors interrupt will be generated.

ROE is cleared by DSP, RSSI, or STOP reset and is cleared by reading the SR with ROE set followed by reading the RX. Clearing RE does not affect ROE.

#### 8.13.4 RSSISR Transmit Data Register Empty (TDE) Bit 6

The RSSI Transmit Data Register Empty flag (TDE) is set when the contents of the Transmit Data Register are transferred to the Transmit Shift Register. When set, TDE indicates that data should be written to the TX or to the time slot register (TSR) before the transmit shift register becomes empty (which would cause an underrun error).

TDE is cleared when the DSP writes to the Transmit Data Register or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, a RSSI Transmit Data interrupt request will be issued when TDE is set. The vector of the interrupt will

depend on the state of the Transmitter Underrun TUE bit. TDE is set by DSP, RSSI, and STOP reset.

### **8.13.5 RSSISR Receive Data Register Full (RDF) Bit 7**

The RSSI Receive Data Register Full flag (RDF) is set when the contents of the Receive Shift Register are transferred to the Receive Data Register. RDF is cleared when the DSP reads the Receive Data Register. If RIE is set, a DSP receive data interrupt request will be issued when RDF is set. The interrupt request vector will depend on the state of the Receiver Overrun ROE bit. RDF is cleared by DSP, RSSI, and STOP reset.

## **8.14 TIME SLOT REGISTER — TSR**

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

## **8.15 NORMAL AND NETWORK OPERATING MODES**

In the normal mode, the frame rate divider determines the word transfer rate — one word is transferred per frame sync during the frame sync time slot. In network mode, a word is (possibly) transferred every time slot.

### **8.15.1 Normal Mode Transmit**

The conditions for data transmission from the RSSI are:

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

When the above conditions occur in normal mode, the next data word will be transferred from TX to the transmit shift register, the TDE flag will be set (transmitter empty), and the transmit interrupt will occur if TIE=1 (transmit interrupt is enabled). The new data word will be transmitted immediately.

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

### 8.15.2 Normal Mode Receive

If the receiver is enabled, then for a continuous clock, each time the frame sync signal is generated (or detected) a data word will be clocked in, and for a gated clock, each time the clock begins a data word will be clocked in. After receiving the data word it will be transferred from the RSSI Receive Shift Register to the Receive Data Register (RX), the RDF flag will be set (Receiver full), and the Receive Interrupt will occur if it is enabled (RIE=1).

The DSP program has to read the data from RX before a new data word is transferred from the Receive Shift Register, otherwise the Receiver Overrun error will be set (ROE).

The transmitter and receiver timing in normal mode for an eight bit word with two words per time slot is shown in Figure 8-9. Both continuous and gated clock are shown with a late word length frame sync.

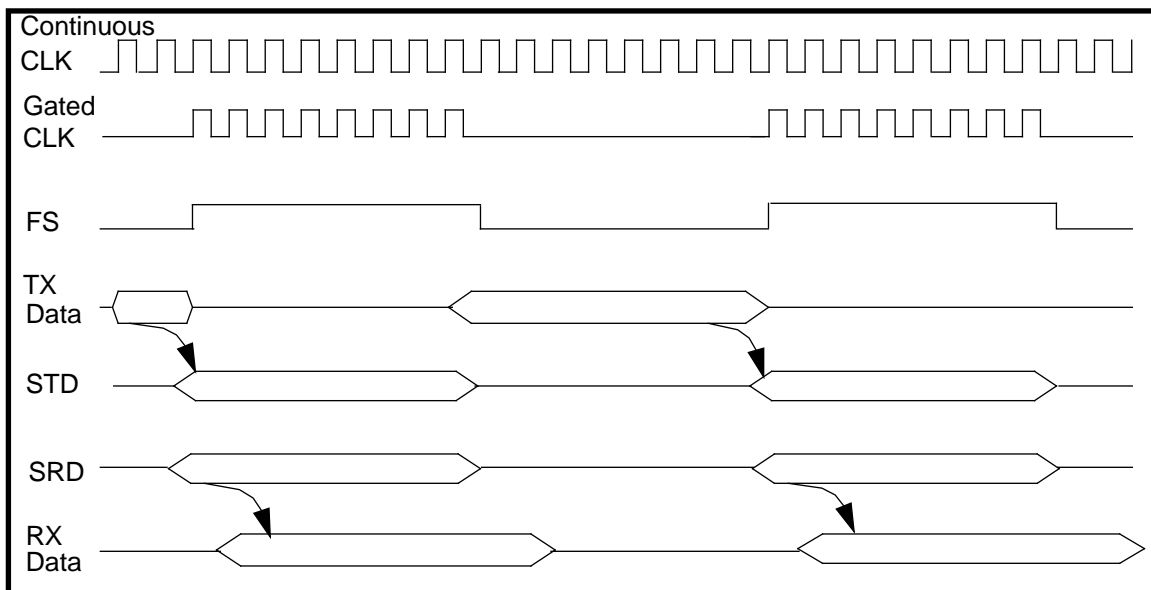


Figure 8-9 Normal Mode Timing

### 8.15.3 Network Mode

In this mode, the RSSI can be used in TDM networks. It is the typical mode in which the DSP would interface to a TDM codec network or a network of DSPs. The DSP may be a master device that controls its own private network or a slave device that is connected to an existing TDM network and occupies a few time slots. The distinction of the network mode is that it identifies each time slot (data word time) and allows the option of ignoring the time slot by writing to TSR or transmitting data during the time slot. The receiver is treated in the same manner except that data is always being shifted into the Receive Shift

Register and transferred to the RX. The DSP will read the receive data register and either use it or discard it.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception can occur in each time slot (rather than in just the frame sync time slot as in normal mode). The frame rate dividers, controlled by DC2, DC1, and DC0 select two to eight time slots per frame.

### 8.15.3.1 Network Mode Transmit

The transmit portion of RSSI is enabled when SSIEN=1 and TE=1. However, for continuous clock, when TE is set, the transmitter will be enabled only after detection of a new word slot (if TE is enabled during a slot other than the first). This is different from the DSP56156 SSI. **Software will have to find the start of the next frame.** For a gated clock, when TE is set, the transmitter will be enabled as soon as the clock begins.

Normal start up sequence for transmission is to:

1. Write the data to be transmitted to the Transmit Register (TX). This clears the TDE flag.
2. Set TE and TIE to enable the transmitter
  - on the next word boundary (continuous clock)
  - or on the next clock signal (gated clock).
3. Enable transmit interrupts.

Alternatively, the DSP programmer may decide to NOT transmit in a time slot by writing to the Time Slot Register (TSR). This will clear the TDE flag just as if data were going to be transmitted but the STD pin will remain in three-state during the time slot.

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

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

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

1. Write data register with data to enable transmission in the next time slot.
2. Write the time slot register to disable transmission in the next time slot.
3. Do nothing — transmit underrun will occur at the beginning of the next time slot and the previous data will be transmitted.

### 8.15.3.2 Network Mode Receive

The receiver portion of the RSSI is enabled when SSIEN=1 and RE =1. However, the receive enable will only take place during that word slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver will be disabled immediately. This is different from the DSP56156 SSI. **Software will have to find the start of the next frame.**

When the word is completely received, it is transferred to the RX data register which sets the RDF flag (Receive Data register full). Setting RDF will cause a receive interrupt to occur if the receiver interrupt is enabled (RIE=1).

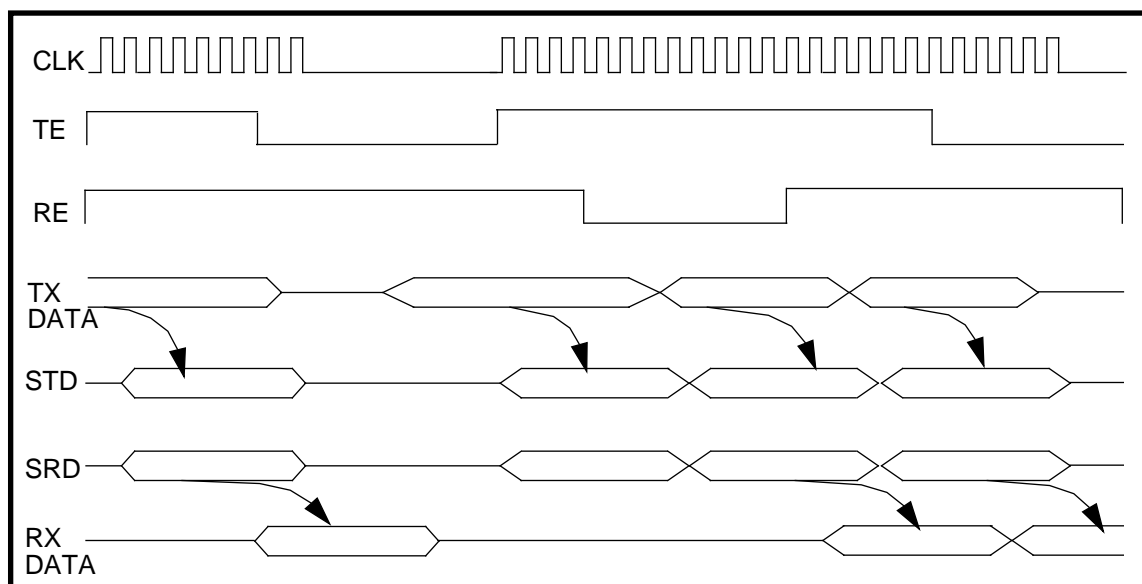
The second data word (second time slot in the frame), begins shifting in immediately after the transfer of the first data word to the RX data register. The DSP program has to read the data from the RX data register (which clears RDF) before the second data word is completely received (ready to transfer to RX data register) or a receive overrun error will occur (ROE is set).

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

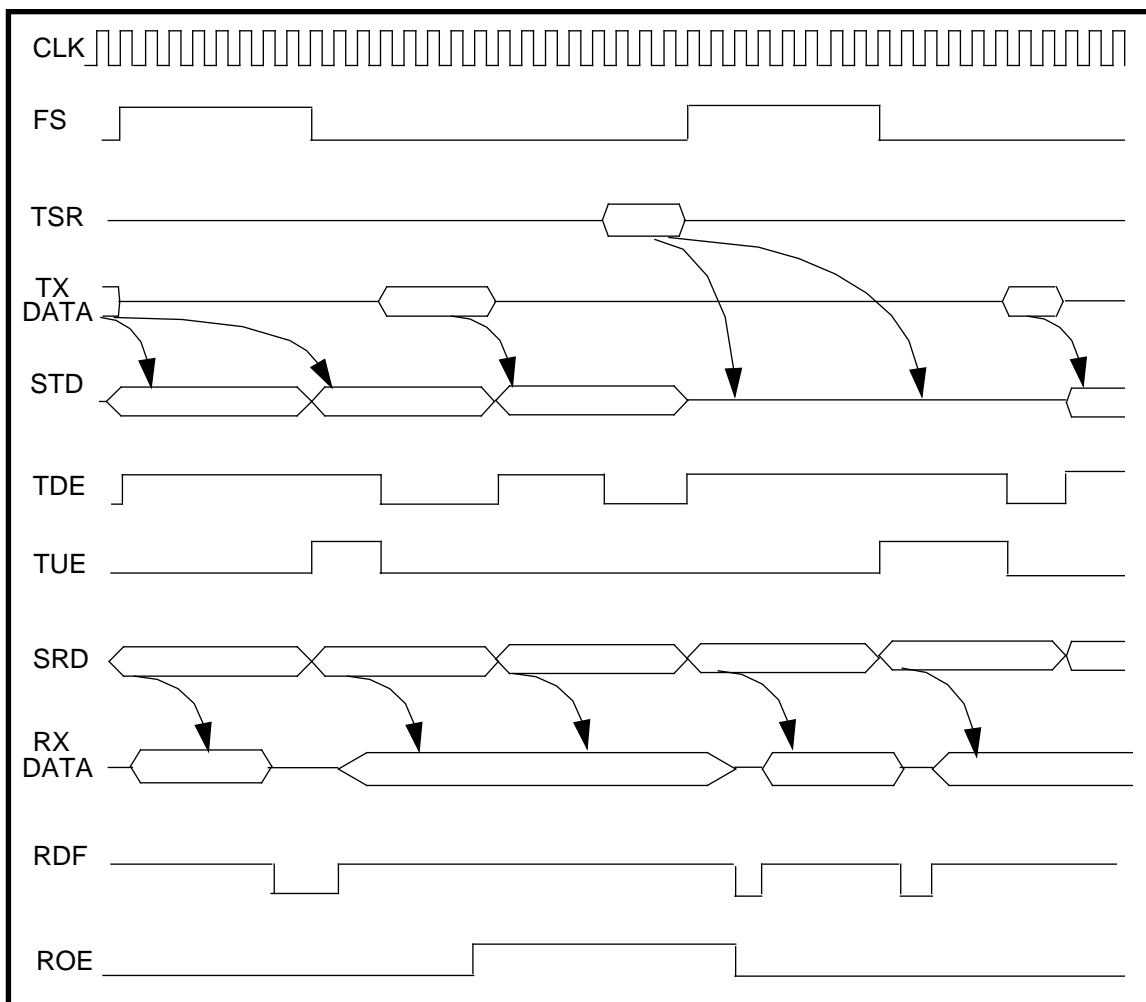
1. Read RX and use the data.
2. Read RX and ignore the data.
3. Do nothing — the receiver overrun exception will occur at the end of the current time slot.

**Note:** For a continuous clock, the optional frame sync output and clock output signals are not affected even if the transmitter and/or receiver are disabled. TE and RE do not disable the bit clock or the frame sync generation. The only way to disable the bit clock and the frame sync generation is to disable the SSI enable bit (SSIEN) in control register B. However, for a gated clock, no frame sync signals will be output and the clock signal is only enabled when TE is enabled and it is a valid time slot.

The transmitter and receiver timing for an eight bit word with three words per frame sync in network mode is shown in Figure 8-10 and Figure 8-11. A gated clock is shown in Figure 8-10 with the TE and RE bits. A continuous clock is shown in Figure 8-11 with the transmit and receive flags.



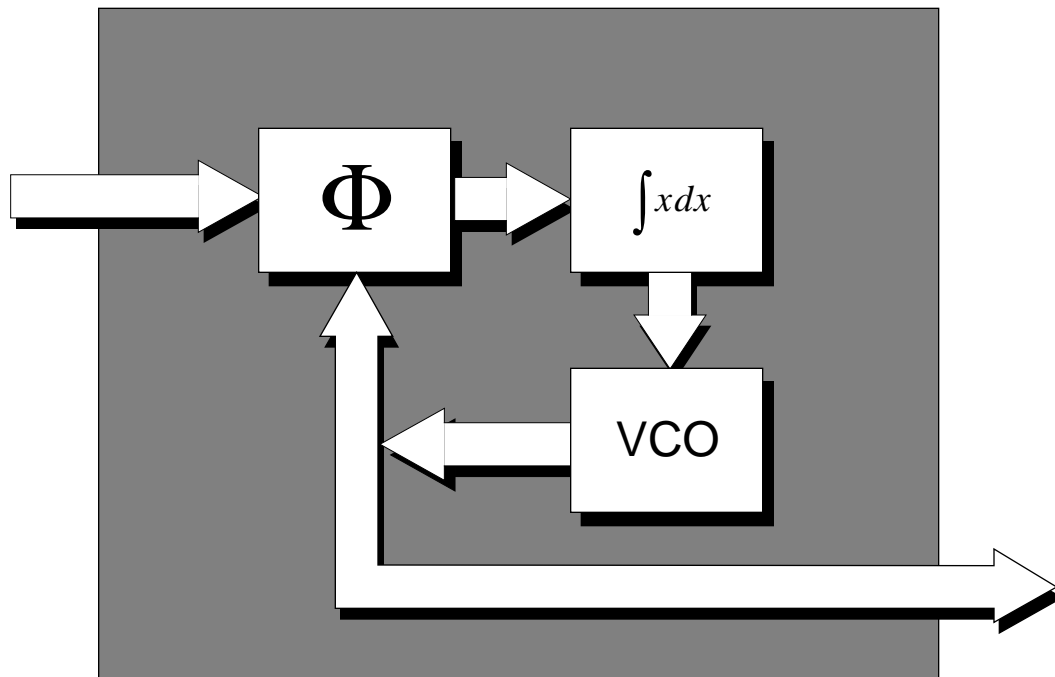
**Figure 8-10 Network Mode Timing (Gated Clock)**



**Figure 8-11 Network Mode Timing (Continuous Clock)**

## SECTION 9

### DSP56166 ON-CHIP PLL





## SECTION CONTENTS

---

9.1	INTRODUCTION .....	9-3
9.2	ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PCR0 .....	9-4
9.2.1.	PCR0 Feedback Divider Bits (YD7-YD0) Bits 0-7 .....	9-4
9.2.2.	PCR0 Input Divider Bits (ID3-ID0) Bits 8-11 .....	9-5
9.2.3.	PCR0 Power Divider Bits (PD3-PD0) Bits 12-15 .....	9-5
9.3	ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PCR1 .....	9-5
9.3.1.	PCR1 Reserved Bits — Bits 0-9 .....	9-5
9.3.2.	PCR1 CLKO Select Bits (CS1-CS0) Bit 10 and 11 .....	9-5
9.3.3.	PCR1 Phase Select Bit (PS) Bit 12 .....	9-6
9.3.4.	PCR1 PLL Power Down Bit (PLLD) Bit 13 .....	9-6
9.3.5.	PCR1 PLL Enable Bit (PLLE) Bit 14 .....	9-6
9.3.6.	PCR1 Voltage Controlled Oscillator Lock Bit (LOCK) Bit 15 .....	9-7

## 9.1 INTRODUCTION

The DSP56166 does not contain an on-chip oscillator. An external system clock must be provided through the EXTAL input pin. The on-chip phase locked loop (PLL) can be used to generate the DSP5616 core system clock or it can be bypassed allowing the DSP5616 core to directly use the clock provided on the EXTAL pin.

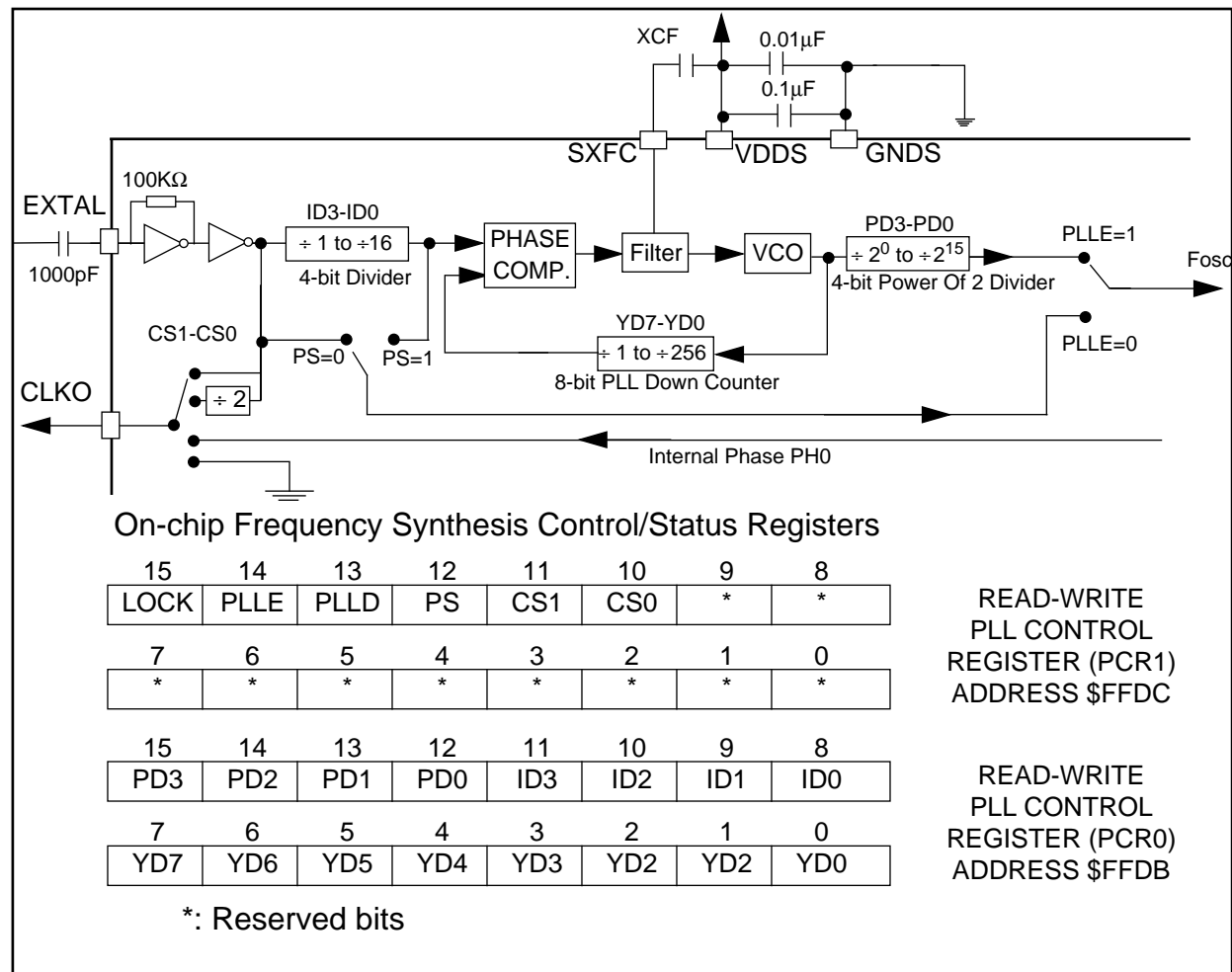
Figure 9-1 shows the general block diagram of the on-chip frequency synthesizer.

The 4-bit divider ID3-ID0 defines the resolution of the PLL and divides the incoming clock rate fed to the PLL. The eight down counter bits YD7-YD0 control down counting in the PLL feedback loop causing it to divide by the value YD+1 (any number between 1 and 256) which effectively multiplies the frequency out of the PLL. The VCO output can be divided down by any power of 2 between  $2^0$  and  $2^{15}$  before entering the core using the 4-bits PD3-PD0 of the control register PCR1. The system frequency on the DSP core is controlled by the frequency control bits of the PLL control register PCR0 as follows:

$$F_{osc} = \{F_{ext} \div [ID+1]\} \times [YD+1] \div (2^{PD})$$

where ID is the value contained in ID3-ID0, YD is the value contained in YD7-YD0, and PD is the value contained in PD3-PD0. Fext is a squared and delayed version of the clock signal applied to the EXTAL input pin.

**Note:** The STOP instruction does not power down the PLL if the PLL is enabled (PLLD=0) when entering the STOP mode. STOP will power down the ID register if the PLL is disabled (PLLD=1) when entering the STOP mode. (see Section 9.3.4).



**Figure 9-1 DSP56166 Frequency Synthesizer Block Diagram and Control Registers**

## 9.2 ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PCR0

The Clock Synthesis Control Register PCR0 is a 16-bit read/write register used to direct the operation of the on-chip clock synthesis. The PCR0 controls the frequency programming of the PLL. The PCR0 control bits are described in the following sections.

All PCR0 bits of are cleared by DSP hardware. Software reset does not affect this register.

### 9.2.1 PCR0 Feedback Divider Bits (YD7-YD0) Bits 0-7

The eight feedback divider bits YD7-YD0 control the down counter in the feedback loop, causing it to divide by the value YD+1 where YD is the value contained in the eight bits. Changing these bits requires a time delay for the Voltage Controlled Oscillator (VCO) to lock again.

The LOCK bit is cleared any time a new value is written to the YD bits.

The resulting DSP core system clock must be within the limits specified by the DSP56166 Technical Data Sheet (order number DSP56166/D). The frequency of the VCO should also remain higher than the minimum value specified in this data sheet.

### **9.2.2 PCR0 Input Divider Bits (ID3-ID0) Bits 8-11**

The four input divider bits are used to divide the input clock frequency by any number between 1 and 16. The output of the divider is used as input for the phase comparator of the PLL. If ID is the value contained in the four bits, the input clock to the PLL is divided by ID+1.

Any time a new value is written to the ID bits, the LOCK bit is cleared.

### **9.2.3 PCR0 Power Divider Bits (PD3-PD0) Bits 12-15**

The four power divider bits are used to divide the VCO output clock frequency by any power of two between  $2^0$  and  $2^{15}$  (i.e., 1, 2, 4, 8, 16, 32, ..., 16384, or 32768). The output of the divider can be used as the operating clock for the DSP core, as shown in Figure 9-1. Writing to the PD bits does not affect the LOCK condition of the PLL.

The PD bits can be used to switch the DSP core back and forth from a high MIPS rate to a very low speed, low power mode without having to wait and check for the PLL to lock on a new frequency.

## **9.3 ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PCR1**

The Clock Synthesis Control Register PCR1 is a 16-bit read/write register used to direct the operation of the on-chip clock synthesizer. The PCR1 control bits are described in the following sections.

All PCR1 bits are cleared by DSP hardware. Software reset does not affect this register.

### **9.3.1 PCR1 Reserved Bits — Bits 0-9**

These bits are reserved and should be written as zero by the user.

### **9.3.2 PCR1 CLKO Select Bits (CS1-CS0) Bits 10 and 11**

The two CLKO Select bits CS1-CS0 enable one of three possible clocks to be output to the CLKO pin when the CD bit in the OMR register is cleared (see Figure 9-1). After hardware reset, the internal DSP core clock PH0 (phase zero) is output to the CLKO pin. PH0 is a delayed version of the DSP core master clock, Fosc. Changing the value of the two bits CS1-CS0 according to Table 9-1, Fext or Fext/2 can be selected to be output on CLKO. Fext is a squared and delayed version of the signal applied to the EXTAL input pin.

Table 9-1 CLKOUT Pin Control

CS1	CS0	CLKO
0	0	PH0
0	1	Reserved
1	0	Fext
1	1	Fext/2

### 9.3.3 PCR1 Phase Select Bit (PS) Bit 12

This bit is used to select the DSP core clock when the PLL output is not selected (PLLE=0). When this bit is cleared, a squared version of EXTAL is selected as Fosc. When this bit is set, the output of the ID divider is selected as Fosc.

### 9.3.4 PCR1 PLL Power Down Bit (PLLD) Bit 13

When the PLLD bit is set, the on-chip PLL is powered down. When this control bit is cleared, the on-chip PLL is turned on. This bit should not be set when the PLLE bit is set.

If the PLL has to be turned off before entering the STOP mode, the following sequence will have to be executed before the STOP instruction:

- Clear the PLLE bit (switch back to EXTAL)
- Set the PLLD bit (power down the PLL)
- Execute the STOP instruction.

Setting the PLLD bit clears the LOCK bit. Setting the PLLD bit powers down the complete PLL block including the PD and YD registers.

### 9.3.5 PCR1 PLL Enable Bit (PLLE) Bit 14

When the PLLE bit is set, the DSP5616 core system clock is generated by the on-chip PLL. Table 9-2 summarizes the function of the three bits — PLLE, PLLD and PS. The state of the PLL is defined by the PLLD bit. When the PLLD bit is set, the PLL is in the power down mode. When the PLLD bit is cleared, the PLL is in the active mode. Before turning the PLL off, the PLLE bit should be cleared in order to by-pass the PLL. The PLL can then be put in power down mode by setting PLLD.

If the output frequency of the PLL has to be changed by re-programming the YD bits while the PLL output is used by the core (PLLE=1; PLLD=0), the following sequence of operations should be performed:

- Clear the PLLE bit to switch back to EXTAL

- Program the YD bits (only after clearing PLLE)
- Wait for the LOCK bit to be set
- Set PLLE after the LOCK bit is tested high.

Table 9-2 PLL Operations

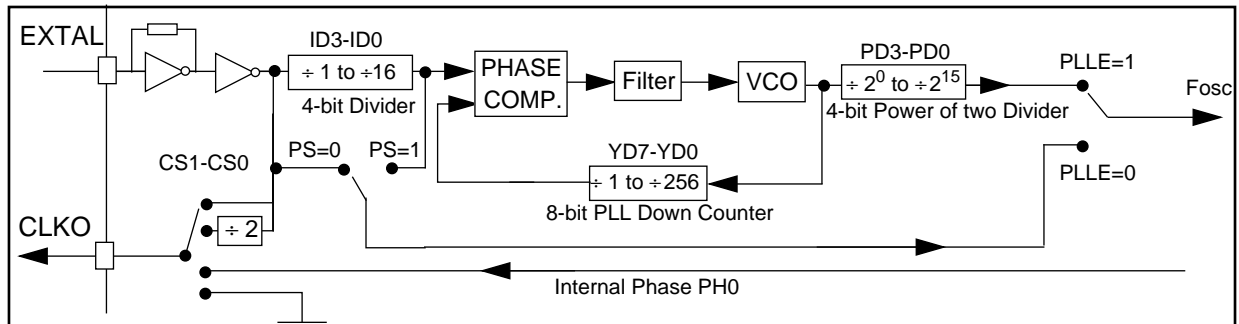
PLLE	PLLD	PS	Fosc	PLL Mode
0	0	0	Fext	Active
0	1	0	Fext	Power Down
0	0	1	$F_{ext} \div [ID+1]$	Active
0	1	1	$F_{ext} \div [ID+1]$	Power Down
1	0	x	$\{F_{ext} \div [ID+1]\} \times [YD+1] \div (2^{PD})$	Active
1	1	x	Reserved	—

### 9.3.6 PCR1 Voltage Controlled Oscillator Lock Bit (LOCK) Bit 15

This status bit shows whether the Voltage Controlled Oscillator (VCO) has locked on the desired frequency or not. When the LOCK bit is set, the VCO has locked; when the LOCK bit is cleared, the VCO has not locked yet. This bit is cleared when setting the PLLD bit and when changing the value of ID or YD bits. The LOCK bit is not cleared when clearing the PLLE bit without changing the values of PLLD, YD, or ID.

This bit is read-only and cannot be written by the DSP core.

# ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PCR1



On-chip Frequency Synthesis Control/Status Register (PCR1) ADDRESS X:\$FFDC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	PLLE	PLLD	PS	CS1	CS0	**	**	**	**	**	**	**	**	**	**

LOCK	0	PLL unlocked
	1	PLL locked
PLLE PLLD	00	PLL active but not used as Fosc
	01	PLL powered down
	10	PLL active and used as Fosc
	11	Reserved
PHASE SELECT	0	Squared EXTAL selected as Fosc if PLLE=0
	1	Squared EXTAL/ID selected as Fosc if PLLE=0
CS1-CS0 CLKO Select	00	PH0 output to CLKO when enabled by the CD bit (bit 7) of the OMR
	01	reserved
	10	Fext output to CLKO when enabled by the CD bit (bit 7) of the OMR
	11	Fext/2 output to CLKO when enabled by the CD bit (bit 7) of the OMR

On-chip Frequency Synthesis Control/Status Register (PCR0) ADDRESS X:\$FFDB

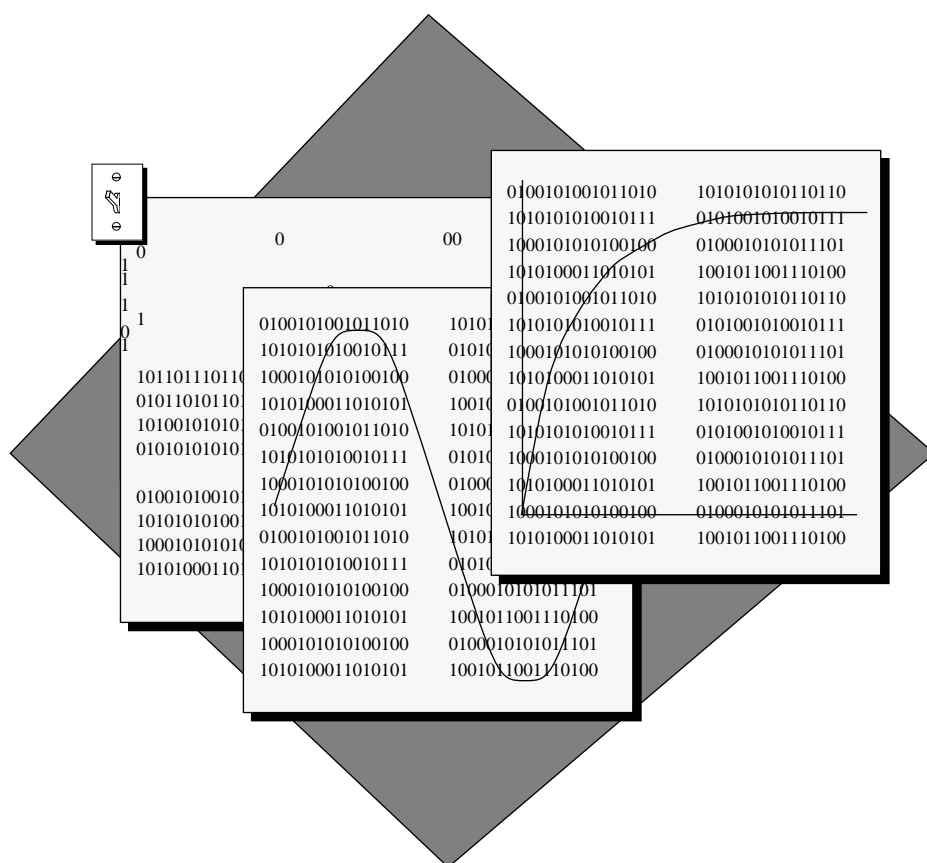
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD3	PD2	PD1	PD0	ID3	ID2	ID1	ID0	YD7	YD6	YD5	YD0	YD3	YD2	YD1	YD0

PD3-PD0 Clock Output Divider	\$0	Divide the VCO output clock by 1 ( $2^0$ )	8	Divide the VCO output clock by 256 ( $2^8$ )
	\$1	Divide the VCO output clock by 2 ( $2^1$ )	9	Divide the VCO output clock by 512 ( $2^9$ )
	\$2	Divide the VCO output clock by 4 ( $2^2$ )	A	Divide the VCO output clock by 1024 ( $2^{10}$ )
	\$3	Divide the VCO output clock by 8 ( $2^3$ )	B	Divide the VCO output clock by 2048 ( $2^{11}$ )
	\$4	Divide the VCO output clock by 16 ( $2^4$ )	C	Divide the VCO output clock by 4096 ( $2^{12}$ )
	\$5	Divide the VCO output clock by 32 ( $2^5$ )	D	Divide the VCO output clock by 8192 ( $2^{13}$ )
	\$6	Divide the VCO output clock by 64 ( $2^6$ )	E	Divide the VCO output clock by 16384 ( $2^{14}$ )
	\$7	Divide the VCO output clock by 128 ( $2^7$ )	F	Divide the VCO output clock by 32768 ( $2^{15}$ )
ID3-ID0 Input Clock Divider	\$0	Divide the input clock by 1	8	Divide the input clock by 9
	\$1	Divide the input clock by 2	9	Divide the input clock by 10
	\$2	Divide the input clock by 3	A	Divide the input clock by 11
	\$3	Divide the input clock by 4	B	Divide the input clock by 12
	\$4	Divide the input clock by 5	C	Divide the input clock by 13
	\$5	Divide the input clock by 6	D	Divide the input clock by 14
	\$6	Divide the input clock by 7	E	Divide the input clock by 15
	\$7	Divide the input clock by 8	F	Divide the input clock by 16
YD7-YD0 VCO Down Counter value	\$YD	Multiplies by YD+1		

Figure 9-2 On-Chip Frequency Synthesizer Programming Model Summary.

## SECTION A

## DSP56166 RAM BOOTSTRAP MODES





## SECTION CONTENTS

---

A.1.	INTRODUCTION.....	A-3
A.2.	BOOTSTRAP ROM.....	A-3
A.2.1.	Bootstrap Control Logic.....	A-3
A.2.2.	Bootstrap Firmware Program.....	A-4

## A.1 INTRODUCTION

The bootstrap feature of the DSP56166 consists of four special on-chip modules: the 2048 words of PRAM, a 64-word bootstrap ROM, the bootstrap control logic, and the bootstrap firmware program.

**Note:** The bootstrap feature is only available on the DSP56166 **RAM** based part. The ROM Based DSP56166 does not have the bootstrap feature available. **As a result, this appendix only applies to DSP56166 RAM based part.**

## A.2 BOOTSTRAP ROM

This 64-word on-chip ROM has been factory programmed to perform the actual bootstrap operation from the memory expansion port (Port A), the Host Interface, or the RSSI0. There is no access to the bootstrap ROM other than through the bootstrap process. Control logic will disable the bootstrap ROM during normal operations.

### A.2.1 Bootstrap Control Logic

The bootstrap mode control logic is activated when the DSP56166 is placed in Operating Mode 0 or 1. The control logic maps the bootstrap ROM into program memory space as long as the DSP56166 remains in Operating Mode 0 or Mode 1. If the DSP is in Operating Mode 0 it will load 4096 bytes from a byte-wide memory (usually an EPROM) beginning at location P:\$C000. If the DSP is in Operating Mode 1 it will load from either the Host Interface or RSSI0 depending on whether P:\$C000 bit 15 is zero or one, respectively. The bootstrap firmware changes operating modes when the bootstrap load is completed. When the DSP56166 exits the reset state in Mode 0 or 1, the following actions occur:

1. The control logic maps the bootstrap ROM into the internal DSP program memory space starting at location \$0000. This P: space is read-only.
2. The control logic forces the entire P: space, including the internal program RAM, to be write-only memory during the bootstrap loading process. Attempts to read from this space will result in fetches from the read-only bootstrap ROM.
3. Program execution begins at location \$0000 in the bootstrap ROM. The bootstrap ROM program is able to perform the PRAM load through either the memory expansion port from a byte-wide external memory, through the Host Interface, or through RSSI0.
4. The bootstrap ROM program executes the following sequence to end the bootstrap operation and begin your program execution.
  - A. Enter Operating Mode 2 by writing to the OMR. This action will be timed to remove the bootstrap ROM from the program memory map and re-enable read/write access to the PRAM.
  - B. The change to Mode 2 is timed exactly to allow the boot program to execute a single cycle instruction then a JMP #00 and begin execution of the program at location \$0000.

The bootstrap mode may also be selected by writing Operating Mode 0 or 1 into the OMR. This initiates a timed operation to map the bootstrap ROM into the program address space after a delay to allow execution of a single cycle instruction and then a JMP #<00 to begin the bootstrap process as described above in steps 1-4. This technique allows the DSP56166 user to reboot the system (with a different program if desired).

## A.2.2 Bootstrap Firmware Program

Bootstrap ROM contains the bootstrap firmware program that performs initial loading of the DSP56166 PRAM. The program is written in DSP5616 core assembly language. It contains three separate methods of initializing the PRAM: loading from a byte-wide memory starting at location P:\$C000, loading through the Host Interface, or loading serially through RSSI0. The particular method used is selected by whether (1) Operating Mode 0 or 1 is chosen and (2) the level of program memory location \$C000, bit 15.

If the DSP is in Operating Mode 0 it will load 4096 bytes from a byte-wide memory (usually an EPROM) located in the lower byte beginning at location P:\$C000 (see Figure B-1 of the applications examples given in **APPENDIX B APPLICATIONS EXAMPLES**). The data contents of the EPROM must be organized as shown below.

Address of External Byte Wide P Memory	Contents Loaded to Internal PRAM at:
P:\$C000	P:\$0000 low byte
P:\$C001	P:\$0000 high byte
•	•
•	•
•	•
P:\$CFFE	P:\$07FF low byte
P:\$CFFF	P:\$07FF high byte

If the DSP is in Operating Mode 1 and bit 15 at location P:\$C000 is low then the DSP will load from the Host Interface. Typically a host microprocessor will be connected to the DSP56166 Host Interface (see Figure B-3 of the applications examples given in **APPENDIX B — APPLICATIONS EXAMPLES**). The host microprocessor must write the Host Interface registers TXH and then TXL with the desired contents of PRAM from location P:\$0000 up to P:\$0FFF. If less than 2048 words are to be loaded, the host programmer can exit the bootstrap program and force the DSP56166 to begin executing at location P:\$0000 by setting HF0=1 in the Host Interface during the bootstrap load. In most systems, the DSP56166 responds so fast that handshaking between the DSP56166 and the host is not necessary.

If the DSP is in Operating Mode 1 and bit 15 at location P:\$C000 is high, then the DSP will load from RSSI0 starting with the least significant byte first.

The bootstrap program listing is shown in Figure A-1.

```

; Bootstrap source code for the Motorola 16-bit DSP
; (C) Copyright 1989 Motorola Inc.
;
; Host Bootstrap, RSSI0 Bootstrap and External Bus Bootstrap
;
; This is the Bootstrap program contained in the DSP56166 RAM Based. This program
; can load the internal program memory from one of 3 external sources.
; The program reads the OMR bits MA and MB to decide which external source to access.
; If MB:MA = 00 - load from 4,096 consecutive byte-wide P: memory locations (starting at P:$C000).
; If MB:MA = 01 - load internal PRAM through the Host Interface if bit 15 of P:$C000 is zero
; and load internal PRAM through RSSI0 if bit 15 of P:$C000 is set.
;
PRAMSIZE      EQU      2048                ; On-chip program RAM size
BOOT          EQU      $C000              ; The location in P: memory
                                                ; where the external byte-wide
                                                ; EPROM is to be mapped
M_PBC         EQU      $FFC0              ; Port B Control Register
M_PCC         EQU      $FFC1              ; Port C Control Register
M_HSR         EQU      $FFE4              ; Host Status Register
M_HRX         EQU      $FFE5              ; Host Receive Data Register
M_CRA0        EQU      $FFD0              ; RSSI0 Control register A
M_CRB0        EQU      $FFD1              ; RSSI0 Control register B
M_SR0         EQU      $FFF0              ; RSSI0 Status register
M_RX0         EQU      $FFF1              ; RSSI0 Serial receive register
;
;          ORG      PL:$0                  ; Bootstrap code starts at P:$0
;
;          MOVE     #M_PBC,R2              ; R2= Port B Control Register
;          MOVE     #BOOT,R1              ; R1= External P: address of
;                                         ; bootstrap byte-wide ROM
;          LEA      (R2)+,R3              ; R3= Port C control Register
;
; If this program is entered by changing the OMR to bootstrap mode, make certain that
; registers M0 and M1 have been set to $FFFF (linear addressing).
; Make sure the BCR register is set to $xxxF since EPROMs are slow.
;
; The first routine will load 4,096 bytes from the external P memory space beginning at
; P:$C000 (bits 7-0). These will be condensed into 2,048 16-bit words and stored in
; contiguous internal PRAM memory locations starting at p:$0.
; Note that the first routine loads data starting with the least significant byte of P:$0 first.
;
; The second routine loads the internal PRAM using the HOST interface logic
; or the RSSI0 interface logic
; It will load 4,096 bytes from the parallel host processor interface if bit 15 of P:$C000 is cleared
; and from the Serial Synchronous Interface RSSI0 if bit 15 of P:$C000 is set.
; These will be condensed into 2,048 16-bit words and stored in contiguous internal PRAM memory
; locations starting at P:$0. Note that when using the RSSI0, the routine loads data starting with the
; least significant byte of P:$0 first.
; If the host processor only wants to load a portion of the p memory, and then start execution of
; the loaded program, the host interface bootstrap load program routine may be killed by setting
; HF0 = 1.

```

**Figure A-1. DSP56166 Bootstrap Program Listing**

```

        MOVE    P:(R1),A          ; Get P:$C000 (clears A0)
        MOVE    A0,R0             ; R0=(0) Starting P: address of
                                   ; internal memory where program
                                   ; will begin loading
        ROL     A                  ; Shift bit 15 into the carry flag
        BCC     <INLOOP           ; Perform load from memory or
                                   ; host interface if carry is zero.
        ORI     #$40,CCR          ; Set L bit if not (0)
        MOVE    R0,X:M_CRA0       ; Set CRA0 of RSSI0 to 8 bit mode
        MOVE    #$2400,A
        MOVE    A,X:M_CRB0        ; Set CRB0 to external clock, sync.mode,
                                   ; Late FS mode with reception enabled
        BFSET   #$E,X:(R3)        ; Set PC1,PC2,PC3 to SRD0,SCK0,RFS0
;
;
INLOOP  MOVE    #PRAMSIZE,B1      ; Load PRAM size into B1
        DO      B1,_LOOP1         ; Load PRAMSIZE instruction words.

        BFTSTH  #1,OMR            ; Perform load from Host
        BCC     <_MEMLD           ; Load from memory if MA=0.
;
;
; This is the second routine. It loads from the Host Interface pins or from the RSSI0 pins.
;
        BLC     <_HOSTLD          ; Load from Host Interface
                                   ; if the limit flag is clear
;
; Bootstrap byte per byte from RSSI0
;
_RSSILD DO #2,_LOOP2
_RSSIWTBFTSTL  #$80,X:M_SR0      ; Test RDF flag
        BCS     _RSSIWT          ; Wait for RDF to go high
        MOVEP   X:M_RX0,B        ; Put receive RSSI0 data in B
        ASR4    B
        ASR4    B
        BRA     <_PACK           ; where the received byte
;
;
; This is the first routine. Its loads from external P: memory
;
;
_MEMLD DO      #2,_LOOP2          ; Each instruction has 2 bytes.
        MOVE    P:(R1)+,B        ; Get 8-bit from external P:
_PACK  MOVE    B1,A2             ; Move the 8-bit into A2
        ASR4    A                ; Shift 4 bit data into A1
        ASR4    A                ; Shift 4 bit data into A1
_LOOP2  BRA     <_STORE          ; Then put the word in P: memory

```

**Figure A-1. Listing of the DSP56166 Bootstrap Program (Continued)**

```

;
; Bootstrap from the parallel host interface
;
;
_HOSTLD  BFSET    #1,X:(R2)          ; Configure Port B as Host Interface.
_LBLA    BFTSTH   #8,X:<<M_HSR       ; Test HF0.
        BRKCS      ; Stop loading if HF0=1.
;
        BFTSTL   #1,X:M_HSR         ; Test HRDF flag
        BCS      _LBLA              ; Wait for HRDF to go high
                                         ; (meaning the data is present)
_STORE   MOVEP    X:M_HRX,A         ; Put 16-bit host data in X0
        MOVE     A,P:(R0)+          ; Store 16-bit result in PRAM
;
_LOOP1
;
        ANDI     #$FE,OMR           ; Clear OMR bit 0
        ORI      #$2,OMR           ; Set the operating mode to 2
                                         ; (and trigger an exit from
                                         ; bootstrap mode).
        AND      #$0,CCR           ; Clear SR as if HW reset and
                                         ; introduce delay needed for
                                         ; operating mode change.
;
        BRA      <$0               ; Start fetching from PRAM.
        END

```

**Figure A-1. Listing of the DSP56166 Bootstrap Program (Continued)**



---

## SECTION B

# DSP56166 APPLICATION EXAMPLES

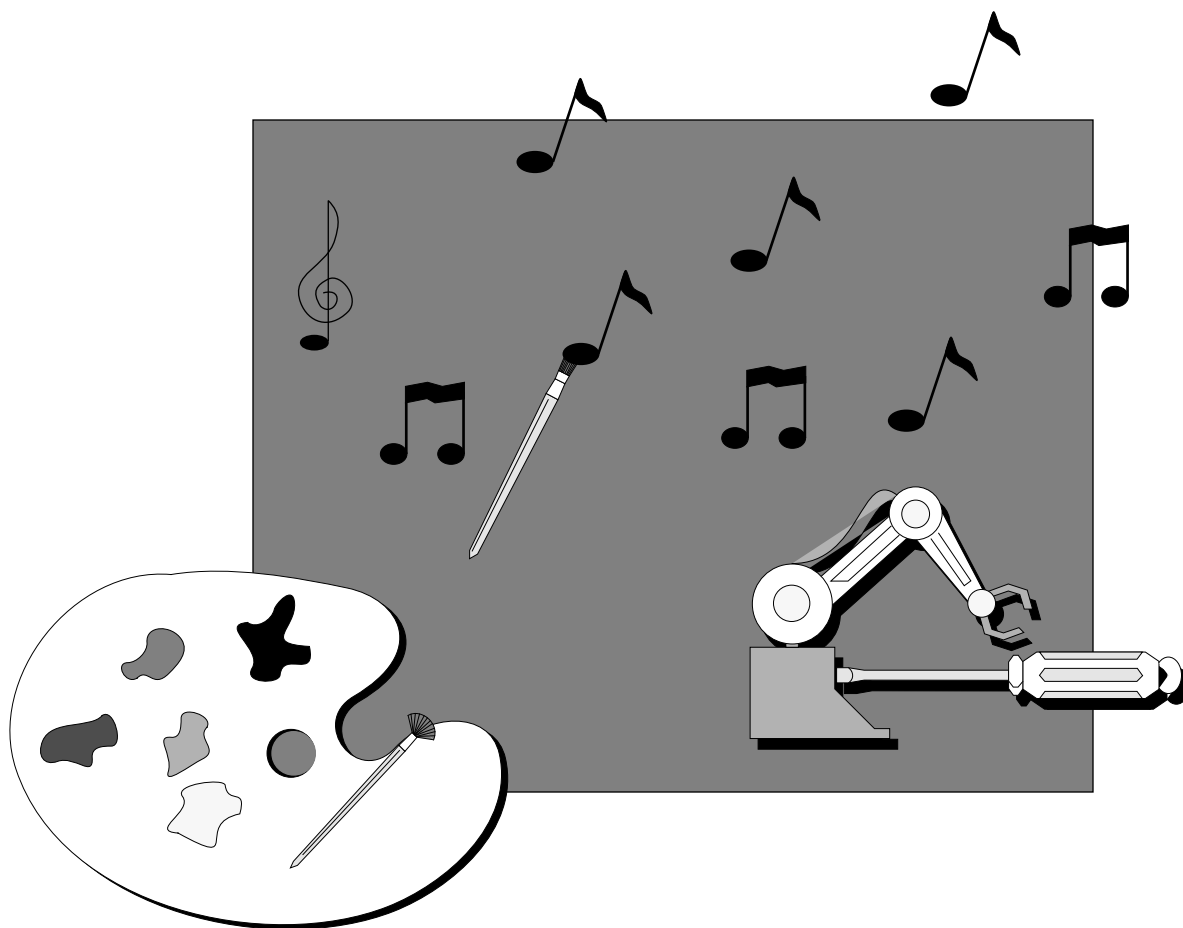
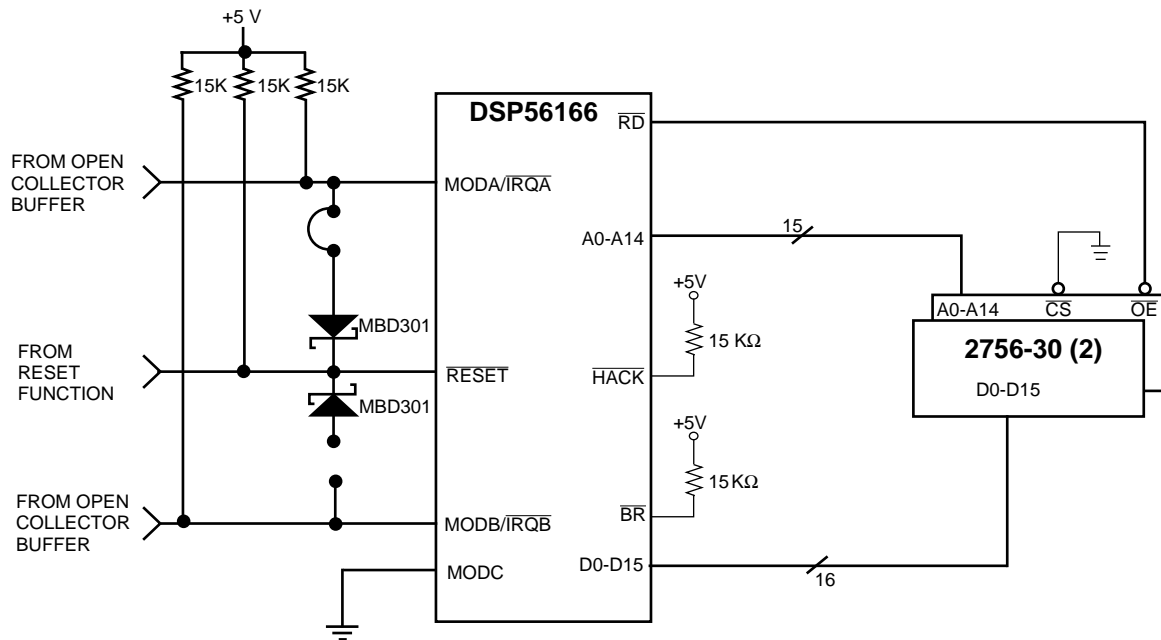




Figure B-2 shows the DSP56166 bootstrapping via the Host Interface from an MC68000.

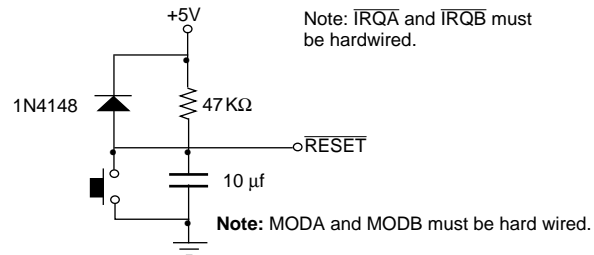


Systems with external program memory can load the on-chip PRAM without using the bootstrap mode. In Figure B-3, the DSP56166 is operated in mode 2 with the reset vector pointed at external program memory location P:\$E000. The programmer can overlay the high speed on-chip PRAM with DSP algorithms by using the MOVEM instruction.

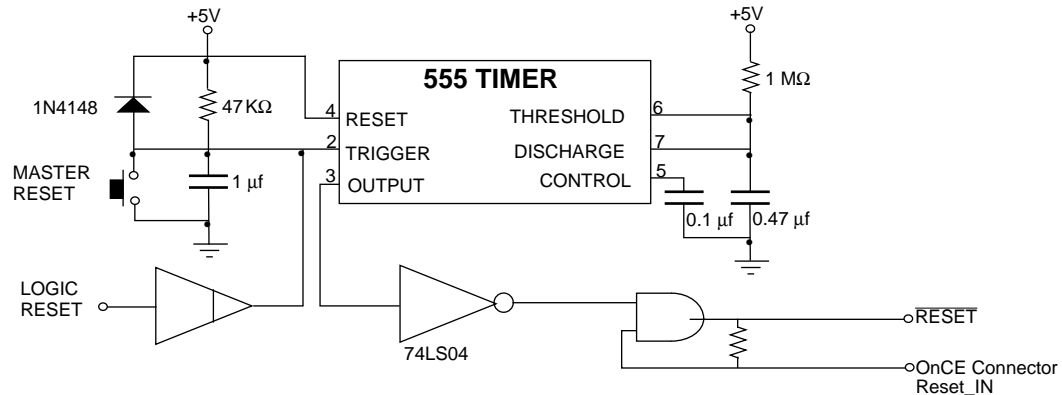


**Figure B-3 32K Words of External Program ROM — Mode 2**

Figure B-4 is a simple manual reset circuit and Figure B-5 adds the ability to remotely reset the DSP.



**Figure B-4 Reset Circuit**



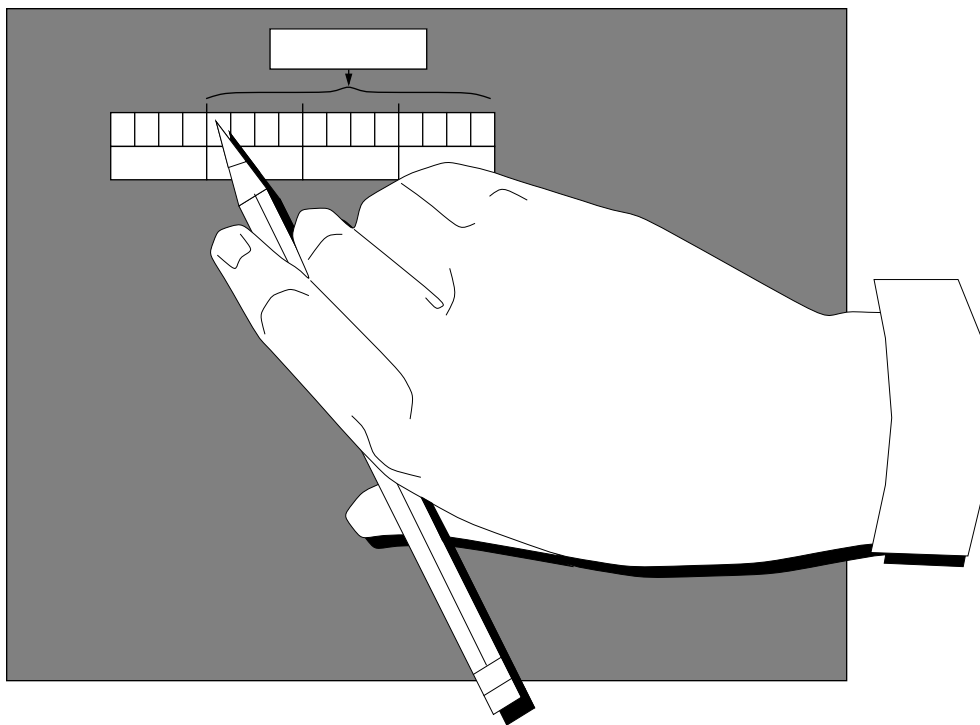
**Figure B-5 Reset Circuit Using 555 Timer**

---

## SECTION C

# DSP56166 PROGRAMMING SHEETS

The following pages are a set of programming sheets intended to be copied and used to simplify programming the various programmable registers in the DSP56166. They are grouped for the core processor and each peripheral. Each register includes the name, address, reset value, and meaning of each bit. There is room to write in the value for each bit and then the hexadecimal equivalent for each register.



# Section Contents

---

C.1.	ADDRESSES .....	C-3
C.2.	INSTRUCTIONS .....	C-5
C.3.	CORE .....	C-18
C.4.	PLL .....	C-22
C.5.	TIMER .....	C-24
C.6.	CODEC .....	C-25
C.7.	GPI/O .....	C-27
C.8.	HOST .....	C-29
C.9.	RSSI .....	C-33

## DSP56166 On-chip Peripheral Memory Map

### PERIPHERAL ADDRESSES

\$FFFF	Reserved for on-chip emulation	\$FFDF	IPR: Interrupt Priority Register
\$FFFE		\$FFDE	BCR: Bus Control Register
\$FFFD		\$FFDD	IPR2: Interrupt Priority Register 2
\$FFFC		\$FFDC	PCR1
\$FFFB		\$FFDB	PCR0
\$FFFA		\$FFDA	BCR2: Bus Control Register 2
\$FFF9	TX/RX RSSI1 TX/RX Registers	\$FFD9	CRB-RSSI1 Control Register B
\$FFF8	SR/TSR RSSI1 Status Register	\$FFD8	CRA-RSSI1 Control Register A
\$FFF7		\$FFD7	
\$FFF6		\$FFD6	
\$FFF5		\$FFD5	
\$FFF4		\$FFD4	
\$FFF3		\$FFD3	
\$FFF2		\$FFD2	
\$FFF1	TX/RX RSSI0 TX/RX Registers	\$FFD1	CRB-RSSI0 Control Register B
\$FFF0	SR/TSR RSSI0 Status Register	\$FFD0	CRA-RSSI0 Control Register A
\$FFEF	Timer Preload Register (TPR)	\$FFCF	
\$FFEE	Timer Compare Register (TCPR)	\$FFCE	
\$FFED	Timer Count Register (TCTR)	\$FFCD	
\$FFEC	Timer Control Register (TCR)	\$FFCC	
\$FFEB		\$FFCB	
\$FFEA		\$FFCA	
\$FFE9	<u>CRX/CTX</u>	\$FFC9	Reserved
\$FFE8	<u>COSR</u>	\$FFC8	CCR1
\$FFE7		\$FFC7	CCR0
\$FFE6		\$FFC6	
\$FFE5	HTX/HRX: Host TX/RX Register	\$FFC5	
\$FFE4	HSR: Host Status Register	\$FFC4	HCR: Host Control Register
\$FFE3	Port C Data Register (PCD)	\$FFC3	Port C Data Direction Register
\$FFE2	Port B Data Register (PBD)	\$FFC2	Port B Data Direction Register
\$FFE1		\$FFC1	Port C Control Register (PCC)
\$FFE0		\$FFC0	Port B Control Register (PBC)

**On-chip Peripherals Memory Map**

# INTERRUPT VECTOR ADDRESSES

Interrupt Starting Address	IPL	Interrupt Source
\$0000	3	Hardware RESET
\$0002	3	Illegal Instruction
\$0004	3	Stack Error
\$0006	3	Reserved
\$0008	3	SWI
\$000A	0-2	IRQA
\$000C	0-2	IRQB
\$000E	0-2	IRQC
\$0010	0-2	RSSI0 Receive Data with Exception Status
\$0012	0-2	RSSI0 Receive Data
\$0014	0-2	RSSI0 Transmit Data with Exception Status
\$0016	0-2	RSSI0 Transmit Data
\$0018	0-2	RSSI1 Receive Data with Exception Status
\$001A	0-2	RSSI1 Receive Data
\$001C	0-2	RSSI1 Transmit Data with Exception Status
\$001E	0-2	RSSI1 Transmit Data
\$0020	0-2	Timer Overflow
\$0022	0-2	Timer Compare
\$0024	0-2	Host DMA Receive Data
\$0026	0-2	Host DMA Transmit Data
\$0028	0-2	Host Receive Data
\$002A	0-2	Host Transmit Data
\$002C	0-2	Host Command (default)
\$002E	0-2	Codec Receive/Transmit
\$0030	0-2	Available for Host Command
\$0032	0-2	Available for Host Command
\$0034	0-2	Available for Host Command
.	.	
.	.	
.	.	
\$007E	0-2	Available for Host Command

Interrupts Starting Addresses and Sources

## INSTRUCTIONS

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
ABS	D	(parallel move) . . . . .	1	2+mv	* * * * * _
ADC	S,D	(no parallel move) . . . . .	1	2	- * * * * *
ADD	S,D	(parallel move) . . . . .	1	2+mv	* * * * * *
AND	S,D	(parallel move) . . . . .	1	2+mv	* - - ? ? 0-
AND(I)	#xx,D	. . . . .	1	2	- ? ? ? ? ?
ASL	D	(parallel move) . . . . .	1	2+mv	* * * * * ?
ASL4	D	(no parallel move) . . . . .	1	2	- ? * * * ?
ASR	D	(parallel move) . . . . .	1	2+mv	* * * * * 0?
ASR4	D	(no parallel move) . . . . .	1	2	- * * * * 0?
ASR16	D	(no parallel move) . . . . .	1	2	- * * * * 0?
BFCHG	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . . . .	2	4+mvb	- * - - - - ?
BFCLR	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . . . .	2	4+mvb	- * - - - - ?
BFSET	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . . . .	2	4+mvb	- * - - - - ?
BFTSTH	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . . . .	2	4+mvb	- * - - - - ?
BFTSTL	#iii,X:<aa> #iii,X:<pp> #iii,X:<ea> #iii,D	. . . . .	2	4+mvb	- * - - - - ?
Bcc	xxxx ee Rn	. . . . .	1+ea	4+jx	- - - - -
BRA	xxxx aa Rn	. . . . .	1+ea	4+jx	- - - - -
BRKcc		. . . . .	1	2/8	- - - - -
BSc	xxxx Rn	. . . . .	1+ea	4+jx	- - - - -
BSR	xxxx Rn	. . . . .	1+ea	4+jx	- - - - -
CHKAU		(no parallel move) . . . . .	1	2	- - - - ? ? ?-
CLR	D	(parallel move) . . . . .	1	2+mv	* * * * * 0-
CLR24	D	(parallel move) . . . . .	1	2+mv	- * * * * *
CMP	S,D	(parallel move) . . . . .	1	2+mv	* * * * * *
CMPM	S,D	(parallel move) . . . . .	1	2+mv	* * * * * *
DEBUG		. . . . .	1	4	- - - - -
DEBUGcc		. . . . .	1	4	- - - - -
DEC	D	(parallel move) . . . . .	1	2+mv	* * * * * *
DEC24	D	(parallel move) . . . . .	1	2+mv	* * * * * ?
DIV	S,D	(parallel move) . . . . .	1	2	- * - - - - ?
DMAC(ss,su,uu)S1,S2,D		(no parallel move) . . . . .	1	2	- * * * * *

Instruction Set Summary



## INSTRUCTIONS

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
DO	X:(Rn),expr #xx,expr S,expr	.....	2	6/10+mv	- * - - - - -
DOFOREVER		expr. ....	2	6	- - - - -
ENDDO		.....	1	2	- - - - -
EOR	S,D	(parallel move) .....	1	2+mv	* * - ? ? 0-
EXT	D	(no parallel move) .....	1	2	- * * * * *
ILLEGAL		(no parallel move) .....	1	8	- - - - -
IMAC	S1,S2,D	(no parallel move) .....	1	2	- * ? ? * ? ?-
IMPY	S1,S2,D	(no parallel move) .....	1	2	- * ? ? * ? ?-
INC	D	(parallel move) .....	1	2+mv	* * * * * *
INC24	D	(parallel move) .....	1	2+mv	* * * * ? *
Jcc	xxxx (Rn)	.....	1+ea	4+jx	- - - - -
JMP	xxxx (Rn)	.....	1+ea	4+jx	- - - - -
JSc	xxxx Rn	.....	1+ea	4+jx	- - - - -
JSR	xxxx	.....	1+ea	4+jx	- - - - -
LSL	D	(parallel move) .....	1	2+mv	* * - ? ? 0?
LSR	D	(parallel move) .....	1	2+mv	* * - ? ? 0?
MAC	(+)S2,S1,D	(one parallel move) .....	1	2+mv	* * * * * *
	S1,S2,D	(two parallel reads)			
	S1,S2,D	D,X:(Rn)+NnS,D			
MACR	(+)S2,S1,D	(one parallel operation) ..	1	2+mv	* * * * * *
	S1,S2,D	(two parallel reads)			
MAC(uu,su)	S1,S2,D	(no parallel move) .....	1	2	* * * * * *
MOVE		(one parallel operation) ..	1+ea	2+mv	* * - - - - -
		(double memory read)			
		(memory access, register move)			
	#xxxx,D				
No parallel data move	(.....)	.....	.mv	mv	- - - - -
Register to register	S,D(.....);	.....	.mv	mv	* ? - - - - -
data move					
Address register update	(.....)ea	.....	.mv	mv	- - - - -
X memory data move	(.....)X:<ea>,D	.....	.mv	mv	* ? - - - - -
	(.....)S,X:<ea>	.....	.mv	mv	
X memory data move	(.....)X:(R2+xx),D	.....	.mv	mv	* ? - - - - -
with short displacement	(.....)S,X:(R2+xx)	.....	.mv	mv	
X memory data write	D,X:(Rn)+NnS,D	.....	.mv	mv	* ? - - - - -
and register data					
move (MPY or MAC)					
Dual X memory data read	(.....)X:<ea>,D1 X:<ea>,D2	.....	.mv	mv	- - - - -
MOVE(C)	X:<ea>,D	.....	1+ea	2+mv	* ? ? ? ? ? ?
	S,X:<ea>				
	#xxxx,D				
	S,D				
	X:(R2+xx),D				
	S,X:(R2+xx)				
MOVE(I)	#xx,D	.....	1	2	- - - - -

Instruction Set Summary — Continued

## INSTRUCTIONS

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	SLEUNZVC
MOVE(M)	P:<ea>,D S,P:<ea> P:(R2+xx),D S,P:(R2+xx) P:<ea>,X:<ea> X:<ea>,P:<ea>	.....1+ea	1+ea	2+mv	* * - - - - -
MOVE(P)	X:<pp>,D X:<pp>,D S,X:<pp> X:<pp>,X:<ea>	.....1	1	4+mv	* * - - - - -
MOVE(S)	X:<aa>,D S,X:<aa>	.....1	1	4+mv	* * - - - - -
MPY	(+)S1,S2,D S1,S2,D S1,S2,D	(one parallel move)..... (two parallel reads) D,X:(Rn)+Nn S,D	1	2+mv	* * * * * -
MPYR	(+)S1,S2,D S1,S2,D	(one parallel move)..... (two parallel reads)	1	2+mv	* * * * * -
MPY(su,uu)	S1,S2,D	(no parallel move).....	1	2	- * * * * -
NEG	D	(parallel move).....	1	2+mv	* * * * * *
NEGC	D	(parallel move).....	1	2	- * * * * *
NOP		.....	1	2	- - - - -
NORM	Rn,D	.....	1	2	- * * * * ? -
NOT	D	(parallel move).....	1	2+mv	* * - - ? ? 0 -
OR	S,D	(parallel move).....	1	2+mv	* * - - ? ? 0 -
ORI	#xx,D	.....	1	2	- ? ? ? ? ? ?
REP	X:(Rn) #xx S	.....	1	4/6+mv	- - - - -
REPcc		.....	1	4/6	- - - - -
RESET		.....	1	4	- - - - -
RND	D	(parallel move).....	1	2+mv	* * * * * -
ROL	D	(parallel move).....	1	2+mv	* * - - ? ? 0 ?
ROR	D	(parallel move).....	1	2+mv	* * - - ? ? 0 ?
RTI		.....	1	4+rx	- ? ? ? ? ? ?
RTS		.....	1	4+rx	- - - - -
SBC	S,D	(parallel move).....	1	2+mv	* * * * * *
STOP		.....	1	n/a	- - - - -
SUB	S,D	(parallel move).....	1	2+mv	* * * * * *
SUBL	S,D	(two parallel reads) (parallel move).....	1	2+mv	* * * * * ? *
SWAP	D	(no parallel move).....	1	2	- - - - -
SWI		.....	1	8	- - - - -
Tcc	(S,D)	.....	1	2	- - - - -
TFR	S,D S,D S,D	R0,Rn (one parallel operation).. (two memory reads) (no parallel operation)...	1 1 1	2+mv	- - - - -
TFR(2)	S,D	(no parallel operation)...	1	2	- * - - - - -
TFR(3)	S1,D1 S1,D1	X:<ea>,D2 S2, X:<ea>	1	2+mv	* * - - - - -
TST	S	(parallel move).....	1	2+mv	0 * * * * 00
TST(2)	S	(no parallel move).....	1	2	- * * * * 00
WAIT		.....	1	n/a	- - - - -
ZERO	D	(no parallel move).....	1	2	- * * * * -

Instruction Set Summary — Continued

# INSTRUCTIONS

## FUNCTIONAL INSTRUCTION SET SUMMARY

### DUAL READ INSTRUCTIONS

DSP56166					
DATA ALU OPERATION		DOUBLE EFFECTIVE ADDRESS		DOUBLE DESTINATION	
Operation	Registers	Read1	Read2	Dest1	Dest2
<b>MOVE</b>		(Rn)+	(R3)+	~F	X0
<b>MAC/R MPY/R</b>	X1, Y1, F X1, Y0, F X0, Y1, F X0, Y0, F	(Rn)+Nn	(R3)+	Y0	X0
		(Rn)+	(R3)+N3	X1	X0
		(Rn)+Nn	(R3)+N3	Y1	X0
		n=[0,2]		X0	X1
<b>ADD SUB TFR</b>	X1, F X0, F Y1, F Y0, F	F = 0 → A F = 1 → B		Y0	X1
				~F	Y0
				Y1	X1
<b>ADD</b>	~F, F				
<b>SUB</b>	~F, F				
<b>TFR</b>	~F, F				

## INSTRUCTIONS

### LMS INSTRUCTION

DSP56166					
DATA ALU OPERATION		DOUBLE TRANSFER			
Operation	Registers	TRANSFER1		TRANSFER2	
<b>MAC MPY</b>	X0, X0, F	~F	(Rn)+Nn	X1	~F
	X1, X0, F	n=[0,2] F = 0 → A F = 1 → B ~F= opposite accumulator		X0	~F
	A1, Y0, F			Y1	~F
	B1, X0, F			Y0	~F
	Y0, X1, F				
	Y1, X1, F				
	Y1, X0, F				
	Y0, X0, F				

### DATA ALU INSTRUCTIONS WITH ONE PARALLEL OPERATION

DSP56166			
DATA ALU OPERATION		PARALLEL MEMORY READ or WRITE	
Operation	Registers	Effective Address	Dest/Source
<b>MAC MPY</b>	±X0, X0, F	(Rn)+ (Rn)+Nn (~F1) (R2+xx)	X1
	±X1, X0, F		X0
	±A1, Y0, F		Y1
	±B1, X0, F		Y0
	±Y0, X1, F		A0
	±Y1, X1, F		B0
	±Y1, X0, F		A
	±Y0, X0, F		B
		ONE ADDRESS UPDATE	
<b>ADD SUB TFR OR/AND EOR CMP/CPM</b>	X1, F X0, F Y1, F Y0, F	Effective Address	
		(Rn)-	
		(Rn)+Nn	
		PARALLEL REGISTER TRANSFER	
		Source	Destination
		X0	~F

# INSTRUCTIONS

## DATA ALU INSTRUCTIONS WITH ONE PARALLEL OPERATION

DSP56166			
DATA ALU OPERATION		PARALLEL MEMORY READ or WRITE	
ADD SUB	X, F Y, F	X1	~F
		Y0	~F
MOVE		Y1	~F
SBC	X, F Y, F	A	X0
		A	X1
CMP/CMPM SUBL, TFR ADD, SUB	~F, F	B	Y0
		B	Y1
RND TST ABS INC/INC24 DEC/DEC24 CLR/CLR24 NEG ASL/ASR		F	~F
		A0	X0
		A0	X1
		B0	Y0
		B0	Y1
NOT ROL/ROR LSL/LSR F		No Transfer	

# INSTRUCTIONS

## BIT FIELD MANIPULATION INSTRUCTIONS

DSP56166		
OPERATION	OPERAND	COMMENTS
<b>BFTSTH</b> #iii, <b>BFTSTL</b> #iii, <b>BFCHG</b> #iii, <b>BFSET</b> #iii, <b>BFCLR</b> #iii,	X:(Rn)	n=[0,3]
	X:<aa>	First 32 words of X memory 5 bit address
	X:<pp>	Last 32 words of X memory 5 bit address
	X1, X0, Y1, Y0, R0, R1, R2, R3, N0, N1, N2, N3 M0, M1, M2, M3 A2, B2, A1, B1, A0, B0, A, B SR, OMR, SP, SSH, SSL, LA, LC	

## EFFECTIVE ADDRESS UPDATE

DSP56166		
OPERATION	SOURCE ADDRESS REGISTER	DESTINATION REGISTER
<b>LEA</b>	(Rn) (Rn)+ (Rn)- (Rn)+Nn n=[0,3]	R0, R1, R2, R3 N0, N1, N2, N3

# INSTRUCTIONS

## JUMP/BRANCH INSTRUCTIONS

DSP56166		
OPERATION	OPERAND	COMMENTS
<b>JSR</b>	(Rn)	n=[0,3]
<b>JMP</b> <b>Jcc</b> <b>JScc</b>	\$xxxx	16-bit absolute address
<b>BSR</b>	(Rn)	n=[0,3]
<b>BRA</b> <b>Bcc</b> <b>BScc</b>	\$xxxx	16-bit absolute address
<b>JSR</b>	AA	8-bit absolute address [0,256]
<b>BRA</b>	aa	8-bit PC relative address [-128,+127]
<b>Bcc</b>	ee	6-bit PC relative address [-32,+31]

## REP and DO INSTRUCTIONS

DSP56166		
OPERATION	OPERAND	COMMENTS
<b>REP</b> <b>DO</b>	X:(Rn)	n=[0,3]
	#xx	8-bit immediate short data
	X1, X0, Y1, Y0, R0, R1, R2, R3, N0, N1, N2, N3 M0, M1, M2, M3 A2, B2, A1, B1, A0, B0, A, B SR, OMR, SP, SSH, SSL, LA, LC	
<b>REPcc</b>	16 conditions	
<b>DO FOREVER</b>		

## INSTRUCTIONS

### SHORT IMMEDIATE MOVE INSTRUCTIONS

DSP56166		
OPERATION	DESTINATION	COMMENTS
<b>MOVE(I)</b> <b>#xx,</b>	X1 X0 Y1 Y0	Immediate short 8-bit signed data (data is put in the least significant byte)

### MOVE — PROGRAM and CONTROL INSTRUCTIONS

DSP56166			
OPERATION	Source/ Destination	Destination/ Source	COMMENTS
<b>MOVE(M)</b>	P:(Rn) P:(Rn)+ P:(Rn)- P:(Rn)+Nn P:(R2+xx)	A, A0, B, B0 X0, X1, Y0, Y1	
<b>MOVE(M)</b>	X:(Rn)+ X:(Rn)+Nn	P:(Rn)+ P:(Rn)+Nn	
<b>MOVE(C)</b>	X:(Rn) X:(Rn)+ X:(Rn)- X:(Rn)+Nn X:(Rn+Nn) X:-(Rn) X:#xxxx #xxxx X:(A1) X:(B1) X:(R2+xx)	All registers	X:#xxxx: Long 16-bit Absolute address  #xxxx: Long 16-bit immediate data
<b>MOVE(C)</b>	All registers	All registers	



# INSTRUCTIONS

## MOVE ABSOLUTE SHORT AND MOVE PERIPHERAL INSTRUCTIONS

DSP56166			
OPERATION	Source/ Destination	Destination/ Source	COMMENTS
<b>MOVE(S)</b>	X:<aa>	A, B, X0, Y0	First 32 words of X memory 5 bit address
<b>MOVE(P)</b>	X:<pp>	A, B, X0, Y0	Last 32 words of X memory 5 bit address
		X:(Rn)+ X:(Rn)+Nn	

## TRANSFER WITH PARALLEL MOVE INSTRUCTION

DSP56166				
OPERATION	REGISTER TRANSFER		PARALLEL MOVE	
	Source	Destination	Source/Dest.	Destination/ Source
<b>TFR(3)</b>	A B	X0, X1, Y0, Y1	X:(Rn)+ X:(Rn)+Nn	X0,X1,Y0,Y1, A0, B0, A, B

**INSTRUCTIONS****REGISTER TRANSFER WITHOUT PARALLEL MOVE INSTRUCTION**

DSP56166		
OPERATION	SOURCE	DESTINATION
TFR(2)	A	X
	B	Y

# INSTRUCTIONS

## REGISTER TRANSFER CONDITIONAL MOVE INSTRUCTION

DSP56166		
OPERATION	Data ALU	Address Reg.
Tcc	A, F	R0, R0
	B, F Y0, F X0, F	R0, Rm

## CONDITIONAL PROGRAM CONTROLLER INSTRUCTIONS

DSP56166	
OPERATION	
BRKcc	
DEBUGcc	

## LOGICAL IMMEDIATE INSTRUCTIONS

DSP56166		
OPERATION	DESTINATION	COMMENTS
ORI     #xx, ANDI    #xx,	CCR MR OMR	8 bit immediate data

## DOUBLE PRECISION DATA ALU INSTRUCTIONS

DSP56166	
DATA ALU OPERATION	
Operation	sign    unsign

# INSTRUCTIONS

## DOUBLE PRECISION DATA ALU INSTRUCTIONS

DSP56166			
DATA ALU OPERATION			
DMAC	Y1,	X0,	F
	X1,	Y1,	F
MPY(su,uu)	X1,	Y0,	F
MAC(su,uu)	X0,	Y0,	F

## INTEGER DATA ALU INSTRUCTIONS

DSP56166	
DATA ALU OPERATION	
Operation	
IMAC	X0, X0, F
IMPY	X1, X0, F
	A1, Y0, F
	B1, X0, F
	Y0, X1, F
	Y1, X1, F
	Y1, X0, F
	Y0, X0, F

## DIVISION INSTRUCTION

DSP56166	
DATA ALU OPERATION	
Operation	
DIV	X1, F
	X0, F
	Y1, F
	Y0, F

## OTHER DATA ALU INSTRUCTIONS

DSP56166		
OPERATION		
<b>NORM</b>	Rn, F	n=[0,3]
<b>TST2</b>	X1, X0, Y1, Y0	Test data registers
<b>ADC</b>	X, F Y, F	
<b>CHKAAU</b>		Set V,N,Z according to last address ALU operation
<b>ZERO</b>	F	Zero F from bit 32 to 39
<b>EXT</b>	F	Sign extend F from bit 31 to 39
<b>SWAP</b>	F	Swap F1 and F0
<b>NEGC</b>	F	Negate with borrow
<b>ASL4</b>	F	
<b>ASR4</b>	F	
<b>ASR16</b>	F	Move A,A0 arithmetic

## SPECIAL INSTRUCTIONS

DSP56166
OPERATION
WAIT
STOP
ENDDO
RESET
RTS
RTI
SWI
DEBUG
NOP

# DSP56166 Core Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 4

## CORE

### Program Memory Wait States

Set to zero for fast memory.

### Data Memory Wait States

Set to zero for fast memory.

### Bus State Status — Read Only

0 = DSP **NOT** a Bus Master  
1 = DSP a Bus Master

### Bus Request Hold

0 = BR Asserted By External Access  
1 = BR Always Asserted

### Port A

#### Bus Control Register (BCR)

X:\$FFDE Read/Write

Reset = \$43FF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RH	BS	*	*	*	*	X4	X3	X2	X1	X0	P4	P3	P2	P1	P0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
				0				0				0			

### External Peripheral Wait States

Set to zero for fast memory.

### Port A

#### Bus Control Register 2 (BCR2)

X:\$FFDA Read/Write

Reset = \$001F

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
												0			

Carry

Overflow

Zero

Negative

Unnormalized

Extension

Limit

Sticky Bit

Interrupt Mask

Scaling Mode

Forever Flag

Loop Flag

### Status Register (SR)

Read/Write

Reset = \$0300

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LF	FV	*	*	S1	S0	I1	I0	S	L	E	U	N	Z	V	C
		0	0												

MR

CCR

\* = Reserved, Program as zero

# DSP56166 Core Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 2 of 4

## CORE

### IRQA Mode

IAL1	IAL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

IAL2	Trigger
0	Level
1	Neg. Edge

### IRQB Mode

IBL1	IBL0	Enabled	IPL
0	0	No	—
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

IBL2	Trigger
0	Level
1	Neg. Edge

### Codec IPL

0 = Lowest Level  
3 = Unmaskable

### Host IPL

0 = Lowest Level  
3 = Unmaskable

### SSI0 IPL

0 = Lowest Level  
3 = Unmaskable

### SSI1 IPL

0 = Lowest Level  
3 = Unmaskable

### Timer IPL

0 = Lowest Level  
3 = Unmaskable

### Interrupt Priority Register (IPR)

X:\$FFDF Read/Write

Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TL1	TL0	S1L1	S1L0	S0L1	S0L0	HL1	HL0	CL1	CL0	IBL2	IBL1	IBL0	IAL2	IAL1	IAL0



Application: \_\_\_\_\_  
\_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 3 of 4

# CORE

**IRQC Mode**

ICL2	ICL1	Enabled	IPL
0	0	Not enabled	—
0	1	Level	1
1	0	Not enabled	—
1	1	Neg. Edge	1

**Interrupt Priority Register 2(IPR2)**  
**X:\$FFDD Read/Write**  
**Reset = \$0000**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	ICL2	ICL1	*	*	*	*
0	0	0	0	0	0	0	0	0	0			0	0	0	0
0				0								0			

\* = Reserved, Program as zero

# DSP56166 Core Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 4 of 4

## CORE

### IRQC Mode

ICL2	ICL1	Enabled	IPL
0	0	Not enabled	—
0	1	Level	1
1	0	Not enabled	—
1	1	Neg. Edge	1

**Interrupt Priority Register 2 (IPR2)**  
X:\$FFDD Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	ICL2	ICL1	*	*	*	*
0	0	0	0	0	0	0	0	0	0			0	0	0	0
0				0								0			

### Operating Mode

00 = Boot: Byte-wide at P:\$C000  
01 = Boot: Host or SSI0  
10 = Int. Mem; Reset at P:\$E000  
11 = Ext. Mem; Reset at P:\$0000

### Bus Arbitration Mode

0 = Slave  
1 = Master

### External X Memory

0 = Internal X Memory enabled  
1 = Internal X Memory disabled

### Saturation

0 = Disable  
1 = Enable

### Rounding

0 = Convergent Rounding  
1 = Two's Complement Rounding

### Stop Delay

0 = 524K T Stabilization  
1 = 28 T Stabilization

### Clock Out

0 = Clock on CLK0 Pin  
1 = Disable

**Operating Mode Register (OMR)**  
Read/Write  
Reset = \$000x

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	CD	SD	R	SA	EX	MC	MB	MA
0	0	0	0	0	0	0	0								
0				0											

\* = Reserved, Program as zero

# DSP56166 Phase Locked Loop Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 1

## P.L.L.

### Clockout Select

CS1	CS0	CLKOUT
0	0	PH0
0	1	Reserved
1	0	Squared $F_{ext}$
1	1	Squared $F_{ext} \div 2$

### Phase Select Bit

0 = ID divider bypassed  
1 = ID divider used

### PLL Power Down

0 = Off  
1 = On

### PLL Enable

0 = Disable  
1 = Enable

### VCO Lock – Read Only

0 = NOT Locked  
1 = Locked

### PLL Control Register 1(PCR1)

X:\$FFDC Read/Write

Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Lock	PLLE	PLLD	PS	CS1	CS0	*	*	*	*	*	*	*	*	*	*
						0	0	0	0	0	0	0	0	0	0
												0		0	

### Feedback Divider

Multiplies Clock Frequency by any value from 1 to 256

### Input Divider

Divides Clock Frequency by 1 to 16

### Power Down Bits

Divide by any power of 2 between  $2^0$  and  $2^{15}$

### PLL Control Register 0 (PCR0)

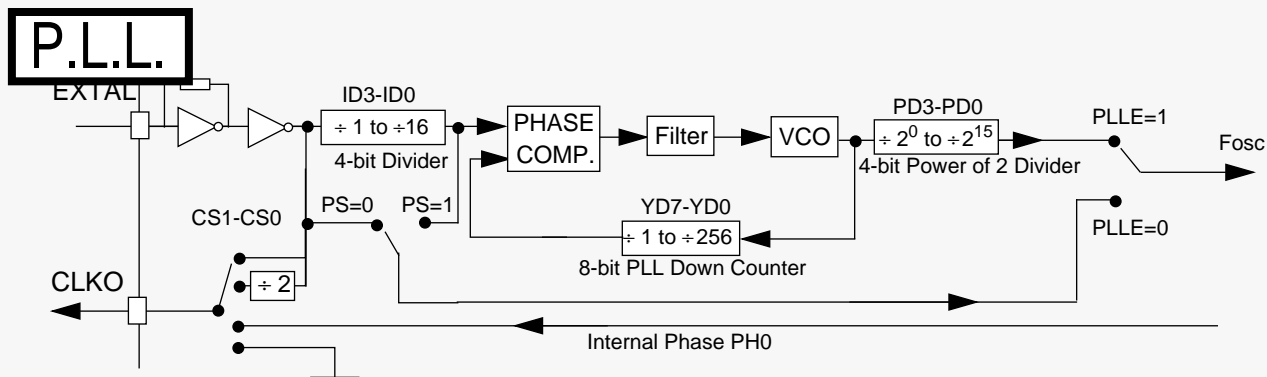
X:\$FFDB Read/Write

Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD3	PD2	PD1	PD0	ID3	ID2	ID1	ID0	YD7	YD6	YD5	YD4	YD3	YD2	YD1	YD0

\* = Reserved, Program as zero

# DSP56166 Phase Locked Loop Programming Sheet



On-chip Frequency Synthesis Control/Status Register (PCR1) ADDRESS X:\$FFDC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	PLLE	PLLD	PS	CS1	CS0	**	**	**	**	**	**	**	**	**	**

LOCK	0	PLL unlocked
	1	PLL locked
PLLE PLLD	00	PLL active but not used as Fosc
	01	PLL powered down
	10	PLL active and used as Fosc
	11	Reserved
Phase Select	0	Squared EXTAL selected as Fosc if PLL=0
	1	Squared EXTAL/ID selected as Fosc if PLL=0
CS1-CS0 CLKO Select	00	PH0 output to CLK0 when enabled by the CD bit (bit 7) of the OMR
	01	Reserved
	10	Fext output to CLK0 when enabled by the CD bit (bit 7) of the OMR
	11	Fext/2 output to CLK0 when enabled by the CD bit (bit 7) of the OMR

On-chip Frequency Synthesis Control/Status Register (PCR0) ADDRESS X:\$FFDB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD3	PD2	PD1	PD0	ID3	ID2	ID1	ID0	YD7	YD6	YD5	YD4	YD3	YD2	YD1	YD0

PD3-PD0 Clock Output Divider	\$0	Divide the VCO output clock by 1 ( $2^0$ )	8	Divide the VCO output clock by 256 ( $2^8$ )
	\$1	Divide the VCO output clock by 2 ( $2^1$ )	9	Divide the VCO output clock by 512 ( $2^9$ )
	\$2	Divide the VCO output clock by 4 ( $2^2$ )	A	Divide the VCO output clock by 1024 ( $2^{10}$ )
	\$3	Divide the VCO output clock by 8 ( $2^3$ )	B	Divide the VCO output clock by 2048 ( $2^{11}$ )
	\$4	Divide the VCO output clock by 16 ( $2^4$ )	C	Divide the VCO output clock by 4096 ( $2^{12}$ )
	\$5	Divide the VCO output clock by 32 ( $2^5$ )	D	Divide the VCO output clock by 8192 ( $2^{13}$ )
	\$6	Divide the VCO output clock by 64 ( $2^6$ )	E	Divide the VCO output clock by 16384 ( $2^{14}$ )
	\$7	Divide the VCO output clock by 128 ( $2^7$ )	F	Divide the VCO output clock by 32768 ( $2^{15}$ )
ID3-ID0 Input Clock Divider	\$0	Divide the input clock by 1	8	Divide the input clock by 9
	\$1	Divide the input clock by 2	9	Divide the input clock by 10
	\$2	Divide the input clock by 3	A	Divide the input clock by 11
	\$3	Divide the input clock by 4	B	Divide the input clock by 12
	\$4	Divide the input clock by 5	C	Divide the input clock by 13
	\$5	Divide the input clock by 6	D	Divide the input clock by 14
	\$6	Divide the input clock by 7	E	Divide the input clock by 15
	\$7	Divide the input clock by 8	F	Divide the input clock by 16
YD7-YD0 VCO Down Counter Value	\$YD	Multiplies by YD+1		

## On-chip Frequency Synthesizer Programming Model

# DSP56166 Timer Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 1

## Timer

### Timer Output Enable

000 = TOUT Disabled  
001 = Compare/Overflow Pulse  
010 = Overflow Pulse  
011 = Compare Pulse  
100 = Overflow/Compare Toggle  
101 = Compare/Overflow Toggle  
110 = Overflow Toggle  
111 = Compare Toggle

### Inverter Bit

0 = Do **NOT** Invert TIN Pin Signal  
1 = Invert TIN Pin Signal

### Timer Enable

0 = Disable Timer  
1 = Enable Timer

### Compare Interrupt Enable

0 = Disable interrupt  
1 = Interrupt DSP after TCTR = TCPR

### Overflow Interrupt Enable

0 = Disable interrupt  
1 = Interrupt DSP when TCTR = 0

### Event Select

0 = Fosc/2 is event clock  
1 = TIN is event clock

### Decrement Ratio

(Count Register Prescaler)

### Timer Control Register (TCR)

X:\$FFEC Read/Write

Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TE	INV	TO2	TO1	TO0	CIE	OIE	ES	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0

Count — Decrement when TPR = 0

### Timer Count Register (TCTR)

X:\$FFED Read/Write

Reset = \$0000

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

Compare Value — Compare with Count Register

### Timer Compare Register (TCPR)

X:\$FFEE Read/Write

Reset = \$0000

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

Number to Load into Count Register

### Timer Preload Register (TPR)

X:\$FFEF Read/Write

Reset = \$0000

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

# DSP56166 Codec Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 2

## Codec

**Input Select**  
0 = MIC Selected  
1 = AUX Selected

**Codec Enable**  
0 = Disabled  
1 = Enable

**Codec Interrupt Enable**  
0 = Disabled  
1 = Enabled

**Microphone Gain Select Bits**  
00 = -6dB  
01 = 0dB  
10 = 6dB  
11 = 17dB

**Codec Loop Back Bit**  
0 = Normal operation  
1 = A/D mod. input D/A mod.

**Clock Select Bit**  
0 = Squared Fext as Input Clock  
1 = PLL output as Input Clock

**Mute Bit**  
0 = Output Muted  
1 = Output **NOT** Muted

### Audio Level Control Bits

0000 = -15dB	1000 = 5dB
0001 = -10dB	1001 = 11dB
0010 = -5dB	1010 = 17dB
0011 = 0dB	1011 = 23dB
0100 = 5dB	1100 = 29dB
0101 = 11dB	1101 = 35dB
0110 = 17dB	1110 = 35dB
0111 = 23dB	1111 = 40dB

**Codec Control Register 1(COCR1)**  
X:\$FFC8 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COIE	COE	INS	MGS1	MGS0	MUT	*	CLS	CLB	*	*	*	VC3	VC2	VC1	VC0
						0			0	0	0				

**Codec Ratio Select Bits**  
Select any decimation/interpolation ratio values between 65 to 128

**Input Divider Bits**  
Divide the input clock to the codec by any value between 1 and 64

**Codec Control Register 0 (COCR0)**  
X:\$FFC7 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	CRS5	CRS4	CRS3	CRS2	CRS1	CRS0	*	*	ED5	ED4	ED3	ED2	ED1	ED0
0	0							0	0						

\* = Reserved, Program as zero

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 2 of 2

# Codec

## Codec Transmit Data Empty

0 = Wait  
 1 = Write Data

## Codec Transmit Underrun Error Flag

0 = OK  
 1 = Error

## Codec Receive Data Full

0 = Wait  
 1 = Read Data

## Codec Receive Overrun Error Flag

0 = OK  
 1 = Error

## Codec Status Register (COSR)

X:\$FFE8 Read Only

Reset = \$04

7	6	5	4	3	2	1	0
*	*	*	*	CRDF	CTDE	CROE	CTUE
0	0	0	0				

Load Under Program Control

## Transmit Data Register (CTX)

X:\$FFE9 Write Only

Reset = \$0000

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

Read Under Program Control

## Receive Data Register (CRX)

X:\$FFE9 Read Only

Reset = \$0000

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

\* = Reserved, Program as zero

# DSP56166 General Purpose I/O Programming Sheet

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 2

**GP I/O**

**Port B**

**Port B Control**  
 0 = General Purpose I/O  
 1 = Host Interface

**Port B  
 Control Register (PBC)**  
 X:\$FFC0 Read/Write  
 Reset = \$0000

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

**Port B Data Direction Control**  
 0 = Input  
 1 = Output

**Port B  
 Data Direction Register (PBDDR)**  
 X:\$FFC2 Read/Write  
 Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0															

**Port B Data (usually loaded by program)**

**Port B  
 Data Register (PBD)**  
 X:\$FFE2 Read/Write  
 Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	PB14	PB13	PB12	PB11	PB10	PB9	PB8	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0															

\* = Reserved, Program as zero



# DSP56166 General Purpose I/O Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 2 of 2

## GP I/O

### Port C

**Port C Pin Control**  
0 = General Purpose I/O Pin  
1 = Peripheral Pin

**Port C  
Control Register (PCC)**  
X:\$FFC1 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	CC11	CC10	CC9	*	CC7	CC6	CC5	CC4	*	CC2	CC1	CC0
0	0	0	0				0					0			
0															

**Port C Data Direction Control**  
0 = Input  
1 = Output

**Port C  
Data Direction  
Register (PCDDR)**  
X:\$FFC3 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	DC11	DC10	DC9	*	DC7	DC6	DC5	DC4	*	DC2	DC1	DC0
0	0	0	0				0					0			
0															

**Port C Data (usually loaded by program)**

**Port C  
Data Register (PCD)**  
X:\$FFE3 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	PC11	PC10	PC9	*	PC7	PC6	PC5	PC4	*	PC2	PC1	PC0
0	0	0	0				0					0			
0															

\* = Reserved, Program as zero

# DSP56166 Host Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 4

## HOST

## Port B

**Port B Control**  
0 = General Purpose I/O  
1 = Host Interface

**Port B**  
**Control Register (PBC)**  
X:\$FFC0 Read/Write  
Reset = \$0000

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

## HOST – DSP SIDE

**Host Receive Interrupt Enable**  
0 = Disable 1 = Enable — Interrupt on HRDF

**Host Transmit Interrupt Enable**  
0 = Disable 1 = Enable — Interrupt on HTDE

**Host Command Interrupt Enable**  
0 = Disable 1 = Enable — Interrupt on HCP

**Host Flags**  
General Purpose Read/Write Flags

**Host Control Register (HCR)**  
X:\$FFC4 Read/Write  
Reset = \$00

7	6	5	4	3	2	1	0
*	*	*	HF3	HF2	HCIE	HTIE	HRIE
0	0	0					

\* = Reserved, Program as zero

# DSP56166 Host Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 2 of 4

# HOST

## HOST – DSP SIDE

**Host Receive Data Full**

0 = Wait    1 = Read

**Host Transmit Data Empty**

0 = Wait    1 = Write

**Host Command Pending**

0 = Wait    1 = Ready

**Host Flags**

Read Only

**DMA Status (Read Only)**

0 = Disabled    1 = Enabled

**Host Status Register (HSR)**

X:\$FFE4 Read Only

Reset = \$02

7	6	5	4	3	2	1	0
DMA	*	*	HF1	HF0	HCP	HTDE	HRDF
	0	0					

Host Receive Data (usually Read by program)

**Host Receive Data Register (HRX)**

X:\$FFE5 Read Only

Reset = \$xxxx

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

Host Transmit Data (usually loaded by program)

**Host Transmit Data Register (HTX)**

X:\$FFE5 Write Only

Reset = \$xxxx

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

\* = Reserved, Program as zero

# DSP56166 Host Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 3 of 4

## HOST

### HOST – HOST PROCESSOR SIDE

#### Receive Request Enable

DMA Off      0 = Interrupts Disabled      1 = Interrupts Enabled  
DMA On      0 = Host → DSP      1 = DSP → Host

#### Transmit Request Enable

DMA Off      0 = Interrupts Disabled      1 = Interrupts Enabled  
DMA On      0 = DSP → Host      1 = Host → DSP

#### Host Flags

Write Only

#### Host Mode Control

00 = DMA Off      01 = Illegal  
10 = 16 Bit DMA      11 = 8 Bit DMA

#### Initialize (Write Only)

0 = No Action      1 = Initialize DMA

#### Interrupt Control Register (ICR)

\$0 Read/Write

Reset = \$00

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

#### Host Vector

Executive Interrupt Routine 0-63

#### Host Command

0 = Idle      1 = Interrupt DSP

#### Command Vector Register (CVR)

\$1 Read/Write

Reset = \$16

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

\* = Reserved, Program as zero

# DSP56166 Host Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 4 of 4

## HOST

## HOST – HOST PROCESSOR SIDE

**Receive Data Register Full**

0 = Wait      1 = Read

**Transmit Data Register Empty**

0 = Wait      1 = Write

**Transmitter Ready**

0 = Data in HI    1 = Data Not in HI

**Host Flags**

Read Only

**DMA Status**

0 = DMA Disabled      1 = DMA Enabled

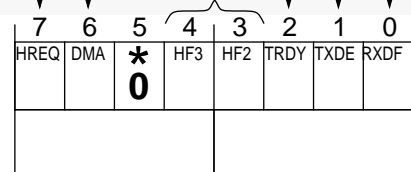
**Host Request**

0 = HREQ Deasserted    1 = HREQ Asserted

**Interrupt Status Register (ISR)**

\$2 Read/Write

Reset = \$03

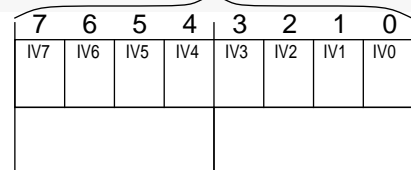


Exception vector number for use by MC68000 processor family vectored interrupts.

**Interrupt Vector Register (IVR)**

\$3 Read/Write

Reset = \$0F

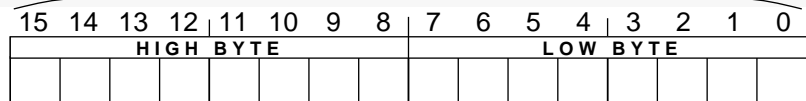


Host Receive Data (usually read by program)

**Receive Byte Registers**

\$6, \$7 Read Only

Reset = \$xxxx

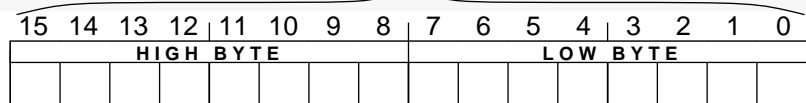


Host Transmit Data (usually loaded by program)

**Transmit Byte Registers**

\$6, \$7 Write Only

Reset = \$xxxx



\* = Reserved, Program as zero

# DSP56166 RSSI Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 3

## RSSI

**RSSI Port C Pin Control**  
0 = General Purpose I/O Pin  
1 = RSSI Pin

**PORT C**  
**RSSI Control Register (PCC)**  
X:\$FFC1 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	CC11	CC10	CC9	*	CC7	CC6	CC5	CC4	*	CC2	CC1	CC0
0	0	0	0				0					0			
0															

**RSSI Receive Data (usually read by program)**

**RSSI Serial Receive Register**  
RSSI0 Address X:\$FFF1 Read Only  
RSSI1 Address X:\$FFF9 Read Only  
Reset = \$xxxx

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

**RSSI Transmit Data (usually loaded by program)**

**RSSI Serial Transmit Register**  
RSSI0 Address X:\$FFF1 Write Only  
RSSI1 Address X:\$FFF9 Write Only  
Reset = \$xxxx

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

\* = Reserved, Program as zero

# DSP56166 RSSI Programming Sheet

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 2 of 3

## RSSI

### Word Length Control

00 = 8 Bits/Word  
01 = Reserved  
10 = 12 Bits/Word  
11 = 16 Bits/Word

### Prescaler Range

0 = / 1  
1 = / 8

### Frame Rate Divider Control

000 = 1  
111 = 8

### Prescale Modulus Select

**RSSI Control Register A (CRA)**  
RSSI0 Address \$FFD0 Read/Write  
RSSI1 Address \$FFD8 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR	WL1	WL0	* 0	* 0	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

### Clock Source Direction

0 = External Clock    1 = Internal Clock

### Clock Polarity

0 = Data Out ↑, Data In ↓    1 = Data Out ↓, Data In ↑

### MSB Position

0 = MSB First    1 = LSB First

### Frame Sync Length

0 = Word Sync    1 = Bit Sync

### Frame Sync Invert

0 = Active High    1 = Active Low

### Enable Bit

0 = RSSI Disable    1 = RSSI Enable

### Mode Select

0 = Normal    1 = Network

### Transmit Enable

0 = Disable    1 = Enable

### Receive Enable

0 = Disable    1 = Enable

### Transmit Interrupt Enable

0 = Disable    1 = Enable

### Receive Interrupt Enable

0 = Disable    1 = Enable

### FS Direction Bit:

0 = FS Input  
1 = FS Output

### Gated Clock Bit:

0 = Continuous Clock  
1 = Gated Clock

### Early FS Bit:

0 = FS on the first bit  
1 = FS one bit earlier

**RSSI Control Register B (CRB)**  
RSSI0 Address \$FFD1 Read/Write  
RSSI1 Address \$FFD9 Read/Write  
Reset = \$0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD	SSIEN	FSI	FSL	SHFD	SCKP	SCKD	FSD	GCK	ELFS	* 0	* 0

\* = Reserved, Program as zero

# DSP56166 RSSI Programming Sheet

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 3 of 3

## RSSI

### Transmit/Receive Frame Sync

0 = Wait    1 = Read RX

### Transmitter Underrun Error

0 = OK    1 = Error

### Receiver Overrun Error

0 = OK    1 = Error

### Transmit Data Register Empty

0 = Wait    1 = Write

### Receive Data Register Full

0 = Wait    1 = Read

### RSSI STATUS REGISTER (SSISR)

RSSI0 ADDRESS \$FFF0 Read Only

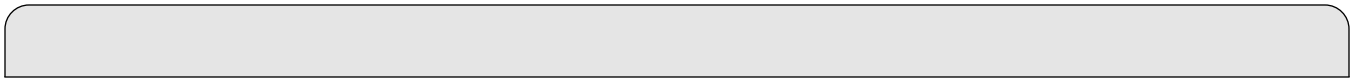
RSSI1 ADDRESS \$FFF8 Read Only

Reset = \$00

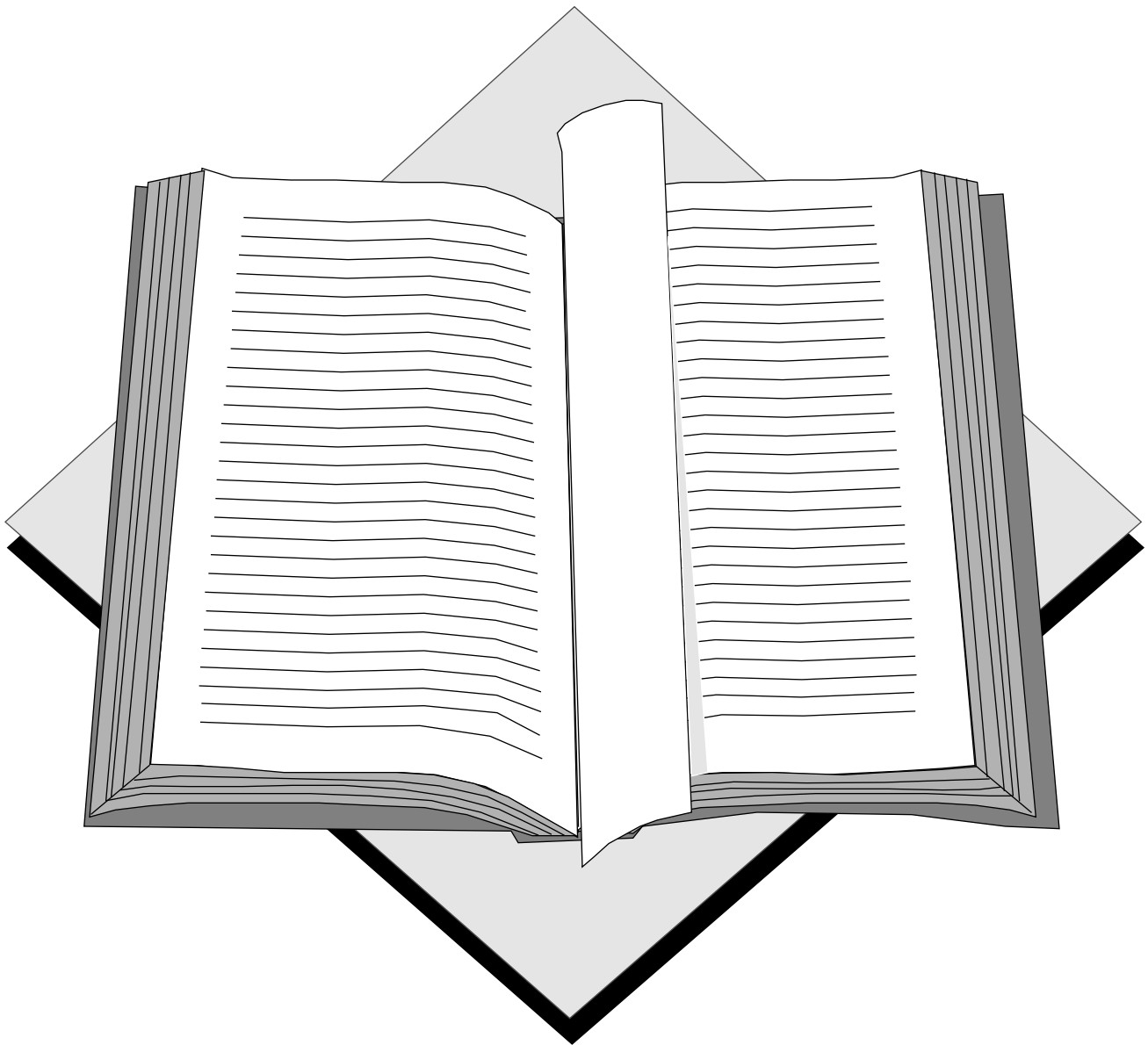
7	6	5	4	3	2	1	0
RDF	TDE	ROE	TUE	TRFS	*	*	*
					0	0	0

\* = Reserved, Program as zero





# INDEX





## Index (Continued)

### —A—

A0-A15 ..... 2-3  
 Address Bus ..... 2-3  
 Address Registers ..... 1-9  
 Analog Ground ..... 2-11  
 AUX ..... 2-15  
 Auxiliary Input ..... 2-15

### —B—

BB ..... 2-9  
 BG ..... 2-8  
 Bit Field Manipulation Instructions .. C-11  
 Bootstrap Control Logic ..... 3-7, A-3  
 Bootstrap Example, Low Cost ..... B-2  
 Bootstrap Firmware Program ..... A-4  
 Bootstrap from the External P Memory A-5  
 Bootstrap from the HI ..... A-7  
 Bootstrap from the SSI0 ..... A-6  
 Bootstrap Mode ..... 3-7  
 Bootstrap Program ..... 3-8  
 Bootstrap Program Listing ..... A-5  
 Bootstrap ROM ..... 3-7, A-3  
 BR ..... 2-7  
 BS ..... 2-5  
 Bus Busy ..... 2-9  
 Bus Control Register ..... 4-3, 4-6  
 Bus Control Register (BCR) ..... C-18  
 Bus Grant ..... 2-8  
 Bus Request ..... 2-7  
 Bus Strobe ..... 2-5

### —C—

CCR ..... 1-21  
 CIE ..... 7-7  
 CLKO ..... 2-11  
 Clock Output ..... 2-11  
 Clock Synthesis Control Register .9-4, 9-5  
 Codec Control Register (COCR) .... C-25  
 Codec Status Register (COSR) .... C-26  
 Command Vector Register ..... 5-7  
 Command Vector Register (CVR) ... C-31

Condition Code Register ..... 1-21  
 Conditional Program  
     Controller Instructions ..... C-15  
 Control Register (PBC) ..... C-27, C-29  
 Control Register (PCC) ..... C-28  
 CS1-CS0 ..... 9-5  
 CVR Host Command Bit ..... 5-9  
 CVR Host Vector ..... 5-7

### —D—

D0-D15 ..... 2-3  
 Data ALU Instructions ..... C-16  
 Data ALU Instructions  
     with One Parallel Operation .. C-9  
 Data Bus ..... 2-3  
 Data Direction Register (PBDDR) .. C-27  
 Data Direction Register (PCDDR) .. C-28  
 Data Register (PBD) ..... C-27  
 Data Register (PCD) ..... C-28  
 DC7-DC0 ..... 7-6  
 Debug Request Input ..... 2-14  
 Debug Serial Clock/Chip Status 1 .. 2-14  
 Debug Serial Input/Chip Status 0 ... 2-14  
 Debug Serial Output ..... 2-14  
 Division Instruction ..... C-16  
 DMA ..... 5-12, 5-17  
 DMA Mode Operation ..... 5-18  
 Double Precision  
     Data ALU Instructions ..... C-15  
 DR ..... 2-14  
 DSCK/OS1 ..... 2-14  
 DSI/OS0 ..... 2-14  
 DSO ..... 2-14  
 DSP Programmer Considerations .. 5-23  
 DSP to Host ..... 5-20, 5-21  
 Dual Read Instructions ..... C-8

### —E—

Effective Address Update ..... C-11  
 ES ..... 7-6  
 Exception Priorities within an IPL ... 1-13  
 EXTAL ..... 2-11

External Clock/Crystal Input . . . . . 2-11  
 External Filter Capacitor . . . . . 2-11

## —F—

Fractional Arithmetic . . . . . 1-8

## —G—

G Bus Data . . . . . 1-31  
 GDB . . . . . 1-4  
 Global Data Bus . . . . . 1-4  
 GNDS . . . . . 2-10

## —H—

H0-H7 . . . . . 2-11  
 HA0-2 . . . . . 2-11  
 HACK . . . . . 2-12  
 HC . . . . . 5-9  
 HCIE . . . . . 5-10  
 HCP . . . . . 5-11  
 HCR . . . . . 5-9  
 HCR Host Command  
     Interrupt Enable . . . . . 5-10  
 HCR Host Flag 2 . . . . . 5-10  
 HCR Host Flag 3 . . . . . 5-10  
 HCR Host Receive Interrupt Enable . 5-10  
 HCR Host Transmit Interrupt Enable 5-10  
 HEN . . . . . 2-12  
 HF0 . . . . . 5-12, 5-13  
 HF1 . . . . . 5-12, 5-14  
 HF2 . . . . . 5-10, 5-17  
 HF3 . . . . . 5-10, 5-17  
 HI . . . . . 5-3  
 HI Programming Model . . . . . 5-5  
 HM1, HM0 . . . . . 5-14  
 Host Acknowledge . . . . . 2-12  
 Host Address 0-2 . . . . . 2-11  
 Host Bootstrap Example . . . . . B-2  
 Host Control Register . . . . . 5-9  
 Host Control Register (HCR) . . . . C-29  
 Host Data Bus . . . . . 2-11  
 Host Enable . . . . . 2-12

Host Interface . . . . . 1-16, 5-3  
 Host Port Usage . . . . . 5-21  
 Host Programmer Considerations . . 5-21  
 Host Read/Write . . . . . 2-11  
 Host Receive Data Register . . . . . 5-6  
 Host Receive Data Register (HRX) . C-30  
 Host Request . . . . . 2-12  
 Host Status Register . . . . . 5-11  
 Host Status Register (HSR) . . . . . C-30  
 Host to DSP . . . . . 5-19  
 Host Transmit Data Register . . . . . 5-5  
 Host Transmit Data Register (HTX) . C-30  
 HR/W . . . . . 2-11  
 HRDF . . . . . 5-11  
 HREQ . . . . . 2-12, 5-17  
 HRIE . . . . . 5-10  
 HRX . . . . . 5-6  
 HSR DMA Status . . . . . 5-12  
 HSR Host Command Pending . . . . . 5-11  
 HSR Host Flag 0 . . . . . 5-12  
 HSR Host Flag 1 . . . . . 5-12  
 HSR Host Receive Data Full . . . . . 5-11  
 HSR Host Transmit Data Empty . . . 5-11  
 HTDE . . . . . 5-11  
 HTIE . . . . . 5-10  
 HV . . . . . 5-7

## —I—

I/O Port Set-up . . . . . 4-3  
 ICR . . . . . 5-12  
 ICR Host Flag 0 . . . . . 5-13  
 ICR Host Flag 1 . . . . . 5-14  
 ICR Host Mode Control . . . . . 5-14  
 ICR Initialize Bit . . . . . 5-15  
 ICR Receive Request Enable . . . . . 5-12  
 ICR Transmit Request Enable . . . . 5-13  
 ID3-ID0 . . . . . 9-5  
 INIT . . . . . 5-15  
 Instruction Set Summary . . . . . C-5  
 Integer Data ALU Instructions . . . . C-16  
 Integer Operations . . . . . 1-8  
 Interrupt Control Register . . . . . 5-12  
 Interrupt Control Register (ICR) . . . C-31

## Index (Continued)

Interrupt Priority Levels . . . . .	1-13	Mode Register . . . . .	1-21
Interrupt Priority Register (IPR) . . . . .	1-12, C-19, C-20, C-21	Mode Select A /External Interrupt Request A . . .	2-9
Interrupt Priority Structure . . . . .	1-10	Mode Select B /External Interrupt Request B . . .	2-9
Interrupt Status Register . . . . .	5-15	Mode Select C /External Interrupt Request C . . .	2-10
Interrupt Status Register (ISR) . . . . .	C-32	Modifier Registers . . . . .	1-9
Interrupt Vector Register . . . . .	5-17	Modulo . . . . .	1-9
Interrupt Vector Register (IVR) . . . . .	C-32	Move — Program and Control Instructions . . . . .	C-13
Interrupts Starting Addresses and Sources . . . . .	C-4	Move Absolute Short Instructions . . .	C-14
INV . . . . .	7-7	Move Peripheral Instructions . . . . .	C-14
IPL . . . . .	1-13	MR . . . . .	1-21
ISR . . . . .	5-15	Multiply-Accumulator . . . . .	1-6
ISR DMA Status . . . . .	5-17		
ISR Host Flag 2 . . . . .	5-17	—O—	
ISR Host Flag 3 . . . . .	5-17	Offset Registers . . . . .	1-9
ISR Host Request . . . . .	5-17	OIE . . . . .	7-6
ISR Receive Data Register Full . . . . .	5-16	On-chip Frequency Synthesizer Programming Model . . . . .	C-23
ISR Transmit Data Register Empty . . .	5-16	Opcode . . . . .	1-31
ISR Transmitter Ready . . . . .	5-16	Operands . . . . .	1-31
IVR . . . . .	5-17	Operating Mode Register (OMR) . . .	C-21
IVR Host Interface Interrupts . . . . .	5-18	Other Data ALU Instructions . . . . .	C-17
—J—			
Jump/Branch Instructions . . . . .	C-12		
—L—		—P—	
Linear . . . . .	1-9	PBC . . . . .	4-7
LMS Instruction . . . . .	C-9	PBD . . . . .	4-7
Logical Immediate Instructions . . . . .	C-15	PBDDR . . . . .	4-7
		PCC . . . . .	4-7
—M—		PCDDR . . . . .	4-7
MAC . . . . .	1-6	PCR0 . . . . .	9-4
MC68020 . . . . .	1-17, 5-3	PCR0 Feedback Divider Bits . . . . .	9-4
MIC . . . . .	2-15	PCR0 Input Divider Bits . . . . .	9-5
Microphone Input . . . . .	2-15	PCR0 Power Divider Bits . . . . .	9-5
MODA/IRQA . . . . .	2-9	PCR1 . . . . .	9-5
MODB/IRQB . . . . .	2-9	PCR1 CLKO Select Bits . . . . .	9-5
MODC/IRQC . . . . .	2-10	PCR1 Phase Select Bit . . . . .	9-6
Mode 0 . . . . .	3-7	PD3-PD0 . . . . .	9-5
Mode 1 . . . . .	3-7	PDB . . . . .	1-4
		PEREN . . . . .	2-5

Peripheral Enable ..... 2-5  
 Peripherals Memory Map ..... C-3  
 Pin Allocations ..... 2-3  
 PLL Control Register (PLCR) ..... C-22  
 Port B ..... 4-7  
 Port B Control Register (PBC) ..... 4-7  
 Port B Data Direction Register ..... 4-7  
 Port B Data Register ..... 4-7  
 Port C ..... 4-7  
 Port C Control Register ..... 4-7  
 Port C Data Direction Register ..... 4-7  
 Port C Data Register ..... 4-7  
 Port C Data Register (PCD) ..... 4-7  
 Port Registers ..... 4-6  
 Power Supply Input ..... 2-11  
 Program/Data Memory Select ..... 2-3  
 PS ..... 9-6  
 PS/DS ..... 2-3

## —R—

R/W ..... 2-5  
 RD ..... 2-5  
 Read Enable ..... 2-5  
 Read/Write ..... 2-5  
 Receive Byte Registers ..... 5-5, C-32  
 Receive Data Register (CRX) ..... C-26  
 Register Transfer Conditional  
   Move Instruction ..... C-15  
 Register Transfer without Parallel  
   Move Instruction ..... C-14  
 REP and DO Instructions ..... C-12  
 RESET ..... 2-10  
 Reset ..... 2-10  
 Reset Circuit ..... B-4  
 Reverse Carry ..... 1-9  
 RREQ ..... 5-12  
 RSSI0 Receive Data ..... 2-13  
 RSSI0 Serial Clock ..... 2-13  
 RSSI0 Transmit Data ..... 2-13  
 RSSI1 Receive Data ..... 2-13  
 RSSI1 Serial Clock ..... 2-13  
 RSSI1 Transmit Data ..... 2-13  
 RXDF ..... 5-16

## —S—

SCK0/PC2 ..... 2-13  
 SCK1/PC7 ..... 2-13  
 Serial Frame Sync 0 ..... 2-13  
 Serial Frame Sync 1 ..... 2-13  
 SFS0/PC4 ..... 2-13  
 SFS1/PC9 ..... 2-13  
 Short Immediate Move Instructions . C-13  
 Speaker Negative Output ..... 2-15  
 Speaker Positive Output ..... 2-15  
 Special Instructions ..... C-17  
 SPKM ..... 2-15  
 SPKP ..... 2-15  
 SRD0/PC1 ..... 2-13  
 SRD1/PC6 ..... 2-13  
 SSI Control Register (PCC) ..... C-33  
 SSI Control Register A (CRA) ..... C-34  
 SSI Control Register B (CRB) ..... C-34  
 SSI Serial Receive Register ..... C-33  
 SSI Serial Transmit Register ..... C-33  
 SSI Status Register (SSISR) ..... C-35  
 Status Register (SR) ..... C-18  
 STD0/PC0 ..... 2-13  
 STD1/PC5 ..... 2-13  
 SXFC ..... 2-11  
 Synthesizer Ground ..... 2-10  
 Synthesizer Power ..... 2-10  
 System Stack (SS) ..... 1-22

## —T—

TA ..... 2-7  
 TCR Compare Interrupt Enable ..... 7-7  
 TCR Decrement Ratio ..... 7-6  
 TCR Event Select ..... 7-6  
 TCR Inverter Bit ..... 7-7  
 TCR Overflow Interrupt Enable ..... 7-6  
 TCR Timer Enable ..... 7-8  
 TCR Timer Output Enable ..... 7-7  
 TE ..... 7-8  
 Timer Architecture ..... 7-3  
 Timer Compare Register . . . 1-16, 7-3, 7-5  
 Timer Compare Register (TCPR) . . . C-24

Timer Control Register . . . . 1-16, 7-3, 7-6  
 Timer Control Register (TCR) . . . . C-24  
 Timer Count Register . . . . . 1-16, 7-3  
 Timer Count Register (TCTR) . . . . C-24  
 Timer Functional Description . . . . . 7-8  
 Timer Input . . . . . 2-12  
 Timer Preload Register . . . . 1-16, 7-3, 7-4  
 Timer Preload Register (TPR) . . . . C-24  
 Timer Resolution . . . . . 7-8  
 TIN . . . . . 2-12  
 TO2-TO0 . . . . . 7-7  
 TOUT . . . . . 2-12  
 Transfer Acknowledge . . . . . 2-7  
 Transfer with  
     Parallel Move Instruction . . . . C-14  
 Transmit Byte Registers . . . . . 5-5, C-32  
 Transmit Data Register (CTX) . . . . C-26  
 TRDY . . . . . 5-16  
 TREQ . . . . . 5-13  
 Two's-complement . . . . . 1-8  
 TXDE . . . . . 5-16

## —Y—

YD7-YD0 . . . . . 9-4

## —V—

VDDA . . . . . 2-11  
 VDDS . . . . . 2-10  
 VDIV . . . . . 2-15  
 Voltage Division Output . . . . . 2-15  
 Voltage Reference Output for the A/D 2-15  
 Voltage Reference Output for the D/A 2-15  
 VRAD . . . . . 2-15  
 VRDA . . . . . 2-15  
 VSSA . . . . . 2-11

## —W—

Wait State . . . . . 4-7  
 WR . . . . . 2-5  
 Write Enable . . . . . 2-5


## —X—

XDB . . . . . 1-4





Order this document by DSP56166UM/AD

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

OnCE is a trade mark of Motorola, Inc.

Motorola Inc., 1994

**XC56166FE E17T Silicon Enhancement**

Issue date: April 28th 1994

***This information constitutes manual errata***

The following enhancements have been made from the E52N silicon to the E17T silicon:

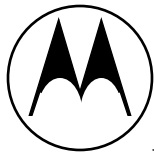
- 1- Increased PROM size from 8Kx16 to 12Kx16
- 2- Added 3 additional address pointers with corresponding address compare registers and address compared interrupt.  
Please refer to the Peripheral Address Generation Unit Revision 0.4 for details
- 3- Added Programmable Absolute Short Addressing Mode.  
Instructions affected are:
 

move(s)	X:<aa>,D
move(s)	S,X:<aa>
bfset/bfclr/bfchg/bftst	#\$iiii,X:<aa>

 Please refer to the Programmable Absolute Short Addressing Mode document for details.
- 4- Added software selectable independent external chip enable using  $\overline{\text{BR}}$  and  $\overline{\text{PEREN}}$ .  
Please refer to the Independent External Chip Enables document for details.
- 5- Added 3.6V-To-5.0V Host Port interface.
- 6- Added voltage level shifter to the CoDec so that the DSP can operate at 3.6V while the CoDec is still at 5.0V.
- 7- The CoDec's MIC and AUX input circuitry was converted from a single-ended op-amp design to a fully differential op-amp design for better power supply noise rejection.  
As a result, the input resistance of these pins have been reduced from,
 

75 Kohms	to 31.25 Kohms at 17 dB gain
	to 31.25 Kohms at 6 dB gain
	to 46.87 Kohms at 0 dB gain
	to 62.50 Kohms at -6 dB gain
- 8- Added line keeper with preset during  $\overline{\text{RESET}}$  to Port A and OnCE pins as follows:  
The Hardware  $\overline{\text{RESET}}$  initializes the pins to the indicated states.  
The keepers hold these values until an external access is performed.  
After performing the external access(s), these pins retain the last driven values (including when the bus is released or tri-stated).

<u>Pin</u>	<u>Added</u>	<u>Preset</u>	<u>Comment</u>
a[15:0]	keeper	-	
d[15:0]	keeper	-	
$\overline{\text{psds}}$	keeper	high	
$\overline{\text{peren}}$	keeper	high	
$\overline{\text{bs}}$	keeper	high	
$\overline{\text{rd}}$	keeper	high	
$\overline{\text{wr}}$	keeper	high	
$\overline{\text{rw}}$	keeper	high	
$\overline{\text{ta}}$	keeper	low	
$\overline{\text{br}}$	keeper	high	
$\overline{\text{bg}}$	keeper	high	See Note 1.
$\overline{\text{bb}}$	keeper	high	
$\overline{\text{dr}}$	keeper	high	OnCE Debug Request.



**MOTOROLA**

**DSP Division**

**DSP56166 - Rev. E17T**

---

Note: While the DSP is in the Bus Master Mode ( $OMR[2] = 1$ ) of operation, the  $\overline{bg}$  pin can be pulled low by setting  $BCR[13]$ , Bus Grant Pull Down (BGPD), bit to high.  
 $\overline{bg}$  pull-down circuit can be disabled by clearing BGPD to low.



## ***XC56166FE E17T Specification Changes***

Issue date: April 28th 1994

### ***This information constitutes manual errata***

The following specification changes were reported as problems in earlier erratas:

- 1- Single stepping through a REPcc instruction, where the instruction being repeated is executed one or more times, will result in incorrect operation of the DSP56166. The ADS program will lose communication with the DSP56166.
- 2- Single stepping through a REPcc instruction where the REP condition is true upon entering the REPCC, or a REP instruction with an initial loop count of zero will result in incorrect operation of the DSP56166. The value contained in R0, M0, and Y0 registers may be corrupted when this occurs. The DSP56166 will function correctly for the case of a DO loop with an initial loop count of zero. Likewise, single stepping the REP instruction with any initial loop count other than zero will execute properly.
- 3- When single stepping through the BRKcc instruction and the condition is true, the instruction immediately following the BRKcc instruction will be displayed by the ADS but will not be executed. Instead, the DSP56166 will correctly execute the instruction at LA+1.

Regarding 1, 2 and 3, a note will be added in the OnCE trace/step mode description of the user's manual as well as a warning in the ADS user's manual for the TRACE and STEP command.

The following are changes to the specification of the DSP56166 and should be noted in the DSP56116 User's Manual. They will be included in the new User's manual.

- No branch instructions are allowed between program memory addresses 0 and \$40. (All types of jump instructions are allowed).

- Note that for the REPcc instruction, the instruction immediately following the REPcc is NOT executed if the condition is true on entry. This means that the example on p. A-183 is not complete as shown:

Note that this is not a specification change, but rather a clarification of the specification.

```

      repnr
      norm r1,a
- should instead be-
      norm r1,a           ;Initialize condition codes in SR
      repnr
      norm r1,a

```

- When a bitfield operation is performed on a 40-bit accumulator (A or B) using a BFTSTL, BFTSTH, BFSET, BFCLR, or BFCHG instruction, the accumulator value is optionally shifted according to the scaling mode bits S0 and S1 in the system status register (SR). If the data out of the shifter indicates that the accumulator extension register is in use, the bitfield operation will be performed on the limited value (limited to the maximum positive or negative saturation constant). In addition, the L flag in the SR will be set accordingly.

- For any instructions which perform a dual read, if the second access (using R3) generates an address which lies in the peripheral memory map, this second fetch will NOT access the corresponding peripheral register but instead accesses external memory at this address. Note that this is not a specification change, but rather a clarification of the specification.

- An instruction which reads the port C data register (PCD) must not immediately follow an instruction which writes the port C data direction register (PCDDR). There must at least be one NOP between the two instructions (or some other instruction).

- An instruction which reads any of the RSSI registers must not immediately follow the RESET instruction. There must at least be one NOP between the two instructions (or some other instruction).



- The BRKcc instruction inside a 2 one-word, one-cycle instruction DO loop will not allow any interrupt to be serviced.
- The TMCTR register must not be read on the instruction immediately following an instruction which disables the timer. There must at least be one NOP between the two instructions (or some other instruction).
- The carry bit "C" will be cleared by the TST and TST2 instructions.
- Interrupts should be disabled when using the CHKAAU instruction to prevent an interrupt from corrupting the status of the address generation unit (AGU). Otherwise, the CHKAAU instruction may not correctly reflect the result of the intended address calculation. Note that the CHKAAU instruction must immediately follow the instruction which updates the AGU. The following code segment will work correctly:

```
ori  #$03,mr          ;disable interrupts
nop                   ;At least three instructions must be executed
nop                   ; between the "ori" and the instruction which
nop                   ; updates the AGU status
move x:(r0)+,a        ;user instruction which updates AGU status
chkaau
andi  #$fe,mr         ;re-enable interrupts to previous level:
                        ; - use "$fe" to allow level 2,3 interrupts
                        ; - use "$fed" to allow level 1,2,3 interrupts
                        ; - use "$fc" to allow level 0,1,2,3 interrupts
```

# DSP56166

## 16-BIT GENERAL PURPOSE DIGITAL SIGNAL PROCESSOR

This document, containing changes, additional features, further explanations, and clarifications, is an addendum to the original document:

*Document Name:* DSP56166 User's Manual  
*Order Number:* DSP56166UM/AD  
*Revision:* Original printing

Change the following:

Page A-6 - Replace the following RSSI bootcode line:

```
BFSET      #$E,X:(R3) ;Set PC1,PC2,PC3 to SRD0,SCK0,RFS0
```

with the following RSSI bootcode line:

```
BFSET      #$17,X:(R3) ;Set PC0,PC1,PC2,PC4 to STD0,SRD0,SCK0,SFS0
```

Motorola and  are registered trademarks of Motorola, Inc.



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

How to reach us:

**USA/Europe:**

Motorola Literature Distribution  
P.O. Box 20912  
Phoenix, Arizona 85036  
1 (800) 441-2447

**MFAX:**

RMFAX0@email.sps.mot.com  
TOUCHTONE (602) 244-6609

**Hong Kong:**

Motorola Semiconductors H.K. Ltd.  
8B Tai Ping Industrial Park  
51 Ting Kok Road  
Tai Po, N.T., Hong Kong  
852-2662928

**Internet:**

<http://www.motorola-dsp.com>

**Japan:**

Nippon Motorola Ltd.  
Tatsumi-SPD-JLDC  
Toshikatsu Otsuki  
6F Seibu-Butsuryu-Center  
3-14-2 Tatsumi Koto-Ku  
Tokyo 135, Japan  
03-3521-8315

**DSP Helpline:**

[dshelp@dsp.sps.mot.com](mailto:dshelp@dsp.sps.mot.com)



**MOTOROLA**