TABLE OF CONTENTS

Table of Contents (Continued)

SECTION 2
DSP56156 PIN DESCRIPTIONS

Table of Contents (Continued)

Paragraph Number	Title	Page Number

SECTION 3

OPERATING MODES AND MEMORY SPACES

Table of Contents (Continued)

Table of Contents (Continued)

SECTION 6

Table of Contents (Continued)

Paragraph Number	Title	Page Number

DSP56156 ON-CHIP SIGMA/DELTA CODEC

Table of Contents (Continued)

SECTION 7
16-BIT TIMER AND EVENT COUNTER

SECTION 8

Paragraph Number	Title	Page Number

SYNCHRONOUS SERIAL INTERFACE (SSI0 and SSI1)

Table of Contents (Continued)

SECTION 9
ON-CHIP FREQUENCY SYNTHESIZER

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

Paragraph Number	Title	Page Number

LIST of FIGURES

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

LIST of TABLES

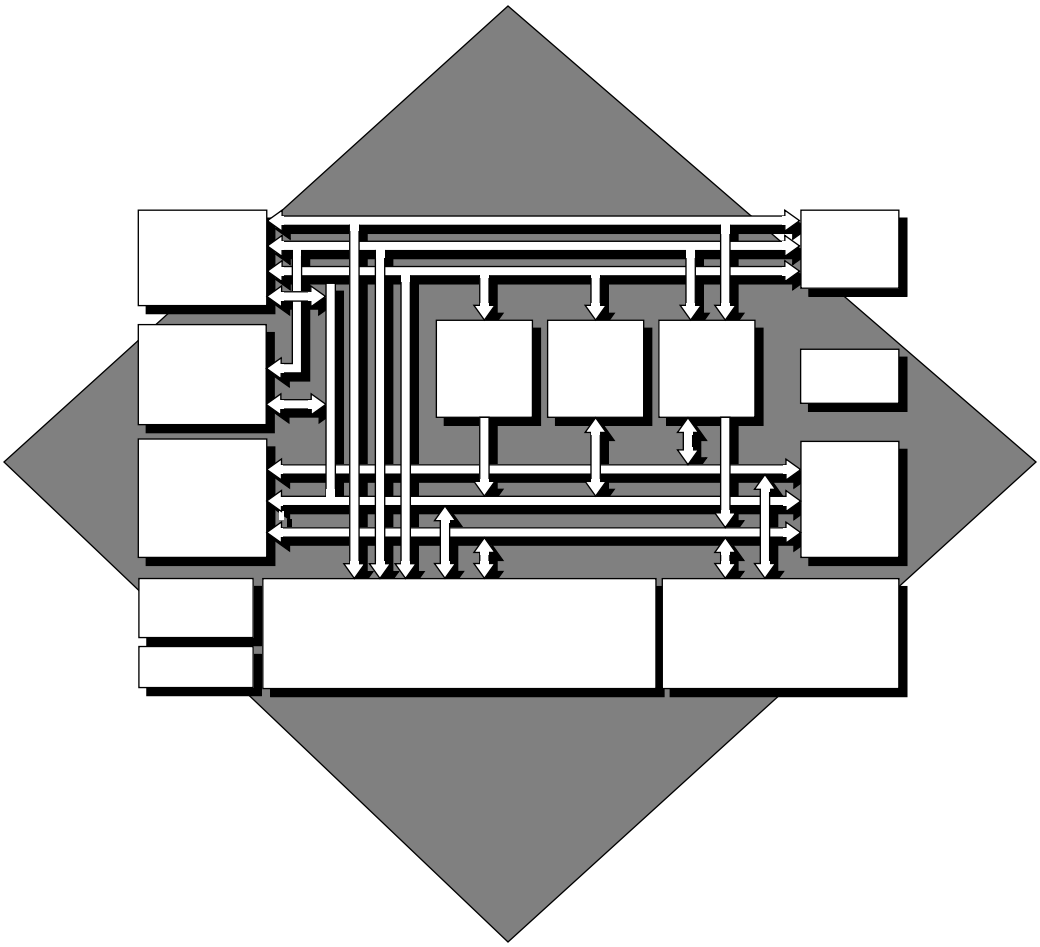
List of Tables (Continued)

List of Tables (Continued)

Table Number	Title	Page Number

SECTION 1

DSP56156 OVERVIEW



SECTION CONTENTS

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 DSP56100 CORE56100 CORE processor (see Figure 1-2) which is common to all DSP56100 family processors (except for the DSP56116) and includes a detailed description of the basic DSP56100 CORE instruction set. The *DSP56156 User's Manual* and *DSP56166 User's Manual* (see Figure 1-1) provide a brief overview of the core processor and detailed descriptions of the memory and peripherals that are chip specific. A *DSP561xx User's Manual* and a *DSP561xx Technical Data Sheet* will be available for any future DSP56100 family member. Electrical specifications and timings for the DSP56156 can be found in the *DSP56156 Technical Data Sheet.*

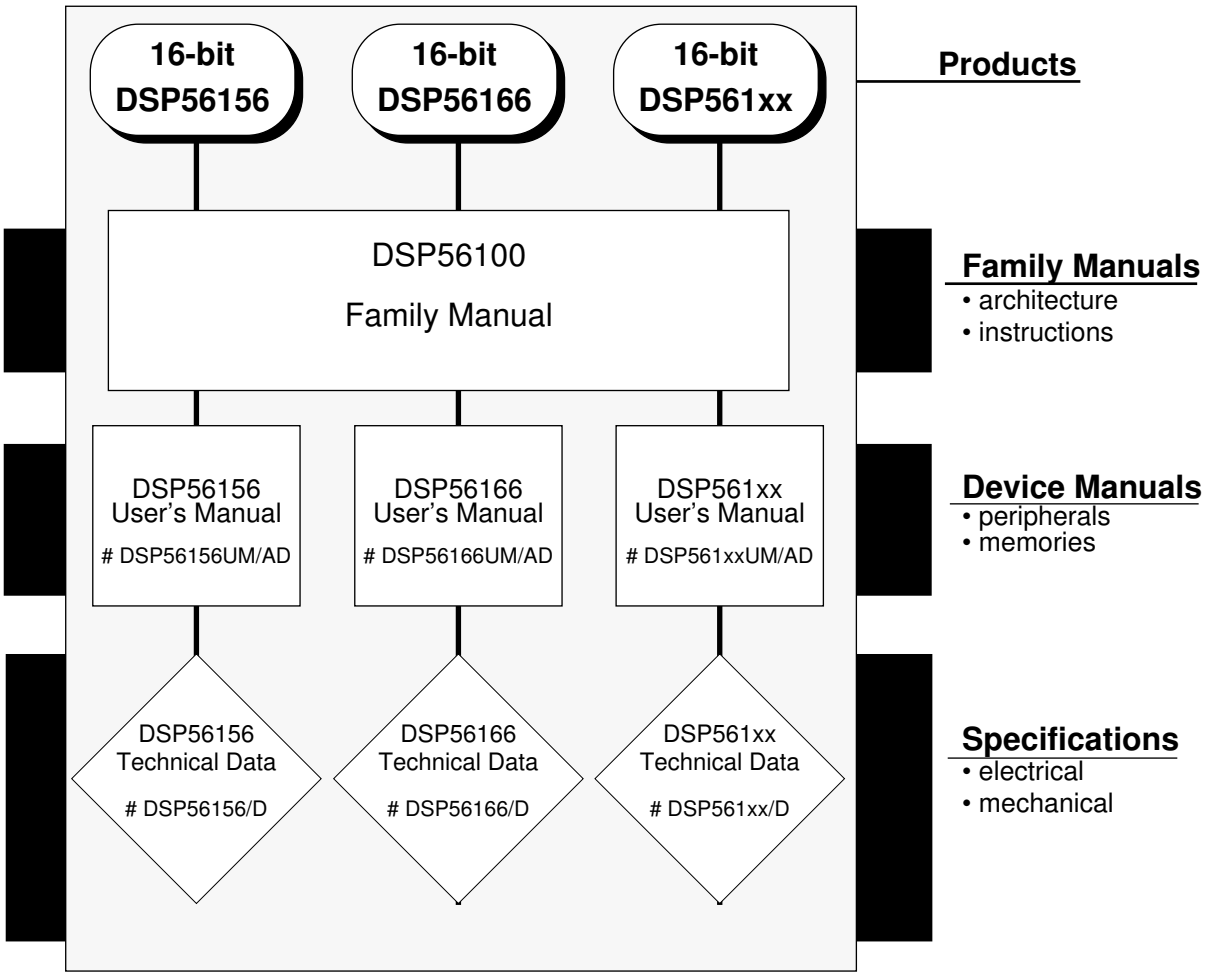


Figure 1-1 DSP56100 Family Product Literature

Table 1-1 DSP56156 Feature List

DSP56100 Family Features

- Up to 30 Million Instructions per Second (MIPS) at 60 MHz.– 33.3 ns Instruction cycle
- Single-cycle 16 x 16-bit parallel Multiply-Accumulate
- 2 x 40-bit accumulators with extension byte
- Fractional and integer arithmetic with support for multiprecision arithmetic
- Highly parallel instruction set with unique DSP addressing modes
- Nested hardware DO loops including infinite loops and DO zero loop
- Two instruction LMS adaptive filter loop
- Fast auto-return interrupts
- Two external interrupt request pins
- Three 16-bit internal data and three 16-bit internal address buses
- Individual programmable wait states on the external bus for program, and data
- On-chip memory-mapped peripheral registers
- Low Power Wait and Stop modes
- On-Chip Emulation (OnCE) for unobtrusive, processor speed independent debugging
- Operating frequency down to DC
- 5V single power supply
- Low power (HCMOS)

DSP56156 On-chip Resources — RAM Version

- 2K x 16 on-chip data RAM
- 2K x 16 on-chip program RAM
- One bootstrap ROM
- Bootstrap loading from external byte wide PROM, Host Interface, or Synchronous Serial Interface 0 (SSI0)
- One external 16-bit address bus
- One external 16-bit data bus
- On-chip peripheral registers memory mapped in data memory space
- On-chip ΣΔ voice band codec (A/D-D/A)
  - – Internal voltage reference
  - – No off-chip components required
- 27 general purpose I/O pins
- On-chip, programmable, frequency synthesizer (PLL)
- Byte-wide Host Interface with DMA support
- Two independent synchronous serial interfaces
  - – built in μ-law and A-law compression/expansion
  - – up to 32 software selectable time slots in network mode
- One 16-bit timer
- 112 pin quad flat pack packaging

DSP56156ROM On-chip Resources

- 2K x 16 on-chip data RAM
- 8K x 16 on-chip program ROM
- One external 16-bit address bus
- One external 16-bit data bus
- On-chip peripheral registers memory mapped in data memory space
- On-chip ΣΔ voice band codec (A/D-D/A)
  - – No off-chip components required
  - – Internal voltage reference (2/5 of positive power supply)
- 27 general purpose I/O pins
- On-chip, programmable PLL
- Byte-wide Host Interface with DMA support
- Two independent synchronous serial interfaces
- One 16-bit timer
- 112 pin quad flat pack packaging

Operational Differences of the ROM Based Part from the RAM Based Part

- PROM area
  P:$2F00 — P:$2FFF is reserved and should not be programmed or accessed by the user

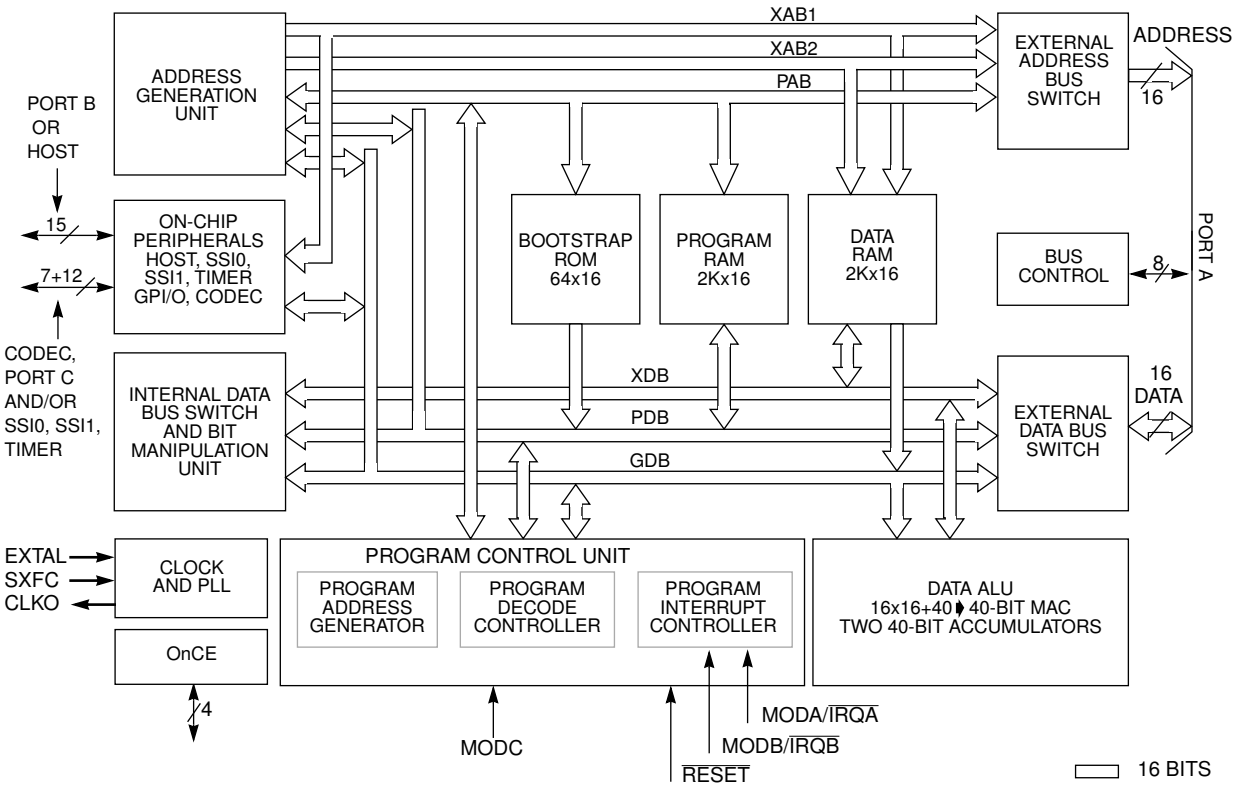


Figure 1-2 Detailed RAM Based Part Block Diagram

A general block diagram of the DSP56156 is shown in Figure 1-2 (see Section 3.2 for information on the ROM based part). The DSP56156 is optimized for applications such as medium to low bit rate speech encoding but can also be used in many other types of applications.

Table 1-1 is a list of the DSP56156 primary features. The core features are common to any product using the DSP56100 CORE processor.

1.2 DSP56100 CORE BLOCK DIAGRAM DESCRIPTION

The heart of the DSP56156 architecture is a 16-bit multiple-bus core processor called the DSP56100 CORE and designed specifically for real-time digital signal processing (DSP). The overall architecture is presented here and can be seen in Figure 1-5. For a detailed description of the core processor, see the *DSP56100 Family Manual*.

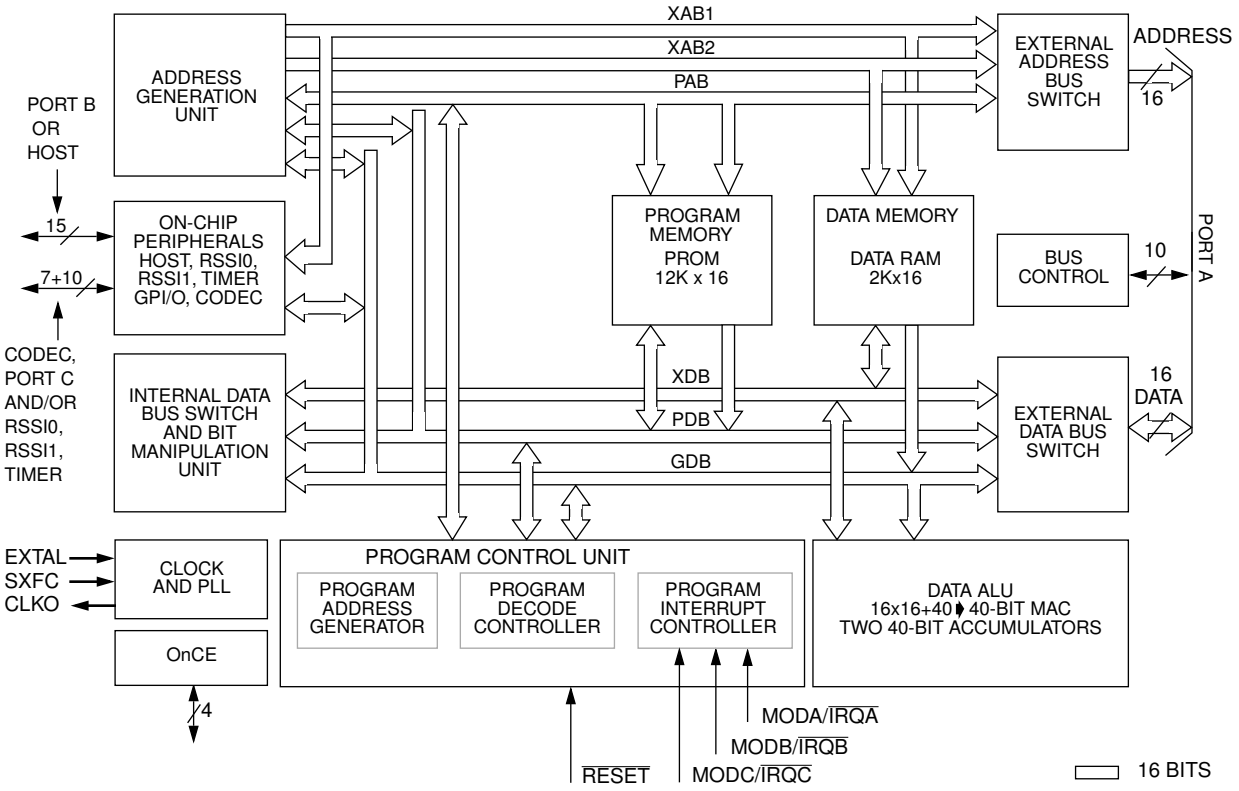


Figure 1-3 Detailed ROM Based Part Block Diagram

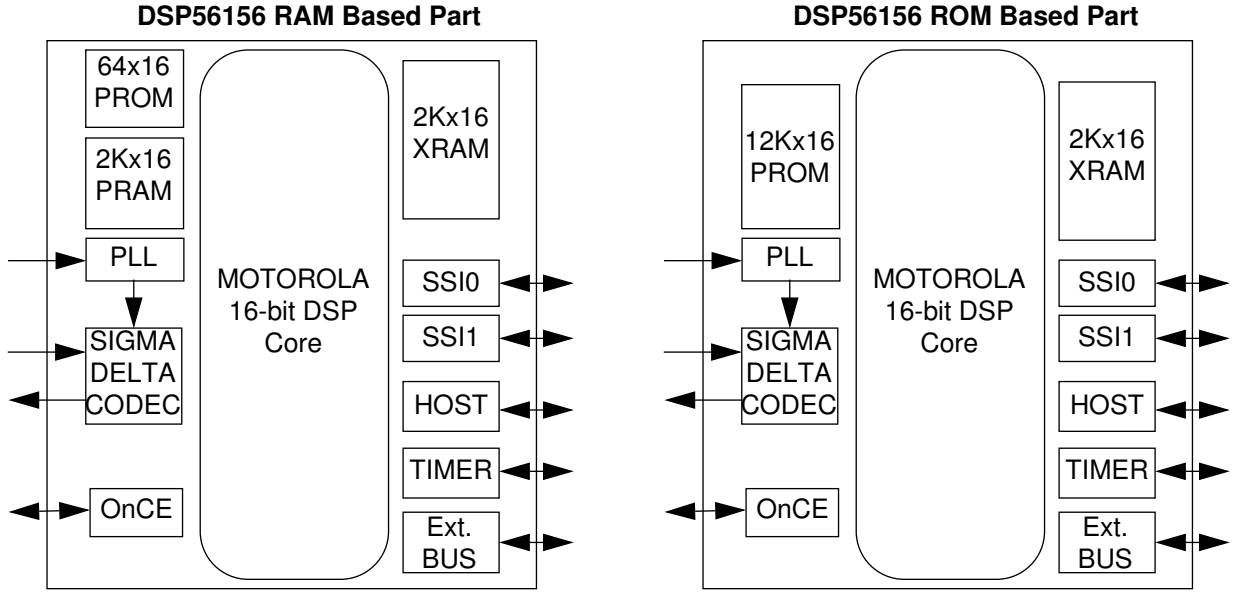


Figure 1-4 DSP56156 RAM and ROM Based Functional Block Diagram

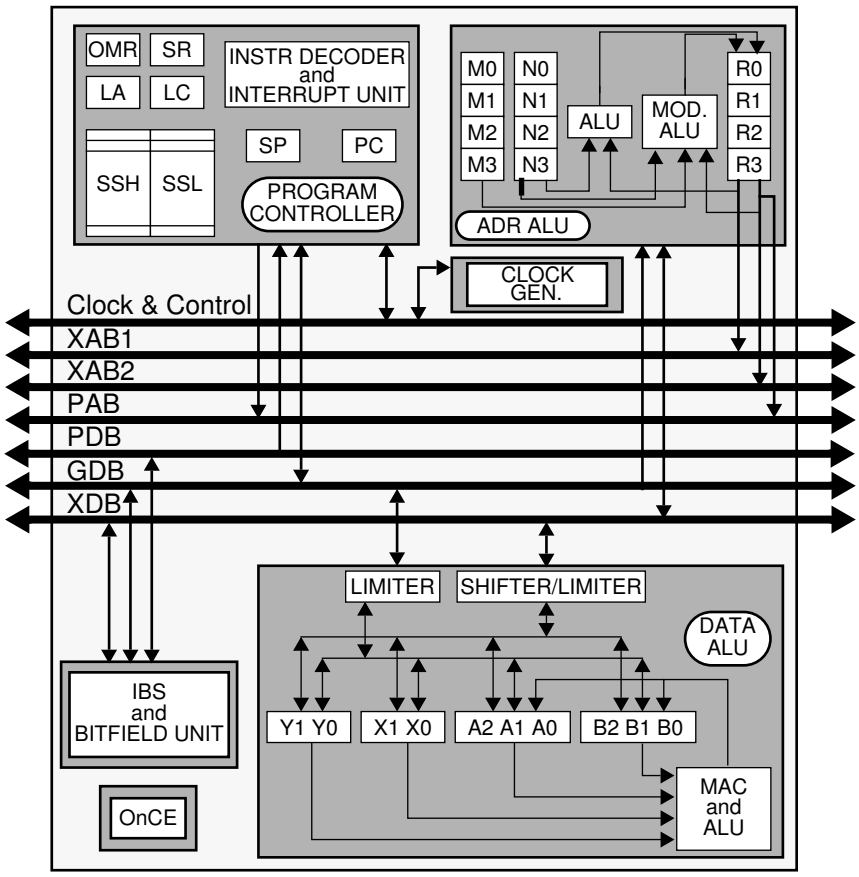


Figure 1-5 DSP56100 CORE Block Diagram

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, 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 prefetches 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.

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 Manual*** for additional information.

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
- a parallel, single cycle, non-pipelined Multiply-Accumulator (MAC) unit

Data ALU registers may be read or written on the XDB and GDB as 16-bit operands (see Figure 1-6). 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.

The DSP56156 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 instructions IMPY or IMAC.

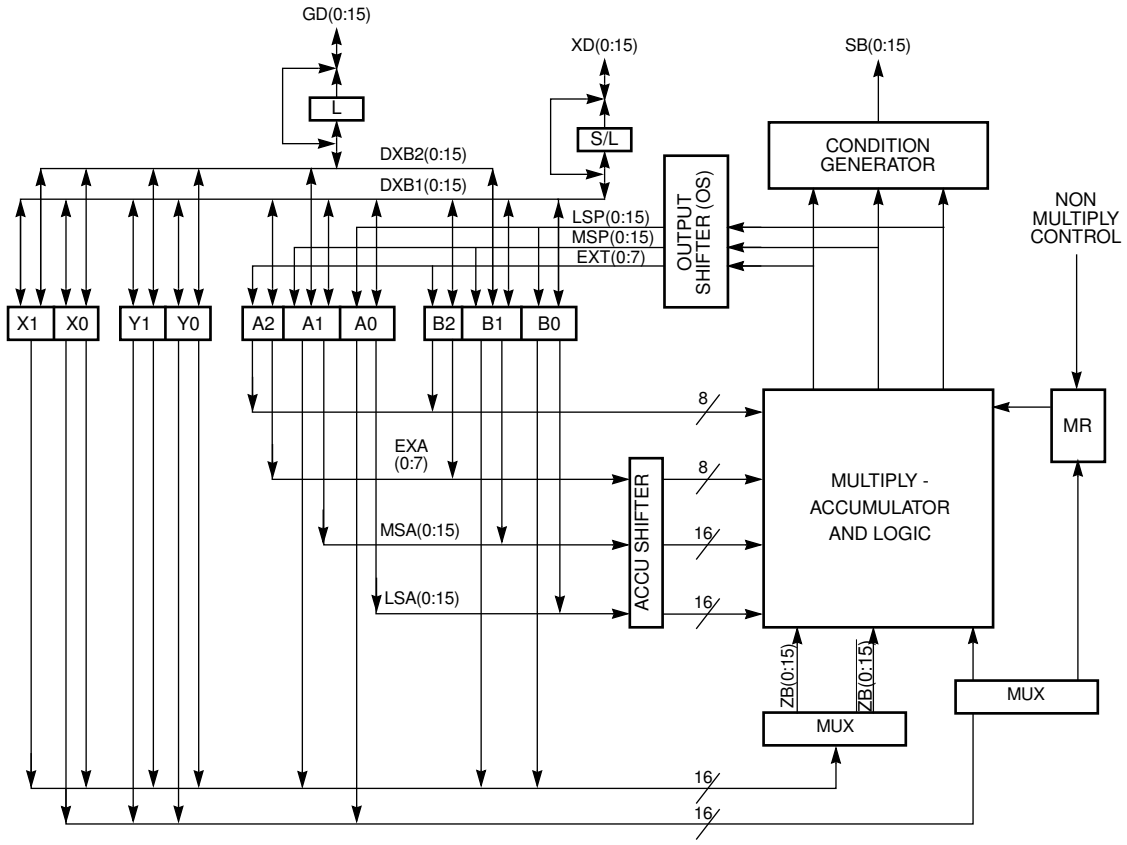


Figure 1-6 Data ALU Architecture Block Diagram

Saturation arithmetic is provided to selectively limit overflow when reading a data ALU accumulator register.

The DSP56156 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 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 1-7). 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 con-

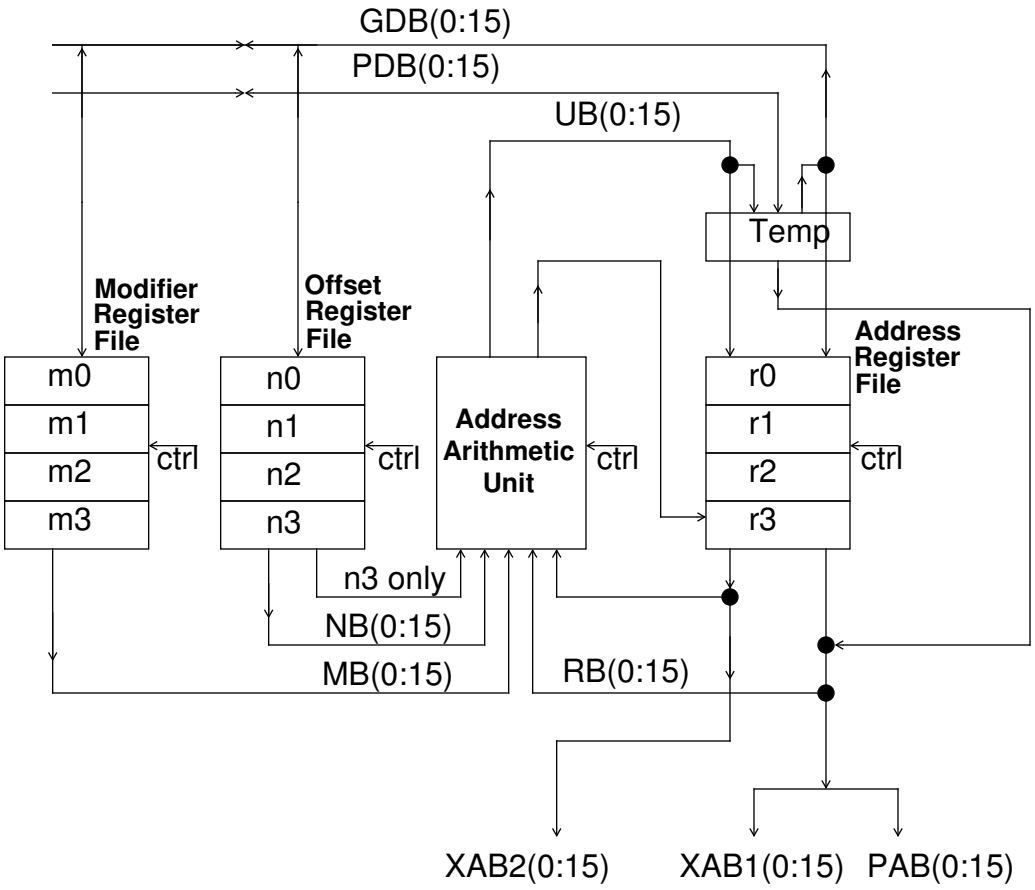


Figure 1-7 AGU Block Diagram

tains 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 address 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.

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 XAB and 65536 locations on the XAB2. See the ***DSP56100 Family 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. The interrupt priority register (IPR) (used to program the

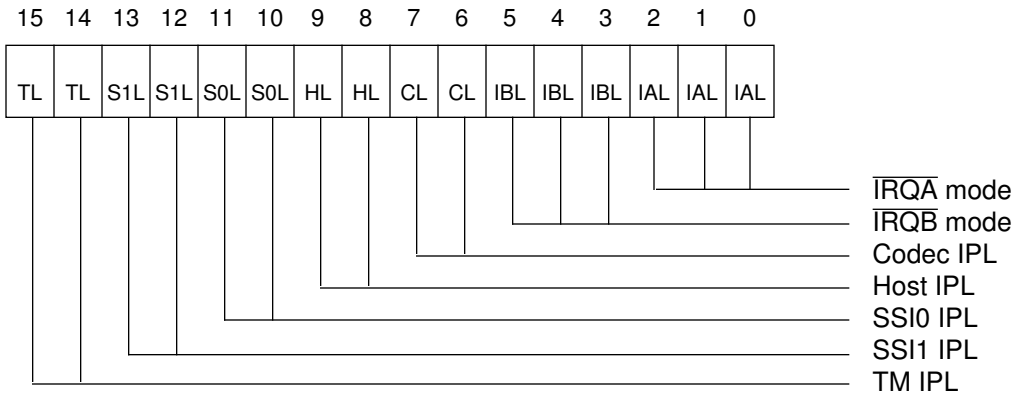


Figure 1-8 Interrupt Priority Register IPR (Address X:$FFDF)

Table 1-2
Status Register Interrupt Mask Bits

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

Table 1-3
Interrupt Priority Level Bits

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

Table 1-4
External Interrupt Trigger Mode Bits

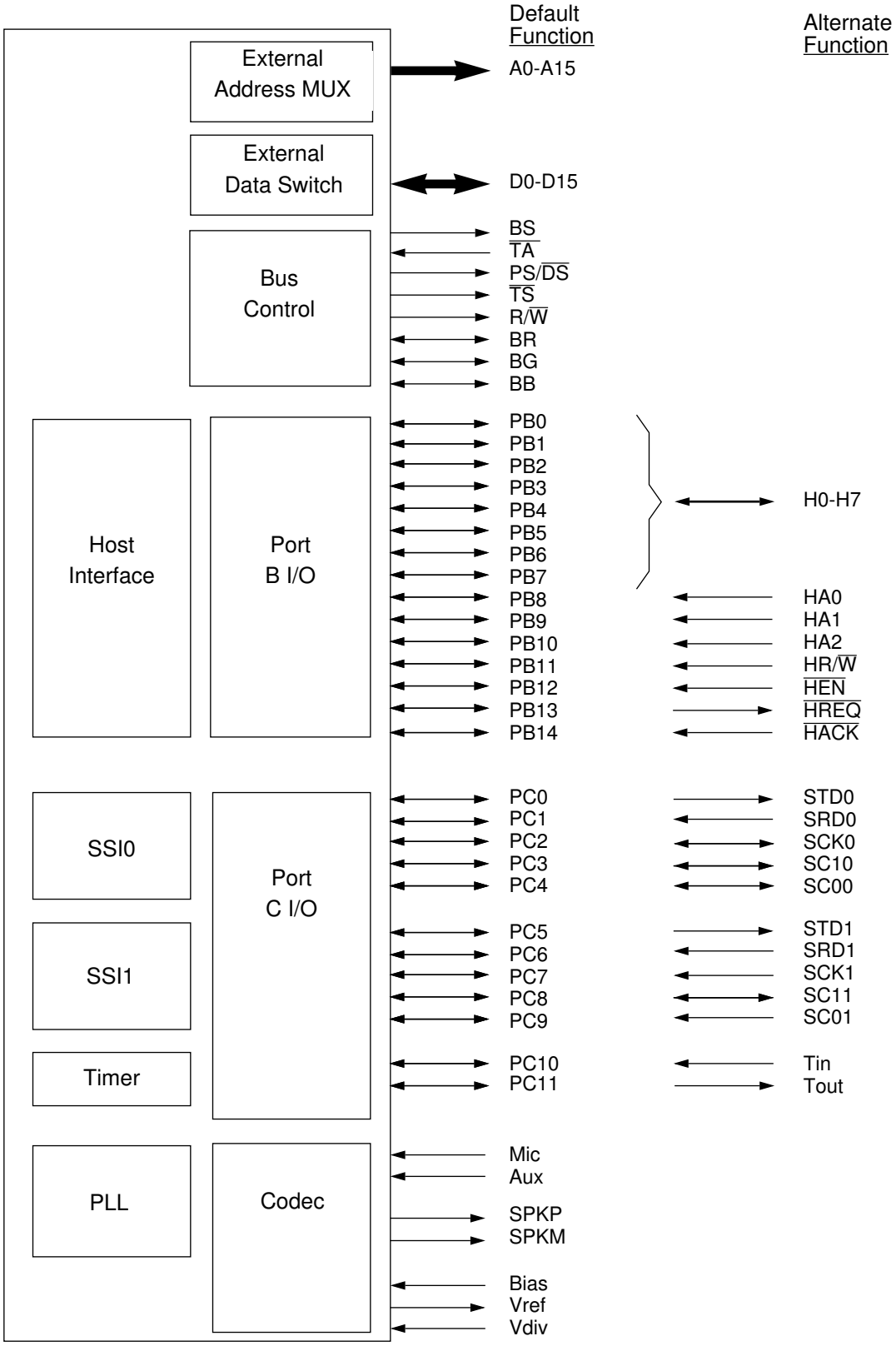
IxL2	Trigger Mode
0	Level
1	Negative Edge

priority level of the interrupts) has control bits for the two external interrupt pins and each of the on-chip peripherals (see Figure 1-8, Table 1-2, and Table 1-4). There are 30 interrupts available and two reserved on the DSP56156. Table 1-6 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-

Table 1-5 Exception Priorities within an IPL

Priority	Exception	Enabled by	IP Reg. Bit No.	Control Register Address
Level 3 (Non-maskable)				
Highest	Hardware $\overline{\text{RESET}}$			
	Illegal Instruction Interrupt	—	—	—
	Stack Error	—	—	—
Lowest	SWI	—	—	—
Level 0, 1, 2 (Maskable)				
Highest	$\overline{\text{IRQA}}$ (External Interrupt)	$\overline{\text{IRQA}}$ mode bits	0, 1	X:$FFDF
	$\overline{\text{IRQB}}$ (External Interrupt)	$\overline{\text{IRQB}}$ mode bits	3, 4	X:$FFDF
	CODEC	COIE	6,7	X:$FFDF
	Host Command Interrupt	HCIE	2	X:$FFC4
	Host/DMA RX Data Interrupt	HRIE	0	X:$FFC4
	Host/DMA TX Data Interrupt	HTIE	1	X:$FFC4
	SSI0 RX Data with Exception Status	RIE	15	X:$FFD1
	SSI0 RX Data	RIE	15	X:$FFD1
	SSI0 TX Data with Exception Status	TIE	14	X:$FFD1
	SSI0 TX Data	TIE	14	X:$FFD1
	SSI1 RX Data with Exception Status	RIE	15	X:$FFD9
	SSI1 RX Data	RIE	15	X:$FFD9
	SSI1 TX Data with Exception Status	TIE	14	X:$FFD9
	SSI1 TX Data	TIE	14	X:$FFD9
	Timer Overflow Interrupt	OIE	9	X:$FFEC
Lowest	Timer Compare Interrupt	CIE	10	X:$FFEC

Figure 1-9 Input/Output Block Diagram


Default Function
Alternate Function
External Address MUX
A0-A15
External Data Switch
D0-D15
Bus Control
BS
TA
PS/DS
TS
R/W
BR
BG
BB
Host Interface
Port B I/O
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
H0-H7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
HA0
HA1
HA2
HR/W
HEN
HREQ
HACK
SSI0
SSI1
Timer
Port C I/O
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9
PC10
PC11
STD0
SRD0
SCK0
SC10
SC00
STD1
SRD1
SCK1
SC11
SC01
Tin
Tout
PLL
Codec
Mic
Aux
SPKP
SPKM
Bias
Vref
Vdiv

bit PC can address 65,536 locations in program memory space. See the ***DSP56100 Family 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 nonmaskable. 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 1-2). 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.

1.2.5.2 Interrupt Priority Levels (IPL)

The interrupt priority level for each on-chip peripheral device and for each external interrupt source ($\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$) 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 Register shown in Figure 1-8. This read/write register specifies the interrupt priority level for each of the interrupting devices (Codec, Host, SSIs, Timer, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$). In addition, this register specifies the trigger mode of both external interrupt sources and it is used to enable or disable the individual external interrupts. This register is cleared on $\overline{\text{RESET}}$. Table 1-3 defines the interrupt priority level bits. Table 1-4 defines the external interrupt trigger mode bits.

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 1-5. The interrupt enable bits for the Host, SSIs, and TM are located in the control registers associated with their respective on-chip peripherals.

1.3 MEMORY ORGANIZATION

Two independent memory spaces of the DSP56156 — X data and program, are described in detail in Section 3. These memory spaces are configured by control bits in the Operating Mode Register. MA and MB control the program memory map and select the reset vector address.

Table 1-6
Interrupt Sources Memory Map

Interrupt Starting Address	IPL	Interrupt Source
P:$0000	3	Hardware $\overline{\text{RESET}}$
P:$0002	3	Illegal Instruction
P:$0004	3	Stack Error
P:$0006	3	Reserved
P:$0008	3	$\overline{\text{SWI}}$
P:$000A	0-2	$\overline{\text{IRQA}}$
P:$000C	0-2	$\overline{\text{IRQB}}$
P:$000E	0-2	Reserved
P:$0010	0-2	SSI0 Receive Data with Exception
P:$0012	0-2	SSI0 Receive Data
P:$0014	0-2	SSI0 Transmit Data with Exception
P:$0016	0-2	SSI0 Transmit Data
P:$0018	0-2	SSI1 Receive Data with Exception
P:$001A	0-2	SSI1 Receive Data
P:$001C	0-2	SSI1 Transmit Data with Exception
P:$001E	0-2	SSI1 Transmit Data
P:$0020	0-2	Timer Overflow
P:$0022	0-2	Timer Compare
P:$0024	0-2	Host DMA Receive Data
P:$0026	0-2	Host DMA Transmit Data
P:$0028	0-2	Host Receive Data
P:$002A	0-2	Host Transmit Data
P:$002C	0-2	Host Command (default)
P:$002E	0-2	Codec Receive/Transmit
P:$0030	0-2	Available for Host Command
P:$0032	0-2	Available for Host Command
P:$0034	0-2	Available for Host Command
P:$0036	0-2	Available for Host Command
P:$0038	0-2	Available for Host Command
P:$003A	0-2	Available for Host Command
P:$003C	0-2	Available for Host Command
P:$003E	0-2	Available for Host Command

1.4 EXTERNAL BUS, I/O, AND ON-CHIP PERIPHERALS

Five on-chip peripherals are provided on the DSP56156: an 8-bit parallel Host MPU/DMA Interface, a 16-bit timer, two Synchronous Serial Interfaces (SSI1 and SSI0), and a sigma-delta codec (see Figure 1-7). The DSP56156 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 DSP56156 also provides 27 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 DSP56156 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 which may be used as general purpose I/O pins or as Host MPU/DMA Interface pins. The Host MPU/DMA Interface provides a dedicated 8-bit paral-

lel port to a host microprocessor or DMA controller and can provide debugging facilities via host exceptions.

Port C is a 12-pin I/O interface which may be used as general purpose I/O pins or as pins for a Timer and two identical Synchronous Serial Interface.

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 which provides the user considerable flexibility in the codec filter characteristics. An external Vref pin allows multiple codecs to be referenced from a single, common voltage reference source.

1.4.1 Memory Expansion Port (Port A)

The DSP56156 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, and other 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 bus control register (BCR). The BCR controls the timing of the bus interface signals, and the data lines. Each of two memory spaces X data and Program data has its own 5-bit 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). In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces.

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 12-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. Note that these registers are read/write making the use of bit manipulation instructions extremely effective.

1.4.3 SSI0 and SSI1

The DSP56156 provides two identical Synchronous Serial Interfaces (SSI's). They are extremely flexible, full-duplex serial interfaces which allow the DSP56156 to communicate with a variety of serial devices. These interfaces include one or more industry standard

codecs, other DSPs, microprocessors, and peripherals. Selectable logarithmic compression and expansion (μ-law and A-law) is available for easier interface with PCM monocircuits and PCM highways. The SSIx interface consists of independent transmitter and receiver sections and a common SSI clock generator. Each of the following characteristics of the SSI can be independently defined: the number of bits per word, the protocol or mode, the clock, and the transmit/receive synchronization. Three modes of operation are available: Normal, Network, and On-Demand. The Normal Mode is typically used to interface with devices on a regular or periodic basis. In this mode the SSI functions with one data word of I/O per frame. The Network Mode provides time slots in addition to a bit clock and a frame synchronization pulse. The SSI functions with from 2 to 32 words of I/O per frame in the Network Mode. This mode is typically used in star or ring Time Division Multiplex (TDM) networks with other DSP56156s and/or codecs. The On-Demand Mode is a data driven mode and is a special case of the Network Mode. There are no time slots defined. This mode is intended to be used to interface to devices on a non-periodic basis. Since the transmitter and receiver sections of the SSI are independent, they may be programmed to be synchronous (use a common clock and frame sync) or asynchronous (use a common clock but different frame sync) with respect to each other. The SSI supports a subset of the Motorola SPI interface. The SSI requires three to six 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 DSP56156 has 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 DSP56156 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 DSP56156. The host may select any one of 31 DSP56156 exception routines to be executed by writing a Vector Address Register in the HI. This flexibility allows the host programmer to execute up to 31 functions preprogrammed in the DSP56156.

1.5 OnCE

OnCE provides hardware/software emulation and debug on the DSP56156 and a means of interacting with the DSP56156 and any memory mapped peripherals non-intrusively so that a user may examine registers, memory, or on-chip 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 DSP56156 into his target system yet retain debug control, especially in the cases of devices specified without external bus. The need for a costly cable which brings out the all processor pins on traditional emulator systems is eliminated.

1.6 PROGRAMMING MODEL

The programmer can view the DSP56156 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 1-10 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.

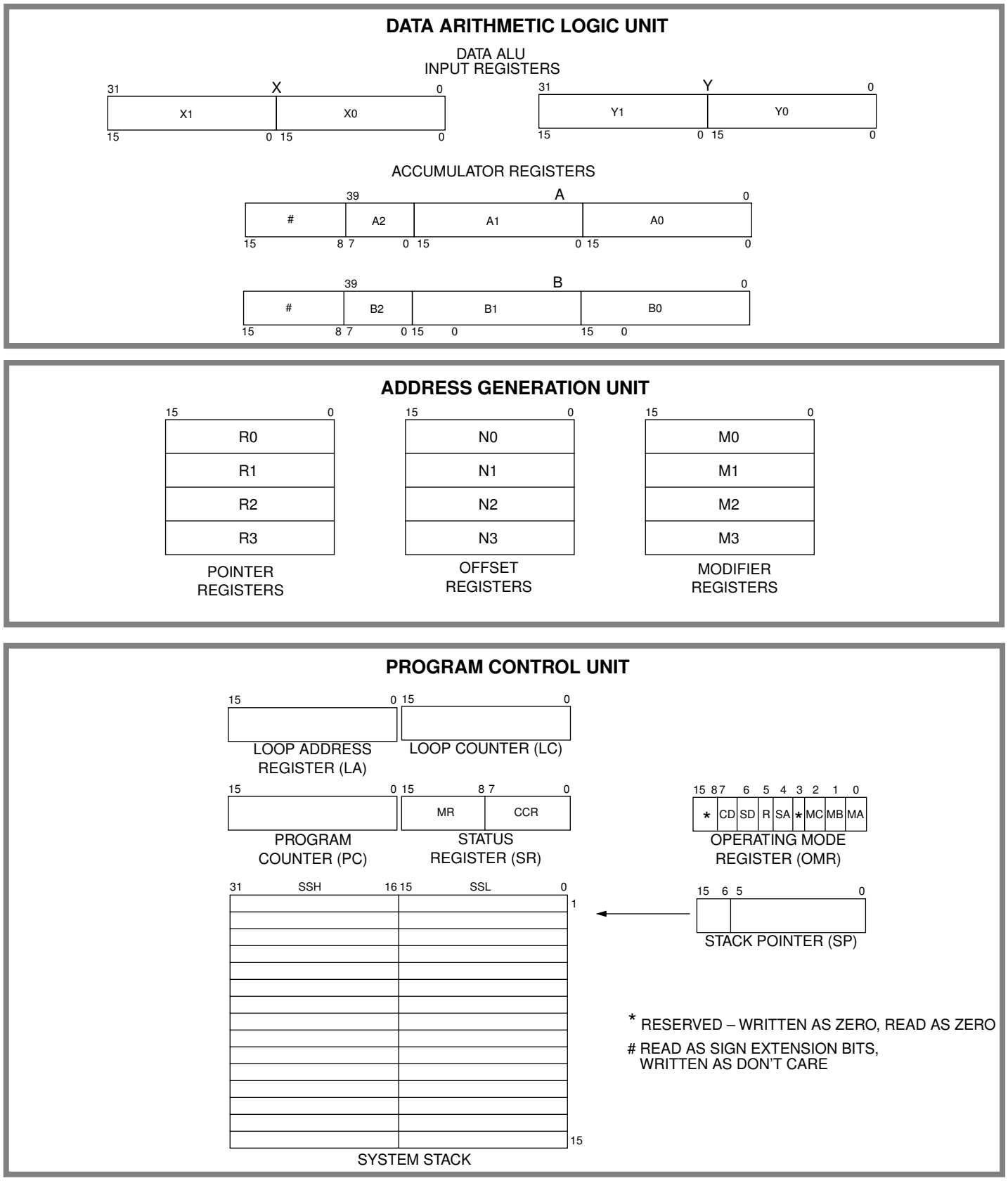


Figure 1-10 DSP56156 Programming Model

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 which serve as data ALU accumulator registers. A2 and B2 are 8-bit latches which 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.

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, 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 operands. 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 1-11. 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 pro-

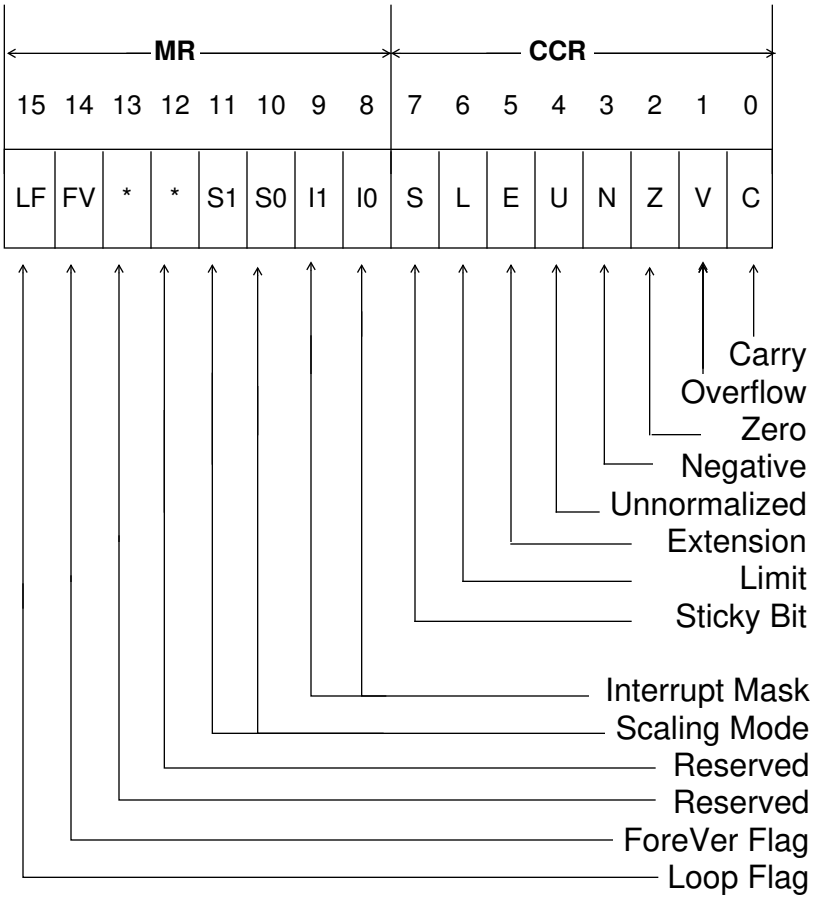


Figure 1-11 Status Register Format

cessor 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 if data limiting occurs when 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, BScc, 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 in 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 banks: High (SSH) and Low (SSL) each 16 bits wide. SSH stores the PC or LA contents; SSL stores the LC or 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 stack pointer register (SP) is a 6-bit register that indicates the location of the top of the SS and the status of the stack (underflow, empty, full, and overflow conditions – see Table 1-7). 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.

1.6.3.7 Operating Mode Register (OMR)

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

Table 1-7 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.

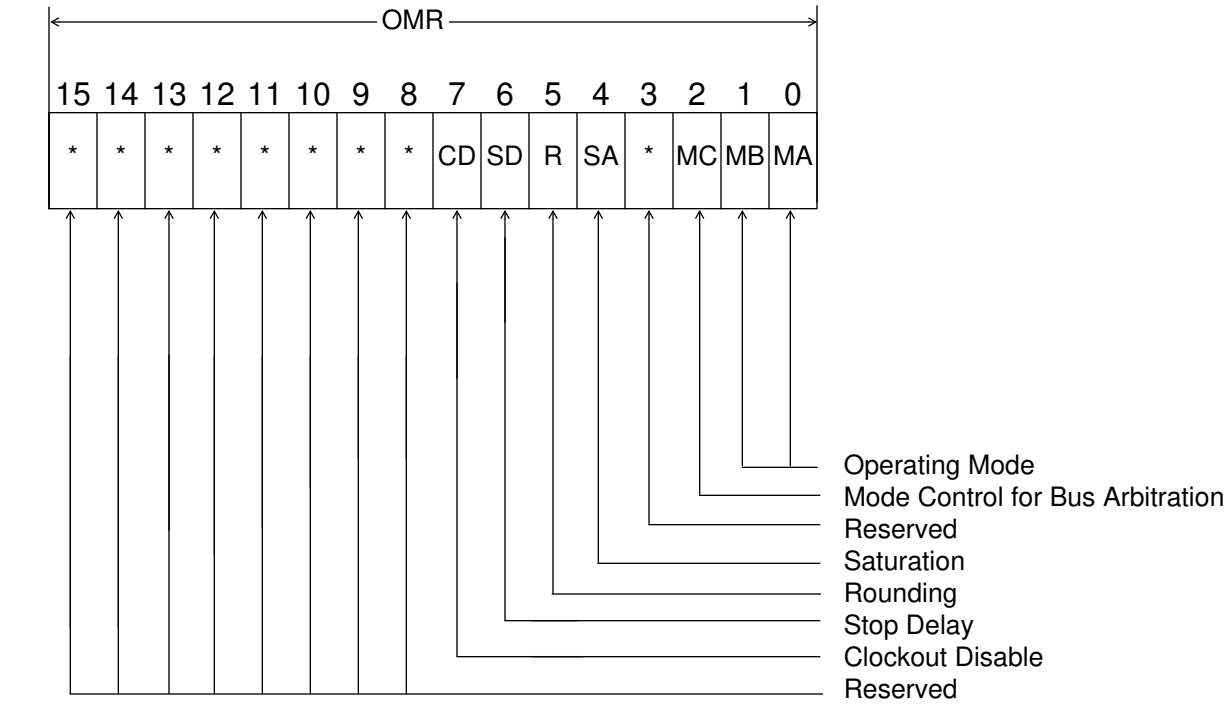
During processor reset, the chip operating mode bits will be loaded from the external Mode Select pins (see Table 1-7 and Table 1-9). The operating mode register format is shown in Figure 1-12.

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

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 1-10.

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.



* = Reserved, read as a zero, write as zero to insure compatibility

Figure 1-12 Operating Mode Register

Table 1-8
Operating Mode Summary — PRAM Part

Operating Mode	MB	MA	Description
Special Bootstrap 1	0	0	Bootstrap from an external byte-wide memory located at P:$C000
Special Bootstrap 2	0	1	Bootstrap from the Host port or SSI0
Normal Expanded	1	0	Internal PRAM enabled External reset at P:$E000
Development Expanded	1	1	Internal program RAM disabled; External reset at P:$000

Table 1-9
Operating Mode Summary — PROM Part

Operating Mode	MB	MA	Description
Single Chip	0	0	Internal PROM enabled Internal Reset at P:$0000
Reserved	0	1	Reserved
Normal Expanded	1	0	Internal PROM enabled External reset at P:$E000
Development Expanded	1	1	Internal PROM disabled External reset at P:$0000

Table 1-10
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, a clock out signal comes out of the CLKO pin. Setting the CD bit will disable the signal coming out of the CLKO pin 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 speed 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 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 following instructions and 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*
CHKAAU	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 word 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 or 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 for ever
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 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, and 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 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 1-11. 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. 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

Table 1-11 DSP56156 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

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 DSP56156 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 contents 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).

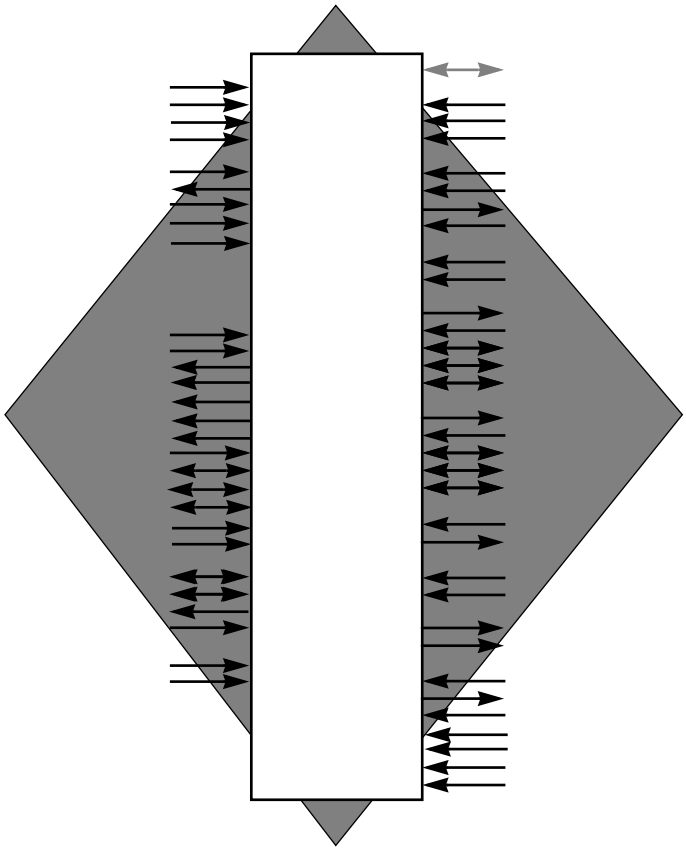
1.7.4.2 Reverse Carry Modifier

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 2π. 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.

SECTION 2

DSP56156 PIN DESCRIPTIONS



SECTION CONTENTS

2.1 INTRODUCTION

The DSP56156 pinout is shown in Figure 2-2. The input and output signals on the chip are organized into the 13 functional groups shown in Table 2-1. Figure 2-1 illustrates the relative timing for the bus signals. See the timing descriptions in the technical data sheet for exact information.

Table 2-1 Functional Group Pin Allocations

Functional Group	Number of Pins
Address and Data Buses	32
Bus Control	9
Interrupt and Mode Control	4
Clock and PLL	3
Host Interface or PIO	15
Timer Interface or PIO	2
SSI Interfaces or PIO	10
On-chip CODEC	7
On-chip emulation (OnCE)	4
Power (Vdd)	9
Ground (Vss)	15
APower (AVdd)	1
AGround (AVss)	1
Total	112

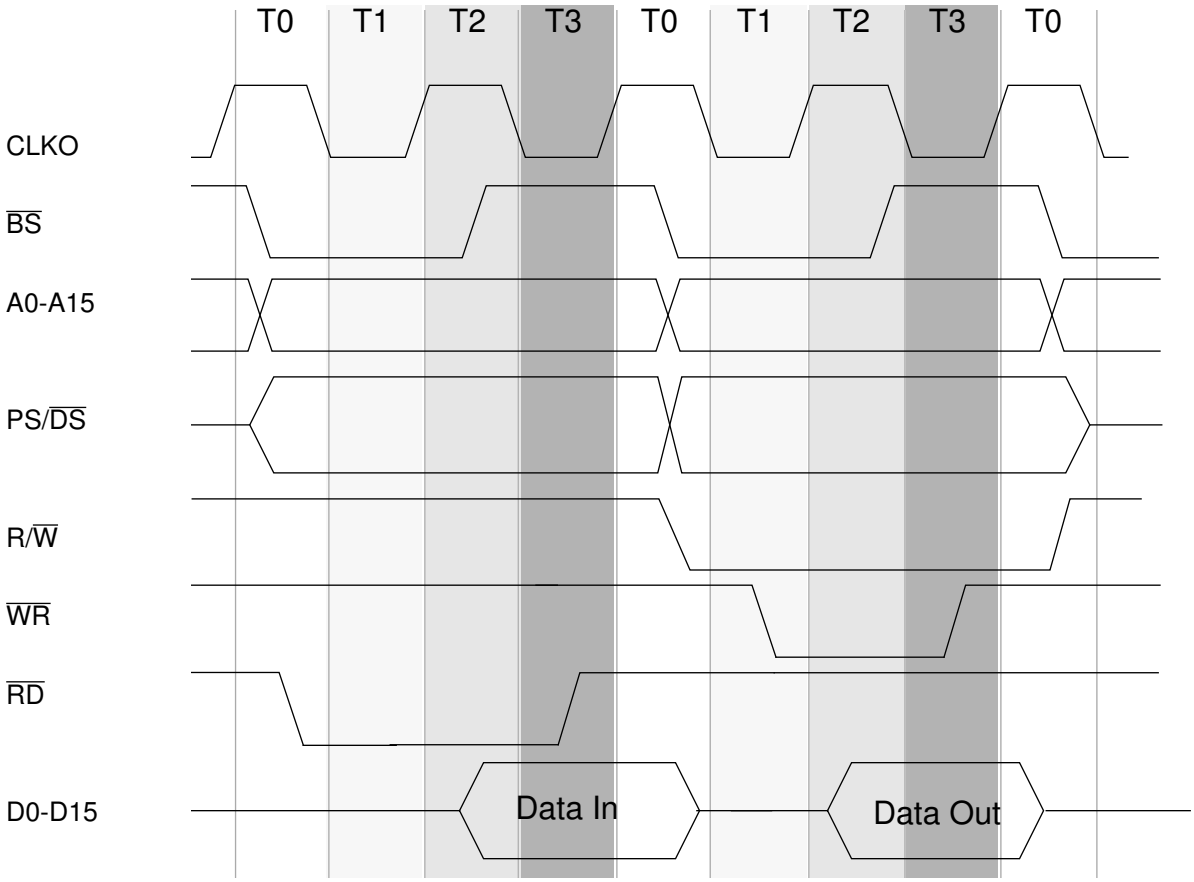
2.2 ADDRESS AND DATA BUS (32 PINS)

A0-A15 (Address Bus) - three state, active high outputs. A0-A15 change in t0 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 or when the DSP is not bus master.

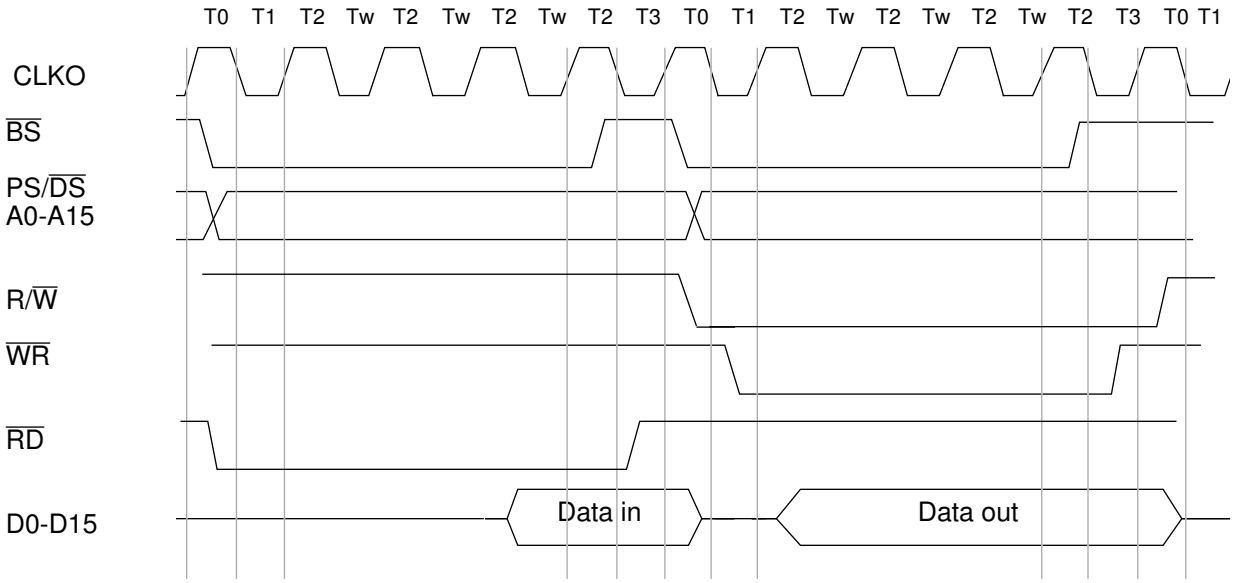
D0-D15 (Data Bus) - three state, active high, bidirectional input/outputs. Read data is sampled on the trailing edge of t2, while write data output is enabled by the leading edge of t2 and three-stated at the leading edge of t0. If there is no external bus activity, D0-D15 are three-stated. D0-D15 are also three-stated during hardware reset.

2.3 BUS CONTROL (9 PINS)

PS/$\overline{\text{DS}}$ (Program /Data Memory Select) - three state active low output. This output is asserted only when external data memory is referenced. PS/$\overline{\text{DS}}$ timing is the same for the A0-A15 address lines. PS/$\overline{\text{DS}}$ is high for program memory access and is low for data memory access. If the external bus is not used during an instruction cycle (t0,t1,t2,t3), PS/$\overline{\text{DS}}$ goes high in t0. PS/$\overline{\text{DS}}$ is in the high impedance state during hardware reset or when the DSP is not bus master.



Bus Operation (Read-Write- 0WT)



Bus Operation (Read-Write- 3WT)

Figure 2-1 Bus Operation

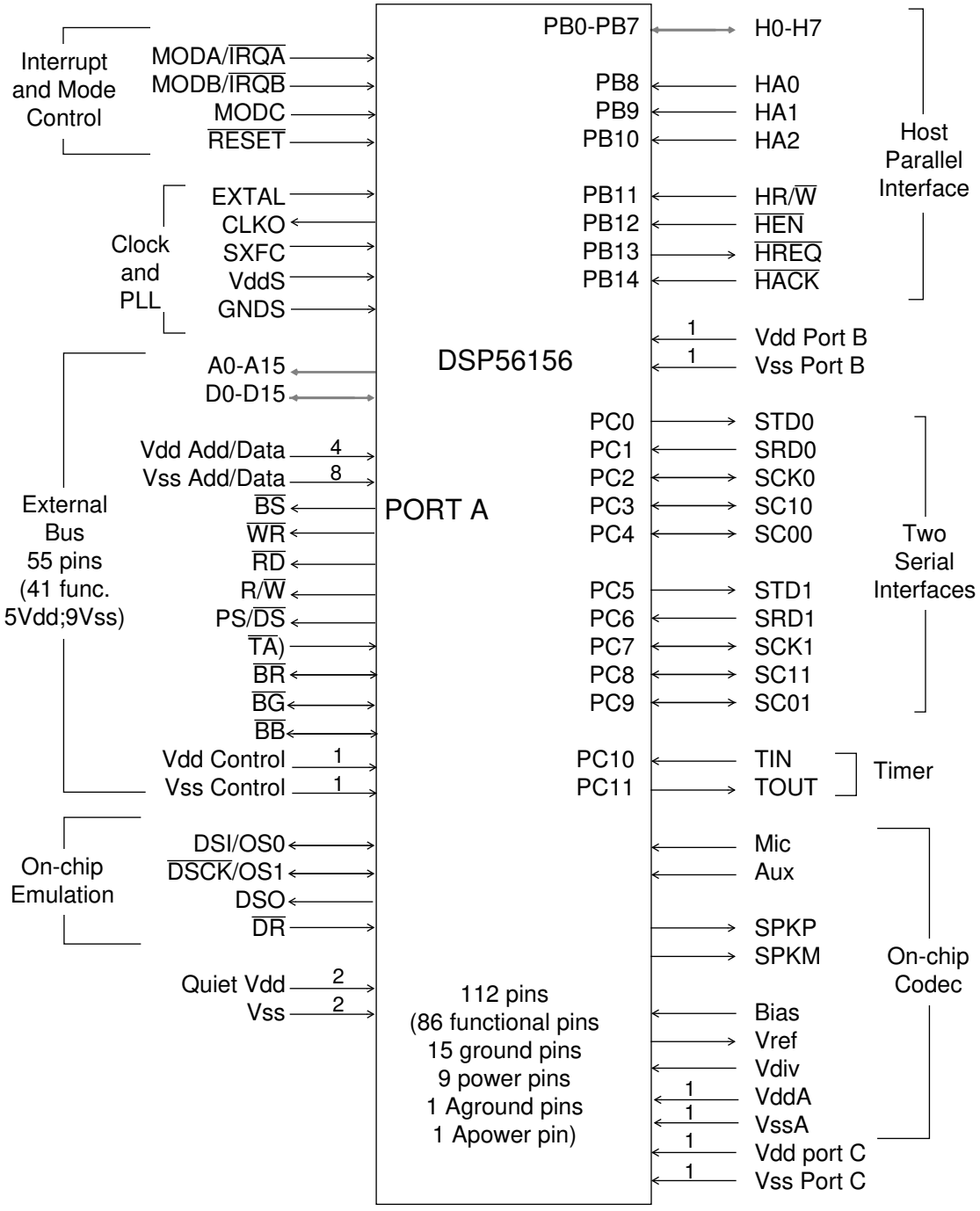


Figure 2-2 DSP56156 Pinout

R/$\overline{W}$ **(Read/Write)- three state, active low output.** Timing is the same as for the address lines, providing an “early write” signal. R/$\overline{W}$ (which changes in 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,t3), R/$\overline{W}$ goes high in t0. R/$\overline{W}$ is three-stated during hardware reset or when the DSP is not bus master.

$\overline{WR}$ **(Write Enable) - three state, active low output.** This output is asserted during external memory write cycles. When $\overline{WR}$ is asserted in t1, the data bus pins D0-D15 become outputs and the DSP puts data on the bus during the leading edge of t2. When $\overline{WR}$ is deasserted in t3, the external data has been latched inside the external device. When $\overline{WR}$ is asserted, it qualifies the A0-A15 and PS/$\overline{DS}$ pins. $\overline{WR}$ can be connected directly to the $\overline{WE}$ pin of a static RAM. $\overline{WR}$ is three-stated during hardware reset or when the DSP is not bus master.

$\overline{RD}$ **(Read Enable) - three state, active low output.** This output is asserted during external memory read cycles. When $\overline{RD}$ is asserted in late t0/early t1, the data bus pins D0-D15 become inputs and an external device is enabled onto the data bus. When $\overline{RD}$ is deasserted in t3, the external data has been latched inside the DSP. When $\overline{RD}$ is asserted, it qualifies the A0-A15 and PS/$\overline{DS}$ pins. $\overline{RD}$ can be connected directly to the $\overline{OE}$ pin of a static RAM or ROM. $\overline{RD}$ is three-stated during hardware reset or when the DSP is not bus master.

$\overline{BS}$ **(Bus Strobe) - three state, active low output.** Asserted at the start of a bus cycle (during t0) and deasserted at the end of the bus cycle (during t2). This pin provides an “early bus start” signal which can be used as address latch and as an “early bus end” signal which can be used by an external bus controller. $\overline{BS}$ is three-stated during hardware reset and when the DSP is not a bus master.

$\overline{TA}$ **TA (Transfer Acknowledge) - active low input.** If there is no external bus activity, the $\overline{TA}$ input is ignored by the DSP. When there is external bus cycle activity, $\overline{TA}$ can be used to insert wait states in the external bus cycle. $\overline{TA}$ is sampled on the leading edge of the clock. Any number of wait states from 1 to infinity may be inserted by using $\overline{TA}$. If $\overline{TA}$ is sampled high on the leading edge of the clock beginning the bus cycle, the bus cycle will end 2T after the $\overline{TA}$ 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 $\overline{TA}$ input or by the Bus Control Register (BCR), whichever is longer. $\overline{TA}$ is still sampled during the leading edge of the clock when wait states are controlled by the BCR value. In that case, $\overline{TA}$ 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. $\overline{TA}$ should always be deasserted during t3 to be sampled high by the leading edge of T0. If $\overline{TA}$ 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 $\overline{TA}$ during the leading edge of T2.

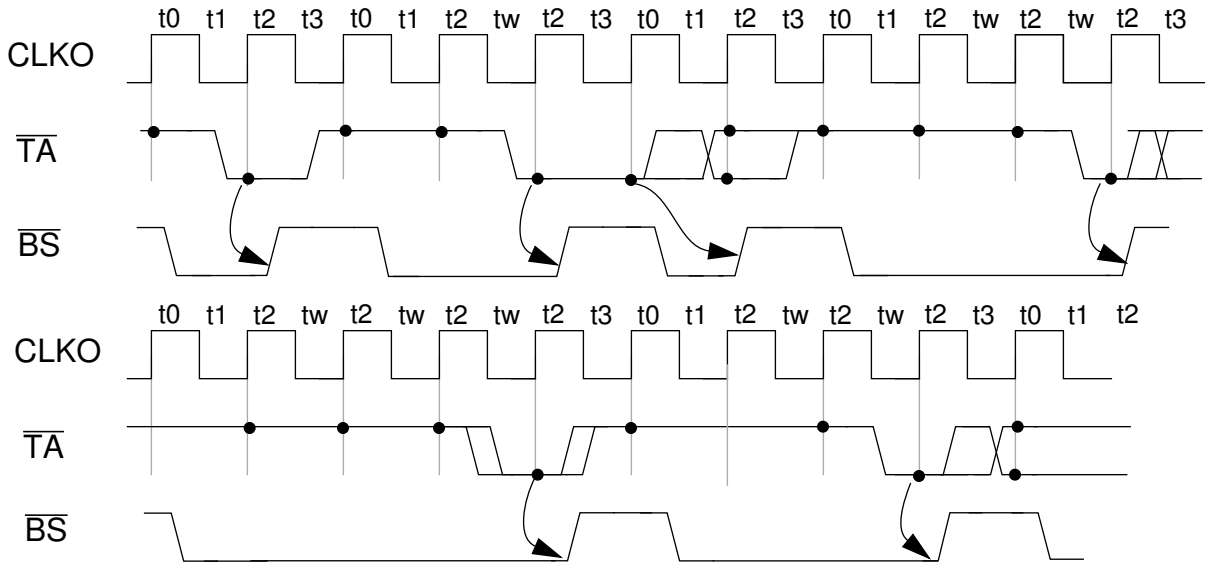


Figure 2-3 $\overline{\text{TA}}$ Controlled Accesses

$\overline{\text{BR}}$ **(Bus Request) - active low output when in master mode, active low input when in slave mode.** After power-on reset, this pin is an input (slave mode). In this mode, the bus request $\overline{\text{BR}}$ allows another device such as a processor or DMA controller to become the master of the DSP external data bus D0-D15 and external address bus A0-A15. The DSP asserts $\overline{\text{BG}}$ a few T states after the $\overline{\text{BR}}$ 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 PS/ $\overline{\text{DS}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$, and R/$\overline{\text{W}}$ at the earliest time possible consistent with proper synchronization. These pins will then be placed in the high impedance state and the $\overline{\text{BB}}$ 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 $\overline{\text{BR}}$ input pin is asserted, the DSP bus controller inserts wait states until the external bus becomes available ($\overline{\text{BR}}$ and $\overline{\text{BB}}$ deasserted). Note that interrupts are not serviced when a DSP instruction is waiting for the bus controller. Note also that $\overline{\text{BR}}$ 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 $\overline{\text{BR}}$ to request the bus mastership. The DSP bus controller will insert wait states until $\overline{\text{BG}}$ input is asserted and will then begin normal bus accesses after the ris-

ing 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 Bus Control Register (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 Bus Control Register (BCR).

In the master mode, $\overline{BR}$ can also be used for non arbitration purpose: if $\overline{BG}$ is always asserted, $\overline{BR}$ is asserted in t0 of every external bus access. It can then be used as a chip select to turn a 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 t0.

$\overline{BG}$ **(Bus Grant) - active low input when in master mode, active low output when in slave mode.** Output after power on reset if the slave is selected, this pin 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 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, the $\overline{BG}$ output is deasserted and the DSP regains control of the external address bus, data bus, and bus control pins when the $\overline{BB}$ pin is sampled high.

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 bus master, active low output when 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 re-

leases bus mastership. $\overline{BB}$ becomes an input when the DSP is no longer the bus master.

2.4 INTERRUPT AND MODE CONTROL (3 PINS)

MODA/$\overline{IRQA}$ (**Mode Select A/External Interrupt Request A)** - This input has two functions - to select the initial chip operating mode and, 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 the 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 standby 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 - to select the initial chip operating mode and, 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 the external interrupt request $\overline{IRQB}$. After reset, the chip operating mode can be changed by software. The $\overline{IRQB}$ 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 **(Mode Select C)** - This input is used to select the initial bus operating mode. When tied high, the external bus is programmed in the master mode ($\overline{BR}$ output and $\overline{BG}$ input) and when tied low the bus is programmed in the slave mode ($\overline{BR}$ input and $\overline{BG}$ output). MODC is read and internally latched in the DSP when the processor exits the reset state. After $\overline{RESET}$, the bus operating mode can be changed by software by writing the MC bit of the OMR register.

$\overline{RESET}$ **(Reset)** - This input is a direct hardware reset of the processor. When $\overline{RESET}$ is asserted, the DSP is initialized and placed in the reset state. A Schmitt trigger input is used for noise immunity. When the reset pin is deasserted, the initial

chip operating mode is latched from the MODA and MODB pins. The internal reset signal should be deasserted synchronous with the internal clocks.

2.5 POWER, GROUND, AND CLOCK (28 PINS)

VCC (8) **(Power)** - power pins.

VSS (14) **(Ground)** - ground pins.

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

GNDS **(Synthesizer Ground)** - This pin supplies a quiet ground source to the PLL to provide greater frequency stability.

VDDA **(Power Supply input)** - This pin is the positive analog supply input. It should be connected to VCC 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.

EXTAL **(External Clock/Crystal Input)** - This input should be connected to an external clock or to an external oscillator. After being squared, the input frequency can be used as the DSP core internal clock. In that case, it is divided by two to produce a four phase instruction cycle clock, the minimum instruction time being two input clock periods.This input frequency is also used, after division, as input clock for the on-chip codec and the on-chip phase locked loop (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). In this case, the pin can be left floating.

SXFC **(External Filter Capacitor)** - This pin is used to add an external capacitor to the PLL filter circuit. A low leakage capacitor should be connected between SXFC and VDDS; it should be located very close to those pins.

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 $\overline{\text{HEN}}$ is asserted. HA0-HA2 may be programmed as general purpose parallel I/O pins called PB8-PB10 when the HI is not being used.

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

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

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

$\overline{\text{HACK}}$ **(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, $\overline{\text{HACK}}$ may be used as a data strobe for HI DMA data transfers. If programmed as an MC68000 Host Interrupt Acknowledge, $\overline{\text{HACK}}$ is used to enable the HI Interrupt Vector Register (IVR) onto the Host Data Bus H0-H7 if the Host Request $\overline{\text{HREQ}}$ output

is asserted. In this case, all other HI control pins are ignored and the HI state is not affected. $\overline{\text{HACK}}$ may be programmed as a general purpose I/O pin called PB14 when the HI is not being used.

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

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 or toggles on a timer overflow event or 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 SYNCHRONOUS SERIAL INTERFACES (SSI0 AND SSI1) (10 PINS)

STD0 **(SSI0 Transmit Data)** - This output pin transmits serial data from the SSI0 Transmit Shift Register. STD0 may be programmed as a general purpose I/O pin called PC0 when the SSI0 STD0 function is not being used.

SRD0 **(SSI0 Receive Data)** - This input pin receives serial data and transfers the data to the SSI0 Receive Shift Register. SRD0 may be programmed as a general purpose I/O pin called PC1 when the SSI0 SRD0 function is not being used.

SCK0 **(SSI0 Serial Clock)** - This bidirectional pin provides the serial bit rate clock for the SSI0 interface. SCK0 may be programmed as a general purpose I/O pin called PC2 when the SSI0 interface is not being used.

SC10 **(Serial Control 1)** - This bidirectional pin is used by the SSI0 serial interface as frame sync I/O or flag I/O. SC10 may be programmed as a general purpose I/O pin called PC3 when the SSI0 is not using this pin.

SC00 **(Serial Control 0)** - This bidirectional pin is used by the SSI0 serial interface as frame sync I/O or flag I/O. SC00 may be programmed as a general purpose I/O pin called PC4 when the SSI0 is not using this pin.

STD1 **(SSI1 Transmit Data)** - This output pin transmits serial data from the SSI1 Transmit Shift Register. STD1 may be programmed as a general purpose I/O pin called PC5 when the SSI1 STD1 function is not being used.

SRD1 **(SSI1 Receive Data)** - This input pin receives serial data and transfers the data to the SSI1 Receive Shift Register. SRD1 may be programmed as a general purpose I/O pin called PC6 when the SSI1 SRD1 function is not being used.

SCK1 **(SSI1 Serial Clock)** - This bidirectional pin provides the serial bit rate clock for the SSI1 interface. SCK1 may be programmed as a general purpose I/O pin called PC7 when the SSI1 interface is not being used.

SC11 **(Serial Control 1)** - This bidirectional pin is used by the SSI1 serial interface as frame sync I/O or flag I/O. SC11 may be programmed as a general purpose I/O pin called PC8 when the SSI1 is not using this pin.

SC01 **(Serial Control 0)** - This bidirectional pin is used by the SSI1 serial interface as frame sync I/O or flag I/O. SC01 may be programmed as a general purpose I/O pin called PC9 when the SSI1 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 ($\overline{DR}$), 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.

$\overline{DR}$ **(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 current 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.

BIAS **(Bias current pin)** - This input is used to determine the bias current for the analog circuitry. Connecting a resistor between BIAS and VGNDA will program the current bias generator. 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.

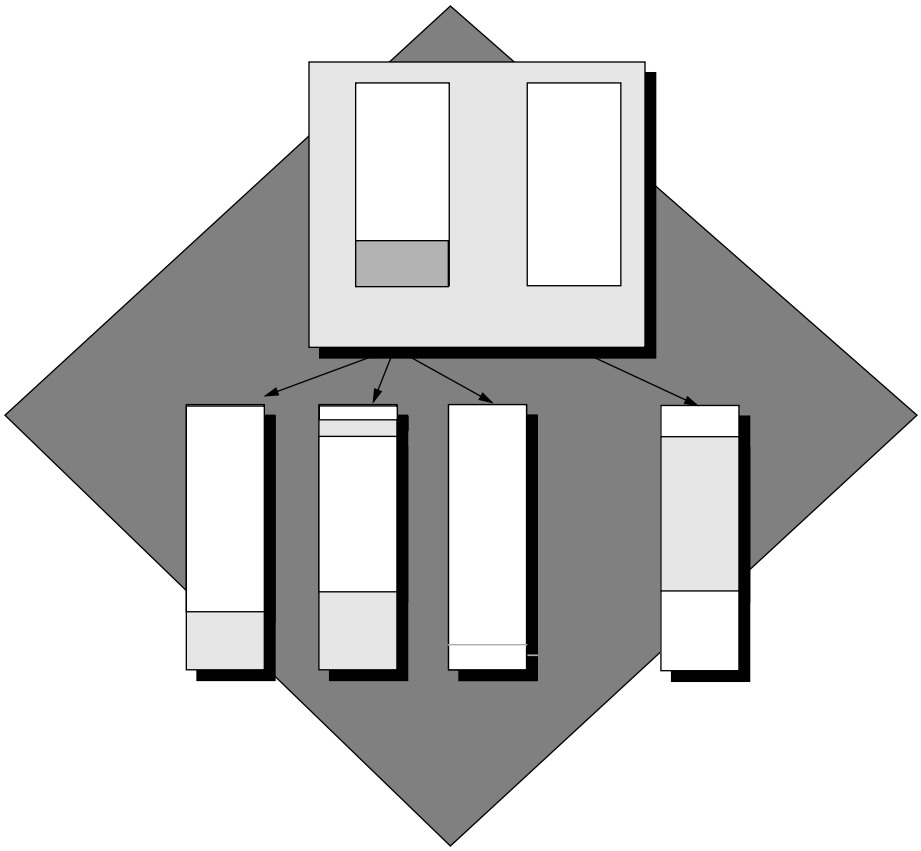
SPKM **(Speaker Negative Output)** - This pin is the negative analog output from the on-chip D/A converter. This pin should be left floating when the codec is not used.

VREF **(Voltage Reference Output)** - This pin is the op-amp buffer output in the reference voltage generator. It has a value of (2/5) VDDA. This pin should always be connected to the ground through two capacitors (typically a 0.1 μf ceramic in parallel with a 15 μf tantalum), even when the codec is not used.

VDIV **(Voltage Division Output)** - This output pin is also the input to the on-chip op-amp buffer in the reference voltage generator. The pin is connected to a resistor divider network located within the codec block which provides a voltage equal to (2/5)VDDA. This pin should be connected to the ground via a capacitor when the codec is used and should be left floating when the codec is not used.

SECTION 3

OPERATING MODES AND MEMORY SPACES



SECTION CONTENTS

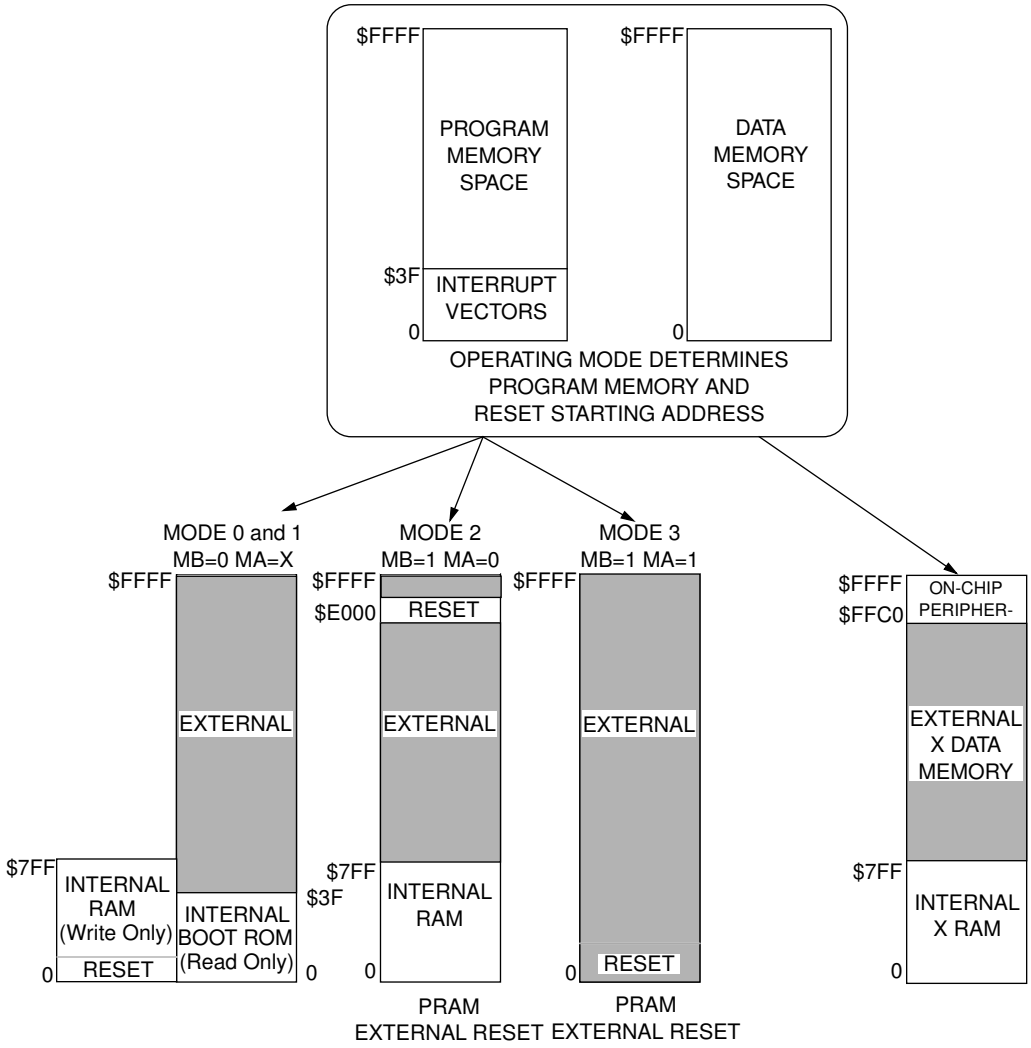


Figure 3-1 DSP56156 RAM Memory

3.1 RAM MEMORY DESCRIPTION

This part of the DSP56156 memory description describes the part that uses **RAM** memory for the **Program Memory**. The memory of the DSP56156 can be partitioned in several ways to provide high-speed parallel operation and additional off-chip memory expansion. Program and data memory are separate. Both the program and data memories can be expanded off-chip.

3.1.1 DSP56156 RAM Part Memory Introduction

The two independent memory spaces of the DSP56156, X data, and program, are shown in Figure 3-1. 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.

3.1.1.1 X Data Memory

The on-chip X data RAM is a 16-bit-wide, internal, static memory occupying the lowest 2048 locations (0–$7FF) in X memory space. The on-chip peripheral registers occupy the top 64 locations of the X data memory ($FFC0–$FFFF). The On-Chip X Data Memory addresses are received from the X address Bus one (XAB1) and X address Bus two (XAB2) and data transfers occur on the X data bus (XDB) and global data bus (GDB). **Two** reads, **one** read, **or one** write can be performed during one instruction cycle on the internal data memory. The on-chip peripherals occupy the top 64 locations in the X data memory space (X:$FFC0-X:$FFFF). X memory may be expanded off-chip for a total of 65,536 addressable locations.

3.1.1.2 Program Memory

On-chip program memory consists of a 2048-location by 16-bit, high-speed RAM that is enabled/disabled by the MA and MB bits in the OMR. The On-Chip Program Memory addresses are received from the program control logic (usually the program counter) or from the address ALU on the PAB. Off-chip program memory may be written using move program memory (MOVEM) instructions. The first 64 locations of the program memory ($0000–$003F) are reserved for interrupt vectors. The program memory may be expanded off-chip for a total of 65,536 addressable locations.

3.1.1.3 Bootstrap Memory

A 64-location program bootstrap ROM is only read by the program controller while in the bootstrap mode, during which, the on-chip program RAM is defined as write-only. 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 SSI0. Data is packed into program RAM least significant byte of P:$0000 first.

Table 3-1
Operating Mode Summary — Program RAM Part

Operating Mode	MB	MA	Description
Special Bootstrap 1	0	0	Bootstrap from an external byte-wide memory located at P:$C000. Internal reset at P:$0000
Special Bootstrap 2	0	1	Bootstrap from the Host port (P:$C000 bit 15=0) or SSI0 (P:$C000 bit 15=0). External 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.

3.1.1.4 Chip Operating Modes

The DSP operating modes determine the memory maps for program and data memories and the startup 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.

3.1.1.4.1 Bootstrap Mode (Mode 0)

Mode 0 is one of two bootstrap modes which have all internal program and data RAM memories enabled (see Figure 3-1). 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 DSP56156 executes a bootstrap program which **loads** program memory **from a byte wide memory** located at P:$C000 (see Table 3-1). Section 3.1.2.2 describes the bootstrap operation. The memory maps for Mode 0 and Mode 1 are identical. The 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 at internal memory location P:$0000. The reset vector location in Mode 2 is at external memory location P:$E000.

3.1.1.4.2 Bootstrap Mode (Mode 1)

Mode 1 is one of two bootstrap modes which have all internal program and data RAM memories enabled (see Figure 3-1). 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 DSP56156 executes a bootstrap program which loads program memory from either the host port or SSI0 depending on whether bit 15 of location P:C000 is a zero (host port) or a one (SSI0). Section 3.1.2.2 describes the bootstrap operation. The memory maps for Mode 0 and Mode 1 are identical. The 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 at internal memory location P:$0000. The reset vector location in Mode 2 is at external memory location P:$E000.

3.1.1.4.3 Normal Expanded Mode (Mode 2)

The normal expanded mode (Mode 2) has the same memory map as Mode 0 and Mode 1 (see Figure 3-1). 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).

3.1.1.4.4 Development Mode (Mode 3)

The development mode is similar to the normal expanded mode except that internal program memory is disabled (see Figure 3-1). 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). DSP56156ROM 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. Bootstrap Mode

3.1.2 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.

3.1.2.1 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 Synchronous

Serial Interface SSI0. No access is provided to the bootstrap ROM other than through the bootstrap process. Control logic will disable the bootstrap ROM during normal operations.

3.1.2.2 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 Synchronous Serial Interface SSI0.
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).

3.1.2.3 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 DSP56100 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 Synchronous Serial Interface SSI0.

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 SSI0 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 Synchronous Serial Interface SSI0. The bootstrap program sets up the SSI0 in 8 bit mode, external clock, and asynchronous mode.

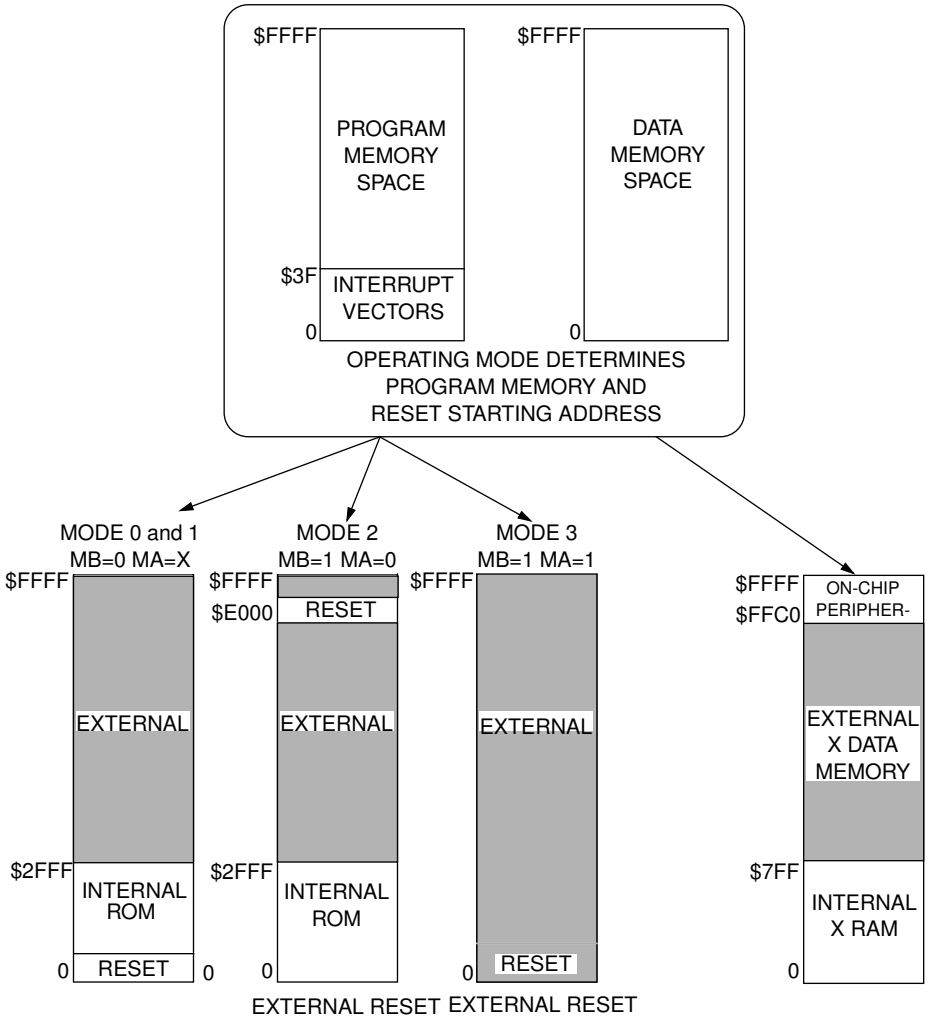


Figure 3-2 DSP56156 ROM Memory Map

3.2 ROM MEMORY DESCRIPTION

This part of the DSP56156 memory description describes the part that uses 12K of **ROM** memory for the **Program Memory**. The memory of the DSP56156 can be partitioned in several ways to provide high-speed parallel operation and additional off-chip memory expansion. Program and data memory are separate. Both the program and data memories can be expanded off-chip.

3.2.1 DSP56156 ROM Part Memory Introduction

The two independent memory spaces of the DSP56156, 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.

3.2.1.1 X Data Memory

The on-chip X data RAM is a 16-bit-wide, internal, static memory occupying the lowest 2048 locations (0–$7FF) in X memory space. The on-chip peripheral registers occupy the top 64 locations of the X data memory ($FFC0–$FFFF). The On-Chip X Data Memory addresses are received from the X address Bus one (XAB1) and X address Bus two (XAB2) and data transfers occur on the X data bus (XDB) and global data bus (GDB). **Two** reads, **one** read, **or one** write can be performed during one instruction cycle on the internal data memory. The on-chip peripherals occupy the top 64 locations in the X data memory space (X:$FFC0-X:$FFFF). X memory may be expanded off-chip for a total of 65,536 addressable locations.

3.2.1.2 Program Memory

On-chip program memory consists of a 12288 location by 16-bit, high-speed ROM that is enabled/disabled by the MA and MB bits in the OMR. The On-Chip Program Memory addresses are received from the program control logic (usually the program counter) or from the address ALU on the PAB. Off-chip program memory may be written using move program memory (MOVEM) instructions. The first 64 locations of the program memory ($0000–$003F) are reserved for interrupt vectors. The program memory may be expanded off-chip for a total of 65,536 addressable locations.

3.2.1.3 Chip Operating Modes

The DSP operating modes determine the memory maps for program and data memories and the startup 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.

3.2.1.3.1 Single-chip Mode (Mode 0)

Mode 0 is one of two single-chip modes which have internal program memory 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. The memory maps for Mode 0 and Mode 1 are identical. The difference between Mode 0 and Mode 2 is the

Table 3-3
Operating Mode Summary — Program ROM Part

Operating Mode	MB	MA	Description
Single Chip	0	X	Internal PROM enabled External reset at P:$0000
Normal Expanded	1	0	Internal PROM enabled External reset at P:$E000
Development Expanded	1	1	Internal program memory disabled External reset at P:$0000

location of the reset vector in program memory. The reset vector location in Mode 0 is located in internal memory at P:$0000; whereas, the reset vector location in Mode 2 is located in external memory at P:$E000.

3.2.1.3.2 Mode 1

Mode 1 is the same as mode 0 on the DSP56156ROM part. It is recommended that this mode not be invoked by the user.

3.2.1.3.3 Normal Expanded Mode (Mode 2)

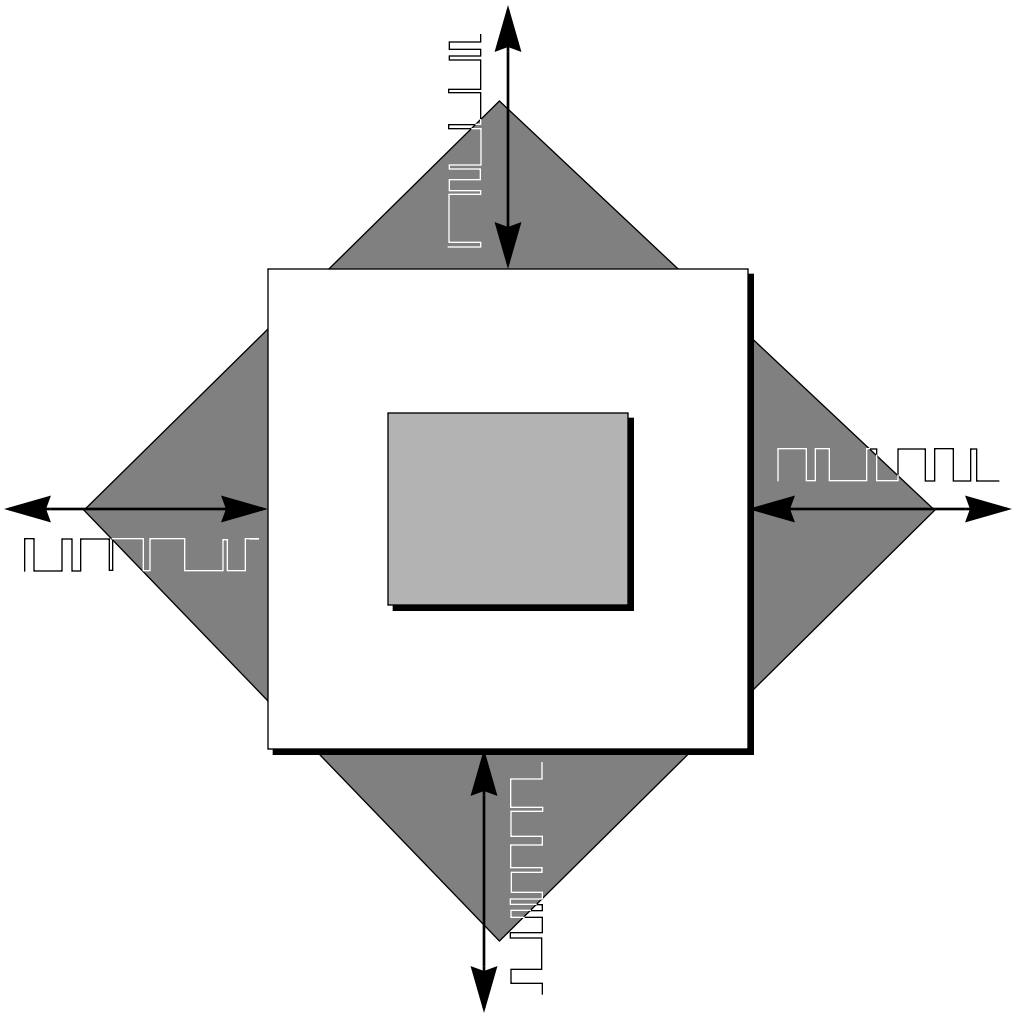
The normal expanded mode (Mode 2) has the same memory map as Mode 0 and Mode 1 (see Figure 3-2). This mode can be entered by either grounding the MA pin and pulling the MB pin high or by writing to the OMR and changing the MA and MB bits (s ee Table 3-3). The reset vector location in Mode 2 is in external program memory at location P:$E000.

3.2.1.3.4 Development Mode (Mode 3)

The development mode is similar to the normal expanded mode except that internal program memory is disabled (see Figure 3-2). 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 3-3). DSP56156ROM chips with bad or obsolete internal program ROM code can be used with external program memory in the development mode. Reset vectors to program memory location P:$0000.

SECTION 4

I/O INTERFACE



SECTION CONTENTS

4.1 INTRODUCTION

The DSP56156 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 DSP56156 also provides 27 programmable I/O pins. These pins may be used as general purpose I/O pins or allocated to an on-chip peripheral. Four on-chip peripherals are provided on the DSP56156: an 8 bit parallel Host MPU/DMA digital interface, a 16-bit timer, and two Synchronous Serial Interfaces (SSI0 and SSI1). These 27 pins are separate from the DSP56156 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 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 12-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 Synchronous Serial Interfaces, SSI0 and SSI1, are identical. They provide high speed synchronous serial data communication capability between the DSP56156 and other serial devices. Support for TDM network configurations allows communication among up to 32 devices. Transparent linear-to-logarithmic companding and expanding is also supported for A-law and μ-law coded data.

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 DSP56156 (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 Register (BCR), located at X:$FFDE, may be programmed to insert wait states in a bus cycle during external data and program memory accesses. The BCR is associated with Port A. Five bits are available in the 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 memory are preset to all ones so that 31 wait states are inserted allowing slow, inexpensive memory to be used. All other Port Control Register bits are cleared on processor reset; i.e., reset sets the BCR to $03FF.

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 SSI 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 Register which controls memory wait states and ports B and C each have registers that select the peripheral to be available and control of that peripheral.

4.2.1.1 Bus Control Register (BCR)

Port A Bus Control Register (BCR) is a 16-bit read/write register. It can be programmed to insert waits states in a bus cycle during external memory accesses. 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.

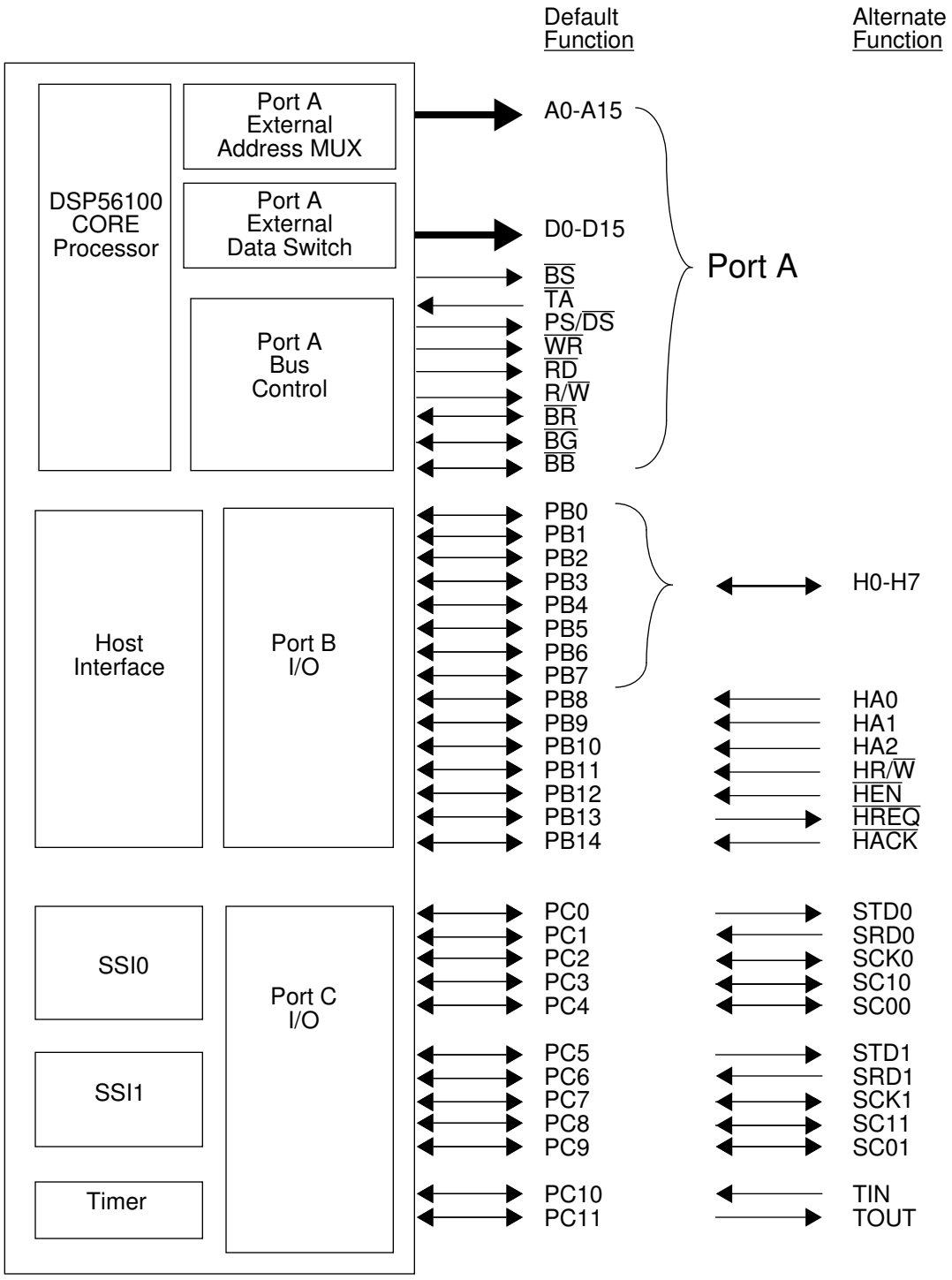


Figure 4-1 DSP56156 Input / Output Block Diagram

Bit 15 of the BCR, the Bus Request Hold bit, (RH), can be used for direct software control of the $\overline{BR}$ pin. When this bit is set, the $\overline{BR}$ pin is asserted even though the DSP does not need the bus. If RH is cleared, the $\overline{BR}$ 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 bus master. If the DSP is not the bus master, BS is cleared. In the slave mode, the BS bit is set when the $\overline{BG}$ output pin is high and cleared when $\overline{BG}$ is low. In the master mode, BS is cleared when the $\overline{BG}$ input pin is high and set when both pins ($\overline{BG}$ and BB) are low. This bit is set by 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), an 12-bit Port C Data Direction Register (PCDDR), and a 12 bit Port C Data Register (PCD). These registers are shown in Figure 4-2. All registers are read/write. Bit manipulation instructions can be used to access individual bits.

RH	BS	*	*	*	*	External X Memory					External P Memory				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PORT A
BUS CONTROL
REGISTER (BCR)
X:$FFDE

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

PORT B
CONTROL
REGISTER (PBC)
X:$FFC0

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

PORT B
DATA DIRECTION
REGISTER (PBDDR)
X:$FFC2

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

PORT B
DATA
REGISTER (PBD)
X:$FFE2

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

PORT C
CONTROL
REGISTER (PCC)
X:$FFC1

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

PORT C
DATA DIRECTION
REGISTER (PCDDR)
X:$FFC3

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

PORT C
DATA
REGISTER (PCD)
X:$FFE3

* Reserved Bits; Read as zero and should be written as zero for future compatibility.

Figure 4-2 I/O Port B and C Programming Models

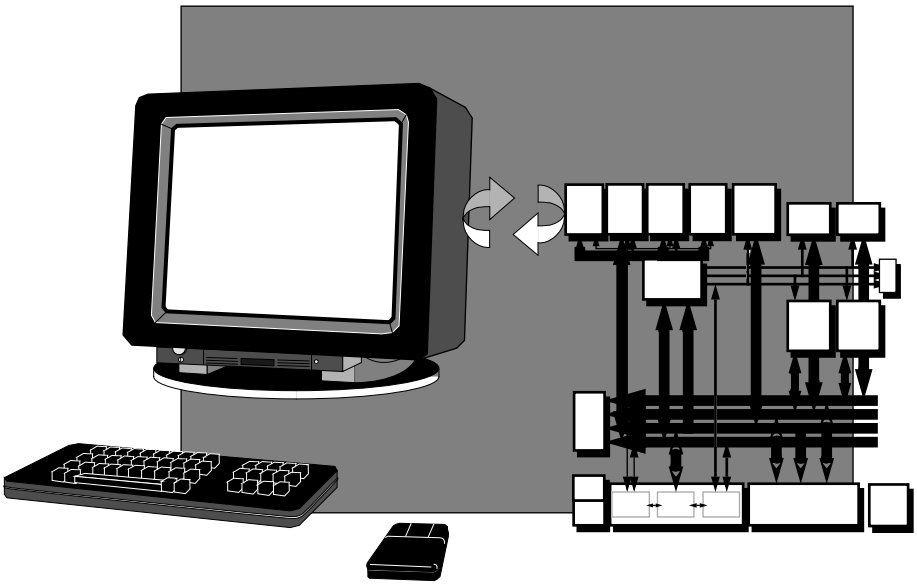
$FFFF	Reserved for on-chip emulation	$FFDF	IPR: Interrupt Priority Register
$FFFE		$FFDE	BCR: Bus Control Register
$FFFD	TSMB1 SSI1 Register	$FFDD	reserved for future use
$FFFC	TSMA1 SSI1 Register	$FFDC	PLCR
$FFFB	RSMB1 SSI1 Register	$FFDB	
$FFFA	RSMA1 SSI1 Register	$FFDA	
$FFF9	TX/RX SSI1 TX/RX Registers	$FFD9	CRB-SSI1 Control Register B
$FFF8	SR/TSR SSI1 Status Register	$FFD8	CRA-SSI1 Control Register A
$FFF7		$FFD7	
$FFF6		$FFD6	
$FFF5	TSMB0 SSI0 Register	$FFD5	
$FFF4	TSMA0 SSI0 Register	$FFD4	
$FFF3	RSMB0 SSI0 Register	$FFD3	
$FFF2	RSMA0 SSI0 Register	$FFD2	
$FFF1	TX/RX SSI0 TX/RX Registers	$FFD1	CRB-SSI0 Control Register B
$FFF0	SR/TSR SSI0 Status Register	$FFD0	CRA-SSI0 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	CRX/CTX	$FFC9	reserved for test
$FFE8	COSR	$FFC8	COCR
$FFE7		$FFC7	
$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)

Note: The underlined addresses indicate

Figure 4-3 DSP56156 On-chip peripherals Memory Map

SECTION 5

HOST INTERFACE



SECTION CONTENTS

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 DSP56156 has an 8-bit bidirectional data bus H0-H7 (PB0-PB7) and 7 control lines 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 DSP56156 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 DSP56156 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 DSP56156 as fast memory and data can be transferred between the host and DSP56156 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 DSP56156 interrupt rate.

The host processor can also issue vectored exception requests to the DSP56156 with the host command feature. The host may select any of the 32 DSP exception routines to be executed by writing a vector address register. This flexibility allows the host programmer to execute a wide number of preprogrammed functions inside the DSP56156. Host exceptions can allow the host processor to read or write DSP56156 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 DSP56156 CPU views the HI as a memory mapped peripheral occupying three 16-bit words in data memory space. The DSP56156 may access the HI as a normal memory-mapped peripheral using standard polled or interrupt programming techniques.

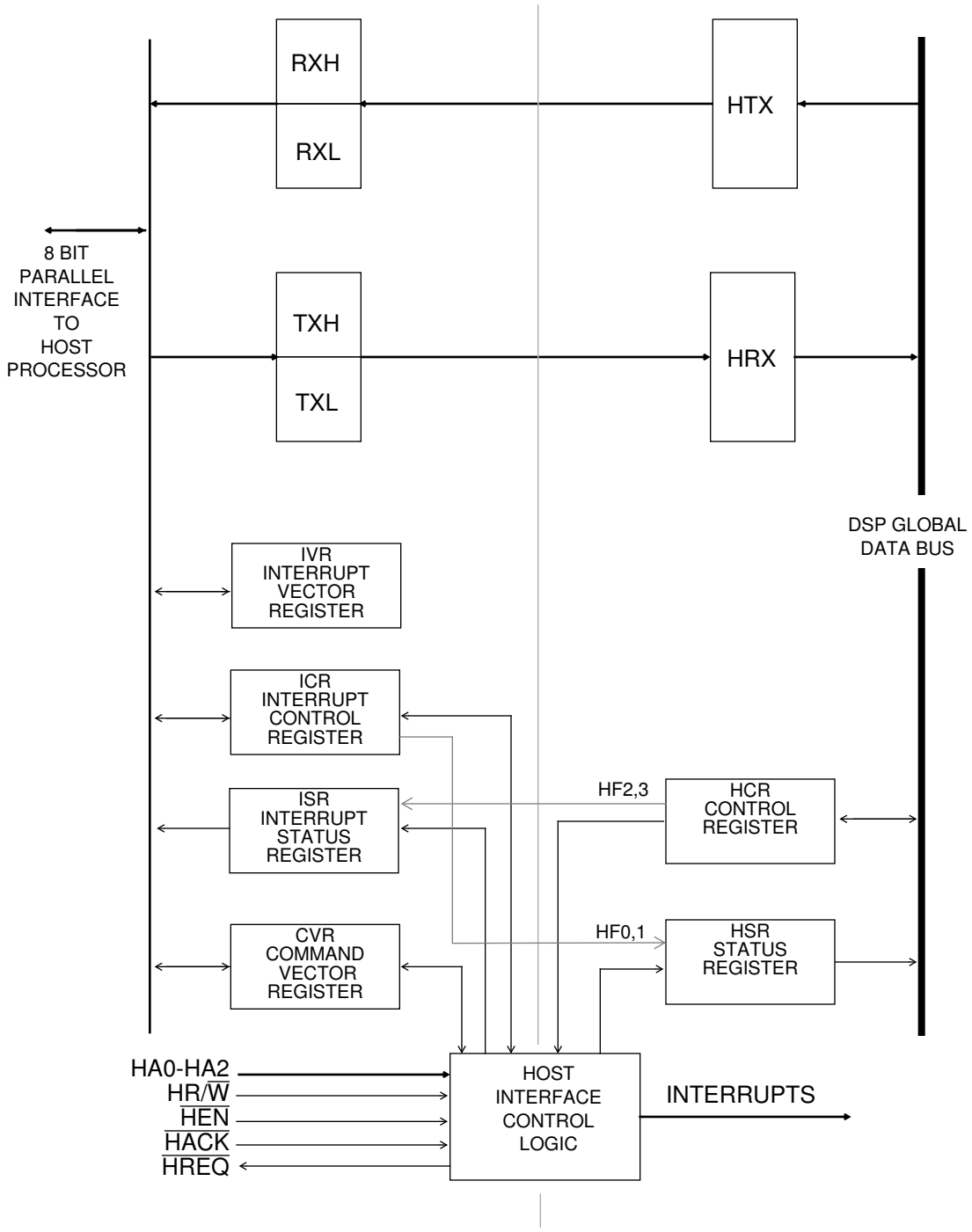


Figure 5-1 Host Interface Block Diagram

5.2 HOST INTERFACE PROGRAMMING MODEL

The HI has two programming models - one for the DSP56156 programmer and one for the host processor programmer. In most cases, the notation used in this manual reflects the DSP56156 perspective. The Host Interface - DSP56156 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 the Receive Low register RXL clears the Receive Data Full RXDF bit. Because reading RXL clears the RXDF status bit, it is normally the last register read during a 16-bit data transfer.

5.5 TRANSMIT BYTE REGISTERS (TXH, TXL)

The Transmit Byte Registers are viewed as two 8-bit write-only registers by the host processor 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

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.

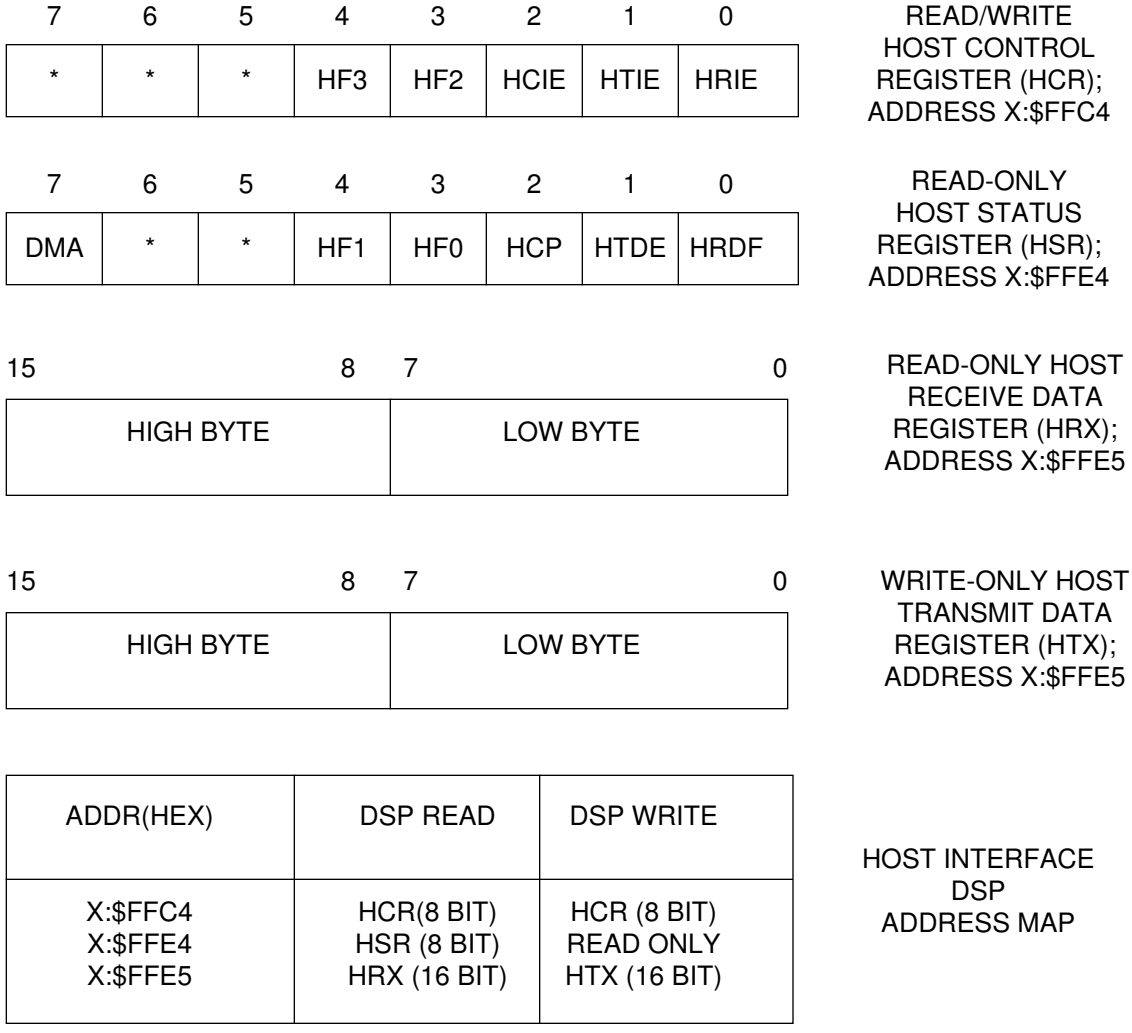


Figure 5-2 Host Interface - DSP Programming Model

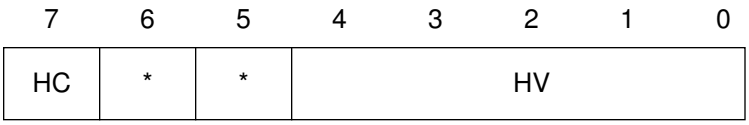
5.6 HOST RECEIVE DATA REGISTER (HRX)

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 Host Command Vector Register (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.



READ/WRITE
COMMAND VECTOR
REGISTER (CVR);
ADDRESS $1

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*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 32 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.

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

READ/WRITE INTERRUPT CONTROL REGISTER (ICR); ADDRESS $0

7	6	5	4	3	2	1	0
HC	*	*	HV				

READ/WRITE COMMAND VECTOR REGISTER (CVR); ADDRESS $1

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

READ-ONLY INTERRUPT STATUS REGISTER (ISR); ADDRESS $2

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

READ/WRITE INTERRUPT VECTOR REGISTER (IVR); ADDRESS $3

15	8	7	0
RXH (ADDR $6) HIGH BYTE		RXL (ADDR $7) LOW BYTE	

READ-ONLY RECEIVE BYTE REGISTERS; ADDRESSES $6,7

15	8	7	0
TXH (ADDR $6) HIGH BYTE		TXL (ADDR $7) LOW BYTE	

WRITE-ONLY TRANSMIT BYTE REGISTERS; ADDRESSES $6,7

ADDR (BIN) HA2	HA1	HA0	HOST READ	HOST WRITE
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)

HOST INTERFACE HOST PROCESSOR ADDRESS MAP

Figure 5-3 Host Interface - Host Processor Programming Model

Table 5-1 Host Interface Interrupt Structure

DSP INTERRUPT STRUCTURE

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

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

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

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:

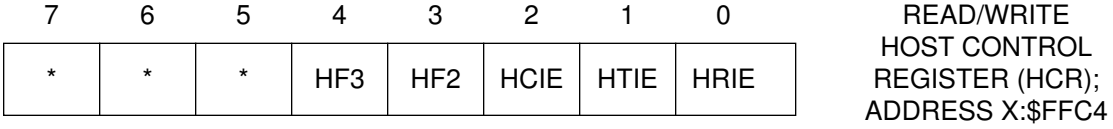
The command exception might be recognized by the DSP and executed before it can be canceled by the host, even if the host clears the HC bit.

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.



READ/WRITE
HOST CONTROL
REGISTER (HCR);
ADDRESS X:$FFC4

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 status and flags of the HI. 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 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
INIT	HM1	HM0	HF1	HF0	*	TREQ	RREQ

READ/WRITE
INTERRUPT CONTROL
REGISTER (ICR)
ADDRESS $0

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 selects 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 2 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 5-5a. 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 5-5a 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 5-5ab INIT Execution Definition

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
HREQ	DMA	*	HF3	HF2	TRDY	TXDE	RXDF

READ-ONLY
INTERRUPT STATUS
REGISTER (ISR)
ADDRESS $2

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.

TRDY = TXDE ^ 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 2 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 2 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
IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0

READ/WRITE
INTERRUPT CONTROL
REGISTER (IVR);
ADDRESS $3

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 (i.e. clearing HRDF or HTDE for example) to clear the interrupt. In the case of Host Command interrupts, the interrupt acknowledge from the program controller will clear the pending interrupt condition.

The host processor interrupts are external and use the Host Request $\overline{\text{HREQ}}$ pin. H$\overline{\text{REQ}}$ 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- 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:

- 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.
- 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, HR/$\overline{\text{W}}$, and $\overline{\text{HEN}}$) 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:

- 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.

- 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, HR/$\overline{W}$, and $\overline{HEN}$) 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 $\overline{\text{HEN}}$ 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 $\overline{\text{HEN}}$ signal and then to block internal updates of the status bits. There is no other minimum $\overline{\text{HEN}}$ assert time relationship to DSP clocks. There is a minimum $\overline{\text{HEN}}$ 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 $\overline{\text{HEN}}$ access for $\overline{\text{HEN}}$ + 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. Because the host does not know exactly when the exception will be recognized, because of synchronization, and because pipelining of exception processing, the DSP may execute the host exception

after the HC bit is cleared. For this reason, 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

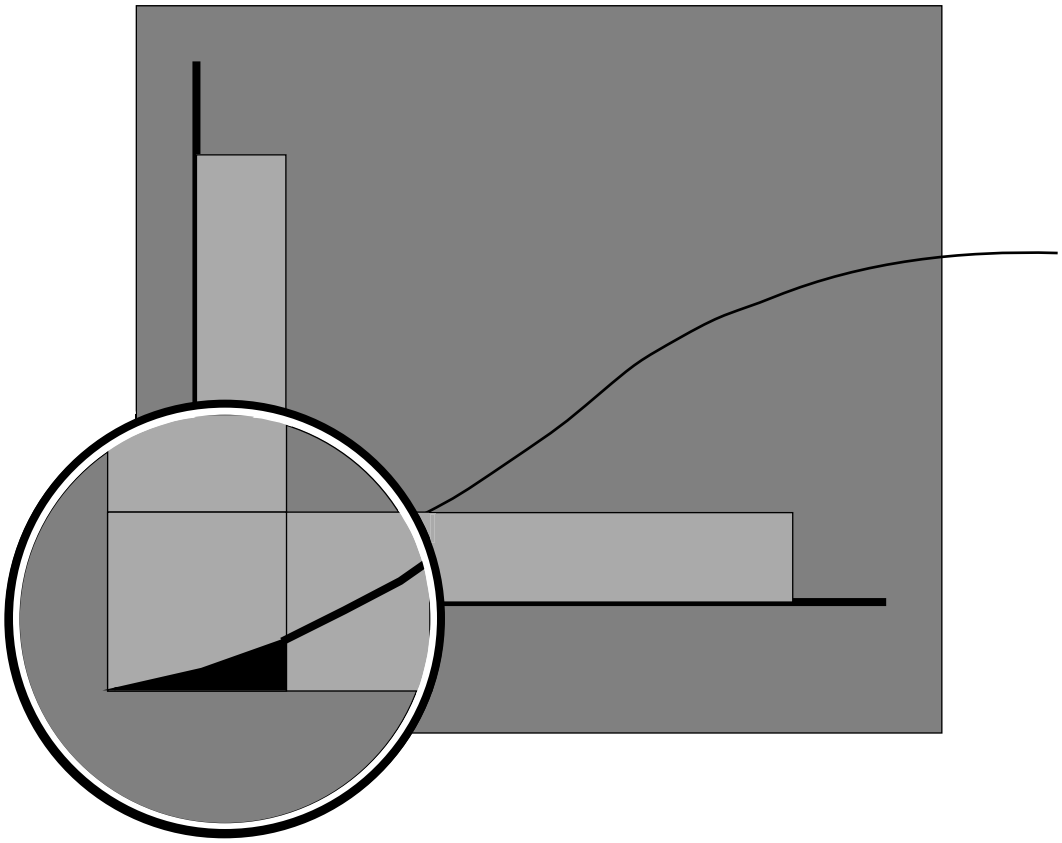
1. Reading HF1 and HF0 as an encoded pair.

   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

DSP56156 ON-CHIP SIGMA/DELTA CODEC



SECTION CONTENTS

6.1 INTRODUCTION

This section describes the DSP56156 Sigma/Delta (ΣΔ) over sampled voice band CODEC block. It discusses the general block diagram of the A/D and D/A sections, the handshake between the DSP56156 core processor and the codec, as well as the last decimation antialiasing filter and first interpolation reconstruction filter performed in software by the DSP56156 core processor.

6.2 GENERAL DESCRIPTION

The ΣΔ over sampled 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 a reference voltage generator, a bias current generator, and a master clock circuit (see Figure 6-1).

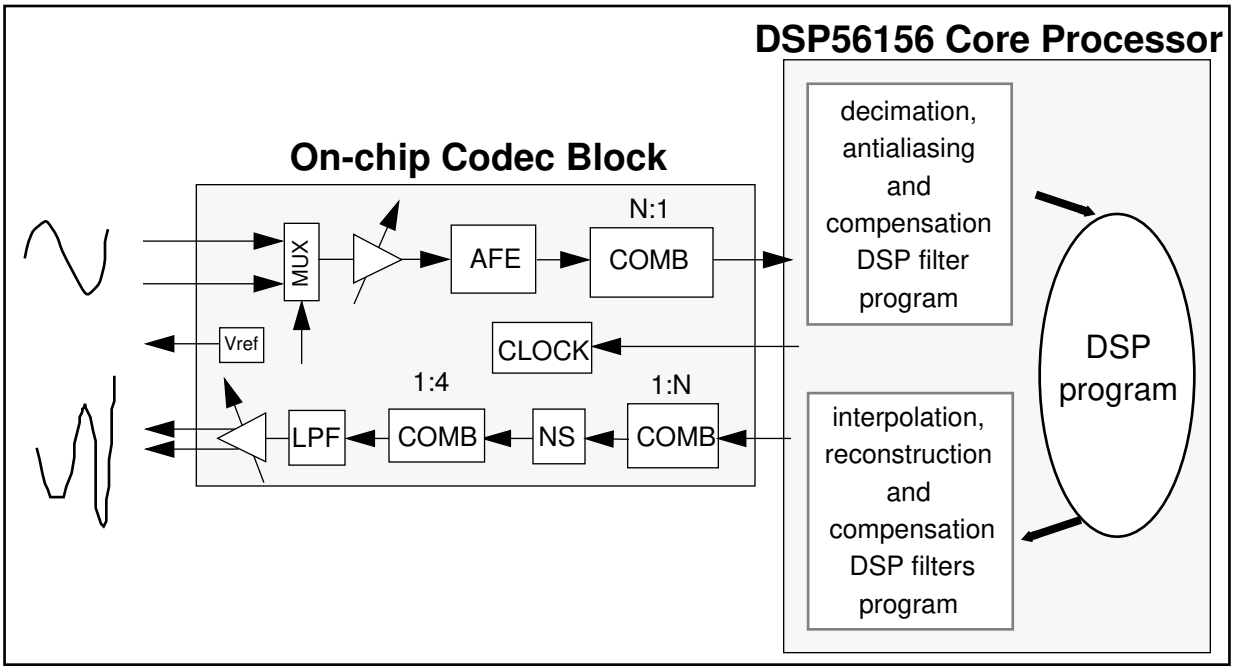


Figure 6-1 DSP56156 On-chip Sigma/Delta Functional Diagram

The A/D converter consists of an analog modulator with selectable input gain, a digital low-pass comb filter, and a parallel bus interface. The analog modulator is a second-order ΣΔ loop constructed of fully differential CMOS switched capacitor circuitry. The analog modulator input is user 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 DSP56156 global data bus.

The D/A converter consists of a second-order comb interpolating filter, a digital ΣΔ modulator, an analog comb decimation filter, an analog low-pass filter, a selectable attenuator, and a differential output stage. The interpolator takes in 16-bit two's complement numbers from the DSP core and up-samples them to a high frequency. The modulator changes these high frequency 16-bit words into a 1-bit stream. The switched capacitor analog filtering removes the out-of-band shaped modulator noise.

This ΣΔ codec block has been designed for maximum flexibility. The user can select one of four decimation (interpolation) ratios for the A/D (D/A) converters. Operating at a nominal sampling rate of 2 MHz, the A/D converter provides a 16-bit digital output with more than 60dB S/(N+D) for input signals in a bandwidth of 0-4 KHz. The D/A converter nominally accepts a 16-bit word at 16 KHz and has a fully differential analog output which provides more than 60dB S/(N+D) in a 0-15 KHz bandwidth, for input signals in a bandwidth of 0-4 KHz. Table 6-1 summarizes the main features of the codec block.

Table 6-1 On-chip Codec Main Features

• 16-bit resolution
• More than 60dB S/(N+D)
• Operates at sampling clock rates between 100 KHz and 3 MHz
• No off-chip components 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 D/A first interpolation filtering stage section are implemented in software by the DSP core in order to reduce the codec cell die area.

6.3 ANALOG I/O DEFINITION

This section describes the Motorola DSP56156 analog input and output characteristics (see Figure 6-2).
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 as 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 Vref±1.0 Vp. The output op-amp is capable of driving a load of 1 kΩ in series with 50 nF between the differential outputs.

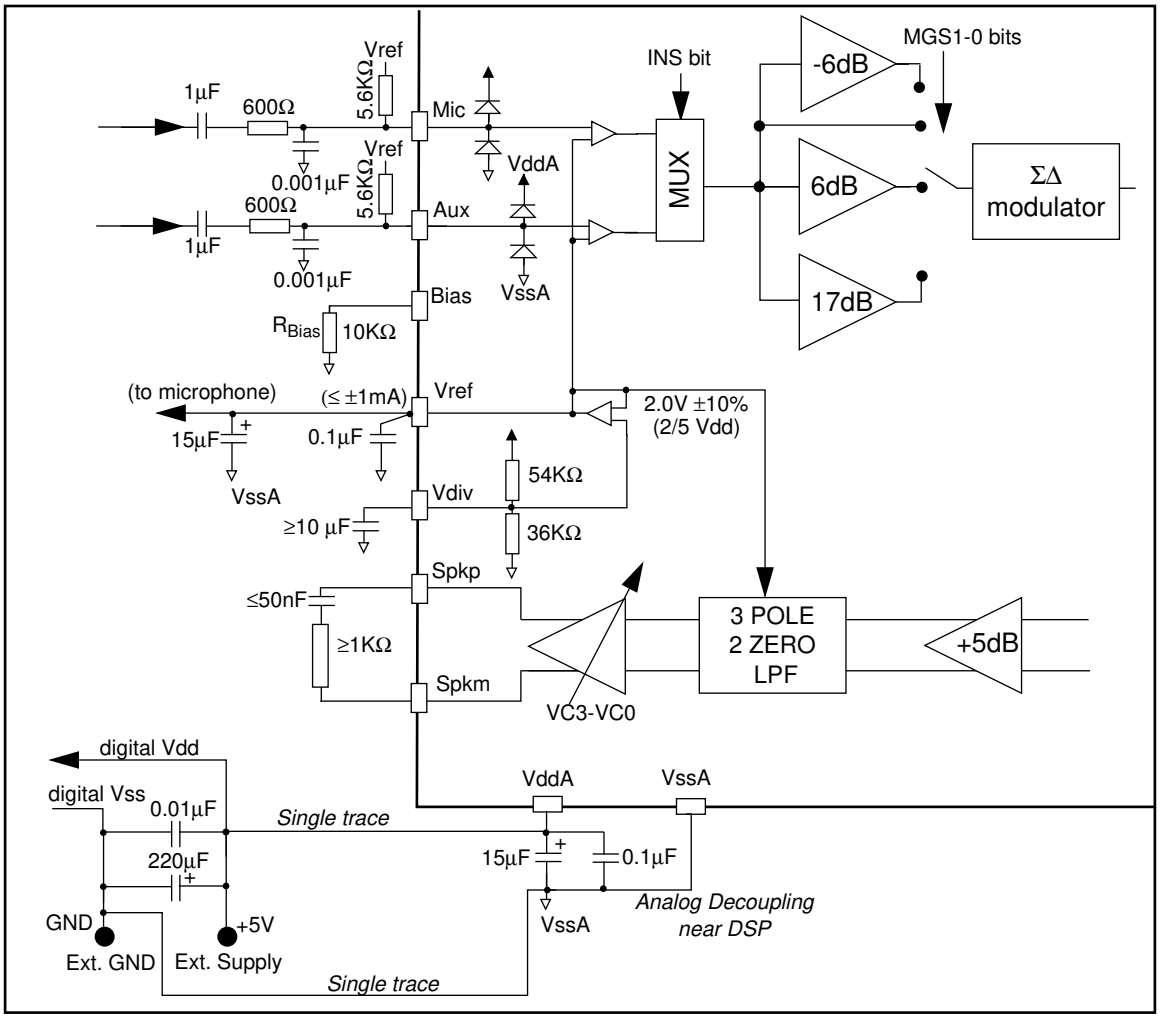


Figure 6-2 DSP56156 Analog Input and Output Diagram

6.4 INTERFACE WITH THE DSP56156 CORE PROCESSOR

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

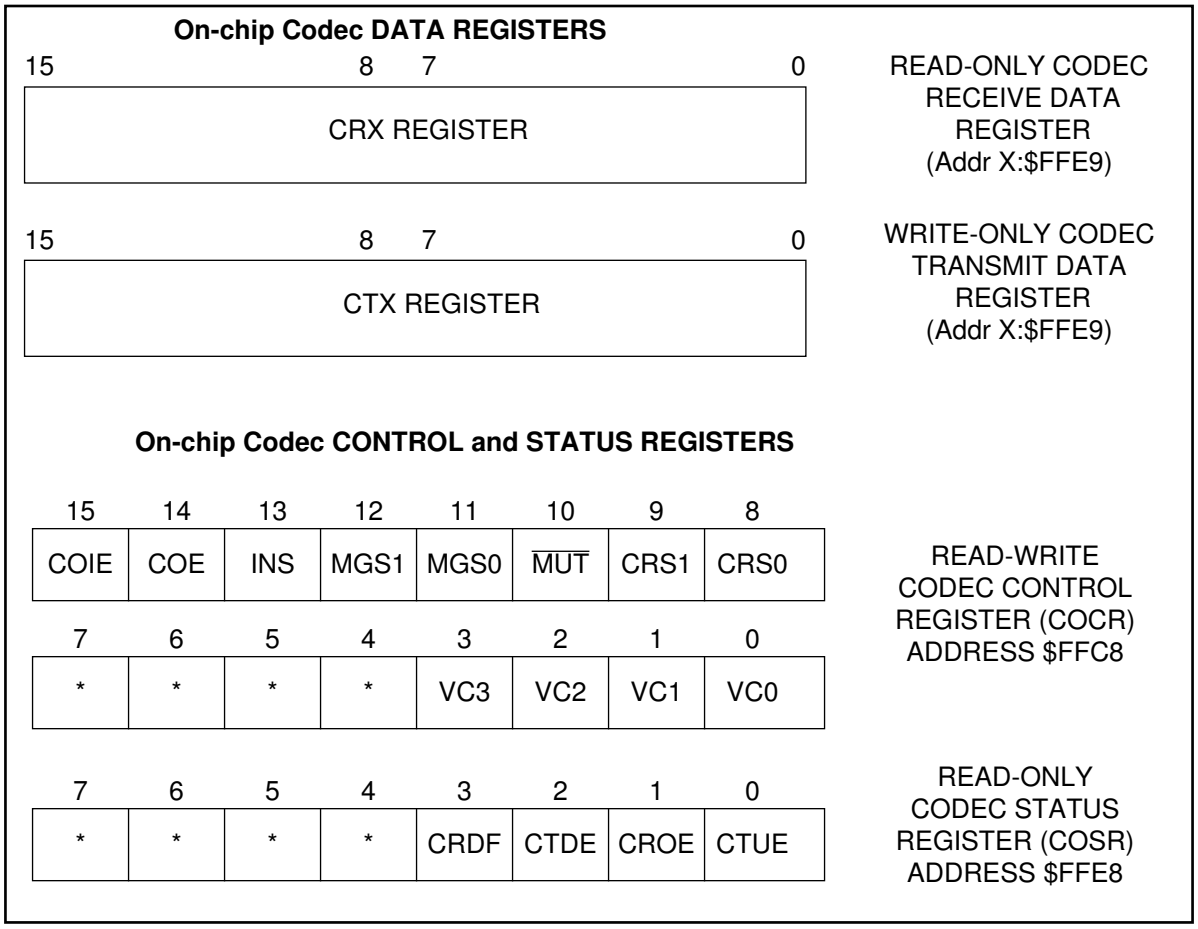
6.4.1 Interface Definition

The ΣΔ section is seen from the core as a memory mapped on-chip peripheral. Data memory locations are used for the receive data register, transmit data register, status register, and control register. One interrupt vector is assigned to the ΣΔ section.

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.

6.4.2 On-chip Codec Programming Model

Figure 6-3 shows the four memory mapped registers (mapped into three memory locations) used by the on-chip codec.



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

Figure 6-3 On-Chip Codec Programming Model

6.4.3 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. The DSP may program the COIE bit to cause a Codec Interrupt when CRDF is set.

6.4.4 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. The DSP may program the COIE bit to cause a Codec Interrupt when CTDE is set.

6.4.5 Codec Control Register (COCR)

The Codec Control Register COCR is a 16-bit read/write register used to direct the on-chip codec operation. The COCR bits are described in the following sections.

All COCR bits are cleared by DSP hardware and software reset.

6.4.5.1 COCR 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-VC2 control the volume between -20 dB and 0 dB by 5 dB steps (see Table 6-2) and VC3 is a boost bit which increases the volume level by 20 dB.

Table 6-2 Audio Level Control

VC3	VC2	VC1	VC0	Relative Level in dB
0	0	0	0	-20
0	0	0	1	-15
0	0	1	0	-10
0	0	1	1	-5
0	1	0	0	0
0	1	0	1	6
0	1	1	0	12
0	1	1	1	18
1	0	0	0	0
1	0	0	1	6
1	0	1	0	12
1	0	1	1	18
1	1	0	0	24
1	1	0	1	30
1	1	1	0	30
1	1	1	1	35

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 6-3 gives an example where the gain of the interpolation digital filter is adjusted in order to provide an output volume control between -20 dB and +35 dB in 5 dB steps.

Table 6-3 Audio Level Control with DSP Filter Gain

VC3	VC2	VC1	VC0	Relative Level dB	Digital Filter Gain	Final Output Level
0	0	0	0	-20	0	-20
0	0	0	1	-15	0	-15
0	0	1	0	-10	0	-10
0	0	1	1	-5	0	-5
0	1	0	0	0	0	0
0	1	0	1	6	-1	5
0	1	1	0	12	-2	10
0	1	1	1	18	-3	15
1	0	0	0	0	0	0
1	0	0	1	6	-1	5
1	0	1	0	12	-2	10
1	0	1	1	18	-3	15
1	1	0	0	24	-4	20
1	1	0	1	30	-5	25
1	1	1	0	30	0	30
1	1	1	1	35	0	35

6.4.5.2 COCR Codec Ratio Select Bits (CRS1-0) Bits 8,9

The Codec Ratio Select Bits are used by the DSP core to program the decimation and interpolation ratio of the on-chip codec comb filter sections. As shown in Table 6-4, 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 overall DC gain of the A/D and D/A sections is discussed in detail Sections 6.5.1 and 6.5.2.

Table 6-4 Decimation/Interpolation Ratio Control

CRS1	CRS0	Decimation Interpolation Ratio Rate	A/D Comb Filter DC Gain		D/A Comb Filter DC Gain	
0	0	125	$125^3/2^{21}$	-0.618 dB	125/128	-0.206 dB
0	1	128	1	0 dB	1	0 dB
1	0	105	$2(105^3)/2^{21}$	0.859 dB	105/128	-1.720 dB
1	1	81	$2(81^3)/2^{21}$	0.118 dB	81/128	-3.974 dB

6.4.5.3 COCR 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.

6.4.5.4 COCR Microphone Gain Select Bits (MGS1-0) Bits 11,12

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 3dB below the full scale saturation values. The full scale saturation analog values result in a maximum digital A/D output ($7FFF) when the A/D comb filter has a 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

6.4.5.5 COCR 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, AUX input is selected.

6.4.5.6 COCR 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.

6.4.5.7 COCR 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 this bit enables the interrupt.

6.4.5.8 COCR Reserved Bits (Bits 4-7)

These bits are reserved. They should be written as zero by the user program and will read as zero.

6.4.6 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.

6.4.6.1 COSR Codec Transmit Under Run Error FLag Bit (CTUE) Bit 0

The Codec Transmit Under Run 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 (under run error). In this case, the previous sample written to the CTX register is re-transmitted to the D/A section.

Hardware and software reset and STOP reset clear CTUE. CTUE is also cleared by reading the COSR with CTUE set followed by writing CTX. Clearing the COE bit in the COCR does not affect CTUE.

6.4.6.2 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 and software reset and STOP reset clear CROE. CROE is also cleared by reading the COSR with CROE set followed by reading CRX. Clearing the COE bit in the COCR does not affect CTUE.

6.4.6.3 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 $\overline{\text{RESET}}$ and RESET instruction) and STOP reset.

6.4.6.4 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 $\overline{\text{RESET}}$ and RESET instruction) and STOP reset.

6.4.6.5 COSR Reserved Bits (Bits 4-15)

These bits are reserved. They should be written as zero by the user program and will read as zero.

Table 6-6 On-Chip Codec Programming Model Summary

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 Under run Error
	1	Transmit Under run Error

On-chip Codec CONTROL (COCR) X:$FFC8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COIE	COE	INS	MGS1	MGS0	$\overline{\text{MUT}}$	CRS1	CRS0	CT3	CT2	*	*	VC3	VC2	VC1	VC0

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 -6 dB
	01	MIC or AUX amplifier gain of 0 dB
	10	MIC or AUX amplifier gain of 6 dB
	11	MIC or AUX amplifier gain of 17 dB
$\overline{\text{MUT}}$	0	Speaker output muted
	1	Speaker output active
CS1-CS0 Ratio Select	00	125 selected as decimation ratio for the A/D and interpolation ratio of the D/A
	01	128 selected as decimation ratio for the A/D and interpolation ratio of the D/A
	10	105 selected as decimation ratio for the A/D and interpolation ratio of the D/A
	11	81 selected as decimation ratio for the A/D and interpolation ratio of the D/A
VC3-VC0 D/A output Gain in dB	$0	-20 dB Output volume gain
	$1	-15 dB Output volume gain
	$2	-10 dB Output volume gain
	$3	-5 dB Output volume gain
	$4	0 dB Output volume gain
	$5	6 dB Output volume gain
	$6	12 dB Output volume gain
	$7	18 dB Output volume gain
	$8	0 dB Output volume gain
	$9	6 dB Output volume gain
	$A	12 dB Output volume gain
	$B	18 dB Output volume gain
	$C	24 dB Output volume gain
	$D	30 dB Output volume gain
	$E	30 dB Output volume gain
	$F	35 dB Output volume gain

6.5 ON-CHIP CODEC FREQUENCY RESPONSE AND GAIN 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.

6.5.1 A/D Section Frequency Response and DC Gain

The DC gain and the A/D comb filter frequency response depends on the decimation rates selected by programming bits CRS1-CRS0 in the COCR. Table 6-7 shows the DC gain of the A/D section as a function of the decimation ratio.

Table 6-7 A/D Section DC Gain

CRS1	CRS0	Decimation Ratio Rate	DC gain of the A/D section	
			dB	Actual
0	0	125	-0.618 dB	0.9313(=$125^3/128^3$)
0	1	128	0 dB	1
1	0	105	0.859 dB	1.104(=2[105^3]/128^3)
1	1	81	0.118 dB	1.0137(=4[81^3]/128^3)

Eqn. 6-1 gives the A/D transfer function and frequency response function:

Eqn. 6-1 A/D Comb Filter Transfer Function

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

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

D: decimation ratio
F: ΣΔ modulator clock
c=1 for D=125,128
c=2 for D=105
c=4 for D=81

Figure 6-4 and Figure 6-5 show an example of the A/D comb filter log magnitude response using a 2.048 MHz master clock and a decimation ratio of D=128. The figures show the frequency response in the bands 0-1.024 MHz and 0-16 KHz (Figure 6-4) and in the band 0-4 KHz (Figure 6-5).

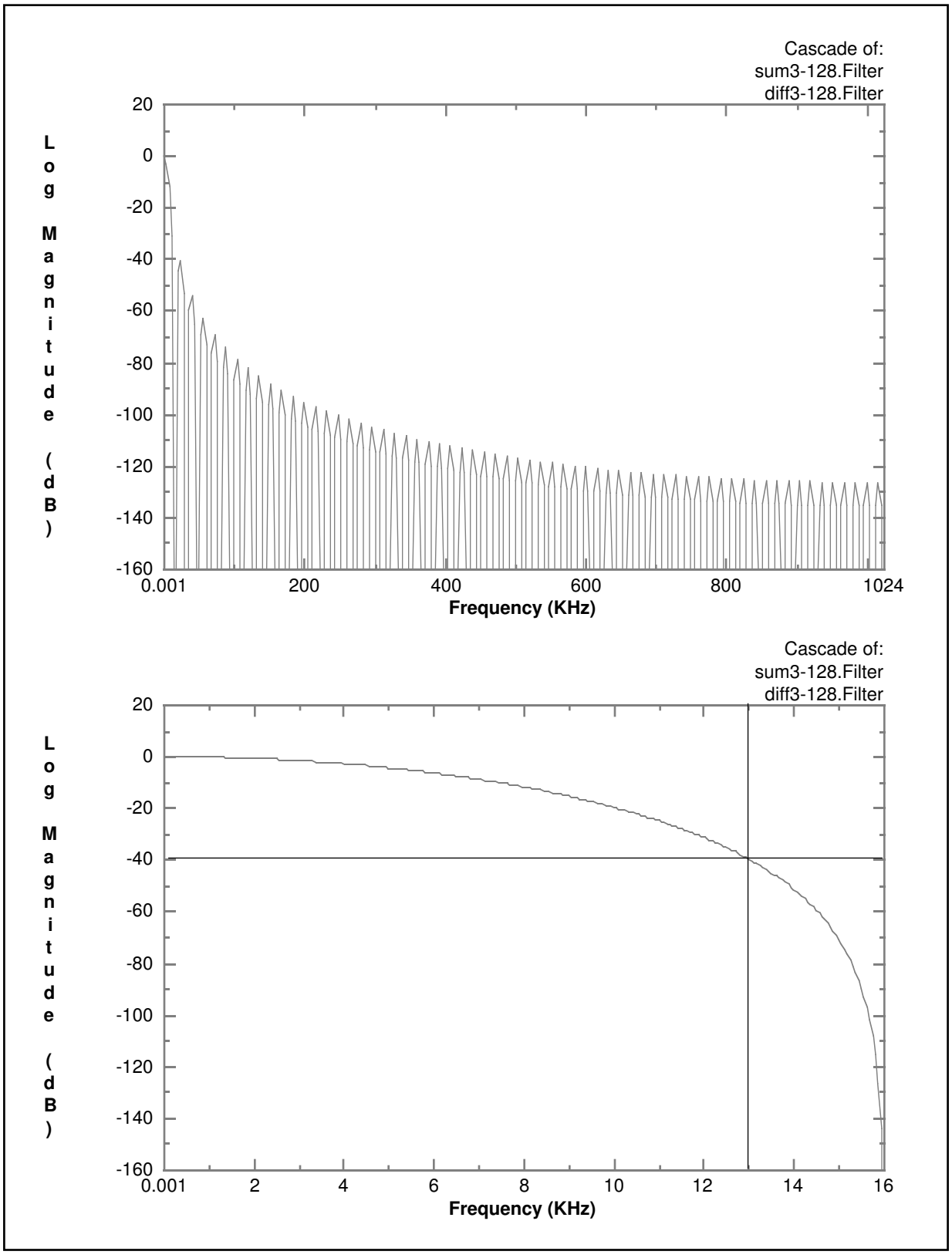


Figure 6-4 Log Magnitude Frequency Response of the A/D Comb Filter for F=2.048 MHz and D=128

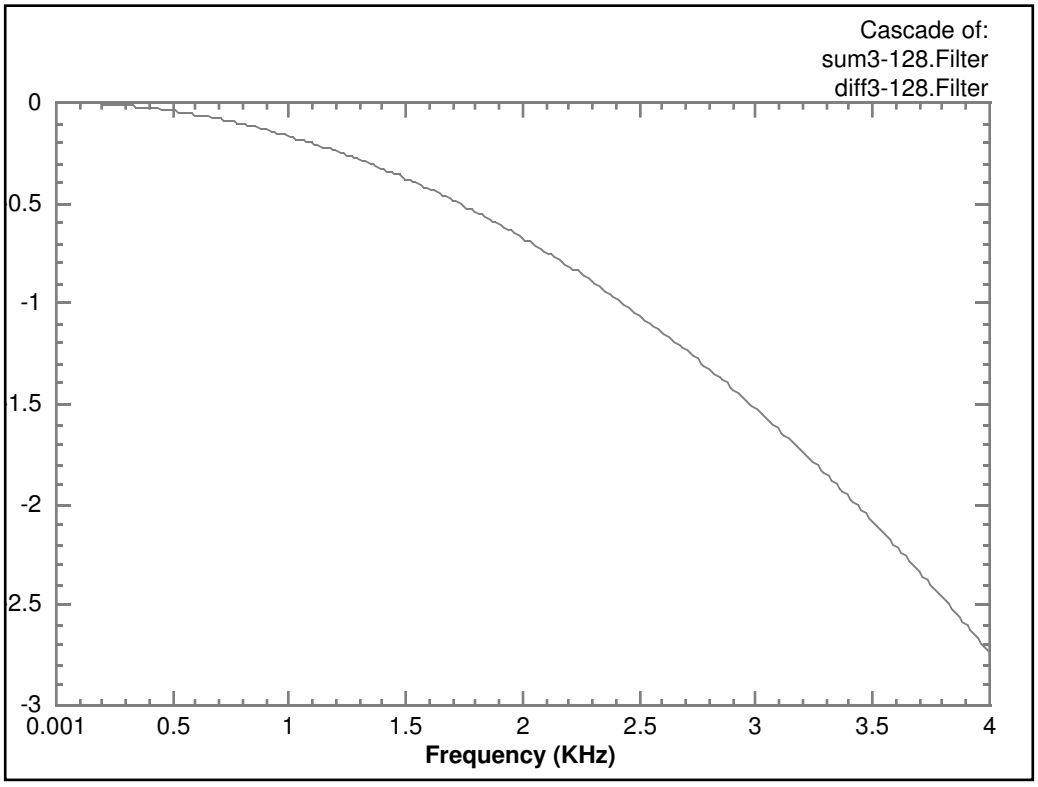


Figure 6-5 Log Magnitude Frequency Response of the A/D Comb Filter in the 0-4 KHz Band for F=2.048 MHz and D=128

The 3 dB drop-off in the band 0-4 KHz shown in Figure 6-5 can be compensated by the decimation filter inside the DSP core. In the present example, with D=128 and F=2.048 MHz, the frequency response of the comb looks like:

$$F(f) = \frac{1}{(128)^3}\left(\frac{\sin\left(\frac{2\pi \times 128}{2 \times 2048} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 2048} \times f\right)}\right)^3$$

The frequency response of the compensating DSP filter can be shaped by 1/F(f) in order to flatten the response of the A/D section.

Table 6-8 gives an example of a 4 biquad IIR low pass filter whose transfer function has been shaped accordingly.

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

Table 6-8 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      $edc9           ; -d2_1 = -0.284627003/2
                dc      $16c4           ; -d1_1 = 0.3557032662/2
                dc      $316a           ; n2_1 = 0.7720730807/2
                dc      $4186           ; n1_1 = 1.023796768/2
                ; Biquad stage no. 2
                dc      $d708           ; -d2_2 = -0.6401634264/2
                dc      $1721           ; -d1_2 = 0.3613604173/2
                dc      $0d0c           ; n2_2 = 0.2038857781/2
                dc      $12dc           ; n1_2 = 0.2946510536/2
                ; Biquad stage no. 3
                dc      $f39a           ; -d2_3 = -0.1937047354/2
                dc      $29f3           ; -d1_3 = 0.6554721197/2
                dc      $3454           ; n2_3 = 0.8176134701/2
                dc      $4328           ; n1_3 = 1.049344022/2
                ; Biquad stage no. 4
                dc      $c6af           ; -d2_4 = -0.8955455145/2
                dc      $0b1d           ; -d1_4 = 20.1736521771/2
                dc      $2ed3           ; n2_4 = 0.7316442217/2
                dc      $1b24           ; n1_4 = 0.42404578/2
```

NOTE: This filter, as well as all the figures representing filter responses, has been generated using ZOLA Technologies, Inc., DSP Designer™ software package.

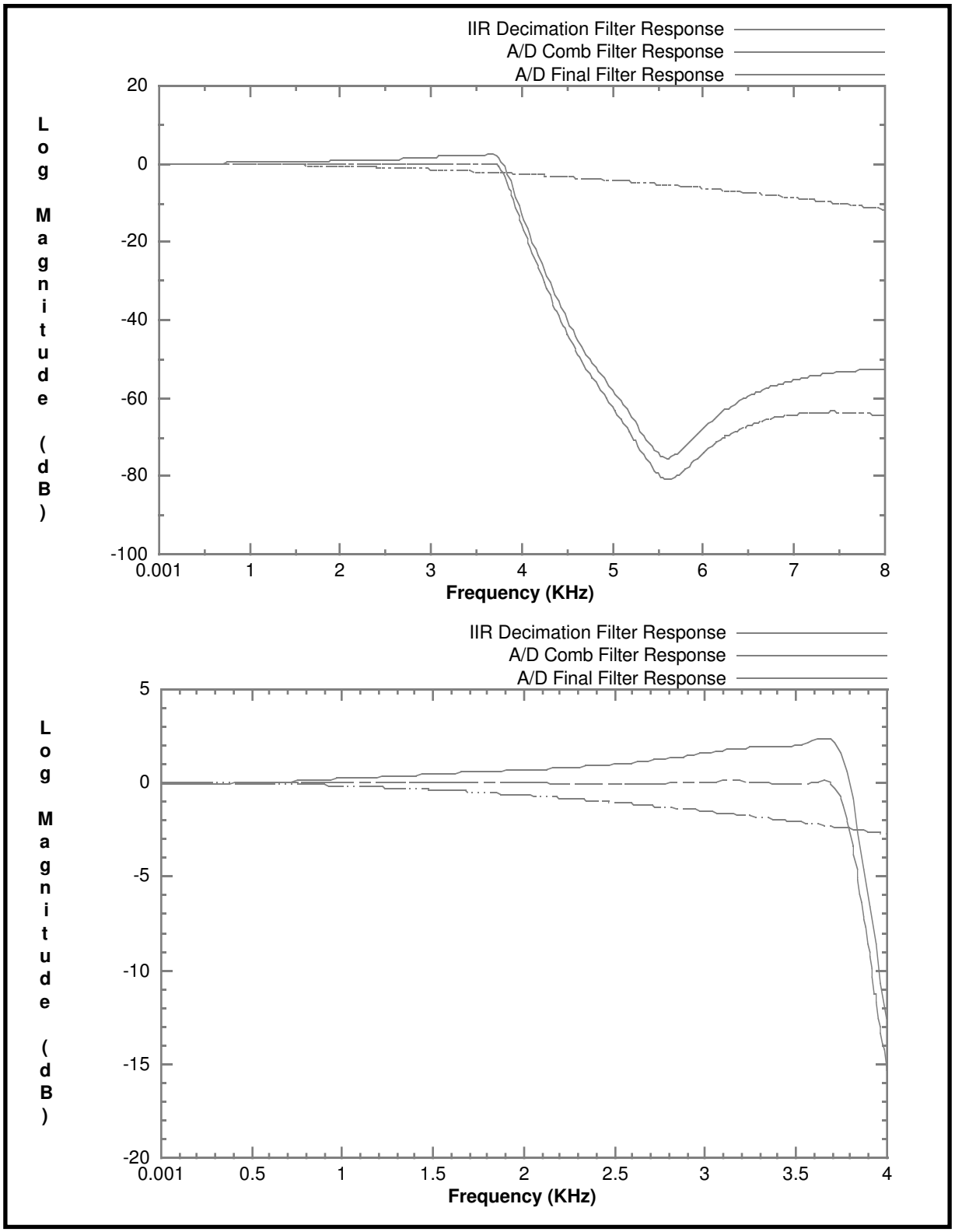


Figure 6-6 IIR Decimation and A/D Section Log Magnitude Frequency Response for F=2.048 MHz and D=128

6.5.2 D/A Section Frequency Response and DC Gain

The D/A section DC gain and the frequency response depend on the interpolation rates selected by programming CRS1-CRS0 in the COCR. In addition, a positive 5 dB DC gain has been introduced between the third order analog comb filter and the analog low pass filter (see Figure 6-2).

Table 6-9 shows the D/A section DC gain when the audio level is set to 0dB by the audio level control bits VC3-VC0 in the COCR.

Table 6-9 D/A Section DC Gain

CRS1	CRS0	Decimation Ratio Rate	DC gain of the D/A 2nd order Comb Filter	DC gain of the D/A section	
0	0	125	-0.206 dB	4.794 dB	1.7366
0	1	128	0 dB	5 dB	1.7783
1	0	105	-1.720 dB	3.28 dB	1.4588
1	1	81	-3.974 dB	1.026 dB	1.1254

The positive gain of the D/A section allows the interpolation digital filter performed by the DSP core to compensate the D/A comb filter frequency response without risk of clipping the higher frequency components in the pass band spectrum.

The on-chip codec D/A section is formed of three different filtering steps which are studied in the next subsections. The frequency responses of the different D/A sections are illustrated in Figure 6-7.

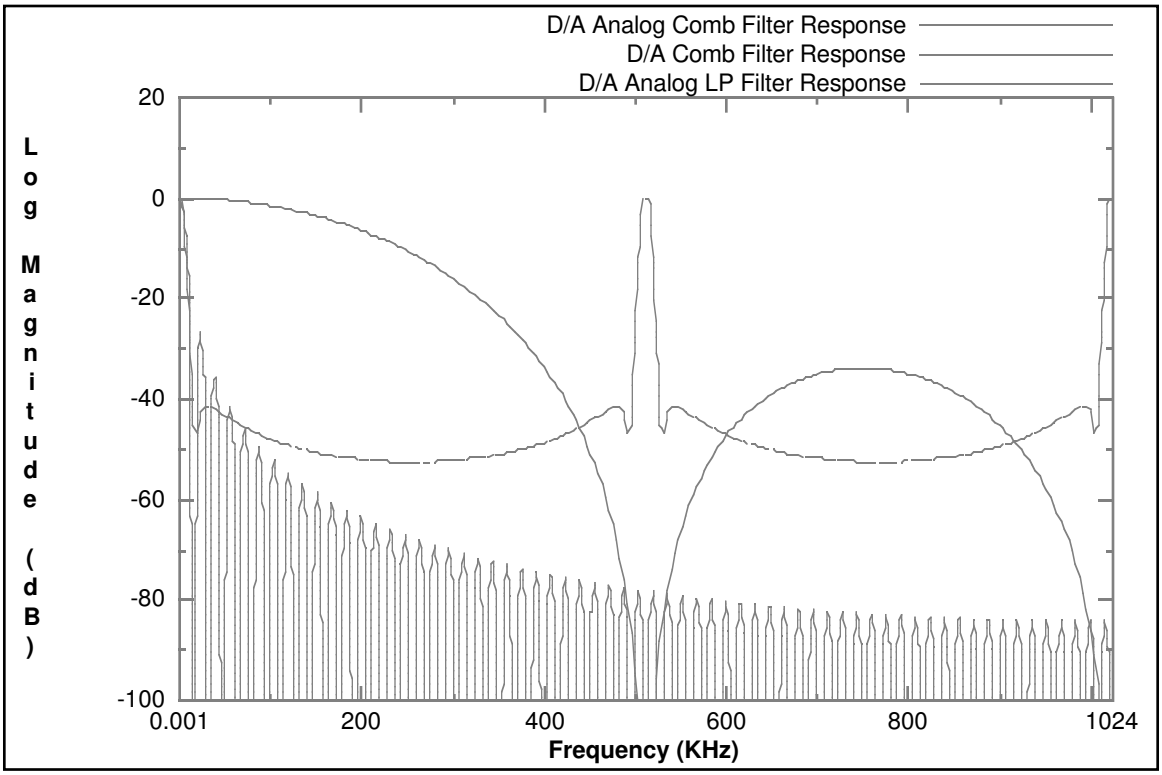


Figure 6-7 Log Magnitude Frequency Responses of the Three Sections in the D/A for F=2.048 MHz and D=128

6.5.2.1 D/A Second Order Digital Comb Filter

This second order comb filter interpolates the 16-bit signal coming from the DSP core. It consists of a digital differentiator which runs at the slow clock rate followed by an integrator at the fast clock rate.
Eqn. 6-2 gives the transfer function and frequency response function:

Eqn. 6-2 D/A Comb Filter Transfer Function

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

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

D: interpolation ratio
F: comb filter integrator clock

Figure 6-8 shows an example of the D/A digital comb filter log magnitude response using a 2.048 MHz master clock and a decimation ratio of D=128. The figure shows the comb filter frequency response in the bands 0-1024 KHz and 0-16 KHz.

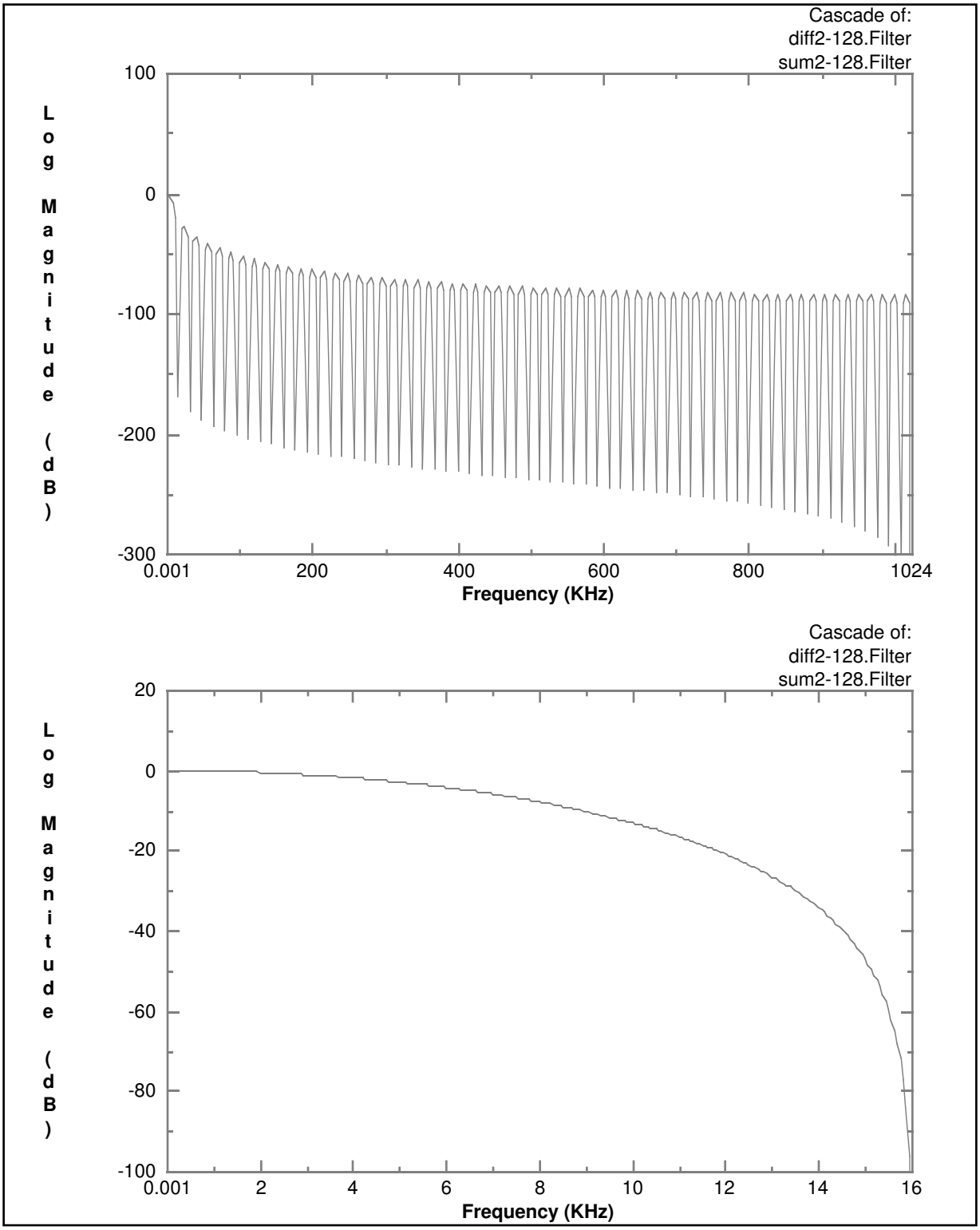


Figure 6-8 Log Magnitude Frequency Response of the D/A Comb Filter for F=2.048MHz and D=128

6.5.2.2 D/A Analog Comb Decimating Filter

The analog comb filter is a third-order decimate-by-four low-pass filter implemented in fully differential switched capacitor circuitry. The 1-bit digital input stream coming from the digital modulator is converted to an analog signal by controlling the switching of input capacitors between VREF and VGNDA. The purpose of the filter is to prevent aliasing of out-of-band noise when the clock rate is reduced by a factor of four. Eqn. 6-3 gives the transfer function and the frequency response of the filter:

Eqn. 6-3 D/A Analog Comb Filter Transfer Function

$$H(z) = \frac{1}{4^3}\left[\frac{1 - z^{-4}}{1 - z^{-1}}\right]^3$$

$$F(f) = \frac{1}{4^3}\left(\frac{\sin\left(\frac{2\pi \times 4}{2F} \times f\right)}{\sin\left(\frac{2\pi}{2F} \times f\right)}\right)^3$$

F: comb filter integrator clock

Figure 6-9 and Figure 6-10 show an example of the D/A analog comb filter log magnitude response using a 2.048 MHz master clock. The figures show the comb filter frequency response in the 0-1024 KHz, 0-256 KHz, and 0-8 KHz bands. It can be noticed that the response in the band 0-8 KHz is flat.

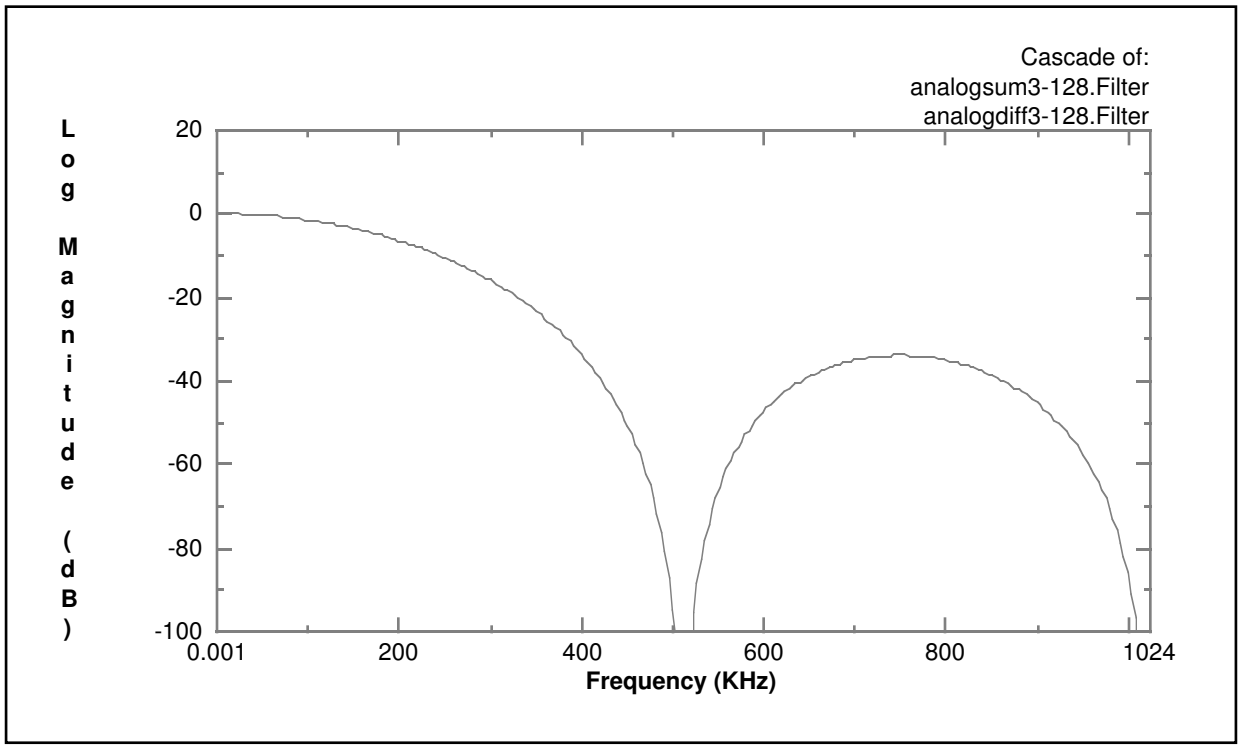


Figure 6-9 Log Magnitude Frequency Response of the D/A Analog Comb Filter for F=2.048 MHz

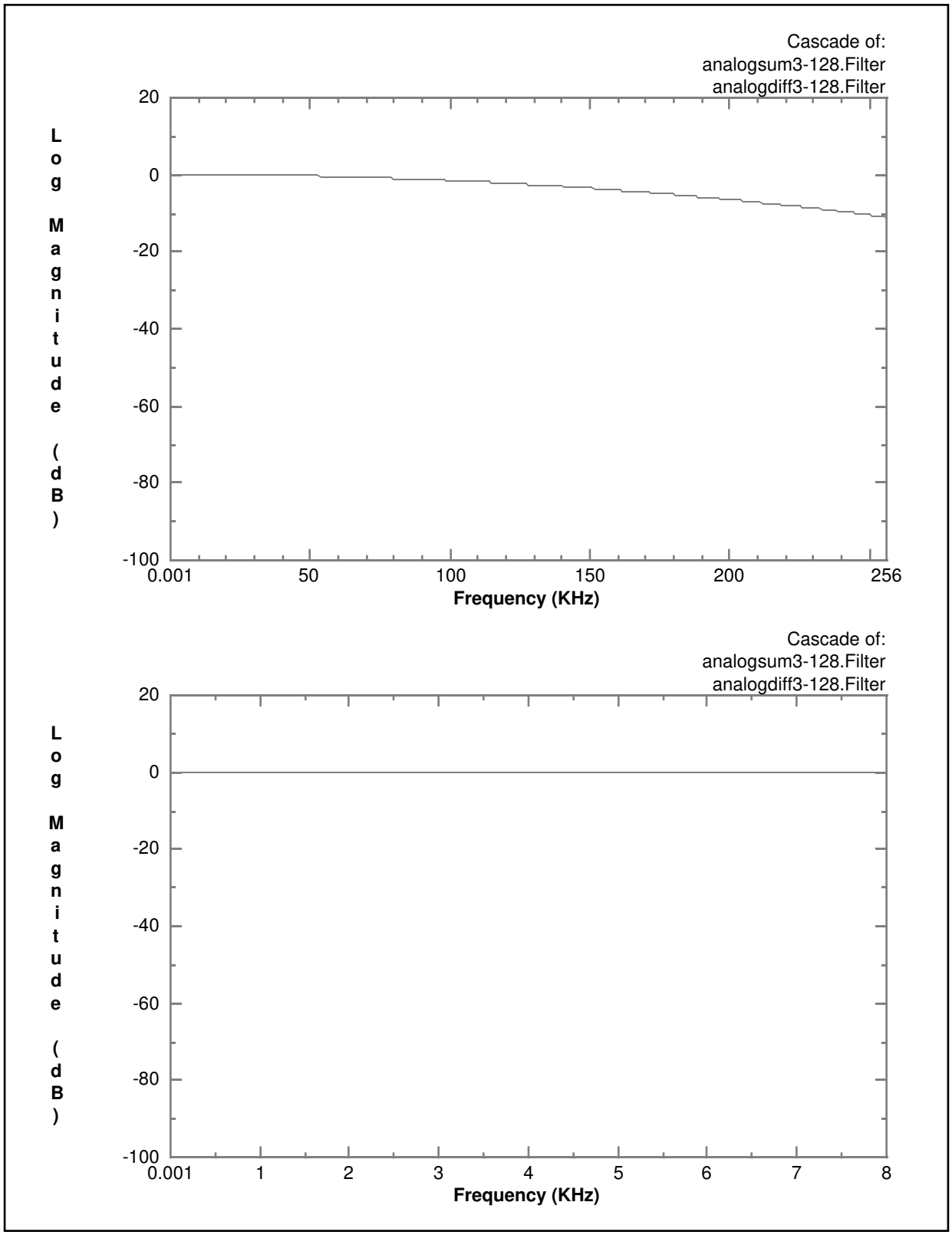


Figure 6-10 Log Magnitude Frequency Response of the D/A Analog Comb Filter for F=2.048 MHz

6.5.2.3 D/A Analog Low Pass Filter

The analog low-pass filter is a three-pole, two zero switched capacitor filter. The purpose of this filter is to attenuate the out-of-band quantization noise and the images created by the up-sampling. This filter is clocked at the slow D/A clock rate. Eqn. 6-4 gives the filter transfer function:

Eqn. 6-4 Analog Low-pass Filter Transfer Function

$$H(z) = \frac{(0.004514)(1 - 1.9532z^{-1} + z^{-2})}{(1 - 0.9501z^{-1})(1 - 1.9504z^{-1} + 0.9546z^{-2})}$$

Figure 6-11 shows an example of the D/A analog low-pass filter log magnitude response using a 512Khz master clock. This figure shows the filter frequency response in the bands 0- 256 KHz and 0-8 KHz.

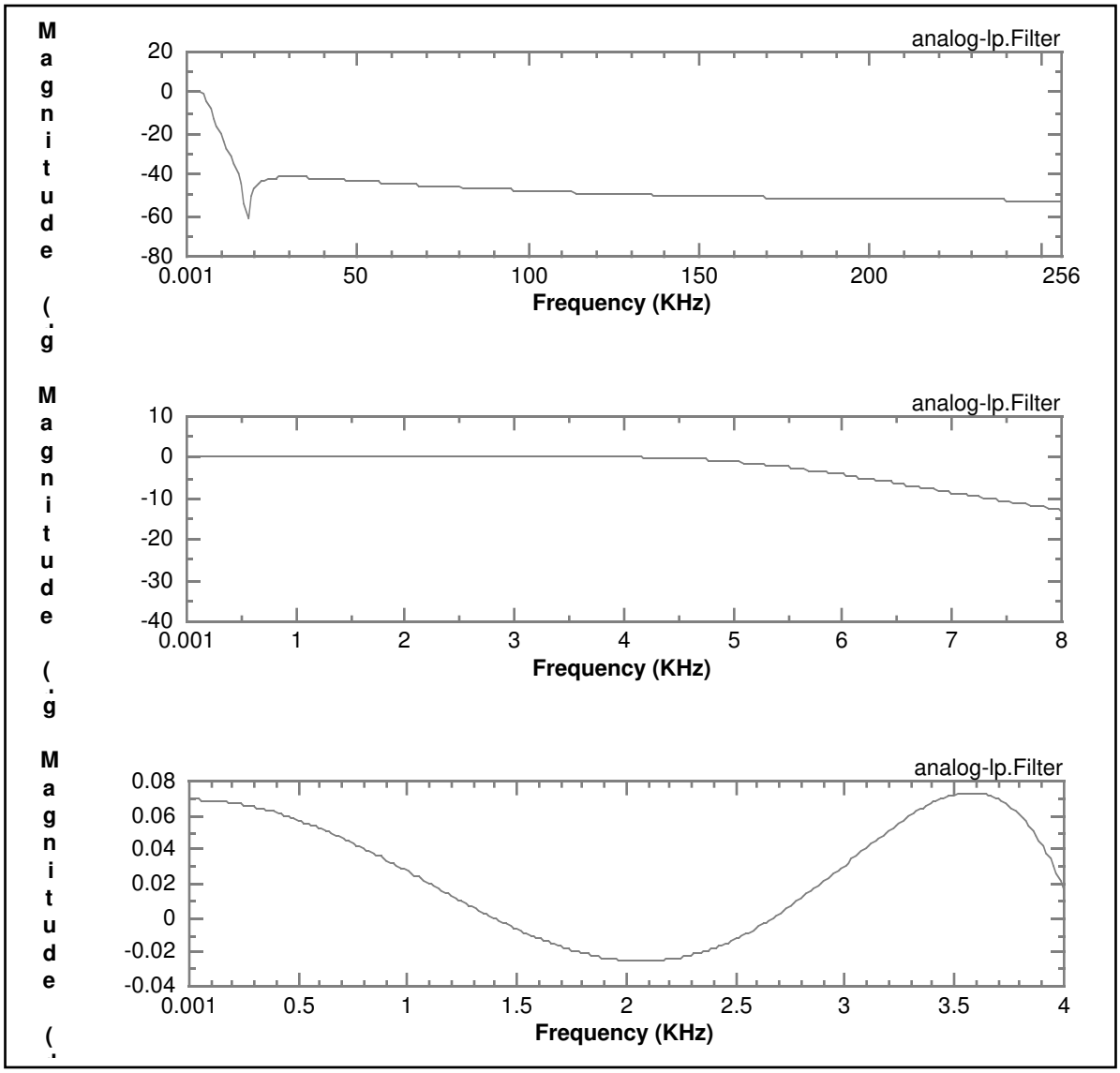


Figure 6-11 Log Magnitude Frequency Response of the D/A Analog Low-pass Filter for F=512 KHz

6.5.2.4 D/A Section Overall Frequency Response

Figure 6-12 and Figure 6-13 show the overall D/A section frequency response.

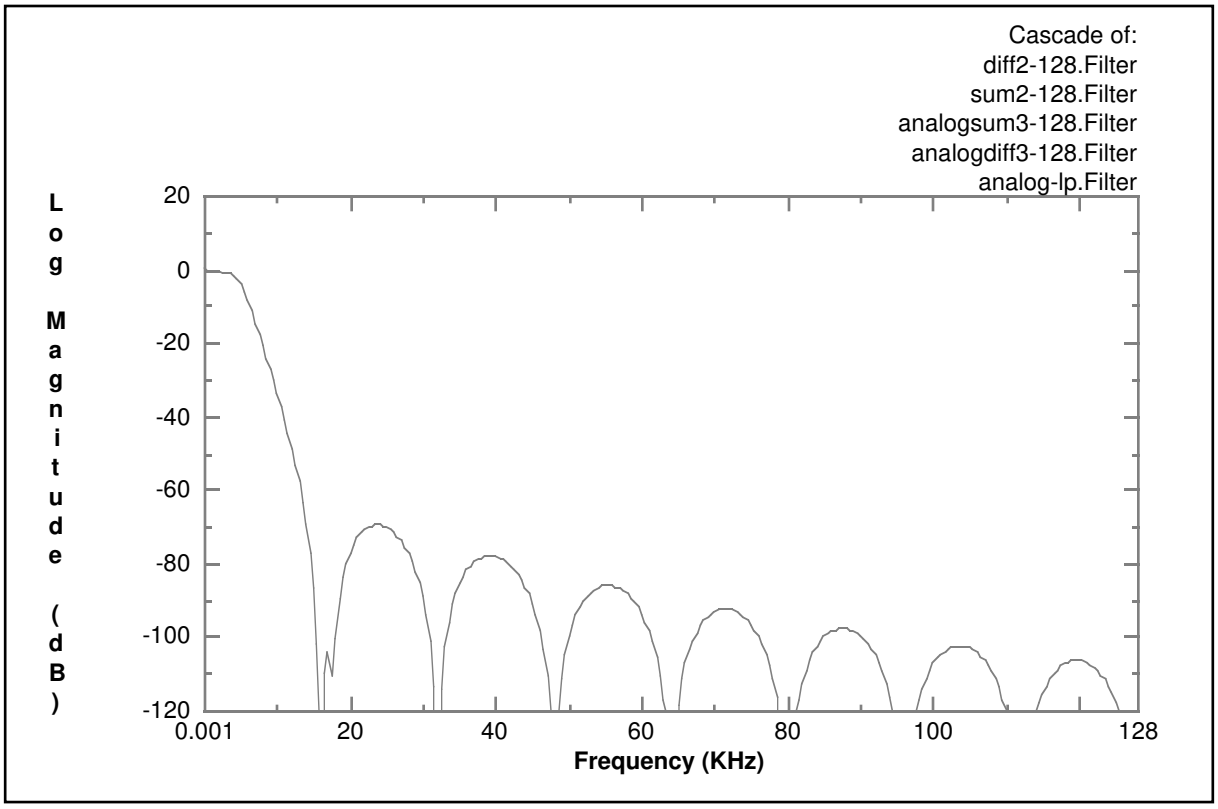


Figure 6-12 Log Magnitude Frequency Response of the D/A Section for F=2.048 MHz and D=128

The frequency response of Figure 6-13(b) is not flat in the 0-4 KHz band. The interpolation filter performed by the DSP core can compensate for this drop.

The D/A analog comb filter frequency response is flat in the audio band and does not need any compensation (see Figure 6-10).

The digital second order D/A comb filter is the only filter that needs to be compensated.

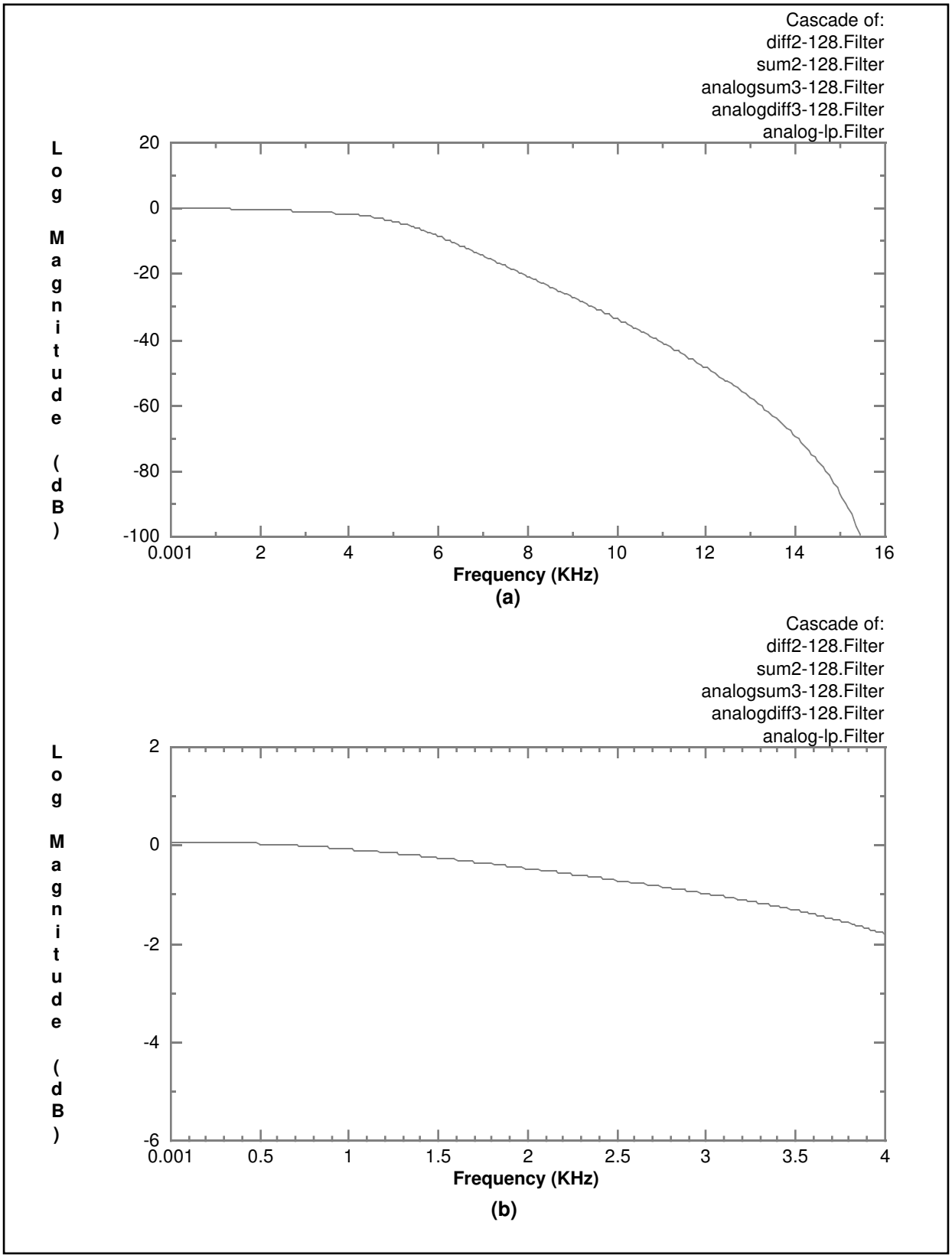


Figure 6-13 Log Magnitude Frequency Response of the D/A Section for F=2.048 MHz and D=128

In the present example, with D=128 and F=2.048 MHz, the frequency response of this second order comb filter is:

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

The frequency response of the interpolation DSP filter can be shaped by 1/F(f) in order to flatten the D/A section response.

Table 6-10 gives an example of a 4 biquad IIR low pass filter whose transfer function has been shaped accordingly.Figure 6-14 shows the filter frequency response and the effects on the overall D/A section response.

Table 6-10 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      $e722           ; -d2_1 = -0.3885309315/2
                dc      $16e0           ; -d1_1 = 0.3574372286/2
                dc      $329e           ; n2_1 = 0.7908895273/2
                dc      $43d3           ; n1_1 = 1.059754603/2
                ; Biquad stage no. 2
                dc      $d947           ; -d2_2 = -0.6050537737/2
                dc      $13eb           ; -d1_2 = 0.3112185033/2
                dc      $1014           ; n2_2 = 0.2512476586/2
                dc      $1146           ; n1_2 = 0.2698979411/2
                ; Biquad stage no. 3
                dc      $f1a9           ; -d2_3 = -0.2240856408/2
                dc      $30ad           ; -d1_3 = 0.7605800752/2
                dc      $3603           ; n2_3 = 0.8439490258/2
                dc      $4614           ; n1_3 = 1.094956692/2
                ; Biquad stage no. 4
                dc      $c72a           ; -d2_4 = -0.8880624617/2
                dc      $0a66           ; -d1_4 = 0.162456827/2
                dc      $2f25           ; n2_4 = 0.7366593399/2
                dc      $1c8f           ; n1_4 = 0.4462262322/2
```

NOTE: This filter, as well as all the figures representing filter responses, has been generated using ZOLA Technologies, Inc., DSP Designer™ software package.

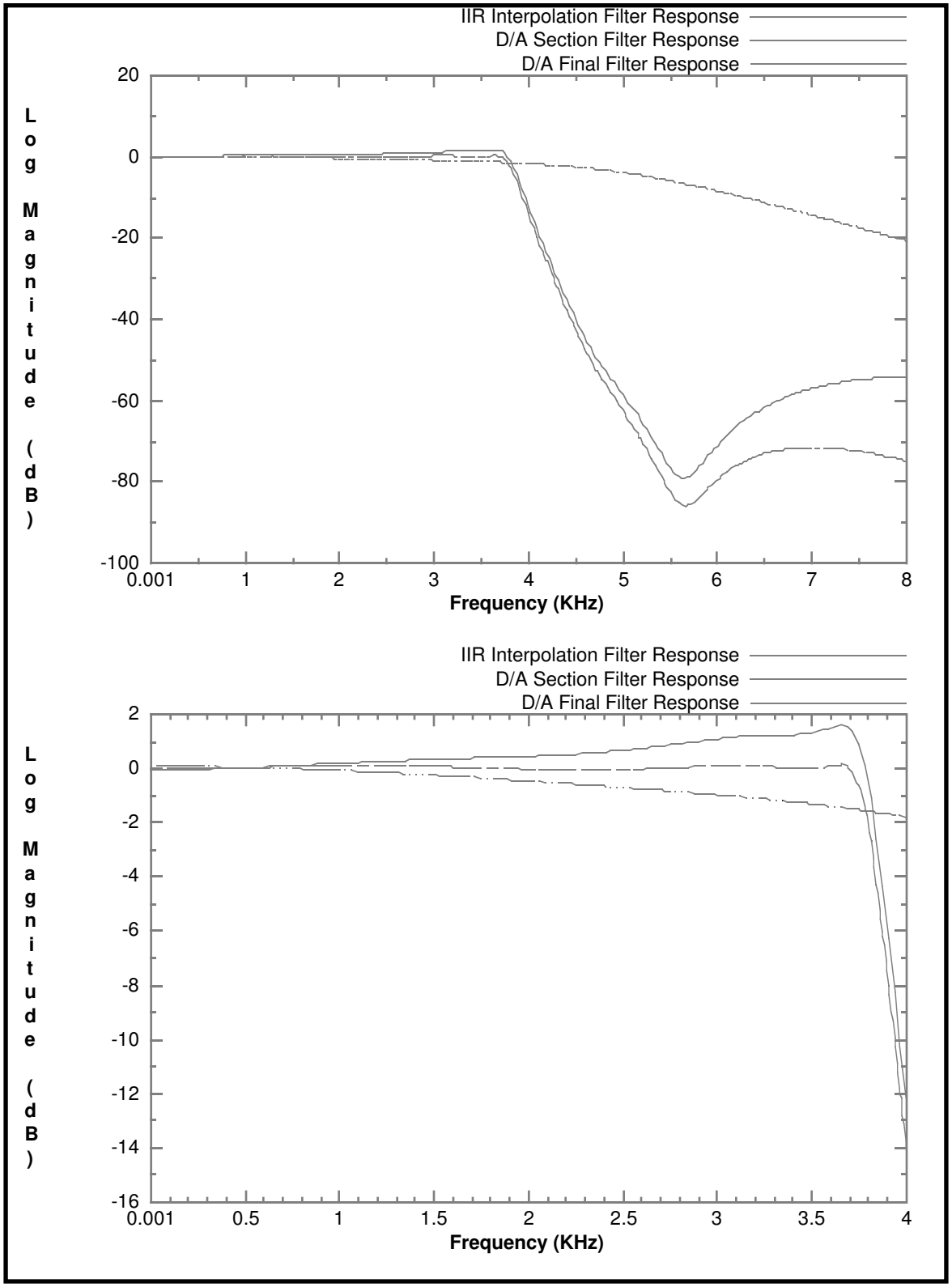


Figure 6-14 IIR Interpolation and D/A Section Log Magnitude Frequency Response for F=2.048 MHz and D=128

6.6 APPLICATION EXAMPLES

Examples one through four demonstrate applications in which different Codec decimation/interpolation ratios are used with different clocking configurations. The fifth and last example is a real-time I/O example using the on-chip codec and PLL. This example includes the DSP initialization as well as the filter code and is intended to be a framework for user applications.

6.6.1 Example 1

This example illustrates an application where the input clock provided on the EXTAL pin is 13 MHz and where the final sampling rate of the data converted is expected to be 8 KHz. The different clock synthesis and decimation/interpolation ratios for the ΣΔ A/D and D/A sections are shown in Figure 6-15.

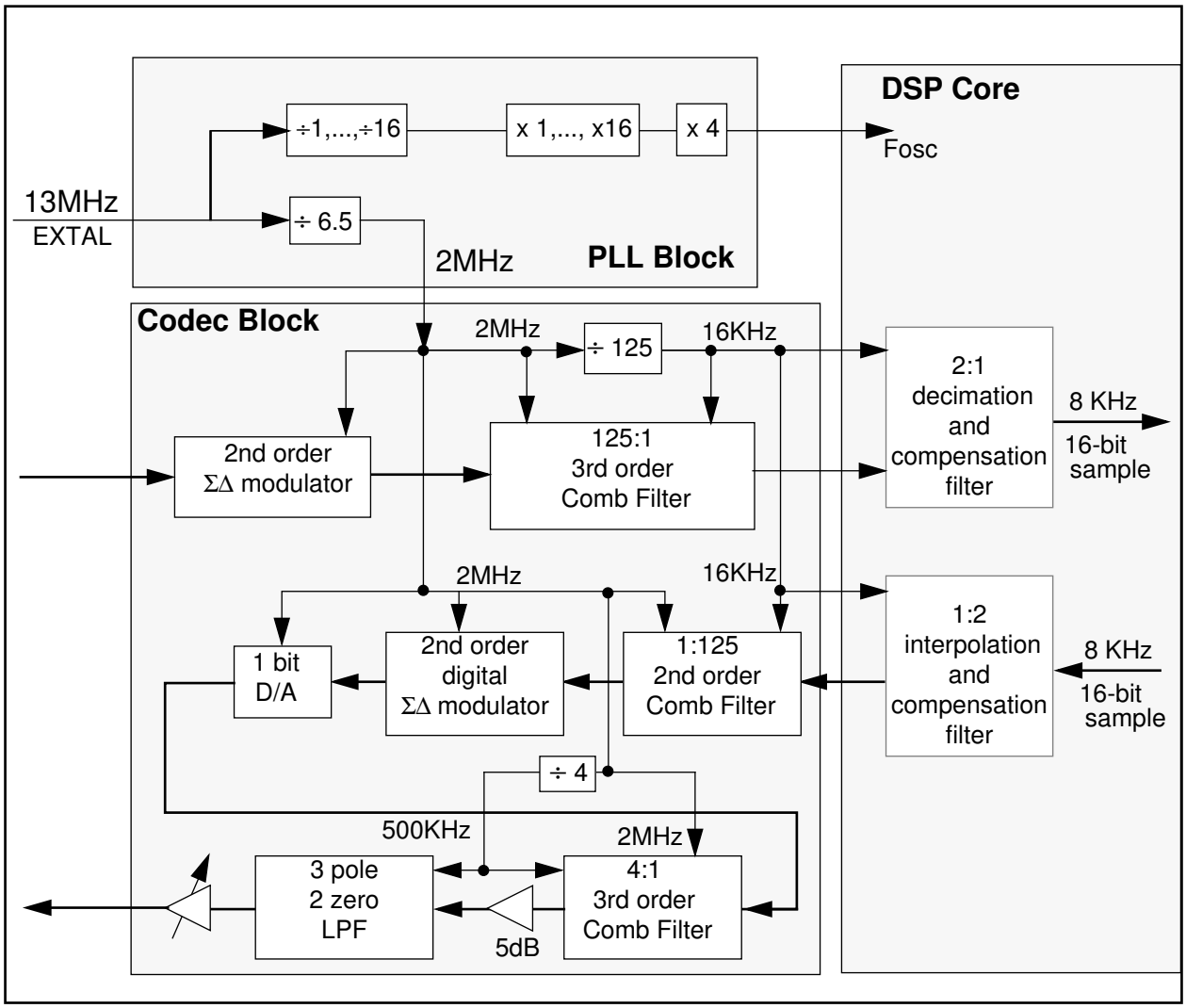


Figure 6-15 Example 1 functional block diagram

Table 6-11 describes the DC response of the codec section when the ratio 125 is selected.

Table 6-11 Codec DC Constant for 125 Decimation/interpolation Ratio

CRS1	CRS0	Decimation Ratio Rate	Output level and attenuation factor of the A/D for a maximum input voltage		Gain of the D/A Section (VC=0dB)	
0	0	125	-0.618 dB	0.9313	4.794 dB	1.7366

In order to compensate this DC response to a unity response, the gain of the A/D transmit decimation filter will have to be divided by 0.9313 and the gain of the receive D/A interpolation filter will have to be multiplied by 1/1.7366

The DSP decimation and interpolation filter may also need to fulfill some predefined frequency response constraints. Figure 6-16 shows the characteristics of the transmit and receive filters that we consider for this example.

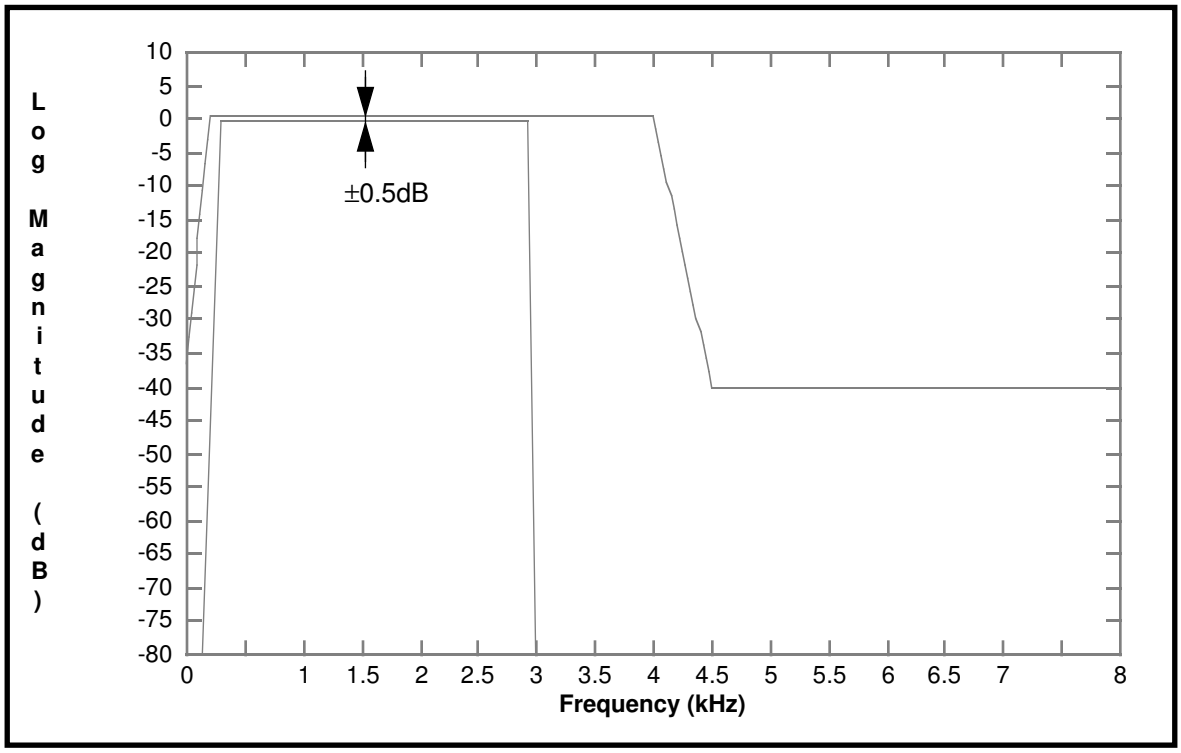


Figure 6-16 Example of Transmit and Receive Filter Performance Constraints

6.6.1.1 A/D Decimation DSP Filter

In addition to antialiasing, the decimation filter also needs to compensate the A/D comb filter response. For a decimation ratio D=125 and a master clock F=2 MHz, the frequency response of the A/D comb filter is:

$$F(f) = \frac{1}{(125)^3}\left(\frac{\sin\left(\frac{2\pi \times 125}{2 \times 2000} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 2000} \times f\right)}\right)^3$$

The frequency response of the decimation filter will have to be shaped by 1/F(f) in order to flatten the response of the A/D section. An example of such a filter is given in Figure 6-17.

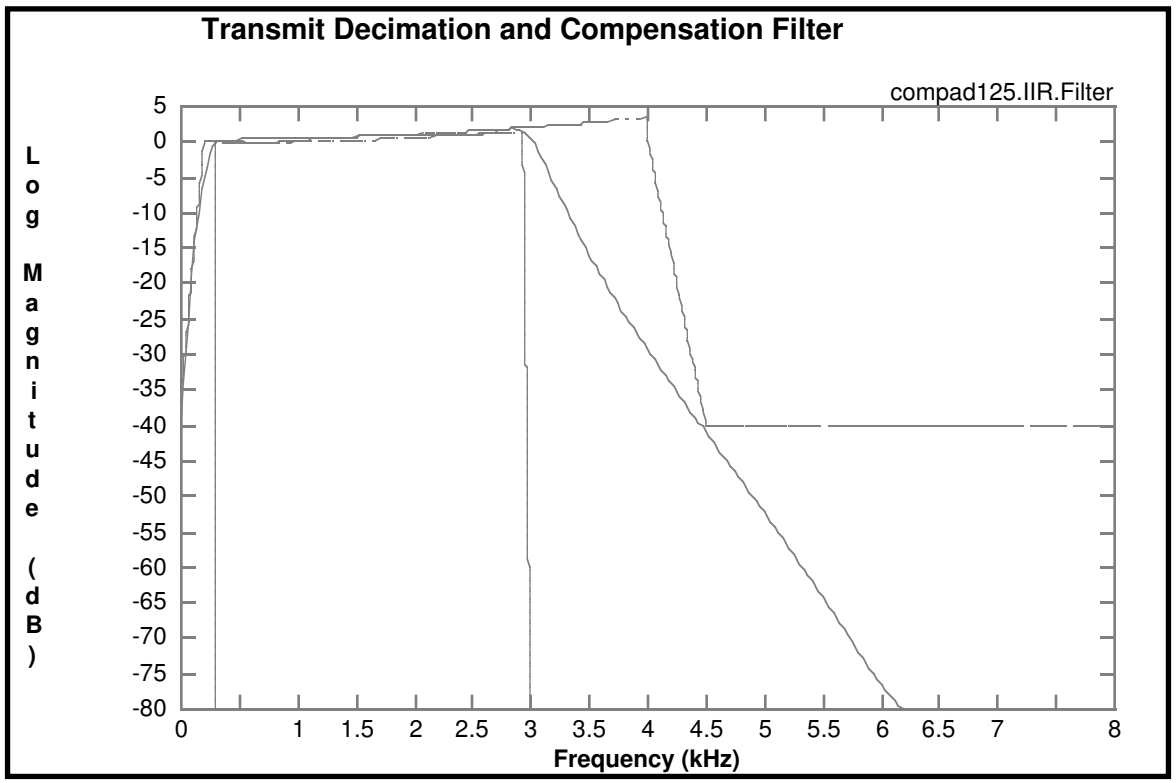


Figure 6-17 Example of a Transmit Antialiasing-decimation IIR Filter

The IIR filter shown in Figure 6-17 is implemented using 4 biquadratic sections. The gain and coefficients of the four sections are listed in Table 6-12.

If a unity gain is expected from the A/D section, the gain of that filter has to be multiplied by 1/0.9313 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-11).

Table 6-12 Coefficient of the Filter Shown in Figure 6-17

```
; Source filter file: “compad125.IIR.Filter”

_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     4               ; number of stages

                dc      $00e1           ; gain = 0.01371465707/2
                ; Biquad stage no. 1
                dc      $ca7f           ; -d2_1 = -0.8360288598/2
                dc      $2dfe           ; -d1_1 = 0.7186447751/2
                dc      $2d53           ; n2_1 = 0.7081856555/2
                dc      $509d           ; n1_1 = 1.259552697/2
                ; Biquad stage no. 2
                dc      $df76           ; -d2_2 = -0.5084528045/2
                dc      $37c5           ; -d1_2 = 0.8713790964/2
                dc      $dadc           ; n2_2 = -0.5803458289/2
                dc      $e66c           ; n1_2 = -0.3996296048/2
                ; Biquad stage no. 3
                dc      $e42a           ; -d2_3 = -0.434960982/2
                dc      $4b16           ; -d1_3 = 1.173228867/2
                dc      $dd3d           ; n2_3 = -0.5431805183/2
                dc      $e3c4           ; n1_3 = -0.4411614171/2
                ; Biquad stage no. 4
                dc      $c60b           ; -d2_4 = -0.9055949256/2
                dc      $7940           ; -d1_4 = 1.894549449/2
                dc      $f477           ; n2_4 = -0.1802356271/2
                dc      $285f           ; n1_4 = 0.6307957449/2
```

Figure 6-18 shows the overall response of the A/D section cascaded with the DSP IIR filter described above.

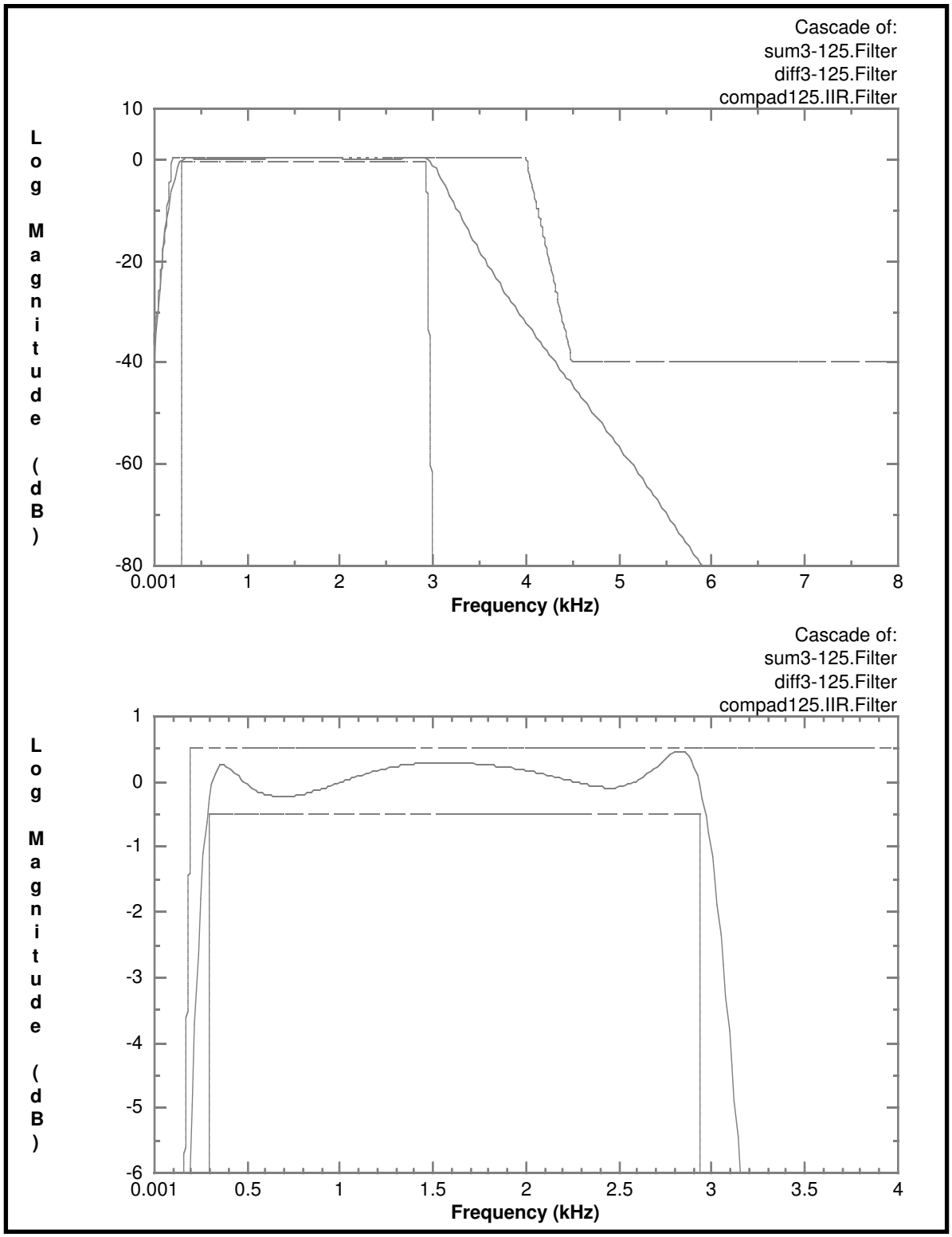


Figure 6-18 Overall Response of the A/D Section when Using the IIR Filter of Figure 6-17.

6.6.1.2 D/A Interpolation Filter

In addition to reconstruction, the D/A interpolation filter also needs to compensate the D/A section frequency response. For a decimation ratio D=125 and a master clock F=2 MHz, the frequency response of the D/A section is:

$$F(f) = \frac{1}{125 \times 128}\left(\frac{\sin\left(\frac{2\pi \times 125}{2 \times 2000} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 2000} \times f\right)}\right)^2$$

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.

An example of such a filter is given in Figure 6-19.

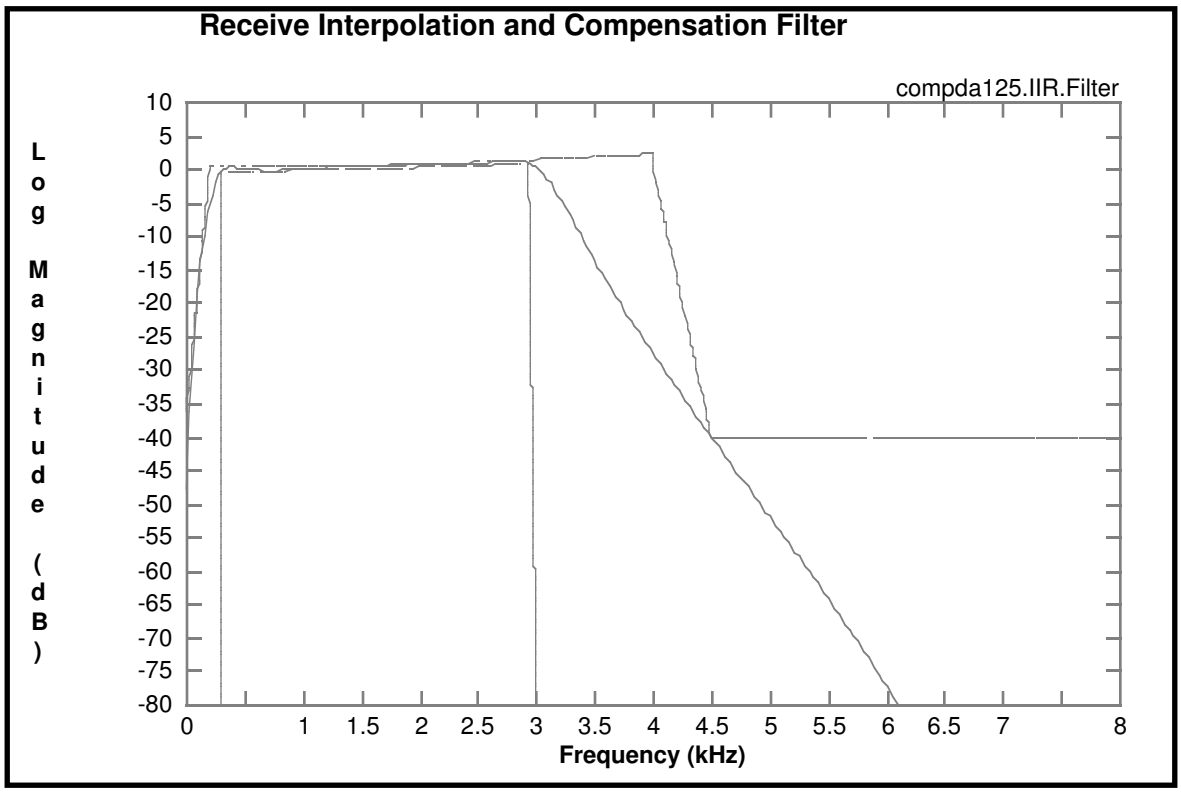


Figure 6-19 Example of a Receive Reconstruction-interpolation IIR Filter

The IIR filter shown in Figure 6-19 is implemented using 4 biquadratic sections. The gain and coefficients of the four section are listed in Table 6-13.

If a unity gain is expected from the D/A section, the gain of that filter has to be multiplied by 1/1.7366 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-11).

Table 6-13 Coefficient of the Filter Shown in Figure 6-19

```
; Source filter file: “compda125.IIR.Filter”
_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     4               ; number of stages

                dc      $00a6           ; gain = 0.01012690668/2

                ; Biquad stage no. 1
                dc      $cd33           ; -d2_1 = -0.7937344327/2
                dc      $2a8b           ; -d1_1 = 0.6647261771/2
                dc      $2fc9           ; n2_1 = 0.7466289714/2
                dc      $631a           ; n1_1 = 1.548444641/2
                ; Biquad stage no. 2
                dc      $e357           ; -d2_2 = -0.4478366909/2
                dc      $2c13           ; -d1_2 = 0.6886420446/2
                dc      $d70a           ; n2_2 = -0.6400153519/2
                dc      $ea14           ; n1_2 = -0.3425524844/2
                ; Biquad stage no. 3
                dc      $e6b6           ; -d2_3 = -0.395159897/2
                dc      $4431           ; -d1_3 = 1.065511376/2
                dc      $dd2e           ; n2_3 = -0.5440372612/2
                dc      $e35a           ; n1_3 = -0.4476206714/2
                ; Biquad stage no. 4
                dc      $c629           ; -d2_4 = -0.9037268127/2
                dc      $7923           ; -d1_4 = 1.892747933/2
                dc      $19a1           ; n2_4 = 0.4004225192/2
                dc      $5060           ; n1_4 = 1.255851238/2
```

Figure 6-20 shows the overall response of the DSP IIR filter cascaded with the D/A section.

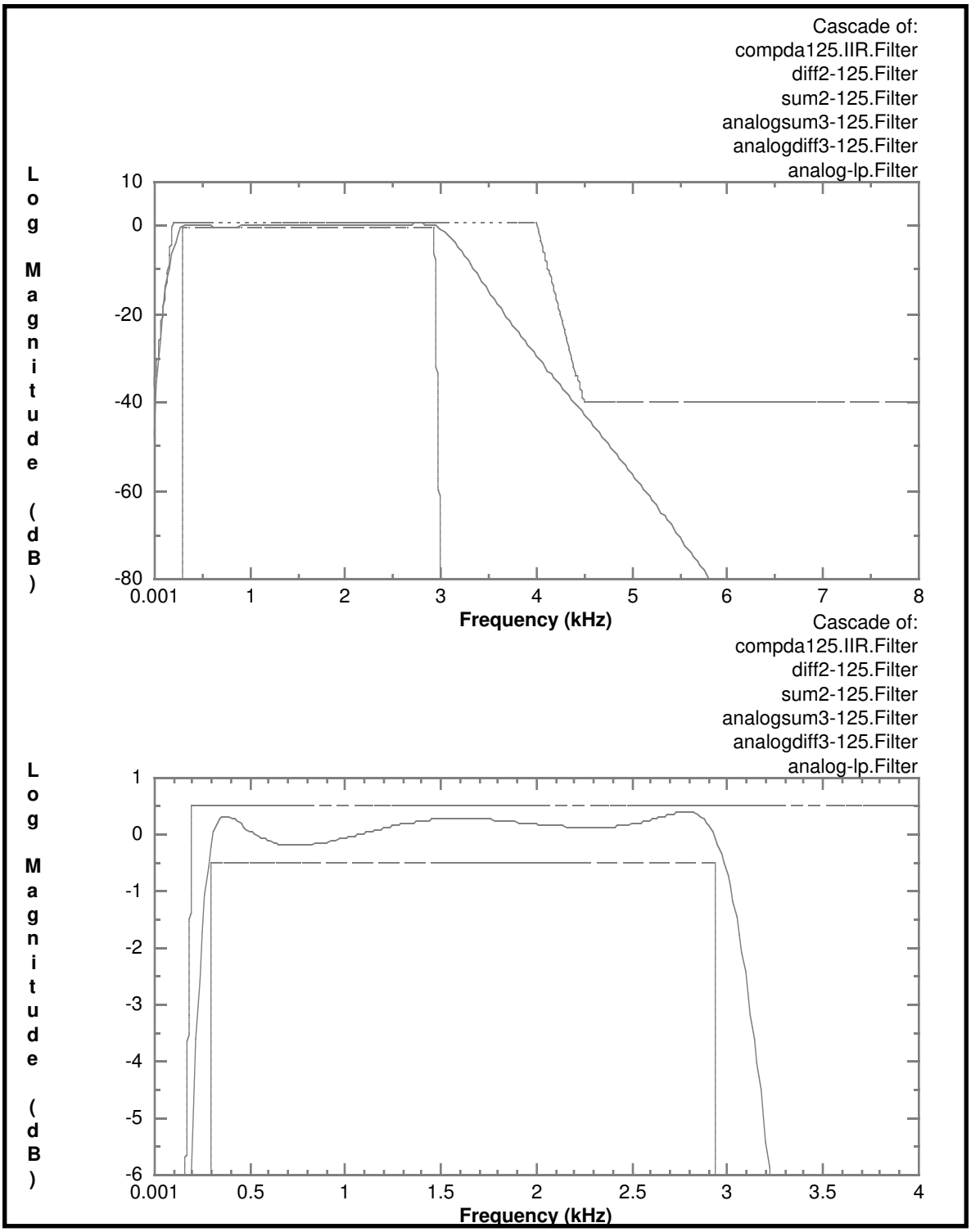


Figure 6-20 Overall Response of the D/A Section When Using the IIR Filter of Figure 6-19

6.6.2 Example 2

This example illustrates an application where the input clock provided on the EXTAL pin is 21 MHz and where the final sampling rate of the data converted is expected to be 8 KHz. The different clock synthesis and decimation/interpolation ratios for the ΣΔ A/D and D/A sections are shown in Figure 6-21.

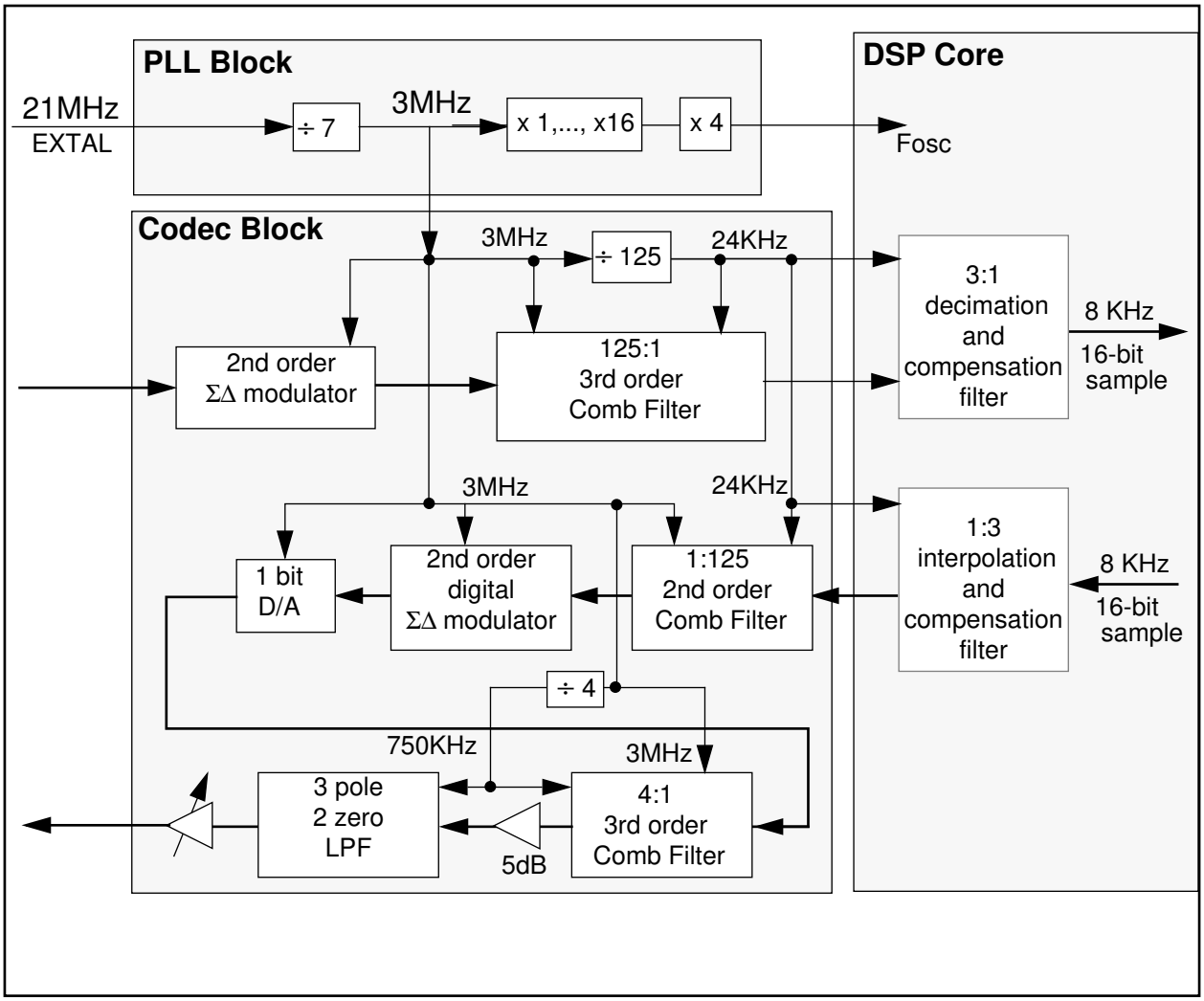


Figure 6-21 Example 2 functional block diagram

Table 6-14 describes the DC response of the codec section when the ratio 125 is selected.

Table 6-14 Codec DC Constant for 125 Decimation/interpolation Ratio

CRS1	CRS0	Decimation Ratio Rate	Output level and attenuation factor of the A/D for a maximum input voltage		Gain of the D/A Section (VC=0dB)	
0	0	125	-0.618 dB	0.9313	4.794 dB	1.7366

In order to compensate this DC response to a unity response, the gain of the A/D transmit decimation filter will have to be divided by 0.9313 and the gain of the receive D/A interpolation filter will have to be multiplied by 1/1.7366.

The DSP decimation and interpolation filter may also need to fulfill some predefined frequency response constraints. Figure 6-22 shows the characteristics of the transmit and receive filters that we consider for this example.

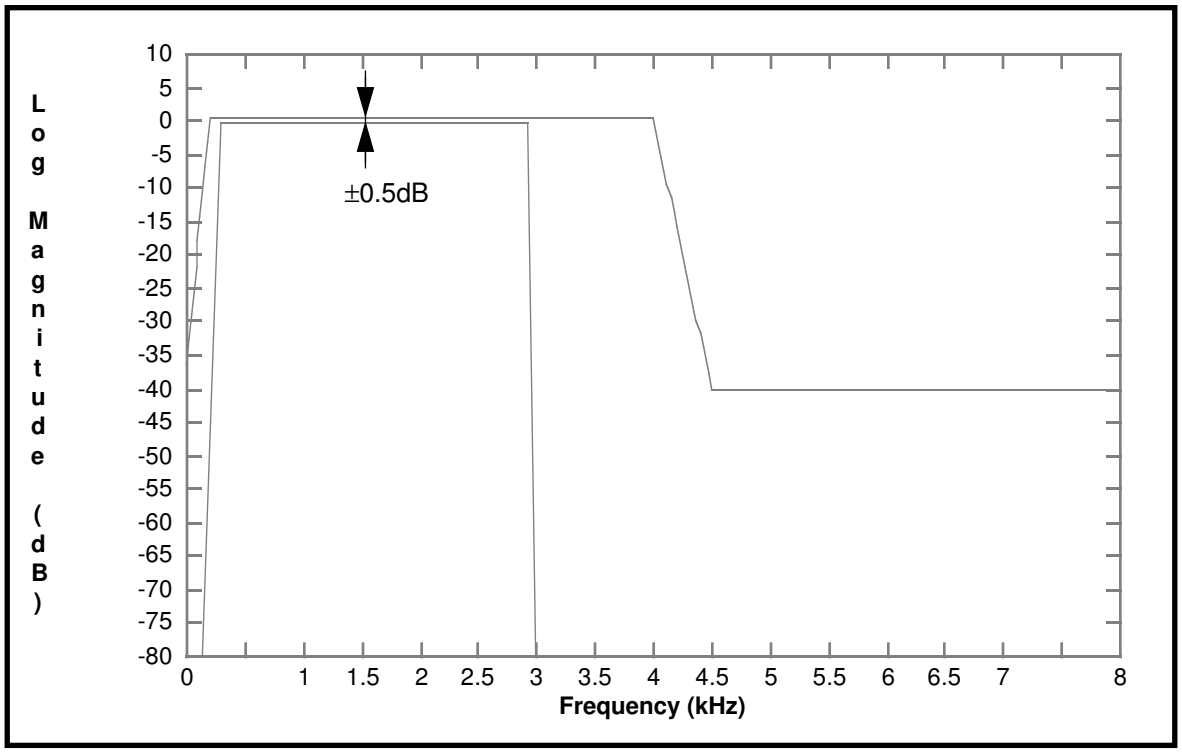


Figure 6-22 Example of Transmit and Receive Filter Performance Constraints

6.6.2.1 A/D Decimation DSP Filter

In addition to antialiasing, the decimation filter also needs to compensate the A/D comb filter response. For a decimation ratio D=125 and a master clock F=3 MHz, the frequency response of the A/D comb filter is:

$$F(f) = \frac{1}{(125)^3}\left(\frac{\sin\left(\frac{2\pi \times 125}{2 \times 3000} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 3000} \times f\right)}\right)^3$$

The frequency response of the decimation filter will have to be shaped by 1/F(f) in order to flatten the response of the A/D section. An example of such a filter is given in Figure 6-23.

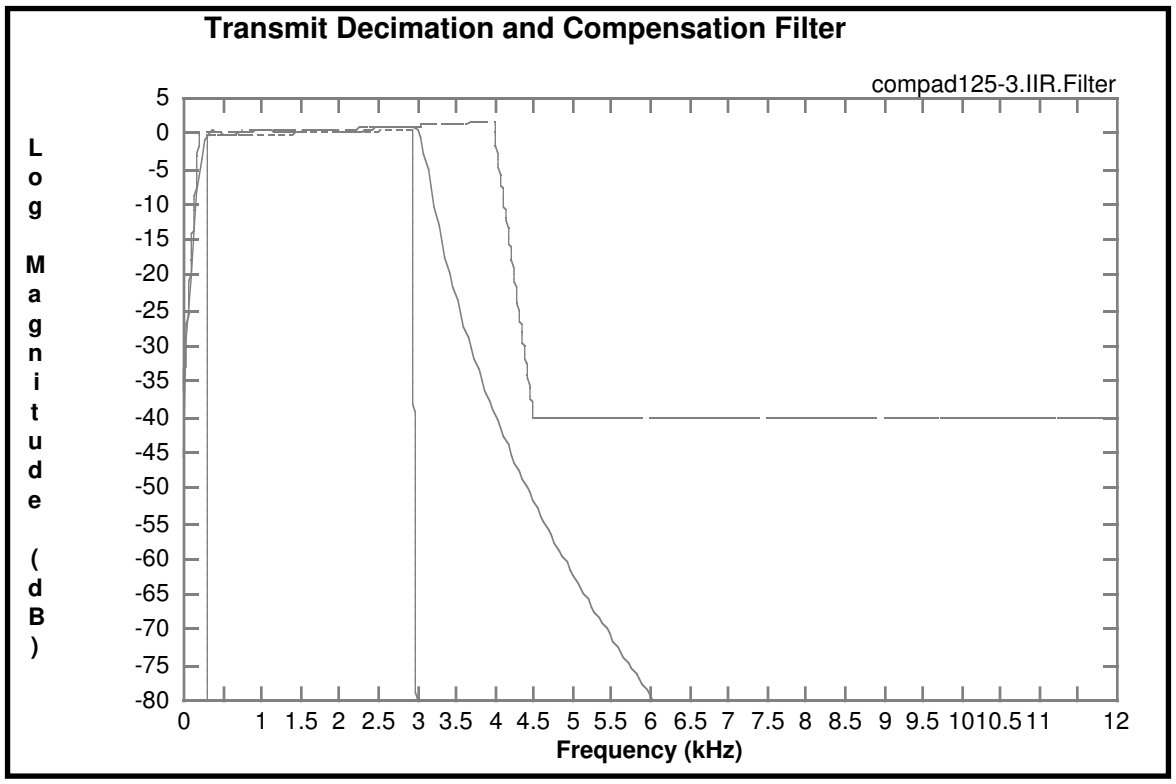


Figure 6-23 Example of a Transmit Antialiasing-decimation IIR Filter

The IIR filter shown in Figure 6-23 is implemented using 5 biquadratic sections. The gain and coefficients of the four section are listed in Table 6-15.

If a unity gain is expected from the A/D section, the gain of that filter has to be multiplied by 1/0.9313 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-14).

Table 6-15 Coefficient of the Filter Shown in Figure 6-23

```
; Source filter file: “compad125-3.IIR.Filter”

_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     5               ; number of biquadratic stages

                dc      $0004           ; g = 0.0002623691665/2

                ; Biquad stage no. 1
                dc      $d100           ; -d2_1 = -0.7344004701/2
                dc      $6173           ; -d1_1 = 1.522624504/2
                dc      $00ce           ; n2_1 = 0.01255399151/2
                dc      $2259           ; n1_1 = 0.5367057372/2
                ; Biquad stage no. 2
                dc      $cb98           ; -d2_2 = -0.8188274379/2
                dc      $5973           ; -d1_2 = 1.39766389/2
                dc      $fe9b           ; n2_2 = -0.02180931293/2
                dc      $1fd1           ; n1_2 = 0.4971358354/2
                ; Biquad stage no. 3
                dc      $d0af           ; -d2_3 = -0.7393345338/2
                dc      $6b37           ; -d1_3 = 1.675226724/2
                dc      $f992           ; n2_3 = -0.1004858072/2
                dc      $1c79           ; n1_3 = 0.4449126054/2
                ; Biquad stage no. 4
                dc      $c4a7           ; -d2_4 = -0.9273113361/2
                dc      $5669           ; -d1_4 = 1.350141276/2
                dc      $d61c           ; n2_4 = -0.6545527468/2
                dc      $ea20           ; n1_4 = -0.34182236/2
                ; Biquad stage no. 5
                dc      $c391           ; -d2_5 = -0.9442726739/2
                dc      $7c12           ; -d1_5 = 1.938610906/2
                dc      $dd40           ; n2_5 = -0.5429834643/2
                dc      $e4dd           ; n1_5 = -0.4240192654/2
```

Figure 6-24 shows the overall response of the A/D section cascaded with the DSP IIR filter described above.

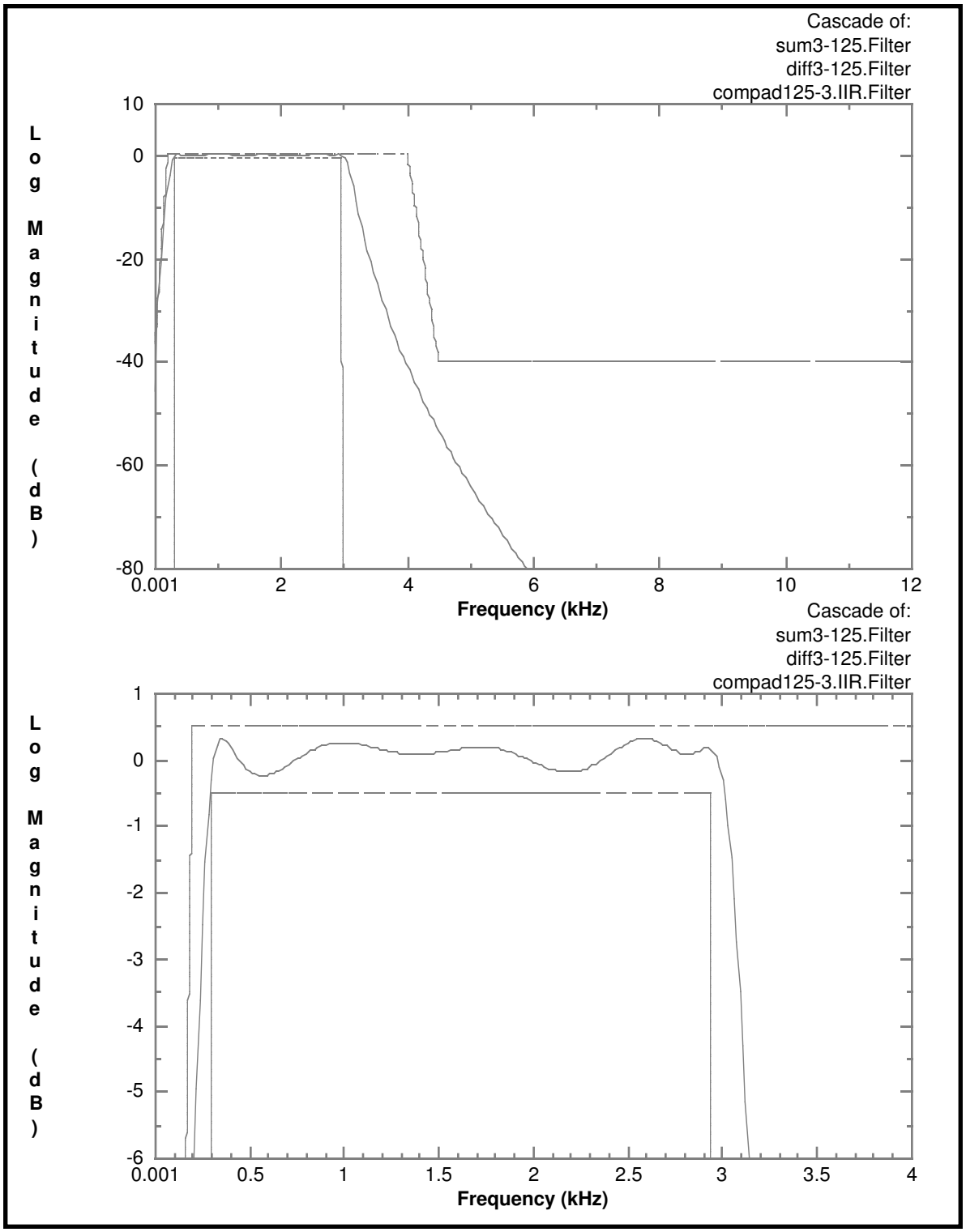


Figure 6-24 Overall Response of the A/D Section when Using the IIR Filter of Figure 6-23.

6.6.2.2 D/A Interpolation Filter

In addition to reconstruction, the D/A interpolation filter also needs to compensate the D/A section frequency response. For a decimation ratio D=125 and a master clock F=2 MHz, the frequency response of the D/A section is:

$$F(f) = \frac{1}{125 \times 128}\left(\frac{\sin\left(\frac{2\pi \times 125}{2 \times 2000} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 2000} \times f\right)}\right)^2$$

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.

An example of such a filter is given in Figure 6-25.

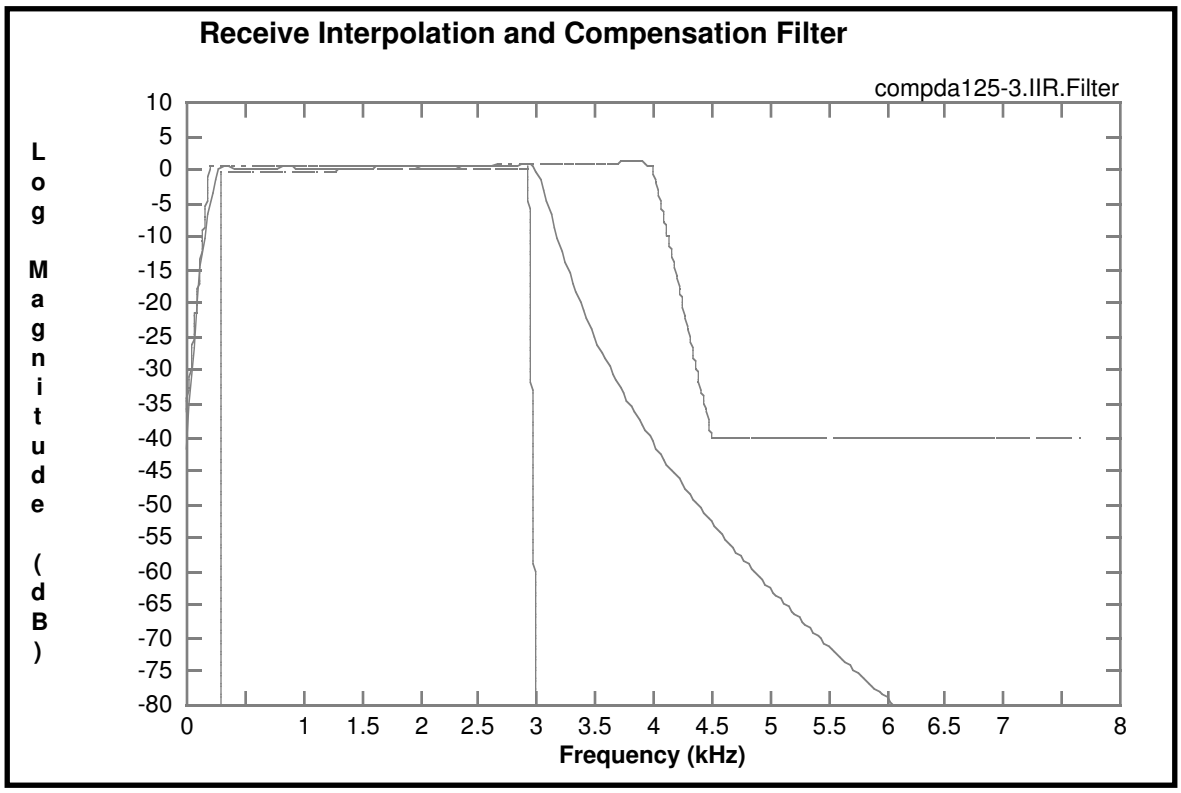


Figure 6-25 Example of a Receive Reconstruction-interpolation IIR Filter

The IIR filter shown in Figure 6-25 is implemented using 5 biquadratic sections. The gain and coefficients of the four section are listed in Table 6-16.

If a unity gain is expected from the D/A section, the gain of that filter has to be multiplied by 1/1.7366 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-14).

Table 6-16 Coefficient of the Filter Shown in Figure 6-25

```
; Source filter file: "compda125-3.IIR.Filter"
_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     5               ; number of biquadratic stages

                dc      $0004           ; g = 0.0002663555811/2

                ; Biquad stage no. 1
                dc      $cfea           ; -d2_1 = -0.7513359622/2
                dc      $62ed           ; -d1_1 = 1.545692138/2
                dc      $fea9           ; n2_1 = -0.02093532311/2
                dc      $1cfb           ; n1_1 = 0.4528080448/2
                ; Biquad stage no. 2
                dc      $cbb2           ; -d2_2 = -0.8172580463/2
                dc      $5a69           ; -d1_2 = 1.412661966/2
                dc      $fc5b           ; n2_2 = -0.05697205133/2
                dc      $1a08           ; n1_2 = 0.4067669553/2
                ; Biquad stage no. 3
                dc      $cfd3           ; -d2_3 = -0.7527639534/2
                dc      $6ca5           ; -d1_3 = 1.697593082/2
                dc      $f760           ; n2_3 = -0.1347748217/2
                dc      $1614           ; n1_3 = 0.3449803812/2
                ; Biquad stage no. 4
                dc      $c43d           ; -d2_4 = -0.9337611394/2
                dc      $5797           ; -d1_4 = 1.368618028/2
                dc      $dd44           ; n2_4 = -0.5427221805/2
                dc      $e362           ; n1_4 = -0.4471418369/2
                ; Biquad stage no. 5
                dc      $c363           ; -d2_5 = -0.9470992435/2
                dc      $7c4e           ; -d1_5 = 1.942249147/2
                dc      $ce92           ; n2_5 = -0.7723597585/2
                dc      $f27d           ; n1_5 = -0.211141353/2
```

Figure 6-26 shows the overall response of the DSP IIR filter cascaded with the D/A section.

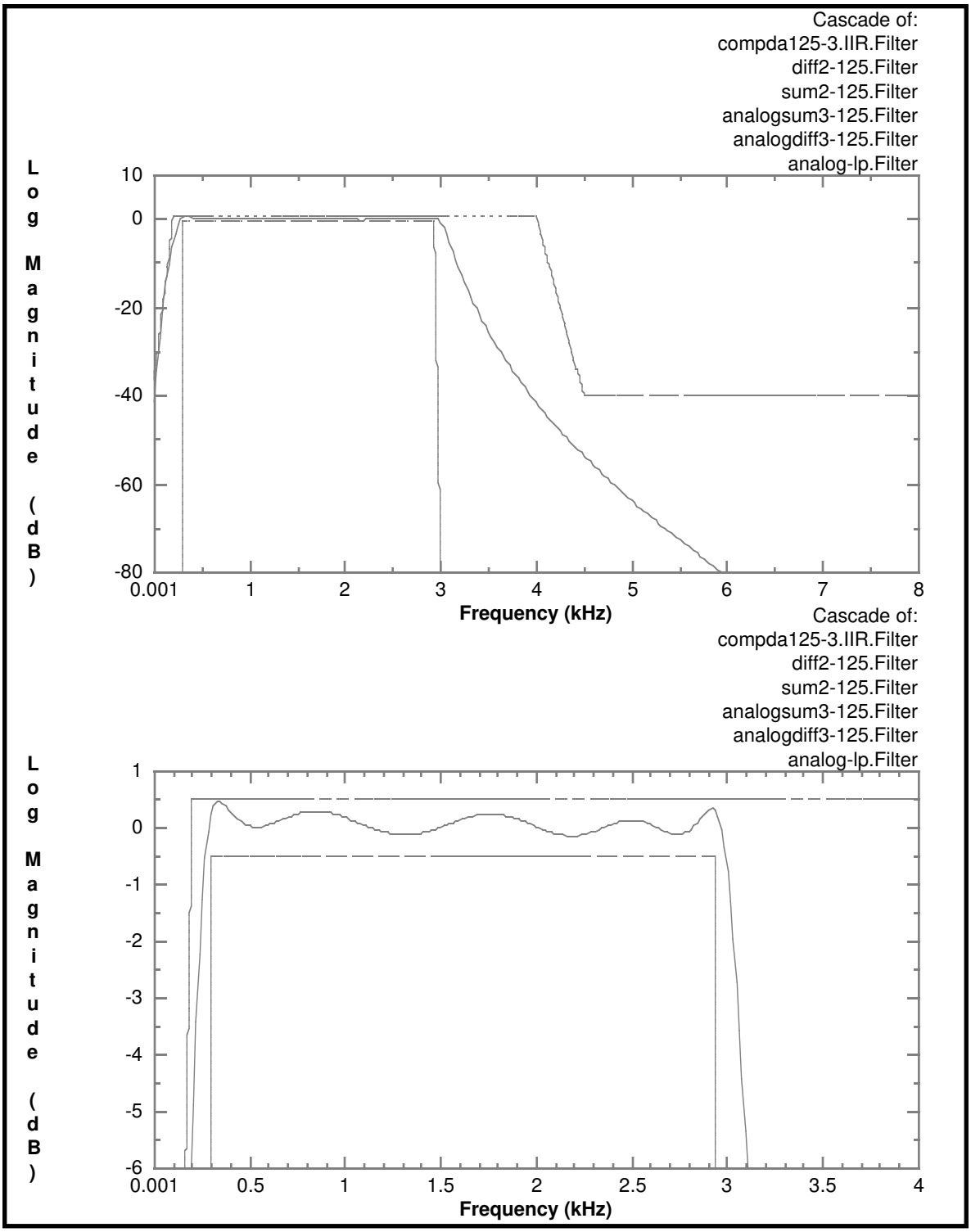


Figure 6-26 Overall Response of the D/A Section When Using the IIR Filter of Figure 6-25

6.6.3 Example 3

This example illustrates an application where the input clock provided on the EXTAL pin is 8.4 MHz and where the final sampling rate of the data converted is expected to be 8 KHz. The different clock synthesis and decimation/interpolation ratios for the ΣΔ A/D and D/A sections are shown in Figure 6-27.

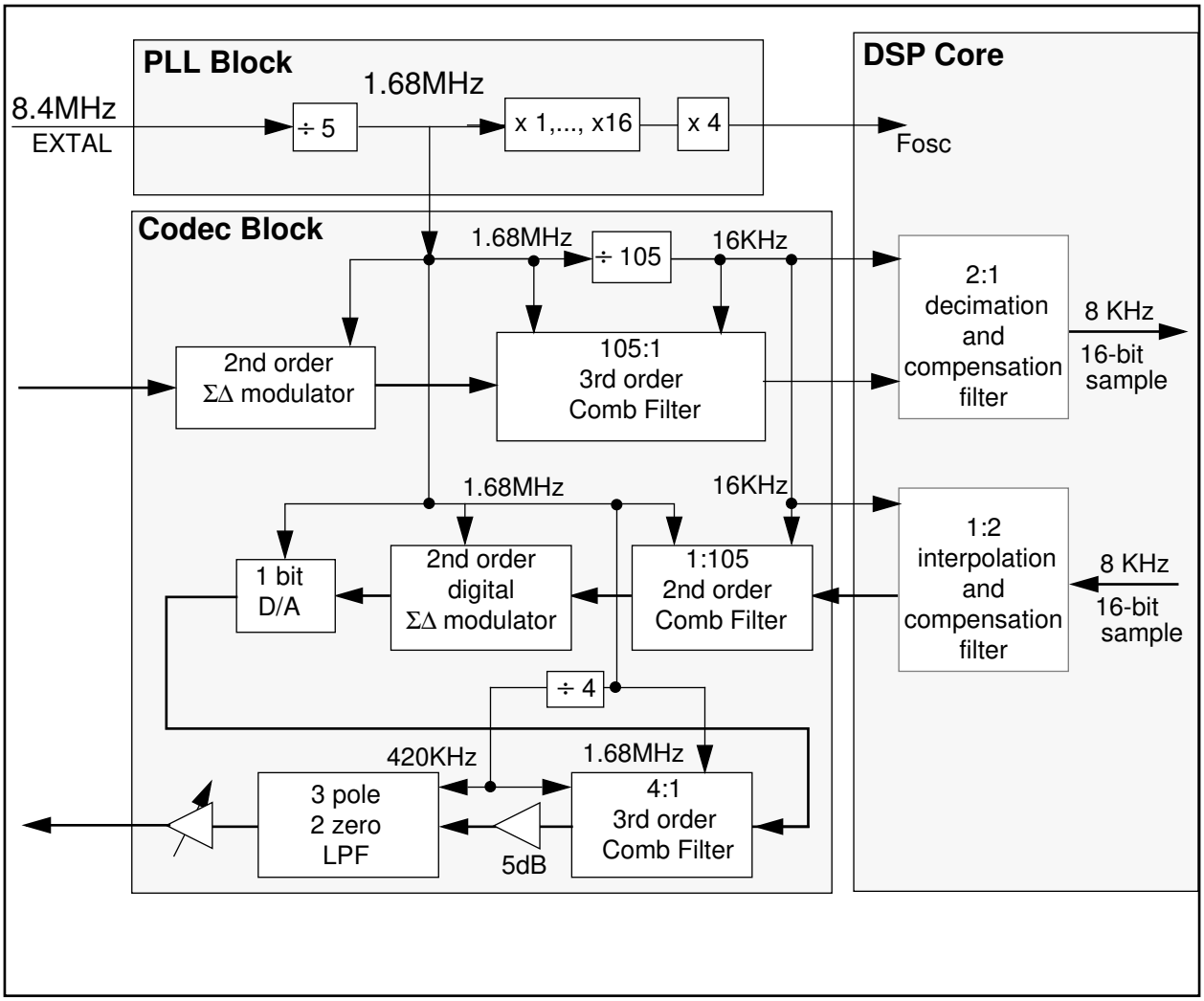


Figure 6-27 Example 3 functional block diagram

Table 6-17 describes the DC response of the codec section when the ratio 105 is selected.

Table 6-17 Codec DC Constant for 105 Decimation/interpolation Ratio

CRS1	CRS0	Decimation Ratio Rate	Output level and attenuation factor of the A/D for a maximum input voltage		Gain of the D/A Section (VC=0dB)	
1	0	105	0.859 dB	1.104	3.28 dB	1.4588

In order to compensate this DC response to a unity response, the gain of the A/D transmit decimation filter will have to be divided by 1.104 and the gain of the receive D/A interpolation filter will have to be multiplied by 1/1.4588

The DSP decimation and interpolation filter may also need to fulfill some predefined frequency response constraints.Figure 6-28 shows the characteristics of the transmit and receive filters that we consider for this example.

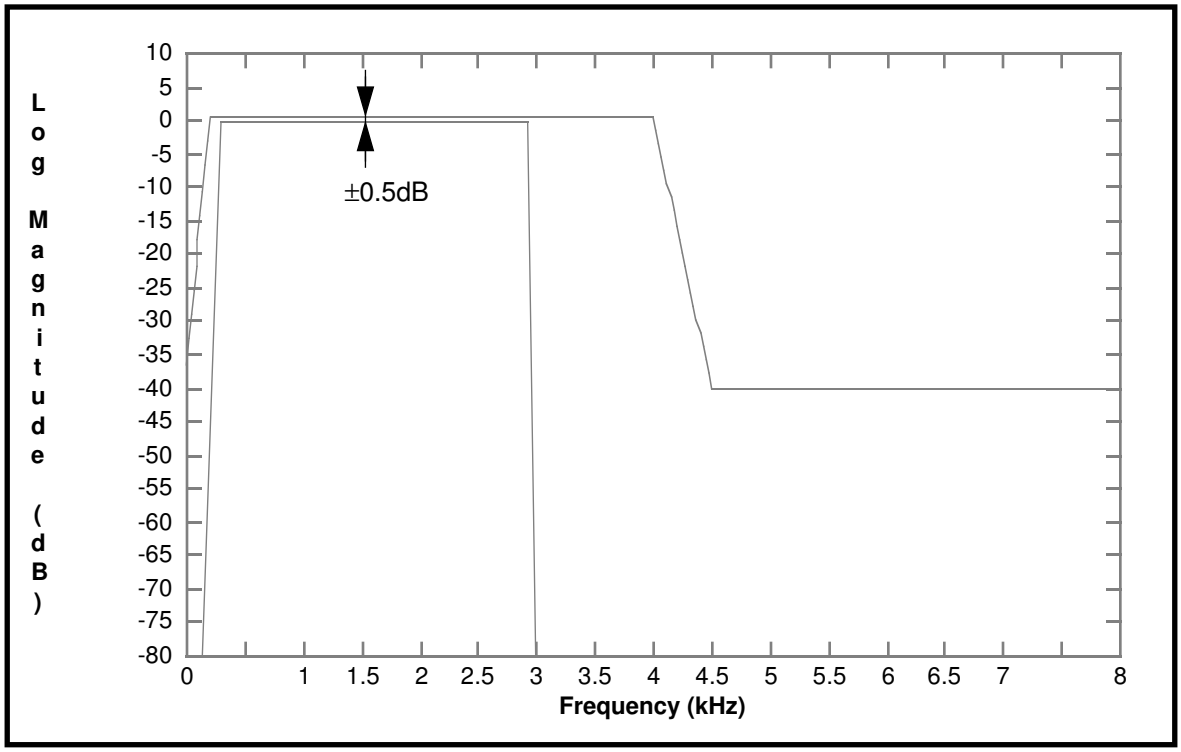


Figure 6-28 Example of Transmit and Receive Filter Performance Constraints

6.6.3.1 A/D Decimation DSP Filter

In addition to antialiasing, the decimation filter also needs to compensate the A/D comb filter response. For a decimation ratio D=105 and a master clock F=1.68 MHz, the frequency response of the A/D comb filter is:

$$F(f) = \frac{2}{(105)^3}\left(\frac{\sin\left(\frac{2\pi \times 105}{2 \times 1680} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 1680} \times f\right)}\right)^3$$

The frequency response of the decimation filter will have to be shaped by 1/F(f) in order to flatten the response of the A/D section. An example of such a filter is given in Figure 6-29.

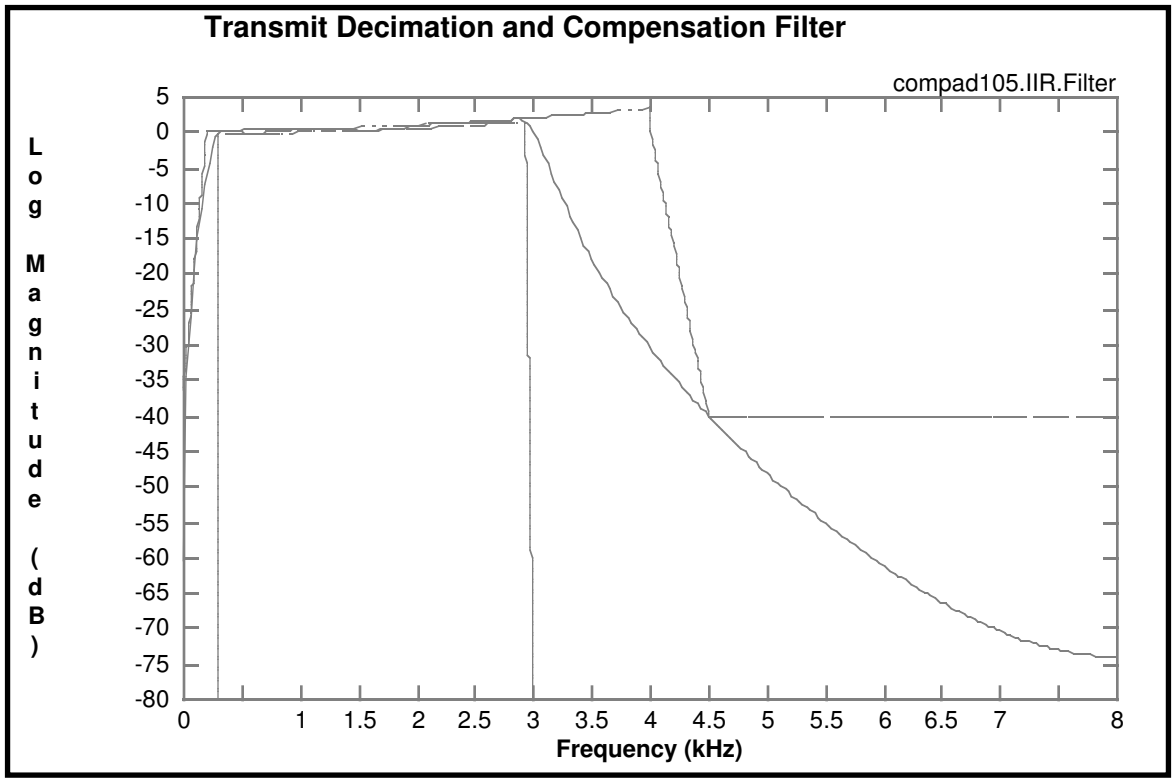


Figure 6-29 Example of GSM Transmit Antialiasing-decimation IIR Filter

The IIR filter shown in Figure 6-29 is implemented using 4 biquadratic sections. The gain and coefficients of the four section are listed in Table 6-18.

If a unity gain is expected from the A/D section, the gain of that filter has to be multiplied by 1/1.104 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-17).

Table 6-18 Coefficient of the Filter Shown in Figure 6-29

```
; Source filter file: “compad105.IIR.Filter”

_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     4               ; number of biquadratic stages

                dc      $01ac           ; g = 0.0261528436/2

                ; Biquad stage no. 1
                dc      $d625           ; -d2_1 = -0.6539936028/2
                dc      $4001           ; -d1_1 = 1.000075495/2
                dc      $00b7           ; n2_1 = 0.01115530244/2
                dc      $197a           ; n1_1 = 0.3980984016/2
                ; Biquad stage no. 2
                dc      $db34           ; -d2_2 = -0.5749677665/2
                dc      $58d6           ; -d1_2 = 1.388032237/2
                dc      $0069           ; n2_2 = 0.00643072253/2
                dc      $183a           ; n1_2 = 0.3785165879/2
                ; Biquad stage no. 3
                dc      $c8c9           ; -d2_3 = -0.8627144322/2
                dc      $2f54           ; -d1_3 = 0.7394786413/2
                dc      $e8e8           ; n2_3 = -0.3608584886/2
                dc      $d922           ; n1_3 = -0.6072824655/2
                ; Biquad stage no. 4
                dc      $c5a7           ; -d2_4 = -0.9116724841/2
                dc      $7996           ; -d1_4 = 1.89978731/2
                dc      $e7fd           ; n2_4 = -0.3751780353/2
                dc      $d812           ; n1_4 = -0.6238880148/2
```

Figure 6-30 shows the overall response of the A/D section cascaded with the DSP IIR filter described above.

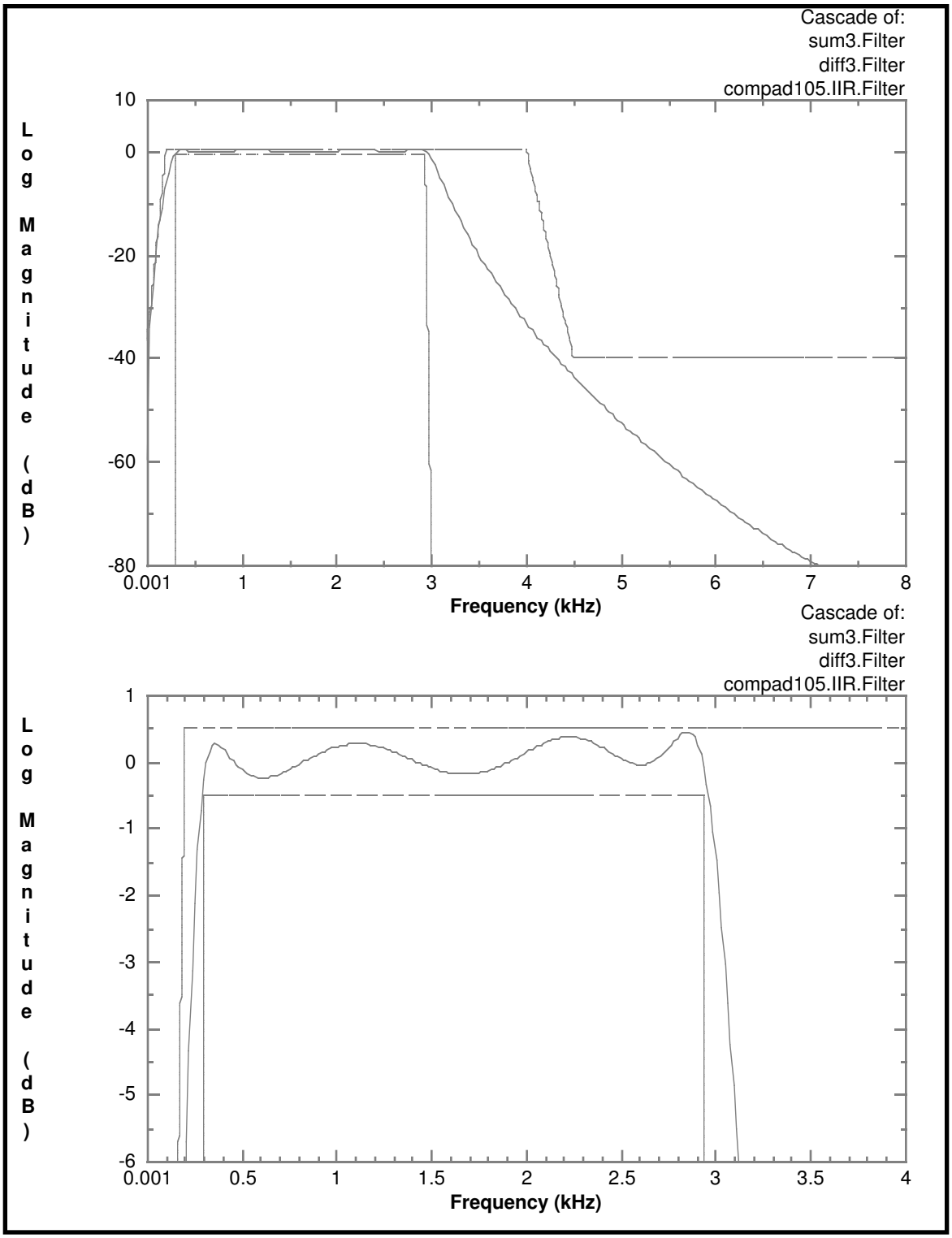


Figure 6-30 Overall Response of the A/D Section when Using the IIR Filter of Figure 6-29.

6.6.3.2 D/A Interpolation Filter

In addition to reconstruction, the D/A interpolation filter also needs to compensate the D/A section frequency response. For a decimation ratio D=105 and a master clock F=1.68 MHz, the frequency response of the D/A section is:

$$F(f) = \frac{1}{105 \times 128}\left(\frac{\sin\left(\frac{2\pi \times 105}{2 \times 1680} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 1680} \times f\right)}\right)^2$$

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.

An example of such a filter is given in Figure 6-31.

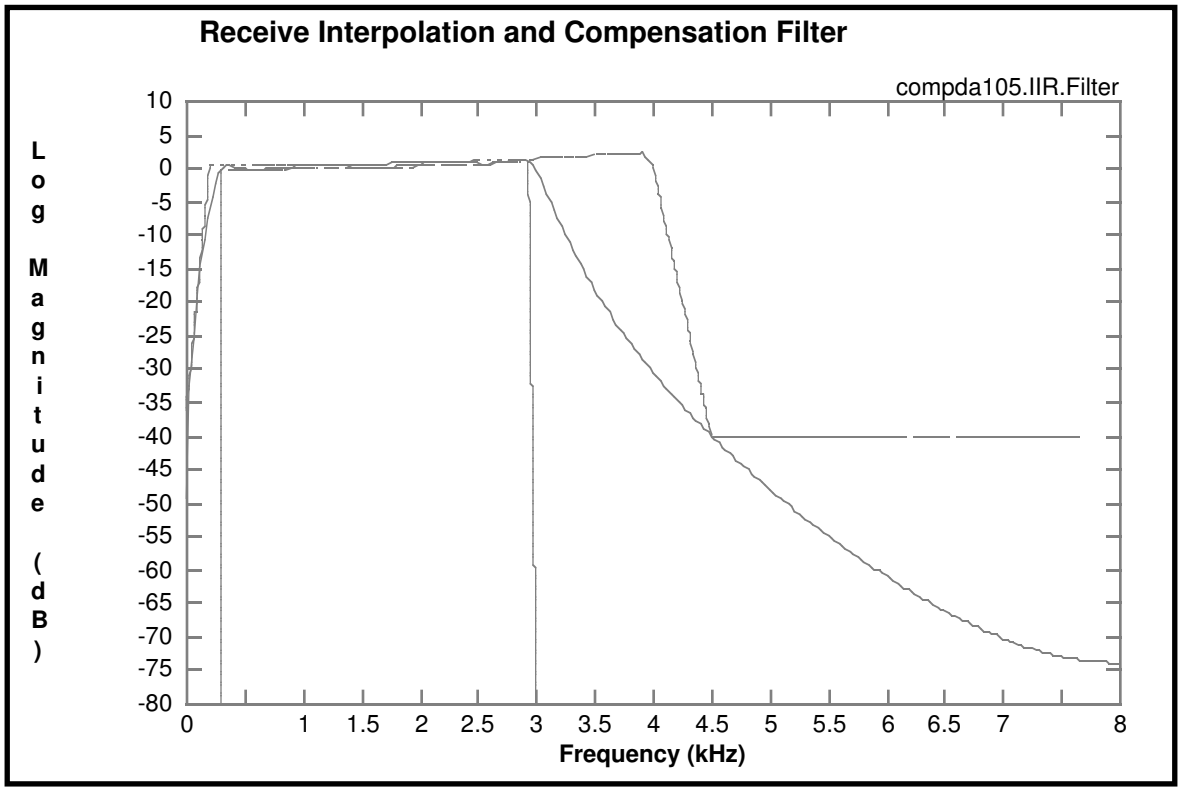


Figure 6-31 Example of a Receive Reconstruction-interpolation IIR Filter

The IIR filter shown in Figure 6-31 is implemented using 4 biquadratic sections. The gain and coefficients of the four section are listed in Table 6-19.

If a unity gain is expected from the D/A section, the gain of that filter has to be multiplied by 1/1.4588 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-17).

Table 6-19 Coefficient of the Filter Shown in Figure 6-31

```
; Source filter file: "compda105.IIR.Filter"
_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     4               ; number of biquadratic stages

                dc      $01aa           ; g = 0.02600974423/2

                ; Biquad stage no. 1
                dc      $d643           ; -d2_1 = -0.6521580312/2
                dc      $40b9           ; -d1_1 = 1.011306724/2
                dc      $ffa6           ; n2_1 = -0.005487472908/2
                dc      $1716           ; n1_1 = 0.3606981345/2
                ; Biquad stage no. 2
                dc      $dab0           ; -d2_2 = -0.5829883186/2
                dc      $59b0           ; -d1_2 = 1.401342029/2
                dc      $ff66           ; n2_2 = -0.009372766805/2
                dc      $15eb           ; n1_2 = 0.3424793475/2
                ; Biquad stage no. 3
                dc      $c8ec           ; -d2_3 = -0.8605882387/2
                dc      $2f65           ; -d1_3 = 0.7405380828/2
                dc      $e878           ; n2_3 = -0.3676882987/2
                dc      $da3c           ; n1_3 = -0.5901159969/2
                ; Biquad stage no. 4
                dc      $c5aa           ; -d2_4 = -0.9114780943/2
                dc      $7990           ; -d1_4 = 1.899401635/2
                dc      $e724           ; n2_4 = -0.3884263941/2
                dc      $d908           ; n1_4 = -0.6089069362/2
```

Figure 6-32 shows the overall response of the DSP IIR filter cascaded with the D/A section.

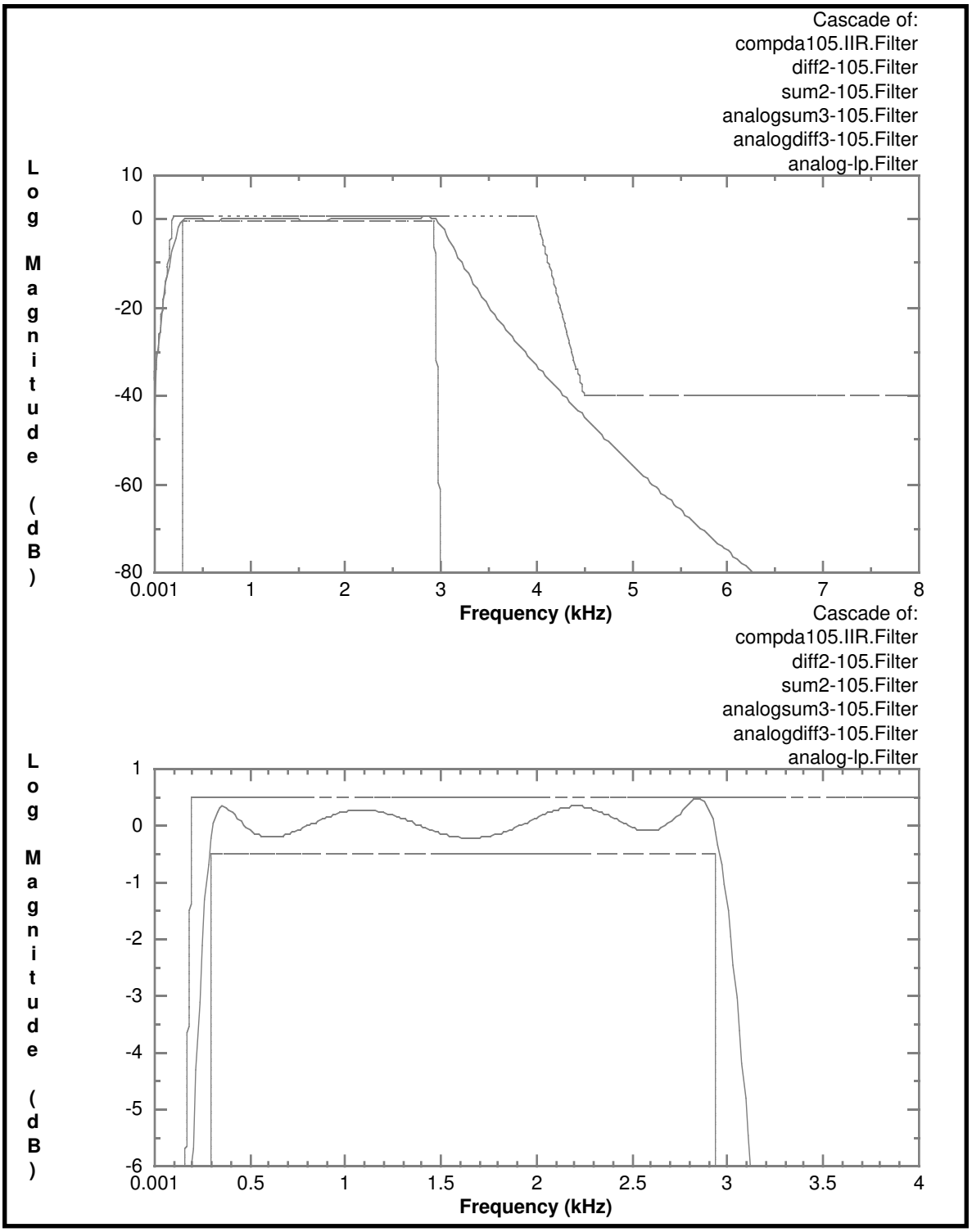


Figure 6-32 Overall Response of the D/A Section When Using the IIR Filter of Figure 6-31

6.6.4 Example 4

This example illustrates an application where the input clock provided on the EXTAL pin is 9.72 MHz and where the final sampling rate of the data converted is expected to be 8 KHz. The different clock synthesis and decimation/interpolation ratios for the ΣΔ A/D and D/A sections are shown in Figure 6-33.

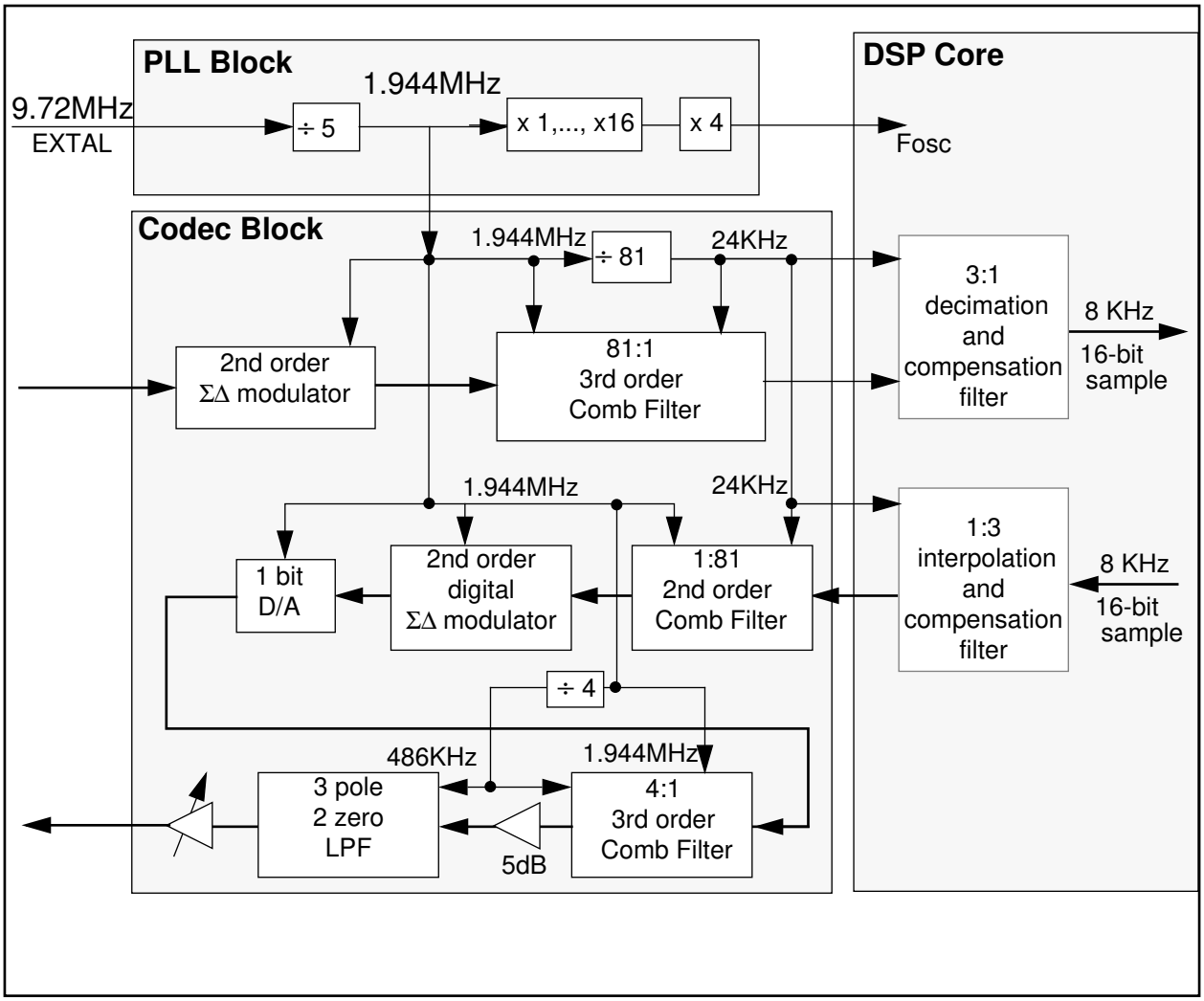


Figure 6-33 Example 4 functional block diagram

Table 6-20 describes the DC response of the codec section when the ratio 81 is selected.

Table 6-20 Codec DC Constant for 81 Decimation/interpolation Ratio

CRS1	CRS0	Decimation Ratio Rate	Output level and attenuation factor of the A/D for a maximum input voltage		Gain of the D/A Section (VC=0dB)	
1	1	81	0.118 dB	1.0137	1.026 dB	1.1254

In order to compensate this DC response to a unity response, the gain of the A/D transmit decimation filter will have to be divided by 1.0137 and the gain of the receive D/A interpolation filter will have to be multiplied by 1/1.1254

The DSP decimation and interpolation filter may also need to fulfill some predefined frequency response constraints.Figure 6-34 shows the characteristics of the transmit and receive filters that we consider for this example.

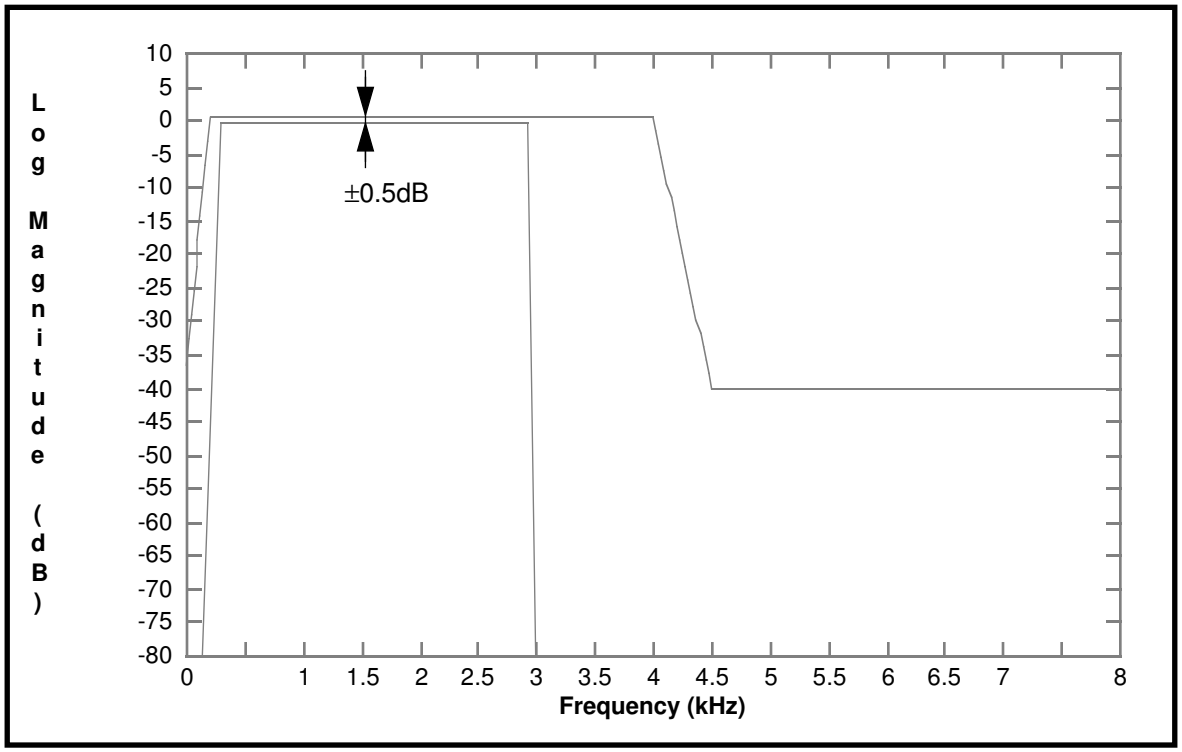


Figure 6-34 Example of Transmit and Receive Filter Performance Constraints

6.6.4.1 A/D Decimation DSP Filter

In addition to antialiasing, the decimation filter also needs to compensate the A/D comb filter response. For a decimation ratio D=81 and a master clock F=1.944 MHz, the frequency response of the A/D comb filter is:

$$F(f) = \frac{4}{(81)^3}\left(\frac{\sin\left(\frac{2\pi \times 81}{2 \times 1944} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 1944} \times f\right)}\right)^3$$

The frequency response of the decimation filter will have to be shaped by 1/F(f) in order to flatten the response of the A/D section. An example of such a filter is given in Figure 6-35.

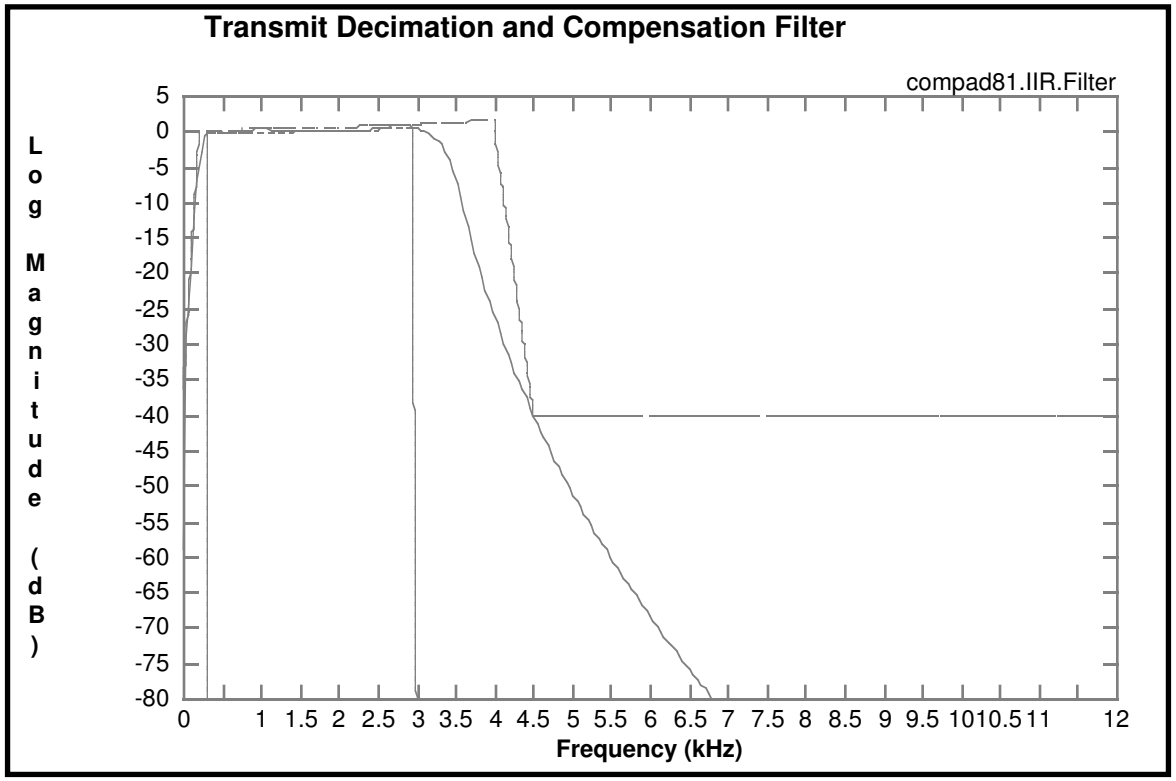


Figure 6-35 Example of a Transmit Antialiasing-decimation IIR Filter

The IIR filter shown in Figure 6-35 is implemented using 5 biquadratic sections. The gain and coefficients of the four section are listed in Table 6-21.

If a unity gain is expected from the A/D section, the gain of that filter has to be multiplied by 1/1.0137 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-20).

Table 6-21 Coefficient of the Filter Shown in Figure 6-35

```
; Source filter file: “compad81.IIR.Filter”

_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     5               ; number of biquadratic stages

                dc      $000d           ; g = 0.0007686455615/2

                ; Biquad stage no. 1
                dc      $d344           ; -d2_1 = -0.6989950203/2
                dc      $5c11           ; -d1_1 = 1.438546964/2
                dc      $0032           ; n2_1 = 0.003056686097/2
                dc      $1ca7           ; n1_1 = 0.4476798556/2
                ; Biquad stage no. 2
                dc      $cc88           ; -d2_2 = -0.8042193348/2
                dc      $531c           ; -d1_2 = 1.298599572/2
                dc      $fe28           ; n2_2 = -0.02879117954/2
                dc      $191c           ; n1_2 = 0.3923349351/2
                ; Biquad stage no. 3
                dc      $d248           ; -d2_3 = -0.7143639147/2
                dc      $691f           ; -d1_3 = 1.64250833/2
                dc      $f9a7           ; n2_3 = -0.09919829751/2
                dc      $12b0           ; n1_3 = 0.2920120584/2
                ; Biquad stage no. 4
                dc      $c704           ; -d2_4 = -0.8904019637/2
                dc      $4c31           ; -d1_4 = 1.190507817/2
                dc      $e31d           ; n2_4 = -0.4513353886/2
                dc      $de6a           ; n1_4 = -0.5247662529/2
                ; Biquad stage no. 5
                dc      $c3c1           ; -d2_5 = -0.9413669532/2
                dc      $7bed           ; -d1_5 = 1.936312495/2
                dc      $d103           ; n2_5 = -0.7341826041/2
                dc      $ef0b           ; n1_5 = -0.2649274056/2
```

Figure 6-36 shows the overall response of the A/D section cascaded with the DSP IIR filter described above.

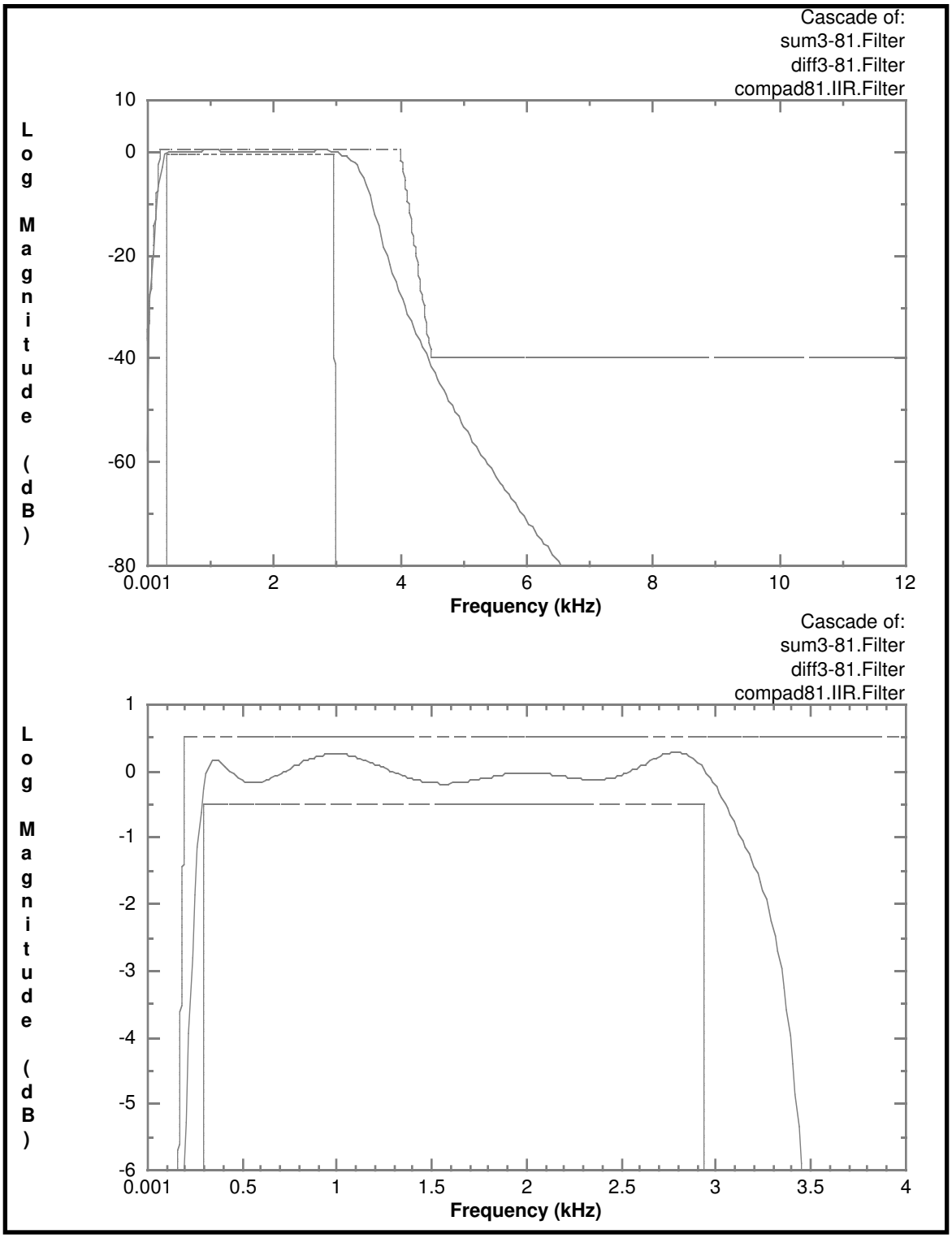


Figure 6-36 Overall Response of the A/D Section when Using the IIR Filter of Figure 6-35

6.6.4.2 D/A Interpolation Filter

In addition to reconstruction, the D/A interpolation filter also needs to compensate the D/A section frequency response. For a decimation ratio D=81 and a master clock F=1.944 MHz, the frequency response of the D/A section is:

$$F(f) = \frac{1}{81 \times 128}\left(\frac{\sin\left(\frac{2\pi \times 81}{2 \times 1944} \times f\right)}{\sin\left(\frac{2\pi}{2 \times 1944} \times f\right)}\right)^2$$

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.

An example of such a filter is given in Figure 6-37.

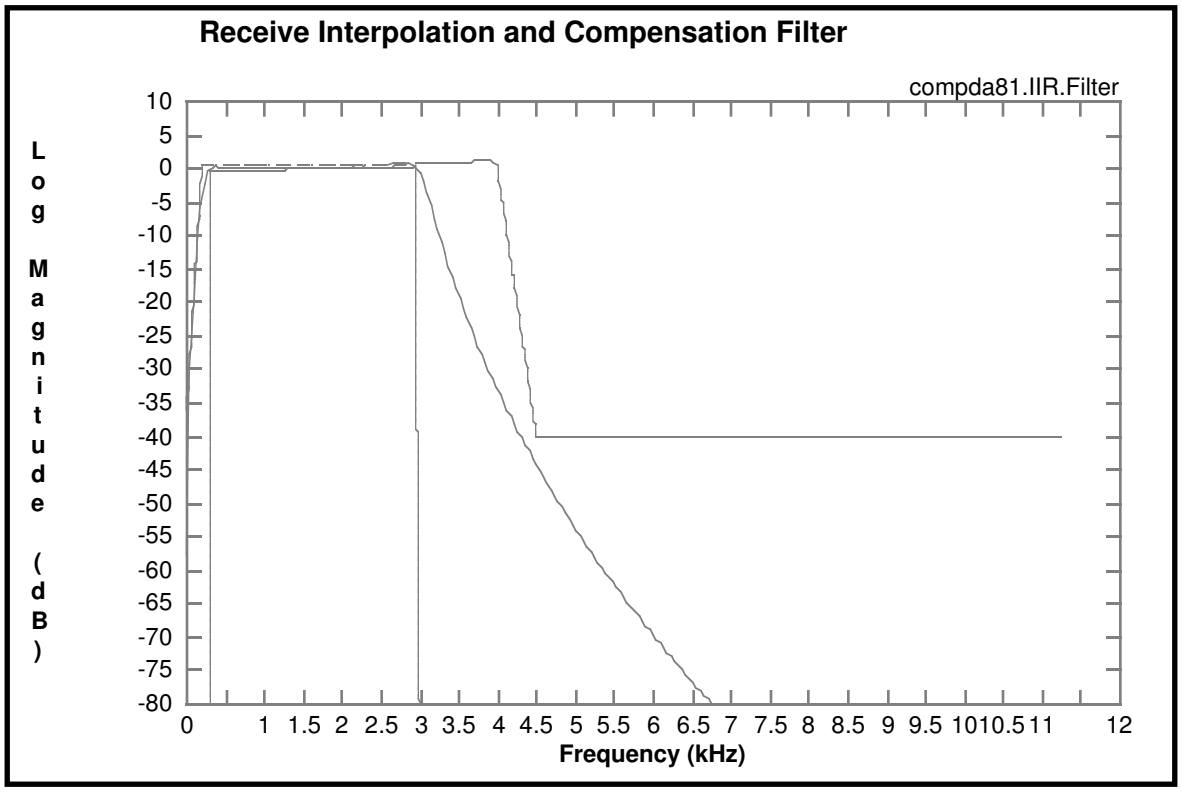


Figure 6-37 Example of a Receive Reconstruction-interpolation IIR Filter

The IIR filter shown in Figure 6-37 is implemented using 4 biquadratic sections. The gain and coefficients of the four section are listed in Table 6-22.

If a unity gain is expected from the D/A section, the gain of that filter has to be multiplied by 1/1.1254 in order to compensate the attenuation of the comb filter at the given decimation ratio (refer to Table 6-20).

Table 6-22 Coefficient of the Filter Shown in Figure 6-37

```
; Source filter file: "compda81.IIR.Filter"
_filter_type    equ     BIQUAD_FILTER_TYPE
_NSTAGES        equ     5               ; number of biquadratic stages

                dc      $000a           ; g = 0.0006268648952/2

                ; Biquad stage no. 1
                dc      $d46c           ; -d2_1 = -0.6808814098/2
                dc      $66ba           ; -d1_1 = 1.605119592/2
                dc      $fea3           ; n2_1 = -0.02132244978/2
                dc      $1978           ; n1_1 = 0.3979240128/2
                ; Biquad stage no. 2
                dc      $c68a           ; -d2_2 = -0.8978265055/2
                dc      $562a           ; -d1_2 = 1.346321716/2
                dc      $fc9d           ; n2_2 = -0.05290585592/2
                dc      $1626           ; n1_2 = 0.3460722284/2
                ; Biquad stage no. 3
                dc      $d631           ; -d2_3 = -0.6532654768/2
                dc      $55aa           ; -d1_3 = 1.338507259/2
                dc      $f8aa           ; n2_3 = -0.114654028/2
                dc      $1158           ; n1_3 = 0.2709873534/2
                ; Biquad stage no. 4
                dc      $db03           ; -d2_4 = -0.5779701159/2
                dc      $4e6d           ; -d1_4 = 1.225377027/2
                dc      $e18e           ; n2_4 = -0.4756898865/2
                dc      $df38           ; n1_4 = -0.512216145/2
                ; Biquad stage no. 5
                dc      $c3da           ; -d2_5 = -0.9398231986/2
                dc      $7bd9           ; -d1_5 = 1.935136984/2
                dc      $ca80           ; n2_5 = -0.8359239371/2
                dc      $f59e           ; n1_5 = -0.1622571097/2
```

Figure 6-38 shows the overall response of the DSP IIR filter cascaded with the D/A section.

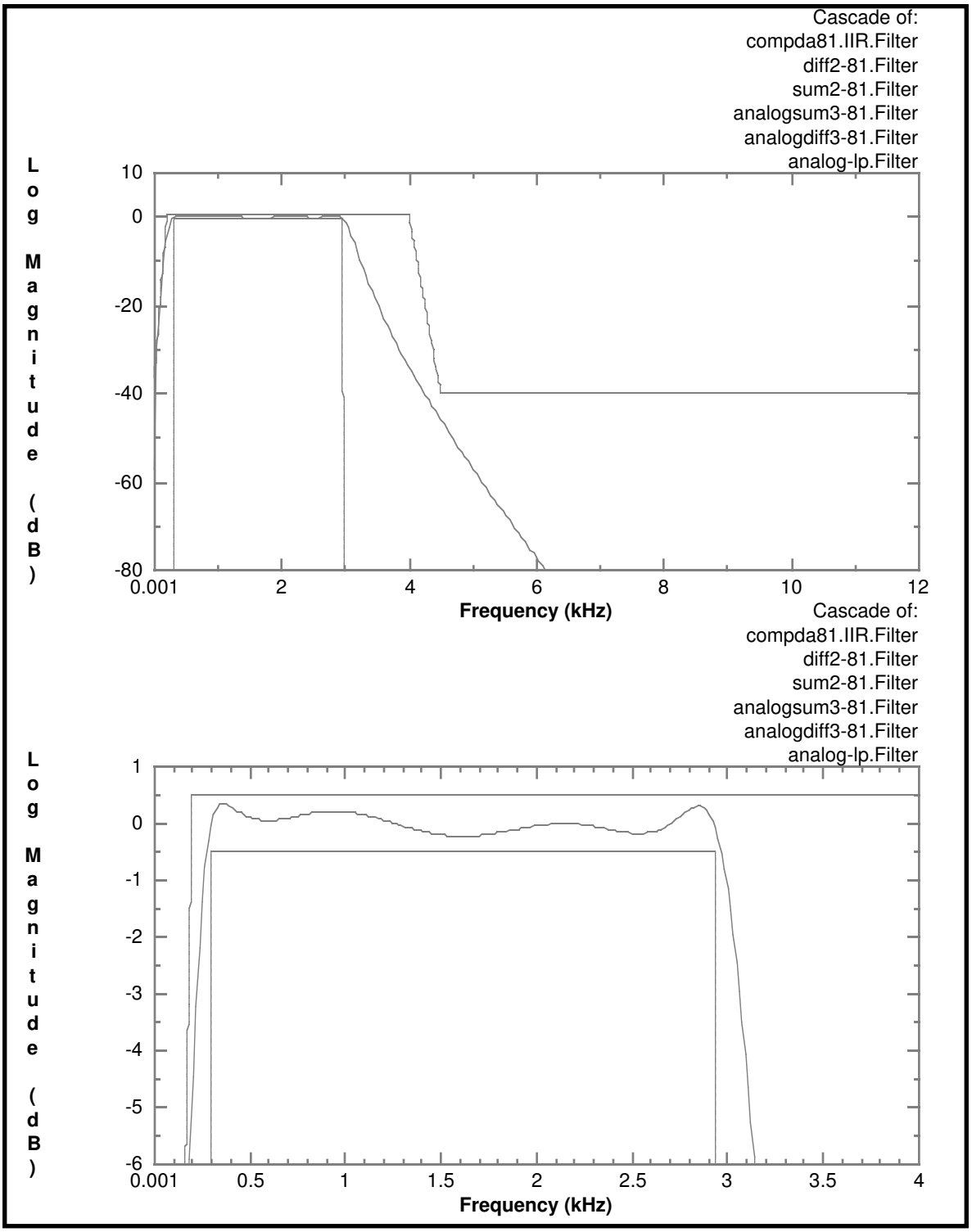


Figure 6-38 Overall Response of the D/A Section When Using the IIR Filter of Figure 6-37

6.6.5 Example 5 — Real-Time I/O Example with On-Chip Codec and PLL

The routine in this example performs real-time input-output for the main DSP routine. As mentioned, the DSP56156 has an on-chip A/D and D/A codec which allows the DSP to communicate with a microphone and a speaker directly. Thus, the set-up sequences for the codec as well as the on-chip frequency synthesizer (PLL) are included. There are several registers to be set before they are used, such as Bus Control Register (BCR), Interrupt Priority Register (IPR), Operating Mode Register (OMR), Codec Control Register (COCR), and PLL Control Register (PLCR). The operational functions for this routine are:

1. Read the comb filter output of the on-chip A/D converter (the comb filter performs the low pass filtering with 125:1 decimation).
2. Perform a 2:1 decimation low pass filter using a 4th-order IIR filter structure.
3. Test the result and save current data for the next interpolation filter section. The main routine can be called at this time.
4. Obtain the output data for the D/A converter or repeat the last output sample for 2:1 interpolation.
5. Run a 1:2 interpolation low pass filter routine using a 4th-order IIR filter.
6. Write the interpolation filter output to the on-chip D/A comb filter section which performs a 1:125 interpolation low pass filter. Note that this routine uses interrupt driven I/O and the master clock for the DSP56156 is synthesized by the PLL circuitry.

```
;*****************************************************************************************************************
; Real-Time I/O routine
;
;       Fext = 32Mhz and plcr = #$4f as an example
;       DSP master clock Fosc = 32/(15+1)x4x(6+1)=56MHz
;       Codec Clock 2MHz = 32M/16
;       Codec output rate 8KHz (125:1 and 2:1 decimation with 2MHz)
;
;       Note: 1. Interrupt driven codec routine
;             2. Master clock is driven from PLL
;*****************************************************************************************************************
;
            opt    cc,mu,now
nstages     equ    4                ;number of IIR stages
Nc          equ    4*nstages+1      ;include room for gain
Nw          equ    2*nstages        ;memory requirement for filter states
start       equ    $40              ;Start address for the program
cocr        equ    $ffc8            ;Codec control register
ctx         equ    $ffe9            ;Codec transmit data reg
crx         equ    $ffe9            ;Codec receive reg
cosr        equ    $ffe8            ;Codec status register
plcr        equ    $ffdc            ;Pll control register
ipr         equ    $ffdf            ;Interrupt Priority Register
```

```
bcr     equ   $ffde             ;Bus control register

        org   x:0
w       ds    Nw                ;x memory reserved for the states
        ds    Nw                ;x memory reserved for the states
;
; Decimation Filter Coefficients (125:1 comb ratio)(2:1 IIR ratio)
;
        dc    $0216             ; gain = $160/0.659=[0.02149566562/2]/0.659
                                ; to take into account the -3.618dB loss of
                                ; the A/D CombFilter
        ; Biquad stage no. 1
        dc    -$07a2            ; d2_1 = 0.1192590276/2
        dc    $e9c9             ; d1_1 = -0.3470842361/2
        dc    $2cf8             ; n2_1 = 0.7026065355/2
        dc    $6905             ; n1_1 = 1.640903034/2
        ; Biquad stage no. 2
        dc    -$3645            ; d2_2 = 0.8479421644/2
        dc    $e976             ; d1_2 = -0.3521904578/2
        dc    $34c4             ; n2_2 = 0.8244448877/2
        dc    $186c             ; n1_2 = 0.3815657788/2
        ; Biquad stage no. 3
        dc    -$119b            ; d2_3 = 0.2750857833/2
        dc    $f0c9             ; d1_3 = -0.2377206739/2
        dc    $2697             ; n2_3 = 0.6029962664/2
        dc    $62e2             ; n1_3 = 1.545046163/2
        ; Biquad stage no. 4
        dc    -$15ea            ; d2_4 = 0.3424061824/2
        dc    $f137             ; d1_4 = -0.2310341613/2
        dc    $3866             ; n2_4 = 0.8811980629/2
        dc    $31cb             ; n1_4 = 0.7780126911/2
;
; Interpolation Filter Coefficients (1:125 comb ratio)(1:2 IIR ratio)
;
        dc    $0123             ; gain=$01fa*0.5758=[0.03087683765/2]*0.5758
                                ; to take into account the +4.794dB gain of
                                ; the D/A CombFilter
        ; Biquad stage no. 1
        dc    -$0661            ; d2_1 = 0.09967390682/2
        dc    $eb0f             ; d1_1 = -0.3272240632/2
        dc    $1b48             ; n2_1 = 0.4262653287/2
        dc    $5692             ; n1_1 = 1.3526638/2
        ; Biquad stage no. 2
        dc    -$35b6            ; d2_2 = 0.8392111942/2
        dc    $ead9             ; d1_2 = -0.3304817315/2
        dc    $34d4             ; n2_2 = 0.8254454107/2
        dc    $1767             ; n1_2 = 0.3656574962/2
        ; Biquad stage no. 3
        dc    -$1078            ; d2_3 = 0.2573392333/2
        dc    $f271             ; d1_3 = -0.2118237522/2
        dc    $1ab7             ; n2_3 = 0.4174162168/2
```

```
            dc      $577b                           ; n1_3 = 1.366865914/2
            ; Biquad stage no. 4
            dc      -$14e9                          ; d2_4 = 0.3267375315/2
            dc      $f302                           ; d1_4 = -0.2029987284/2
            dc      $3998                           ; n2_4 = 0.8999033261/2
            dc      $3080                           ; n1_4 = 0.7578121768/2
;           temporary storage for I/O test purpose
flag        dc      0
inbuf       dc      0                               ;input data buffer
outbuf      dc      0                               ;output data buffer
pre_out     dc      0                               ;output buffer for interpolation
;*********************************************************
;
;
            org     p:$0                            ;reset interrupt vector
            jmp     start                           ;jump to main routine
            org     p:$2E                           ;codec interrupt vector
            jsr     codec                           ;codec interrupt service routine
            org     p:start                         ;main program start address
            ori     #$03,mr                         ;disable all interrupts
            move    #>0,x0
            move    x0,x:bcr                        ;Zero wait (no delay on bus access)
            move    #$c408,x0                       ;coie=coe=1;msg0=0(-6dB);CRS=0,MUT~=1;V3-V0=$8
            move    x0,x:cocr                       ;set Codec Control Register
            move    #plcr,r0                        ;get the PLL control register address
            move    #>$6f,x0                        ;divide by 16, 32M/16=2M for Codec clock
                                                    ;Core master clock Fosc = 32/(15+1)x4x(6+1)=56M
            move    x0,x:(r0)                       ;set PLL control register(not include bit4-7)
lock        bftsth  #$8000,x:(r0)                   ;check PLL register for VCO lock?
            bcc     lock                            ;if not, loop around until it is locked
            bfset   #$4000,x:(r0)                   ;enable PLL clock
            ori     #$80,omr                        ;disable clock out, may be quite for codec and PLL
            move    #$00c0,x0
            move    x0,x:ipr                        ;codec IPL enable level 2
            andi    #$fc,mr                         ;enable chip PL to level 2
lo          bra     lo                              ;loop forever for test (wait for interrupt)
;***********************************
;           codec interrupt routine
;***********************************
codec       move    x:crx,a                         ; read data from a/d
            move    #w,r0
            move    #w+Nw+Nw,r3
            move    #w+Nw+Nw+3,r1
            move    #-1,n0
            move    #4,n1
            move    #2,n3
            bfset   #$800,sr                        ; enable Scale Up mode
            asr     a               x:(r3)+,x0      ; a = x(n)/2,x0=g/2
            move    a,y0                            ; y0 = x(n)
            mpy     y0,x0,a                          ; a = g/2 * x(n)
            move    x:(r0)+,y0      x:(r3)+,x0      ; y0 = w(n-2), x0 = a2/2
            do      #nstages,_end1
```

```
            mac   y0,x0,a       x:(r0)+n0,y1   x:(r3)+n3,x0   ;=a2/2w(n-2)y1=w(n-1)x0=a1/2
            macr   -y1,x0,a     y1,x:(r0)+                    ;a-=a1/2 * w(n-1);w'(n-2)=w(n-1)
            move  x:(r1)+n1,x0  x:(r3)+,x1                    ;x0 = b2/2; x1=b1/2
            mac   y0,x0,a       a,x:(r0)+                     ;a+= b2/2*w(n-2);w'(n-1) = <a>
            mac   y1,x1,a       x:(r0)+,y0     x:(r3)+,x0     ;a+= b1/2 * w(n-1)
_end1
            rnd   a             (r0)-
            move  x:(r1)+,y0                                  ;dummy move for r1 address to next filter
            bfclr #$800,sr
;*****************************
; decimation/interpolation
;*****************************
            move  #flag,r2
            move  a,x:outbuf                                  ;for loop back test purpose only
            bfchg #$0001,x:(r2)                               ;check flag for 2:1 decimation 1:2 interpolation
            jcc   chan2
            move  x:pre_out,a                                 ;get previous output for interpolation
            jmp   intpol                                      ;jump intpol without storing input sample
chan2
            move  a,x:inbuf                                   ;store decimated input data to input buffer
            move  a,x:pre_out                                 ;save sample for next interpolated output
;*****************************
; start interpolation filter
;*****************************
intpol
            bfset #$800,sr
            move  a,y0                                        ; y0 = x(n)
            mpy   y0,x0,a                                     ; a = g/2 * x(n)
            move  x:(r0)+,y0    x:(r3)+,x0                    ; y0 = w(n-2), x0 = a2/2
            do    #nstages,_end2
            mac   y0,x0,a       x:(r0)+n0,y1   x:(r3)+n3,x0   ;=a2/2w(n-2)y1=w(n-1)x0=a1/2
            macr  -y1,x0,a      y1,x:(r0)+                    ;a-=a1/2 * w(n-1);w'(n-2)=w(n-1)
            move  x:(r1)+n1,x0  x:(r3)+,x1                    ;x0 = b2/2; x1=b1/2
            mac   y0,x0,a       a,x:(r0)+                     ;a+= b2/2*w(n-2);w'(n-1) = <a>
            mac   y1,x1,a       x:(r0)+,y0     x:(r3)+,x0     ;a+= b1/2 * w(n-1)
_end2
            rnd   a
            move  a,x:ctx                                     ;output it to the d/a
            bfclr #$800,sr
            nop
            rti
            end
```

6.7 DSP PROGRAM FLOWCHART

The last A/D section decimation-compensation filter as well as the first D/A section interpolation/ compensation filters are performed by the DSP core of the DSP56156 processor. The same in-

terrupt routine can be used to perform these two functions since the two codec sections (A/D and D/A) are synchronous.

The time the DSP core spends on the decimation/compensation and interpolation/compensation filters is a function of:

- the antialiasing and reconstruction filter characteristics
- the characteristic of the compensation that needs to be performed
- the master DSP core clock speed
- the number of registers that need to be saved and restored
- the number and size of general tasks handled by the interrupt routine itself

Figure 6-39 shows an example of a flowchart for a long interrupt routine.

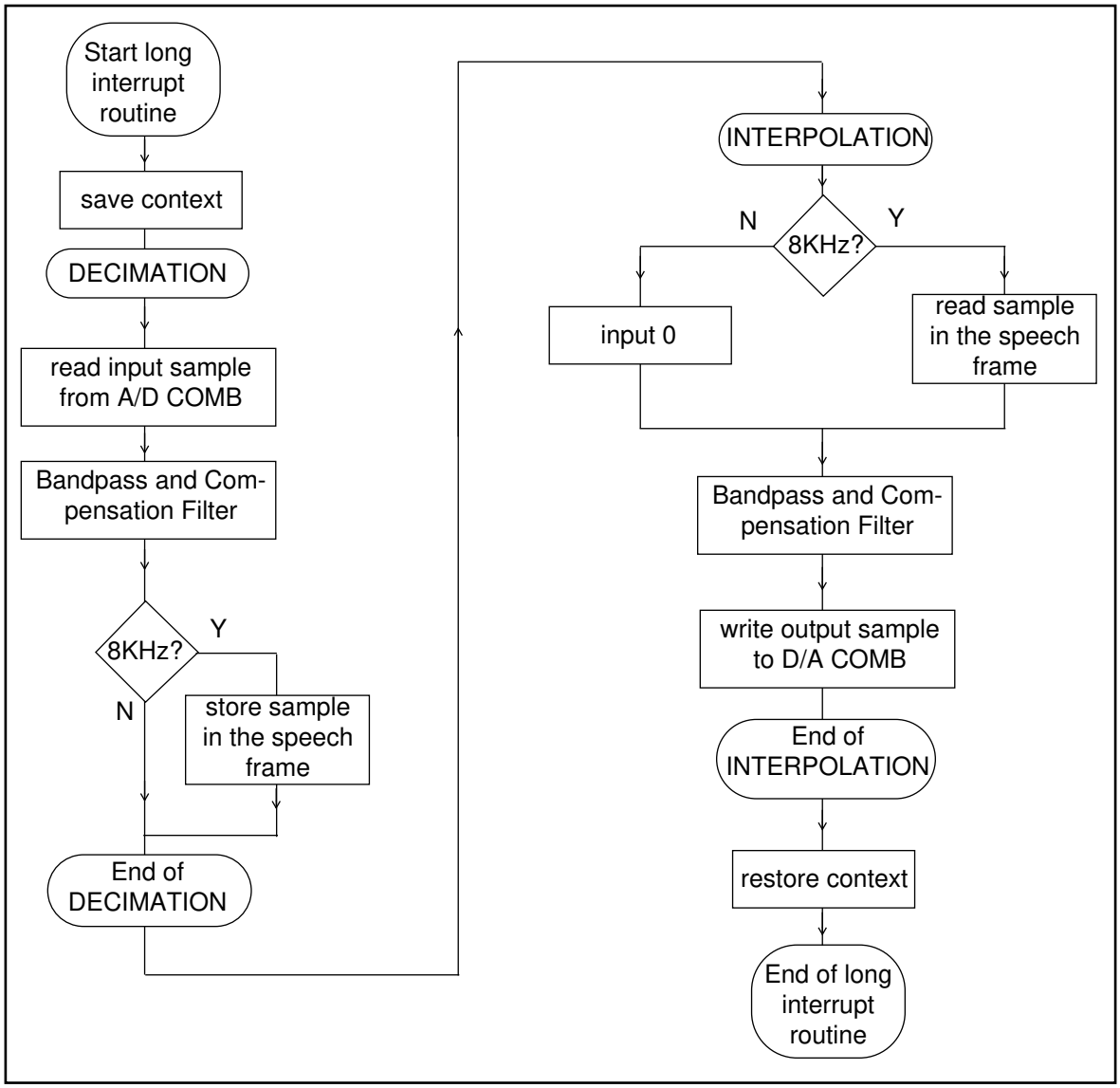
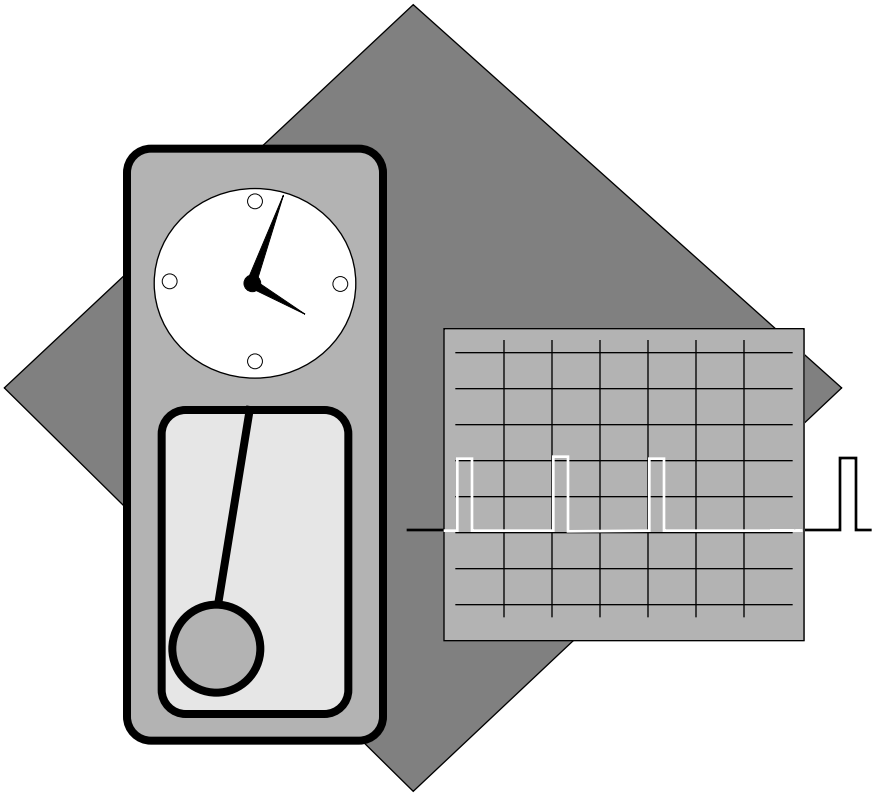


Figure 6-39 Flowchart of a Decimation/Interpolation Routine

SECTION 7

16-BIT TIMER AND EVENT COUNTER



SECTION CONTENTS

7.1 INTRODUCTION

This section describes 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. Note that these pins must be enabled by setting CC10 and CC11 respectively in the Port C Control Register.

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 in the Port C Control Register.

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 register 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).

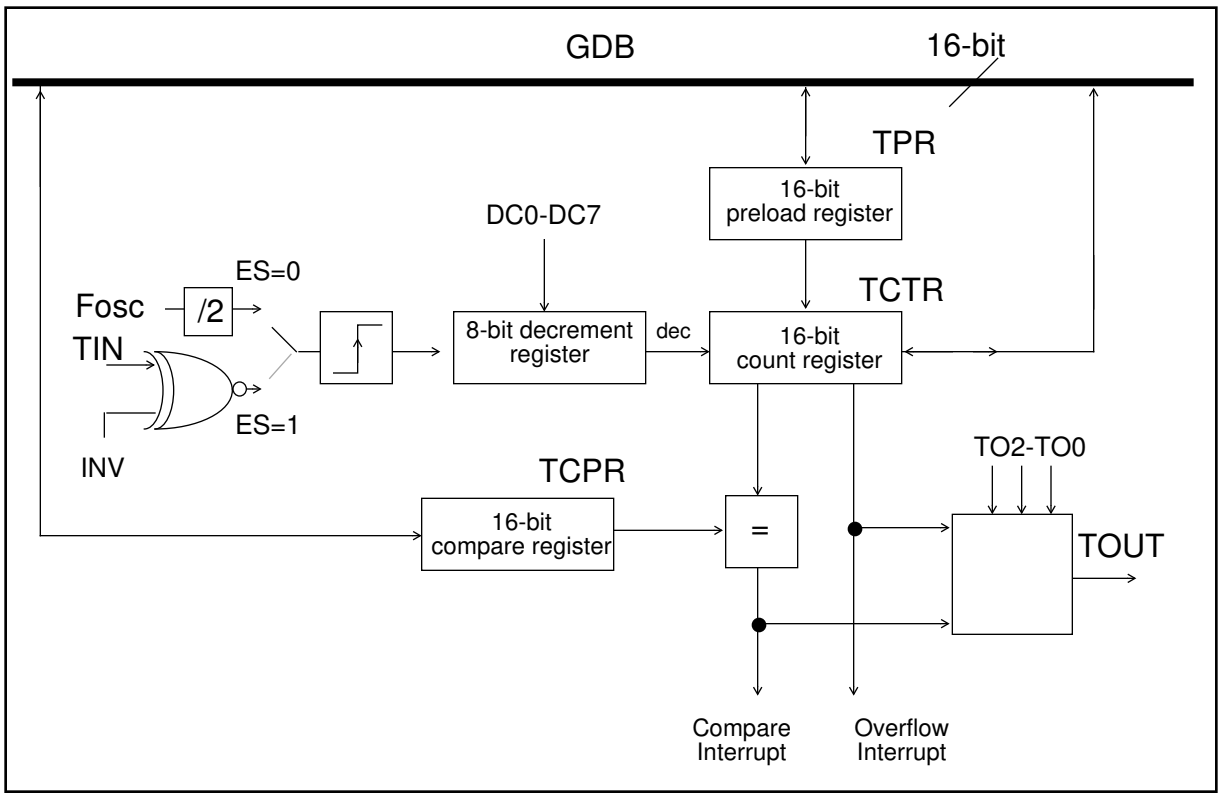


Figure 7-1 16-bit Timer General Block Diagram

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).

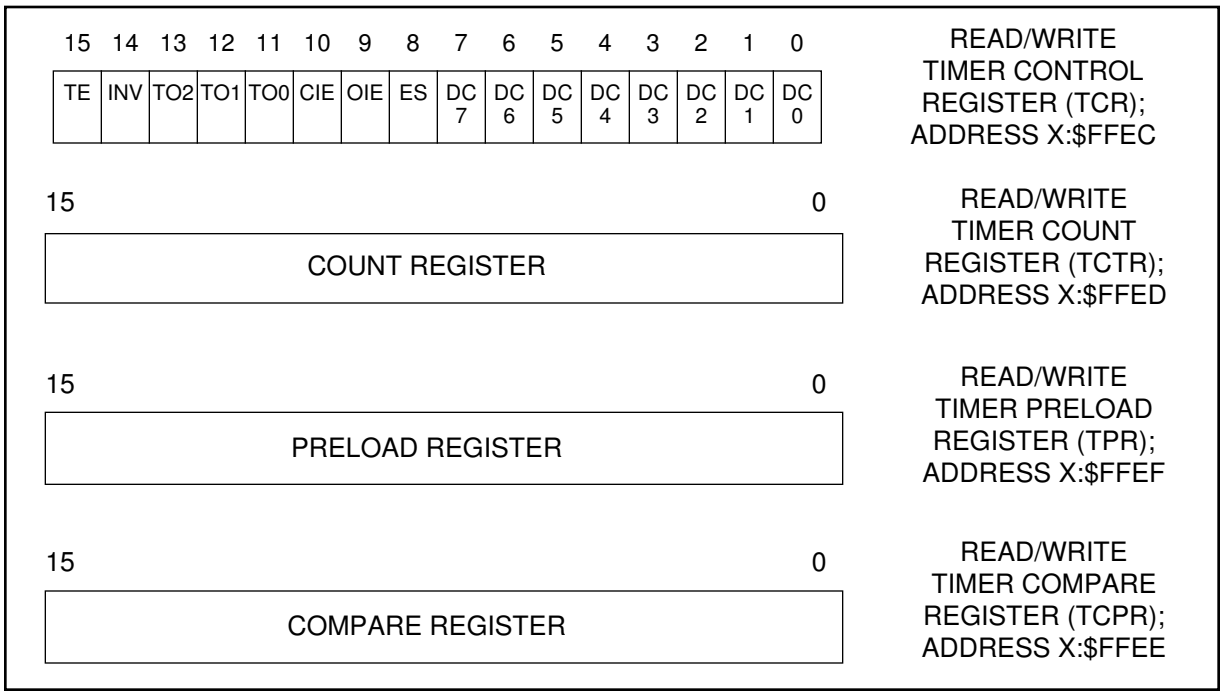


Figure 7-2 Timer Programming Model

7.5 TIMER COMPARE REGISTER (TCPR)

The output compare register is a 16-bit read/write register which is used to program an action to occur at a specific time 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 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.

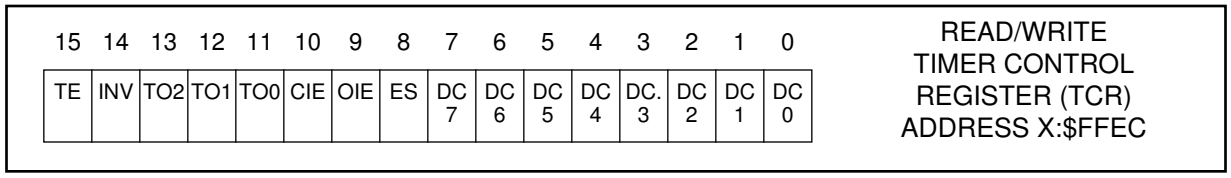


Figure 7-3 Timer Control Register

7.6.1 TCR Decrement Ratio (DC7-DC0) Bit 0-7

DC7-DC0 are 8 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 gets 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 set 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 in disabled. OIE bit is cleared on hardware $\overline{\text{RESET}}$ and software reset (RESET instruction).

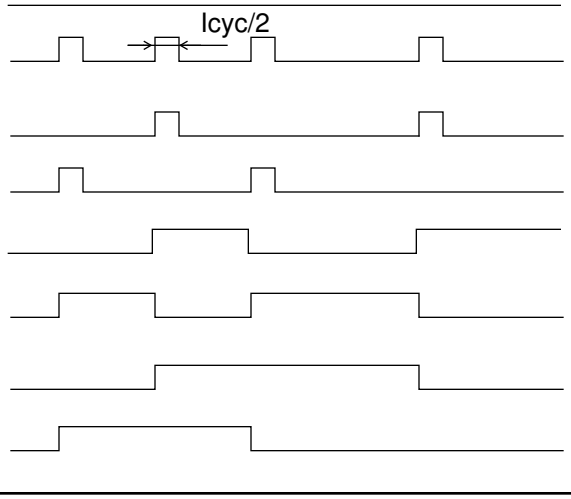
7.6.4 TCR Compare Interrupt Enable (CIE) Bit 10

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 in disabled. CIE bit is cleared on hardware $\overline{\text{RESET}}$ and software reset (RESET instruction).

7.6.5 TCR Timer Output Enable (TO2-TO0) Bit 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 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 rate (overflow interrupt using internal event, Fosc/2) that is provided by the 16-bit count register, the 16-bit preload register and the 8-bit decrement register in combination. The overflow interrupt occurs every (PRELOAD+1)*(DC7-DC0+1) input clock cycles.

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 µs
51 ns (39 MHz-19.5MIPS))	0 255	3.342 ms 855.6 ms	51 ns 13.06 µs
74 ns (27 MHz-13.5MIPS)	0 255	4.85 ms 1.242 s	74 ns 18.94 µs

7.8 FUNCTIONAL DESCRIPTION OF THE TIMER

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

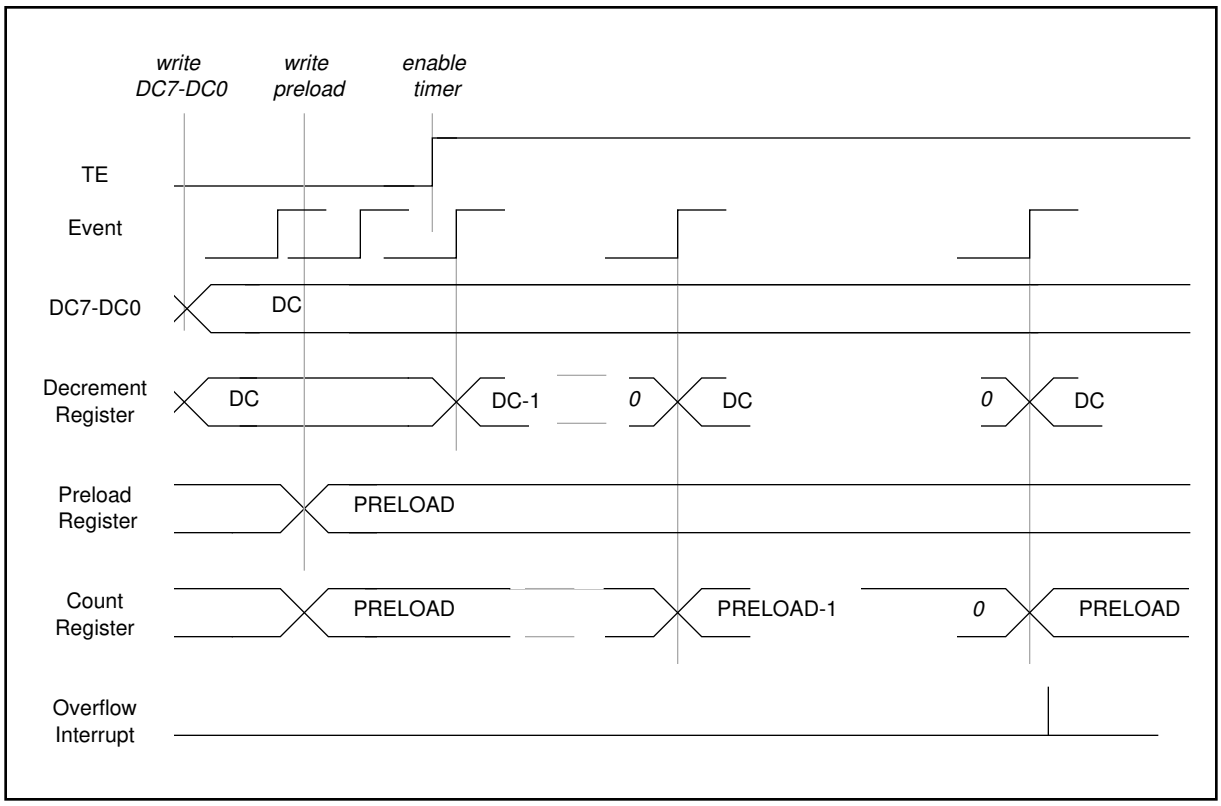


Figure 7-4 Standard Timer Operation with Overflow Interrupt

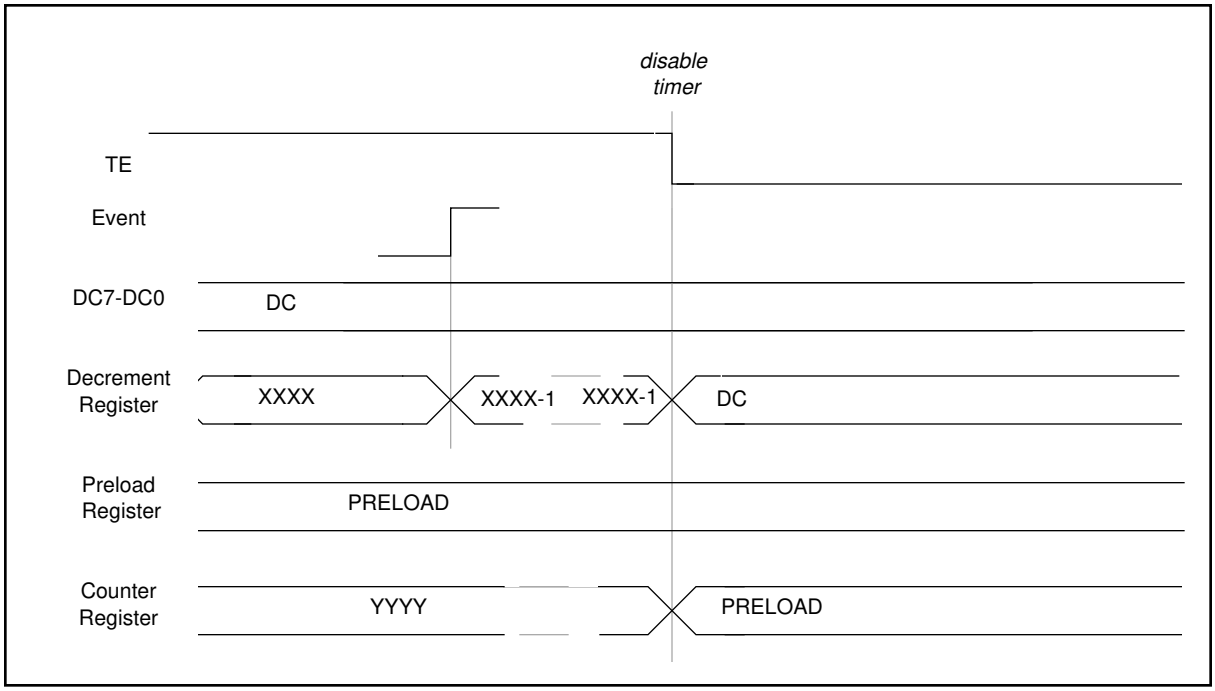


Figure 7-5 Standard Timer Disable

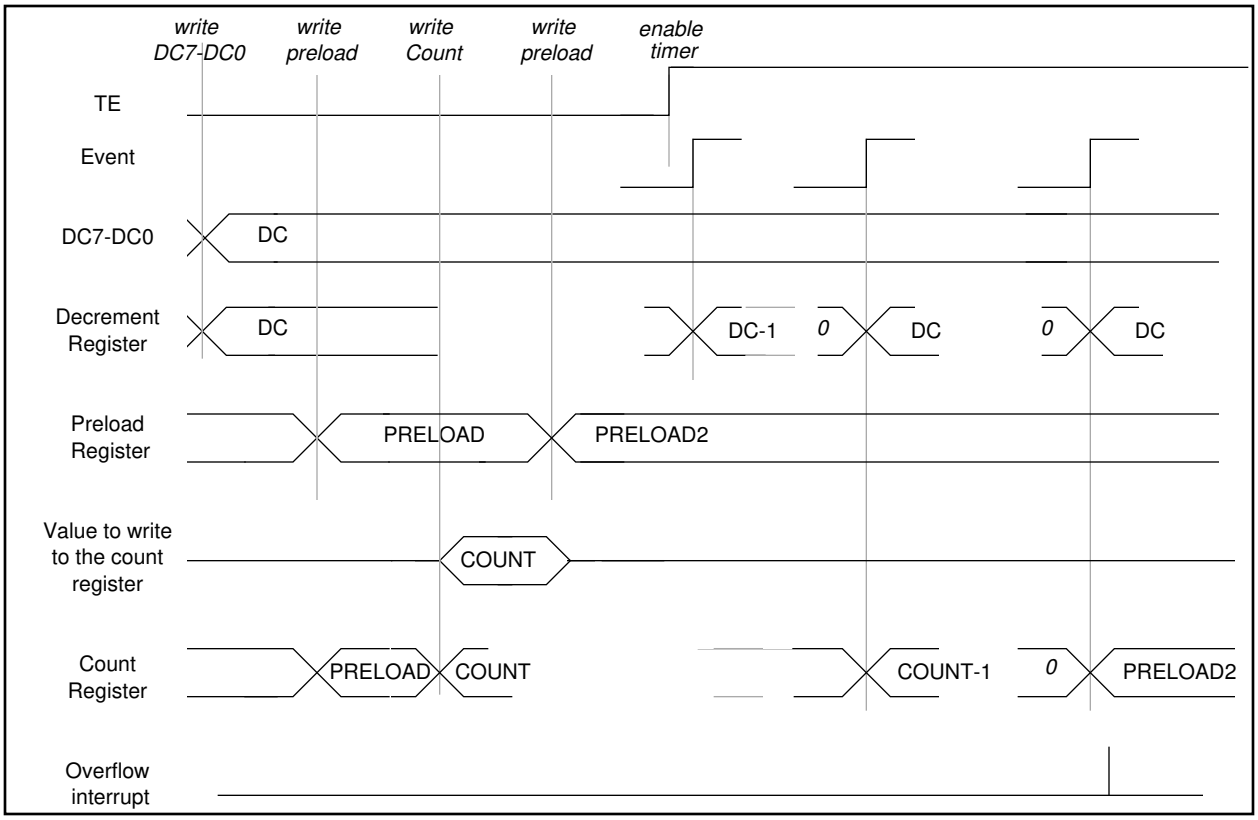


Figure 7-6
Write to the Count Register
After Writing to the Preload Register when the Timer is Disabled

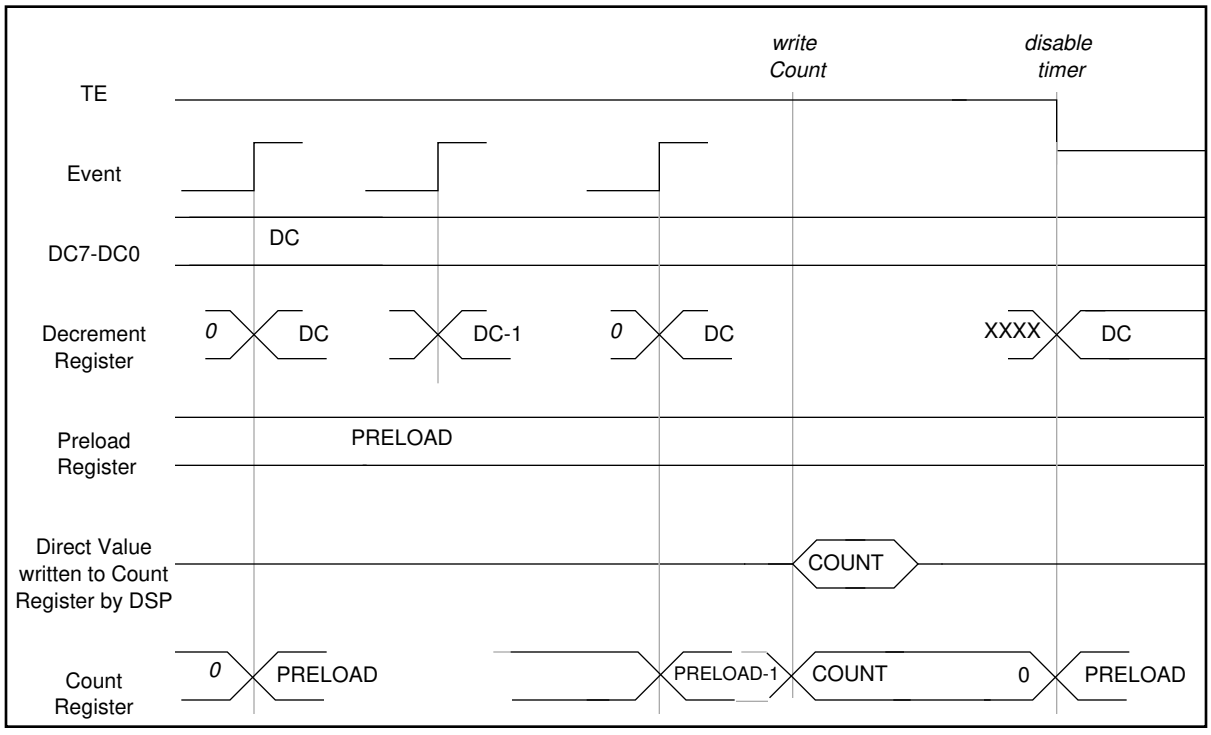


Figure 7-7 Timer Disable After a Write to the Count Register

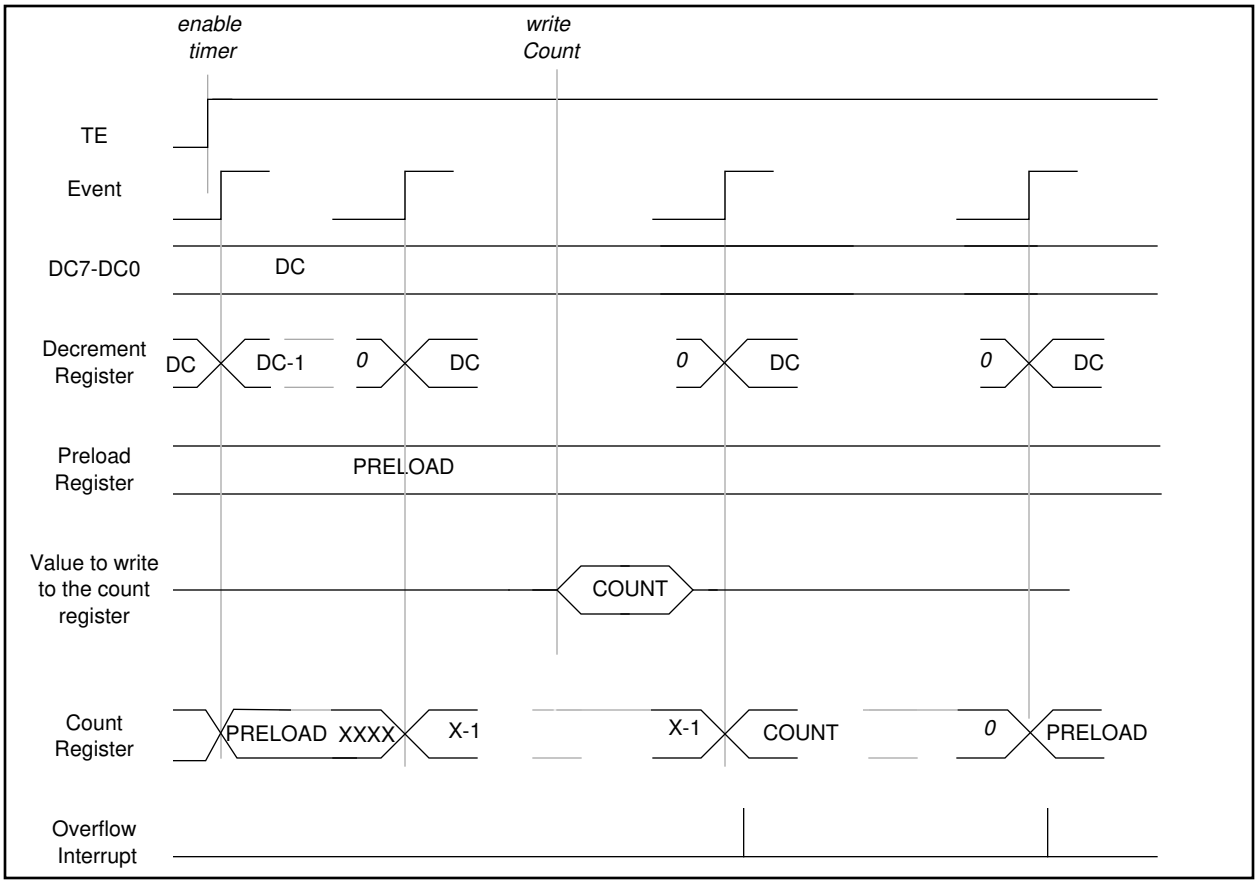


Figure 7-8 Write to the Count Register when the Timer is Enabled

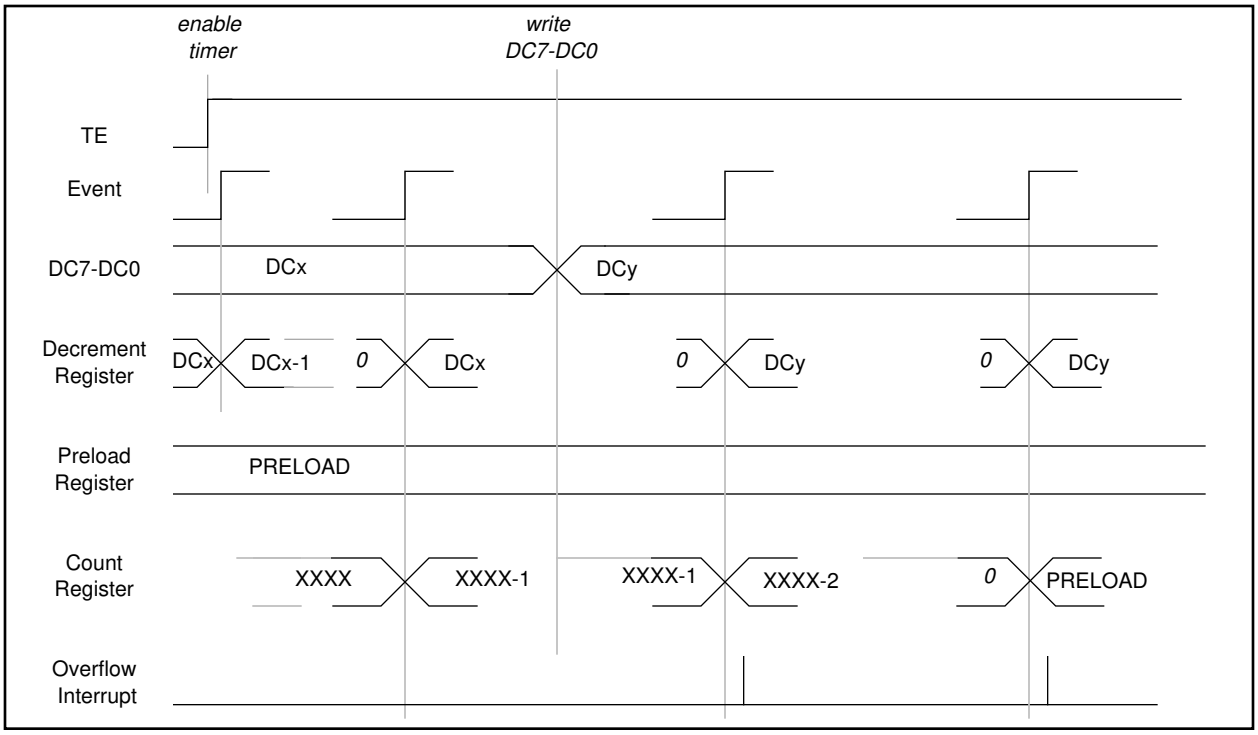


Figure 7-9 Write to DC7-DC0 when the Timer is Enabled

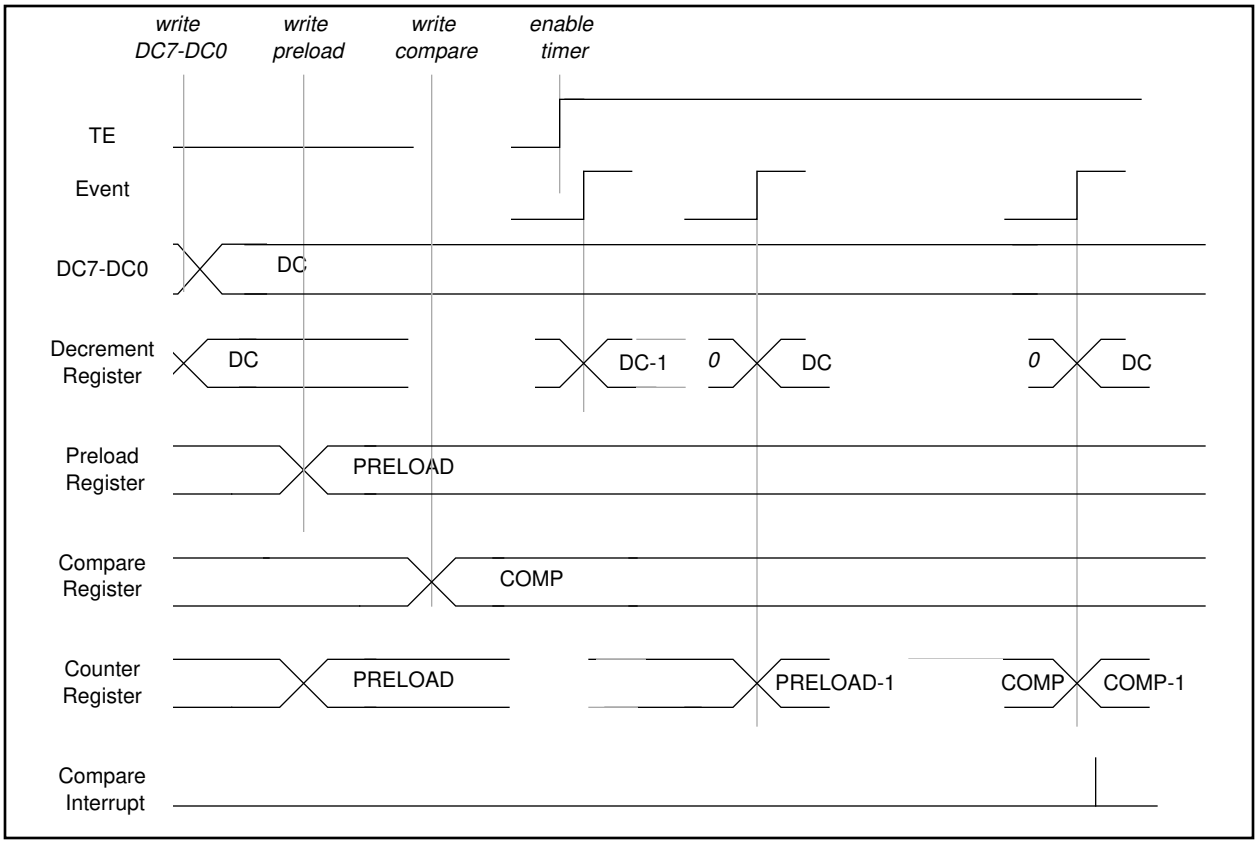
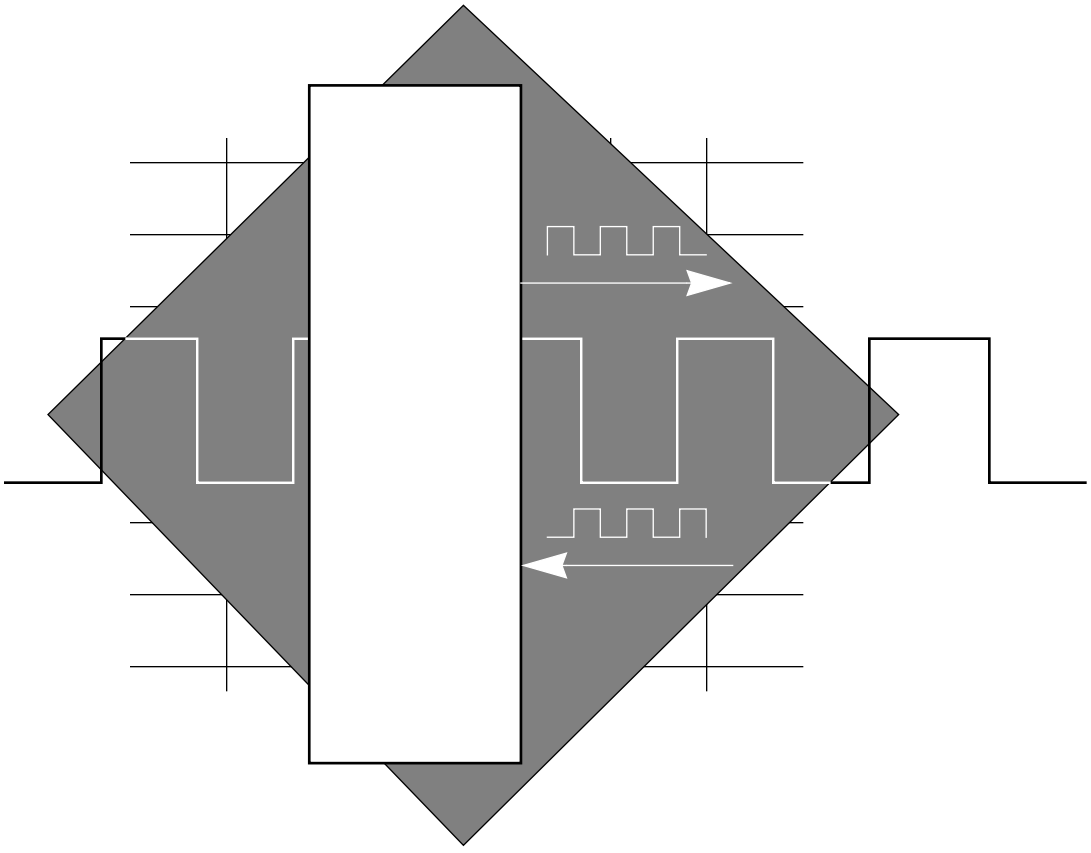


Figure 7-10 Standard Timer Operation with Compare Interrupt

SECTION 8

SYNCHRONOUS SERIAL INTERFACE (SSI0 and SSI1)



SECTION CONTENTS

8.1 INTRODUCTION

The DSP56156 contains two identical Synchronous Serial Interfaces (SSI's) named SSI0 and SSI1. This section describes both. In cases where the text or a figure applies equally to both SSI's, they will be referred to as the SSI. In cases where the information differs between the SSI's such as when pin numbers are mentioned, control addresses are mentioned or the operation affects only one of the two SSI's like a personal reset, the SSI will be referred to as SSIx meaning SSI0 or SSI1 – whichever applies.

The SSI 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, or microprocessors and peripherals. A selectable logarithmic compression and expansion is available for easier interface with PCM monocircuits and PCM highways. The SSI interface consists of independent transmitter and receiver sections and a common SSI clock generator.

8.2 SSI OPERATING MODES

The SSI has several basic operating modes. These modes can be programmed by several bits in the SSI Control registers. The Table 8-1 below lists the SSI operating modes:

Table 8-1 SSI Operating Modes

TX, RX Sections	Serial Clock	Protocol
Asynchronous	Continuous	Normal
Synchronous	Continuous	Normal
Synchronous	Continuous	Network
Asynchronous	Continuous	Network

The transmit and receive sections of this interface may be synchronous or asynchronous; that is, the transmitter and the receiver may use common clock and synchronization signals or they may have independent frame sync signals but the same bit clock. The SYN bit in SSI Control Register B selects synchronous or asynchronous operation. Since the SSI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

Normal or network protocol may also be selected. For normal protocol the SSI functions with one data word of I/O per frame. For network protocol, 2 to 32 data words of I/O may be used per frame. Network mode is used in Time Division Multiplexed (TDM) networks of codecs or DSPs. These distinctions result in the basic operating modes which allow the SSI to communicate with a wide variety of devices.

8.3 SSI 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 SSI clock generator is used to derive bit clock and frame sync signals from the DSP internal system clock. The SSI clock generator consists of a selectable fixed prescaler and a programmable prescaler for bit rate clock generation and also a programmable frame rate divider and a word length divider for frame rate sync signal generation.

8.4 SSIx DATA AND CONTROL PINS

The SSIx has five dedicated I/O pins for each SSI:

- Transmit data STDx (PC0 for SSI0 and PC5 for SSI1)
- Receive data SRDx (PC1 for SSI0 and PC6 for SSI1)
- Serial clock SCKx (PC2 for SSI0 and PC7 for SSI1)
- Serial Control Pin 1 SC1x (PC3 for SSI0 and PC8 for SSI1)
- Serial Control Pin 0 SC0x (PC4 for SSI0 and PC9 for SSI1)

Figure 8-1 through Figure 8-5 show the main configurations and the following paragraphs describe the uses of these pins for each of the SSIx operating modes.These figures do not represent all possible configurations, e.g., SCKx and FS don't have to be in the same direction. **Note that the first pin name in these figures apply to SSI0 and the second applies to SSI1 i.e., PC2/PC7 means PC2 for SSI0 and PC7 for SSI1.**

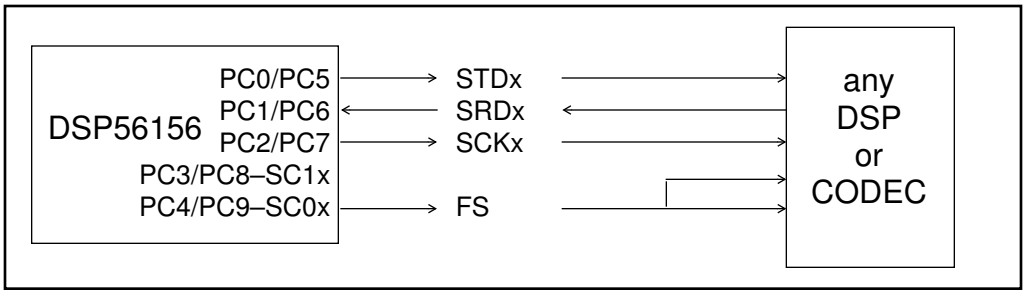


Figure 8-1 SSIx Internal Clock, Synchronous Operation

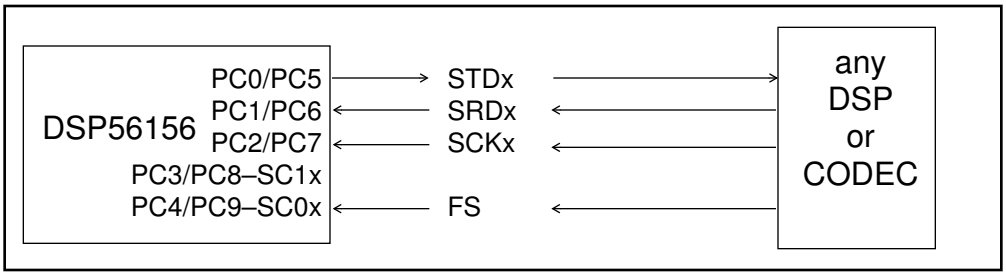


Figure 8-2 SSIx External Clock, Synchronous Operation

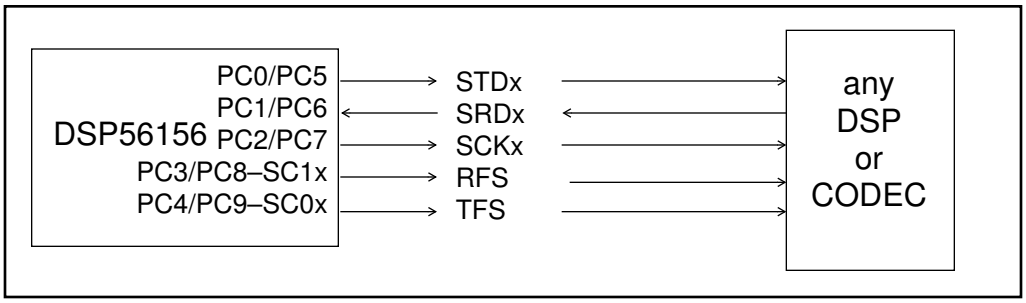


Figure 8-3 SSIx Internal Clock, Asynchronous Operation

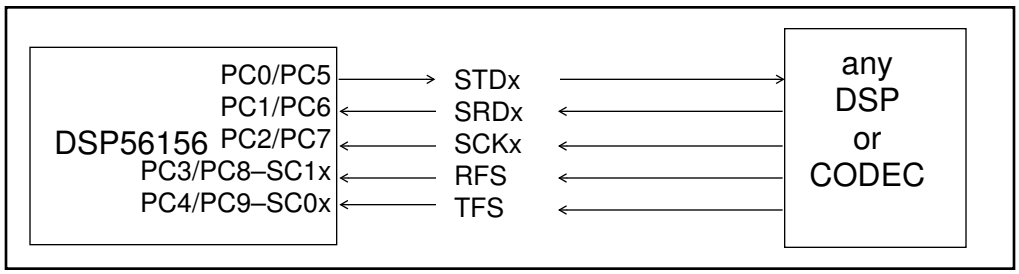


Figure 8-4 SSIx External Clock, Asynchronous Operation

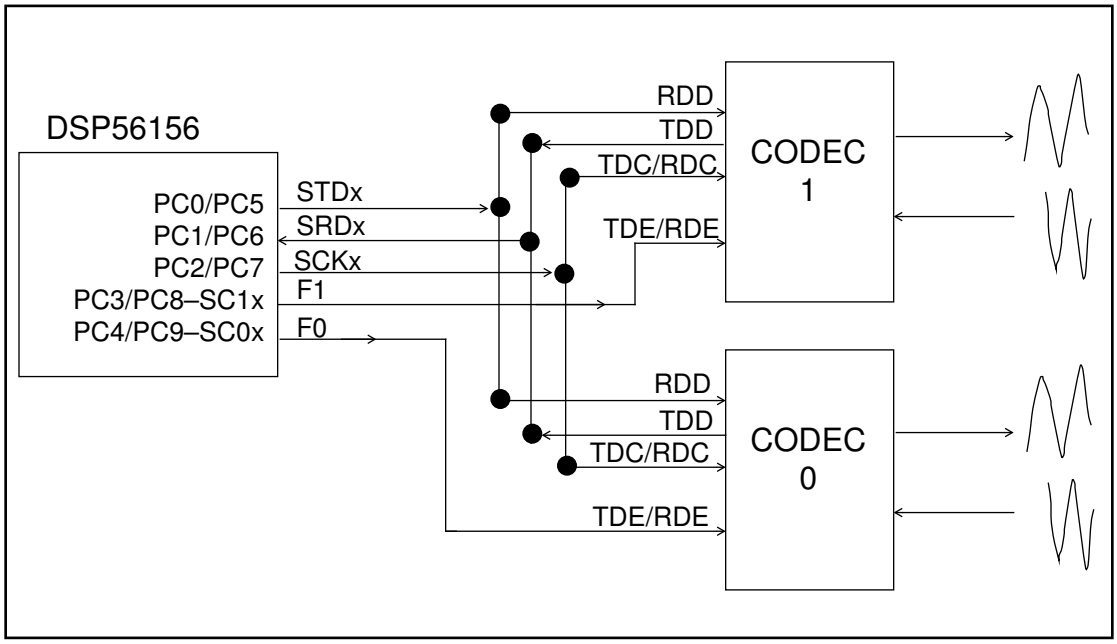


Figure 8-5 SSIx Internal Clock, Synchronous Operation Dual Codec Interface

Figure 8-6 shows the internal clock path connections in block diagram form. The serial bit clock can be internal or external depending on SCKD bit in the control register.

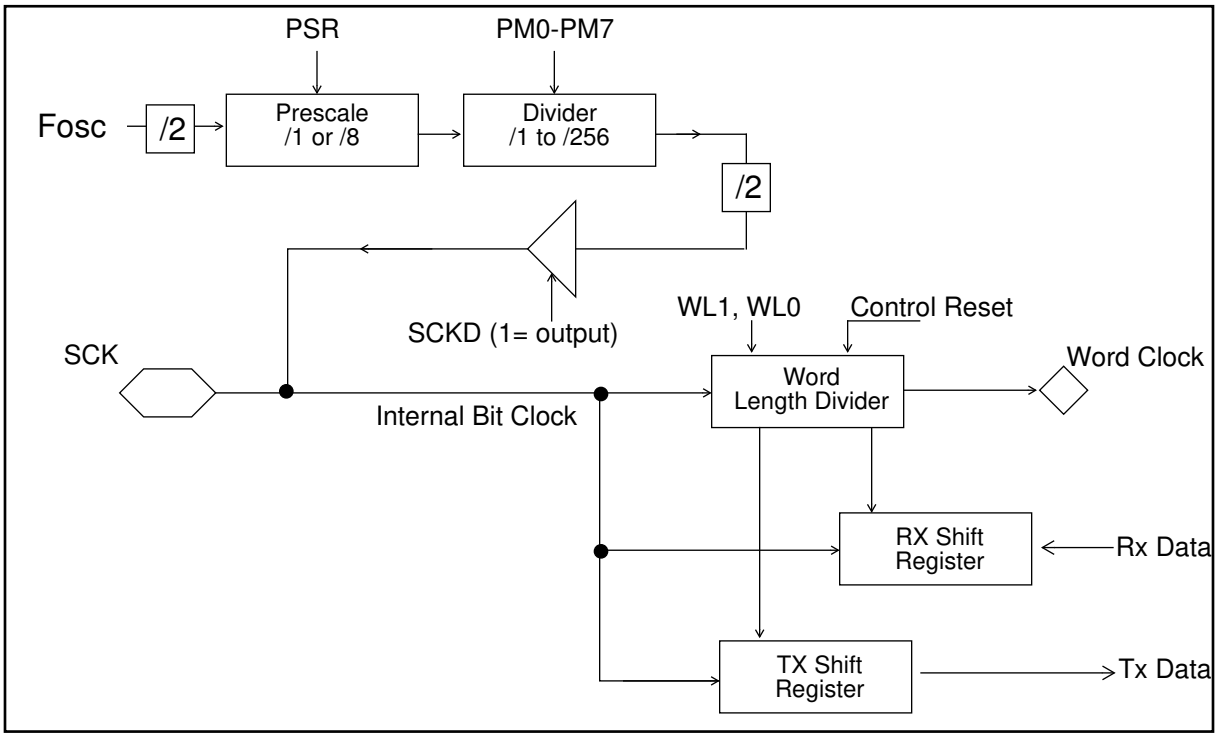


Figure 8-6 SSI Clock Generator Functional Block Diagram

Table 8-4 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 (DC4-DC0) bit and the word length (WL1-WL0) bits of CRA.

Figure 8-7 shows the functions of the two pins SC1x and SC0x according to the setting of CRB flags.

8.4.1 Serial Transmit Data Pin - STDx

The Serial Transmit Data Pin (STD) is used for 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 after the trailing edge of the bit clock after the last bit of a word is transmitted.

8.4.2 Serial Receive Data Pin - SRDx

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

8.4.3 Serial Clock - SCK

The Serial Clock (SCK) pin is used as a clock input or output used by both the transmitter and receiver in synchronous modes and in asynchronous modes.

8.4.4 Serial Control - SC1x

The function of this pin is determined by the flags SYNC, FSD0 and FSD1 of control register B (CRB) — see Table 8-4. In Asynchronous mode (SYNC=0), this pin is the receiver frame sync I/O. For synchronous mode (SYNC=1), with bits FSD0 and FSD1 set, pin SCIx is used as an output flag. When SC1x is configured as an output flag (FSD0=1; FSD0=1), this pin is controlled by bit OF1 in CRB. When SCIx is configured as an input or output (with synchronous or asynchronous operations), this pin will update status bit IF1 of the SSI status register as described in Section 8.13.1.

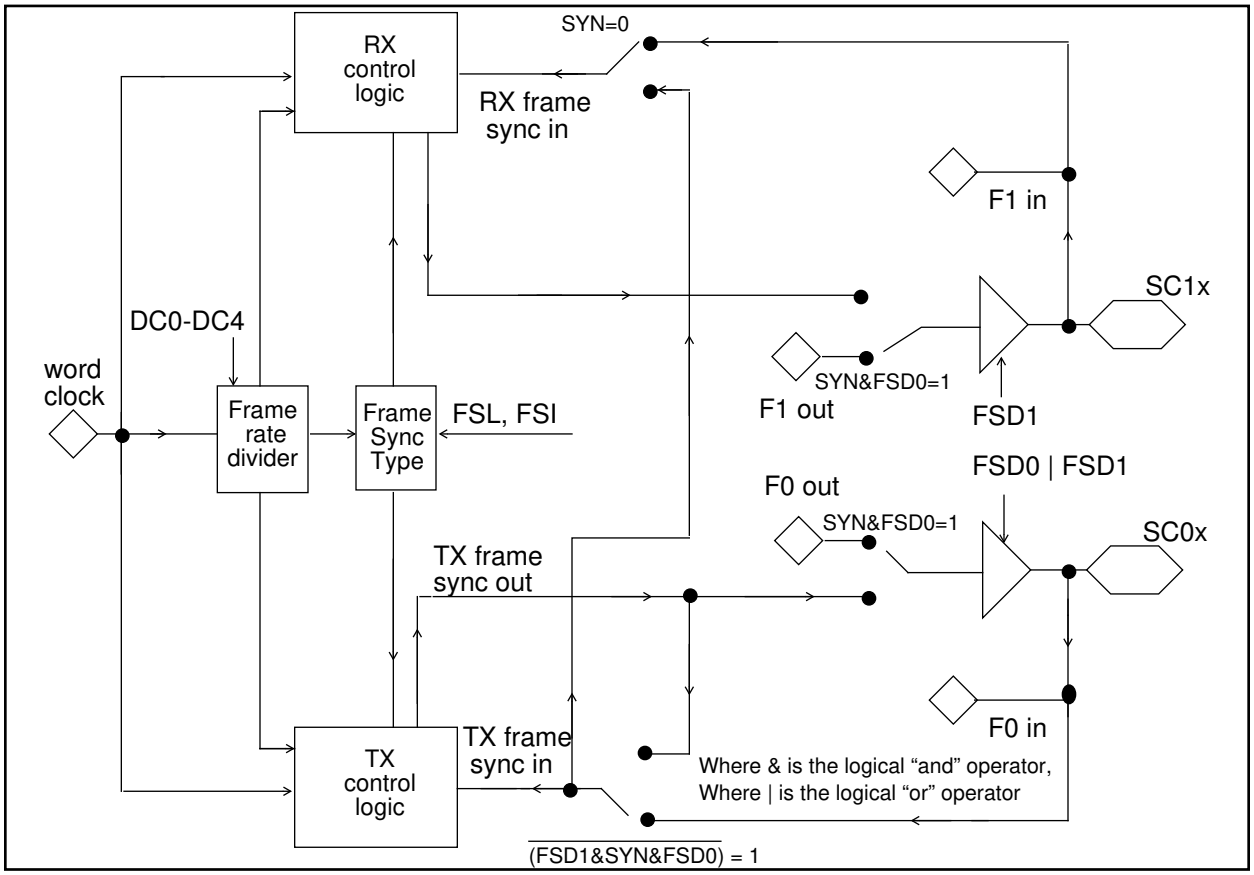


Figure 8-7 SSIx Frame Sync Generator Functional Block Diagram

8.4.5 Serial Control - SC0x

The function of this pin is determined by the SYNC, FSD0 and FSD1 flags in the CRB. In Asynchronous mode (SYNC=0), this pin is the transmitter frame sync I/O. For synchronous mode (SYNC=1), this pin is used as a frame sync I/O if bit FSD0 is cleared. If bits FSD0 and FSD1 are set, this pin is used as an output flag. When configured as an output flag (FSD0=1; FSD1=1), this pin is controlled by bit OF0 in the CRB and changes synchronously with the first transmitted bit of the data. Control status bit IF0 of the SSI status register will update regardless of the SYNC, FSD0 or FSD1 bits in the CRB.

8.5 SSI RESET AND INITIALIZATION PROCEDURE

The SSI is affected by three types of reset:

DSP Reset This reset is generated by either the DSP hardware reset (generated by asserting the $\overline{\text{RESET}}$ pin) or software reset (generated by executing the RESET instruction). The DSP reset clears the Port Control Register bits,

which configures all I/O pins as general purpose input. The SSI will remain in the reset state while all SSI pins are programmed as general purpose I/O (CC0-CC4/CC5-CC9 cleared) and will become active only when at least one of the SSIx I/O pins is programmed as NOT general purpose I/O. All status and control bits in the SSIx are affected as described below.

SSIx Reset The SSIx personal reset is generated when CC0-CC4/CC5-CC9 bits are cleared. This returns the SSIx pins to general purpose I/O pins. The SSIx status bits are preset to the same state produced by the DSP reset; however, the SSIx control bits are unaffected. The SSIx personal reset is useful for selective reset of the SSIx interface without changing the present SSIx control bits setup 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 SSI status bits are preset to the same state produced by the DSP reset. The SSI control bits are unaffected. The SSI pins remain defined as SSIx pins. The STOP reset condition is like the personal reset condition except that the SSI pins do not revert to general purpose I/O pins.

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

1. DSP reset or SSIx reset
2. Program SSIx control registers
3. Configure SSIx pins (at least one) as not general purpose I/O

The DSP programmer should use the DSP or SSIx reset before changing the MOD, SYN, FSI, FSL, MSB, SCKP, SCKD, FSD1, A/MU, FSD0 control bits to ensure proper operation of the SSIx interface. That is, these control bits should not be changed during SSIx operation.

Note: The SSIx clock must go low for at least four complete periods to ensure proper SSIx reset.

8.6 SSIx INTERFACE PROGRAMMING MODEL

The registers comprising the SSI interface are shown in Figure 8-8. Note that standard Codec devices label the Most Significant Bit as bit 0, whereas the DSP labels the LSB as bit 0. Therefore, when using a standard Codec (requiring LSB first), the SHFD bit in the CRB should be cleared (MSB first).

8.7 SSI TRANSMIT SHIFT REGISTER

This is a 16-bit shift register that contains the data being transmitted. Data is shifted out to the serial transmit data STD pin by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. The number of bits shifted out before the shift register is considered empty and may be written to again can be 8, 12, or 16 bits as determined by the Word Length control bits in the SSI Control Register A (WL1-WL0).

The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is shifted out of this register with the most significant bit (MSB) first when the SHFD bit of the control register B is cleared. If the SHFD bit is set, the LSB is output first. The Transmit Shift Register cannot be directly accessed by the programmer.

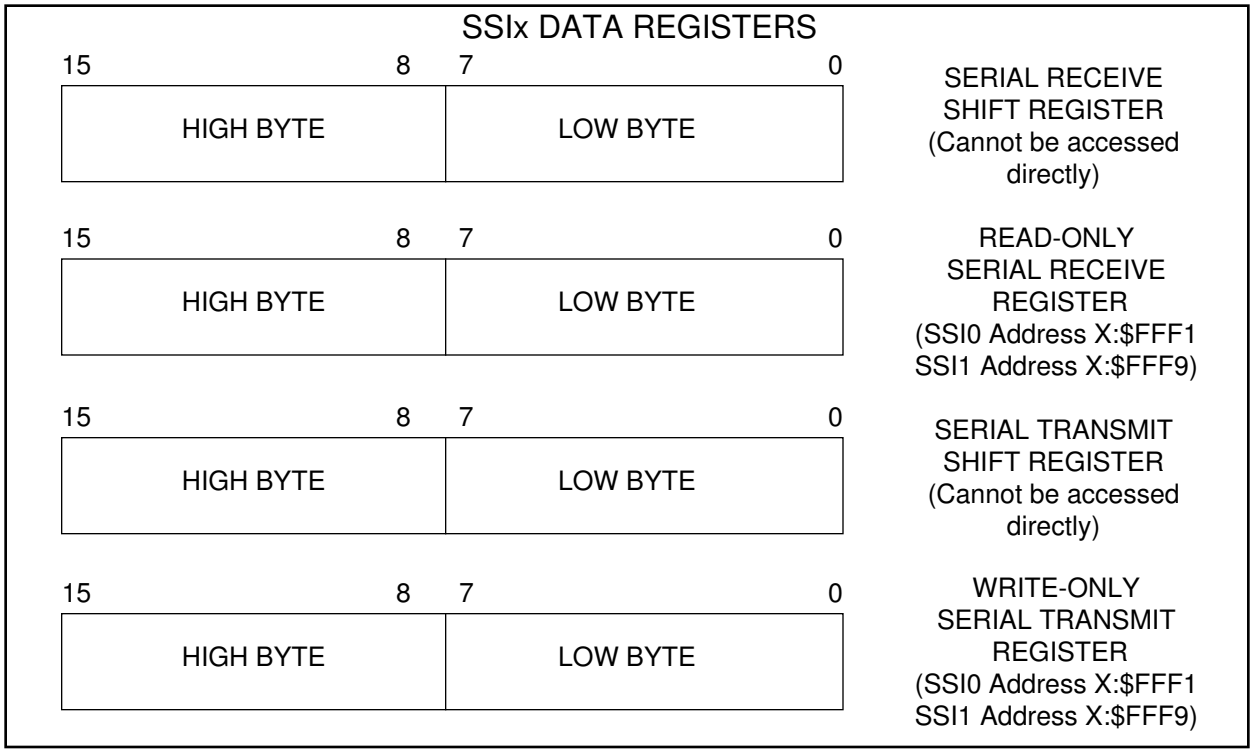


Figure 8-8 SSIx Programming Model

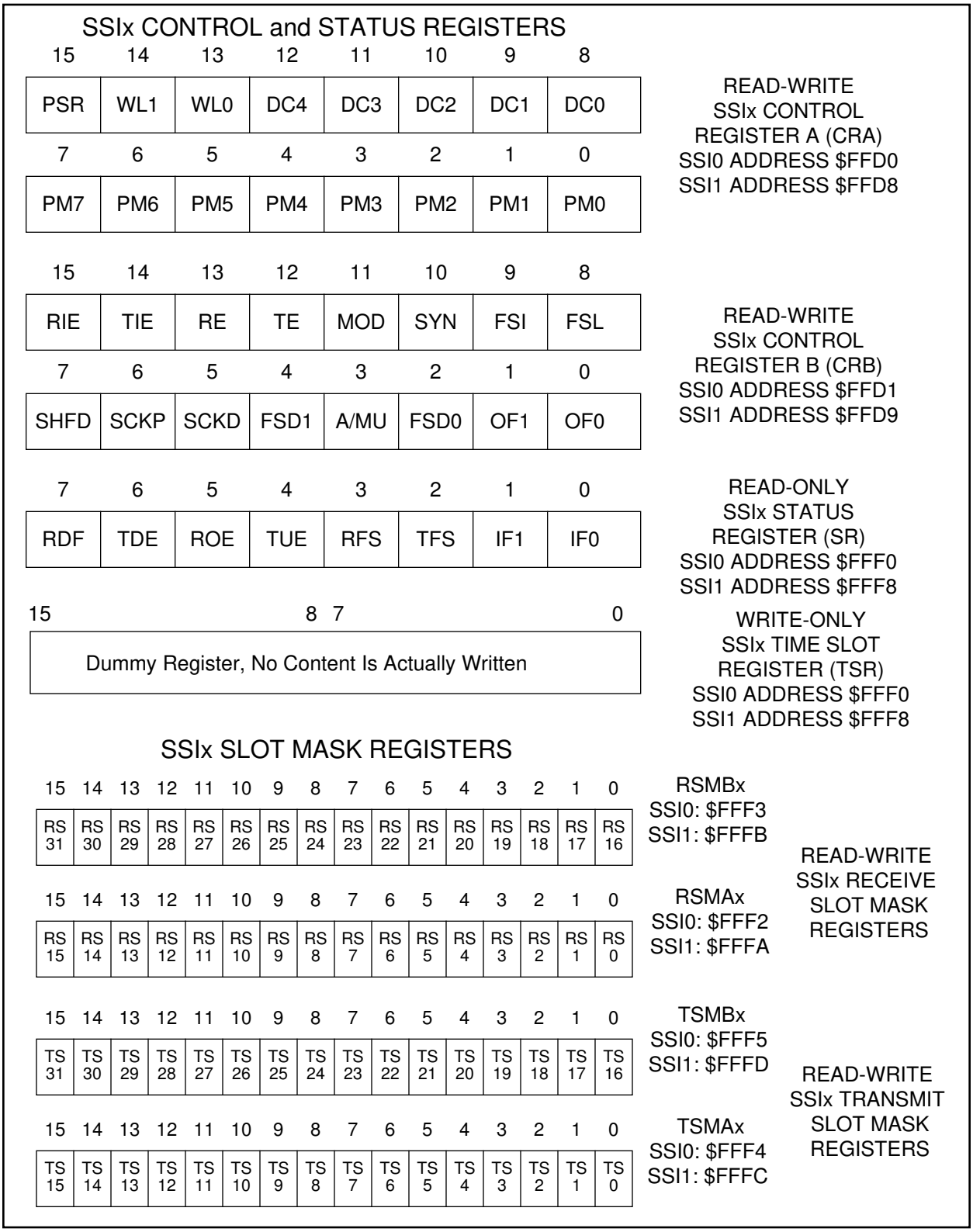


Figure 8-8 SSIx Programming Model (Continued)

8.8 SSI 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 when it becomes empty. 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. Tx is memory mapped to X:$FFF1 for SSI0 and X:$FFF9 for SSI1.

Note: 1. When FSL=1, if the data is written into TX just between the frame sync and the transmission of the first bit, the data will not be transmitted. TDE and TUE will be set when the first bit is transmitted.

2. When the A/MU law is enabled, the data to be transmitted during the first enabled slot of a frame should be written to the TX register before the second to last bit of the last slot of the previous frame. Otherwise a transmit underrun error occurs.

8.9 SSI RECEIVE SHIFT REGISTER

This is a 16-bit shift register that receives the incoming data from the serial receive data (SRD) pin. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync input/output is asserted. Data is assumed to be received most significant bit (MSB) first if the SHFD bit of CRB is cleared. If the SHFD bit is set, the data is assumed to be received least significant bit first. Data is transferred to the SSI Receive Data Register after 8, 12, or 16 bits have been shifted in depending on the Word Length control bits (WL1-WL0) in SSI Control Register A. The receive shift register cannot be directly accessed by the programmer.

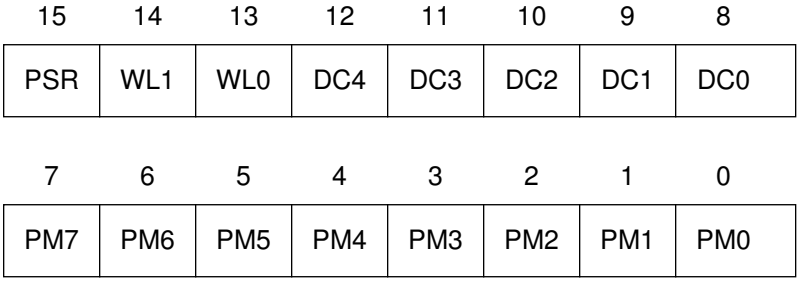
8.10 SSI RECEIVE DATA REGISTER (RX)

The SSI 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. Rx is memory mapped to X:$FFF1 for SSI0 and X:$FFF9 for SSI1.

8.11 SSI CONTROL REGISTER A (CRA)

The SSI Control Register A is one of two 16-bit read/write control registers used to direct the operation of the SSI. The CRA controls the SSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The DSP reset

clears all CRA bits. SSIx reset and STOP reset do not affect the CRA bits. CRA is memory mapped to X:$FFD0 for SSI0 and X:$FFD8 for SSI1. The CRA control bits are described in the following paragraphs.



15	14	13	12	11	10	9	8
PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0

7	6	5	4	3	2	1	0
PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

READ-WRITE
SSIx CONTROL
REGISTER A (CRA);
ADDRESS $FFD0 — SSI0
ADDRESS $FFD8 — SSI1

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

The Prescale Modulus Select bits specify the divide ratio of the prescale divider in the SSI clock generator. This prescaler is used only in internal clock mode to divide the internal clock of the DSP. A divide ratio from 1 to 256 (PM=$00 to $FF) may be selected.The bit clock output is available at SCK. The bit clock output is also used internally 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 PM0-PM7 values that can be used in order to generate different bit clocks:

The bit clock on the SSI 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 (DC0…DC4) Bits 8-12

The Frame Rate Divider Control bits control the divide ratio for the programmable frame rate dividers. It operates on the word clock. In network mode this ratio may be interpreted as the number of words per frame.

In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (DC = 00000 to 11111) for normal mode and 2 to 32 (DC = 00001 to 11111) for network mode.

Table 8-2 SSI Bit Clock as a Function of Fosc and PM0-PM7 (PSR=0)

Fosc (MHz)	Max bit Clock (MHz)	PM0-PM7 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	-	-	-
65.536	16.384	7	-	-	127($7F)	255($FF)
73.728	18.432	8	-	-	143($8F)	-

Examples:

in 8-bit word network mode: DC0-DC4= 26, PM0-PM7=8, PSR=0, Fosc = 62.22MHz would give a bit clock of 62.22Mhz ÷ [4x9] = 1.728 kHz for a 27 slot TDM multiplex of 8-bit words. The sampling rate for every word (FS rate) would then be 1.728 kHz ÷ [27x8] = 8kHz.

in 8-bit word normal mode: DC0-DC4= 1, PM0-PM7=9, PSR=1, Fosc = 40.96MHz would give a bit clock of 40.96Mhz ÷ [8x4x10] = 128 kHz. The 8-bit word rate being equal to 2, the sampling rate (FS rate) would then be 128 kHz ÷ [2x8] = 8kHz.

A divide ratio of one (DC=00000) in network mode is a special case. In normal mode, a divide ratio of one (DC=00000) provides continuous periodic data word transfer. Note that a 1-bit sync (FSL=1) must be used in this case.

Note: The frame divider control bits have to be written before the first three serial clock cycles of the last slot of a frame in order to become active at the beginning of the next frame.

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

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

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

When WL1-WL0= 01, the received logarithmic byte is automatically expanded to a 13 or 14 bit word in the RX register. The result is left justified in the RX register. When transmitting, a 16-bit left justified word written to the TX register will be truncated to 13/14 bits before logarithmic compression. The A/MU bit 3 of CRB will select which compression law, A or MU, is used.

Note: When the A/MU law is enabled, the data to be transmitted should be written to the TX register before the second to last clock of the previous slot. Otherwise a transmit underrun error occurs.

Table 8-3 SSI Data Word Lengths

WL1	WL0	Number of bits/word
0	0	8
0	1	8 with log exp/comp
1	0	12
1	1	16

8.11.4 CRA Prescaler Range (PSR) Bit 15

The Prescaler Range controls a fixed divide-by-8 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-8 prescaler is operational. This allows a 128kHz master clock to be generated for Motorola codecs. The maximum internally generated bit clock frequency is Fosc/4 and the minimum internally generated bit clock frequency is Fosc/(4*8*256).

8.12 SSI CONTROL REGISTER B (CRB)

The SSI Control Register B is one of two, 16-bit read/write control registers used to direct the operation of the SSI. CRB controls the direction of the bit clock pin (SCK) and the function of the SC1x and SC0x pins. Interrupt enable bits for each data register interrupt are provided in this control register. SSI operating modes are also selected in this register. The DSP reset clears all CRB bits. SSIx reset and STOP reset do not affect the CRB bits. The SSI Control Register B bits are described in the following paragraphs.

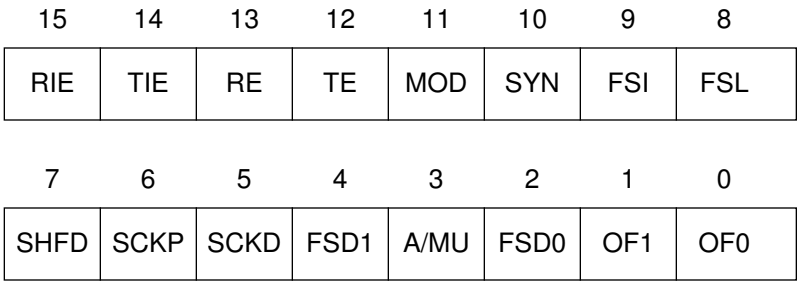


Figure 8-9 SSI Control Register B

8.12.1 CRB Serial Output Flag 0 and 1 (OF0, OF1) Bit 0, 1

When SSI is in the synchronous mode and when the FSD0 and FSD1 bits are set (indicating that pins SC0x and SC1x are used as output flags) data present in OF0 and OF1 will be written to SC0x and SC1x at the beginning of the frame in normal mode or at the beginning of the next valid time slot in network mode.

8.12.2 Transmit and Receive Frame Sync Directions - (FSD0, FSD1) Bit 2,4

The Frame Sync Direction bits (FSD1, FSD0) determine the direction of SC1x and SC0x and whether the frame sync or the flags are used. If FSD0=0 and FSD1=0, then both pins are inputs. If FSD0=1 and FSD1=0, then SC1x is an input and SC0x is an output. If FSD1=1, then both pins are outputs. SC0x and SC1x are both used as frame syncs (SC0x only if SYN=1) and SSISR flag inputs. Output pins reflect either the frame sync (FSD0=0 and FSD1=1) or the flags (FSD0=1, FSD1=1, & SYN = 1). Table 8-4 shows the functions of the two pins SC1x and SC0x according to the definition of CRB flags SYN, FSD1 and FSD0.

Table 8-4 Function of SC1x and SC0x Pins

CRB Flags SYN	FSD0	FSD1	Mode	SC1x	SC0x	Comments
0	0	0	Async	RFS in	TFS in	Illegal for on-demand
0	0	1	Async	RFS out	TFS out	
0	1	0	Async	RFS in	TFS out	
0	1	1	-	-	-	(Reserved)
1	0	0	Sync	-	FS in	Illegal for on-demand
1	0	1	Sync	-	FS out	
1	1	0	-	-	-	(Reserved)
1	1	1	Sync	F1 out	F0 out	Flags used for sync as in Figure 8-5

RFS: Receive Frame Sync; TFS: Transmit Frame Sync; FS: Frame Sync

8.12.3 CRB A/Mu Law Selection Bit (A/MU) Bit 3

When WL1-WL0 control bit of CRA are programmed for 8-bit exchange with logarithmic expansion/compression, the bit A/MU selects which law is used by the expanding/companding hardware. This bit is a don't care bit otherwise.

If A/Mu=0, the A law is selected and if A/MU=1, the μ law is selected. Companding/Expanding hardware follows CCITT recommendation G.711.

8.12.4 Transmit and Receive Frame Sync Directions - (FSD1) Bit 4

See Paragraph 8.12.2.

8.12.5 CRB Clock Source Direction (SCKD) Bit 5

The Clock Source Direction bit 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 SSI clock generator. This clock appears at the SCK pin (SCKD=1). When SCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCK pin and an external clock source may drive this pin to clock the Transmit Shift Register and the Receive Shift Register in either mode.

8.12.6 CRB Clock Polarity Bit (SCKP) Bit 6

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

8.12.7 CRB MSB Position Bit (SHFD) Bit 7

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

8.12.8 CRB Frame Sync Length (FSL) Bit 8

The Frame Sync Length bit selects the type of frame sync to be generated or recognized. If FSL=1, the frame sync will be one bit-clock long during the bit period immediately preceding the first bit period of the word being transferred. If FSL=0, the frame sync will be one data word in length.

8.12.9 CRB Frame Sync Invert (FSI) Bit 9

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

8.12.10 CRB Sync/Async (SYN) Bit 10

The Sync/Async control bit controls whether receive and transmit functions of the SSI occur synchronously or asynchronously with respect to each other. When SYN is cleared, asynchronous mode is chosen and separate frame sync signals are used for the transmit and receive sections. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals.

8.12.11 CRB SSI Mode Select (MOD) Bit 11

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

8.12.12 CRB SSI Transmit Enable (TE) Bit 12

The SSI Transmit Enable bit enables the transfer of data from the TX register to the Transmit Shift Register. When TE is set and a frame sync is detected, the transmit portion of the SSI is enabled. When TE is cleared, the transmitter will be disabled after completing transmission of data currently in the SSI Transmit Shift Register.The serial output is then 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 get cleared but data will not be transferred to the Transmit Shift Register.

Normal transmit enable sequence for transmit is to write data to TX or to TSR **before** setting TE. Normal transmit disable sequence is to clear TE and TIE **after** TDE=1.

In the network mode, the operation of clearing TE and setting it again will disable the transmitter after completion of transmission of a current data word until the beginning of a new data frame period. During the disabled time period, the STD pin will remain in three-state.

Note: TE does not inhibit TDE or transmitter interrupts. TE does not affect the generation of frame sync.

8.12.13 CRB SSI Receive Enable (RE) Bit 13

When the SSI Receive Enable bit is set, the receive portion of the SSI is enabled. When this bit is cleared, the receiver will be disabled by inhibiting data transfer into RX. If data is being received while this bit is cleared, the rest of the word will be shifted in and transferred to the SSI Receive Data Register.

In network mode, the operation of clearing RE and setting it again will disable the receiver after reception of the current data word until the beginning of the new data frame.

Note: RE does not inhibit RDF or receiver interrupts. RE does not affect the generation of a frame sync.

8.12.14 CRB SSI Transmit Interrupt Enable (TIE) Bit 14

When the SSI Transmit Interrupt Enable bit is set, the program controller will be interrupted when the SSI Transmit Data Register Empty flag (TDE) in the SSI 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 with the TE bit. Writing data to the TX or TSR register will clear TDE thus clearing the interrupt.

There are two transmit data interrupts which 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

8.12.15 CRB SSI Receive Interrupt Enable (RIE) Bit 15

When the SSI Receive Interrupt Enable bit is set, the program controller will be interrupted when the SSI Receive Data Register Full flag (RDF) in the SSI 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 SSI STATUS REGISTER (SSISR)

The SSI Status Register is an 8-bit read only status register used by the DSP to interrogate the status and serial input flags of the SSI. The status bits are described in the following paragraphs.

7	6	5	4	3	2	1	0
RDF	TDE	ROE	TUE	RFS	TFS	IF1	IF0

READ-ONLY
SSIx STATUS
REGISTER (SR)
SSI0 ADDRESS $FFF0
SSI1 ADDRESS $FFF8

Note: All the flags in the SSISR are updated one SSI clock after the beginning of a time slot.

8.13.1 SSISR Serial Input Flag 1 and 0 (IF0, IF1) Bit 0, 1

The SSI always latches data present on the SC1x and SC0x pins during reception of the first received bit. IF0 and IF1 are always updated with this data when the receive shift register is transferred into the receive data register. Hardware, software, SSI individual, and STOP resets will clear IF0 and IF1.

8.13.2 SSISR Transmit Frame Sync (TFS) Bit 2

When set the Transmit Frame Sync flag (TFS) indicates that the first bit of the first active word of the frame has been transmitted. Data written to the transmit data register when TFS is set will be transmitted (in network mode) during the second active time slot in the frame.

In network mode, TFS is set during transmission of the first active slot of the frame. It will then be cleared when starting transmission of the next active slot of the frame. If the first time slot of the frame is not enabled, this bit will be set on the leading edge of the first active slot.

Note: In normal mode or in network mode with only one enabled slot, TFS will always read as a one when transmitting data because there is only one time slot per frame.

TFS is cleared by DSP, SSIx or STOP reset and is not affected by TE.

8.13.3 SSISR Receive Frame Sync (RFS) Bit 3

When set, the Receive Frame Sync flag (RFS) indicates that reception of the word in the Serial Receive Data Register (RX) occurred during the first active slot in the frame. When RFS is cleared and a word is received, it indicates (only in the network mode) that the reception of that word did not occur during the first active slot.

In network mode, if the first slot is disabled, this flag will be set after reception in the first active slot and will only be cleared after receiving the next enabled time slot into the RX register.

Note: In normal mode or network mode with one enabled slot, RFS will always be set when reading data because data is received only in the first active time slot.

RFS is cleared by DSP, SSIx or STOP reset and is not affected by RE.

8.13.4 SSISR Transmitter Underrun Error (TUE) Bit 4

The Transmitter Underrun Error 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 active time slot requires transmit data.

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 and, if a transmit interrupt occurs with TUE clear, the Transmit Data Without Errors interrupt will be generated.

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

8.13.5 SSISR Receiver Overrun Error (ROE) Bit 5

The Receiver Overrun Error flag 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 the DSP, SSIx, or STOP reset, and is cleared by reading the SSISR with ROE set followed by reading the RX. Clearing RE does not affect ROE.

8.13.6 SSISR Transmit Data Register Empty (TDE) Bit 6

The SSI Transmit Data Register Empty flag 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 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 SSI Transmit Data interrupt request will be issued when TDE is set. The interrupt vector will depend on the state of the Transmitter Underrun TUE bit. TDE is set by the DSP, SSIx, and STOP reset.

8.13.7 SSISR Receive Data Register Full (RDF) Bit 7

The SSI Receive Data Register Full flag is set when the contents of the Receive Shift Register are transferred to the Receive Data Register. When set, RDF indicates that data should be read from RX so that the next word can be received without an overrun error.

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 vector request will depend on the state of the Receiver Overrun ROE bit. RDF is cleared by the DSP, SSIx, and STOP reset.

8.14 TIME SLOT REGISTER - TSR

The Time Slot Register is used when the data is not to be transmitted (i.e., a blank time slot) 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 for that time slot.

8.15 TRANSMIT SLOT MASK REGISTERS - TSMAx AND TSMBx

The Transmit Slot Mask Registers are two 16-bit read/write registers. They are used by the transmitter in network mode to determine for each slot whether to transmit a data word and generate a transmitter empty condition (TDE=1), or to three-state the transmit data pin, STD. TSMAx and TSMBx should be seen as only one 32-bit register, TSMx. Bit number N in TSMx is an enable/disable control bit for transmission in slot number N.

When bit number N in TSMx is cleared, the transmit data pin STD is three-stated during transmit time slot number N. The data is still transferred from the Transmit Data Register to the transmit shift register and the Transmitter Data Empty flag (TDE) is set. Also the Transmitter Underrun Error flag is not set. This means that during a disabled slot, no Transmitter Empty interrupt is generated (TDE=0). The DSP is interrupted by activity in enabled slots only. Data that is written to the Transmit Data Register when servicing this request is transmitted in the next enabled transmit time slot.

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

The 32-bit data written into TSMx is transferred to the Transmit Slot Mask Shift Register (TSMS) during the last slot of the last frame. Writing to TSMx will not affect the state of the active (enabled) transmit slots during the current transmit frame but only that of the next frame.

Using the slot mask in TSMx does not conflict with using TSR. Even if a slot is enabled in TSMx, the user may choose to write to TSR instead of writing to the transmit data register TX. This will cause the transmit data pin to be three-stated during the next slot.

After DSP reset, the Transmit Slot Mask Register is preset to $FFFFFFFF, which means that all 32 possible slots are enabled for data transmission.

Note: The Transmit Slot Mask Registers have to be written before the last three serial clock cycles of a frame in order to become active at the beginning of the next frame.

8.16 TRANSMIT SLOT MASK SHIFT REGISTER - TSMS

The Transmit Slot Mask Shift Register is a 32-bit shift register. At the end of each frame, it is loaded with the 32-bit mask from TSM. Then, it is shifted one bit to the right near the end of each transmitted word. The LSB of TSMS is used as an enable/disable for transmitting data to the transmit data pin (STD) and setting the TDE flag after transmitting data. This register is not accessible to the programmer.

8.17 RECEIVE SLOT MASK REGISTERS - RSMAx AND RSMBx

The Receive Slot Mask Registers are two 16-bit read/write registers. They are used by the receiver in network mode to determine for each slot whether to receive a data word and generate a receiver full condition (RDF=1), or to ignore the received data. RSMAx and RSMBx should be seen as only one 32-bit register RSMx. Bit number N in RSMx is an enable/disable control bit for receiving data in slot number N.

When bit number N in RSMx is cleared, the data from the receive data pin is shifted into the Receive Shift Register during slot number N. Data is not transferred from the Receive Shift Register to the Receive Data Register and the Receiver Full flag (RDF) is not set. Also the Receiver Overrun Error flag is not set. This means that during a disabled slot, no Receiver Full interrupt is generated. The DSP is interrupted (RDF=0). The DSP is interrupted by activity in enabled slots only.

When bit number N in RSMx is set, the receive sequence is as usual: data which is shifted into the receive shift register is transferred to the Receive Data register and the RDF flag is set.

The 32-bit mask which is written into RSMx, is transferred to the Receive Slot Mask Shift Register (RSMS) during the last slot of the last frame. Writing to RSMx will not affect the state of the active (enabled) receive slots during the current receive frame, but only that of the next frame.

After DSP reset, the Receive Slot Mask Register is preset to $FFFFFFFF, which means that all 32 possible slots are enabled for data reception.

Note: The Receive Slot Mask Registers have to be written before the last three serial clock cycles of a frame in order to become active at the beginning of the next frame.

8.18 RECEIVE SLOT MASK SHIFT REGISTER - RSMS

The Receive Slot Mask Shift Register (RSMS) is a 32-bit shift register. At the beginning of each receive frame, it is loaded with the 32-bit mask from RSM. Then, it is shifted one bit to the right near the end of each received word. The LSB of RSMSR is used as an enable/disable for receiving a word and setting the RDF flag. This register is not accessible to the programmer.

8.19 SSI OPERATING MODES

The SSI on-chip peripheral can operate in three operating modes – Normal, Network, and On-Demand. In the Normal and Network modes, data is transferred and interrupts are generated (if enabled) periodically. The On-Demand mode is a special mode which permits data to be transferred serially when necessary in a non-periodic fashion.

Table 8-5 SSI modes

Mode	SYN	SC1x	SC0x
Normal	Async	RFS In RFS Out RFS In	TFS In TFS Out TFS Out
	Sync	— —	FS In FS Out
Network	Async	RFS In RFS Out RFS In	TFS In TFS Out TFS Out
	Sync	— —	FS In FS Out
On-Demand	Async	RFS Out RFS In	TFS Out TFS Out
	Sync	—	FS Out

8.19.1 Normal Operating Mode

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.

8.19.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI are:

1. Transmitter Enabled (TE=1)
2. Frame sync becomes active

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. The optional frame sync output and clock outputs are not three-stated even if both receiver and transmitter are disabled.

The program must write another word to TX before the next frame sync is received or TUE will be set and a Transmitter Underrun Error will be generated.

8.19.1.2 Normal Mode Receive

If the receiver is enabled, then each time the frame sync signal is generated (or detected) a data word will be clocked in. After receiving the data word it will be transferred from the SSI 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).

8.19.2 Network Mode

In this mode the SSI can be used in TDM networks and 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 one or more time slots. The distinction of the network mode is that active time slots (data word time) are identified by the transmit and receive slot mask registers (TSMA0, TSMB0, RSMA0, and RSMB0). For the transmitter, this allows the option of ignoring a time slot or transmitting data during that 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 when the corresponding time slot is enabled. The DSP will read the receive data register and either use or discard the data according to the time slot register and mask.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission or reception can occur in each time slot (rather than in just the frame sync time slot as in the normal mode). The frame rate dividers, controlled by DC4, DC3, DC2, DC1, and DC0 control the number of time slots per frame from 2 to 32.

8.19.2.1 Network Mode Transmit

The transmit portion of the SSI is enabled when TE=1. However, when TE is sets the transmitter will be enabled only after detection of a new data frame sync. This allows the SSI to synchronize to the network timing.

Normal start up sequence for transmission in the first time slot is to write the data to be transmitted to the Transmit Register (TX), this clears the TDE flag. Then set TE and TIE to enable the transmitter on the next frame sync and to enable transmit interrupts.

Alternatively, the DSP programmer may decide **not** to transmit in the first time slot by writing (any data) to the Time Slot Register (TSR). This will clear the TDE flag just as if data were going to be transmitted, but the STD pin will remain in three-state for the first time slot. The programmer then sets TE and TIE as above.

When the frame sync is detected (or generated), 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 (1) poll TDE, or (2) use interrupts to reload the TX register with new data for the next time slot, or (3) write to the TSR to prevent transmitting in the next active time slot. The transmit and receive slot mask registers control which time slots will be used. Failing to reload TX (or writing to TSR) before the next active time slot will cause a transmitter underrun and the TUE error bit will be set.

Clearing TE and setting it again will disable the transmitter after completion of transmission of the current data word until the beginning of the next frame sync period. During that time the STD pin will be three-stated. TE should be cleared after TDE gets set to ensure that all pending data is transmitted.

To summarize, the network mode transmitter generates interrupts every **enabled** time slot (TE=1, TIE=1) and requires the DSP program to respond to each **enabled** time slot. These responses may be:

1. Write TX data register with data for 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 active time slot and the previous data will be transmitted

8.19.2.2 Network Mode Receive

The receiver portion of SSI is enabled when RE is set; however, the receive enable will take place only after detection of a new data frame. After detection of the new data frame, the first data word will be shifted into the Receive Shift Register during the first active slot. When the word is completely received, it is transferred to the RX 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).

During the second active time slot, the second data word begins shifting in. The DSP program must read the data from RX (which clears RDF) before the second data word is completely received (ready to transfer to RX) or a receive overrun error will occur (ROE gets set).

If the RE bit is cleared and set again by the DSP programmer, the receiver, after receiving the current time slot in progress, will be disabled until the next frame sync (first time slot). This mechanism allows the DSP programmer to ignore data in the last portion of a data frame.

Note: The optional frame sync output and clock output signals are not affected even if the transmitter and/or receiver are disabled. TE and RE do not disable bit clock and frame sync generation.

To summarize, in the Network mode, 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 next active time slot

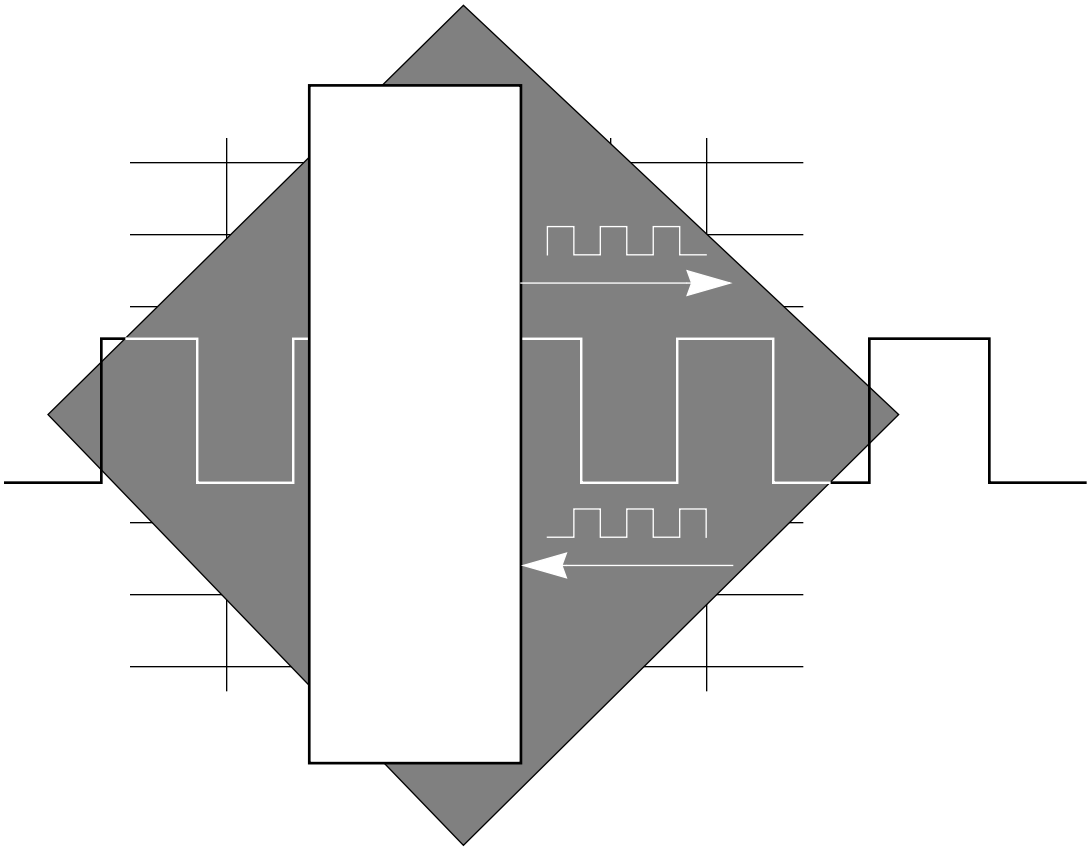
8.19.2.3 On-Demand Mode

A divide ratio of one (DC=00000) in the network mode is a special case. This is the only data driven mode of the SSI and is defined as the on-demand mode. This special case

will not generate a periodic frame sync. A frame sync pulse will be generated only when there is data available to transmit. The On-demand mode requires that the transmit frame sync be internal (output) — i.e., the on-demand mode is disallowed whenever the transmit frame sync is configured as an input. Data transmission is data driven and is enabled by writing data into the TX register. Receive and transmit interrupts function with the TDE and RDF flags as normally; however, transmit underruns (TUE) are impossible for on-demand transmission.

SECTION 8

SYNCHRONOUS SERIAL INTERFACE (SSI0 and SSI1)



SECTION CONTENTS

8.1 INTRODUCTION

The DSP56156 contains two identical Synchronous Serial Interfaces (SSI's) named SSI0 and SSI1. This section describes both. In cases where the text or a figure applies equally to both SSI's, they will be referred to as the SSI. In cases where the information differs between the SSI's such as when pin numbers are mentioned, control addresses are mentioned or the operation affects only one of the two SSI's like a personal reset, the SSI will be referred to as SSIx meaning SSI0 or SSI1 – whichever applies.

The SSI 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, or microprocessors and peripherals. A selectable logarithmic compression and expansion is available for easier interface with PCM monocircuits and PCM highways. The SSI interface consists of independent transmitter and receiver sections and a common SSI clock generator.

8.2 SSI OPERATING MODES

The SSI has several basic operating modes. These modes can be programmed by several bits in the SSI Control registers. The Table 8-1 below lists the SSI operating modes:

Table 8-1 SSI Operating Modes

TX, RX Sections	Serial Clock	Protocol
Asynchronous	Continuous	Normal
Synchronous	Continuous	Normal
Synchronous	Continuous	Network
Asynchronous	Continuous	Network

The transmit and receive sections of this interface may be synchronous or asynchronous; that is, the transmitter and the receiver may use common clock and synchronization signals or they may have independent frame sync signals but the same bit clock. The SYN bit in SSI Control Register B selects synchronous or asynchronous operation. Since the SSI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

Normal or network protocol may also be selected. For normal protocol the SSI functions with one data word of I/O per frame. For network protocol, 2 to 32 data words of I/O may be used per frame. Network mode is used in Time Division Multiplexed (TDM) networks of codecs or DSPs. These distinctions result in the basic operating modes which allow the SSI to communicate with a wide variety of devices.

8.3 SSI 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 SSI clock generator is used to derive bit clock and frame sync signals from the DSP internal system clock. The SSI clock generator consists of a selectable fixed prescaler and a programmable prescaler for bit rate clock generation and also a programmable frame rate divider and a word length divider for frame rate sync signal generation.

8.4 SSIx DATA AND CONTROL PINS

The SSIx has five dedicated I/O pins for each SSI:

- Transmit data STDx (PC0 for SSI0 and PC5 for SSI1)
- Receive data SRDx (PC1 for SSI0 and PC6 for SSI1)
- Serial clock SCKx (PC2 for SSI0 and PC7 for SSI1)
- Serial Control Pin 1 SC1x (PC3 for SSI0 and PC8 for SSI1)
- Serial Control Pin 0 SC0x (PC4 for SSI0 and PC9 for SSI1)

Figure 8-1 through Figure 8-5 show the main configurations and the following paragraphs describe the uses of these pins for each of the SSIx operating modes.These figures do not represent all possible configurations, e.g., SCKx and FS don't have to be in the same direction. **Note that the first pin name in these figures apply to SSI0 and the second applies to SSI1 i.e., PC2/PC7 means PC2 for SSI0 and PC7 for SSI1.**

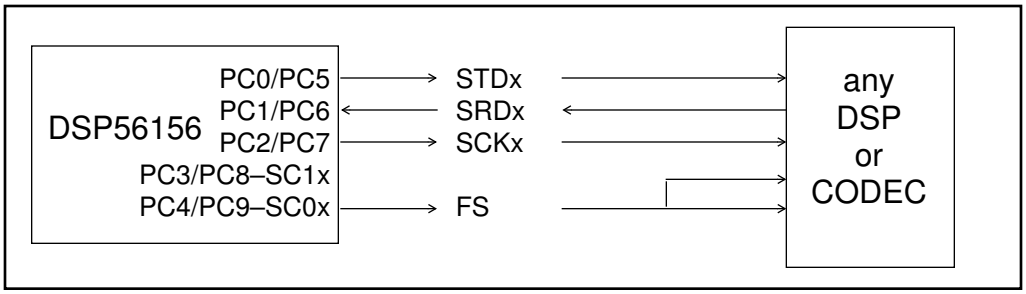


Figure 8-1 SSIx Internal Clock, Synchronous Operation

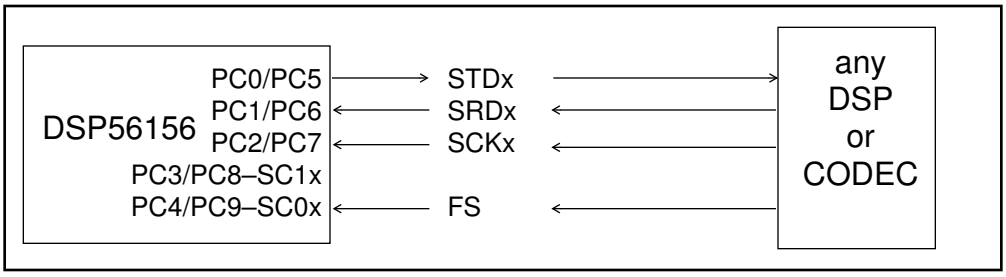


Figure 8-2 SSIx External Clock, Synchronous Operation

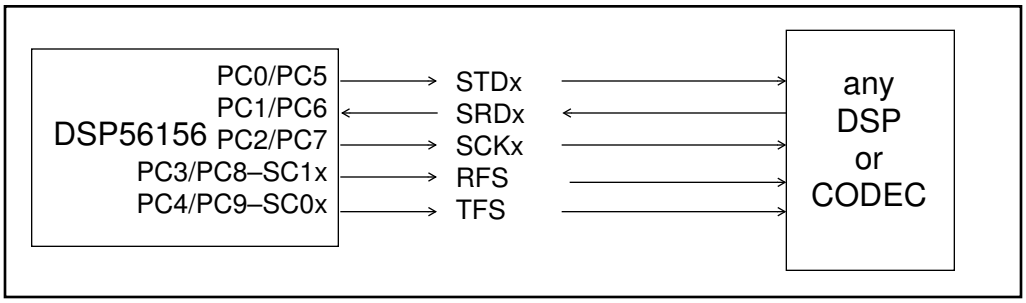


Figure 8-3 SSIx Internal Clock, Asynchronous Operation

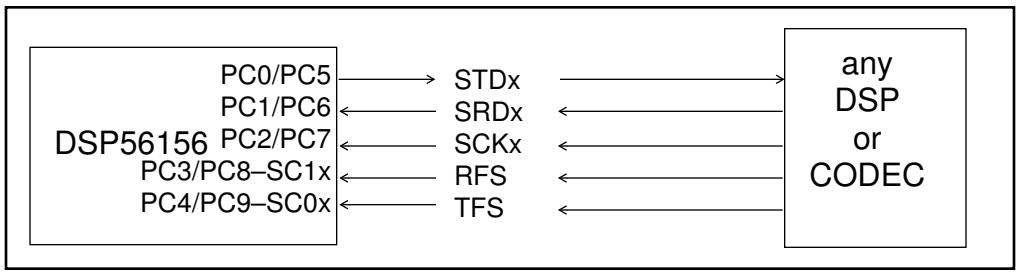


Figure 8-4 SSIx External Clock, Asynchronous Operation

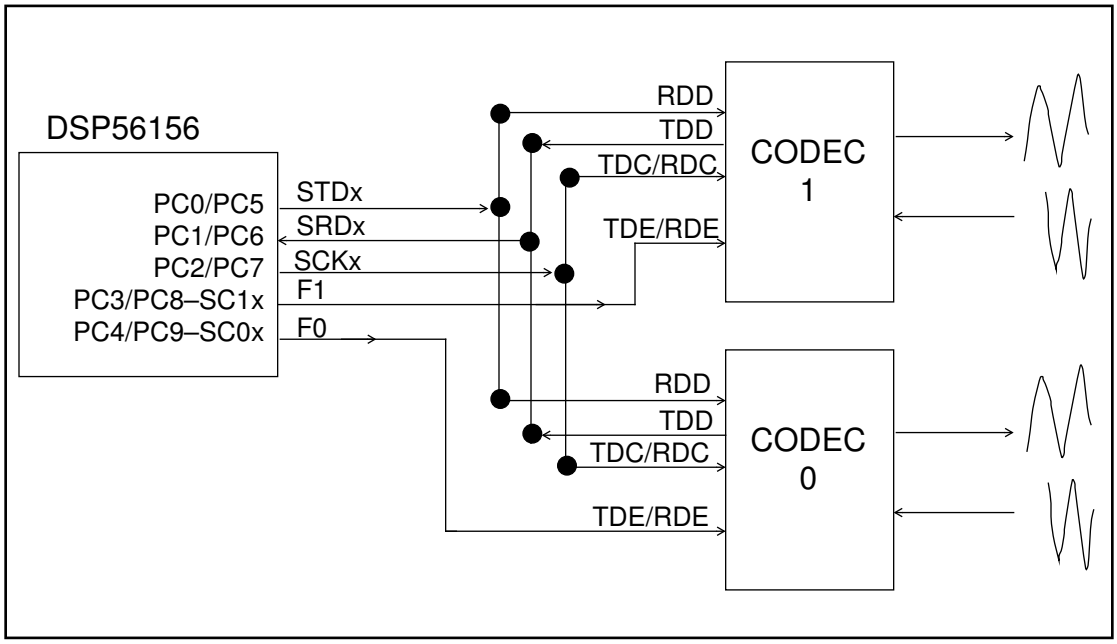


Figure 8-5 SSIx Internal Clock, Synchronous Operation Dual Codec Interface

Figure 8-6 shows the internal clock path connections in block diagram form. The serial bit clock can be internal or external depending on SCKD bit in the control register.

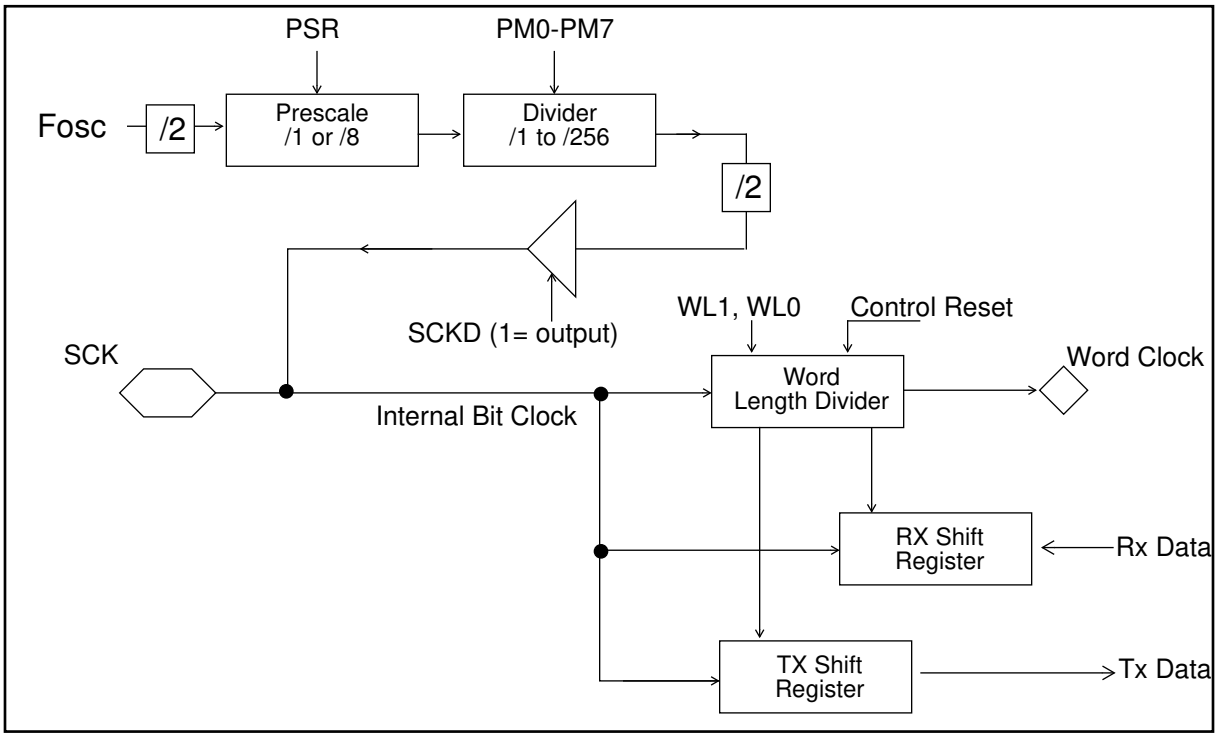


Figure 8-6 SSI Clock Generator Functional Block Diagram

Table 8-4 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 (DC4-DC0) bit and the word length (WL1-WL0) bits of CRA.

Figure 8-7 shows the functions of the two pins SC1x and SC0x according to the setting of CRB flags.

8.4.1 Serial Transmit Data Pin - STDx

The Serial Transmit Data Pin (STD) is used for 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 after the trailing edge of the bit clock after the last bit of a word is transmitted.

8.4.2 Serial Receive Data Pin - SRDx

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

8.4.3 Serial Clock - SCK

The Serial Clock (SCK) pin is used as a clock input or output used by both the transmitter and receiver in synchronous modes and in asynchronous modes.

8.4.4 Serial Control - SC1x

The function of this pin is determined by the flags SYNC, FSD0 and FSD1 of control register B (CRB) — see Table 8-4. In Asynchronous mode (SYNC=0), this pin is the receiver frame sync I/O. For synchronous mode (SYNC=1), with bits FSD0 and FSD1 set, pin SCIx is used as an output flag. When SC1x is configured as an output flag (FSD0=1; FSD0=1), this pin is controlled by bit OF1 in CRB. When SCIx is configured as an input or output (with synchronous or asynchronous operations), this pin will update status bit IF1 of the SSI status register as described in Section 8.13.1.

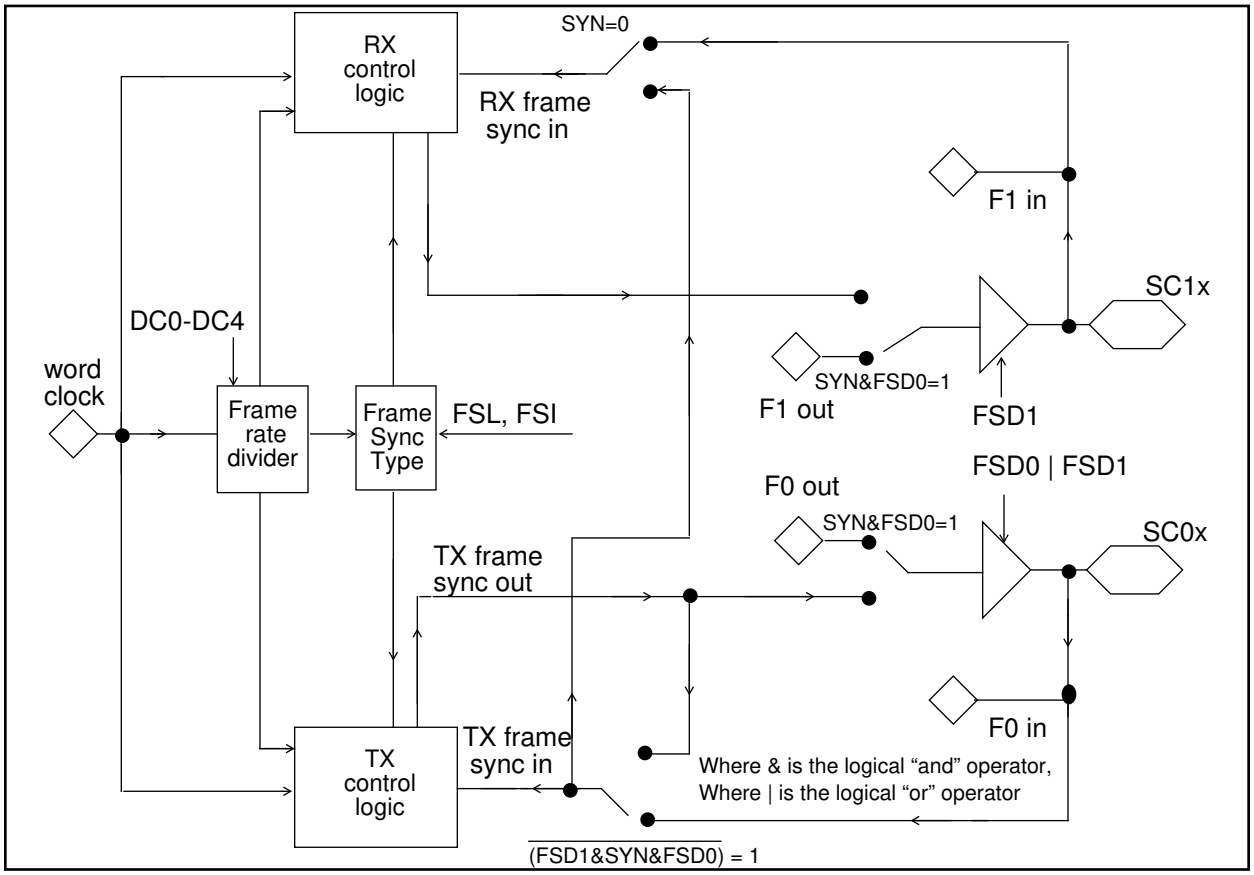


Figure 8-7 SSIx Frame Sync Generator Functional Block Diagram

8.4.5 Serial Control - SC0x

The function of this pin is determined by the SYNC, FSD0 and FSD1 flags in the CRB. In Asynchronous mode (SYNC=0), this pin is the transmitter frame sync I/O. For synchronous mode (SYNC=1), this pin is used as a frame sync I/O if bit FSD0 is cleared. If bits FSD0 and FSD1 are set, this pin is used as an output flag. When configured as an output flag (FSD0=1; FSD1=1), this pin is controlled by bit OF0 in the CRB and changes synchronously with the first transmitted bit of the data. Control status bit IF0 of the SSI status register will update regardless of the SYNC, FSD0 or FSD1 bits in the CRB.

8.5 SSI RESET AND INITIALIZATION PROCEDURE

The SSI is affected by three types of reset:

DSP Reset This reset is generated by either the DSP hardware reset (generated by asserting the $\overline{\text{RESET}}$ pin) or software reset (generated by executing the RESET instruction). The DSP reset clears the Port Control Register bits,

which configures all I/O pins as general purpose input. The SSI will remain in the reset state while all SSI pins are programmed as general purpose I/O (CC0-CC4/CC5-CC9 cleared) and will become active only when at least one of the SSIx I/O pins is programmed as NOT general purpose I/O. All status and control bits in the SSIx are affected as described below.

SSIx Reset The SSIx personal reset is generated when CC0-CC4/CC5-CC9 bits are cleared. This returns the SSIx pins to general purpose I/O pins. The SSIx status bits are preset to the same state produced by the DSP reset; however, the SSIx control bits are unaffected. The SSIx personal reset is useful for selective reset of the SSIx interface without changing the present SSIx control bits setup 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 SSI status bits are preset to the same state produced by the DSP reset. The SSI control bits are unaffected. The SSI pins remain defined as SSIx pins. The STOP reset condition is like the personal reset condition except that the SSI pins do not revert to general purpose I/O pins.

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

1. DSP reset or SSIx reset
2. Program SSIx control registers
3. Configure SSIx pins (at least one) as not general purpose I/O

The DSP programmer should use the DSP or SSIx reset before changing the MOD, SYN, FSI, FSL, MSB, SCKP, SCKD, FSD1, A/MU, FSD0 control bits to ensure proper operation of the SSIx interface. That is, these control bits should not be changed during SSIx operation.

Note: The SSIx clock must go low for at least four complete periods to ensure proper SSIx reset.

8.6 SSIx INTERFACE PROGRAMMING MODEL

The registers comprising the SSI interface are shown in Figure 8-8. Note that standard Codec devices label the Most Significant Bit as bit 0, whereas the DSP labels the LSB as bit 0. Therefore, when using a standard Codec (requiring LSB first), the SHFD bit in the CRB should be cleared (MSB first).

8.7 SSI TRANSMIT SHIFT REGISTER

This is a 16-bit shift register that contains the data being transmitted. Data is shifted out to the serial transmit data STD pin by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. The number of bits shifted out before the shift register is considered empty and may be written to again can be 8, 12, or 16 bits as determined by the Word Length control bits in the SSI Control Register A (WL1-WL0).

The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is shifted out of this register with the most significant bit (MSB) first when the SHFD bit of the control register B is cleared. If the SHFD bit is set, the LSB is output first. The Transmit Shift Register cannot be directly accessed by the programmer.

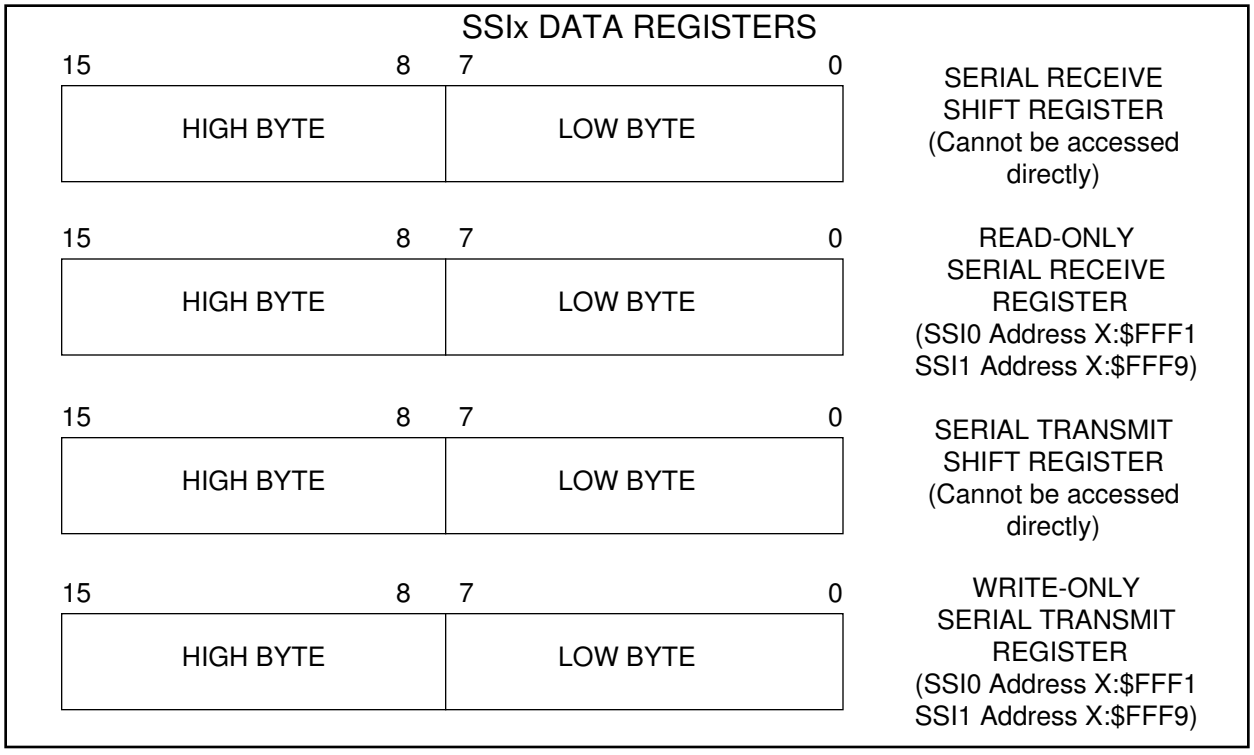


Figure 8-8 SSIx Programming Model

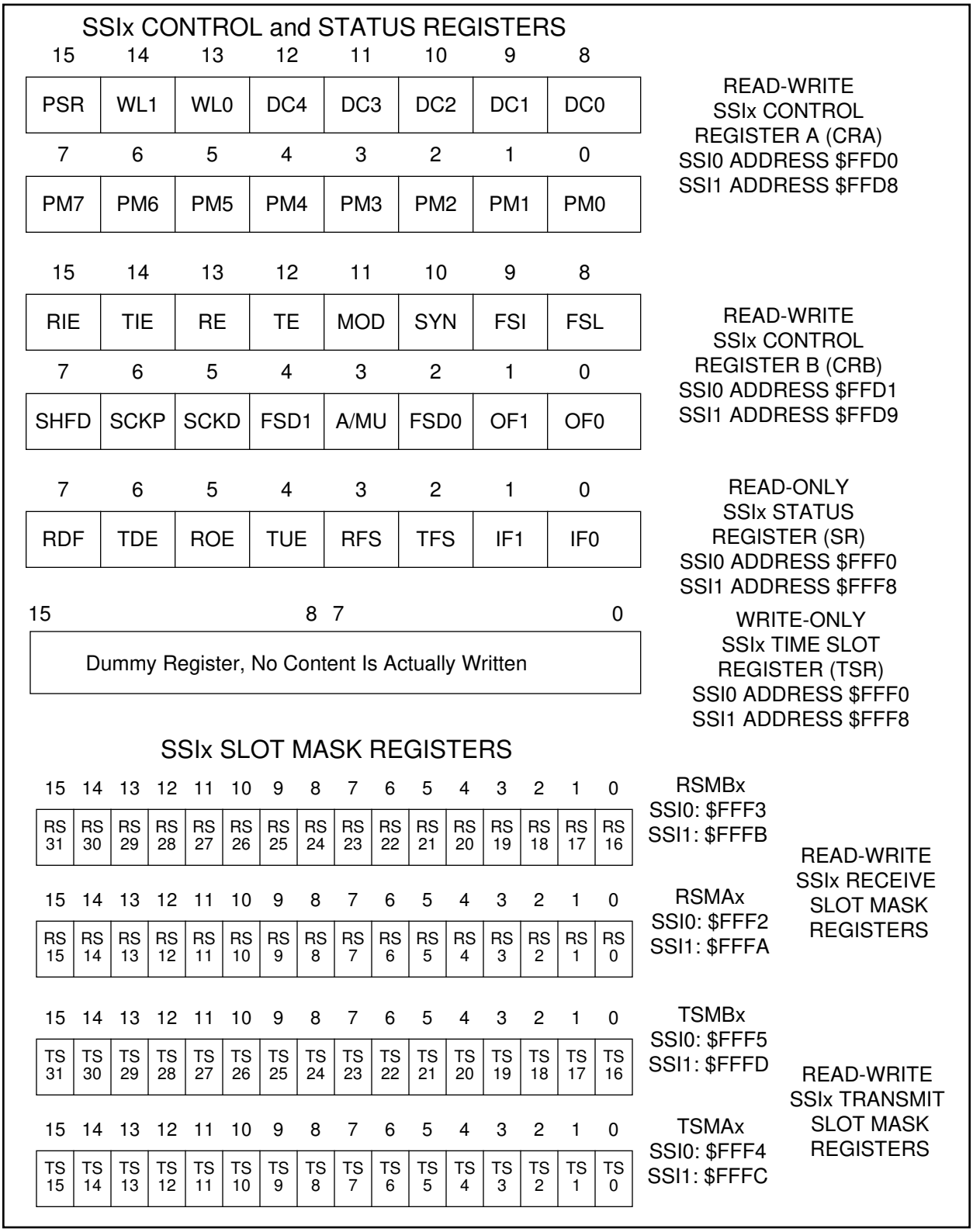


Figure 8-8 SSIx Programming Model (Continued)

8.8 SSI 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 when it becomes empty. 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. Tx is memory mapped to X:$FFF1 for SSI0 and X:$FFF9 for SSI1.

Note: 1. When FSL=1, if the data is written into TX just between the frame sync and the transmission of the first bit, the data will not be transmitted. TDE and TUE will be set when the first bit is transmitted.

2. When the A/MU law is enabled, the data to be transmitted during the first enabled slot of a frame should be written to the TX register before the second to last bit of the last slot of the previous frame. Otherwise a transmit underrun error occurs.

8.9 SSI RECEIVE SHIFT REGISTER

This is a 16-bit shift register that receives the incoming data from the serial receive data (SRD) pin. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync input/output is asserted. Data is assumed to be received most significant bit (MSB) first if the SHFD bit of CRB is cleared. If the SHFD bit is set, the data is assumed to be received least significant bit first. Data is transferred to the SSI Receive Data Register after 8, 12, or 16 bits have been shifted in depending on the Word Length control bits (WL1-WL0) in SSI Control Register A. The receive shift register cannot be directly accessed by the programmer.

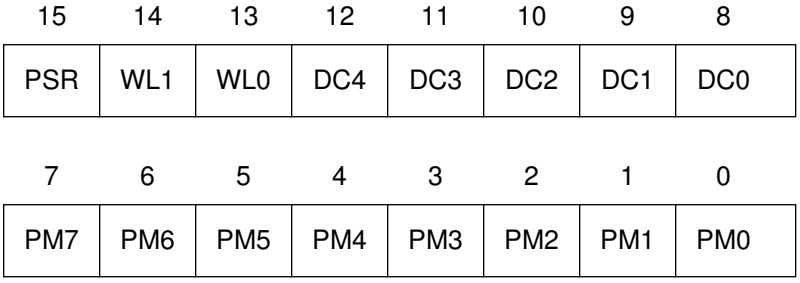
8.10 SSI RECEIVE DATA REGISTER (RX)

The SSI 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. Rx is memory mapped to X:$FFF1 for SSI0 and X:$FFF9 for SSI1.

8.11 SSI CONTROL REGISTER A (CRA)

The SSI Control Register A is one of two 16-bit read/write control registers used to direct the operation of the SSI. The CRA controls the SSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The DSP reset

clears all CRA bits. SSIx reset and STOP reset do not affect the CRA bits. CRA is memory mapped to X:$FFD0 for SSI0 and X:$FFD8 for SSI1. The CRA control bits are described in the following paragraphs.



15	14	13	12	11	10	9	8
PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0

7	6	5	4	3	2	1	0
PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

READ-WRITE
SSIx CONTROL
REGISTER A (CRA);
ADDRESS $FFD0 — SSI0
ADDRESS $FFD8 — SSI1

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

The Prescale Modulus Select bits specify the divide ratio of the prescale divider in the SSI clock generator. This prescaler is used only in internal clock mode to divide the internal clock of the DSP. A divide ratio from 1 to 256 (PM=$00 to $FF) may be selected.The bit clock output is available at SCK. The bit clock output is also used internally 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 PM0-PM7 values that can be used in order to generate different bit clocks:

The bit clock on the SSI 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 (DC0…DC4) Bits 8-12

The Frame Rate Divider Control bits control the divide ratio for the programmable frame rate dividers. It operates on the word clock. In network mode this ratio may be interpreted as the number of words per frame.

In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (DC = 00000 to 11111) for normal mode and 2 to 32 (DC = 00001 to 11111) for network mode.

Table 8-2 SSI Bit Clock as a Function of Fosc and PM0-PM7 (PSR=0)

Fosc (MHz)	Max bit Clock (MHz)	PM0-PM7 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	-	-	-
65.536	16.384	7	-	-	127($7F)	255($FF)
73.728	18.432	8	-	-	143($8F)	-

Examples:

in 8-bit word network mode: DC0-DC4= 26, PM0-PM7=8, PSR=0, Fosc = 62.22MHz would give a bit clock of 62.22Mhz ÷ [4x9] = 1.728 kHz for a 27 slot TDM multiplex of 8-bit words. The sampling rate for every word (FS rate) would then be 1.728 kHz ÷ [27x8] = 8kHz.

in 8-bit word normal mode: DC0-DC4= 1, PM0-PM7=9, PSR=1, Fosc = 40.96MHz would give a bit clock of 40.96Mhz ÷ [8x4x10] = 128 kHz. The 8-bit word rate being equal to 2, the sampling rate (FS rate) would then be 128 kHz ÷ [2x8] = 8kHz.

A divide ratio of one (DC=00000) in network mode is a special case. In normal mode, a divide ratio of one (DC=00000) provides continuous periodic data word transfer. Note that a 1-bit sync (FSL=1) must be used in this case.

Note: The frame divider control bits have to be written before the first three serial clock cycles of the last slot of a frame in order to become active at the beginning of the next frame.

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

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

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

When WL1-WL0= 01, the received logarithmic byte is automatically expanded to a 13 or 14 bit word in the RX register. The result is left justified in the RX register. When transmitting, a 16-bit left justified word written to the TX register will be truncated to 13/14 bits before logarithmic compression. The A/MU bit 3 of CRB will select which compression law, A or MU, is used.

Note: When the A/MU law is enabled, the data to be transmitted should be written to the TX register before the second to last clock of the previous slot. Otherwise a transmit underrun error occurs.

Table 8-3 SSI Data Word Lengths

WL1	WL0	Number of bits/word
0	0	8
0	1	8 with log exp/comp
1	0	12
1	1	16

8.11.4 CRA Prescaler Range (PSR) Bit 15

The Prescaler Range controls a fixed divide-by-8 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-8 prescaler is operational. This allows a 128kHz master clock to be generated for Motorola codecs. The maximum internally generated bit clock frequency is Fosc/4 and the minimum internally generated bit clock frequency is Fosc/(4*8*256).

8.12 SSI CONTROL REGISTER B (CRB)

The SSI Control Register B is one of two, 16-bit read/write control registers used to direct the operation of the SSI. CRB controls the direction of the bit clock pin (SCK) and the function of the SC1x and SC0x pins. Interrupt enable bits for each data register interrupt are provided in this control register. SSI operating modes are also selected in this register. The DSP reset clears all CRB bits. SSIx reset and STOP reset do not affect the CRB bits. The SSI Control Register B bits are described in the following paragraphs.

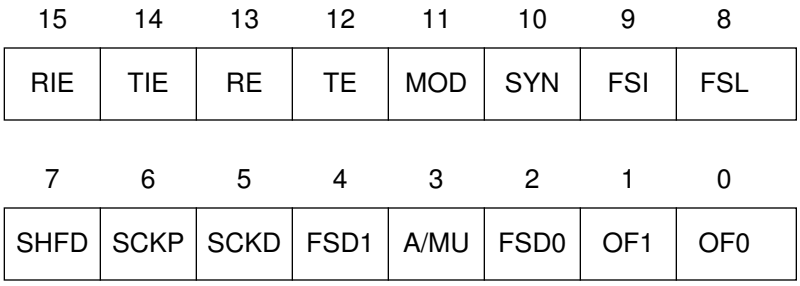


Figure 8-9 SSI Control Register B

8.12.1 CRB Serial Output Flag 0 and 1 (OF0, OF1) Bit 0, 1

When SSI is in the synchronous mode and when the FSD0 and FSD1 bits are set (indicating that pins SC0x and SC1x are used as output flags) data present in OF0 and OF1 will be written to SC0x and SC1x at the beginning of the frame in normal mode or at the beginning of the next valid time slot in network mode.

8.12.2 Transmit and Receive Frame Sync Directions - (FSD0, FSD1) Bit 2,4

The Frame Sync Direction bits (FSD1, FSD0) determine the direction of SC1x and SC0x and whether the frame sync or the flags are used. If FSD0=0 and FSD1=0, then both pins are inputs. If FSD0=1 and FSD1=0, then SC1x is an input and SC0x is an output. If FSD1=1, then both pins are outputs. SC0x and SC1x are both used as frame syncs (SC0x only if SYN=1) and SSISR flag inputs. Output pins reflect either the frame sync (FSD0=0 and FSD1=1) or the flags (FSD0=1, FSD1=1, & SYN = 1). Table 8-4 shows the functions of the two pins SC1x and SC0x according to the definition of CRB flags SYN, FSD1 and FSD0.

Table 8-4 Function of SC1x and SC0x Pins

CRB Flags SYN	FSD0	FSD1	Mode	SC1x	SC0x	Comments
0	0	0	Async	RFS in	TFS in	Illegal for on-demand
0	0	1	Async	RFS out	TFS out	
0	1	0	Async	RFS in	TFS out	
0	1	1	-	-	-	(Reserved)
1	0	0	Sync	-	FS in	Illegal for on-demand
1	0	1	Sync	-	FS out	
1	1	0	-	-	-	(Reserved)
1	1	1	Sync	F1 out	F0 out	Flags used for sync as in Figure 8-5

RFS: Receive Frame Sync; TFS: Transmit Frame Sync; FS: Frame Sync

8.12.3 CRB A/Mu Law Selection Bit (A/MU) Bit 3

When WL1-WL0 control bit of CRA are programmed for 8-bit exchange with logarithmic expansion/compression, the bit A/MU selects which law is used by the expanding/companding hardware. This bit is a don't care bit otherwise.

If A/Mu=0, the A law is selected and if A/MU=1, the μ law is selected. Companding/Expanding hardware follows CCITT recommendation G.711.

8.12.4 Transmit and Receive Frame Sync Directions - (FSD1) Bit 4

See Paragraph 8.12.2.

8.12.5 CRB Clock Source Direction (SCKD) Bit 5

The Clock Source Direction bit 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 SSI clock generator. This clock appears at the SCK pin (SCKD=1). When SCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCK pin and an external clock source may drive this pin to clock the Transmit Shift Register and the Receive Shift Register in either mode.

8.12.6 CRB Clock Polarity Bit (SCKP) Bit 6

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

8.12.7 CRB MSB Position Bit (SHFD) Bit 7

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

8.12.8 CRB Frame Sync Length (FSL) Bit 8

The Frame Sync Length bit selects the type of frame sync to be generated or recognized. If FSL=1, the frame sync will be one bit-clock long during the bit period immediately preceding the first bit period of the word being transferred. If FSL=0, the frame sync will be one data word in length.

8.12.9 CRB Frame Sync Invert (FSI) Bit 9

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

8.12.10 CRB Sync/Async (SYN) Bit 10

The Sync/Async control bit controls whether receive and transmit functions of the SSI occur synchronously or asynchronously with respect to each other. When SYN is cleared, asynchronous mode is chosen and separate frame sync signals are used for the transmit and receive sections. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals.

8.12.11 CRB SSI Mode Select (MOD) Bit 11

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

8.12.12 CRB SSI Transmit Enable (TE) Bit 12

The SSI Transmit Enable bit enables the transfer of data from the TX register to the Transmit Shift Register. When TE is set and a frame sync is detected, the transmit portion of the SSI is enabled. When TE is cleared, the transmitter will be disabled after completing transmission of data currently in the SSI Transmit Shift Register.The serial output is then 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 get cleared but data will not be transferred to the Transmit Shift Register.

Normal transmit enable sequence for transmit is to write data to TX or to TSR **before** setting TE. Normal transmit disable sequence is to clear TE and TIE **after** TDE=1.

In the network mode, the operation of clearing TE and setting it again will disable the transmitter after completion of transmission of a current data word until the beginning of a new data frame period. During the disabled time period, the STD pin will remain in three-state.

Note: TE does not inhibit TDE or transmitter interrupts. TE does not affect the generation of frame sync.

8.12.13 CRB SSI Receive Enable (RE) Bit 13

When the SSI Receive Enable bit is set, the receive portion of the SSI is enabled. When this bit is cleared, the receiver will be disabled by inhibiting data transfer into RX. If data is being received while this bit is cleared, the rest of the word will be shifted in and transferred to the SSI Receive Data Register.

In network mode, the operation of clearing RE and setting it again will disable the receiver after reception of the current data word until the beginning of the new data frame.

Note: RE does not inhibit RDF or receiver interrupts. RE does not affect the generation of a frame sync.

8.12.14 CRB SSI Transmit Interrupt Enable (TIE) Bit 14

When the SSI Transmit Interrupt Enable bit is set, the program controller will be interrupted when the SSI Transmit Data Register Empty flag (TDE) in the SSI 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 with the TE bit. Writing data to the TX or TSR register will clear TDE thus clearing the interrupt.

There are two transmit data interrupts which 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

8.12.15 CRB SSI Receive Interrupt Enable (RIE) Bit 15

When the SSI Receive Interrupt Enable bit is set, the program controller will be interrupted when the SSI Receive Data Register Full flag (RDF) in the SSI 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 SSI STATUS REGISTER (SSISR)

The SSI Status Register is an 8-bit read only status register used by the DSP to interrogate the status and serial input flags of the SSI. The status bits are described in the following paragraphs.

7	6	5	4	3	2	1	0
RDF	TDE	ROE	TUE	RFS	TFS	IF1	IF0

READ-ONLY
SSIx STATUS
REGISTER (SR)
SSI0 ADDRESS $FFF0
SSI1 ADDRESS $FFF8

Note: All the flags in the SSISR are updated one SSI clock after the beginning of a time slot.

8.13.1 SSISR Serial Input Flag 1 and 0 (IF0, IF1) Bit 0, 1

The SSI always latches data present on the SC1x and SC0x pins during reception of the first received bit. IF0 and IF1 are always updated with this data when the receive shift register is transferred into the receive data register. Hardware, software, SSI individual, and STOP resets will clear IF0 and IF1.

8.13.2 SSISR Transmit Frame Sync (TFS) Bit 2

When set the Transmit Frame Sync flag (TFS) indicates that the first bit of the first active word of the frame has been transmitted. Data written to the transmit data register when TFS is set will be transmitted (in network mode) during the second active time slot in the frame.

In network mode, TFS is set during transmission of the first active slot of the frame. It will then be cleared when starting transmission of the next active slot of the frame. If the first time slot of the frame is not enabled, this bit will be set on the leading edge of the first active slot.

Note: In normal mode or in network mode with only one enabled slot, TFS will always read as a one when transmitting data because there is only one time slot per frame.

TFS is cleared by DSP, SSIx or STOP reset and is not affected by TE.

8.13.3 SSISR Receive Frame Sync (RFS) Bit 3

When set, the Receive Frame Sync flag (RFS) indicates that reception of the word in the Serial Receive Data Register (RX) occurred during the first active slot in the frame. When RFS is cleared and a word is received, it indicates (only in the network mode) that the reception of that word did not occur during the first active slot.

In network mode, if the first slot is disabled, this flag will be set after reception in the first active slot and will only be cleared after receiving the next enabled time slot into the RX register.

Note: In normal mode or network mode with one enabled slot, RFS will always be set when reading data because data is received only in the first active time slot.

RFS is cleared by DSP, SSIx or STOP reset and is not affected by RE.

8.13.4 SSISR Transmitter Underrun Error (TUE) Bit 4

The Transmitter Underrun Error 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 active time slot requires transmit data.

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 and, if a transmit interrupt occurs with TUE clear, the Transmit Data Without Errors interrupt will be generated.

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

8.13.5 SSISR Receiver Overrun Error (ROE) Bit 5

The Receiver Overrun Error flag 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 the DSP, SSIx, or STOP reset, and is cleared by reading the SSISR with ROE set followed by reading the RX. Clearing RE does not affect ROE.

8.13.6 SSISR Transmit Data Register Empty (TDE) Bit 6

The SSI Transmit Data Register Empty flag 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 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 SSI Transmit Data interrupt request will be issued when TDE is set. The interrupt vector will depend on the state of the Transmitter Underrun TUE bit. TDE is set by the DSP, SSIx, and STOP reset.

8.13.7 SSISR Receive Data Register Full (RDF) Bit 7

The SSI Receive Data Register Full flag is set when the contents of the Receive Shift Register are transferred to the Receive Data Register. When set, RDF indicates that data should be read from RX so that the next word can be received without an overrun error.

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 vector request will depend on the state of the Receiver Overrun ROE bit. RDF is cleared by the DSP, SSIx, and STOP reset.

8.14 TIME SLOT REGISTER - TSR

The Time Slot Register is used when the data is not to be transmitted (i.e., a blank time slot) 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 for that time slot.

8.15 TRANSMIT SLOT MASK REGISTERS - TSMAx AND TSMBx

The Transmit Slot Mask Registers are two 16-bit read/write registers. They are used by the transmitter in network mode to determine for each slot whether to transmit a data word and generate a transmitter empty condition (TDE=1), or to three-state the transmit data pin, STD. TSMAx and TSMBx should be seen as only one 32-bit register, TSMx. Bit number N in TSMx is an enable/disable control bit for transmission in slot number N.

When bit number N in TSMx is cleared, the transmit data pin STD is three-stated during transmit time slot number N. The data is still transferred from the Transmit Data Register to the transmit shift register and the Transmitter Data Empty flag (TDE) is set. Also the Transmitter Underrun Error flag is not set. This means that during a disabled slot, no Transmitter Empty interrupt is generated (TDE=0). The DSP is interrupted by activity in enabled slots only. Data that is written to the Transmit Data Register when servicing this request is transmitted in the next enabled transmit time slot.

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

The 32-bit data written into TSMx is transferred to the Transmit Slot Mask Shift Register (TSMS) during the last slot of the last frame. Writing to TSMx will not affect the state of the active (enabled) transmit slots during the current transmit frame but only that of the next frame.

Using the slot mask in TSMx does not conflict with using TSR. Even if a slot is enabled in TSMx, the user may choose to write to TSR instead of writing to the transmit data register TX. This will cause the transmit data pin to be three-stated during the next slot.

After DSP reset, the Transmit Slot Mask Register is preset to $FFFFFFFF, which means that all 32 possible slots are enabled for data transmission.

Note: The Transmit Slot Mask Registers have to be written before the last three serial clock cycles of a frame in order to become active at the beginning of the next frame.

8.16 TRANSMIT SLOT MASK SHIFT REGISTER - TSMS

The Transmit Slot Mask Shift Register is a 32-bit shift register. At the end of each frame, it is loaded with the 32-bit mask from TSM. Then, it is shifted one bit to the right near the end of each transmitted word. The LSB of TSMS is used as an enable/disable for transmitting data to the transmit data pin (STD) and setting the TDE flag after transmitting data. This register is not accessible to the programmer.

8.17 RECEIVE SLOT MASK REGISTERS - RSMAx AND RSMBx

The Receive Slot Mask Registers are two 16-bit read/write registers. They are used by the receiver in network mode to determine for each slot whether to receive a data word and generate a receiver full condition (RDF=1), or to ignore the received data. RSMAx and RSMBx should be seen as only one 32-bit register RSMx. Bit number N in RSMx is an enable/disable control bit for receiving data in slot number N.

When bit number N in RSMx is cleared, the data from the receive data pin is shifted into the Receive Shift Register during slot number N. Data is not transferred from the Receive Shift Register to the Receive Data Register and the Receiver Full flag (RDF) is not set. Also the Receiver Overrun Error flag is not set. This means that during a disabled slot, no Receiver Full interrupt is generated. The DSP is interrupted (RDF=0). The DSP is interrupted by activity in enabled slots only.

When bit number N in RSMx is set, the receive sequence is as usual: data which is shifted into the receive shift register is transferred to the Receive Data register and the RDF flag is set.

The 32-bit mask which is written into RSMx, is transferred to the Receive Slot Mask Shift Register (RSMS) during the last slot of the last frame. Writing to RSMx will not affect the state of the active (enabled) receive slots during the current receive frame, but only that of the next frame.

After DSP reset, the Receive Slot Mask Register is preset to $FFFFFFFF, which means that all 32 possible slots are enabled for data reception.

Note: The Receive Slot Mask Registers have to be written before the last three serial clock cycles of a frame in order to become active at the beginning of the next frame.

8.18 RECEIVE SLOT MASK SHIFT REGISTER - RSMS

The Receive Slot Mask Shift Register (RSMS) is a 32-bit shift register. At the beginning of each receive frame, it is loaded with the 32-bit mask from RSM. Then, it is shifted one bit to the right near the end of each received word. The LSB of RSMSR is used as an enable/disable for receiving a word and setting the RDF flag. This register is not accessible to the programmer.

8.19 SSI OPERATING MODES

The SSI on-chip peripheral can operate in three operating modes – Normal, Network, and On-Demand. In the Normal and Network modes, data is transferred and interrupts are generated (if enabled) periodically. The On-Demand mode is a special mode which permits data to be transferred serially when necessary in a non-periodic fashion.

Table 8-5 SSI modes

Mode	SYN	SC1x	SC0x
Normal	Async	RFS In RFS Out RFS In	TFS In TFS Out TFS Out
	Sync	— —	FS In FS Out
Network	Async	RFS In RFS Out RFS In	TFS In TFS Out TFS Out
	Sync	— —	FS In FS Out
On-Demand	Async	RFS Out RFS In	TFS Out TFS Out
	Sync	—	FS Out

8.19.1 Normal Operating Mode

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.

8.19.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI are:

1. Transmitter Enabled (TE=1)
2. Frame sync becomes active

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. The optional frame sync output and clock outputs are not three-stated even if both receiver and transmitter are disabled.

The program must write another word to TX before the next frame sync is received or TUE will be set and a Transmitter Underrun Error will be generated.

8.19.1.2 Normal Mode Receive

If the receiver is enabled, then each time the frame sync signal is generated (or detected) a data word will be clocked in. After receiving the data word it will be transferred from the SSI 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).

8.19.2 Network Mode

In this mode the SSI can be used in TDM networks and 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 one or more time slots. The distinction of the network mode is that active time slots (data word time) are identified by the transmit and receive slot mask registers (TSMA0, TSMB0, RSMA0, and RSMB0). For the transmitter, this allows the option of ignoring a time slot or transmitting data during that 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 when the corresponding time slot is enabled. The DSP will read the receive data register and either use or discard the data according to the time slot register and mask.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission or reception can occur in each time slot (rather than in just the frame sync time slot as in the normal mode). The frame rate dividers, controlled by DC4, DC3, DC2, DC1, and DC0 control the number of time slots per frame from 2 to 32.

8.19.2.1 Network Mode Transmit

The transmit portion of the SSI is enabled when TE=1. However, when TE is sets the transmitter will be enabled only after detection of a new data frame sync. This allows the SSI to synchronize to the network timing.

Normal start up sequence for transmission in the first time slot is to write the data to be transmitted to the Transmit Register (TX), this clears the TDE flag. Then set TE and TIE to enable the transmitter on the next frame sync and to enable transmit interrupts.

Alternatively, the DSP programmer may decide **not** to transmit in the first time slot by writing (any data) to the Time Slot Register (TSR). This will clear the TDE flag just as if data were going to be transmitted, but the STD pin will remain in three-state for the first time slot. The programmer then sets TE and TIE as above.

When the frame sync is detected (or generated), 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 (1) poll TDE, or (2) use interrupts to reload the TX register with new data for the next time slot, or (3) write to the TSR to prevent transmitting in the next active time slot. The transmit and receive slot mask registers control which time slots will be used. Failing to reload TX (or writing to TSR) before the next active time slot will cause a transmitter underrun and the TUE error bit will be set.

Clearing TE and setting it again will disable the transmitter after completion of transmission of the current data word until the beginning of the next frame sync period. During that time the STD pin will be three-stated. TE should be cleared after TDE gets set to ensure that all pending data is transmitted.

To summarize, the network mode transmitter generates interrupts every **enabled** time slot (TE=1, TIE=1) and requires the DSP program to respond to each **enabled** time slot. These responses may be:

1. Write TX data register with data for 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 active time slot and the previous data will be transmitted

8.19.2.2 Network Mode Receive

The receiver portion of SSI is enabled when RE is set; however, the receive enable will take place only after detection of a new data frame. After detection of the new data frame, the first data word will be shifted into the Receive Shift Register during the first active slot. When the word is completely received, it is transferred to the RX 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).

During the second active time slot, the second data word begins shifting in. The DSP program must read the data from RX (which clears RDF) before the second data word is completely received (ready to transfer to RX) or a receive overrun error will occur (ROE gets set).

If the RE bit is cleared and set again by the DSP programmer, the receiver, after receiving the current time slot in progress, will be disabled until the next frame sync (first time slot). This mechanism allows the DSP programmer to ignore data in the last portion of a data frame.

Note: The optional frame sync output and clock output signals are not affected even if the transmitter and/or receiver are disabled. TE and RE do not disable bit clock and frame sync generation.

To summarize, in the Network mode, 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 next active time slot

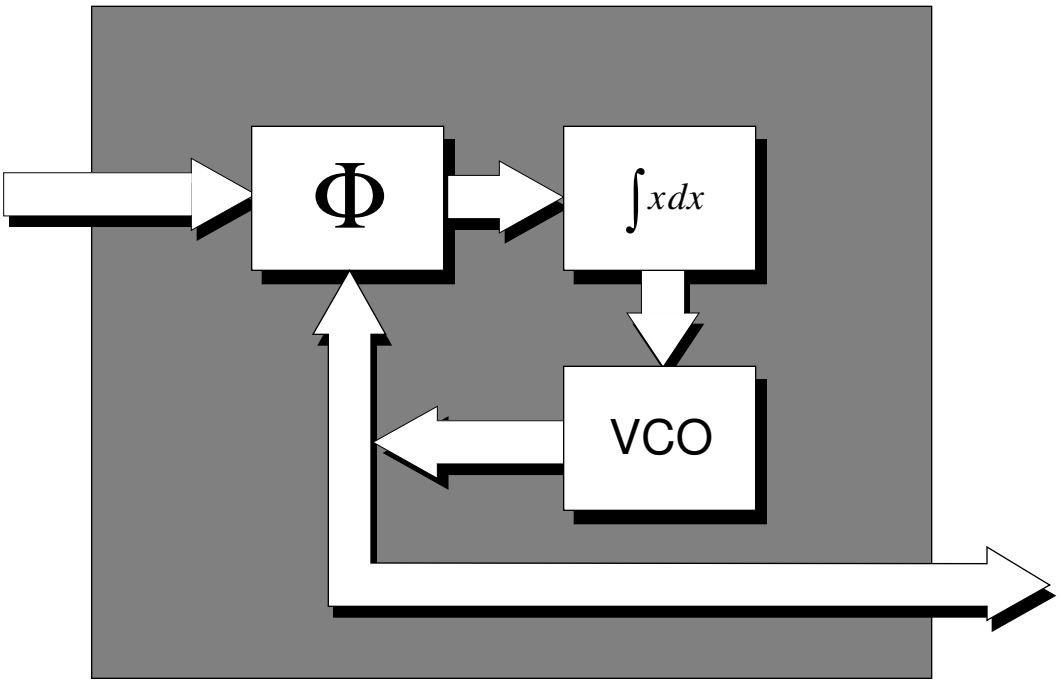
8.19.2.3 On-Demand Mode

A divide ratio of one (DC=00000) in the network mode is a special case. This is the only data driven mode of the SSI and is defined as the on-demand mode. This special case

will not generate a periodic frame sync. A frame sync pulse will be generated only when there is data available to transmit. The On-demand mode requires that the transmit frame sync be internal (output) — i.e., the on-demand mode is disallowed whenever the transmit frame sync is configured as an input. Data transmission is data driven and is enabled by writing data into the TX register. Receive and transmit interrupts function with the TDE and RDF flags as normally; however, transmit underruns (TUE) are impossible for on-demand transmission.

SECTION 9

ON-CHIP FREQUENCY SYNTHESIZER



SECTION CONTENTS

9.1 INTRODUCTION

The PLL performs frequency multiplication to allow the processor to use almost any available external system clock for full speed operation, while also supplying an output clock synchronized to a synthesized internal core clock. It improves the synchronous timing of the processor's external memory port, eliminating the timing skew common on other processors.

The PLL's ability to operate at a high internal frequency using a low frequency input offers two immediate benefits. Lower frequency inputs reduce the overall electromagnetic interference generated by a system, and the ability to oscillate at different frequencies reduces costs by eliminating the need to add additional oscillators to a system.

This PLL is strictly used for generating clocks in the DSP and is not intended to lock on a signal to be processed. The basic operation of the PLL is as follows. The phase comparator compares its two inputs and generates an difference signal which is integrated by the filter. The filter output is used to control the voltage controlled oscillator (VCO) which, if the VCO is fed directly into the phase comparator, forms a closed negative feedback loop that will cause the VCO to oscillate at the same phase and frequency as the input clock. If the output of the VCO is divided by n before being fed to the phase comparator, then the VCO must oscillate at n times the input clock rate so that the output of the divider which is fed to the phase comparator is the same frequency and phase as the input clock.

The DSP56156 does not contain an on-chip oscillator. An external system clock must be connected to the EXTAL pin. Figure 9-1 shows the on-chip frequency synthesizer general block diagram.This clock, after being squared, can be divided on-chip by a four bit divider and used as master clock for the codec block and for the on-chip phase-locked loop (PLL) block. The codec input clock can also be sourced by a fixed, input clock divided by 6.5, signal. The PLL generates the DSP core system clock but can also be bypassed allowing the core to directly use the clock provided on the EXTAL pin.

Note: This Section replaces the PLL description in the *DSP56100 Family Manual* for the DSP56156. There are some differences in the control registers for the PLL for the DSP56156 that are unique to this part.

9.1.1 PLL Components

The PLL block diagram is shown in Figure 9-1. The components of the PLL are described in the following sections and a programming model summary (Figure 9-3) is at the end of the section.

9.1.1.1 Phase Comparator and Filter

The Phase Comparator detects any phase difference between the external clock and an

internal clock phase that is generated by the VCO down counter. At the point where there is negligible phase difference and the frequency of the two inputs is identical, the PLL is in the “locked” state.

The filter receives signals from the phase comparator, and either increases or decreases the voltage fed to the Voltage Controlled Oscillator (VCO). An external capacitor is connected to the SXFC pin (described in Section 2.5) and determines the PLL operating characteristics.

When the PLL is in the stage where it is acquiring the proper phase and frequency, it operates in a wide bandwidth mode. After the PLL locks on to the proper phase/frequency, it reverts to the narrow bandwidth mode, which is useful for tracking small changes in the EXTAL clock due to frequency drift.

9.1.1.2 Voltage Controlled Oscillator (VCO)

The VCO is controlled by the phase difference between the two inputs to the phase comparator. The VCO output is divided by four and then by the 4-bit VCO count down counter (see Figure 9-1) which lowers the frequency fed to the phase comparator. This forms a negative feedback loop that requires that the VCO run at a faster rate than the reference input to the phase comparator. The net effect is that the down counter becomes a frequency multiplier.

9.1.1.3 Dividers

The four divider bits, ED3-ED0, define the on-chip PLL resolution. The four down counter bits, YD3-YD0, control the down counting in the PLL feedback loop causing it to divide by the value YD+1. The DSP core system frequency is controlled (first divided by the contents in ED and then multiplied by the contents of YD) by the PLL Control Register (PLCR) frequency control bits as follows:

$$F_{osc} = [F_{ext} \div (ED + 1)] \times 4[YD+1]$$

where ED is the value contained in ED3-ED0,YD the value contained in YD3-YD0, and Fext is the clock frequency applied to the EXTAL 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 ED register and the CLKO circuitry if the PLL is disabled (PLLD=1) and if the CLKO is turned off (CD=0 in OMR) when entering the STOP mode (see section 9.3.5).

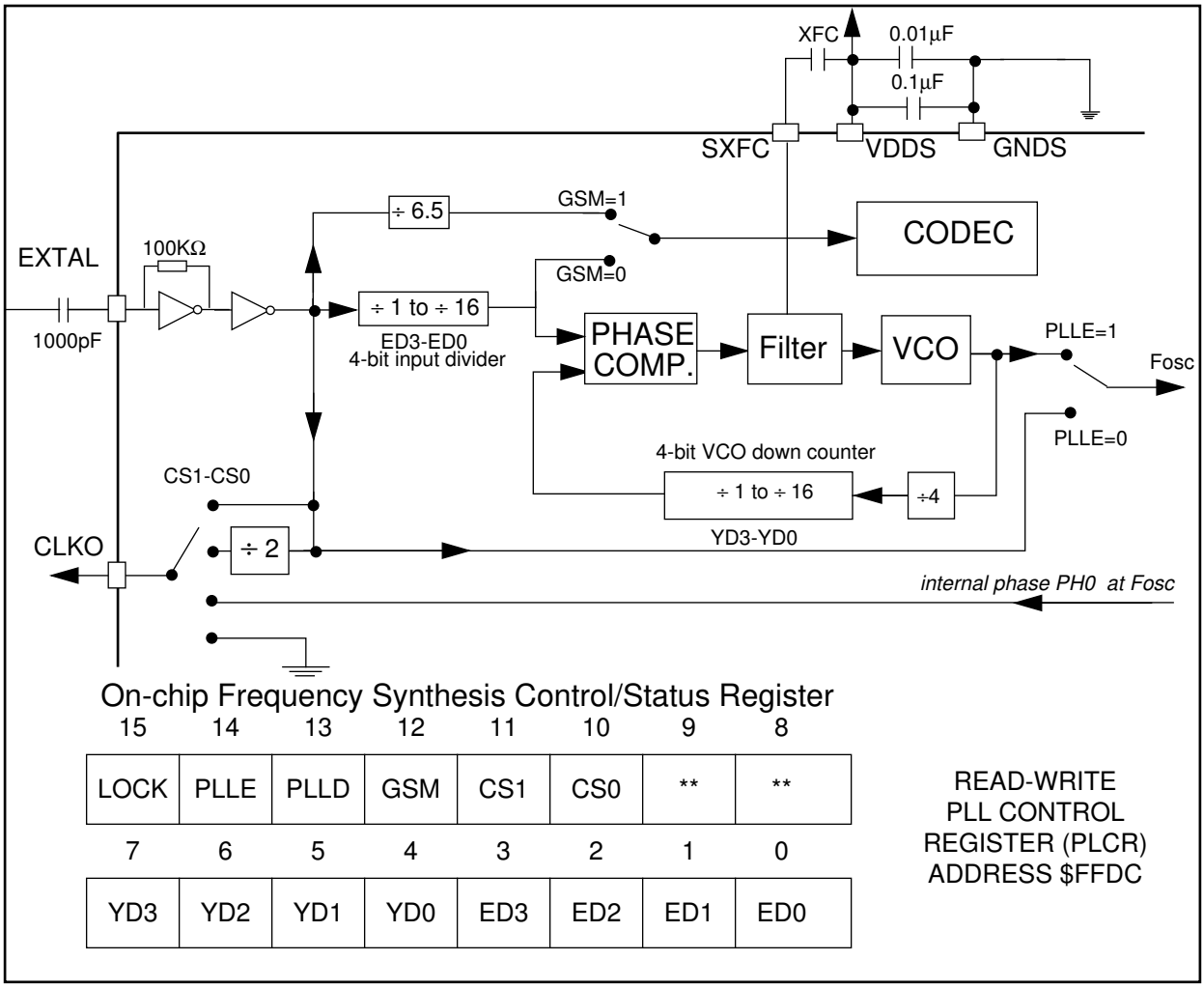


Figure 9-1 Frequency Synthesis Block Diagram and Control Register

9.2 ON-CHIP CLOCK SYNTHESIS EXAMPLES

Three examples are given in Figure 9-2 in order to illustrate the functionality of the frequency synthesis block.

9.2.1 Example One

In the first example, the 4-bit input divider is not used to provide the codec clock. Instead, the 6.5 divider generates a 2 MHz clock to the codec from the 13 MHz external clock. The input divider and the 4-bit PLL down counter can be used to select the desired operating frequency for the DSP core. The PLL output frequency cannot be lower than 10MHz and higher than the maximum DSP core operating frequency. The PLL can also be disabled (PLLE=0), in which case the core will directly use the 13 MHz clock and will run at 6.5 MIPS.

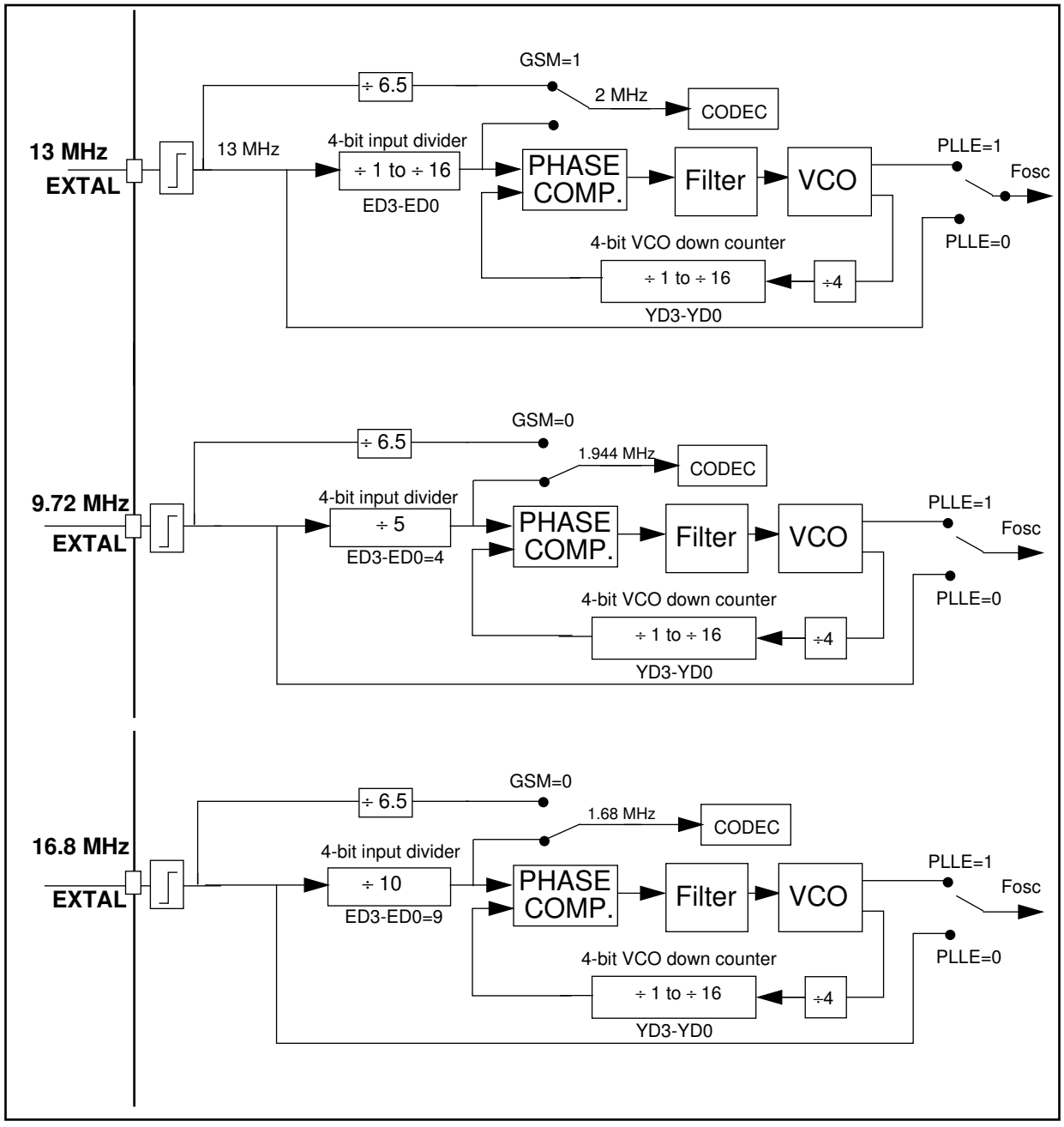


Figure 9-2 Three On-chip Clock Synthesis Examples

9.2.2 Example Two

In the second example, the 4-bit input divider divides the input clock by 5 providing a 1.944 MHz clock to the ΣΔ codec and to the PLL. The PLL can multiply the 1.944 MHz clock up

to the desired DSP core operating frequency. The PLL can also be disabled (PLLE=0), in which case the core will directly use the 9.72 MHz clock and will run at 4.8 MIPS.

9.2.3 Example Three

In the third example, the 4-bit input divider divides the input clock (16.8MHz) by 10, providing a 1.68 MHz clock to the ΣΔ codec and to the PLL. The PLL can multiply the 1.68 MHz clock up to the desired DSP core operating frequency. The PLL can also be disabled (PLLE=0), in which case the core will directly use the 16.8 MHz clock and will run at 8.4 MIPS.

9.3 ON-CHIP CLOCK SYNTHESIS CONTROL REGISTER PLCR

The Clock Synthesis Control Register, PLCR, is a 16-bit read/write register used to direct the on-chip clock synthesis operation. The PLCR controls two programmable dividers and can enable or disable the on-chip PLL. The PLCR control bits are described in the following sections. All PLCR bits are cleared by DSP hardware reset. Software reset does not affect this register.

9.3.1 PLCR Input Divider Bits (ED3-ED0) Bits 0-3

The four input divider bits are used to divide the input clock frequency by any number between 1 and 16. The divider output is used as an input clock for the on-chip codec section and for the on chip PLL as shown in Figure 9-1. If ED is the value contained in the four bits, the external clock is divided by ED+1. Any time a new value is written to the ED bits, the LOCK bit is cleared.

Care should be taken not to exceed the codec maximum operating frequency and to remain in the frequency stability domain of the PLL.

9.3.2 PLCR Feedback Divider Bits (YD3-YD0) Bits 4-7

The four feedback divider bits, YD3-YD0, cause the down counter in the feedback loop to divide by the value YD+1 where YD is the value contained in YD3-YD0. Changing these bits requires a time delay for stabilization so that the Voltage Controlled Oscillator (VCO) can relock. Any time a new value is written to the YD bits, the LOCK bit is cleared.

The resulting DSP core system clock must be within the limits specified by DSP56156 technical data sheet.

9.3.3 PLCR Clockout Select Bits (CS1-CS0) Bits 10-11

The two Clockout Select bits, CS1-CS0, determine the frequency put on the CLCKO pin when the CD bit in the OMR register is cleared (CD=0). After hardware reset, the internal

DSP core phase PH0 is put on the CLCKO pin. PH0 is a delayed version of the DSP core master clock Fosc. Fext or Fext/2 (Fext being a squared and delayed version of the signal applied to the EXTAL input pin) can also be selected on CLCKO by changing CS1-CS0 according to Table 9-1

Table 9-1 CLKOUT Pin Control

CS1	CS0	CLKOUT
0	0	PH0
0	1	reserved
1	0	Squared Fext
1	1	Squared Fext/2

9.3.4 GSM Bit (GSM) Bit 12

This bit is used to select the on-chip codec clock. When the GSM bit is cleared, the output of the four bit divider ED3-ED0 is selected as the on-chip codec clock. When this bit is set, the clock applied to EXTAL is divided by 6.5 and selected for the codec clock. The status of this bit does not affect the input clock of the on-chip PLL. Like all COCR bits, the GSM bit is cleared by the DSP hardware reset and is not affected by software reset.

9.3.5 PLCR PLL Power Down Bit (PLLD) Bit 13

When the PLLD bit is set, the on-chip PLL is put in the power down mode and stops operating. When this control bit is cleared, the on-chip PLL is turned on. This bit cannot be set when the PLLE bit is set.

Since the STOP instruction does not power down the PLL, 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.

9.3.6 PLCR PLL Enable Bit (PLLE) Bit 14

When the PLLE bit is set, the DSP core system clock is generated by the on-chip PLL. When this control bit is cleared, the external frequency applied to the EXTAL pin is used as the system clock for the core, by-passing the PLL. The PLL state is defined by the PLLD bit: when the PLLD bit is set, the PLL is in the power down mode and when the PLLD bit is cleared, the PLL is in the active mode. Table 9-2 summarizes the function of PLLE and PLLD.

Table 9-2 PLL Operations

PLLE	PLLD	Fosc	PLL Mode
0	0	EXTAL	Active
0	1	EXTAL	Power Down
1	0	PLLout	Active
1	1	reserved	

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 the power down mode by setting PLLD.

If the PLL output frequency has to be changed by re-programming the ED and/or YD bits while the PLL output is used by the core (PLLE=1;PLLD=0), the following sequence should be performed:

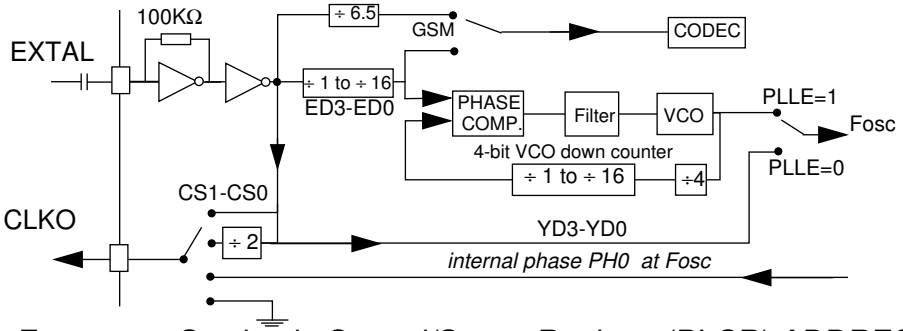
- clear PLLE bit to switch back to EXTAL
- program YD and/or ED bits (only after clearing PLLE)
- wait for the LOCK bit to be set
- set PLLE after the LOCK bit is set

9.3.7 PLCR Voltage Controlled Oscillator Lock Bit (LOCK) Bit 15

This status bit indicates if 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 by setting the PLLD bit and by changing the value of ED or YD. The LOCK bit is not cleared when clearing the PLLE bit unless PLLD, YD, and ED are also changed. This bit is read-only and cannot be written by the DSP core.

9.3.8 PLCR Reserved Bits (Bits 8-9,12)

These bits are reserved and should be written as zero by the user. They read as a logic zero.



On-chip Frequency Synthesis Control/Status Register (PLCR) ADDRESS X:$FFDC

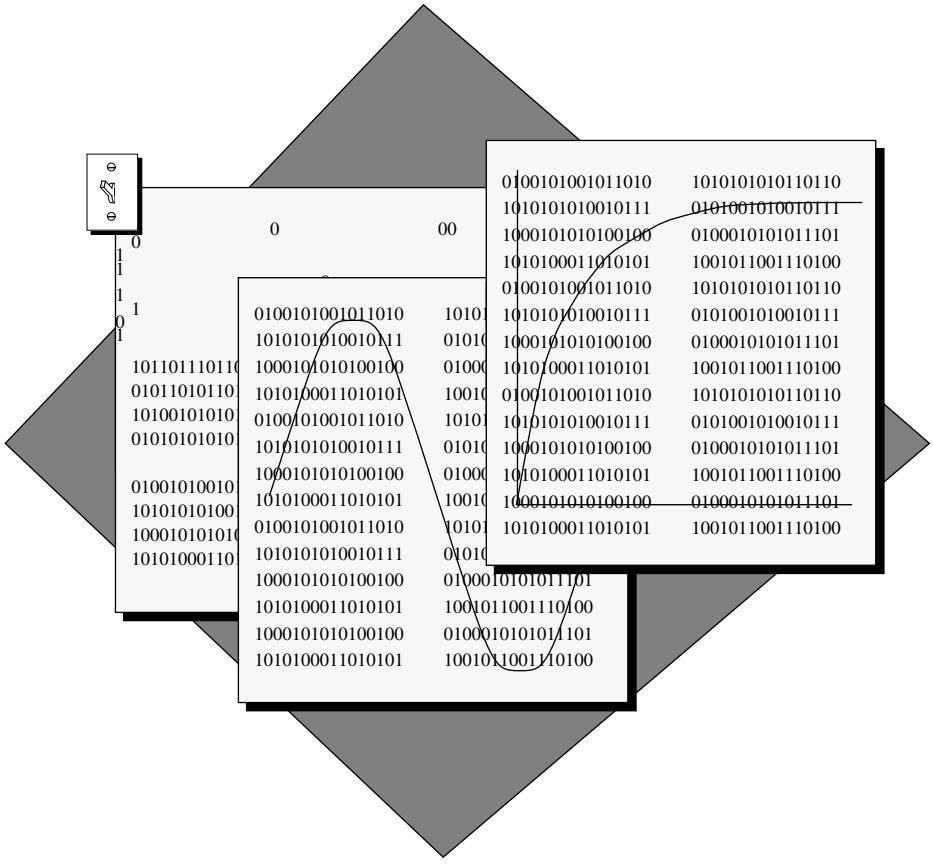
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	PLLE	PLLD	GSM	CS1	CS0	**	**	YD3	YD2	YD1	YD0	ED3	ED2	ED1	ED0

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
GSM	0	Output of the 4 bit divider selected as codec clock input
	1	Output of the 6.5 divider selected as codec clock input
CS1-CS0 CLKO Select	0	PH0 output to CLKO when enabled by the CD bit (bit 7) of the OMR
	1	reserved
	2	Fext output to CLKO when enabled by the CD bit (bit 7) of the OMR
	3	Fext/2 output to CLKO when enabled by the CD bit (bit 7) of the OMR
ED3-ED0 Clock Input Divider	$0	Divide the input clock by 1
	$1	Divide the input clock by 2
	$2	Divide the input clock by 3
	$3	Divide the input clock by 4
	$4	Divide the input clock by 5
	$5	Divide the input clock by 6
	$6	Divide the input clock by 7
	$7	Divide the input clock by 8
	$8	Divide the input clock by 9
	$9	Divide the input clock by 10
	$A	Divide the input clock by 11
	$B	Divide the input clock by 12
	$C	Divide the input clock by 13
	$D	Divide the input clock by 14
	$E	Divide the input clock by 15
	$F	Divide the input clock by 16
YD3-YD0 VCO Down Counter	$0	PLL multiplies the clock by 4
	$1	PLL multiplies the clock by 8
	$2	PLL multiplies the clock by 12
	$3	PLL multiplies the clock by 16
	$4	PLL multiplies the clock by 20
	$5	PLL multiplies the clock by 24
	$6	PLL multiplies the clock by 28
	$7	PLL multiplies the clock by 32
	$8	PLL multiplies the clock by 36
	$9	PLL multiplies the clock by 40
	$A	PLL multiplies the clock by 44
	$B	PLL multiplies the clock by 48
	$C	PLL multiplies the clock by 52
	$D	PLL multiplies the clock by 56
	$E	PLL multiplies the clock by 60
	$F	PLL multiplies the clock by 64

Figure 9-3 On-chip Frequency Synthesizer Programming Model Summary

APPENDIX A

BOOTSTRAP MODE OPERATING MODE 0 OR 1



SECTION CONTENTS

A.1 INTRODUCTION

The bootstrap feature of the DSP56156 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 program RAM parts. Program ROM parts do not have the bootstrap feature available. **As a result, this appendix only applies to program RAM parts.**

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 SSI0. You have 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 DSP56156 is placed in Operating Mode 0 or 1. The control logic maps the bootstrap ROM into program memory space as long as the DSP56156 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:$C000. If the DSP is in Operating Mode 1 it will load from either the Host Interface or SSI0 depending on whether P:$C000 bit 15 is zero or one respectively. The bootstrap firmware changes operating modes when the bootstrap load is completed. When the DSP56156 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 SSI0.
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.

You may also select the bootstrap mode 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 (e.g., see Bootstrap code for DSP56156) to begin the bootstrap process as described above in steps 1-4. This technique allows the DSP56156 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 DSP56156 PRAM. The program is written in DSP56100 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 SSI0. The particular method used is selected by (1) whether 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 DSP56156 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 DSP56156 to begin executing at location P:$0000 by setting HF0=1 in the Host Interface during the bootstrap load. In most systems, the DSP56156 responds so fast that handshaking between the DSP56156 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 SSI0 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, SSI0 Bootstrap and External Bus Bootstrap
;
; This is the Bootstrap program contained in the DSP56156 RAM based part. 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 SSI0 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       ; SSI0 Control register A
M_CRB0          EQU     $FFD1       ; SSI0 Control register B
M_SR0           EQU     $FFF0       ; SSI0 Status register
M_RX0           EQU     $FFF1       ; SSI0 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 SSI0 interface logic
; It will load 4,096 bytes from the parallel host processor interface if bit 15 of P:$C000 is cleared
; and from the Serial Synchronous Interface SSI0 if bit 15 of P:$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 SSI0, 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 DSP56156 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 SSI0 to 8 bit mode
            MOVE        #$2000,A
            MOVE        A,X:M_CRB0              ; Set CRB0 to external clock, async.mode
                                                ; with reception enabled
            BFSET       #$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 SSI0 pins.
;
            BLC         <_HOSTLD                ; Load from Host Interface
                                                ; if the limit flag is clear
; Bootstrap byte per byte from SSI0
;
_SSILD      DO #2,_LOOP2
_SSIWT      BFTSTL      #$80,X:M_SR0            ; Test RDF flag
            BCS         _SSIWT                  ; Wait for RDF to go high
            MOVEP       X:M_RX0,B               ; Put receive SSI0 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 DSP56156 Bootstrap Program (Continued)

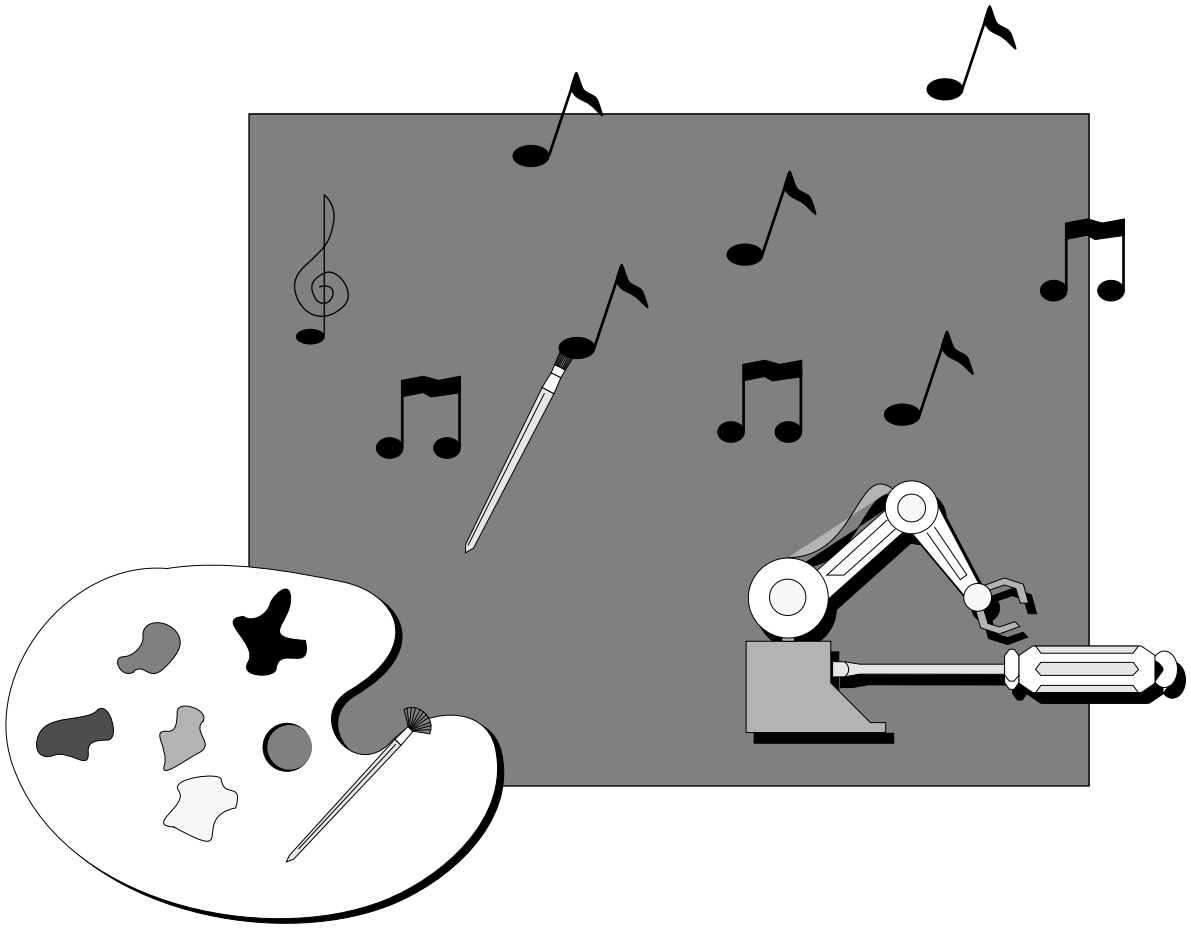
```
;
; Bootstrap from the parallel host interface
;
_HOSTLD   BFSET    #1,X:(R2)          ; Configure Port B as Host Interface.
_LBLA     BFTSTH   #8,X:<<M_HSR       ; Test HF0.
          BRKCS                       ; Stop loading if HF0=1.
;
          BFTSTL   #1,X:M_HSR         ; Test HRDF flag
          BCS      _LBLA              ; Wait for HRDF to go high
                                      ; (meaning the data is present)
          MOVEP    X:M_HRX,A          ; Put 16-bit host data in X0
_STORE    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 DSP56156 Bootstrap Program (Continued)

APPENDIX B

DSP56156 APPLICATION EXAMPLES



SECTION CONTENTS

The lowest cost DSP56156-based system is shown in Figure B-1. This system uses no run time external memory and requires only two chips; the DSP56156 and a low cost EPROM.

Figure B-2 shows the DSP56156 bootstrapping via the Host Interface from an MC68000.

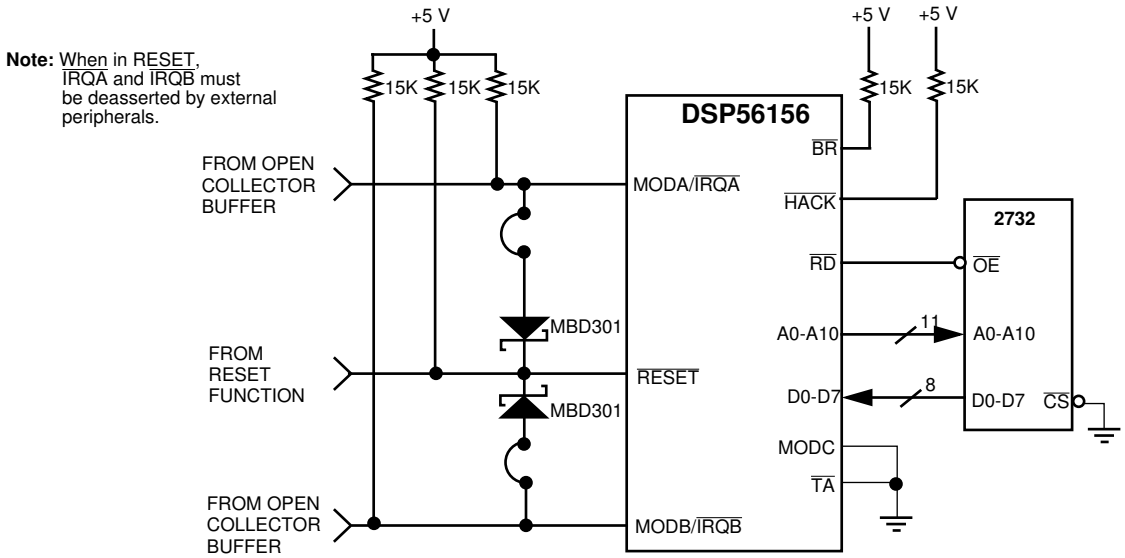


Figure B-1 No Glue Logic, Low Cost Memory Port Bootstrap — Mode 0

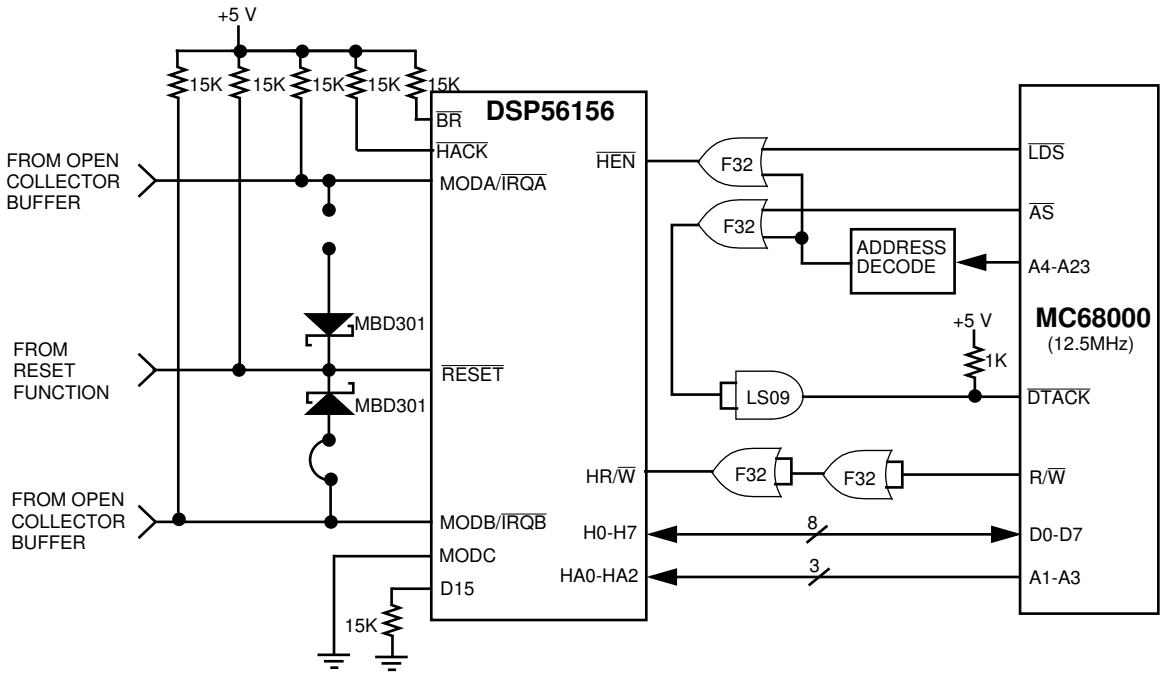


Figure B-2 DSP56156 Host Bootstrap Example — Mode 1

Systems with external program memory can load the on-chip PRAM without using the bootstrap mode. In Figure B-3, the DSP56156 is operated in mode 2 with the reset vector at external program memory at location $E000. The programmer can overlay the high speed on-chip PRAM with DSP algorithms by using the MOVEM instruction.

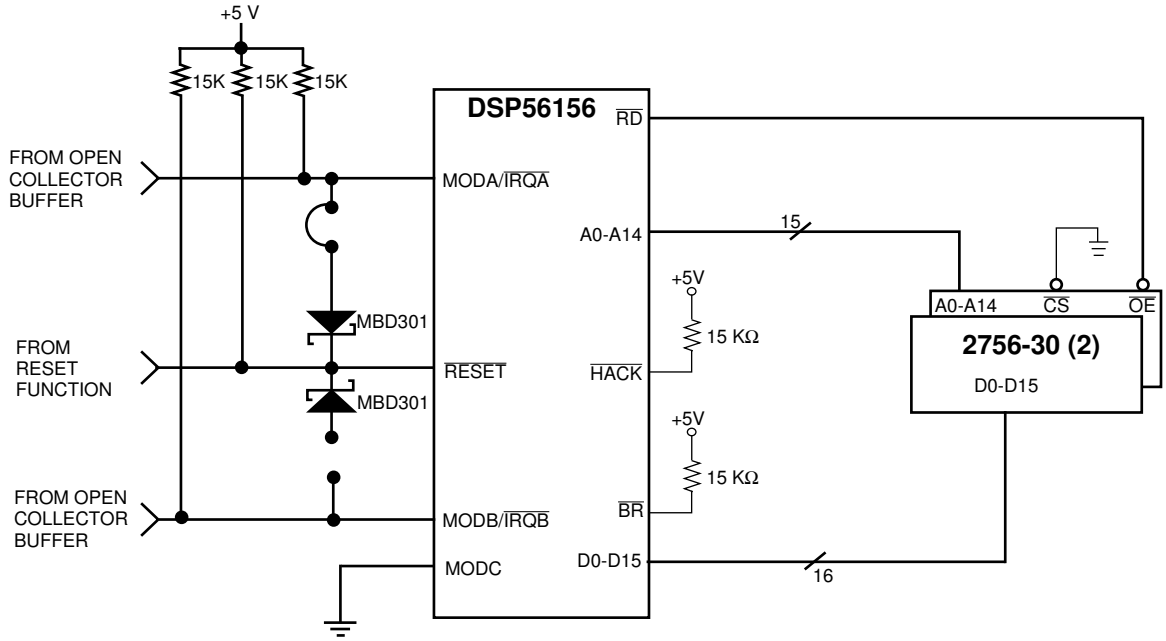


Figure B-3 32K Words of External Program ROM — Mode 2

Figure B-4 is a simple manual reset circuit and Figure B-5 adds the ability to remotely reset the DSP.

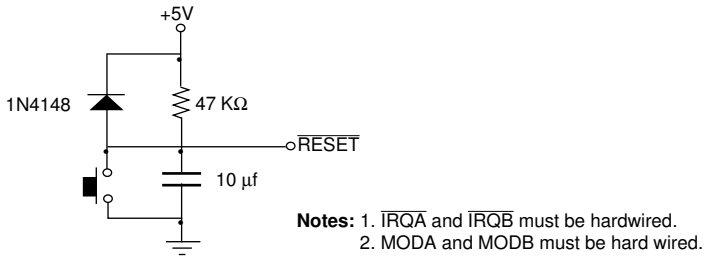


Figure B-4 Reset Circuit

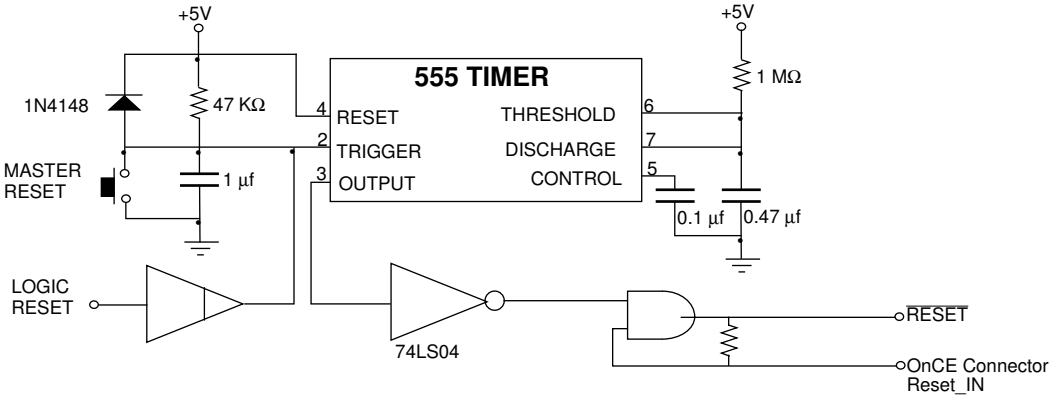
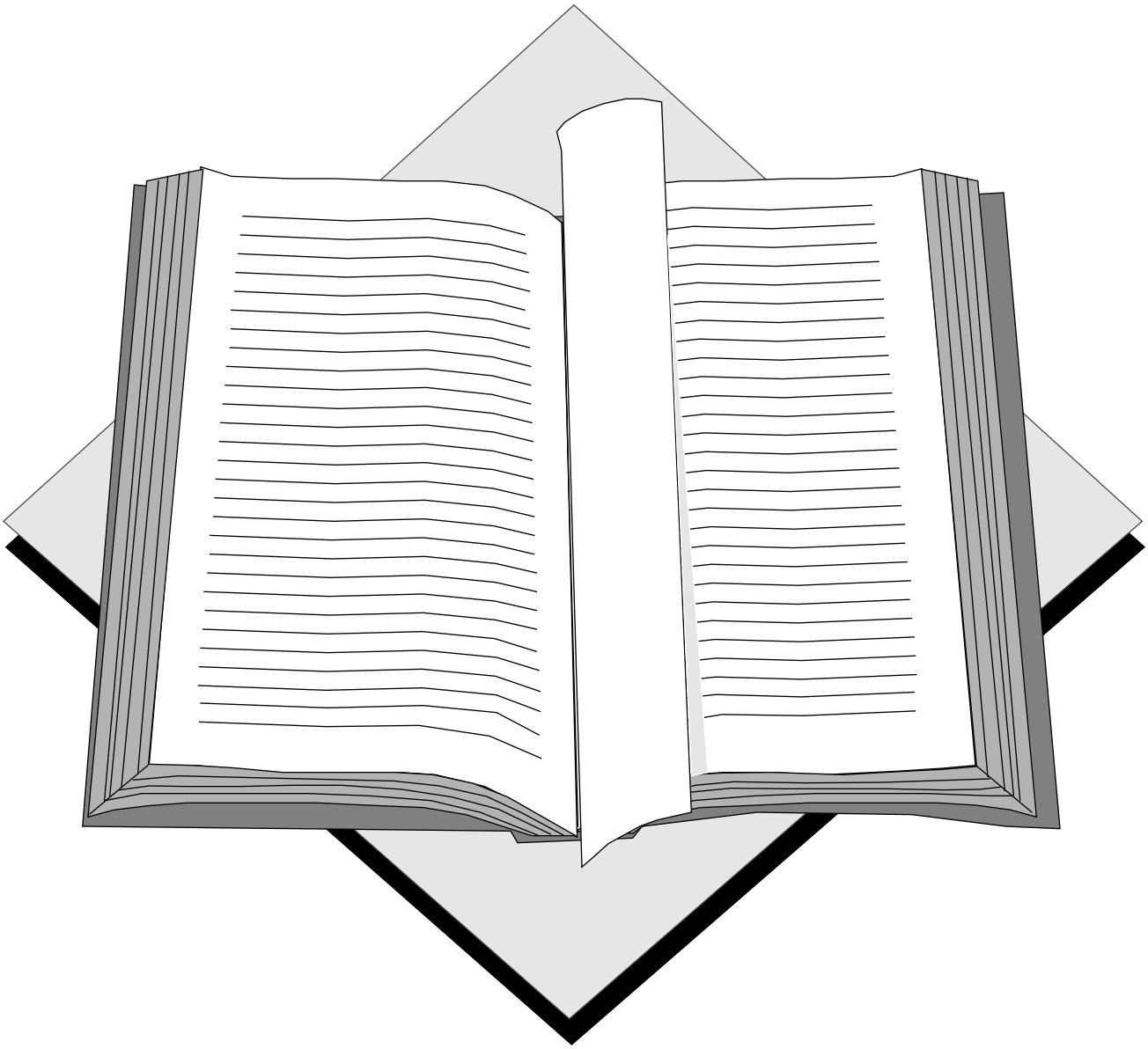


Figure B-5 Reset Circuit Using 555 Timer

INDEX



INDEX

—D—

—R—

—S—

—T—

—V—

—W—

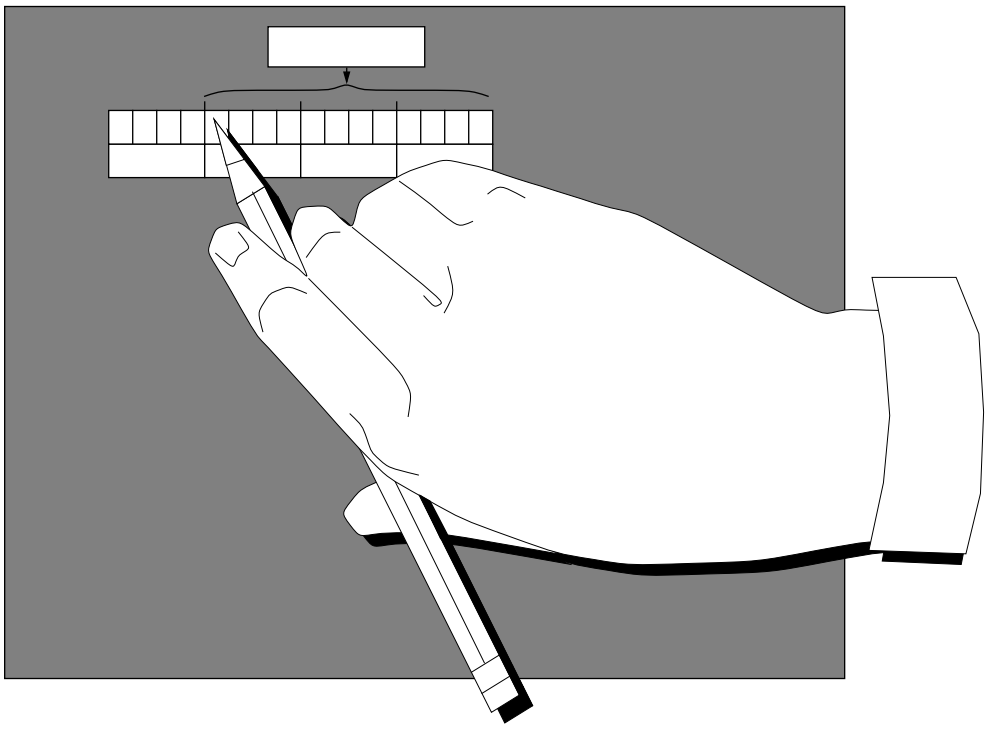
—X—

—Y—

APPENDIX C

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 DSP56156. These programming sheets 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 also the hexadecimal equivalent for each register.



SECTION CONTENTS

PERIPHERAL ADDRESSES

$FFFF	Reserved for on-chip emulation	$FFDF	IPR: Interrupt Priority Register
$FFFE		$FFDE	BCR: Bus Control Register
$FFFD	TSMB1 SSI1 Register	$FFDD	reserved for future use
$FFFC	TSMA1 SSI1 Register	$FFDC	PLCR
$FFFB	RSMB1 SSI1 Register	$FFDB	
$FFFA	RSMA1 SSI1 Register	$FFDA	
$FFF9	TX/RX SSI1 TX/RX Registers	$FFD9	CRB-SSI1 Control Register B
$FFF8	SR/TSR SSI1 Status Register	$FFD8	CRA-SSI1 Control Register A
$FFF7		$FFD7	
$FFF6		$FFD6	
$FFF5	TSMB0 SSI0 Register	$FFD5	
$FFF4	TSMA0 SSI0 Register	$FFD4	
$FFF3	RSMB0 SSI0 Register	$FFD3	
$FFF2	RSMA0 SSI0 Register	$FFD2	
$FFF1	TX/RX SSI0 TX/RX Registers	$FFD1	CRB-SSI0 Control Register B
$FFF0	SR/TSR SSI0 Status Register	$FFD0	CRA-SSI0 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	CRX/CTX	$FFC9	reserved for test
$FFE8	COSR	$FFC8	COCR
$FFE7		$FFC7	
$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 C-1 On-chip Peripherals

INTERRUPT VECTOR ADDRESSES

Table C-1 Interrupts Starting Addresses and Sources

Interrupt Starting Address	IPL	Interrupt Source
$0000	3	Hardware $\overline{\text{RESET}}$
$0002	3	Illegal Instruction
$0004	3	Stack Error
$0006	3	Reserved
$0008	3	SWI
$000A	0-2	$\overline{\text{IRQA}}$
$000C	0-2	$\overline{\text{IRQB}}$
$000E	0-2	Reserved
$0010	0-2	SSI0 Receive Data with Exception Status
$0012	0-2	SSI0 Receive Data
$0014	0-2	SSI0Transmit Data with Exception Status
$0016	0-2	SSI0 Transmit Data
$0018	0-2	SSI1 Receive Data with Exception Status
$001A	0-2	SSI1 Receive Data
$001C	0-2	SSI1 Transmit Data with Exception Status
$001E	0-2	SSI1 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 Transmit/Receive
$0030	0-2	Available for Host Command
$0032	0-2	Available for Host Command
$0034	0-2	Available for Host Command
$0036	0-2	Available for Host Command
$0038	0-2	Available for Host Command
$003A	0-2	Available for Host Command
$003C	0-2	Available for Host Command
$003E	0-2	Available for Host Command

INSTRUCTIONS

Table C-2 Instruction Set Summary — Sheet 1 of 3

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	S L E U N Z V C
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	#iiii,X:<aa> #iiii,X:<pp> #iiii,X:<ea> #iiii,D		2	4+mvb	- * - - - - - ?
BFCLR	#iiii,X:<aa> #iiii,X:<pp> #iiii,X:<ea> #iiii,D		2	4+mvb	- * - - - - - ?
BFSET	#iiii,X:<aa> #iiii,X:<pp> #iiii,X:<ea> #iiii,D		2	4+mvb	- * - - - - - ?
BFTSTH	#iiii,X:<aa> #iiii,X:<pp> #iiii,X:<ea> #iiii,D		2	4+mvb	- * - - - - - ?
BFTSTL	#iiii,X:<aa> #iiii,X:<pp> #iiii,X:<ea> #iiii,D		2	4+mvb	- * - - - - - ?
Bcc	xxxx ee Rn		1+ea	4+jx	- - - - - - - -
BRA	xxxx aa Rn		1+ea	4+jx	- - - - - - - -
BRKcc			1	2/8	- - - - - - - -
BScc	xxxx Rn		1+ea	4+jx	- - - - - - - -
BSR	xxxx Rn		1+ea	4+jx	- - - - - - - -
CHKAAU		(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	- * * * * * * -

INSTRUCTIONS

Table C-2 Instruction Set Summary — Sheet 2 of 3

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	- - - - - - - -
JScc	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 S1,S2,D S1,S2,D	(one parallel move). . . . . 1 (two parallel reads) D,X:(Rn)+NnS,D		2+mv	* * * * * * * -
MACR	(±)S2,S1,D S1,S2,D	(one parallel operation). . 1 (two parallel reads)		2+mv	* * * * * * * -
MAC(uu,su)	S1,S2,D	(no parallel move).	1	2	* * * * * * * -
MOVE	 #xxxx,D	(one parallel operation). 1+ea (double memory read) (memory access, register move)		2+mv	* * - - - - - -
No parallel data move		(.....)	mv	mv	- - - - - - - -
Register to register data move		S,D(.....);	mv	mv	* ?- - - - - -
Address register update		(.....)ea	mv	mv	- - - - - - - -
X memory data move		(.....)X:<ea>,D.	mv	mv	* ?- - - - - -
		(.....)S,X:<ea>.	mv	mv	
X memory data move with short displacement		(.....)X:(R2+xx),D	mv	mv	* ?- - - - - -
		(.....)S,X:(R2+xx).	mv	mv	
X memory data write and register data move (MPY or MAC)		D,X:(Rn)+NnS,D.	mv	mv	* ?- - - - - -
Dual X memory data read		(.....)X:<ea>,D1 X:<ea>,D2.	mv	mv	- - - - - - - -
MOVE(C)	X:<ea>,D S,X:<ea> #xxxx,D S,D X:(R2+xx),D S,X:(R2+xx)	. . . . .	1+ea	2+mvc	* ?? ? ? ? ??
MOVE(I)	#xx,D	. . . . .	1	2	- - - - - - - -

INSTRUCTIONS

Table C-2 Instruction Set Summary — Sheet 3 of 3

Mnemonic	Syntax	Parallel Moves	Instruction Program Words	Osc. Clock Cycles	S L E U N Z V C
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	2+mvm	* * - - - - - -
MOVE(P)	X:<pp>,D X:<pp>,D S,X:<pp> X:<pp>,X:<ea>	. . . . .	1	4+mvp	* * - - - - - -
MOVE(S)	X:<aa>,D S,X:<aa>	. . . . .	1	4+mvp	* * - - - - - -
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 S,D	(parallel move) (two parallel reads)	1	2+mv	* * * * * * * *
SUBL	S,D	(parallel move)	1	2+mv	* * * * * * ? *
SWAP	D	(no parallel move)	1	2	- - - - - - - -
SWI			1	8	- - - - - - - -
Tcc	(S,D) S,D	R0,Rn	1	2	- - - - - - - -
TFR	S,D S,D	(one parallel operation) (two memory reads)	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 * * * * * 0 0
TST(2)	S	(no parallel move)	1	2	- * * * * * 0 0
WAIT			1	n/a	- - - - - - - -
ZERO	D	(no parallel move)	1	2	- * * * * * * -

INSTRUCTIONS

Table C-3 Dual Read Instructions

DSP56156					
DATA ALU OPERATION		**DOUBLE EFFECTIVE ADDRESS**		**DOUBLE DESTINATION**	
Oper.	**Reg.**	**Read1**	**Read2**	**Dest1**	**Dest2**
MOVE		(Rn)+	(R3)+	~F	X0
MAC/R MPY/R	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] F = 0 -> A F = 1 -> B		X0	X1
ADD SUB TFR	X1,F X0,F Y1,F Y0,F			Y0	X1
				~F	Y0
				Y1	X1
ADD	~F,F				
SUB	~F,F				
TFR	~F,F				

Table C-4 Lms Instruction

DSP56156					
DATA ALU OPERATION		**DOUBLE TRANSFER**			
Oper.	**Reg.**	**TRANSFER1**		**TRANSFER2**	
MAC MPY	X0,X0,F	~F	(Rn)+Nn	X1	~F
	X1,X0,F	n=[0,2] F = 0 -> A F = 1 -> B ~F= opposite acc.		X0	~F
	A1,Y0,F			Y1	~F
	B1,X0,F			Y0	~F
	Y0,X1,F				
	Y1,X1,F				
	Y1,X0,F				
	Y0,X0,F				

INSTRUCTIONS

Table C-5 Data ALU Instructions with One Parallel Operation

DSP56156			
DATA ALU OPERATION		**PARALLEL MEMORY READ or WRITE**	
Oper.	**Reg.**	**Eff. Address**	**Dest/Source**
MAC **MPY**	±X0,X0,F ±X1,X0,F ±A1,Y0,F ±B1,X0,F ±Y0,X1,F ±Y1,X1,F ±Y1,X0,F ±Y0,X0,F	(Rn)+ (Rn)+Nn (~F1) (R2+xx)	X1 X0 Y1 Y0 A0 B0 A B
		ONE ADDRESS UPDATE	
ADD **SUB** **TFR** **OR/AND** **EOR** **CMP/CMPM**	X1,F X0,F Y1,F Y0,F	**Eff. Address**	
		(Rn)-	
		(Rn)+Nn	
		PARALLEL REGISTER TRANSFER	
		Source	**Destination**
ADD **SUB**	X,F Y,F	X0	~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** **NOT** **ROL/ROR** **LSL/LSR**	F	F	~F
		A0	X0
		A0	X1
		B0	Y0
		B0	Y1
		No Transfer	

INSTRUCTIONS

Table C-6 Bit Field Manipulation Instructions

DSP56156		
OPERATION	**OPERAND**	**COMMENTS**
BFTSTH #iiii, **BFTSTL #iiii,** **BFCHG #iiii,** **BFSET #iiii,** **BFCLR #iiii,**	X:(Rn)	n=[0,3]
	X:<aa>	first 32 word of X memory 5 bit address
	X:<pp>	last 32 word 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	

Table C-7 Effective Address Update

DSP56156		
OPERATION	**SOURCE ADDRESS REGISTER**	**DESTINATION REGISTER**
LEA	(Rn) (Rn)+ (Rn)- (Rn)+Nn n=[0,3]	R0,R1,R2,R3 N0,N1,N2,N3

INSTRUCTIONS

Table C-8 Jump/branch Instructions

DSP56156		
OPERATION	**OPERAND**	**COMMENTS**
JSR **JMP** **Jcc** **JScc**	(Rn)	n=[0,3]
	$xxxx	16-bit absolute address
BSR **BRA** **Bcc** **BScc**	(Rn)	n=[0,3]
	$xxxx	16-bit absolute address
JSR	AA	8-bit absolute address [0,256]
BRA	aa	8-bit PC relative address [-128,+127]
Bcc	ee	6-bit PC relative address [-32,+31]

Table C-9 REP and DO Instructions

DSP56156		
OPERATION	**OPERAND**	**COMMENTS**
REP **DO**	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	
REPcc	16 conditions	
DO FOREVER		

INSTRUCTIONS

Table C-10 Short Immediate Move Instructions

DSP56156		
OPERATION	**DESTINATION**	**COMMENTS**
MOVE(I) #xx,	X1 X0 Y1 Y0	immediate short 8 bit signed data (data is put in the LSByte)

Table C-11 Move — Program and Control Instructions

DSP56156			
OPERATION	**Source/Dest.**	**Dest./Source**	**COMMENTS**
MOVE(M)	P:(Rn) P:(Rn)+ P:(Rn)- P:(Rn)+Nn P:(R2+xx)	A, A0, B, B0 X0, X1, Y0, Y1	
MOVE(M)	X:(Rn)+ X:(Rn)+Nn	P:(Rn)+ P:(Rn)+Nn	
MOVE(C)	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
MOVE(C)	All registers	All registers	

INSTRUCTIONS

Table C-12 Move Absolute Short and Move Peripheral Instructions

DSP56156			
OPERATION	**Source/Dest.**	**Dest./Source**	**COMMENTS**
MOVE(S)	X:<aa>	A, B, X0, Y0	first 32 word of X memory 5 bit address
MOVE(P)	X:<pp>	A, B, X0, Y0	last 32 word of X memory 5 bit address
		X:(Rn)+ X:(Rn)+Nn	

Table C-13 Transfer with Parallel Move Instruction

DSP56156				
OPERATION	**REGISTER TRANSFER**		**PARALLEL MOVE**	
	Source	**Dest.**	**Source/Dest.**	**Dest./Source**
TFR(3)	A B	X0, X1, Y0, Y1	X:(Rn)+ X:(Rn)+Nn	X0,X1,Y0,Y1, A0, B0, A, B

Table C-14 Register Transfer Without Parallel Move Instruction

DSP56156		
OPERATION	**SOURCE**	**DESTINATION**
TFR(2)	A B	X Y

INSTRUCTIONS

Table C-15 Register Transfer Conditional Move Instruction

DSP56156		
OPERATION	**Data ALU**	**Address Reg.**
Tcc	A, F B, F Y0, F X0, F	R0,R0
		R0,Rm

Table C-16 Conditional Program Controller Instructions

DSP56156
OPERATION
BRKcc
DEBUGcc

Table C-17 Logical Immediate Instructions

DSP56156		
OPERATION	**DESTINATION**	**COMMENTS**
ORI #xx, **ANDI #xx,**	CCR MR OMR	8 bit immediate data

INSTRUCTIONS

Table C-18 Double Precision Data Alu Instructions

DSP56156		
DATA ALU OPERATION		
Operation	**sign**	**unsign**
DMAC	Y1, X0, F X1, Y1, F X1, Y0, F X0, Y0, F	
MPY(su,uu) **MAC(su,uu)**		

Table C-19 Integer Data ALU Instructions

DSP56156	
DATA ALU OPERATION	
Operation	
IMAC **IMPY**	X0,X0,F X1,X0,F A1,Y0,F B1,X0,F Y0,X1,F Y1,X1,F Y1,X0,F Y0,X0,F

Table C-20 Division Instruction

DSP56156	
DATA ALU OPERATION	
Operation	
DIV	X1,F X0,F Y1,F Y0,F

INSTRUCTIONS

Table C-21 Other Data ALU Instructions

DSP56156		
OPERATION		
Norm	Rn,F	n=[0,3]
TST2	X1,X0,Y1,Y0	Test data registers
ADC	X,**F** **Y,F**	
CHKAAU		Set V,N,Z according to last address ALU operation
ZERO	F	zero F from bit 32 to 39
EXT	F	s. ext. F from bit 31 to 39
SWAP	F	swap F1 and F0
NEGC	F	negate with borrow
ASL4	F	
ASR4	F	
ASR16	F	move A,A0 arithmetic

INSTRUCTIONS

Table C-22 Special Instructions

DSP56156
OPERATION
WAIT
STOP
ENDDO
RESET
RTS
RTI
SWI
DEBUG
NOP

Application: _____ Date: _____

_____ Programmer: _____

Sheet 1 of 3

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 = $\overline{BR}$ Asserted By External Access
1 = $\overline{BR}$ 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
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				0				0			0				

* = Reserved, Program as zero

Figure C-2 Bus Control Register (BCR)

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												

* = Reserved, Program as zero

MR **CCR**

Figure C-3 Status Register (SR)

Application: _____ Date: _____

_____ Programmer: _____

CORE

$\overline{\text{IRQA}}$ 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

$\overline{\text{IRQB}}$ 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

* = Reserved, Program as zero

Figure C-4 Interrupt Priority Register (IPR)

Application: _____ Date: _____

_____ Programmer: _____

CORE

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

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 CLKO 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	*	MC	MB	MA
0	0	0	0	0	0	0	0					0			
0				0											

Figure C-5 Operating Mode Register (OMR)

* = Reserved, Program as zero

Application: _____ Date: _____

_____ Programmer: _____

Sheet 1 of 1

P.L.L.

Input Divider
Divides Clock Frequency by 1 to 16

Feedback Divider
Multiplies Clock Frequency by 4 to 64; Increments of 4

Clockout Select

CS1	CS0	CLKOUT
0	0	PH0
0	1	Reserved
1	0	Squared F_{ext}
1	1	Squared $F_{ext} \prod \pi \div 2$

GSM (Codec Clock Source)
0 = ÷Input Divider
1 = ÷6.5

PLL Power Down
0 = Off
1 = On

PLL Enable
0 = Disable
1 = Enable

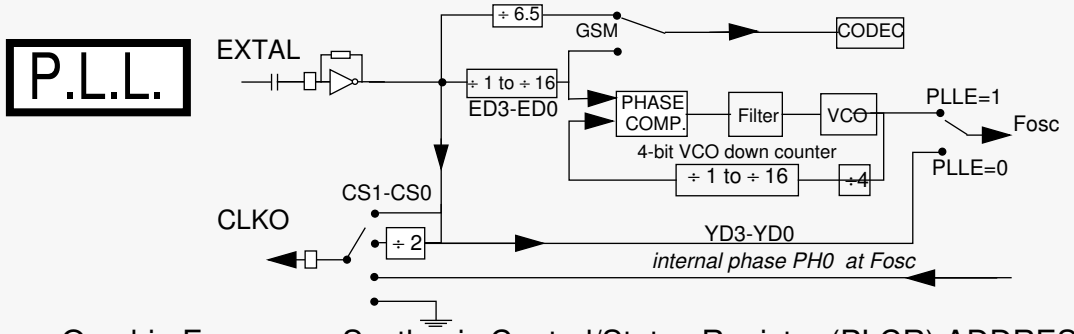
VCO Lock – Read Only
0 = NOT Locked
1 = Locked

PLL Control Register (PLCR)
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	GSM	CS1	CS0	*	*	YD3	YD2	YD1	YD0	ED3	ED2	ED1	ED0
						0	0								

Figure C-6 PLL Control Register (PLCR)

* = Reserved, Program as zero



On-chip Frequency Synthesis Control/Status Register (PLCR) ADDRESS X:$FFDC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	PLLE	PLLD	GSM	CS1	CS0	**	**	YD3	YD2	YD1	YD0	ED3	ED2	ED1	ED0

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
GSM	0	Output of the 4 bit divider selected as codec clock input
	1	Output of the 6.5 divider selected as codec clock input
CS1-CS0 CLKO Select	0	PH0 output to CLKO when enabled by the CD bit (bit 7) of the OMR
	1	reserved
	2	Fext output to CLKO when enabled by the CD bit (bit 7) of the OMR
	3	Fext/2 output to CLKO when enabled by the CD bit (bit 7) of the OMR
ED3-ED0 Clock Input Divider	$0	Divide the input clock by 1
	$1	Divide the input clock by 2
	$2	Divide the input clock by 3
	$3	Divide the input clock by 4
	$4	Divide the input clock by 5
	$5	Divide the input clock by 6
	$6	Divide the input clock by 7
	$7	Divide the input clock by 8
	$8	Divide the input clock by 9
	$9	Divide the input clock by 10
	$A	Divide the input clock by 11
	$B	Divide the input clock by 12
	$C	Divide the input clock by 13
	$D	Divide the input clock by 14
	$E	Divide the input clock by 15
	$F	Divide the input clock by 16
YD3-YD0 VCO Down Counter	$0	PLL multiplies the clock by 4
	$1	PLL multiplies the clock by 8
	$2	PLL multiplies the clock by 12
	$3	PLL multiplies the clock by 16
	$4	PLL multiplies the clock by 20
	$5	PLL multiplies the clock by 24
	$6	PLL multiplies the clock by 28
	$7	PLL multiplies the clock by 32
	$8	PLL multiplies the clock by 36
	$9	PLL multiplies the clock by 40
	$A	PLL multiplies the clock by 44
	$B	PLL multiplies the clock by 48
	$C	PLL multiplies the clock by 52
	$D	PLL multiplies the clock by 56
	$E	PLL multiplies the clock by 60
	$F	PLL multiplies the clock by 64

Figure C-7 On-chip Frequency Synthesizer Programming Model Summary

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

Figure C-8 Timer Control Register (TCR)

Timer

Count — Decremented when TPR = 0

Timer Count Register (TCTR)
X:$FFED Read/Write
Reset = $0000

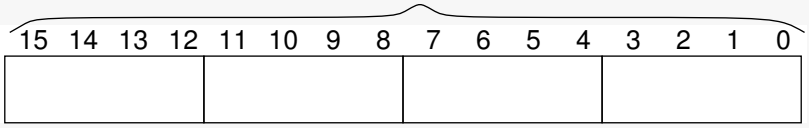


Figure C-9 Timer Count Register (TCTR)

Compare Value — Compare with Count Register

Timer Compare Register (TCPR)
X:$FFEE Read/Write
Reset = $0000

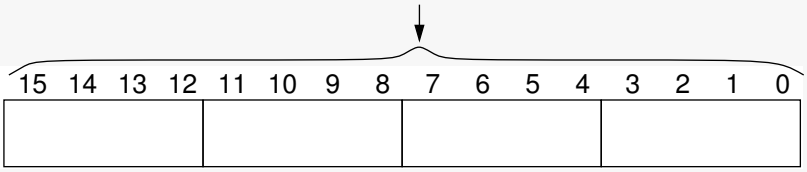


Figure C-10 Timer Compare Register (TCPR)

Number to Load into Count Register

Timer Preload Register (TPR)
X:$FFEF Read/Write
Reset = $0000

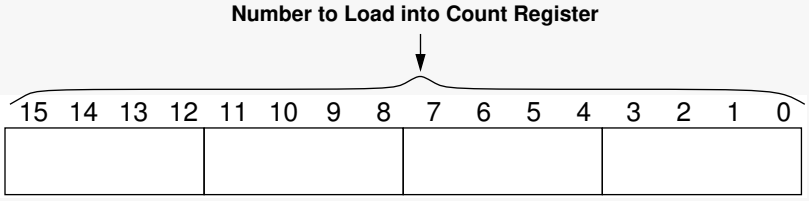


Figure C-11 Timer Preload Register (TPR)

Application: _____ Date: _____

_____ Programmer: _____

Codec

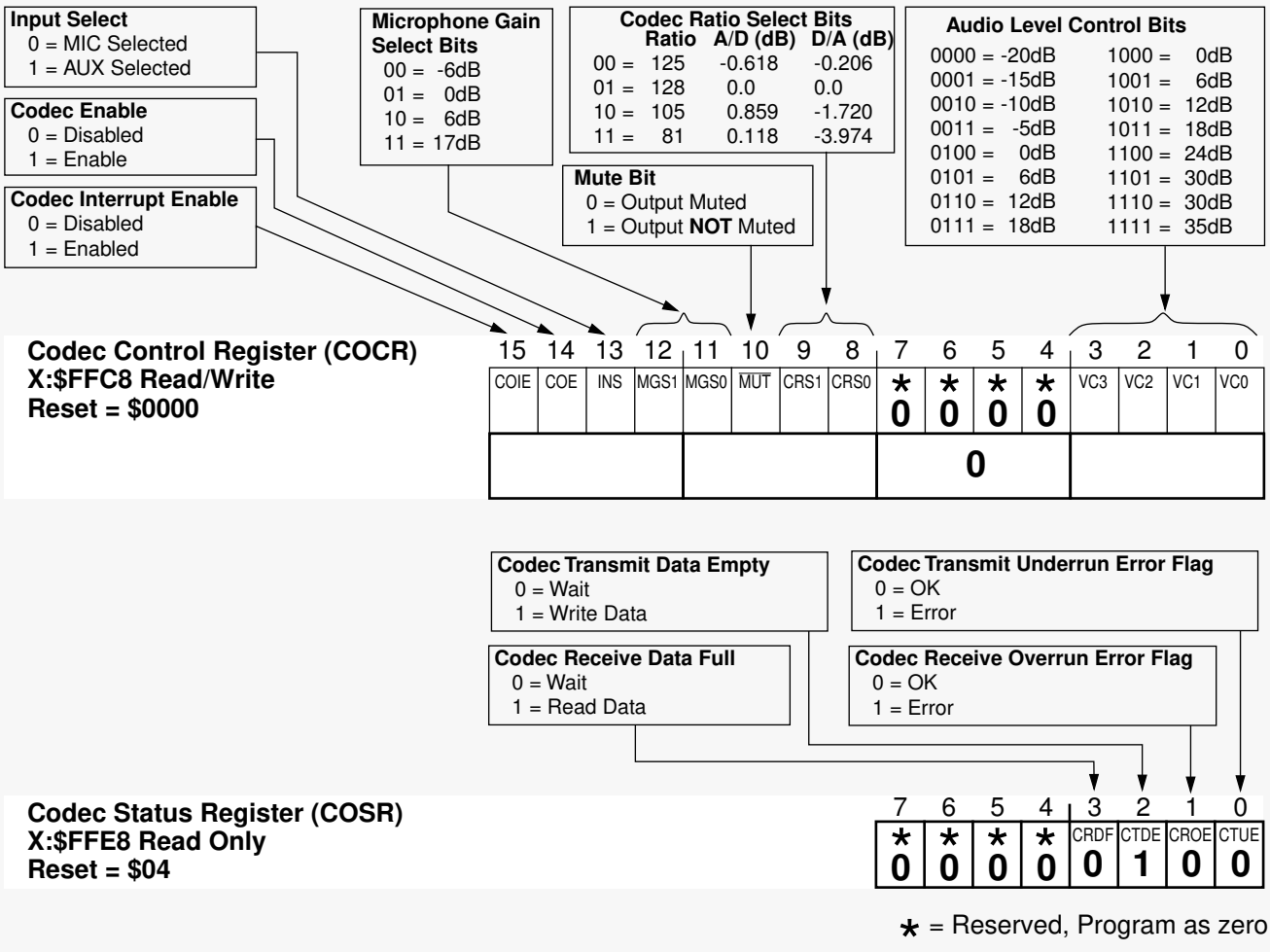


Figure C-12 Codec Status Register (COSR)

Codec

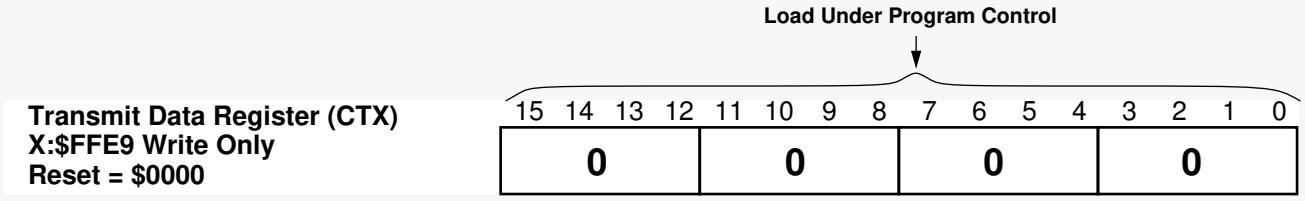


Figure C-13 Transmit Data Register (CTX)

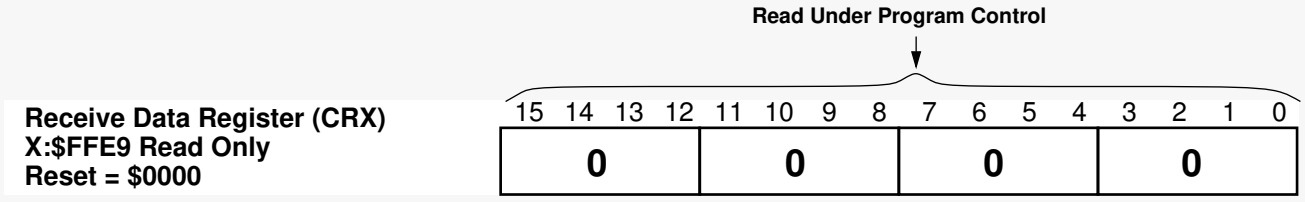


Figure C-14 Receive Data Register (CRX)

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			

* = Reserved, Program as zero

Figure C-15 Control Register (PBC)

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															

* = Reserved, Program as zero

Figure C-16 Data Direction Register (PBDDR)

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

Figure C-17 Data Register (PBD)

Application: _____ Date: _____

_____ Programmer: _____

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	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
0	0	0	0												
0															

* = Reserved, Program as zero

Figure C-18 Control Register (PCC)

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	DC8	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
0	0	0	0												
0															

* = Reserved, Program as zero

Figure C-19 Data Direction Register (PCDDR)

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	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0												
0															

* = Reserved, Program as zero

Figure C-20 Data Register (PCD)

Application: _____ Date: _____

_____ Programmer: _____

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			

* = Reserved, Program as zero

Figure C-21 Control Register (PBC)

HOST – DSP SIDE

Host Receive Interrupt Enable
0 = ∏ Disable1 = ∏ Enable — Interrupt on HRDF

Host Transmit Interrupt Enable
0 = ∏ Disable1 = ∏ Enable — Interrupt on HTDE

Host Command Interrupt Enable
0 = ∏ Disable1 = ∏ 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

Figure C-22 Host Control Register (HCR)

Application: _____ Date: _____

_____ Programmer: _____

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					

* = Reserved, Program as zero

Figure C-23 Host Status Register (HSR)

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							

Figure C-24 Host Receive Data Register

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							

Figure C-25 Host Transmit Data Register (HTX)

Application: _____ Date: _____

_____ Programmer: _____

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		

* = Reserved, Program as zero

Figure C-26 Interrupt Control Register (ICR)

Host Vector
Executive Interrupt Routine 0-31

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	*	*	HV4	HV3	HV2	HV1	HV0
	0	0					

* = Reserved, Program as zero

Figure C-27 Command Vector Register (CVR)

Application: _____ Date: _____

_____ Programmer: _____

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 = ∏ $\overline{\text{HREQ}}$ Deasserted 1 = ∏ $\overline{\text{HREQ}}$ Asserted

Interrupt Status Register (ISR)
$2 Read/Write
Reset = $03

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

* = Reserved, Program as zero

Figure C-28 Interrupt Status Register (ISR)

HOST HOST – HOST PROCESSOR SIDE

Interrupt Vector Register (IVR)
$3 Read/Write
Reset = $0F

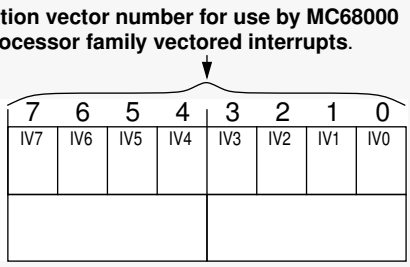


Figure C-29 Interrupt Vector Register (IVR)

Receive Byte Registers
$6, $7 Read Only
Reset = $xxxx

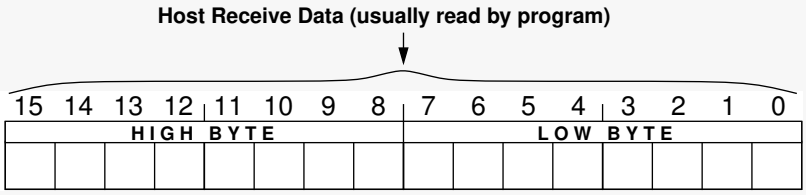


Figure C-30 Receive Byte Registers

Transmit Byte Registers
$6, $7 Write Only
Reset = $xxxx

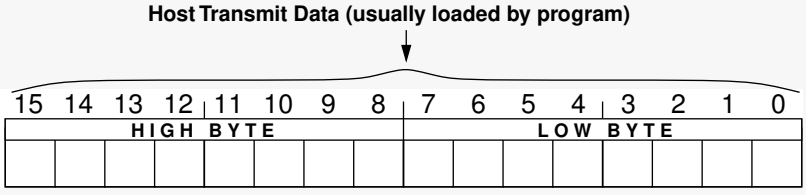


Figure C-31 Transmit Byte Registers

Application: _____ Date: _____

_____ Programmer: _____

SSI

SSI Port C Pin Control
0 = General Purpose I/O Pin
1 = SSI Pin

PORT C
SSI 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	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
0	0	0	0												
0															

* = Reserved, Program as zero

Figure C-32 SSI Control Register (PCC)

SSI Receive Data (usually read by program)

SSI Serial Receive Register
SSI0 Address X:$FFF1 Read Only
SSI1 Address X:$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							

Figure C-33 SSI Serial Receive Register

SSI Transmit Data (usually loaded by program)

SSI Serial Transmit Register
SSI0 Address X:$FFF1 Write Only
SSI1 Address X:$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							

Figure C-34 SSI Serial Transmit Register

Application: _____ Date: _____

_____ Programmer: _____

SSI

SSI Receive Slot Mask
RSMAx
SSI0: $FFF2 Read/Write
SSI1: $FFFA Read/Write
Reset = $FFFF

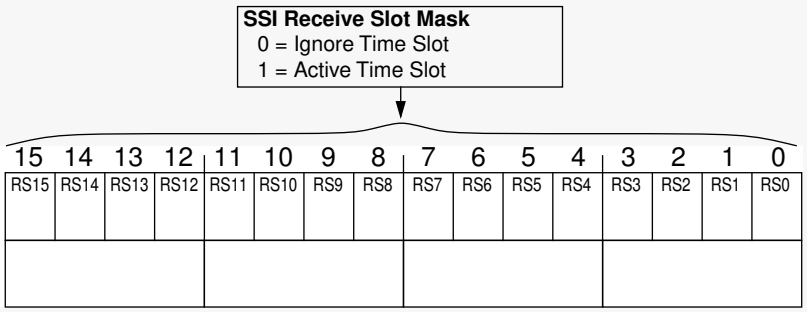


Figure C-35 SSI Receive Slot Mask

SSI Receive Slot Mask
RSMBx
SSI0: $FFF3 Read/Write
SSI1: $FFFB Read/Write
Reset = $FFFF

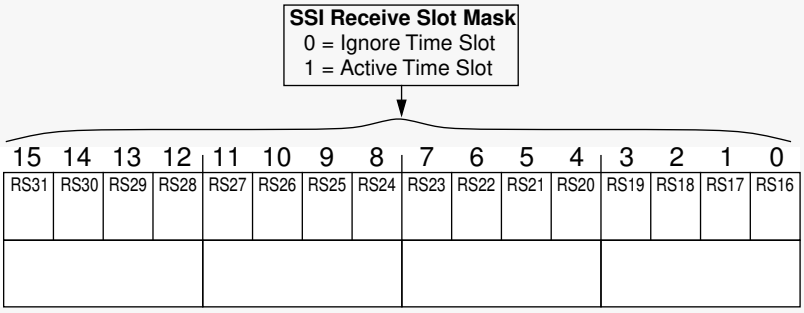
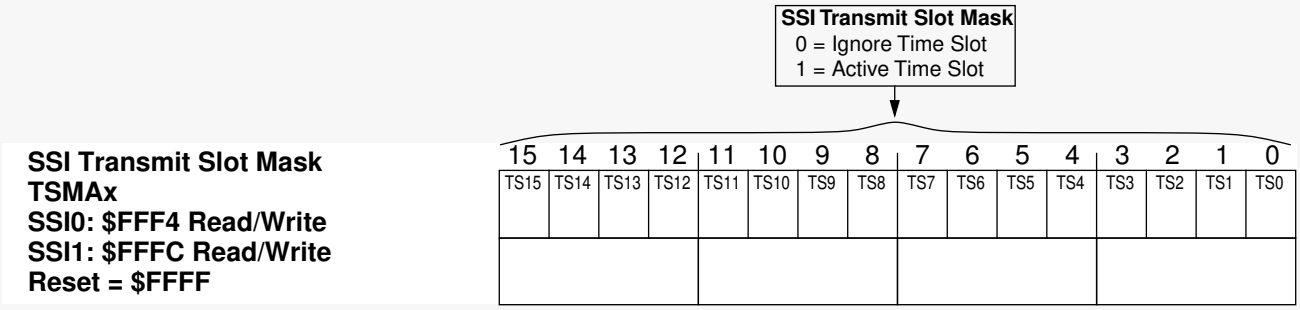


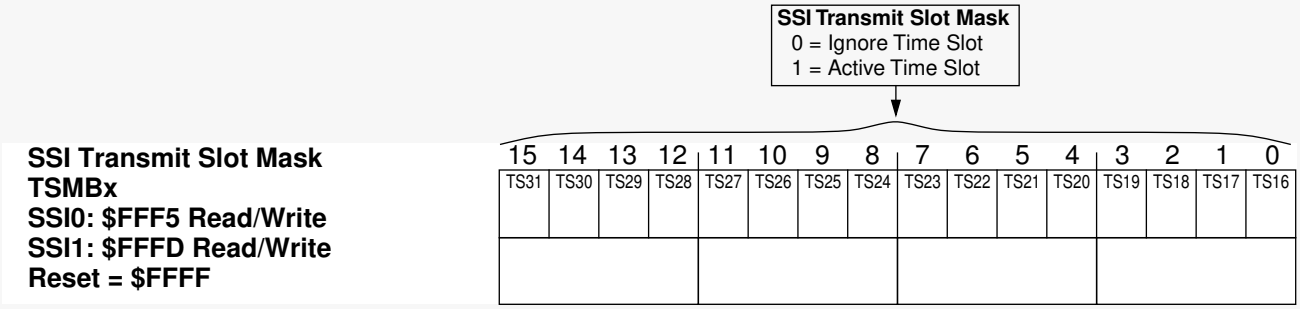
Figure C-36 SSI Receive Slot Mask

SSI



SSI Transmit Slot Mask
TSMAx
SSI0: $FFF4 Read/Write
SSI1: $FFFC Read/Write
Reset = $FFFF

Figure C-37 SSI Transmit Slot Mask



SSI Transmit Slot Mask
TSMBx
SSI0: $FFF5 Read/Write
SSI1: $FFFD Read/Write
Reset = $FFFF

Figure C-38 SSI Transmit Slot Mask

Application: _____ Date: _____

_____ Programmer: _____

SSI

Word Length Control
00 = 8 Bits/Word
01 = 8 with Log Exp/Comp
10 = 12 Bits/Word
11 = 16 Bits/Word

Prescaler Range
0 = / 1
1 = / 8

Frame Rate Divider Control
00000 = 1
11111 = 32

Prescale Modulus Select

SSI Control Register A (CRA)
SSI0 Address $FFD0 Read/Write
SSI1 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	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

Figure C-39 SSI Control Register A (CRA)

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

Sync/Async
0 = ∏ Async 1 = Sync

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

Frame Sync Directions

SYN	FSD0	FSD1	Mode	SC1x	SC0x
0	0	0	Async	RFS in	TFS in
0	0	1	Async	RFS out	TFS out
0	1	0	Async	RFS in	TFS out
0	1	1	—	—	—
1	0	0	Sync	—	FS in
1	0	1	Sync	—	FS out
1	1	0	—	—	—
1	1	1	Sync	F1 out	F0 out

A/Mu Law Selection
Iff WL1 = 0 and WL0 = 1
0 = A Law
1 = μ Law

Serial Output Flags
Enabled when SYN = 1
and FSDx = 1

SSI Control Register B (CRB)
SSI0 Address $FFD1 Read/Write
SSI1 Address $FFD9 Read/Write
Reset = $0000

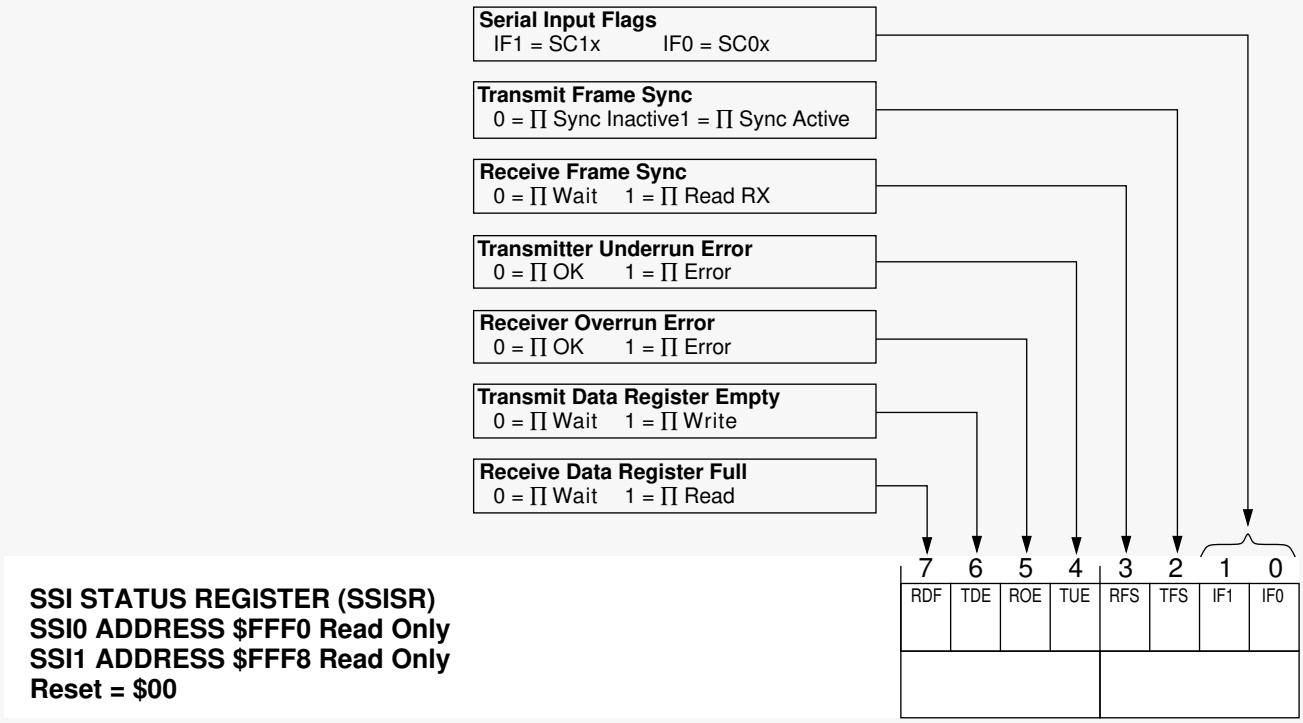
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD	SYN	FSI	FSL	SHFD	SCKP	SCKD	FSD1	A/MU	FSD0	OF1	OF0

Figure C-40 SSI Control Register B (CRB)

Application: _____ Date: _____

_____ Programmer: _____

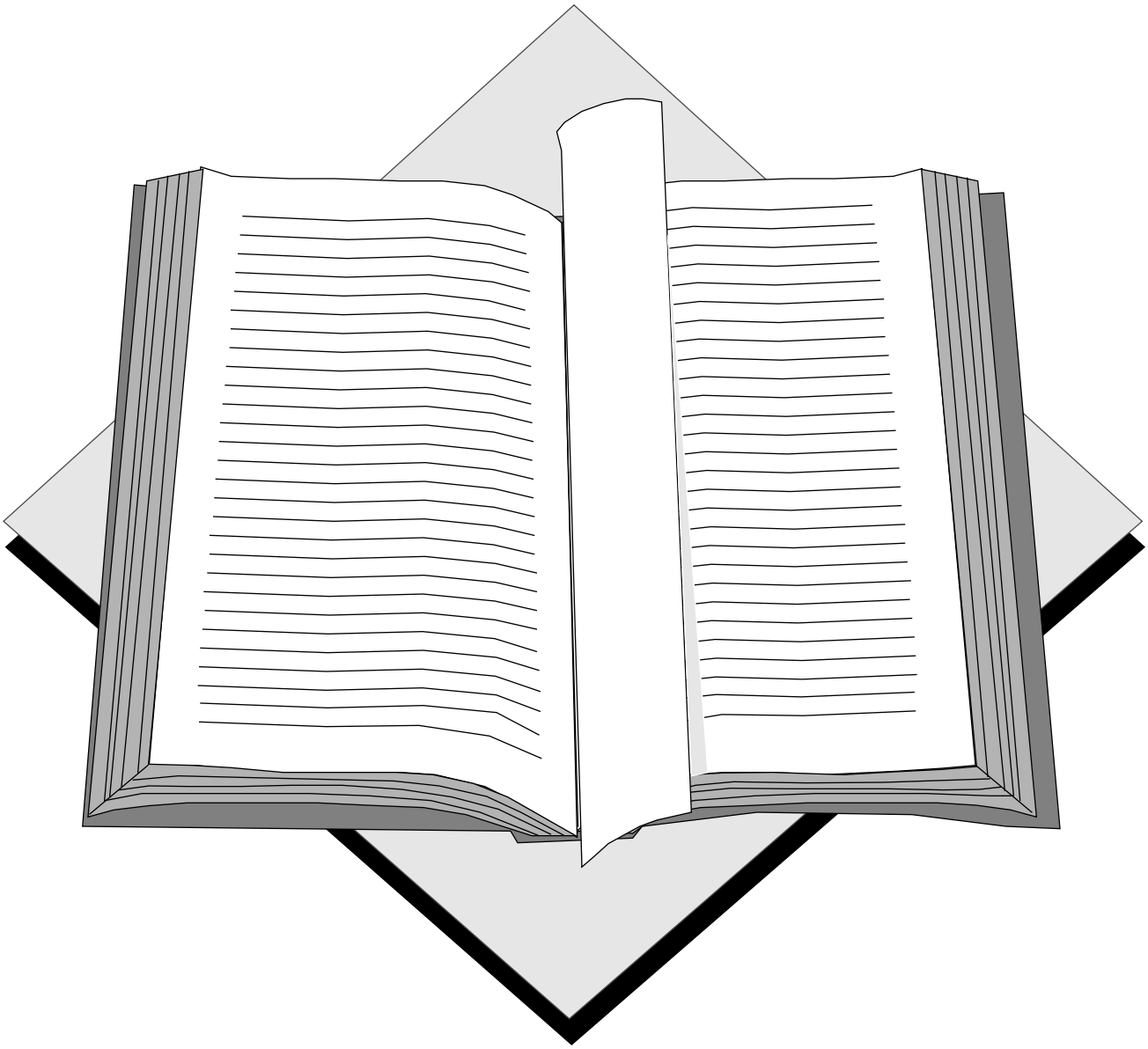
SSI



SSI STATUS REGISTER (SSISR)
SSI0 ADDRESS $FFF0 Read Only
SSI1 ADDRESS $FFF8 Read Only
Reset = $00

Figure C-41 SSI Status Register (SSISR)

INDEX



INDEX

—D—

—R—

—S—

—T—

—V—

—W—

—X—

—Y—