Order this document
by DSP56001/D

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

DSP56001

24-Bit General Purpose Digital Signal Processor

Pin Grid Array (PGA)
Available in an 88 pin ceramic through-hole package.

Ceramic Quad Flat Pack (CQFP)
Available in a 132 pin, small footprint, surface mount package.

Plastic Quad Flat Pack (PQFP)
Available in a 132 pin, small footprint, surface mount package (Not shown with the Molded Carrier Ring).

The DSP56001 is a member of Motorola's family of HCMOS, low-power, general purpose Digital Signal Processors. The DSP56001 features 512 words of full speed, on-chip program RAM (PRAM) memory, two 256 word data RAMs, two preprogrammed data ROMs, and special on-chip bootstrap hardware to permit convenient loading of user programs into the program RAM. It is an off-the-shelf part since the program memory is user programmable. The core of the processor consists of three execution units operating in parallel — the data ALU, the address generation unit, and the program controller. The DSP56001 has MCU-style on-chip peripherals, program and data memory, as well as a memory expansion port. The MPU-style programming model and instruction set make writing efficient, compact code, straightforward.

The high throughput of the DSP56001 makes it well-suited for communication, high-speed control, numeric processing, computer and audio applications. The key features which facilitate this throughput are:

- **Speed** — At 13.5 million instructions per second (MIPS) with a 27 MHz clock, the DSP56001 can execute a 1024 point complex Fast Fourier Transform in 2.45 milliseconds.
- **Precision** — The data paths are 24 bits wide thereby providing 144 dB of dynamic range; intermediate results held in the 56-bit accumulators can range over 336 dB.
- **Parallelism** — The data ALU, address arithmetic units, and program controller operate in parallel so that an instruction prefetch, a 24x24-bit multiplication, a 56-bit addition, two data moves, and two address pointer updates using one of three types of arithmetic (linear, modulo, or reverse carry) can be executed in a single instruction cycle. This parallelism allows a four coefficient Infinite Impulse Response (IIR) filter section to be executed in only four cycles, the theoretical minimum for a single multiplier architecture.
- **Integration** — In addition to the three independent execution units, the DSP56001 has six on-chip memories, three on-chip MCU style peripherals (Serial Communication Interface, Synchronous Serial Interface, and Host Interface), a clock generator and seven buses (three address and four data), making the overall system functionally complete and powerful, but also very low cost, low power, and compact.
- **Invisible Pipeline** — The three-stage instruction pipeline is essentially invisible to the programmer thus allowing straightforward program development in either assembly language or a high-level language such as ANSI C.
- **Instruction Set** — The 62 instruction mnemonics are MCU-like making the transition from programming microprocessors to programming the DSP56001 digital signal processor as easy as possible. The orthogonal syntax supports control of the parallel execution units. This syntax provides 12,808,830 different instruction variations using the 62 instruction mnemonics. The no-overhead DO instruction and the REPEAT (REP) instruction make writing straight-line code obsolete.
- **DSP56000/DSP56001 Compatibility** — The DSP56001 is identical to the DSP56000 except that it has 512x24-bits of on-chip program RAM instead of 3.75K of program ROM; a 32x24-bit bootstrap ROM for loading the program RAM from either a byte-wide memory mapped ROM or via the Host Interface; and the on-chip X and Y Data ROMs have been preprogrammed as positive Mu- and A-Law to linear expansion tables and a full, four quadrant sine wave table, respectively.
- **Low Power** — As a CMOS part, the DSP56001 is inherently very low power; however, three other features can reduce power consumption to an exceptionally low level.
  — The WAIT instruction shuts off the clock in the central processor portion of the DSP56001.
  — The STOP instruction halts the internal oscillator.
  — Power increases linearly (approximately) with frequency; thus, reducing the clock frequency reduces power consumption.

This document contains information on a new product. Specifications and information herein are subject to change without notice.

SIGNAL DESCRIPTION

The DSP56001 is an 88-pin integrated circuit available in surface mount (CQFP and PQFP) or pin-grid array packaging. Its input and output signals are organized into seven functional groups which are listed below and shown in Figure 1.

Port A Address and Data Buses
Port A Bus Control
Interrupt and Mode Control
Power and Clock
Host Interface or Port B I/O
Serial Communications Interface or Port C I/O
Synchronous Serial Interface or Port C I/O

PORT A ADDRESS AND DATA BUS

Address Bus (A0-A15)

These three-state output pins specify the address for external program and data memory accesses. To minimize power dissipation, A0-A15 do not change state when external memory spaces are not being accessed.

Data Bus (D0-D23)

These pins provide the bidirectional data bus for external program and data memory accesses. D0-D23 are in the high-impedance state when the bus grant signal is asserted.

PORT A BUS CONTROL

Program Memory Select ($\overline{PS}$)

This three-state output is asserted only when external program memory is referenced.

Data Memory Select ($\overline{DS}$)

This three-state output is asserted only when external data memory is referenced.

X/$\overline{Y}$ Select (X/$\overline{Y}$)

This three-state output selects which external data memory space (X or Y) is referenced by data memory select ($\overline{DS}$).

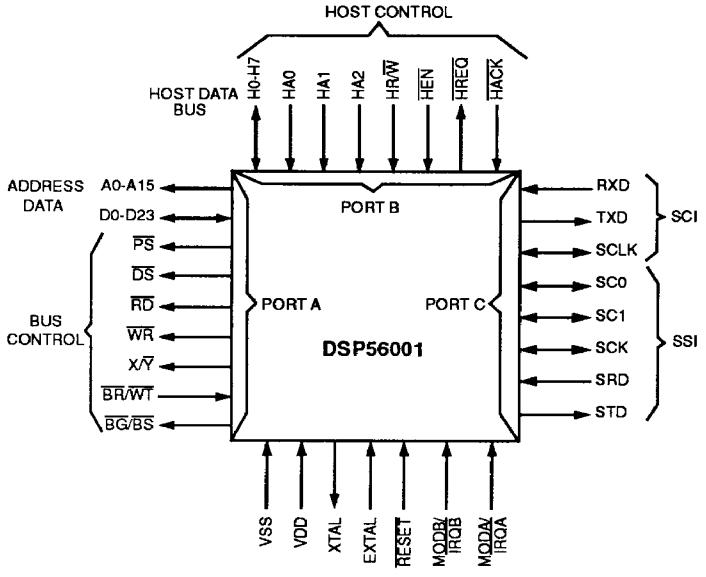


Figure 1. Functional Signal Groups

Read Enable ($\overline{RD}$)

This three-state output is asserted to read external memory on the data bus D0-D23.

Write Enable ($\overline{WR}$)

This three-state output is asserted to write external memory on the data bus D0-D23.

Bus Request ($\overline{BR}$/$\overline{WT}$)

The bus request input $\overline{BR}$ allows another device such as a processor or DMA controller to become the master of external data bus D0-D23 and external address bus A0-A15. When operating mode register (OMR) bit 7 is clear and $\overline{BR}$ is asserted, the DSP56001 will always release the external data bus D0-D23, address bus A0-A15, and bus control pins $\overline{PS}$, $\overline{DS}$, X/$\overline{Y}$, $\overline{RD}$, and $\overline{WR}$ (i. e., Port A), by placing these pins in the high-impedance state after execution of the current instruction has been completed. **The $\overline{BR}$ pin should be pulled up when not in use.**

If OMR bit 7 is set, this pin is an input that allows an external device to force wait states during an external Port A operation for as long as $\overline{WT}$ is asserted.

Bus Grant ($\overline{BG}$/$\overline{BS}$)

If OMR bit 7 is clear, this output is asserted to acknowledge an external bus request after Port A has been released. If OMR bit 7 is set, this pin is bus strobe and is asserted when the DSP accesses Port A.

INTERRUPT AND MODE CONTROL

Mode Select A/External Interrupt Request A (MODA/$\overline{IRQA}$),

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

These two inputs have dual functions: 1) to select the initial chip operating mode and 2) to receive an interrupt request from an external source. MODA and MODB are read and internally latched in the DSP when the processor exits the RESET state. Therefore these two pins should be forced into the proper state during reset. After leaving the RESET state, the MODA and MODB pins automatically change to external interrupt requests $\overline{IRQA}$ and $\overline{IRQB}$. After leaving the reset state the chip operating mode can be changed by software. $\overline{IRQA}$ and $\overline{IRQB}$ may be programmed to be level sensitive or negative edge triggered. When edge triggered, triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal, however, the probability of noise on $\overline{IRQA}$ or $\overline{IRQB}$ generating multiple interrupts increases with increasing fall time of the interrupt signal.

Reset ($\overline{RESET}$)

This Schmitt trigger input pin is used to reset the DSP56001. When $\overline{RESET}$ is asserted, the DSP56001 is initialized and placed in the reset state. When the $\overline{RESET}$ signal is deasserted, the initial chip operating mode is latched from the MODA and MODB pins. When coming out of reset, deassertion occurs at a voltage level and is not directly related to the rise time of the reset signal; however, the probability of noise on $\overline{RESET}$ generating multiple resets increases with increasing rise time of the reset signal.

POWER AND CLOCK

Power (V_{CC}), Ground (GND)

There are five sets of power and ground pins, two pairs for internal logic, one power and two ground for Port A address and control pins, one power and two ground for Port A data pins, and one pair for peripherals. Refer to the pin assignments in the **LAYOUT PRACTICES** section.

External Clock/Crystal Input (EXTAL)

EXTAL may be used to interface the crystal oscillator input to an external crystal or an external clock.

Crystal Output (XTAL)

This output connects the internal crystal oscillator output to an external crystal. If an external clock is used, XTAL should not be connected.

HOST INTERFACE

Host Data Bus (H0-H7)

This bidirectional data bus is used to transfer data between the host processor and the DSP56001. 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 is not being used.

Host Address (HA0-HA2)

These inputs provide the address selection for each Host Interface register. HA0-HA2 may be programmed as general purpose parallel I/O pins called PB8-PB10 when the Host Interface is not being used.

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

This input selects the direction of data transfer for each host processor access. HR/$\overline{W}$ may be programmed as a general purpose I/O pin called PB11 when the Host Interface is not being used.

Host Enable ($\overline{HEN}$)

This input enables a data transfer on the host data bus. When $\overline{HEN}$ is asserted and HR/$\overline{W}$ is high, H0-H7 become outputs, and DSP56001 data may be read by the host processor, When $\overline{HEN}$ is asserted and HR/$\overline{W}$ is low, H0-H7 become inputs and host data is latched inside the DSP when $\overline{HEN}$ is deasserted. Normally a chip select signal, derived from host address decoding and an enable clock, is used to generate $\overline{HEN}$. $\overline{HEN}$ may be programmed as a general purpose I/O pin called PB12 when the Host Interface is not being used.

Host Request ($\overline{HREQ}$)

This open-drain output signal is used by the DSP56001 Host Interface to request service from the host processor, DMA controller, or simple external controller. $\overline{HREQ}$ may be programmed as a general purpose I/O pin (not open-drain) called PB13 when the Host interface is not being used. $\overline{HREQ}$ should be pulled high when not in use.

Host Acknowledge ($\overline{HACK}$)

This input has two functions: 1) to receive a Host Acknowledge handshake signal for DMA transfers and, 2) to receive a Host Interrupt Acknowledge compatible with MC68000 Family processors. $\overline{HACK}$ may be programmed as a general purpose I/O pin called PB14 when the Host Interface is not being used. **$\overline{HACK}$ should be pulled high when not in use.**

SERIAL COMMUNICATIONS INTERFACE (SCI)

Receive Data (RXD)

This input receives byte-oriented data into the SCI Receive Shift Register. Input data is sampled on the positive edge of the Receive Clock. RXD may be programmed as a general purpose I/O pin called PC0 when the SCI is not being used.

Transmit Data (TXD)

This output transmits serial data from the SCI Transmit Shift Register. Data changes on the negative edge of the transmit clock. This output is stable on the positive edge of the transmit clock. TXD may be programmed as a general purpose I/O pin called PC1 when the SCI is not being used.

SCI Serial Clock (SCLK)

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

SYNCHRONOUS SERIAL INTERFACE (SSI)

Serial Control Zero (SC0)

This bidirectional pin is used for control by the SSI. SC0 may be programmed as a general purpose I/O pin called PC3 when the SSI is not being used.

Serial Control One (SC1)

This bidirectional pin is used for control by the SSI. SC1 may be programmed as a general purpose I/O pin called PC4 when the SSI is not being used.

Serial Control Two (SC2)

This bidirectional pin is used for control by the SSI. SC2 may be programmed as a general purpose I/O pin called PC5 when the SSI is not being used.

SSI Serial Clock (SCK)

This bidirectional pin provides the serial bit rate clock for the SSI when only one clock is used. SCK may be programmed as a general purpose I/O pin called PC6 when the SSI is not being used.

SSI Receive Data (SRD)

This input pin receives serial data into the SSI Receive Shift Register. SRD may be programmed as a general purpose I/O pin called PC7 when the SSI is not being used.

SSI Transmit Data (STD)

This output pin transmits serial data from the SSI Transmit Shift Register. STD may be programmed as a general purpose I/O pin called PC8 when the SSI is not being used.

BLOCK DIAGRAM DESCRIPTION

The DSP56001 architecture has been designed to maximize throughput in data intensive Digital Signal Processing (DSP) applications. This objective resulted in a dual-natured, expandable architecture with sophisticated on-chip peripherals and general purpose I/O. It is dual-natured in that there are two independent expandable data memory spaces, two address arithmetic units, and a data ALU which has two accumulators and two shifter/limiters. The duality of the architecture makes it easier to write software for DSP applications. For example, data is naturally partitioned into X and Y spaces for graphics and image processing applications, into coefficient and data spaces for filtering and transformations, and Into real and imaginary spaces for performing complex arithmetic.

The major components of the DSP56001 are:

- Data Buses
- Address Buses
- Data ALU
- Address Generation Unit
- X Data Memory
- Y Data Memory
- Program Memory
- Bootstrap ROM
- Program Control Unit
- Input/Output
  - Expansion Port
  - General Purpose I/O
  - Host Interface
  - Serial Communications Interface
  - Synchronous Serial Interface

These components are depicted in Figure 2 and described in the following paragraphs.

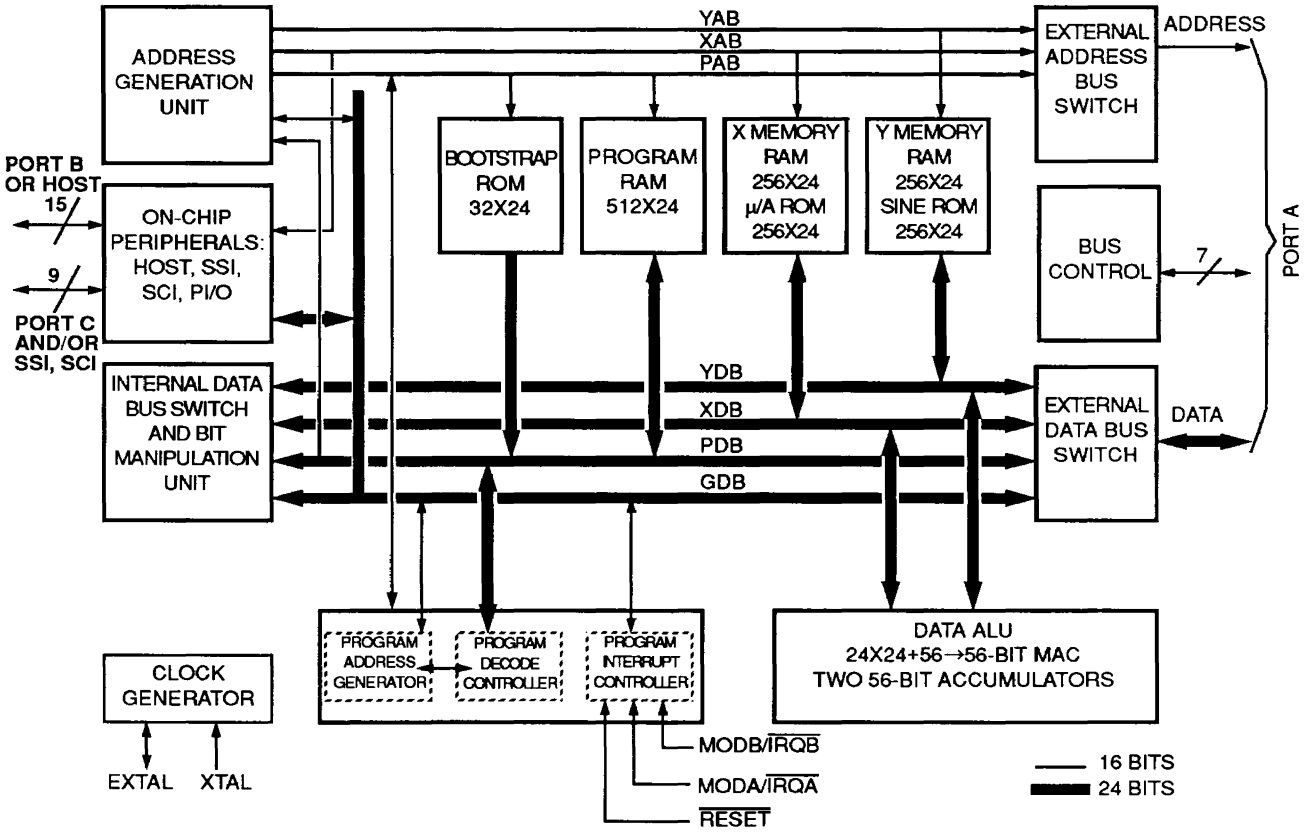


Figure 2. DSP56001 Block Diagram

DATA BUSES

Data movement on the chip occurs over four bidirectional 24-bit buses — the X data bus (XDB), the Y data bus (YDB), and the global data bus (GDB). The XDB and YDB may also be treated by certain instructions as one 48-bit data bus by concatenation of XDB and YDB. Data transfers between and data ALU and the two data memories (X data memory and Y data memory) occur over the XDB and YDB, respectively. The XDB and YDB are kept local on the chip to maximize speed and minimize power dissipation. All other data transfers such as I/O transfers to peripherals occur over the GDB. Instruction word prefetches take place in parallel over the PDB. Transfers between buses are accomplished through the internal bus switch.

ADDRESS BUSES

Addresses are specified for internal X data memory and Y data memory on two unidirectional 16-bit buses — the X address bus (XAB) and the Y address bus (YAB). Program memory addresses are specified on the program address bus (PAB). External memory spaces are addressed via a single 16-bit unidirectional address bus driven by a three input multiplexer that can select either the XAB, YAB, or PAB. There is no speed penalty if only one external memory space is accessed in an instruction. If two or three external memory spaces are accessed in a single instruction, there will be a one or two instruction cycle execution delay, respectively. A bus arbitrator controls external access.

DATA ALU

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

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

The 24-bit data words provide 144 dB of dynamic range. This is sufficient for most real world applications since the majority of A/Ds and D/As are 16 bits or less, and certainly not greater than 24 bits. The 56-bit accumulation internal to the data ALU provides 336 dB of internal dynamic range so that there will be no loss of precision due to intermediate processing.

The data shifter/limiters provide special post processing on data read from the ALU accumulator registers A and B out to the XDB or YDB. Two independent shifter/limiter operations are used — one for the XDB and one for the YDB.

The data shifters are capable of shifting data one bit to the left or one bit to the right as well as passing the data unshifted. Each data shifter has a 24-bit output with overflow indication. The data shifters are controlled by the scaling mode bits in the status register. These shifters permit dynamic scaling of fixed point data without modifying the program code by simply programming the scaling mode bits. This permits block floating-point algorithms to be implemented in a regular fashion. For example, FFT routines can use this feature to selectively scale each butterfly pass.

Saturation arithmetic is provided to minimize errors due to overflow. Overflow occurs when a source operand requires more bits for accurate representation that there are available in the destination. For example, if the source operand were 01.100 (+1.5 decimal) and the destination register were only four bits, the destination register would contain 1.100 (-1.5 decimal) after the transfer assuming signed fractional arithmetic. This is clearly in error. Overflow has occurred. To minimize the error due to overflow, it is preferable to write the maximum (or "limited") value the destination can assume in the destination. In the example, the "limited" value would be 0.111 (+0.875 decimal). This is clearly closer to +1.5 than -1.5 and, therefore, introduces less error.

In the DSP56001, the data ALU accumulators A and B have extension bits. Therefore, when the extension bits are in use and either A or B is the source being read over the XDB or YDB, limiting will occur. In the DSP56001, the limiters place a "limited" value on the XDB or YDB. Limiting is performed on the content of A or B after the content has been shifted in the shifter. There are two limiters. This allows two word operands to be limited independently in the same instruction cycle. The two data limiters can also be combined to form one 48-bit data limiter for long word operands. If the contents of the selected source accumulator can be represented in the destination operand size without overflow (that is, the accumulator extension register is not in use) the data limiter is disabled and the operand is not modified. If the content of the selected source accumulator cannot be represented without overflow in the destination operand size, the data limiter will substitute a "limited" data value having maximum magnitude and the same sign as the source accumulator: 7FFFFF for 24-bit or 7FFFFF FFFFFF for 48-bit positive numbers, 8000 for 24-bit or 800000 000000 for 48-bit negative numbers. The shifter value in the accumulator register itself is not changed and can be reused within the data ALU. When limiting does occur, a flag in the condition code register is set and latched.

ADDRESS GENERATION UNIT

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

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

MEMORIES

Three independent memory spaces of the DSP56001 — X data, Y data, and program, are shown in Figure 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. DE controls the X and Y data memory maps, enabling the internal X and Y data ROMs.

X Data Memory

On-chip X data RAM is a 24-bit wide internal memory which occupies the lowest 256 locations in X memory space. The on-chip X data ROM occupies locations 256 through 511 in X data memory space when enabled by setting DE=1 in the Operating Mode Register. The X data ROM is factory programmed with positive Mu-law and A-law expansion tables (see **APPENDIX C MU-LAW/A-LAW EXPANSION TABLES**) useful in telecommunication applications. The on-chip peripheral registers occupy the top 64 locations. Addresses are received from the XAB, and data transfers to the data ALU occur on the XDB. X memory may be expanded to 64k off-chip.

Y Data Memory

On-chip Y data RAM is a 24-bit wide internal memory which occupies the lowest 256 locations in Y memory space. The on-chip Y data ROM occupies locations 256 through 511 in Y data memory space when enabled by setting DE=1. The Y data ROM is factory programmed with a full, four quadrant, sine wave table (see **APPENDIX D SINE WAVE TABLE**) useful for FFTs, DFTs, and wave form generation. It is recommended that the off-chip peripheral registers be mapped into the top 64 locations to take advantage of the MOVEP instruction. Addresses are received from the YAB and data transfers to the data ALU occur on the YDB. Y memory may be expanded to 64k off-chip.

Program Memory

On-chip program RAM memory (PRAM), consists of a 512 location by 24-bit high speed RAM which is enabled by the MA and MB bits in the Operating Mode register. Addresses are received from the program control logic (usually the program counter) over the PAB. Program memory may be written using MOVEM instructions. The interrupt vectors for the on-chip resources are located in the bottom 64 locations of program memory. Program memory may be expanded to 64k off-chip.

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

The bootstrap mode, described in Appendix A, provides a convenient, low cost method to load the DSP56001 PRAM with a program after power-on reset. It allows loading the PRAM from a single, inexpensive EPROM or via the Host Interface using a host processor as shown in Figures B-1, B-2, and B-3 of **APPENDIX B APPLICATION EXAMPLES**.

Bootstrap ROM

Bootstrap ROM is a 32 location by 24-bit factory programmed ROM which is used only in the bootstrap mode, Operating Mode 1. The Bootstrap ROM is not accessible by the user and is disabled in normal operating modes. Refer to **APPENDIX A BOOTSTRAP OPERATING MODE — OPERATING MODE 1** for a full description of the bootstrap feature of the DSP56001.

PROGRAM CONTROL UNIT

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

INPUT/OUTPUT

The I/O capability of the DSP56001 is extensive and advanced. A variety of system interfacing configurations are facilitated by this I/O structure including multiple DSP56001 systems with or without a host processor, global bus systems with bus arbitration, and many serial configurations, all with minimal glue logic. Each I/O interface has its own control, status, and data registers and is treated as memory-mapped I/O by the DSP56001. Each interface has several dedicated interrupt vector addresses and control bits to enable/disable interrupts (see Figure 4). This minimizes the overhead associated with servicing the device since each interrupt source can have its own service routine. The interrupt vectors can be programmed to one of three maskable priority levels.

Specifically, the I/O structure consists of an extremely flexible expansion port, Port A, and 24 additional I/O pins as well as two general purpose interrupt pins, $\overline{IRQA}$ and $\overline{IRQB}$. The 24 additional pins may be used as general purpose I/O pins, called Port B and Port C or allocated to on-chip peripherals under software control. Three on-chip peripherals are provided on the DSP56001 — an 8-bit parallel Host (MPU/DMA) Interface, a Serial Communications Interface (SCI), and a Synchronous Serial Interface (SSI). Port B is a 15-pin I/O interface which may be used as general purpose I/O pins or as Host Interface pins. Port C is a 9-pin I/O interface which may be used as general purpose I/O pins or as SCI and SSI pins. These interfaces are described in the following paragraphs.

Expansion Port (Port A)

The DSP56001 expansion port is designed to synchronously interface over a common 24-bit data bus with a wide variety of memory and peripheral devices such as high speed static RAMs, slower memory devices, and other DSPs and MPUs in master/slave configurations. This capability is possible because the expansion bus timing is programmable. The expansion bus timing is controlled by a bus control register (BCR). The BCR controls the timing of the bus interface signals $\overline{RD}$ and $\overline{WR}$, and the data lines. Each of four memory spaces X data, Y data, Program data, and I/O has its own 4-bit BCR which can be pro-

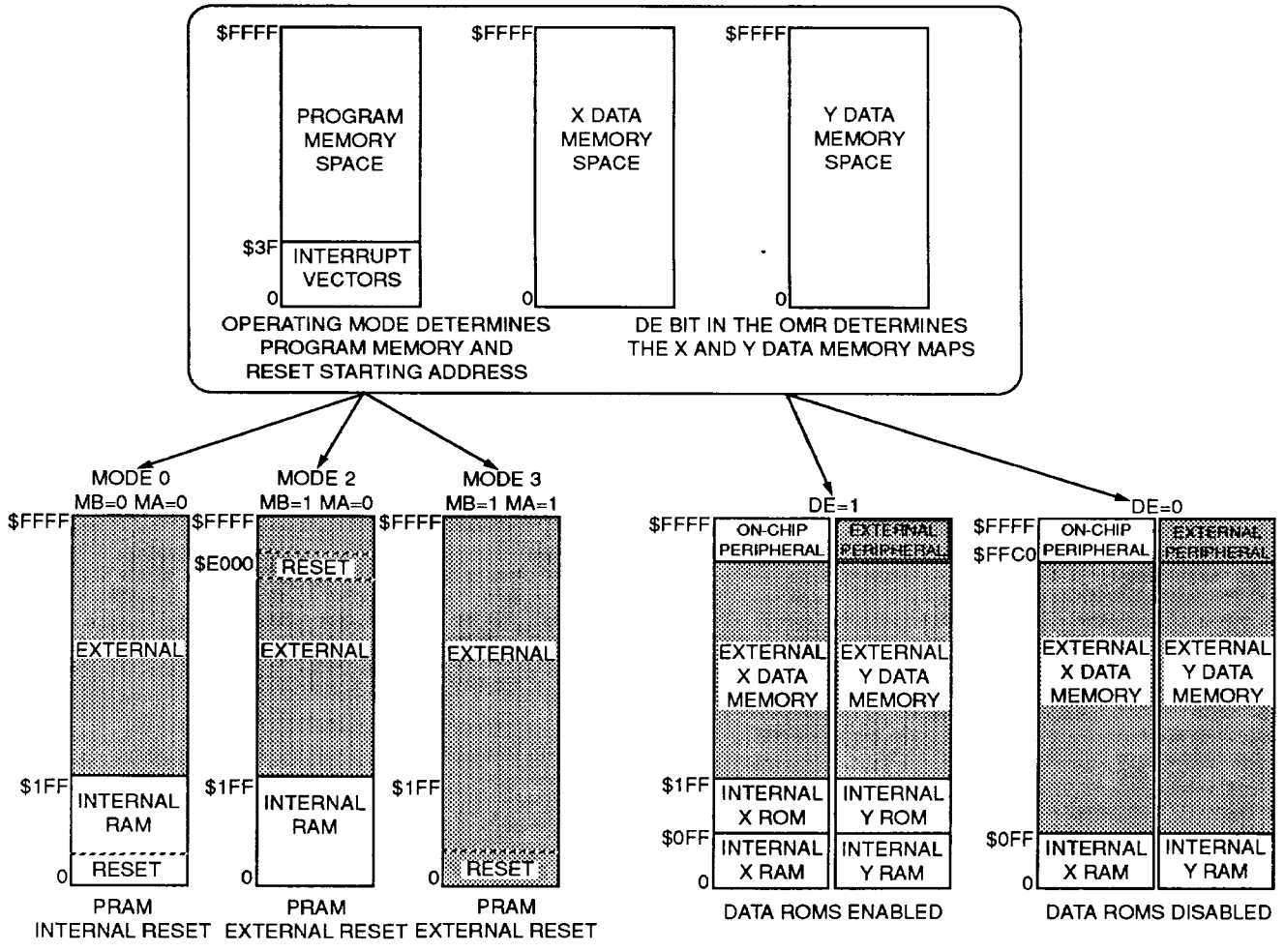


Figure 3. DSP56001 Memory Map

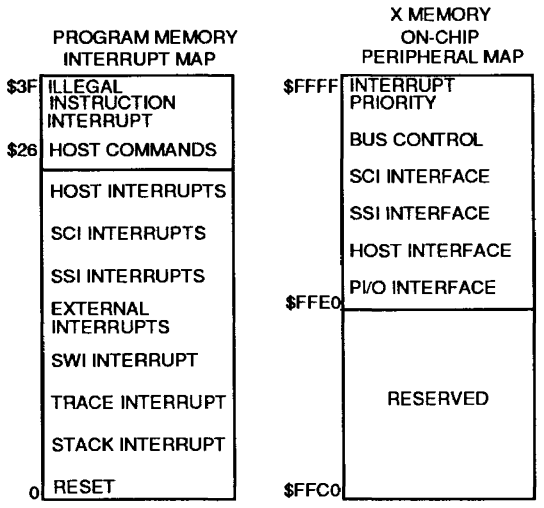


Figure 4.
Interrupt and Peripheral Register Memory Maps

grammed for up to 15 WAIT states (one WAIT state is equal to a clock period or equivalently, one-half of an instruction cycle). In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces.

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 9-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. Also note that data written to a GP I/O pin appears on the pin approximately one or two T states after completion of the execution cycle of that instruction. As a result, if two GP I/O pins are connected and one is written to, an additional instruction time must be allowed for signal propagation before reading the input pin.

HOST INTERFACE

The Host Interface is a byte-wide, full duplex parallel port which may be connected directly to the data bus of a host processor. The host processor may be any of a number of industry standard microcomputers or microprocessors, another DSP, or DMA hardware. The DSP56001 Host Interface has an 8-bit bidirectional data bus H0-H7 (PB0-PB7) and seven dedicated control lines: HA0, HA1, HA2, HR/$\overline{W}$, $\overline{HEN}$, $\overline{HREQ}$, and $\overline{HACK}$ (PB9-PB15) to control data transfers. The Host Interface appears to the host processor as a memory mapped peripheral occupying eight bytes in the host processor address space. Separate transmit and receive data registers are double-buffered to allow the DSP56001 and host processor to transfer data efficiently at high speed. Host processor communication with the Host Interface is accomplished using standard host processor data move instructions and addressing modes. Handshake flags are provided for polled or interrupt driven data transfers with the host processor. DMA hardware may be used with the $\overline{HREQ}$ and $\overline{HACK}$ lines to transfer data without host processor intervention.

One of the most innovative features of the Host Interface is the Host Command feature. With this feature, the host processor can issue vectored exception requests to the DSP56001. The host may select any one of 31 DSP56001 exception routines to be executed by writing a Vector Address Register in the Host Interface. This flexibility allows the host programmer to execute up to 31 functions preprogrammed in the DSP56001. For example, host exceptions can allow the host processor to read or write DSP56001 registers, X, Y, or program memory locations, force exception handlers for SSI, SCI, $\overline{IRQA}$, and $\overline{IRQB}$ exception routines, and perform control and debugging operations to aid program development, if the appropriate exception routines are implemented in the DSP56001.

SERIAL COMMUNICATION INTERFACE (SCI)

The Serial Communications Interface (SCI) provides a full-duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems. The communication can be either direct or via RS232C-type lines. This interface uses three dedicated pins — transmit data (TXD), receive data (RXD), and SCI serial clock (SCLK). It supports industry standard asynchronous bit rates and protocols as well as high speed (up to 3.375 Mbits/sec) synchronous data transmission. The asynchronous protocols include a multidrop mode for master/slave operation. The Serial Communication Interface consists of separate transmit and receive sections having operations which can be asynchronous with respect to each other. A programmable baud rate generator is included to generate the transmit and/or receive clocks. An enable bit and interrupt vector have been included so that the baud rate generator can function as a general purpose timer when it is not being used by the SCI peripheral.

SYNCHRONOUS SERIAL INTERFACE (SSI)

The Synchronous Serial Interface (SSI) is an extremely flexible, full-duplex serial interface which allows the DSP56001 to communicate with a variety of serial devices. These include one or more industry standard codecs, other DSPs, microprocessors, and peripherals. Each of the following characteristics of the SSI can be independently defined: the number of bits per word, 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 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 DSP56001s and/or codecs. The On-Demand mode is a data driven mode. There are no time slots defined. This mode is intended to be used to interface to devices on a non-periodic basis. The clock can be programmed to be continuous or gated. Since the transmitter and receiver sections of the SSI are independent, they may be programmed to be synchronous (use a common clock) or asynchronous (do not use a common clock) 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. A matrix of SSI operating modes and typical applications is provided in Table 1.

PROGRAMMING MODEL DESCRIPTION

The programmer can view the DSP56001 architecture as three execution units operating in parallel. The three execution units are the data ALU, address generation unit, and program controller. The programming model appears like that of a conventional MPU. The programming model is shown in Figure 5 and is described in the following paragraphs.

Table 1. SSI Operating Modes

Mode (Protocol)	Serial Clock	Relative Tx-Rx Timing	Typical Applications
Normal	Continuous	Asynchronous/Synchronous	Asynchronous/Synchronous Codec
Normal	Gated	Asynchronous	Periodic DSP-to-DSP
Normal	Gated	Synchronous	Periodic DSP-to-A/D and D/A
On-Demand	Continuous	Asynchronous	DSP-to MCU
On-Demand	Continuous	Synchronous	P-to-S and S-to-P Conversion
On-Demand	Gated	Asynchronous	DSP-to-DSP
On-Demand	Gated	Synchronous	DSP-to-SPI Peripherals
Network	Continuous	Asynchronous/Synchronous	TDM Codecs/DSP Networks

DATA ALU

The data ALU features 24-bit input/output data registers which can be concatenated to handle 48-bit data, two 56-bit accumulators, automatic scaling, and saturation arithmetic.

DATA ALU INPUT REGISTERS (X1, X0, Y1, Y0)

The data ALU input registers are four 24-bit general purpose data registers which may be treated as four independent 24-bit registers or as two 48-bit registers, called X and Y, developed by the concatenation of X1:X0 and Y1:Y0, respectively. The register with the highest number is the most-significant word. These registers serve as input pipeline registers between the XDB and YDB and the multiply-accumulator unit (MAC). They are used as data ALU source operands as well and allow new operands to be loaded for the next instruction while the register contents are used by the current instruction. They may also be read back to the appropriate data bus to implement memory delay operations and save/restore operations for interrupt service routines.

DATA ALU ACCUMULATOR REGISTERS (A2, A1, A0, B2, B1, B0)

The six data ALU accumulator registers A2, A1, A0, B2, B1, and B0 form two general purpose 56-bit accumulators, A and B, developed by the concatenation, A2:A1:A0 and B2:B1:B0, respectively. These registers are used for arithmetic calculations and data manipulation. The four registers (A1, A0, B1, and B0) are 24 bits wide, and the two registers (A2 and B2) are 8 bits wide. All of these registers can be accessed as word operands. The register with the highest number is the most-significant word in the full 56-bit accumulator; the register with the lowest number is the least-significant word.

These accumulators can be viewed as being 48 bits long with 8-bit extensions to accommodate word growth in vector arithmetic. The registers A2 and B2 are called accumulator extension registers. Automatic sign extension is provided when writing to the full 56-bit accumulators A or B with a 48- or 24-bit operand. The low-order portion will be automatically zeroed when a 24-bit operand is written to form a valid 56-bit operand. The registers may also be written without sign extension or zero fill by specifying the individual register name.

When accumulator registers A or B are saved, they may be optionally scaled one bit left or one bit right for block floating-point arithmetic. Reading the full A or B accumulators over the XDB and YDB is protected against overflow by substituting a limiting constant for the transferred data. The content of A or B is not affected should limiting occur; only the value transferred over the XDB and YDB is limited. This overflow protection is performed after the content of the accumulator has been optionally shifted according to the scaling mode. Note that only when the full accumulator, A or B as opposed to A0, A1, A2, B0, B1, or B2, is specified as the source for a data move over the XDB and YDB will shifting and limiting be performed. The accumulator registers can also serve as pipeline registers between the MAC unit and the XDB and YDB. They are used as both data ALU source and destination operands.

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.

Pointer Register Files (R0-R3 and R4-R7)

The eight pointer registers, R0-R7, are 16 bits wide and may contain addresses or general purpose data. The 16-bit address in a selected pointer register is used in the calculation of the effective address of an operand. When supporting parallel X and Y data memory moves, the pointer registers must be viewed as two separate files, R0-R3 and R4-R7, one file for each bus. The content of an Rn may point to data directly and/or may be pre- or post-updated according to the addressing mode selected. Modifier registers, Mn are always used if an Rn is updated. Offset registers, Nn, are used for the update by offset addressing modes. The pointer register modification is performed by one of the two modulo arithmetic units.

Offset Register Files (N0-N3 and N4-N7)

The eight offset registers, N0-N7, are 16 bits wide and may contain offset values used to increment and decrement address registers in indexed address register update calculations, or they may be used for 16-bit general purpose storage. For example, the contents of an offset register may be used to step through a table at some rate (e.g., five locations per step for waveform generation) or may specify the offset into a table or the base of the table for indexed addressing. Each address register, Rn, has its own offset register, Nn, associated with it.

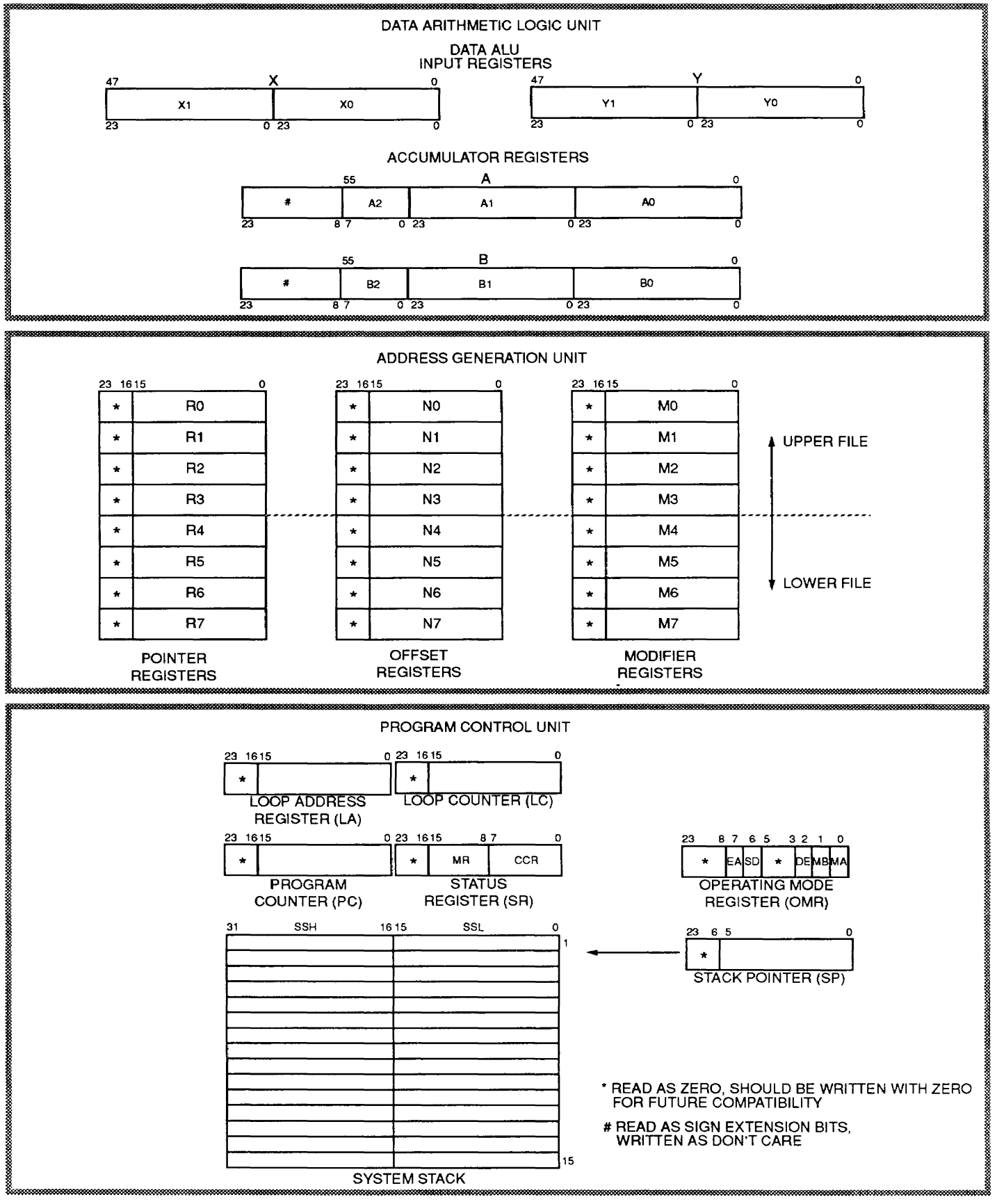


Figure 5. DSP56001 Programming Model

Modifier Register Files (M0-M3 and M4-M7)

The eight modifier registers, M0-M7, are 16 bits wide. The content of Mn defines the type of address arithmetic to be performed for addressing mode calculations. The address generation units support linear, modulo, and reverse carry arithmetic types for all address register indirect addressing modes. For the case of modulo arithmetic, the content of Mn also specifies the modulus. Each address register, Rn, has its own modifier register Mn, associated with it. Each modifier register is set to $FFFF on processor reset which specifies linear arithmetic as the default type for address register update calculations.

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, complete status register, and system stack. With the exception of the program counter, all registers are read/write to facilitate system debug.

Program Counter (PC)

This 16-bit register contains the address of the next location to be fetched from program memory space. This special purpose address register is stacked when program looping is initiated, or when a jump to subroutine (JSR) is performed.

Status Register (SR)

The status register is a 16-bit register consisting of an 8-bit mode register (MR) and an 8-bit condition code register (CCR). SR is stacked when program looping is initialized, or when a jump to subroutine (JSR) is performed. The status register format is shown in Figure 6.

MR is a special purpose control register which defines the current system state of the processor. The MR bits are affected by processor reset, exception processing, the DO, ENDDO, RTI, and SWI instructions, and by instructions which directly reference the MR register, namely, ORI and ANDI.

CCR is a special purpose control register which defines the current user state of the processor at any given time. The CCR bits are affected by data ALU operations and by instructions which directly reference the CCR register, namely, ORI and ANDI. The CCR bits are not affected by parallel move operations unless data limiting occurs when reading the A or B accumulators.

Loop Counter (LC)

The loop counter is a special 16-bit counter used to specify the number of times a hardware program loop is to be repeated. This register is stacked by a DO instruction and unstacked by end of loop processing or by execution of an ENDDO instruction. The loop counter may be read under program control allowing the number of times a loop has been executed to be monitored during execution.

Loop Address Register (LA)

The content of the loop address register indicates the location of the last instruction word in a program loop. This register is stacked by a DO instruction and unstacked by end of loop processing or by execution of an ENDDO instruction.

System Stack (SS)

The system stack is a separate internal memory which stores the PC and SR for subroutine calls and long interrupts. The stack will also store the LC and LA registers in addition to the PC and SR registers for program looping. The SS is in stack memory space, and its address is always inherent and implied by the current instruction. The system stack memory is 32 bits wide and 15 locations deep.

When a subroutine call or long interrupt occurs, the contents of the PC and SR are stored (pushed) on the top location in the system stack. When a return from subroutine occurs, the contents of the top location in the system stack are transferred (pulled) to the PC only. When a return from interrupt occurs, the contents of the top location in the system stack are transferred (pulled) to both the PC and SR.

The interrupt subsystem of the DSP56001 is vector based and prioritized. Interrupt vectors point to two consecutive locations in program memory. If one of the two words fetched by the interrupt controller is a jump to subroutine instruction, a long interrupt routine is formed and a context switch is performed using the stack. If neither interrupt instruction word causes a change of control flow, then the two interrupt instruction words fetched constitute a fast interrupt routine. The fast interrupt routine provides exception processing with no overhead. This mechanism is commonly used to move data between memory and an I/O device.

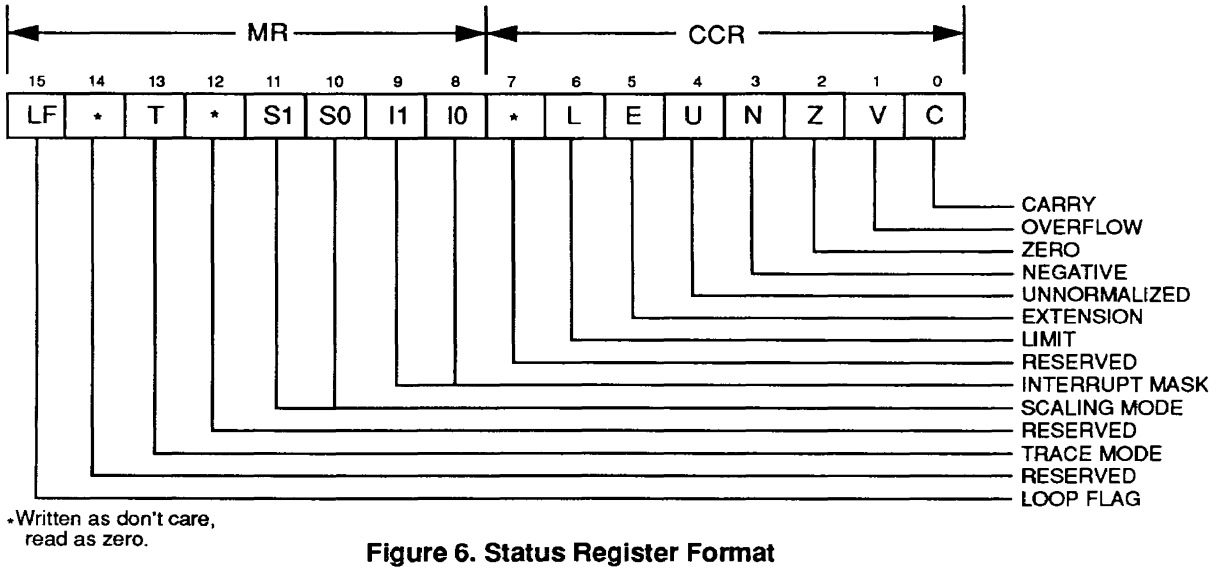


Figure 6. Status Register Format

The system stack is also used to implement no-overhead nested hardware DO loops. Each stack location can be addressed as two separate 16-bit registers — system stack high (SSH), and system stack low (SSL). This facilitates creating software stacks for unlimited nesting.

Stack Pointer (SP)

The stack pointer register (SP) is a 6-bit register that indicates the location of the top of the system stack and the status of the stack (underflow, empty, full, and overflow conditions). The stack pointer is referenced implicitly by some instructions (DO, REP, JSR, RTI, etc.) or directly by the MOVEC instruction. The stack pointer register format is shown in Figure 7.

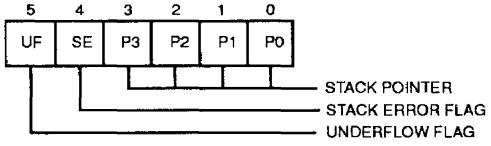


Figure 7. Stack Pointer Format

Operating Mode Register (OMR)

The operating mode register is a 5-bit register which defines the current operation mode of the processor, i.e., the memory maps for program and data memories as well as the start-up procedure, recovery delay time when exiting the stop mode, and bus control pin operation. The OMR bits are only affected by processor reset and by instructions which directly reference the OMR. During processor reset, the chip operating mode bits, MB and MA, will be loaded from the external mode select, A and B, pins. The Data ROM Enable bit (DE) will be cleared, disabling the X and Y on-chip lookup table ROMs, the stop delay bit (SD) and the external access bit (EA) will also be cleared. The operating mode register format is shown in Figure 8. Table 2 summarizes the DSP56001 operating modes. Tables 3 and 4 show the program and data memory spaces.

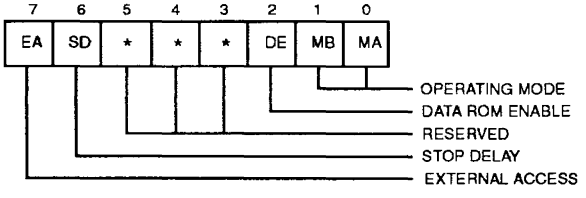


Figure 8. Operating Mode Register Format

Table 2. Operating Mode Summary

Operating Mode	MB	MA	Description
0	0	0	PRAM enabled, Reset at $0000 (internal).
1	0	1	Special Bootstrap mode, after PRAM loading mode 2 is automatically selected.
2	1	0	PRAM enabled, Reset at $E000 (external).
3	1	1	PRAM disabled, Reset at $0000 (external).

Table 3. Program Memory Space

Operating Mode	MB	MA	Description
0 and 2	X	0	Internal RAM: $0000–$01FF External: $0200–$FFFF
3	1	1	External: $0000–$FFFF

Table 4. Data Memory Space

DROM Enable DE	Y Data Memory Space Map	X Data Memory Space Map
0	Internal RAM: $0000–$00FF External: $0100–$FFFF	Internal RAM: $0000–$00FF External: $0100–$FFBF On-chip peripherals: $FFC0–$FFFF
1	Internal RAM: $0000–$00FF Internal ROM: $0100–$01FF External: $0200–$FFFF	Internal RAM: $0000–$00FF Internal ROM: $0100–$01FF External: $0200–$FFBF On-chip peripherals: $FFC0–$FFFF

Table 5. Stop Delay Control

OMR Bit 6	Delay When Exiting Stop Mode
0	65,545•cyc where cyc = 2T
1	17•cyc

Table 6. External Access Control

OMR Bit 7	BR Pin (Input)	BG Pin (Output)
0	Bus Request (BR)	Bus Grant (BG)
1	Wait (WT)	Bus Strobe (BS)

INSTRUCTION SET SUMMARY

The DSP56001 instruction set has been designed to be as orthogonal as possible to allow flexible, independent, concurrent control of the data ALU, address generation unit, and program control execution units during each instruction cycle. This maximizes throughput and minimizes program storage requirements. The rich instruction set features DSP oriented instructions such as CMPM, NORM, RND, MACR, SUBL, SUBR, ADDL, ADDR, DO, and REP which are summarized below. The instruction set is divided into the following groups:

Arithmetic
Logical
Bit Manipulation
Loop
Move
Program Control

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 so that the data ALU operation indicated by the instruction does not use the X data bus, the Y data bus, or the global data bus. This method allows for parallel data movement over the X and Y data buses or over the global data bus during a data ALU operation. The availability of these buses permits 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 destination is either of the 56-bit accumulators. The following are the arithmetic instructions.

ABS	Absolute Value
ADC	Add Long with Carry
ADD	Add
ADDL	Shift Left and Add Accumulators
ADDR	Shift Right and Add Accumulators
ASL	Arithmetic Shift Accumulator Left
ASR	Arithmetic Shift Accumulator Right
CLR	Clear Accumulator
CMP	Compare
CMPM	Compare Magnitude
DIV	Divide Iteration*
MAC	Signed Multiply-Accumulate
MACR	Signed Multiply-Accumulate and Round
MPY	Signed Multiply
MPYR	Signed Multiply and Round
NEG	Negate Accumulator
NORM	Normalize Accumulator Iteration*
RND	Round Accumulator
SBC	Subtract Long with Carry
SUB	Subtract
SUBL	Shift left and Subtract Accumulators
SUBR	Shift Right and Subtract Accumulators
Tcc	Transfer Conditionally*
TFR	Transfer Data ALU Register
TST	Test Accumulator

*These instructions do not allow parallel data moves.

The CMPM instruction affects the condition code bits according to the results of the subtraction of the absolute values of two operands. This instruction, together with Tcc, is useful in determining maximum and minimum values in blocks of data.

The NORM instruction performs a 1 bit normalization on the content of an accumulator register and updates the content of the specified address register according to the normalization. This instruction is particularly useful in implementing floating-point routines.

The RND instruction performs convergent rounding on the content of an accumulator register in a manner consistent with the scaling mode operation.

The MACR instruction is one of the most powerful instructions in the instruction set. It performs a signed multiply-accumulate with convergent rounding and allows two parallel data moves in one instruction. These rounding instructions minimize the effects of roundoff errors.

The ADDL, ADDR, SUBL, and SUBR instructions multiply or divide the content of the accumulator register by two before the addition or subtraction operation is performed. They are particularly useful for implementing Radix-2 Decimation In Time (DIT) Fast Fourier Transforms (FFT) and interpolating between two data values.

LOGICAL INSTRUCTIONS

The logical instructions perform all of the logical operations within the data ALU. They affect all of the condition code register bits. Logical instructions are register-based. Optional data transfers may be specified with most logical instructions. This allows for parallel data movement over XDB, YDB, or GDB during a data ALU logical operation. This allows new data to be prefetched for use in following instructions and results, calculated in previous instructions, to be stored. These instructions execute in one instruction cycle. The destination is either A1 or B1, except for ANDI or ORI. The following are the logical instructions.

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

*These instructions do not allow parallel data moves.

BIT MANIPULATION INSTRUCTIONS

There are two basic groups of bit manipulation instructions. One group tests the state of any single bit in a memory location and then optionally sets, clears, or inverts the bit. The other group tests the state of any single bit in a memory location and jumps (or jumps to subroutine) if the bit is set or clear. The carry bit of the condition code register will contain the result of the bit test for the first group. The following are the bit manipulation instructions. Parallel data moves are not allowed with any of these instructions.

BCLR	Bit Test and Clear
BSET	Bit Test and Set
BCHG	Bit Test and Change
BTST	Bit Test on Memory
JCLR	Jump if Bit Clear
JSET	Jump if Bit Set
JSCLR	Jump to Subroutine if Bit Clear
JSSET	Jump to Subroutine if Bit Set

LOOP INSTRUCTIONS

The DO and ENDDO instructions make writing straight line code practically unnecessary. The DO instruction sets up a hardware loop by initiating a program loop, setting up looping parameters and then restores the system stack when terminating a loop. Initialization includes saving registers LA and LC, used by a program loop, on the system stack so that program loops can be nested. The address of the first instruction in a program loop is also saved on the stack to allow no-overhead looping. Single instruction DO loops can be implemented. DO loops are interruptible. An indirect address can be used to specify the "loop count" in the DO instruction. This facilitates parameter passing. The ENDDO instruction is used to terminate a DO loop prematurely. It is used to restore the stack. These instructions do not allow parallel data moves. The following are the loop instruction definitions.

DO	Start Hardware Loop
ENDDO	End Current DO Loop

MOVE INSTRUCTIONS

The move instructions perform data movement over XDB, YDB, and GDB as well as the PDB. Move instructions do not affect the condition code register except the limit bit, L, if limiting is performed when reading data ALU accumulator registers A or B. The MOVE instruction provides all of the parallel data move operations and can be considered

to be a data ALU no-op with parallel moves. The following are the move instructions.

LUA	Load Updated Address
MOVE	Move Data Registers
MOVEC	Move Control Register
MOVEM	Move Program Memory
MOVEP	Move Peripheral Data

PROGRAM CONTROL INSTRUCTIONS

The program control instructions include jumps, conditional 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. Optional parallel data transfers over XDB, YDB, and GDB are not allowed in the program control instructions. The REP instruction repeats the next instruction without refetching the instruction to maximize throughput. Because the instruction repeated is not refetched, a REP operation is not interruptible. An interruptible repeat instruction can be implemented using a single instruction DO loop. After a STOP instruction is executed, all processor activity is suspended and the oscillator is gated off. When the WAIT instruction is executed, internal processing is halted and the processor waits for an interrupt. The STOP and WAIT states are low power states. The following are the program control instructions

Illegal	Force an Illegal Instruction Interrupt
Jcc	Jump Conditionally
JMP	Jump
JScc	Jump to Subroutine Conditionally
JSR	Jump to Subroutine
NOP	No Operation
REP	Repeat Next Instruction
RESET	Reset On-Chip Peripheral Devices
RTI	Return from Interrupt
RTS	Return from Subroutine
STOP	Stop Instruction Processing†
SWI	Software Interrupt
WAIT	Wait for Interrupt†

†Low power standby modes

INSTRUCTION FORMATS

Instructions are one or two words in length. The instruction and its length is specified by the first word of the instruction. The second word may contain an absolute address or immediate data. The assembly language source code for a typical one word instruction is shown below. The source code is organized into four fields.

Opcode	Operands	X Bus Data	Y Bus Data
Mac	X0,Y0,A	X:(R0)+ ,X0	Y:(R4)+ ,Y0

The opcode field typically indicates the data ALU operation to be performed; it may also specify a move, address generation, or program control operation. The field specifies the operands to be used by the opcode. The X Bus Data field specifies an optional data transfer over the XDB and the addressing mode to be used. The Y Bus Data field specifies an optional data transfer over the YDB and the addressing mode to be used. The memory space qualifiers X:, Y:, P:, and L: (long memory space) indicate which memory space is being referenced. The Opcode field must always be included in the source code. The optional X:, Y:, fields can be interchanged.

The DSP56001 allows parallel processing by the data ALU, address generation unit, and program controller. For example, in the instruction word above:

- the data ALU performs the designated ALU operation,
- the address generation unit performs data transfers specified with address register updates, and

All of these operations are accomplished in one instruction cycle. In addition, the program control unit may be processing an active hardware DO loop. When an instruction is more than one word in length, an additional instruction execution cycle may be required. Operations involving the data ALU are register-based (i.e., all operands are in data ALU registers) and, therefore, do not utilize the data buses. This allows the programmer to keep each execution unit busy by specifying memory accesses in parallel over the XDB, YDB, or GDB. An instruction which is memory-oriented (such as a bit manipulation instruction) or an instruction that causes a control flow change (such as a jump) does not allow parallel data moves during its execution.

ADDRESSING MODES

The addressing modes are grouped into three categories — register direct, address register indirect, and special. These addressing modes are summarized in Table 7. All address calculations are performed in the address generation unit to minimize execution time and loop overhead. 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. 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, R3;N3;M3, R4;N4;M4, R5;N5;M5, R6;N6;M6, and R7;N7;M7. This structure is unique and extremely powerful in general, and particularly powerful in setting up DSP oriented data structures. All address register indirect modes use at least one set of address registers and the XY memory reference uses two sets of address registers: one set for X memory space and one set for Y memory space.

The special addressing modes include immediate and absolute modes as well as implied references to the PC, system stack, and program memory.

ADDRESS ARITHMETIC MODIFIERS (Mn)

The address arithmetic modifiers allow the DSP address generation unit to support linear, reverse-carry, and modulo address arithmetic for all address register indirect modes. These special address arithmetic types 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 pointer registers rather than moving large blocks of data.

The content of the address arithmetic modifier register, Mn, defines the type of address arithmetic to be performed for addressing mode calculations. For the case of modulo arithmetic, the content of Mn also specifies the modulus. The three types of arithmetic are discussed below.

Linear Arithmetic (Mn = $FFFF)

The address modification is performed using normal 16-bit (modulo 65,536) linear arithmetic (two's complement). A 16-bit offset, Nn, 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 +65,535).

Reverse-Carry Arithmetic (Mn = $0000)

The address modification is performed by propagating the carry in the reverse direction, that is, from the most significant bit (MSB) to the least-significant bit (LSB). This is equivalent to bit-reversing (i.e., redefining the MSB as the LSB and the next MSB as bit 1, etc.) the content of Rn and the offset value Nn, adding normally, and then bit-reversing the result. If the (Rn)+Nn addressing mode is used with this address modifier type, and Nn contains the value 2^{k-1}, then postincrementing by +Nn is equivalent to incrementing Rn by 1 and bit-reversing the k LSBs of Rn. This address arithmetic is useful for performing 2k point Fast Fourier Transforms (FFTs). The range of values for Nn is 0 to +65,535. This allows bit-reversed addressing for FFTs having up to 65,536 points.

As an example, consider a 1,024 point complex FFT (k= 10) with real data stored in X memory and imaginary data stored in Y memory. Then Nn would contain the value 512 and postincrementing by +Nn would generate the address sequence 0, 512, 256, 768, 128, 640, ... This is the scrambled FFT data order for sequential frequency points from 0 to 2 pi. The base address must have at least k zeros so that the reverse-carry modifier works when the base address of the FFT data buffer is a multiple of 2^k, such as 2048, 3072, in the example. The use of addressing modes other than postincrement by +Nn is possible but may not provide a useful result.

Modulo Arithmetic (Mn = Modulus - 1)

The address modification is performed modulo M, where M ranges from 2 to +32,768. Modulo M arithmetic causes the address register value to remain within an address range of size M defined by a lower and upper address boundary. The value m=M-1 is stored in the modifier register Mn. The lower boundary (base address) value must have zeroes in the k LSBs, where $2^k \geq M$, and therefore must be a multiple of 2^k. The upper boundary is the lower boundary plus the modulo size minus one (base address plus M-1). Since $M \leq 2^k$, once M is chosen, a sequential series of memory blocks each of length 2^k is created where these circular buffers can be located. If $M \leq 2^k$, there will be a space between sequential circular buffers of 2^k-M. For example, to create a circular buffer of 21 stages, M is 21 and the lower address boundary must have its five least-significant bits equal to zero ($2^k \geq 21$, thus $k \geq 5$). The Mn register is loaded with the value 20. The lower boundary may be chosen as 0, 32, 64, 96, 128, 160, etc. The upper boundary of the buffer is then the lower boundary plus 21. The address pointer is not required to start at the lower address boundary or end on the upper address boundary; it may initially point anywhere within the defined modulo address range. Note that neither the lower nor the upper boundary of the modulo region is stored; only the size of the modulo region is stored in Mn. Assuming the (Rn)+ indirect addressing mode, if the address register pointer increments past the upper boundary of the buffer (base address plus M-1), it will wrap around through the base address (lower boundary). Alternatively, assuming the (Rn) indirect addressing

Table 7. Address Modes Summary

Addressing Mode	Modifier MMMM	Memory/Registers Referenced								
		S	C	D	A	P	X	Y	L	XY
Register Direct										
Data or Control Register	No	x	x	x						
Address Register	No				x					
Address Modifier Register	No				x					
Address Offset Register	No	-			x					
Address Register Indirect										
No Update	Yes					x	x	x	x	x
Postincrement by 1	Yes					x	x	x	x	x
Postdecrement by 1	Yes					x	x	x	x	x
Postincrement by Offset Nn	Yes					x	x	x	x	x
Postdecrement by Offset Nn	Yes					x	x	x	x	
Indexed by Offset Nn	Yes					x	x	x	x	
Predecrement by 1	Yes					x	x	x	x	
Special										
Immediate Data	No					x				
Absolute Address	No					x	x	x	x	
Immediate Short Data	No					x				
Short Jump Address	No					x				
Absolute Short Address	No					x	x	x	x	
I/O Short Address	No						x	x		
Implicit	No	x	x			x				

Where

MMMM = Address Modifier
S = Stack Reference
C = Program Control Unit Register Reference
D = Data ALU Register Reference
A = Address ALURegisterReference
P = Program MemoryReference
X = X Memory Reference
Y = Y Memory Reference
L = L Memory Reference
XY = XY Memory Reference

mode, if the address decrements past the lower boundary (base address), it will wrap around through the base address plus M-1 (upper boundary).

If an offset, Nn, is used in the address calculations, the 16-bit value |Nn| must be less than or equal to M for proper modulo addressing. If Nn>M the result is data dependent and unpredictable except for the special case where $Nn = L \cdot 2^K$, a multiple of the block size where L is a positive integer. For this case, when using the (Rn)+Nn addressing mode, the pointer Rn will jump linearly to the same relative address in a new buffer L blocks forward in memory. Similarly, for (Rn)-Nn the pointer will jump back L blocks in memory. The range of values for Nn is –32,768 to +32,767. The modulo arithmetic unit will automatically wrap the address pointer around by the required amount. This type of address modification is useful in creating circular buffers for FIFOs (queues), delay lines, and sample buffers up to 32,768 words long. It is also useful for decimation, interpolation, and waveform generation. The special case of (Rn) $\pm$ Nn mod M with $Nn=L \cdot 2^K$ is useful for performing the same algorithm on multiple blocks of data in memory, for example, parallel IIR filtering.

WAIT PROCESSING STATE

The wait processing state is a low power-consumption state entered by execution of the WAIT instruction. In the wait state, the internal clock is disabled from all internal circuitry except the internal peripherals (e.g., the interrupt controller, the SCI, SSI, and HI). All internal processing is halted until an unmasked interrupt occurs or until the DSP is reset. The $\overline{BR}/\overline{BG}$ circuits remain active during the wait state.

STOP PROCESSING STATE

The stop processing state, which is the lowest power-consumption state, is entered by the execution of the STOP instruction. In the stop state, the clock oscillator is gated off; whereas, in the wait mode, the clock oscillator remains active. The chip clears all peripheral interrupts (HI, SSI, and SCI) and external interrupts (IRQA, IRQB, and NMI) when entering the stop state. Trace, SWI, or stack errors that were pending, remain pending. The priority levels of the peripherals remain as they were before the STOP instruction was executed. The SCI, SSI, and HI are held in their respective individual reset states while in the stop state.

All activity in the processor is halted until (1) a low level is applied to the $\overline{IRQA}$ pin or (2) a low level is applied to the $\overline{RESET}$ pin. Either of these actions will gate on the oscillator, and, after a clock stabilization delay, clocks to the processor and peripherals will be re-enabled. The clock stabilization delay period is determined by the stop delay (SD) bit in the OMR.

APPLICATION DEVELOPMENT TOOLS

SOFTWARE

All software support products run on the following platforms — IBM™ PC, Macintosh™ II, SUN-3™ workstation. The software, written in C, consists of an assembler, linker, and simulator which are marketed as an integrated product. The ordering information is as follows:

Host Platform	Operating System	Order Number
IBM-PC	DOS 2.x, 3.x	DSP56000CLASA
Macintosh II	MAC OS 6.x	DSP56000CLASB
SUN-3	SunOS™ 3.5	DSP56000CLASC

IBM™ is a trademark of International Business Machines.
Macintosh™ is a trademark of Apple Computer, Inc.
SUN-3™ is a trademark of Sun Microsystems, Inc.
SunOS™ is a trademark of Sun Microsystems, Inc.

Macro Cross Assembler

The Macro Cross Assembler program is a full-featured macro cross assembler that translates one or more source fields containing DSP instruction mnemonics, operands, and assembler directives into relocatable object modules that are relocated and linked by the DSP56000 Linker in the Relocation mode. In the Absolute mode, the assembler will generate absolute load files. The assembler recognizes the full instruction set and all addressing modes of the DSP56000. This includes support for separate X and Y data memory spaces and data transfer operations in parallel with the data ALU operations.

This assembler offers the usual complement of features found in modern assemblers, such as conditional assembly, file inclusion, nested macros with support for macro libraries (via the MACLIB directive), and modular programming constructs ordinarily found only in higher level languages.

The unique architecture and parallel operation of the DSP demands special purpose facilities and programming aids which this assembler readily provides. These include built-in functions for common transcendental math computations such as sine, cosine, log, and square root functions; arbitrary expressions and modulo operations; and directives to define circular and bit-reversed data buffers. Moreover, the assembler incorporates extensive error checking and reporting to indicate programming violations peculiar to the digital signal processing environment or stemming from the advanced features of the DSP. These include errors for improper nesting of hardware DO loops and improper address boundaries for circular data buffers and bit-reversed buffers.

The assembler also generates source code listings which include numbered source lines, optional titles and subtitles, optional instruction cycle counts, symbol table and cross-reference listings, and memory use reports.

To summarize, features of the assembler are:

- Produces relocatable object modules compatible with the DSP linker program in the Relocation mode
- Produces absolute load files compatible with the Simulator program (SIM56000) in the Absolute mode
- Supports full instruction set, memory spaces, and parallel data transfer fields of the DSP
- Modular programming features including local labels, sections, and external definition/reference directives
- Nested macro libraries
- Complex expression evaluation including boolean operators
- Built-in functions for data conversion, string comparison, and common transcendental math operations
- Directives to define circular and bit-reversed buffers
- Extensive error checking and reporting

Linker/Librarian

The linker relocates and links relocatable object modules from the Macro Cross Assembler to create an absolute load file which can be loaded directly into the DSP56000/DSP56001- simulator or converted to Motorola S-record format for PROM burning.

The librarian utility will merge into a single file multiple separate relocatable object modules. This facilitates not having to reassemble known bug-free routines every time the mainline program is assembled.

Simulator Program

The Simulator program is a software tool for developing programs and algorithms for the DSP56000 family of DSPs. This program exactly emulates all of the functions of the DSP including all on-chip peripheral operations, the entire internal and external memory space, all memory and register updates associated with program code execution, and all exception processing activity. This enables the Simulator program to provide you with an accurate measurement of code execution time which is so critical in digital signal processing applications.

The Simulator program executes DSP object code which has been generated using the Linker or the Simulator's internal single-line assembler. The object code is loaded into the simulated DSP memory map. Instruction execution can proceed until a user-defined breakpoint is encountered; or in single-step mode, stopping after each instruction has been executed. During program debug, the registers or memory locations may be displayed or changed.

The Simulator package includes linkable object code libraries of simulator functions that were used to create the simulator. The libraries allow a customized simulator to be built and integrated with unique system simulations. Source code for some of the functions, such as the terminal I/O functions and external memory accesses, is provided to allow close simulation of the particular application.

To summarize, features of the Simulator program are:

- Simulates the DSP56001 (default) or DSP56000
- Simulation of multiple DSP devices
- Linkable object code libraries
  - Nondisplay simulator library
  - Display simulator library
- C language source code for
  - Screen management functions
  - External memory reference functions
  - Terminal I/O functions
  - Simulation examples
- Single stepping through object programs
- Conditional or unconditional breakpoints
- Program patching using a resident single-line assembler/disassembler
- Instruction and cycle timing counters
- Session and/or command logging for later reference
- ASCII input/output files for peripherals
- Help file and help line display of simulator commands
- Loading and saving of files to/from simulator memory
- Macro command definition and execution
- Display enable/disable of registers and memory
- Hexadecimal/decimal/binary calculator

HARDWARE

DSP56000 Application Development System (ADS)

The DSP56001-based Application Development System (ADS) is a three component development tool for designing, debugging, and evaluating DSP56000 and DSP56001 target system equipment. The ADS is fully compatible with the DSP56000CLASx design-in software package and may act as an accelerator for testing simulated DSP56000 family algorithms. The ADS can be used with any of the following computers — IBM-PC, Macintosh II, VAX, or SUN-3. There are different user interface programs and interface boards that run on each of these machines; however, the Application Development Module (ADM) board is the same for all four machines. In Figure 9, an IBM PC using the MS-DOS operating system acts as the host platform to interface with the DSP56000 hardware. The three ADS components are an ADM board, an IBM PC bus interface board, and an MS-DOS based user interface program that runs on the IBM PC and interacts with the user.

The ADM hardware is shown in Figure 10. Hardware features include:

- Full speed operation at 20.48 or 27 MHz
- Multiple ADM support with programmable ADM addressing
- 8192 words of configurable RAM for DSP56001 code development
- 2048 words of monitor EPROM expandable to 4096 words
- 96-pin Eurocard connector for accessing all DSP56001 pins
- Separate connectors for accessing serial or host/DMA ports
- Stand-alone operation of ADM after initial development
- No external supply required when connected to IBM PC

The additional peripheral port connectors are particularly useful when developing multi-DSP56001 systems using multiple ADMs. Jumper options allow changing clock inputs; DSP56001 operating mode on reset; reconfiguration of RAM partitioning between Program, X, or Y memory spaces; and address relocation of RAM and/or ROM.

Note that the ADS makes use of the Non-Maskable Interrupt (NMI) function on the DSP56001. A non-maskable interrupt is generated by raising the MODB/$\overline{\text{IRQB}}$ pin to 10 volts. This high voltage presents a potential long-term reliability risk for the DSP56001 and, therefore, the use of the NMI function in running DSP56001 based applications is NOT recommended. The NMI function is intended solely for developing applications.

The features of the DSP56000 ADS user interface program are:

- Single/multiple stepping through DSP56000 object programs
- As many as 99 conditional or unconditional breakpoints
- Program patching using a single-line assembler/disassembler
- Session and/or command logging for later reference
- Loading and saving of files to/from ADM memory
- Macro command definition and execution
- Display enable/disable of registers and memory
- Debug commands that support multiple (up to 8) ADMs
- Hexadecimal/l/decimal/binary calculator
- Host platform system commands accessible from within ADS user interface program
- Multiple host platform operating system input/output file access from DSP56000 object programs
- Fully compatible with the DSP56000CLASx design-in software package

The order number is DSP56000ADSx for the 20.48 MHz ADS or DSP56000ADSx27 for the 27 MHz ADS where the x is A, B, C, or D and indicates the host computer (the same as for the DSP56000CLASx software).

DSP ELECTRONIC BULLETIN BOARD — DR. BUB

Dr. BuB is Motorola Digital Signal Processor Operation's 24-hour electronic bulletin board. This bulletin board offers the DSP community information on Motorola's DSP products, including:

- Current documentation on
  - new products
  - improvements to existing products
- Application notes
  - new
  - updates to existing notes
- Question and answer forum

To logon to the bulletin board, follow these instructions:

1. Dial (512) 891-DSP1 (891-3771) to access 1200 baud Bell 212A modems, (512) 891-DSP2 (891-3772) to access V.22 modems, or (512) 891-DSP3 (891-3773) to access 2400 baud modems. **Be sure to set** the character format to 7 bit data, even parity, and 1 stop bit.
2. Once the connection has been established, respond to the prompt "Dr. BuB login:" with the word "guest" (lowercase required) followed by a carriage return. No password is necessary; if the prompt "password" appears, simply press the carriage return.
3. Finally, identify your terminal type. Terminal type is "dumb" if you have no terminal emulation software. Otherwise select the terminal type from the list of over 100 terminals that are available.

Once you have logged on to Dr. BuB, a series of menus will provide selections guiding you in use of the system.

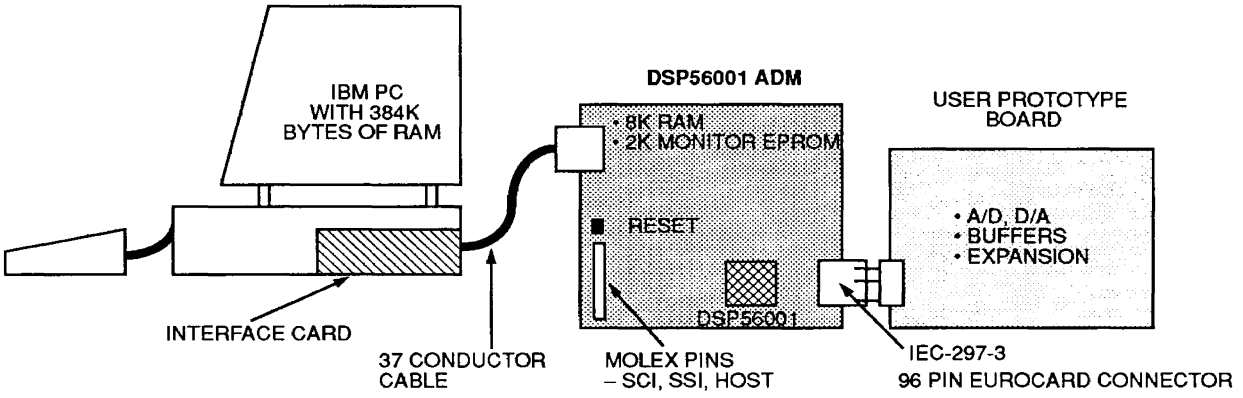


Figure 9. Application Development System Components

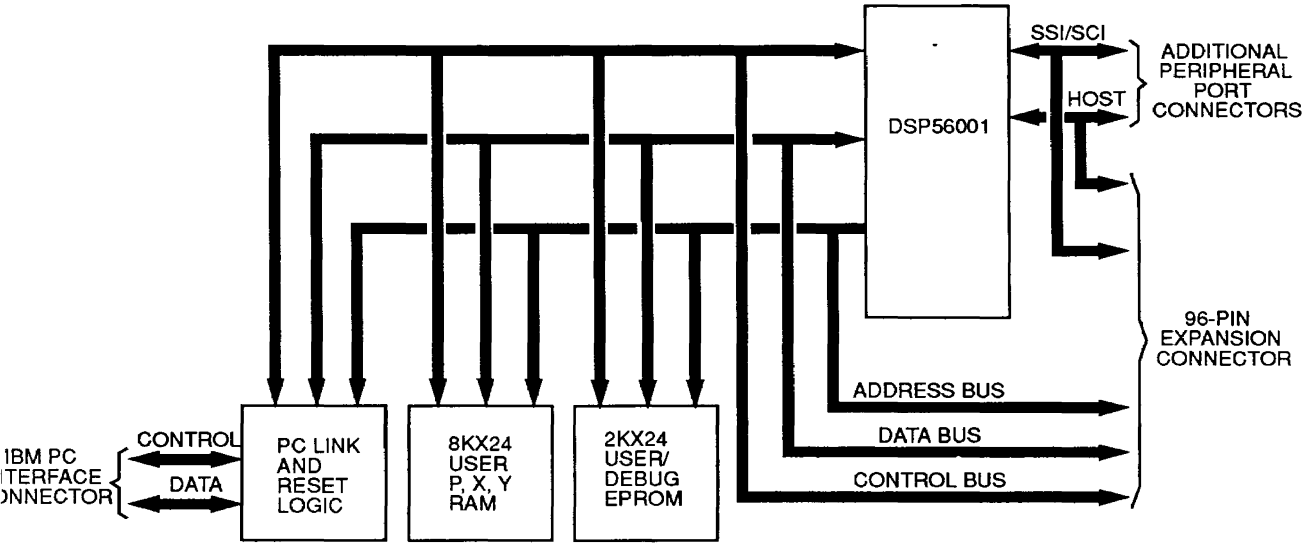


Figure 10. Application Development Module — Block Diagram

DSP56001 Electrical Characteristics
Electrical Specifications

The DSP is fabricated in high density CMOS with TTL compatible inputs and outputs.

Maximum Ratings (V_{SS} = 0 Vdc)

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	V
All Input Voltages	Vin	V_{SS}- 0.5 to V_{CC} + 0.5	V
Current Drain per Pin excluding V_{CC} and V_{SS}	I	10	mA
Operating Temperature Range	T_J	-40 to +105	°C
Storage Temperature	Tstg	-55 to +150	°C

Maximum Electrical Ratings

Thermal Characteristics - PGA Package

Characteristics Thermal Resistance - Ceramic	Symbol	Value	Rating
Junction to Ambient	Θ_{JA}	27	°C/W
Junction to Case (estimated)	Θ_{JC}	6.5	°C/W

Thermal Characteristics - SLAM Package

Characteristics Thermal Resistance - Ceramic	Symbol	Value	Rating
Junction to Ambient	Θ_{JA}	34	°C/W
Junction to Case (estimated)	Θ_{JC}	1.0	°C/W

Thermal Characteristics - CQFP Package

Characteristics Thermal Resistance - Ceramic	Symbol	Value	Rating
Junction to Ambient	Θ_{JA}	40	°C/W
Junction to Case (estimated)	Θ_{JC}	7.0	°C/W

Thermal Characteristics - PQFP Package

Characteristics Thermal Resistance - Ceramic	Symbol	Value	Rating
Junction to Ambient	Θ_{JA}	47	°C/W
Junction to Case (estimated)	Θ_{JC}	13.0	°C/W

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either Gnd or V_{CC}).

Power Considerations

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \times \Theta_{JA}) \quad (1)$$

Where:

T_A = Ambient Temperature, °C

Θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{CC} \times V_{CC}$, Watts - Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins - User Determined

For most applications $P_{I/O} << P_{INT}$ and can be neglected; however, $P_{I/O} + P_{INT}$ **must not** exceed P_d. An appropriate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K/(T_J + 273° C) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \times (T_A + 273° C) + \Theta_{JA} \times P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation (2) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A. The total thermal resistance of a package (Θ_{JA}) can be separated into two components, Θ_{JC} and C_A, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (Θ_{JC}) and from the case to the outside ambient (C_A). These terms are related by the equation:

$$\Theta_{JA} = \Theta_{JC} + C_A \quad (4)$$

Θ_{JC} is device related and cannot be influenced by the user. However, C_A is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling, and thermal convection. Thus, good thermal management on the part of the user can significantly reduce C_A so that Θ_{JA} approximately equals Θ_{JC}. Substitution of Θ_{JC} for Θ_{JA} in equation (1) will result in a lower semiconductor junction temperature. Values for thermal resistance presented in this document, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices", and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User-derived values for thermal resistance may differ.

Layout Practices

Each V_{CC} pin on the DSP56001 should be provided with a low-impedance path to + 5 volts. Each GND pin should likewise be provided with a low-impedance path to ground. The power supply pins drive four distinct groups of logic on chip. They are:

V_{CC}	GND	Function
G12,C6	G11,B7	Internal Logic supply pins
L8	L6,L9	Address bus output buffer supply pins
G3	D3,J3	Data bus output buffer supply pins
C9	E11	Port B and C output buffer supply pins

Power and Ground Connections for PGA

V_{CC}	GND	Function
35, 36, 128, 129	33, 34, 130, 131	Internal Logic supply pins
63, 64	55, 56, 73, 74	Address bus output buffer supply pins
100, 101	90, 91, 111, 112	Data bus output buffer supply pins
12, 13	23, 24	Port B and C output buffer supply pins

Power and Ground Connections for CQFP and PQFP

Power and Ground Connections

The V_{CC} power supply should be bypassed to ground using at least four 0.1 uF by- pass capacitors located either underneath the chip or as close as possible to the four sides of the package. The capacitor leads and associated printed circuit traces connecting to chip V_{CC} and Gnd should be kept to less than 1/2" per capacitor lead. A four-layer board is recommended, employing two inner layers as V_{CC} and Gnd planes. All output pins on the DSP56001 have fast rise and fall times — typically less than 3 ns. with a 10 pf. load. Printed circuit (PC) trace interconnection length should be minimized in order to minimize undershoot and reflections caused by these fast output switching times. This recommendation particularly applies to the address and data buses as well as the $\overline{RD}$, $\overline{WR}$, $\overline{IRQA}$, $\overline{IRQB}$, and $\overline{HEN}$ pins. Maximum PC trace lengths on the order of 6" are recommended. Capacitance calculations should consider all device loads as well as parasitic capacitances due to the PC traces. Attention to proper PCB layout and bypassing becomes especially critical in systems with higher capacitive loads because these loads create higher transient currents in the V_{CC} and GND circuits. Pull up/down all unused inputs or signals that will be inputs during reset.

Signal Stability

When designing hardware to interface with the Host Interface, it is important to ensure that all signals be clean and free from noise. Particular attention should be given to the quality of the Host Enable ($\overline{HEN}$). All inputs to the port should be stable when $\overline{HEN}$ is asserted and should remain stable until $\overline{HEN}$ has fully returned to the deasserted state. It is important to note that such phenomena as ground-bounce and cross-talk can inadvertently cause $\overline{HEN}$ to temporarily rise above $V_{il\ max}$. Should this occur without completing the full logic transition to $V_{ih\ min}$, the DSP56001 Host Port may not correctly update the port status information which can result in storing two or more copies of a single down loaded data word. Of course, if a full logic transition occurs, the part will complete a normal data transfer operation.

DSP56001 Electrical Characteristics

DC Electrical Characteristics (V_{CC} = 5.0 Vdc ± 10%; T_J = -40 to +105° C)

Characteristics	Symbol	Min	Typ	Max	Unit
Supply Voltage	V_{CC}	4.5	5.0	5.5	V
Input High Voltage Except EXTAL, $\overline{\text{RESET}}$, MODA, MODB	V_{IH}	2.0	—	V_{CC}	V
Input Low Voltage Except EXTAL, MODA, MODB	V_{IL}	-0.5	—	0.8	V
Input High Voltage EXTAL	V_{IHC}	4.0	—	V_{CC}	V
Input Low Voltage EXTAL	V_{ILC}	-0.5	—	0.6	V
Input High Voltage $\overline{\text{RESET}}$	V_{IHR}	2.5	—	V_{CC}	V
Input High Voltage MODA and MODB	V_{IHM}	3.5	—	V_{CC}	V
Input Low Voltage MODA and MODB	V_{ILM}	-0.5	—	2.0	V
Input Leakage Current EXTAL, $\overline{\text{RESET}}$, MODA, MODB, $\overline{\text{BR}}$	I_{in}	-1	—	1	uA
Three-State (Off-State) Input Current (@2.4 V/0.4 V)	I_{TSI}	-10	—	10	uA
Output High Voltage (I_{OH} = -0.4 mA)	V_{OH}	2.4	—	—	V
Output Low Voltage (I_{OL} = 1.6 mA; $\overline{\text{RD}}$, $\overline{\text{WR}}$ I_{OL} = 1.6 mA; Open Drain $\overline{\text{HREQ}}$ I_{OL} = 6.7 mA, TXD IOL = 6.7 mA)	V_{OL}	—	—	0.4	V
Total Supply Current 5. 5 V, 27 MHz 5. 5 V, 20 MHz in WAIT Mode (see Note 1) in STOP Mode (see Note 1)	I_{DD27} I_{DD20} I_{DDW} I_{DDS}	— — — —	130 100 10 100	155 115 20 2000	mA mA mA µA
Input Capacitance (see Note 2)	Cin	—	10	—	pF

Notes:

1. In order to obtain these results all inputs must be terminated (i.e., not allowed to float).
2. Periodically sampled and not 100% tested.

AC Electrical Characteristics

The timing waveforms in the **AC Electrical Characteristics** are tested with a V_{IL} maximum of 0.5 V and a V_{IH} minimum of 2.4 V for all pins, except EXTAL, $\overline{\text{RESET}}$, MODA, and MODB. These four pins are tested using the input levels set forth in the **DC Electrical Characteristics**. AC timing specifications which are referenced to a device input signal are measured in production with respect to the 50% point of the respective input signal's transition. DSP56001 output levels are measured with the production test machine V_{OL} and V_{OH} reference levels set at 0.8 V and 2.0 V respectively.

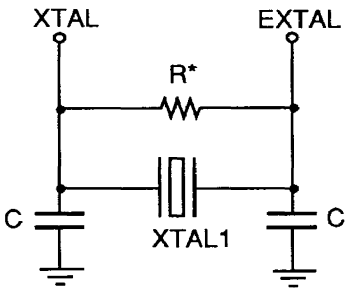
AC Electrical Characteristics - Clock Operation

The DSP56001 system clock may be derived from the on-chip crystal oscillator as shown in Clock Figure 1, or it may be externally supplied. An externally supplied square wave voltage source should be connected to EXTAL, leaving XTAL physically unconnected (see Clock Figure 2) to the board or socket. The rise and fall time of this external clock should be 5 ns maximum.

Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
	Frequency of Operation (EXTAL Pin)	4.0	20.5	4.0	27.0	MHz
1	External Clock Input High (tch) — EXTAL Pin (see Note 1 and 2)	22	150	17	150	ns
2	External Clock Input Low (tcl) — EXTAL Pin (see Note 1 and 2)	22	150	17	150	ns
3	Clock Cycle Time = cyc = 2T	48.75	250	37.5	250	ns
4	Instruction Cycle Time = Icyc = 4T	97.5	500	75	500	ns

Notes:

1. External Clock Input High and External Clock Input Low are measured at 50% of the input transition. tch and tcl are dependent on the duty cycle.
2. T = Icyc / 4 is used in the electrical characteristics. T represents an average which is independent of the duty cycle.

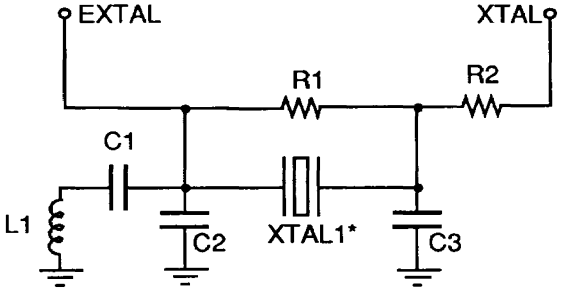


Fundamental Frequency Crystal Oscillator

Suggested Component Values

For f_{osc} = 4 MHz
R = 680KΩ ± 10%
C = 82 pf ± 20%
For f_{osc} = 20 MHz
R = 680KΩ ± 10%
C = 47 pf ± 20%
XTAL1 = 20.48 MHz

Notes:
*R limits crystal current



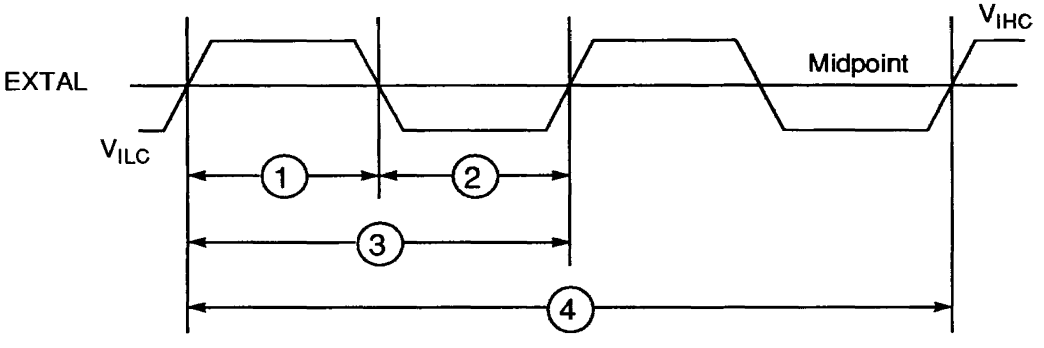
3rd Overtone Crystal Oscillator

Suggested Component Values

R1 = 470KΩ ± 10%
R2 = 330Ω ± 10%
C1 = 0.1 µf ± 20%
C2 = 26 pf ± 20%
C3 = 20 pf ± 10%
L1 = 5.21 µH ± 10%
XTAL = 27 MHz, AT cut 10 pf holder load, 50Ω max series resistance. (International Crystal Manufacturer part number 436161-27.00)

Notes:
(1) *3rd overtone crystal.
(2) R2 limits crystal current
(3) Reference Benjamin Parzen, The Design of Crystal and Other Harmonic Oscillators, John Wiley & Sons, 1983

Clock Figure 1. Crystal Oscillator Circuits



Note: The midpoint is V_{ILC} + 0.5 (V_{IHC} - V_{ILC}).

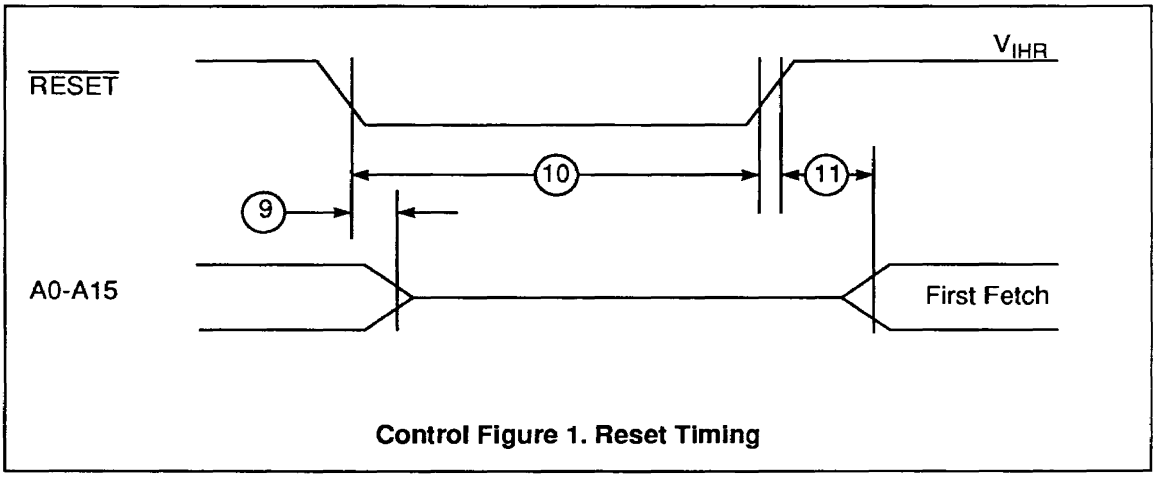
Clock Figure 2. External Clock Timing

DSP56001 Electrical Characteristics

AC Electrical Characteristics - Reset, Stop, Mode Select and Interrupt Timing

(V_{CC} = 5.0 Vdc ± 10%, T_J = -40 to +105° C, CL = 50 pF + 1 TTL Load)
(See Control Figure 1 through 8)
cyc = Clock cycle = 1/2 instruction cycle = 2 T cycles
WS = Number of wait states (1 WS = 1 cyc = 2T) programmed into external bus access using BCR (WS = 0 - 15)
tch = Clock high period
tcl = Clock low period

Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
9	Delay from $\overline{\text{RESET}}$ Assertion to Address High Impedance (periodically sampled and not 100% tested).	—	50	—	38	ns
10	Minimum Stabilization Duration — Internal Osc. (see Note 1)	75000•cyc	—	75000*cyc	—	ns
	Minimum Stabilization Duration — External clock (see Note 2)	25•cyc	—	25*cyc	—	ns
11	Delay from Asynchronous $\overline{\text{RESET}}$ Deassertion to First External Address Output (Internal Reset Negation)	8•cyc	9•cyc+40	8*cyc	9*cyc+31	ns
12	Synchronous Reset Setup Time from $\overline{\text{RESET}}$ Deassertion to Falling Edge of External Clock	20	cyc-10	15	cyc-8	ns
13	Synchronous Reset Delay Time from the Synchronous Falling Edge of External Clock to the First External Address Output	8•cyc+5	8•cyc+30	8*cyc+5	8*cyc+23	ns
14	Mode Select Setup Time	100	—	77	—	ns
15	Mode Select Hold Time	0	—	0	—	ns
16	Minimum Edge-Triggered Interrupt Request Width (assertion)	25	—	17	—	ns
16a	Minimum Edge-Triggered Interrupt Request Width (deassertion)	15	—	10	—	ns



Control Figure 1. Reset Timing

AC Electrical Characteristics - Reset, Stop, Mode Select, and Interrupt Timing (Continued)

NOTE

*When using fast interrupts and **IRQA and IRQB** are defined as **level**-sensitive, then timings 19 through 22 apply to prevent multiple interrupt service. To avoid these timing restrictions, the negative edge-triggered mode is recommended when using fast interrupt. Long interrupts are recommended when using level-sensitive mode.*

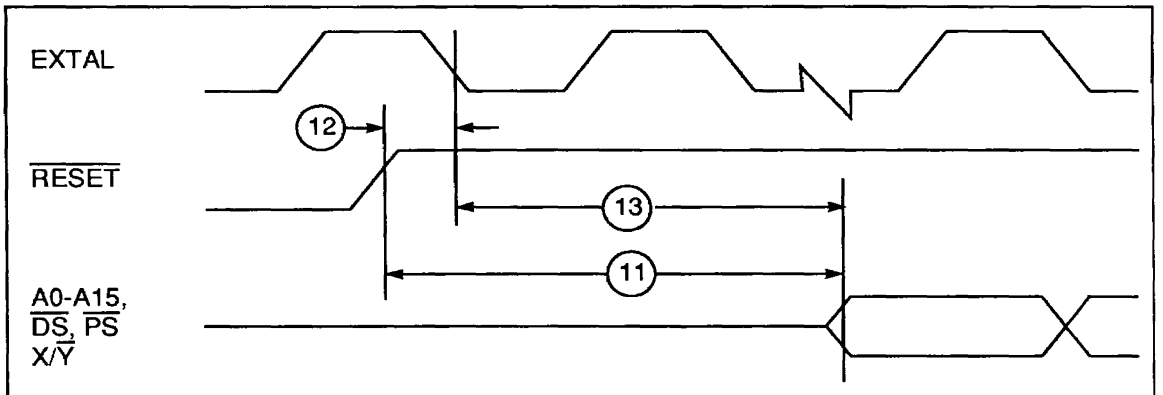
Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
17	Delay from $\overline{IRQA}$, $\overline{IRQB}$ Assertion to External Memory Access Address Out Valid Caused by First Interrupt Instruction Fetch Caused by First Interrupt Instruction Execution	 5*cyc+tch 9*cyc+tch	 — —	 5*cyc+tch 9*cyc+tch	 — —	 ns ns
18	Delay from $\overline{IRQA}$, $\overline{IRQB}$ Assertion to General Purpose Transfer Output Valid Caused by First Interrupt Instruction Execution	11*cyc+tch	—	11*cyc+tch	—	ns
19	Delay from Address Output Valid Caused by First Interrupt Instruction Execution to Interrupt Request Deassertion for Level Sensitive Fast Interrupts	—	2*cyc+tcl+(cyc *WS)-44	—	2*cyc+tcl+(cyc *WS)-34	ns
20	Delay from $\overline{RD}$ Assertion to Interrupt Request Deassertion for Level Sensitive Fast Interrupts	—	2*cyc+(cyc *WS)-40	—	2*cyc+(cyc *WS)-31	ns
21	Delay from $\overline{WR}$ Assertion to Interrupt Request Deassertion for Level Sensitive Fast Interrupts WS=0 WS>0	— —	2*cyc-40 cyc+tcl+(cyc *WS)-40	— —	2*cyc-31 cyc+tcl+(cyc *WS)-31	ns ns
22	Delay from General-Purpose Output Valid to Interrupt Request Deassertion for Level Sensitive Fast Interrupts - If Second Interrupt Instruction is: Single Cycle Two Cycles	— —	tcl-60 (2*cyc)+tcl-60	— —	tcl-46 (2*cyc)+tcl-46	ns ns

AC Electrical Characteristics - Reset, Stop, Mode Select, and Interrupt Timing (Continued)

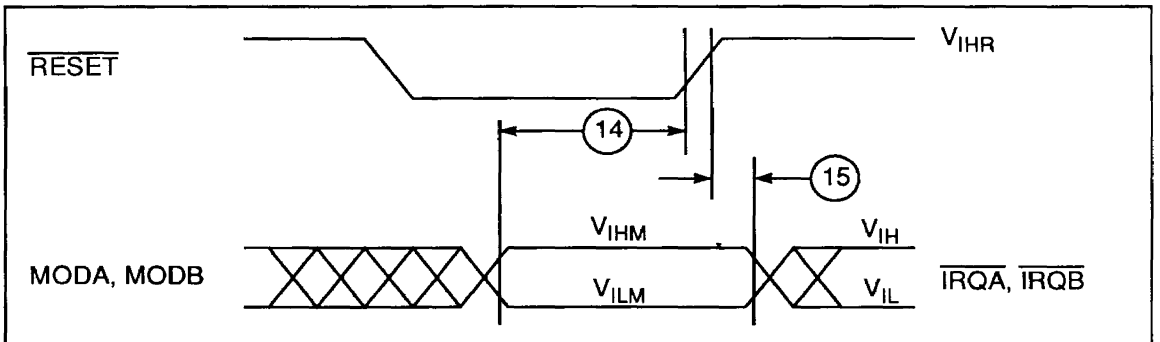
Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
23	Synchronous Interrupt Setup Time from $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$ Assertion to the Synchronous Rising Edge of External Clock (see Notes 5, 6)	25	cyc-10	19	cyc-8	ns
24	Synchronous Interrupt Delay Time from the Synchronous Rising Edge of External Clock to the First External Address Output Valid Caused by the First Instruction Fetch after Coming out of Wait State (see Notes 3, 5)	13*cyc+ tch+8	13*cyc+ tch+30	13*cyc+ tch+6	13*cyc+ tch+23	ns
25	Duration for $\overline{\text{IRQA}}$ Assertion to Recover from Stop State (see note 4)	25	—	19	—	ns
26	Delay from $\overline{\text{IRQA}}$ Assertion to Fetch of First Instruction (for Stop) for OMR bit 6 = 0 OMR bit 6 = 1 (see notes 1, 2, and 7).	 65545*cyc 17*cyc	 — —	 65545*cyc 17*cyc	 — —	 ns ns
27	Duration for Level Sensitive $\overline{\text{IRQA}}$ Assertion to Fetch of First Interrupt Instruction (for Stop) for OMR bit 6 = 0 OMR bit 6 = 1 (see notes 1, 2, and 7).	 65533*cyc+tcl 5*cyc+tcl	 — —	 65533*cyc+tcl 5*cyc+tcl	 — —	 ns ns
28	Delay from Level Sensitive $\overline{\text{IRQA}}$ Assertion to Fetch of First Interrupt Instruction (for Stop) for OMR bit 6 = 0 OMR bit 6 = 1 (see notes 1, 2, and 7).	 65545*cyc 17*cyc	 — —	 65545*cyc 17*cyc	 — —	 ns ns

Notes:

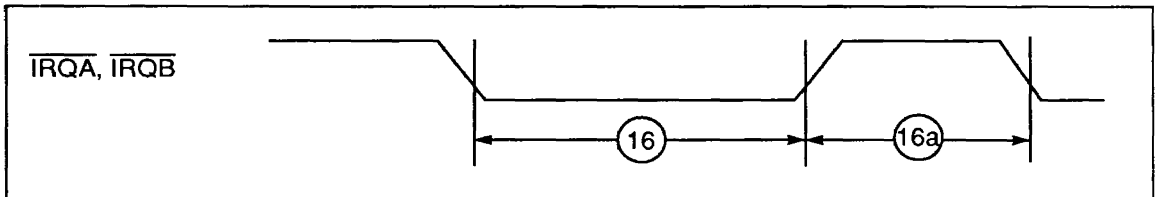
1. A clock stabilization delay is required when using the on-chip crystal oscillator in two cases:
   1) after power-on reset, and
   2) when recovering from Stop mode.
   During this stabilization period, T will not be constant. Since this stabilization period varies, a delay of 150,000T is typically allowed to assure that the oscillator is stabilized before executing programs. While it is possible to set OMR bit 6 = 1 when using the internal crystal oscillator, it is not recommended and these specifications do not guarantee timings for that case. See Section 8.5 in the *DSP56000/DSP56001 User's Manual* for additional information.
2. Circuit stabilization delay is required during reset when using an external clock in two cases:
   1) after power-on reset, and
   2) when recovering from Stop mode.
3. For Revision B silicon, the min and max numbers are 12cyc+Tch+8 and 12cyc+Tch+30, respectively.
4. The minimum is specified for the duration of an edge triggered $\overline{\text{IRQA}}$ interrupt required to recover from the STOP state without having the $\overline{\text{IRQA}}$ interrupt accepted.
5. Timing #23 is for all IRQx interrupts while timing #24 is only when exiting WAIT.
6. Timing #23 triggers off T1 in the normal state and off T1/T3 when exiting the WAIT state.
7. The timings in the table are for Rev. C parts. The timings for Rev. C parts are shorter by 1 cyc than the Rev. B parts when OMR6=0.



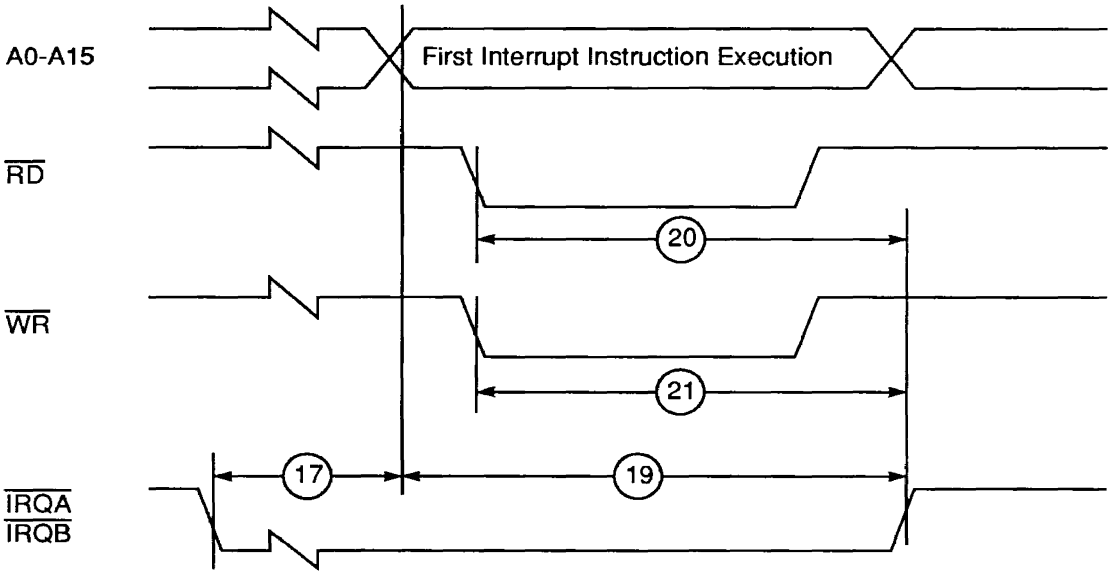
Control Figure 2. Synchronous Reset Timing



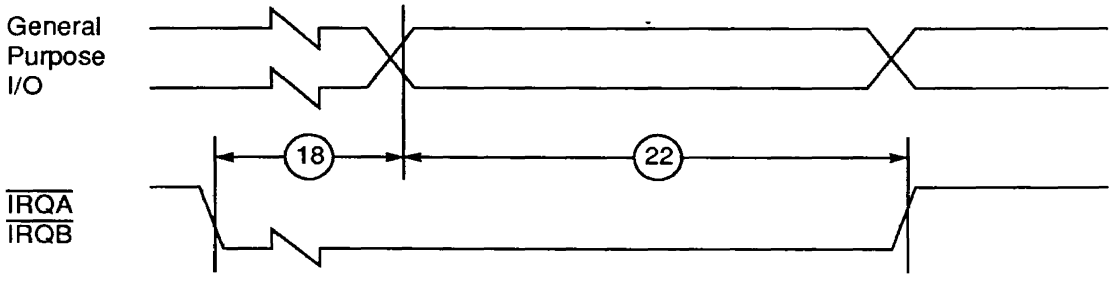
Control Figure 3. Operating Mode Select Timing



Control Figure 4. External Interrupt Timing (Negative Edge-Triggered)

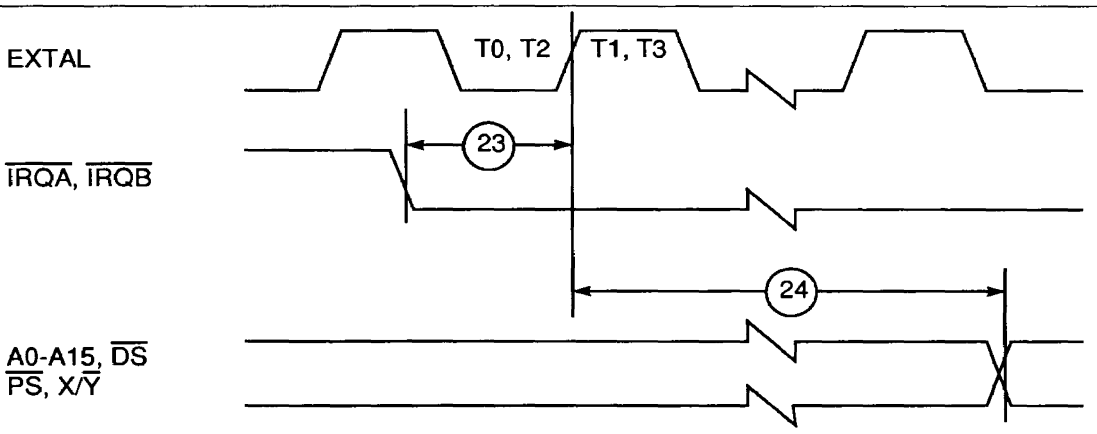


a) First Interrupt Instruction Execution

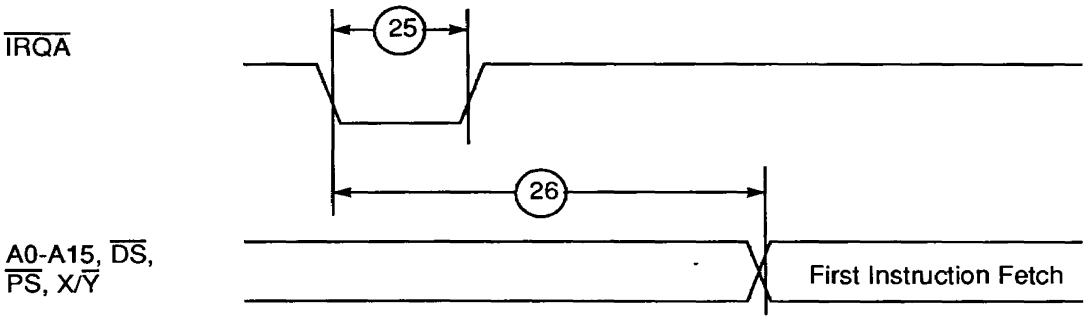


b) General Purpose I/O

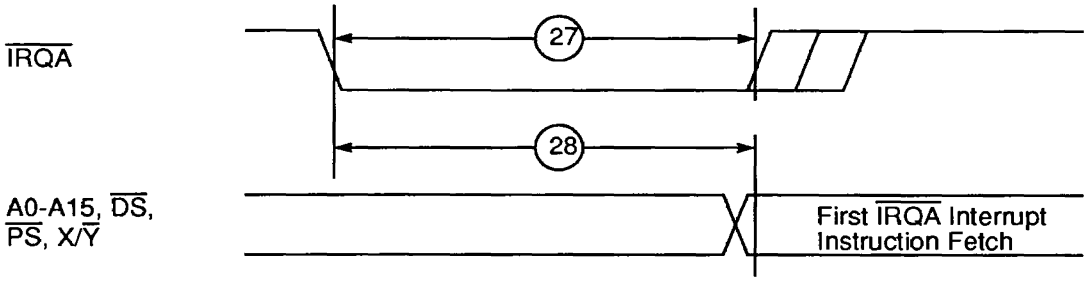
Control Figure 5. External Level-Sensitive Fast Interrupt Timing



Control Figure 6. Synchronous Interrupt and Synchronous Wait State Timing



Control Figure 7. Recovery from Stop State Using IRQA



Control Figure 8. Recovery from Stop State Using IRQA Interrupt Service

HOST PORT USAGE CONSIDERATIONS

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

Host Programmer Considerations

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

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

3. Synchronization of Status Bits from DSP to Host
   HC, HREQ, DMA, HF3, HF2, TRDY, TXDE, and RXDF (refer to *DSP56000/DSP56001 User's Manual*, I/O Interface section, Host/DMA Interface Programming Model for descriptions of these status bits) status bits are set or cleared from inside the DSP and read by the Host processor. The Host can read these status bits very quickly without regard to the clock rate used by the DSP, but the possibility exists that the state of the bit could be changing during the read operation. This is generally not a system problem, since the bit will be read correctly in the next pass of any Host polling routine.

   However, if the Host asserts the $\overline{\text{HEN}}$ for more than timing number 31a (T31a), with a minimum cycle time of timing number 32a (T32a), then the status is guaranteed to be stable.

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

   Solution:
   a. Read the bits twice and check for consensus.
   b. Assert $\overline{\text{HEN}}$ access for T31a so that status bit transitions are stabilized.

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

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

DSP Programmer Considerations

1. Reading HF0 and HF1 as an Encoded Pair
   DMA, HF1, HF0, and HCP, HTDE, and HRDF (refer to *DSP56000/DSP56001 User's Manual*, I/O Interface section, Host/DMA Interface Programming Model for descriptions of these status bits) status bits are set or cleared by the Host processor side of the interface. These bits are individually synchronized to the DSP clock.

   A potential problem exists when reading status bits HF1 and HF2 as an encoded pair, i.e., the four combinations 00, 01, 10, and 11 each have significance. A very small probability exists that the DSP will read the status bits synchronized during transition. The solution to this potential problem is to read the bits twice for consensus.

DSP56001 Electrical Characteristics

AC Electrical Characteristics - Host I/O Timing

(V_{CC} = 5.0 Vdc ± 10%, T_J = -40 to +105° C, CL = 50 pF + 1 TTL Load,
see Host Figures 1 through 6)
cyc = Clock cycle = 1/2 instruction cycle = 2 T cycles
tHSDL = Host Synchronization Delay Time
Active low lines should be "pulled up" in a manner consistent with the AC and DC specifications

Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
30	Host Synchronous Delay (see Note 1)	tcl	cyc+tcl	tcl	cyc+tcl	ns
31	$\overline{HEN}/\overline{HACK}$ Assertion Width (see Note 2) a. CVR, ICR, ISR Read (see Note 4) b. Read c. Write	 cyc+60 50 25	 — — —	 cyc+46 39 19	 — — —	ns
32	$\overline{HEN}/\overline{HACK}$ Deassertion Width (see Note 2 and 5)	25	—	19	—	ns
32a	Minimum Cycle Time Between Two $\overline{HEN}$ Assertion for Consecutive CVR, ICR, and ISR Reads (see Note 2)	2•cyc+60	—	2•cyc+46	—	ns
33	Host Data Input Setup Time Before $\overline{HEN}/\overline{HACK}$ Deassertion	5	—	4	—	ns
34	Host Data Input Hold Time After $\overline{HEN}/\overline{HACK}$ Deassertion	5	—	4	—	ns
35	$\overline{HEN}/\overline{HACK}$ Assertion to Output Data Active from High Impedance	0	—	0	—	ns
36	$\overline{HEN}/\overline{HACK}$ Assertion to Output Data Valid (periodically sampled, and not 100% tested)	—	50	—	39	ns
37	$\overline{HEN}/\overline{HACK}$ Deassertion to Output Data High Impedance	—	35	—	27	ns
38	Output Data Hold Time After $\overline{HEN}/\overline{HACK}$ Deassertion	5	—	4	—	ns
39	HR/$\overline{W}$ Low Setup Time Before $\overline{HEN}$ Assertion	0	—	0	—	ns
40	HR/$\overline{W}$ Low Hold Time After $\overline{HEN}$ Deassertion	5	—	4	—	ns
41	HR/$\overline{W}$ High Setup Time to $\overline{HEN}$ Assertion	0	—	0	—	ns
42	HR/$\overline{W}$ High Hold Time After $\overline{HEN}/\overline{HACK}$ Deassertion	5	—	4	—	ns
43	HA0-HA2 Setup Time Before $\overline{HEN}$ Assertion	0	—	0	—	ns
44	HA0-HA2 Hold Time After $\overline{HEN}$ Deassertion	5	—	4	—	ns
45	DMA $\overline{HACK}$ Assertion to $\overline{HREQ}$ Deassertion (see Note 3)	5	60	4	46	ns

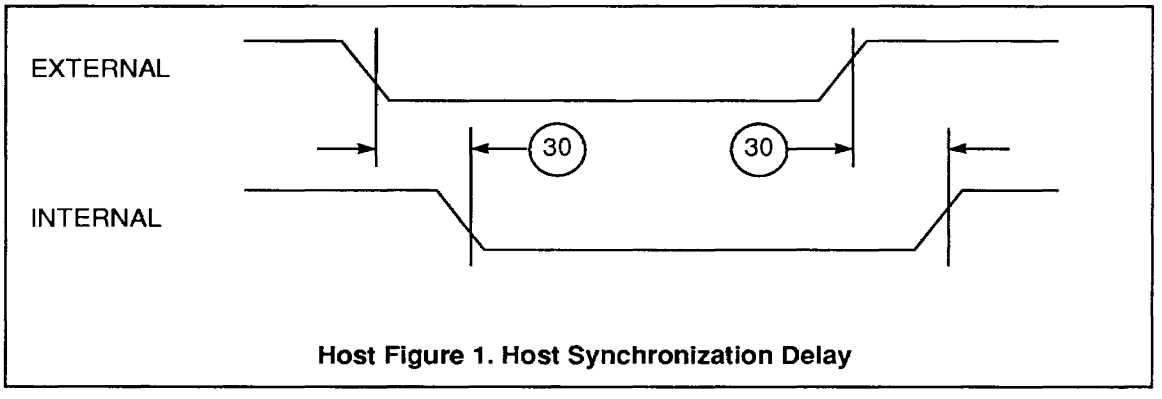
AC Electrical Characteristics - Host I/O Timing (Continued)

(V_{CC} = 5.0 Vdc ± 10%, T_J = -40 to +105° C, CL = 50 pF + 1 TTL Load,
see Host Figures 1 through 6)
cyc = Clock cycle = 1/2 instruction cycle = 2 T cycles
tHSDL = Host Synchronization Delay Time
Active low lines should be "pulled up" in a manner consistent with the AC and DC specifications

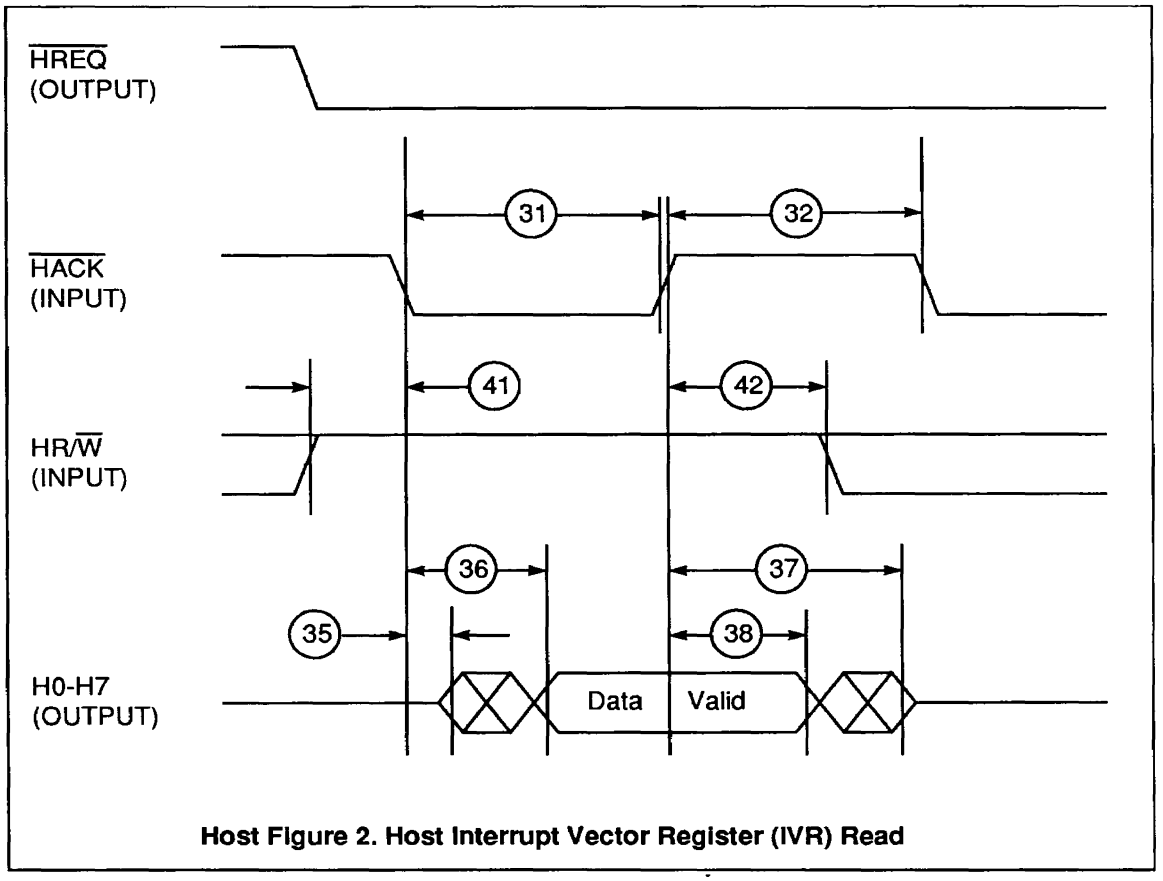
Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
46	DMA $\overline{\text{HACK}}$ Deassertion to $\overline{\text{HREQ}}$ Assertion (see Note 3) for DMA RXL Read	tHSDL+cyc+tch+5	—	tHSDL+cyc+tch+4	—	ns
	for DMA TXL Write	tHSDL+cyc+5	—	tHSDL+cyc+4	—	ns
	for All Other Cases	5	—	4	—	ns
47	Delay from $\overline{\text{HEN}}$ Deassertion to $\overline{\text{HREQ}}$ Assertion for RXL Read (see Note 3)	tHSDL+cyc+tch+5	—	tHSDL+cyc+tch+4	—	ns
48	Delay from $\overline{\text{HEN}}$ Deassertion to $\overline{\text{HREQ}}$ Assertion for TXL Write (see Note 3)	tHSDL+cyc+5	—	tHSDL+cyc+4	—	ns
49	Delay from $\overline{\text{HEN}}$ Assertion to $\overline{\text{HREQ}}$ Deassertion for RXL Read, TXL Write (see Note 3)	5	75	4	70	ns

Notes:

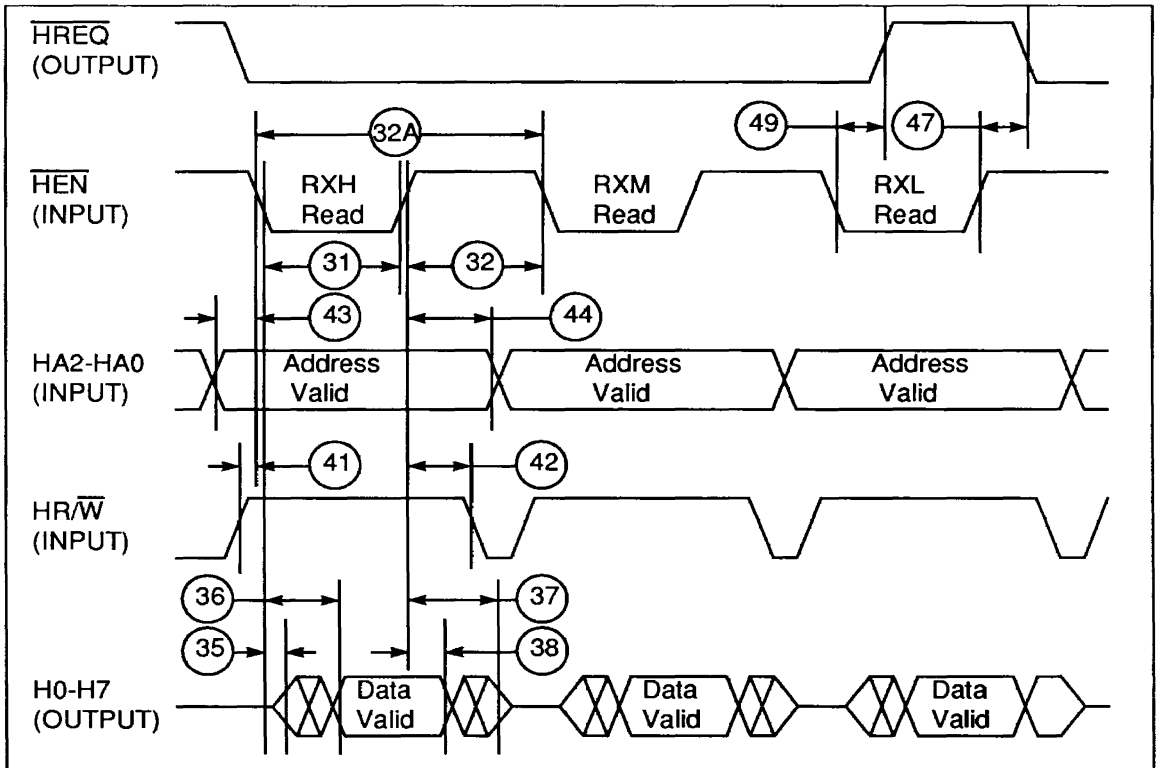
1. "Host synchronization delay (tHSDL)" is the time period required for the DSP56001 to sample any external asynchronous input signal, determine whether it is high or low, and synchronize it to the DSP56001 internal clock.
2. See **HOST PORT USAGE CONSIDERATIONS**.
3. $\overline{\text{HREQ}}$ is pulled up by a 1kΩ resistor.
4. This timing must be adhered to only if two consecutive reads from one of these registers are executed.
5. It is recommended that timing #32 be 2cyc+tch+10 min for 20.5 MHz and 2cyc+tch+7 min for 27 MHz if two consecutive writes to TXL are executed without polling TXDE or $\overline{\text{HREQ}}$.



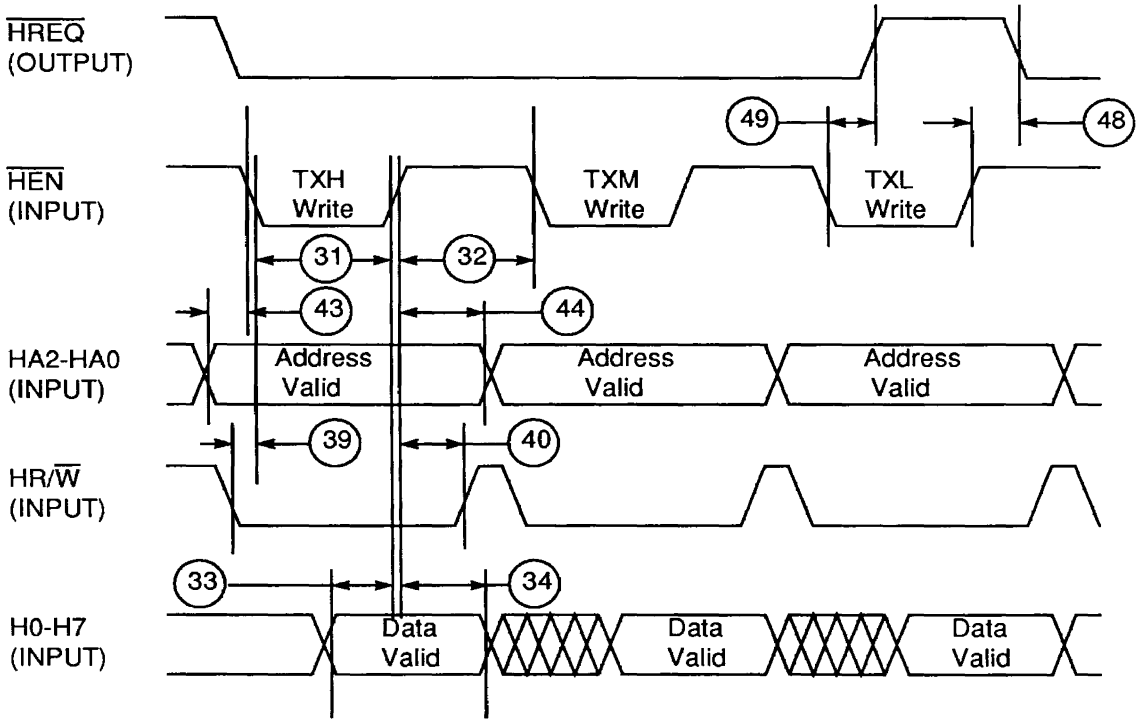
Host Figure 1. Host Synchronization Delay



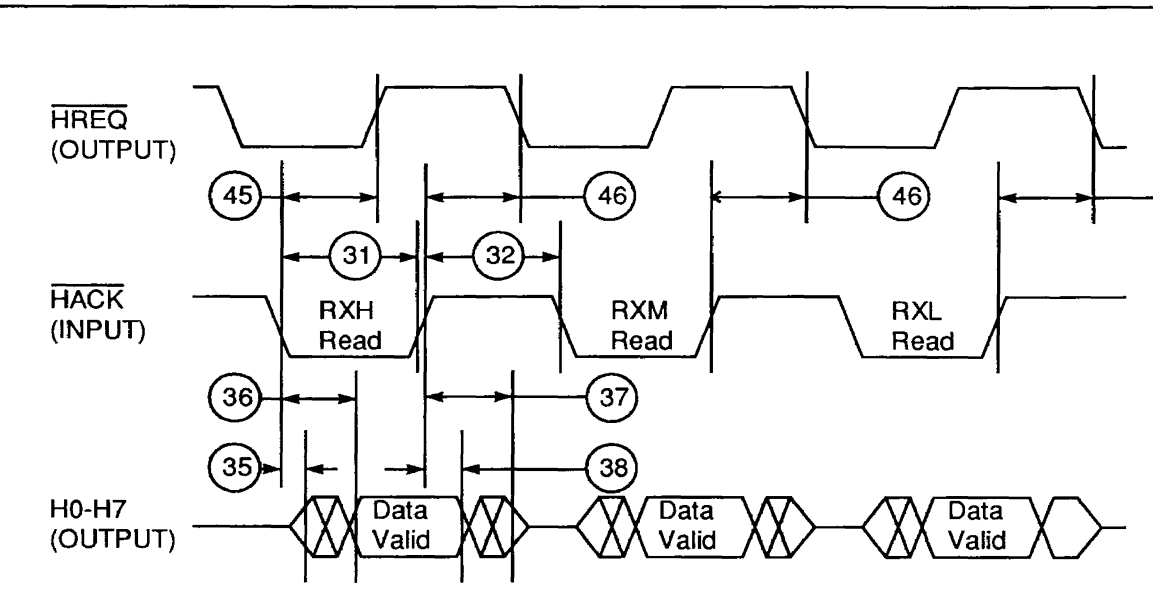
Host Figure 2. Host Interrupt Vector Register (IVR) Read



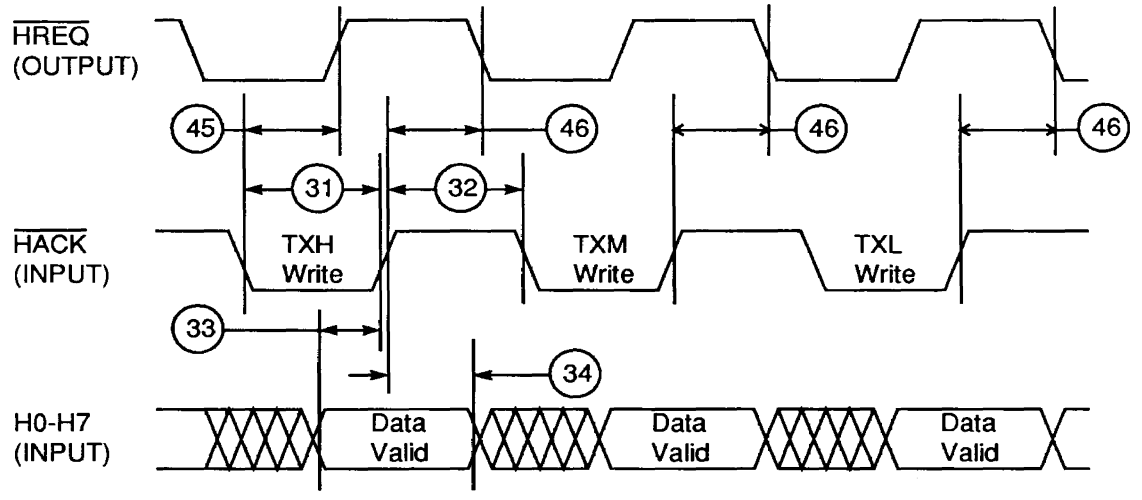
Host Figure 3. Host Read Cycle (Non-DMA Mode)



Host Figure 4. Host Write Cycle (Non-DMA Mode)



Host Figure 5. Host DMA Read Cycle



Host Figure 6. Host DMA Write Cycle

DSP56001 Electrical Characteristics

AC Electrical Characteristics - SCI Timing

(V_{CC} = 5.0 Vdc ± 10%, T_J = -40 to +105° C, CL = 50 pF + 1 TTL Load,
see SCI Figures 1 and 2)
cyc = Clock cycle = 1/2 instruction cycle = 2 T cycles
tSCC = Synchronous Clock Cycle Time (for internal clock tSCC is determined by the SCI clock control register and Icyc.)

SCI Synchronous Mode Timing

Num.	Characteristics	20.5 MHz		27 MHz		Unit
		Min	Max	Min	Max	
55	Synchronous Clock Cycle — tSCC	8*cyc	—	8*cyc	—	ns
56	Clock Low Period	tSCC/2-20	—	tSCC/2-15	—	ns
57	Clock High Period	tSCC/2-20	—	tSCC/2-15	—	ns
59	Output Data Setup to Clock Falling Edge (Internal Clock)	tSCC/4+tcl-50	—	tSCC/4+tcl-39	—	ns
60	Output Data Hold After Clock Rising Edge (Internal Clock)	tSCC/4-tcl-15	—	tSCC/4-tcl-11	—	ns
61	Input Data Setup Time Before Clock Rising Edge (Internal Clock)	tSCC/4+tcl+45	—	tSCC/4+tcl+35	—	ns
62	Input Data Not Valid Before Clock Rising Edge (Internal Clock)	—	tSCC/4+tcl-10	—	tSCC/4+tcl-8	ns
63	Clock Falling Edge to Output Data Valid (External Clock)	—	63	—	48	ns
64	Output Data Hold After Clock Rising Edge (External Clock)	cyc+12	—	cyc+9	—	ns
65	Input Data Setup Time Before Clock Rising Edge (External Clock)	30	—	23	—	ns
66	Input Data Hold Time After Clock Rising Edge (External Clock)	40	—	31	—	ns

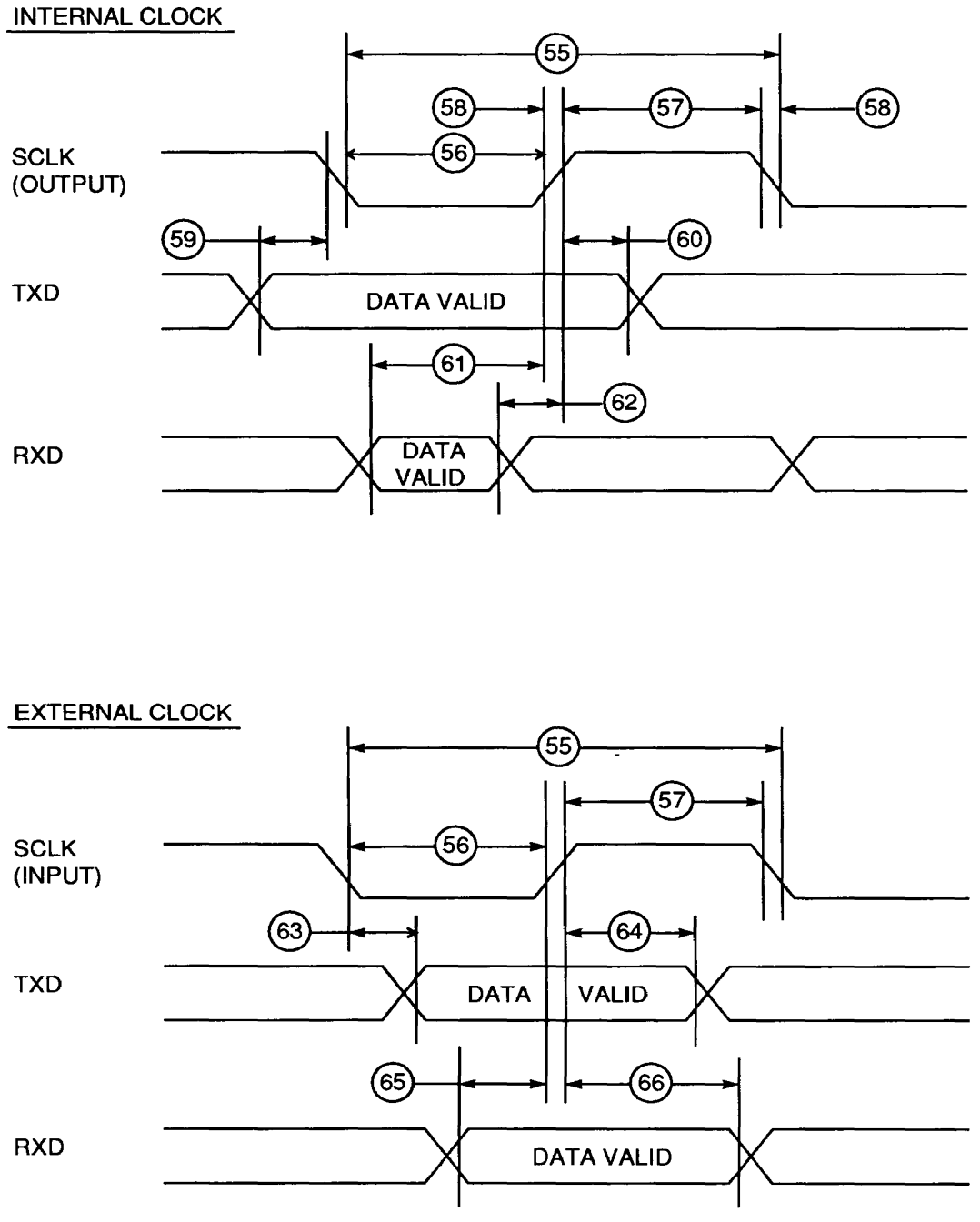
DSP56001 Electrical Characteristics

AC Electrical Characteristics - SCI Timing

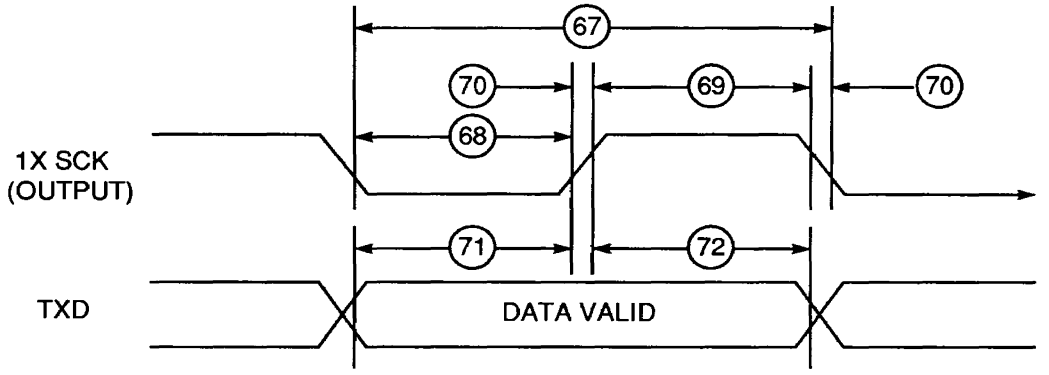
(V_{CC} = 5.0 Vdc ± 10%, T_J = -40 to +105° C, CL = 50 pF + 1 TTL Load,
see SCI Figures 1 and 2)
cyc = Clock cycle = 1/2 instruction cycle = 2 T cycles
tACC = Asynchronous clock cycle time
tACC = Asynchronous Clock Cycle Time (for internal clock tACC is determined by the SCI clock control register and Icyc)

SCI Asynchronous Mode Timing - 1X Clock

Num.	Characteristics	20.5 MHz		27 MHz		Unit
		Min	Max	Min	Max	
67	Asynchronous Clock Cycle	64•cyc	—	64•cyc	—	ns
68	Clock Low Period	tACC/2-20	—	tACC/2-15	—	ns
69	Clock High Period	tACC/2-20	—	tACC/2-15	—	ns
71	Output Data Setup to Clock Rising Edge (Internal Clock)	tACC/2-100	—	tACC/2-77	—	ns
72	Output Data Hold After Clock Rising Edge (Internal Clock)	tACC/2-100	—	tACC/2-77	—	ns



SCI Figure 1. SCI Synchronous Mode Timing



Note: In the wire-OR mode, TXD can be pulled up by 1KΩ

SCI Figure 2. SCI Asynchronous Mode Timing

DSP56001 Electrical Characteristics

AC Electrical Characteristics - SSI Timing

(V_{CC} = 5.0 Vdc ± 10%, T_J = -40 to +105° C, CL = 50 pF + 1 TTL Load,
see SSI Figures 1 and 2)
cyc = Clock cycle = 1/2 instruction cycle = 2 T cycles
tSSICC = SSI clock cycle time
TXC (SCK Pin) = Transmit Clock
RXC (SC0 or SCK Pin) = Receive Clock
FST (SC2 Pin) = Transmit Frame Sync
FSR (SC1 or SC2 Pin) = Receive Frame Sync
i ck = Internal Clock
x ck = External Clock
g ck = Gated Clock
i ck a = Internal Clock, Asynchronous Mode (Asynchronous implies that TXC and RXC are two different clocks)
i ck s = Internal Clock, Synchronous Mode (Synchronous implies that TXC and RXC are the same clock)
bl = bit length
wl = word length

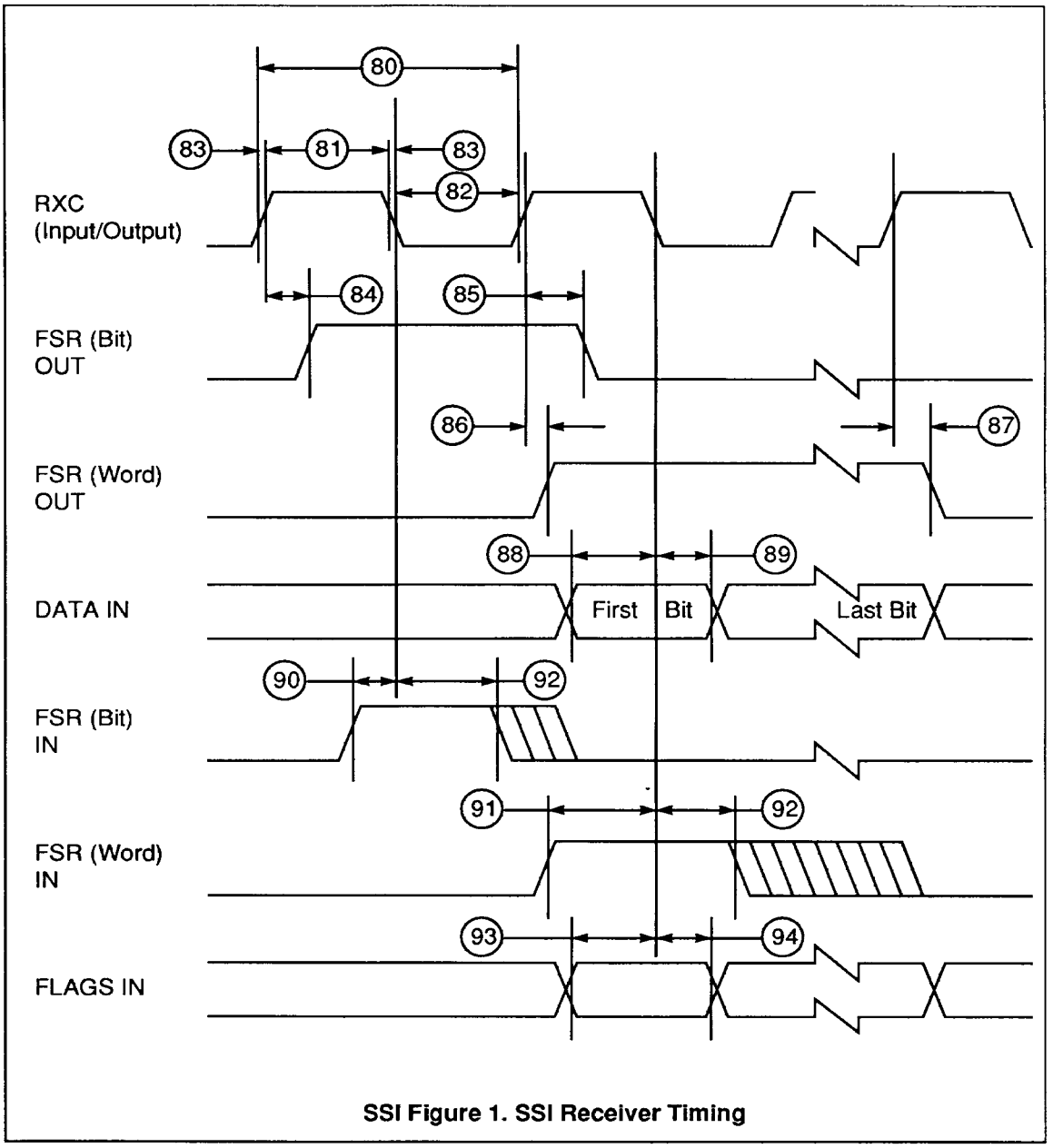
Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Case	Unit
80	Clock Cycle (see Note 1)	4*cyc	—	4*cyc	—	—	ns
81	Clock High Period	tSSICC/2-20	—	tSSICC/2-15	—	—	ns
82	Clock Low Period	tSSICC/2-20	—	tSSICC/2-15	—	—	ns
84	RXC Rising Edge to FSR Out (bl) High	— —	80 50	— —	61 38	x ck i ck a	ns
85	RXC Rising Edge to FSR Out (bl) Low	— —	70 40	— —	54 31	x ck i ck a	ns
86	RXC Rising Edge to FSR Out (wl) High	— —	70 40	— —	54 31	x ck i ck a	ns
87	RXC Rising Edge to FSR Out (wl) Low	— —	70 40	— —	54 31	x ck i ck a	ns
88	Data In Setup Time Before RXC (SCK in Synchronous Mode) Falling Edge	15 35 25	— — —	12 27 19	— — —	x ck i ck a i ck s	ns
89	Data In Hold Time After RXC Falling Edge	35 5	— —	27 4	— —	x ck i ck	ns
90	FSR Input (bl) High Before RXC Falling Edge	15 35	— —	12 27	— —	x ck i ck a	ns
91	FSR Input (wl) High Before RXC Falling Edge	20 55	— —	15 42	— —	x ck i ck a	ns
92	FSR Input Hold Time After RXC Falling Edge	35 5	— —	27 4	— —	x ck i ck	ns

AC Electrical Characteristics - SSI Timing (Continued)

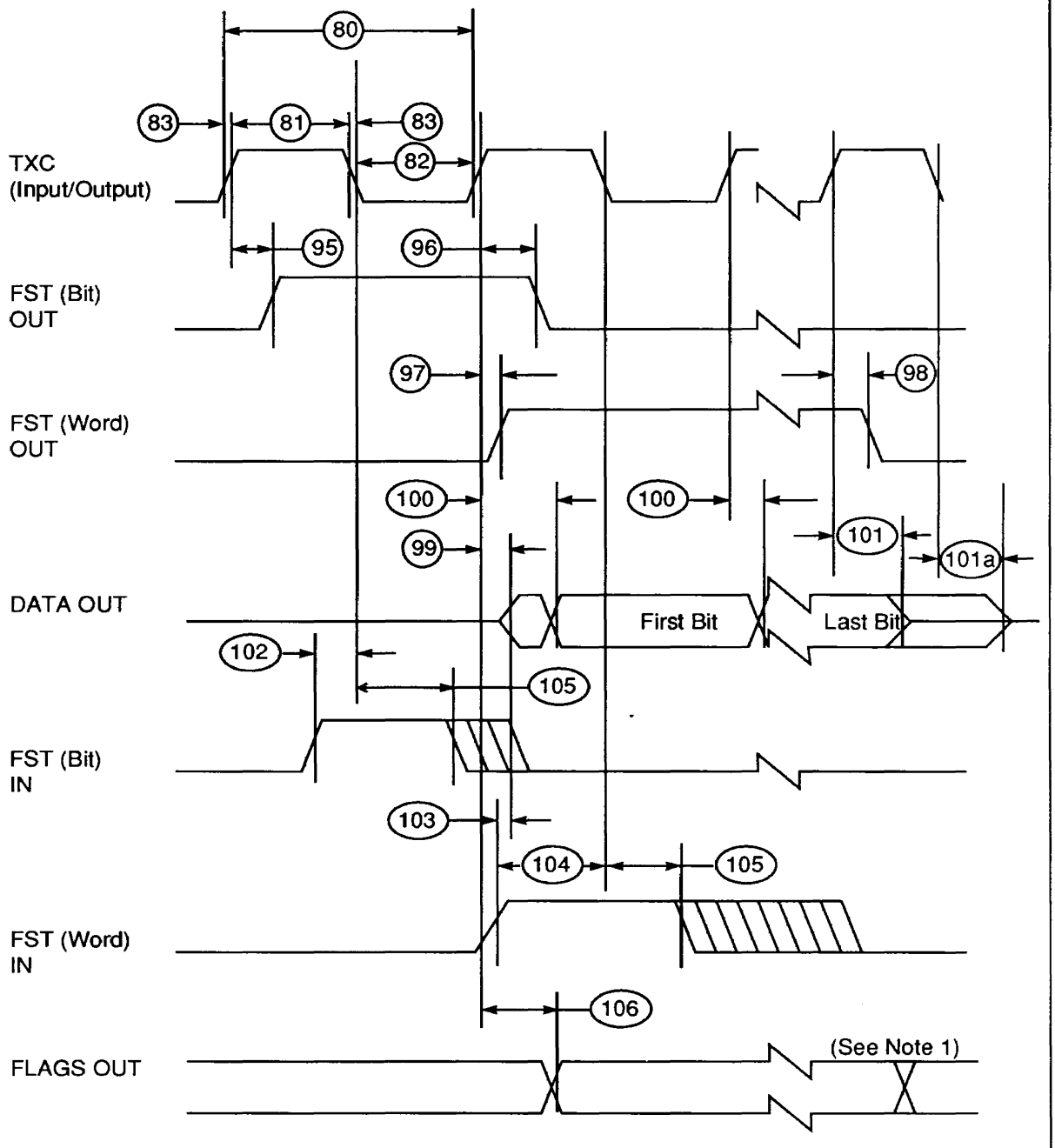
Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Case	Unit
93	Flags Input Setup Before RXC Falling Edge	30 50	— —	23 39	— —	x ck i ck s	ns
94	Flags Input Hold Time After RXC Falling Edge	35 5	— —	27 4	— —	x ck i ck s	ns
95	TXC Rising Edge to FST Out (bl) High	— —	70 30	— —	54 23	x ck i ck	ns
96	TXC Rising Edge to FST Out (bl) Low	— —	65 35	— —	50 27	x ck i ck	ns
97	TXC Rising Edge to FST Out (wl) High	— —	65 35	— —	50 27	x ck i ck	ns
98	TXC Rising Edge to FST Out (wl) Low	— —	65 35	— —	50 27	x ck i ck	ns
99	TXC Rising Edge to Data Out Enable from High Impedance	— —	65 40	— —	50 31	x ck i ck	ns
100	TXC Rising Edge to Data Out Valid	— —	65 40	— —	50 31	x ck i ck	ns
101	TXC Rising Edge to Data Out High Impedance (periodically sampled, and not 100% tested)	— —	70 40	— —	54 31	x ck i ck	ns
101a	TXC Falling Edge to Data Out High Impedance for Gated Clock Mode Only	cyc+tch	—	cyc+tch	—	g ck	ns
102	FST Input (bl) Setup Time Before TXC Falling Edge	15 35	— —	12 27	— —	x ck i ck	ns
103	FST Input (wl) to Data Out Enable from High Impedance	—	60	—	46	—	ns
104	FST Input (wl) Setup Time Before TXC Falling Edge	20 55	— —	15 42	— —	x ck i ck	ns
105	FST Input Hold Time After TXC Falling Edge	35 5	— —	27 4	— —	x ck i ck	ns
106	Flag Output Valid After TXC Rising Edge	— —	70 40	— —	54 31	x ck i ck	ns

Note:

1. For internal clock, External Clock Cycle is defined by Icyc and SSI control register.



SSI Figure 1. SSI Receiver Timing



Note:

1. In the Network mode, output flag transitions can occur at the start of each time slot within the frame. In the Normal mode, the output flag state is asserted for the entire frame period.

SSI Figure 2. SSI Transmitter Timing

DSP56001 Electrical Characteristics

AC Electrical Characteristics — Capacitance Derating — External Bus Asynchronous Timing

V_{CC} = 5.0 Vdc ± 10%, T_J = -40 to +105° C, CL = 50 pF + 1 TTL Load, see Bus Figures 1 and 2)

cyc = Clock cycle = 1/2 instruction cycle = 2 T cycles
WS = Number of Wait States, Determined by BCR Register (WS = 0 to 15)

The DSP56001 External Bus Timing Specifications are designed and tested at the maximum capacitive load of 50 pF, including stray capacitance. Typically, the drive capability of the External Bus pins (A0-A15, D0-D23, $\overline{PS}$, $\overline{DS}$, $\overline{RD}$, $\overline{WR}$, X/$\overline{Y}$) derates linearly at 1 ns per 12 pF of additional capacitance from 50 pF to 250 pF of loading. Port B and C pins derate linearly at 1 ns per 5 pF of additional capacitance from 50 pF to 250 pF of loading.

Active low inputs should be "pulled up" in a manner consistent with the AC and DC specifications.

To conserve power, when an internal memory access follows an external memory access, the $\overline{RD}$ and $\overline{WR}$ strobes remain deasserted and A0-A15 and X/$\overline{Y}$ do not change from their previous state. Both $\overline{PS}$ and $\overline{DS}$ will be deasserted (they do not change between two external accesses to the same memory space) indicating that no external memory access is occurring. If $\overline{BR}$ has been asserted, then the bus signals will be three-stated according to the timing information in this data sheet.

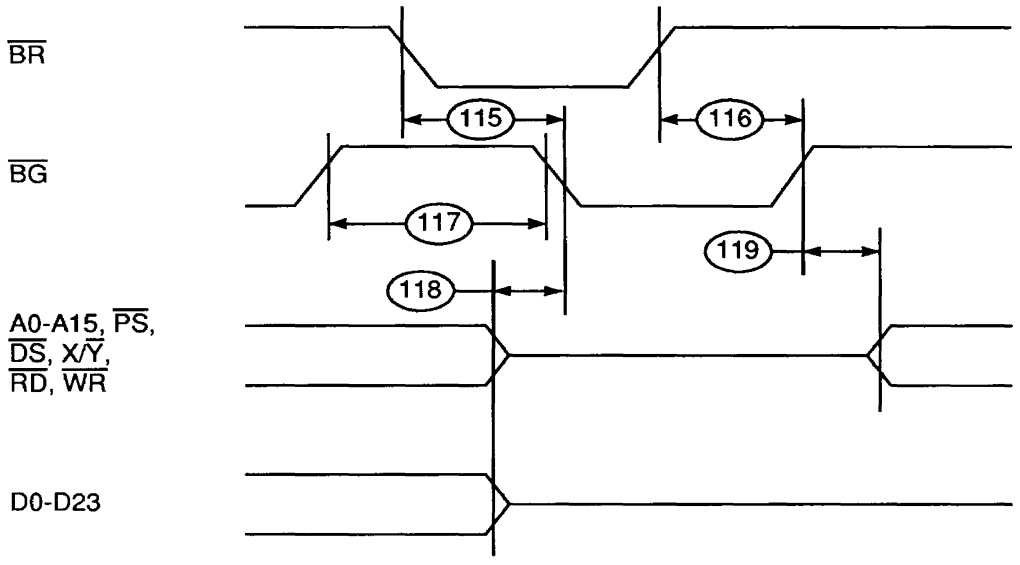
Num.	Characteristics		20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
115	Delay from $\overline{BR}$ Assertion to $\overline{BG}$ Assertion	(see Note 1)	2*cyc+tch	4*cyc+tch+20	2*cyc+tch	4*cyc+tch+15	ns
		(see Note 2)	cyc+tch	4*cyc+tch+(cyc*WS)+20	cyc+tch	4*cyc+tch+(cyc*WS)+15	ns
		(see Note 3)	cyc+tch	6*cyc+tch+(2*cyc*WS)+20	cyc+tch	6*cyc+tch+(2*cyc*WS)+15	ns
		(see Note 4)	Infinity	—	Infinity	—	
		(see Note 5)	tch+4	cyc+tch+30	tch+3	cyc+tch+23	ns
116	Delay from $\overline{BR}$ Deassertion to $\overline{BG}$ Deassertion		2*cyc	4*cyc+20	2*cyc	4*cyc+15	ns
117	$\overline{BG}$ Deassertion Duration		2*cyc-10	—	2*cyc-8	—	ns
118	Delay from Address, Data, and Control Bus High Impedance to $\overline{BG}$ Assertion		0	—	0	—	ns
119	Delay from $\overline{BG}$ Deassertion to Address, Data, and Control Bus Enabled		—	tch-10	—	tch-8	ns
120	Address Valid to $\overline{WR}$ Assertion	WS = 0	tcl-9	tcl+5	tcl-7	tcl+5	ns
		WS > 0	cyc-9	cyc+5	cyc-7	cyc+5	
121	$\overline{WR}$ Assertion Width	WS = 0	cyc-9	—	cyc-7	—	ns
		WS > 0	WS*cyc+tcl-9	—	WS*cyc+tcl-7	—	
122	$\overline{WR}$ Deassertion to Address Not Valid		tch-12	—	tch-9	—	ns
123	$\overline{WR}$ Assertion to Data Out Valid	WS = 0	tch-9	tch+10	tch-7	tch+8	ns
		WS > 0	0	10	0	8	
124	Data Out Hold Time from $\overline{WR}$ Deassertion (the maximum specification is periodically sampled, and not 100% tested)		tch-9	tch+7	tch-7	tch+6	ns
125	Data Out Setup Time to $\overline{WR}$ Deassertion (see note 6)	WS = 0	tcl-5	—	tcl-5	—	ns
		WS > 0	WS*cyc+tcl-5	—	WS*cyc+tcl-5	—	
126	$\overline{RD}$ Deassertion to Address Not Valid		tch-9	—	tch-7	—	ns

AC Electrical Characteristics - External Bus Asynchronous Timing (Continued)

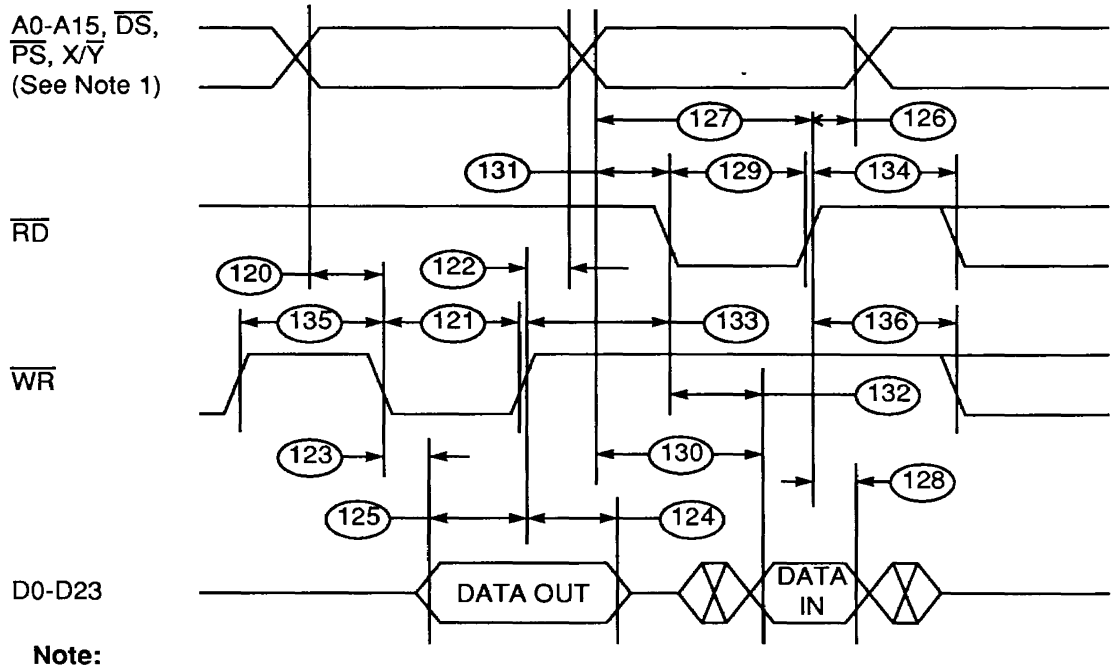
Num.	Characteristics		20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
127	Address Valid to $\overline{RD}$ deassertion	WS = 0	cyc+tcl-8	—	cyc+tcl-6	—	ns
		WS > 0	((WS+1)•cyc)+tcl-8	—	((WS+1)•cyc)+tcl-6	—	ns
128	Input Data Hold Time to $\overline{RD}$ Deassertion		2	—	2	—	ns
129	$\overline{RD}$ Assertion Width	WS = 0	cyc-9	—	cyc-7	—	ns
		WS > 0	((WS+1)•cyc)-9	—	((WS+1)•cyc)-7	—	
130	Address Valid to Input Data Valid	WS = 0	—	cyc+tcl-18	—	cyc+tcl-14	ns
		WS > 0	—	((WS+1)•cyc)+tcl-18	—	((WS+1)•cyc)+tcl-14	
131	Address Valid to $\overline{RD}$ Assertion		tcl-9	tcl+5	tcl-7	tcl+5	ns
132	$\overline{RD}$ Assertion to Input Data Valid	WS = 0	—	cyc-14	—	cyc-11	ns
		WS > 0	—	((WS+1)•cyc)-14	—	((WS+1)•cyc)-11	
133	$\overline{WR}$ Deassertion to $\overline{RD}$ Assertion		cyc-15	—	cyc-12	—	ns
134	$\overline{RD}$ Deassertion to $\overline{RD}$ Assertion		cyc-10	—	cyc-8	—	ns
135	$\overline{WR}$ Deassertion to $\overline{WR}$ Assertion	WS = 0	cyc-15	—	cyc-12	—	ns
		WS > 0	cyc+tch-15	—	cyc+tch-12	—	ns
136	$\overline{RD}$ Deassertion to $\overline{WR}$ Assertion	WS = 0	cyc-10	—	cyc-8	—	ns
		WS > 0	cyc+tch-10	—	cyc+tch-8	—	ns

Notes:

1. With no external access from the DSP.
2. During external read or write access.
3. During external read-modify-write access.
4. During the STOP mode the external bus will not be released and $\overline{BG}$ will not go low. However, if the bus is released ($\overline{BG}$ = 0) and the STOP instruction is executed while $\overline{BG}$ = 0 then the bus will remain released while the DSP is in the stop state and $\overline{BG}$ will remain low.
5. During the WAIT mode the $\overline{BR}$/$\overline{BG}$ circuits remain active.
6. Typical values at 5V are:
   at 20.5 MHz and WS=0, Min = tcl-4
   at 20.5 MHz and WS>0, Min = WS•cyc+tcl-4
   at 27 MHz and WS=0, Min = tcl-3
   at 27 MHz and WS>0, Min = WS•cyc+tcl-3



Async. Bus Figure 1. Bus Request / Bus Grant Timing



Note:
1. During Read-Modify-Write instructions and internal instructions, the address lines do not change state.

Async. Bus Figure 2. External Bus Asynchronous Timing

DSP56001 Electrical Characteristics

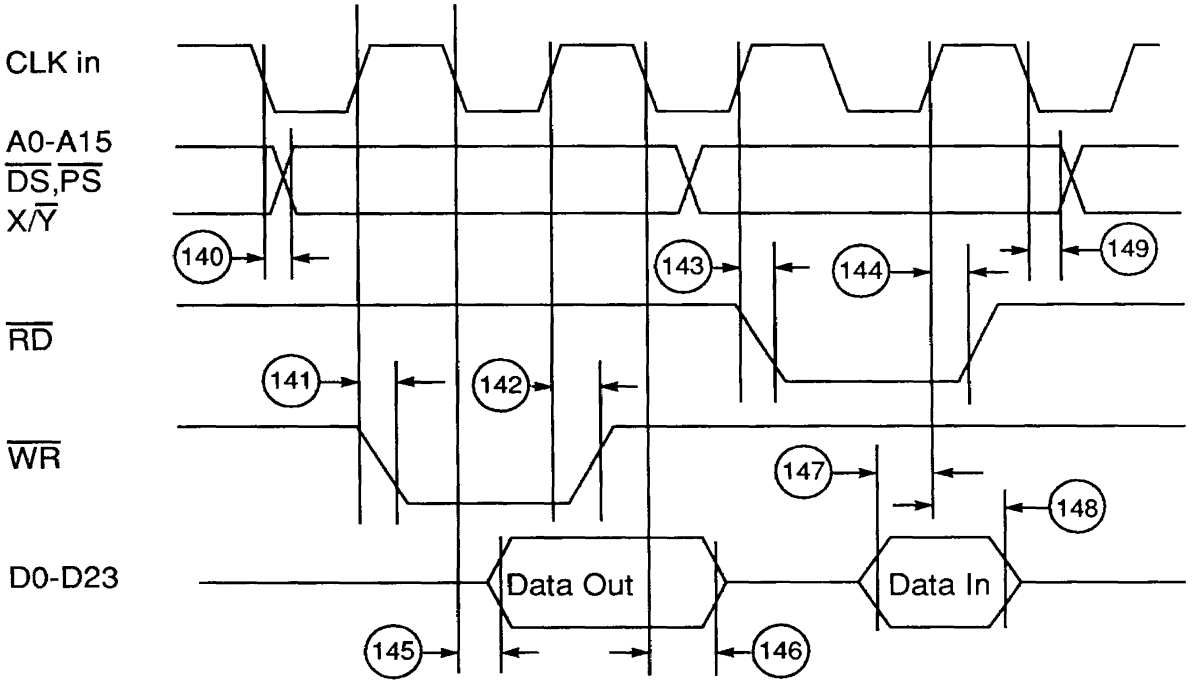
AC Electrical Characteristics - External Bus Synchronous Timing

V_{CC} = 5.0 Vdc ± 10%; T_J = -40 to 105° C

Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
140	Clk Low Transition To Address Valid	—	24	—	19	ns
141	Clk High Transition To $\overline{WR}$ Assertion WS=0 (see Note 2) WS>0	0 0	19 tch+19	0 0	15 tch+15	ns ns
142	Clk High Transition To $\overline{WR}$ Deassertion	5	21	5	16	ns
143	Clk High Transition To $\overline{RD}$ Assertion	0	19	0	15	ns
144	Clk High Transition To $\overline{RD}$ Deassertion	5	17	5	13	ns
145	Clk Low Transition To Data-Out Valid	—	25	—	19	ns
146	Clk Low Transition To Data-Out Invalid (see Note 3)	5	—	4	—	ns
147	Data-In Valid To Clk High Transition (Setup)	0	—	0	—	ns
148	Clk High Transition To Data-In Invalid (Hold)	12	—	12	—	ns
149	Clk Low To Address Invalid (see Note 3)	3	—	3	—	ns

Notes:

1. AC timing specifications which are referenced to a device input signal are measured in production with respect to the 50% point of the respective input signal's transition.
2. WS are wait state values specified in the BCR.
3. Clk low to data-out invalid (spec. 146) and Clk low to address invalid (spec. 149) indicate the time after which data/address are no longer guaranteed to be valid.



Sync. Bus Figure 1. DSP56001 Synchronous Bus Timing

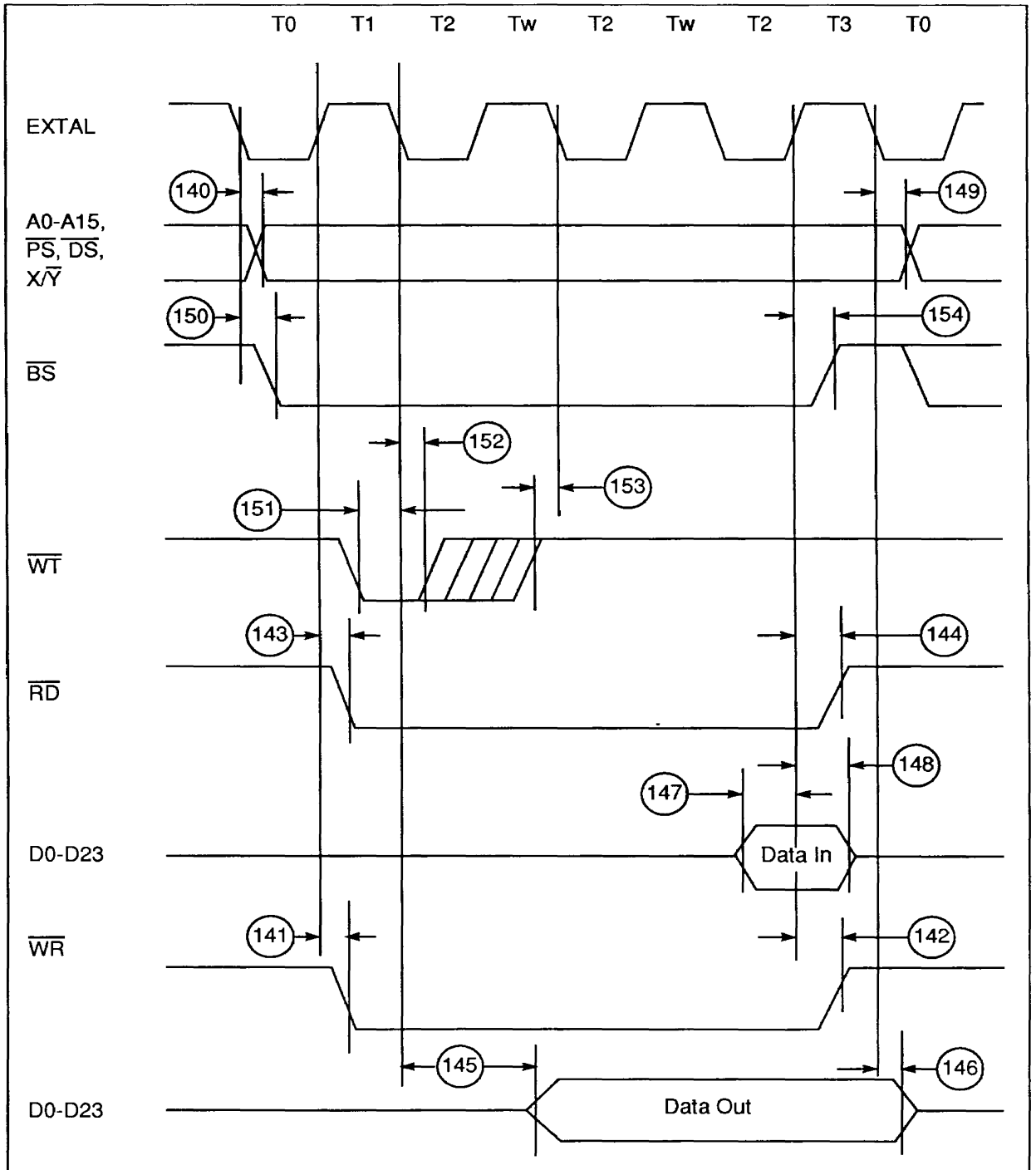
Note: During Read-Modify-Write Instructions, the address lines do not change states.

AC Electrical Characteristics - Bus Strobe / Wait Timing

Num.	Characteristics	20.5 MHz Min	20.5 MHz Max	27 MHz Min	27 MHz Max	Unit
150	Clk Low Transition To $\overline{BS}$ Assertion	4	24	3	19	ns
151	$\overline{WT}$ Assertion To Clk Low Transition (setup time)	4	—	3	—	ns
152	Clk Low Transition To $\overline{WT}$ Deassertion For Minimum Timing	14	cyc-8	11	cyc-6	ns
153	$\overline{WT}$ Deassertion To Clk Low Transition For Maximum Timing (2 wait states)	8	—	6	—	ns
154	Clk High Transition To $\overline{BS}$ Deassertion	5	26	4	20	ns
155	$\overline{BS}$ Assertion To Address Valid	-2	10	-2	8	ns
156	$\overline{BS}$ Assertion To $\overline{WT}$ Assertion (See Note 2)	0	cyc-15	0	cyc-11	ns
157	$\overline{BS}$ Assertion To $\overline{WT}$ Deassertion (See Note 2 and Note 4) WS $\leq$ 2 WS $\geq$ 2	cyc (WS-1)•cyc	2•cyc-15 WS•cyc-15	cyc (WS-1)•cyc	2•cyc-11 WS•cyc-11	ns
158	$\overline{WT}$ Deassertion To $\overline{BS}$ Deassertion	cyc+tcl	2•cyc+tcl+23	cyc+tcl	2•cyc+tcl+17	ns
159	Minimum $\overline{BS}$ Deassertion Width For Consecutive External Accesses	tch-7	—	tch-6	—	ns
160	$\overline{BS}$ Deassertion To Address Invalid (see Note 3)	tch-10	—	tch-8	—	ns
161	Data-In Valid To $\overline{RD}$ Deassertion (Set Up)	16	—	12	—	ns

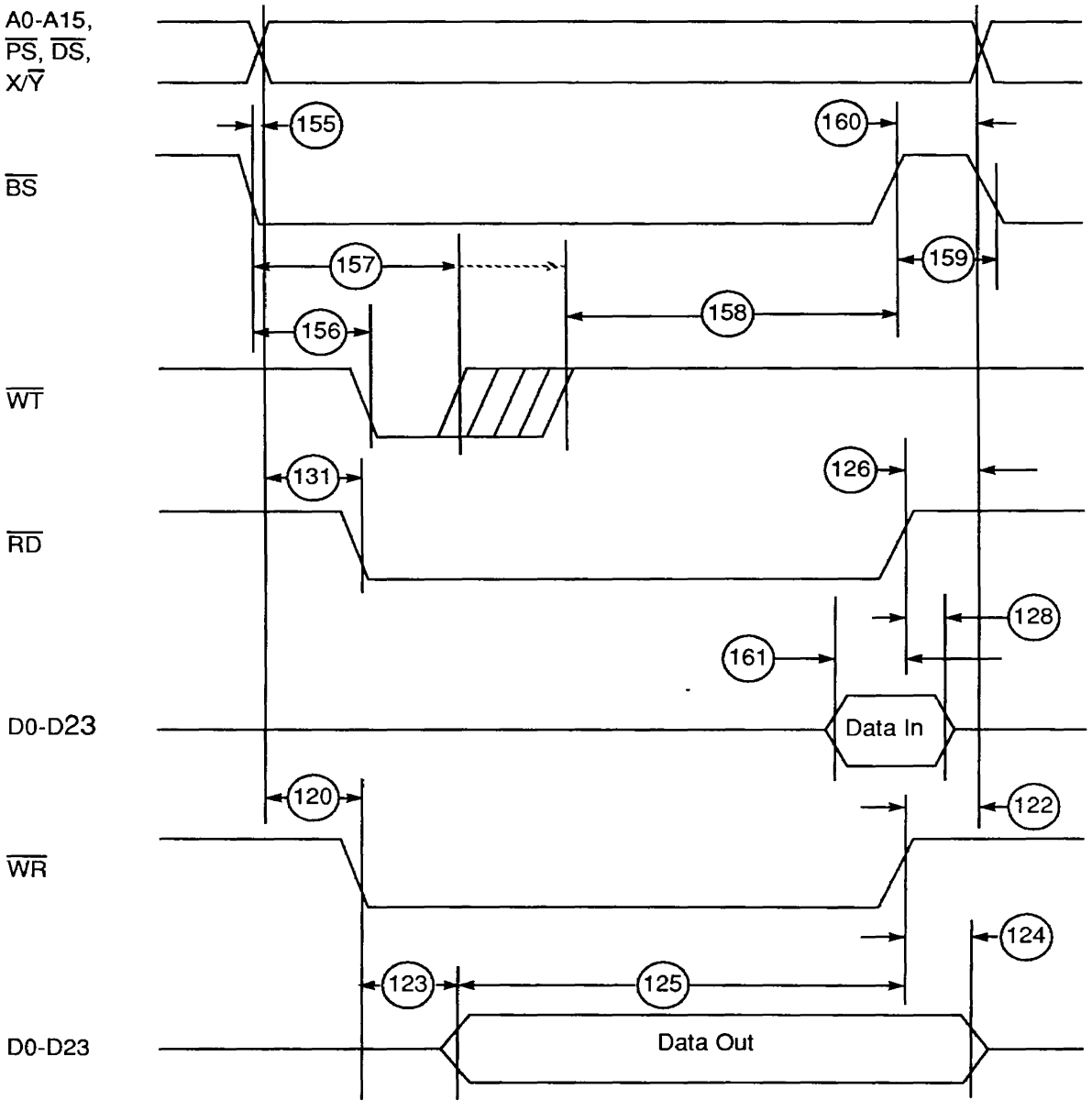
Note:

1. AC timing specifications which are referenced to a device input signal are measured in production with respect to the 50% point of the respective input signal's transition.
2. If wait states are also inserted using the BCR and if the number of wait states is greater than 2, then specification numbers 156 and 157 can be increased accordingly.
3. $\overline{BS}$ deassertion to address invalid indicates the time after which the address are no longer guaranteed to be valid.
4. The minimum number of wait states when using $\overline{BS}/\overline{WT}$ is two (2).
5. For read-modify-write instructions, the address lines will not change states between the read and the write cycle. However, $\overline{BS}$ will deassert before asserting again for the write cycle. If wait states are desired for each of the read and write cycle, the $\overline{WT}$ pin must be asserted once for each cycle.



Bus Arbitration Figure 1. DSP56001 Synchronous BS / WT Timings

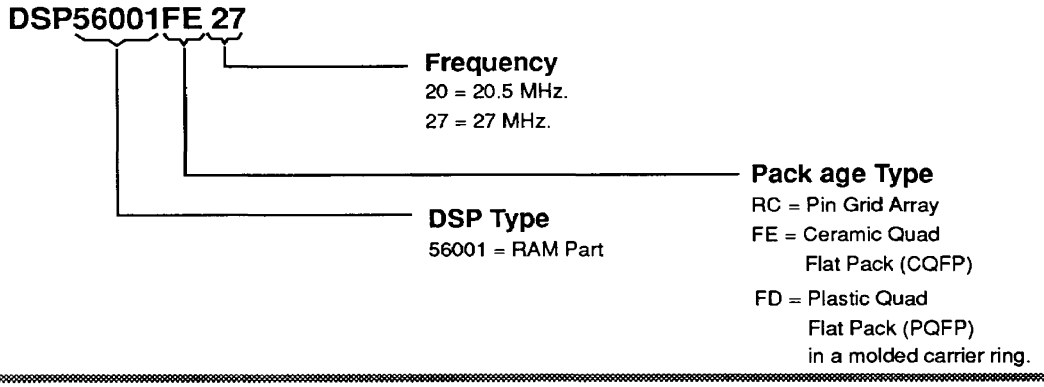
Note: During Read-Modify-Write Instructions, the address lines do not change state. However, BS will deassert before asserting again for the write cycle.



Bus Arbitration Figure 2. DSP56001 Asynchronous BS / WT Timings

Note: During Read-Modify-Write Instructions, the address lines will not change states. However, BS will deassert before asserting again for the write cycle.

ORDERING INFORMATION



DSP56001 SOCKET INFORMATION

PGA

Supplier	Telephone	Socket Type	Part Number	Comment
Advanced Interconnections				
	(401) 823-5200	Standard 88 Pin	4CS088-01TG[2]	Includes Cutout in Center
AMP				
	(717) 564-0100	Standard 88 Pin	1-916223-3	Low Insertion Force
			1-55283-9	ZIF Production
		Standard 128 Pin	1-55383-4	ZIF Burn-In and Test
Robinson Nugent				
	(812) 945-0211	Custom Pinout	PGA-088CM3P-S-TG[3]	
			PGA-088CHP3-SL-TG[3]	High Temp, Longer Leads
Samtec				
	(812) 944-6733	Standard 120 Pin	MVAS-120-ZSTT-13[1]	Includes Cutout in Center
		Custom 88 Pin	CPAS-88-ZSTT-13BF[1]	No Cutout

NOTES:

1. Please specify wirewrap and plating options. The part numbers shown specify low profile solder tail pins having a tin contact and tin shell.
2. Please specify wirewrap and plating options. The part number shown specifies gold contact and tin shell.
3. Cutout in the center, unused holes are plugged, solder tail.

CQFP

Supplier	Telephone	Socket Type	Part Number	Comment
AMP				
	(717)564-0100	—	822054-2[1]	Converts CQFP to fit AMP's 132 position PQFP "Micro-Pitch Socket".

NOTES:

1. This part is not a socket. It is a converter that allows a CQFP part to be used in the PQFP socket described below.

PQFP

Supplier	Telephone	Socket Type	Part Number	Comment
AMP				
	(717)564-0100	132 Pin	821949-5[1]	Housing Sub-Assembly and Cover for 132 position PQFP "Micro-Pitch Socket".
			821942-1[1]	

NOTES:

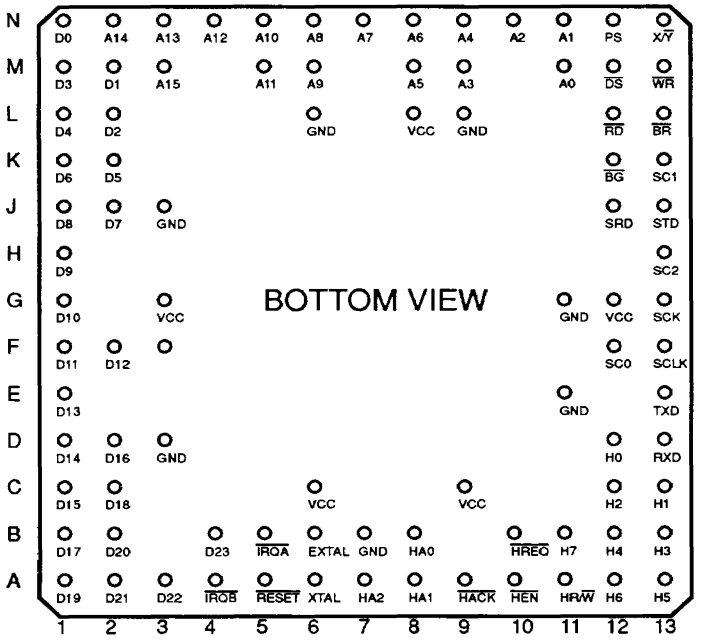
1. One housing sub-assembly and one cover are required for each socket.

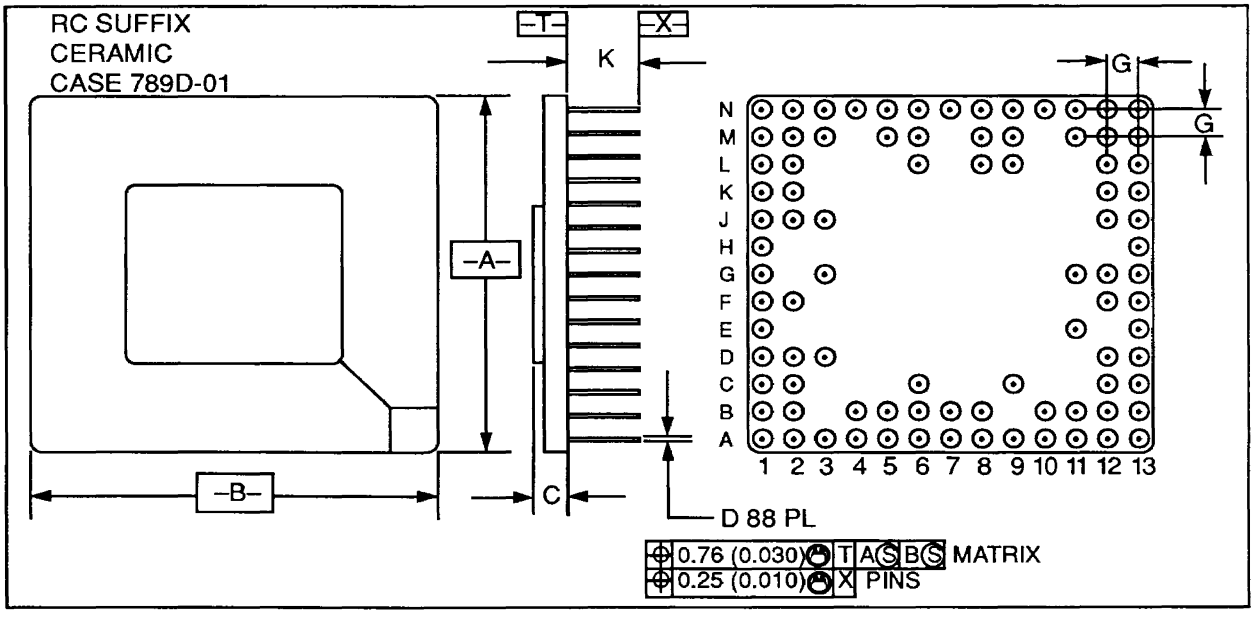
SLAM

Supplier	Telephone	Socket Type	Part Number	Comment
Robinson Nugent				
	(812) 945-0211	Custom 100 Pin	LCC-100P-DSP	100 Pin Through-hole Socket

Pin Grid Array Mechanical Specification

PIN ASSIGNMENT





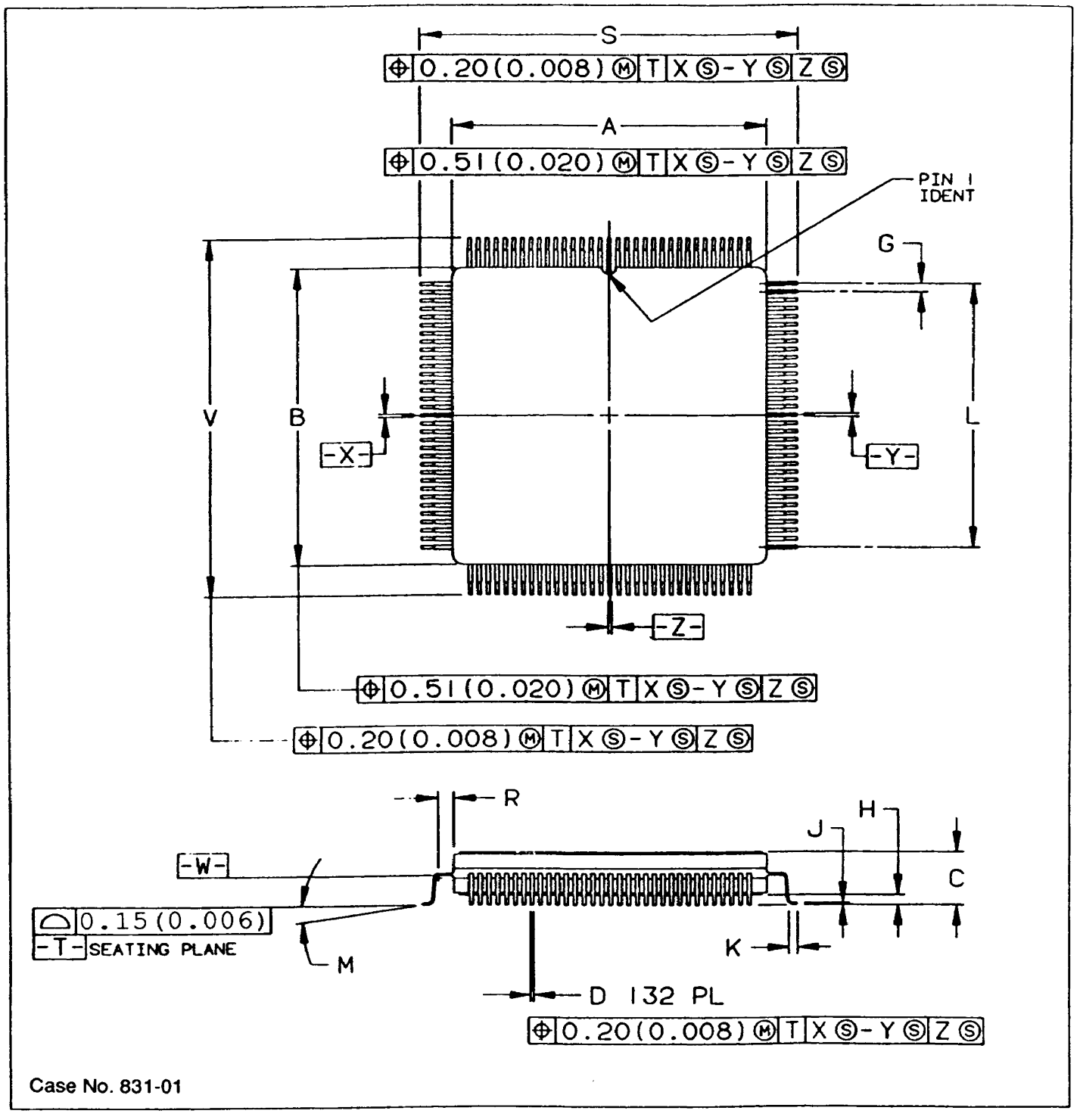
DIM	MILLIMETERS MIN	MAX	INCHES MIN	MAX
A	34.04	35.05	1.340	1.380
B	34.04	35.05	1.340	1.380
C	2.16	3.04	0.085	0.120
D	0.44	0.55	0.017	0.022
G	2.54 BSC		0.100 BSC	
K	4.20	5.08	0.165	0.200

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M. 1982.
2. CONTOLLING DIMENSION: INCH.

Mechanical Specification Table 1. CQFP and PQFP Pin Out

PIN #	FUNCTION	PIN #	FUNCTION	PIN #	FUNCTION	PIN #	FUNCTION
17	NO CONNECT	116	NO CONNECT	83	NO CONNECT	50	NO CONNECT
16	H4	115	D20	82	D1	49	$\overline{DS}$
15	H5	114	D19	81	D0	48	X/$\overline{Y}$
14	H6	113	D18	80	A15	47	$\overline{RD}$
13	PERIPHERAL VCC	112	DATA BUS GND	79	A14	46	$\overline{WR}$
12	PERIPHERAL VCC	111	DATA BUS GND	78	NO CONNECT	45	$\overline{BR}$
11	H7	110	NO CONNECT	77	A13	44	NO CONNECT
10	$\overline{HREQ}$	109	D17	76	A12	43	$\overline{BG}$
9	HR/$\overline{W}$	108	D16	75	A11	42	SRD
8	$\overline{HEN}$	107	NO CONNECT	74	ADDRESS BUS GND	41	NO CONNECT
7	NO CONNECT	106	D15	73	ADDRESS BUS GND	40	SC1
6	$\overline{HACK}$	105	D14	72	NO CONNECT	39	STD
5	HA0	104	D13	71	A10	38	NO CONNECT
4	NO CONNECT	103	NO CONNECT	70	A9	37	SC2
3	NO CONNECT	102	D12	69	NO CONNECT	36	INTERNAL LOGIC VCC
2	HA1	101	DATA BUS VCC	68	A8	35	INTERNAL LOGIC VCC
1	HA2	100	DATA BUS VCC	67	A7	34	INTERNAL LOGIC GND
132	NO CONNECT	99	D11	66	NO CONNECT	33	INTERNAL LOGIC GND
131	INTERNAL LOGIC GND	98	NO CONNECT	65	A6	32	SCK
130	INTERNAL LOGIC GND	97	D10	64	ADDRESS BUS VCC	31	SC0
129	INTERNAL LOGIC VCC	96	D9	63	ADDRESS BUS VCC	30	NO CONNECT
128	INTERNAL LOGIC VCC	95	NO CONNECT	62	NO CONNECT	29	SCLK
127	EXTAL	94	D8	61	A5	28	TXD
126	XTAL	93	D7	60	A4	27	RXD
125	NO CONNECT	92	D6	59	NO CONNECT	26	NO CONNECT
124	$\overline{RESET}$	91	DATA BUS GND	58	A3	25	H0
123	MODA/$\overline{IRQA}$	90	DATA BUS GND	57	A2	24	PERIPHERAL GND
122	NO CONNECT	89	NO CONNECT	56	ADDRESS BUS GND	23	PERIPHERAL GND
121	NMI/MODB/$\overline{IRQB}$	88	D5	55	ADDRESS BUS GND	22	H1
120	D23	87	D4	54	A1	21	NO CONNECT
119	D22	86	D3	53	A0	20	H2
118	D21	85	D2	52	$\overline{PS}$	19	H3
117	NO CONNECT	84	NO CONNECT	51	NO CONNECT	18	NO CONNECT

Note: Do not connect to "NO CONNECT" pins.
"NO CONNECT" pins are reserved for future enhancements.



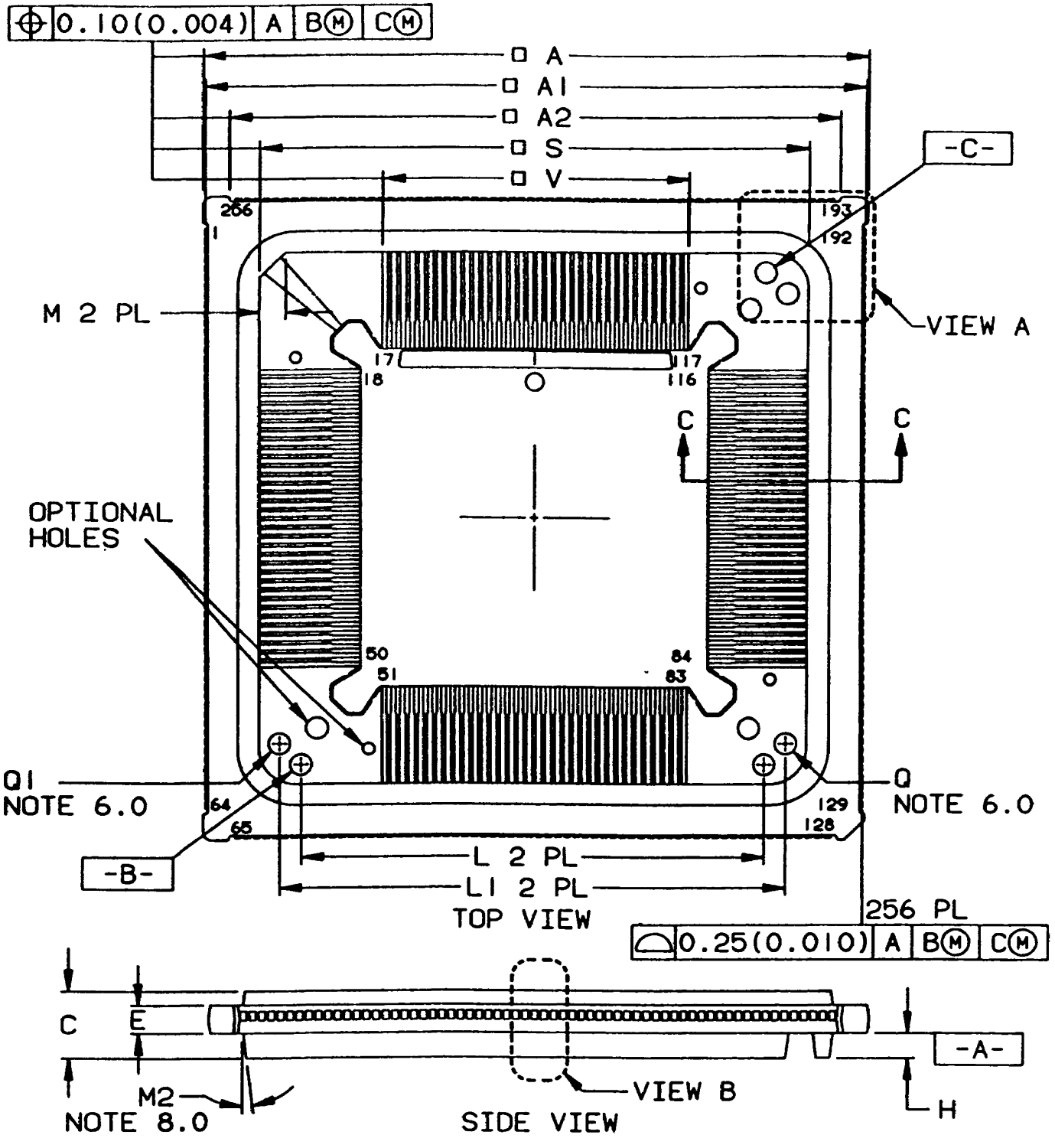
Mechanical Specification Figure 1. Ceramic Quad Flat Pack

NOTES

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIM A AND B DEFINE MAXIMUM CERAMIC BODY DIMENSIONS INCLUDING GLASS PROTRUSION AND MISMATCH OF CERAMIC BODY TOP AND BOTTOM.
4. DATUM PLANE -W- IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS X-Y AND Z TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM -W-.
6. DIM S AND V TO BE DETERMINED AT SEATING PLANE, DATUM -T-.
7. DIM A AND B TO BE DETERMINED AT DATUM PLANE -W-.

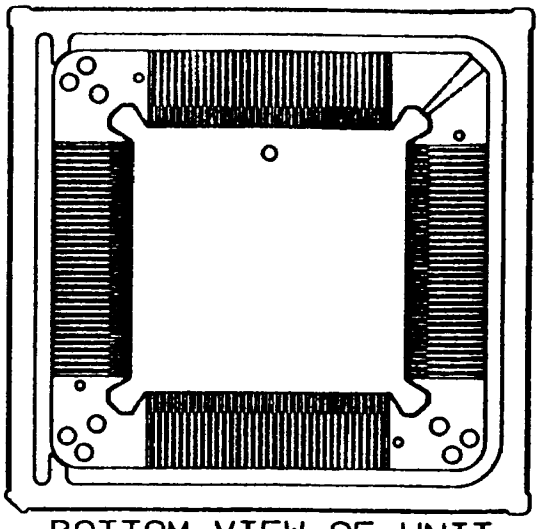
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	21.85	22.86	0.860	0.900
B	21.85	22.86	0.860	0.900
C	3.94	4.31	0.155	0.170
D	0.204	0.292	0.0080	0.0115
G	0.64 BSC		0.025 BSC	
H	0.64	0.88	0.025	0.035
J	0.13	0.20	0.005	0.008
K	0.51	0.76	0.020	0.030
L	20.32 REF		0.800 REF	
M	0°	8°	0°	8°
R	0.64	---	0.025	---
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

Mechanical Specification Figure 1. Ceramic Quad Flat Pack (Continued)

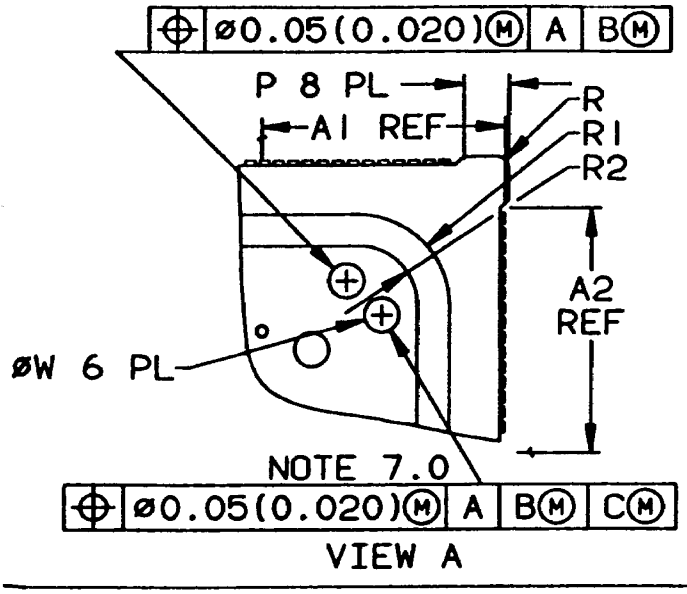


Case No. 878-01

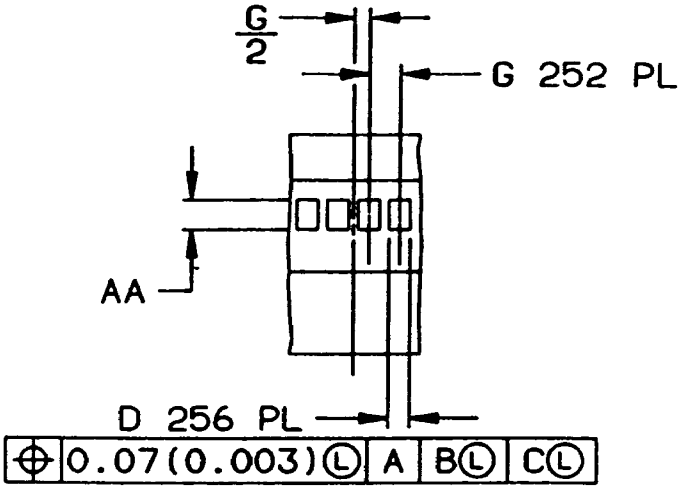
Mechanical Specification Figure 2. Plastic Quad Flat Pack



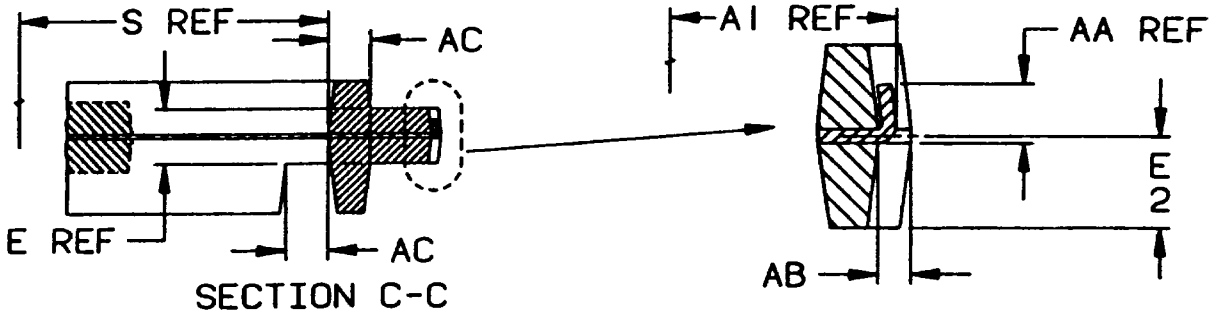
BOTTOM VIEW OF UNIT
IN MOLDED CARRIER RING



VIEW A

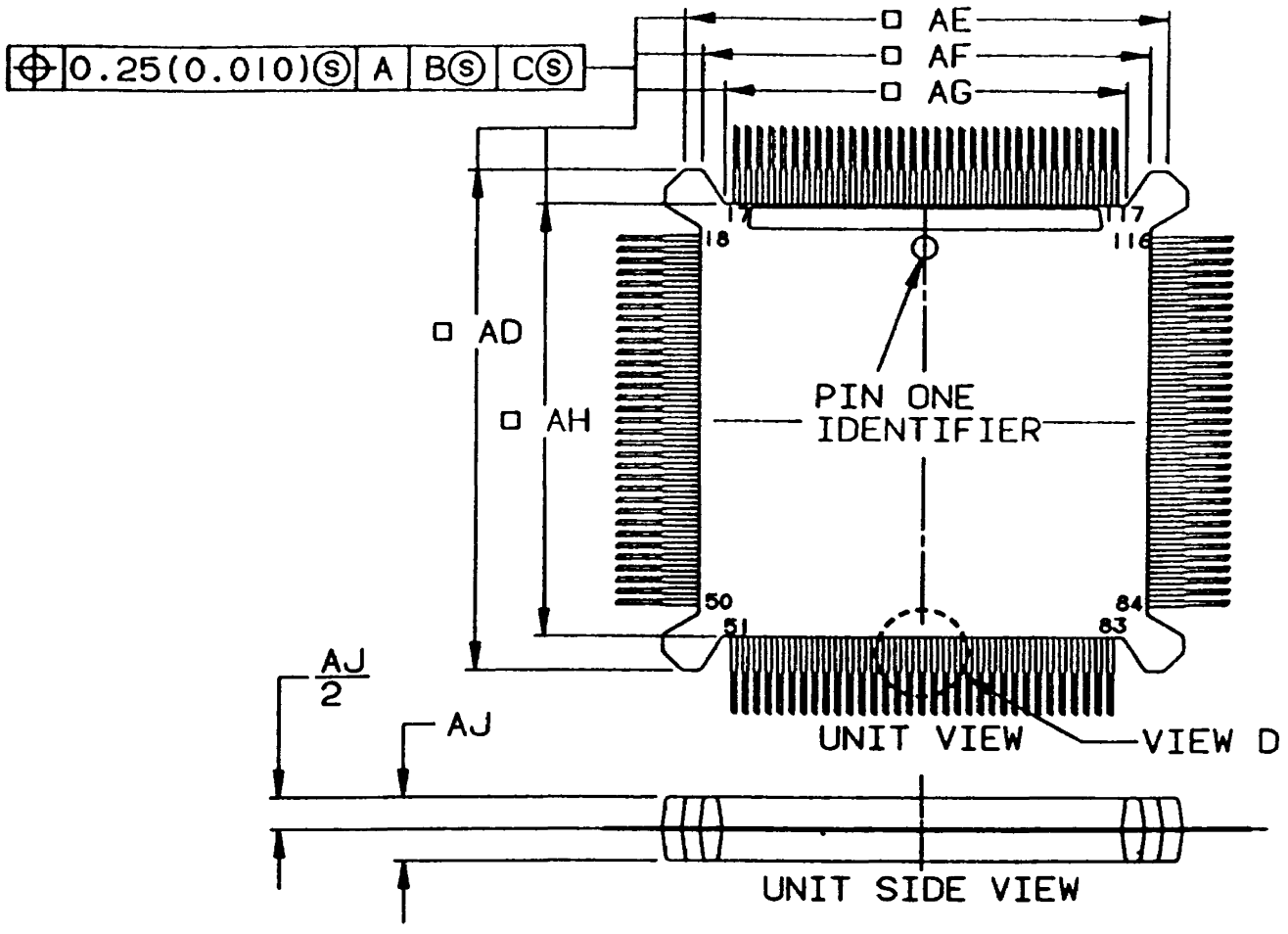


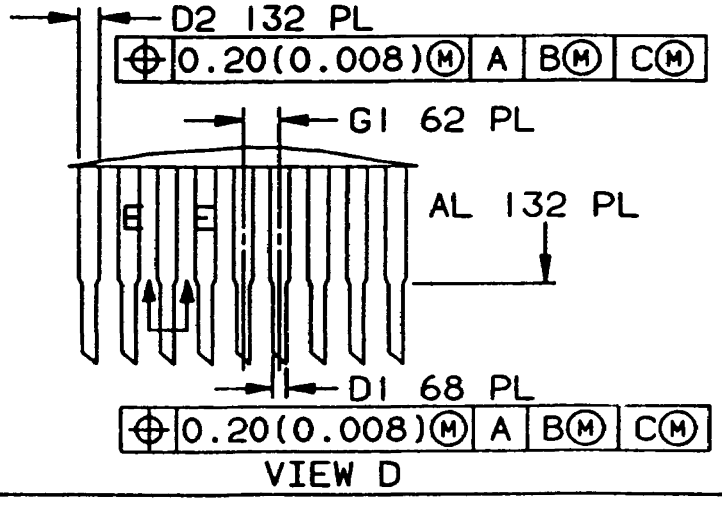
VIEW B

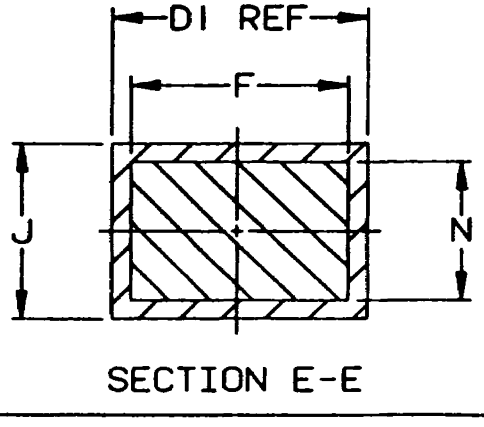


SECTION C-C

Mechanical Specification Figure 2. Plastic Quad Flat Pack (Continued)







Mechanical Specification Figure 2. Plastic Quad Flat Pack (Continued)

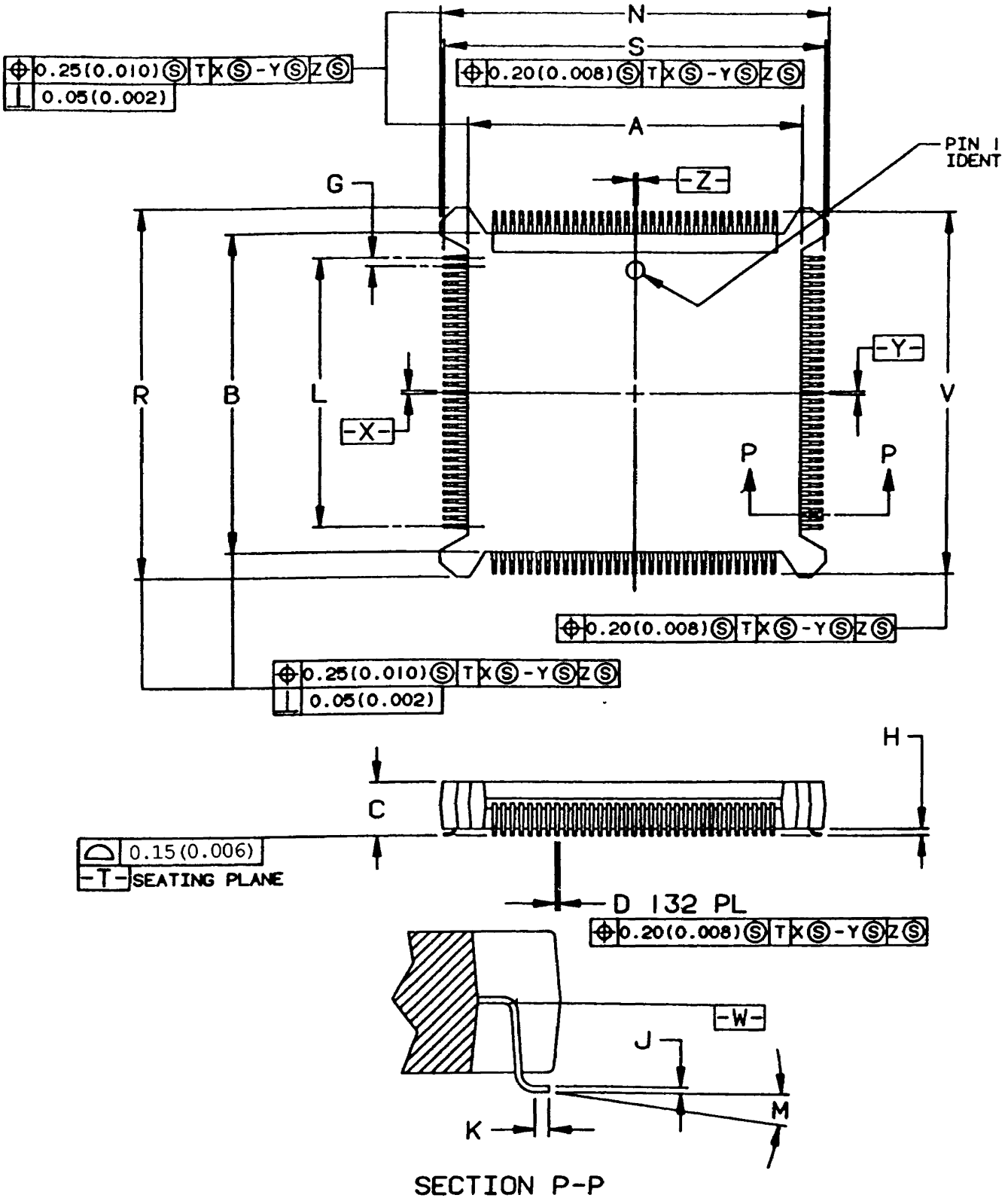
NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. A, AD, AE, AF AND AH DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS 0.20(0.008) PER SIDE. FOR AE DIMENSION IS 0.18(0.007).
4. A, S1 AND AH DIMENSIONS INCLUDE MOLD MISMATCH, AND ARE MEASURED AT THE PARTING LINE.
5. UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE SYMMETRICAL ABOUT CENTERLINES.
6. B AND C DATUM HOLES ARE TO BE USED FOR TRIM, FORM AND EXCISE OF THE MOLDED PACKAGE ONLY. HOLES Q1 AND Q2 ARE TO BE USED FOR ELECTRICAL TESTING ONLY.
7. NON-DATUM HOLES ONLY.
8. APPLIES TO RING AND PACKAGE FEATURES.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	45.87	46.13	1.806	1.816
A1	45.70 BSC		1.799 BSC	
A2	41.37	41.63	1.629	1.639
C	4.70	4.90	0.185	0.193
D	0.40	0.50	0.016	0.020
D1	0.21	0.30	0.008	0.012
D2	0.31	0.40	0.012	0.016
E	1.90	2.10	0.075	0.083
F	0.19	0.27	0.007	0.011
G	0.65 BSC		0.026 BSC	
G1	0.635 BSC		0.025 BSC	
H	1.70	1.90	0.067	0.075
J	0.16	0.20	0.006	0.008
L	32.20 BSC		1.268 BSC	
L1	35.20 BSC		1.386 BSC	
M	1.30	2.30	0.051	0.091

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
M2	6°	8°	6°	8°
N	0.145	0.16	0.0057	0.0063
P	1.77	2.03	0.070	0.080
R	0.40	0.60	0.016	0.024
R1	3.50	4.50	0.138	0.177
R2	2.00	3.00	0.079	0.118
S	37.87	38.13	1.491	1.501
V	21.21	21.31	0.835	0.839
W	1.45	1.55	0.057	0.061
AA	0.45	0.85	0.018	0.033
AB	0.30	0.60	0.012	0.024
AC	1.37	1.63	0.054	0.064
AD	27.88	28.01	1.098	1.103
AE	25.79	25.93	1.015	1.021
AF	23.91	24.05	0.941	0.947
AG	21.52		0.847	0.853
AH	24.06	24.20	0.947	0.953
AJ	3.46	3.66	0.136	0.144
AL	14.00	14.10	0.551	0.555

Mechanical Specification Figure 2. Plastic Quad Flat Pack (Continued)



Mechanical Specification Figure 3. Plastic Quad Flat Pack Without Ring — Reference Only

NOTES

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSIONS A, B, N, AND R DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION FOR DIMENSIONS A AND B IS 0.25(0.010). FOR DIMENSIONS N AND R IS 0.18(0.007).
4. DATUM PLANE -W- IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS X-Y AND Z TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM -W-.
6. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM -T-.
7. DIMENSIONS A, B, N, AND R TO BE DETERMINED AT DATUM PLANE -W-.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	24.06	24.20	0.947	0.953
B	24.06	24.20	0.947	0.953
C	4.07	4.57	0.160	0.180
D	0.21	0.30	0.008	0.012
G	0.64 BSC		0.025 BSC	
H	0.51	1.01	0.020	0.040
J	0.16	0.20	0.006	0.008
K	0.51	0.76	0.020	0.030
L	20.32 REF		0.800 REF	
M	0°	8°	0°	8°
N	27.88	28.01	1.097	1.103
R	27.88	28.01	1.097	1.103
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

Mechanical Specification Figure 3. Plastic Quad Flat Pack Without Ring — Reference Only (Continued)

APPENDIX A
BOOTSTRAP MODE — OPERATING MODE 1

The bootstrap feature of the DSP56001 consists of four special on-chip modules: the 512 words of PRAM, a 32-word bootstrap ROM, the bootstrap control logic, and the bootstrap firmware program.

BOOTSTRAP ROM

This 32-word on-chip ROM has been factory programmed to perform the actual bootstrap operation from the memory expansion port (Port A) or from the Host Interface. You have no access to the bootstrap ROM other than through the bootstrap process. Control logic will disable the bootstrap ROM during normal operations.

BOOTSTRAP CONTROL LOGIC

The bootstrap mode control logic is activated when the DSP56001 is placed in Operating Mode 1. The control logic maps the bootstrap ROM into program memory space as long as the DSP56001 remains in Operating Mode 1. The bootstrap firmware changes operating modes when the bootstrap load is completed. When the DSP56001 exits the reset state in Mode 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 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, or through the Host Interface.
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 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 DSP56001) to begin the bootstrap process as described above in steps 1-4. This technique allows the DSP56001 user to reboot the system (with a different program if desired).

BOOTSTRAP FIRMWARE PROGRAM

Bootstrap ROM contains the bootstrap firmware program that performs initial loading of the DSP56001 PRAM. The program is written in DSP56000/DSP56001 assembly language. It contains two separate methods of initializing the PRAM: loading from a byte-wide memory starting at location P:$C000 or loading through the Host Interface. The particular method used is selected by the level of program memory location $C000, bit 23. If location P:$C000, bit 23 is read as a one, the external bus version of the bootstrap program will be selected. Typically, a byte wide EPROM will be connected to the DSP56001 Address and Data Bus as shown in 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:$0000 mid byte
P:$C002	P:$0000 high byte
•	•
•	•
•	•
P:$C5FD	P:$01FF low byte
P:$C5FE	P:$01FF mid byte
P:$C5FF	P:$01FF high byte

If location P:$C000, bit 23 is read as a zero, the Host Interface version of the bootstrap program will be selected. Typically a host microprocessor will be connected to the DSP56001 Host Interface. The host microprocessor must write the Host Interface registers THX, TXM, and then TSL with the desired contents of PRAM from location P:$0000 up to P:$01FF. If less than 512 words are to be loaded, the host programmer can exit the bootstrap program and force the DSP56001 to begin executing at location P:$0000 by setting HF0=1 in the Host Interface during the bootstrap load. In most systems, the DSP56001 responds so fast that handshaking between the DSP56001 and the host is not necessary.

The bootstrap program is shown in flowchart form in Figure A-1 and in assembler listing format in Figure A-2.

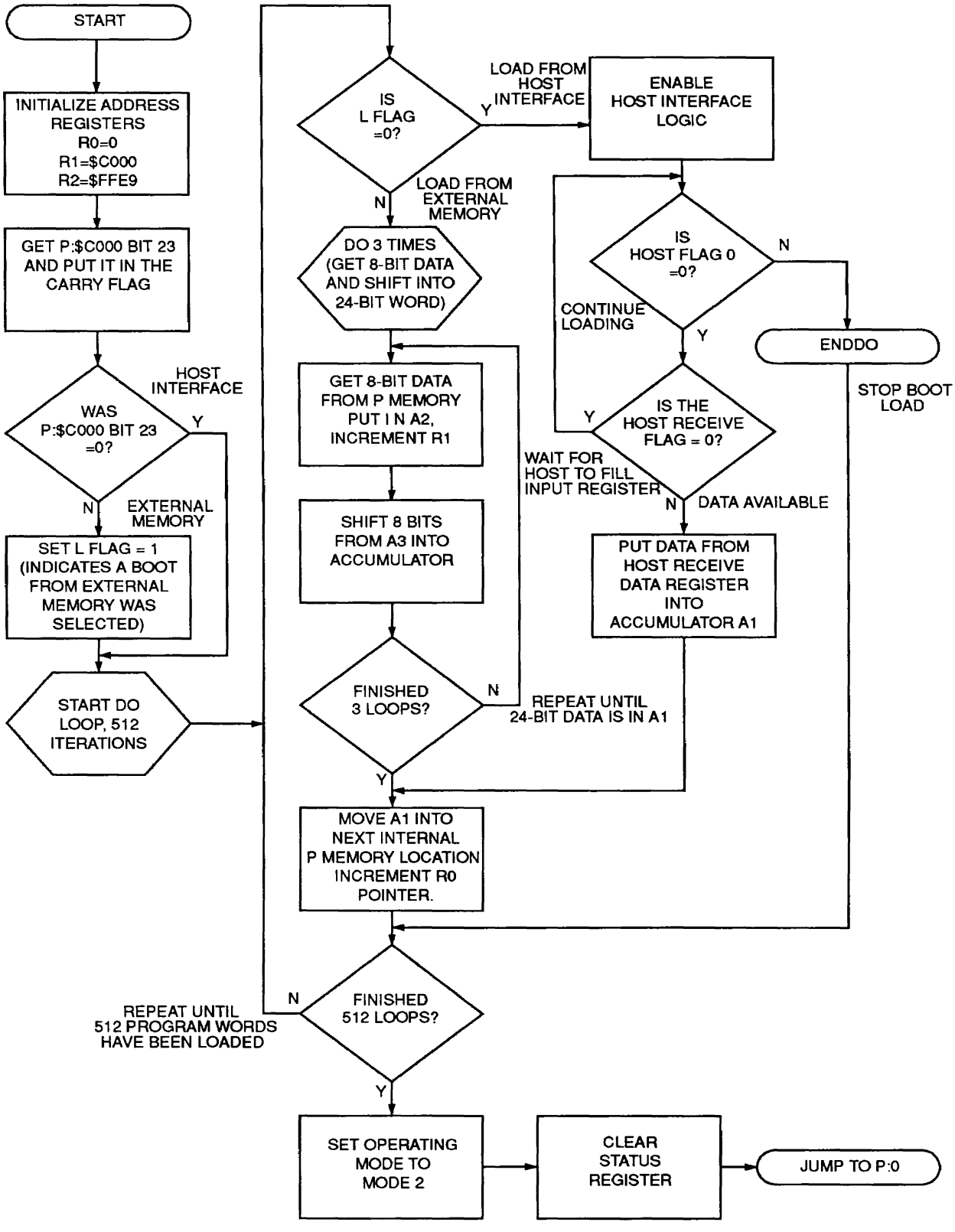


Figure A-1. Bootstrap Program Flowchart

```
1                                   PAGE 132,50,0,10
2                                   ; BOOTSTRAP SOURCE CODE FOR DSP56001 - (C) Copyright 1986 Motorola Inc.
4                                   ; Host algorithm / AND / external bus method
6                                   ; This is the Bootstrap source code contained in the DSP56001 32 word boot ROM.
7                                   ; This program can load the internal program memory from one of two external sources.
8                                   ; The program reads P:$C000 bit 23 to decide which external source to access. If
9                                   ; P:$C000 bit 23 = 0 then it loads internal PRAM from H0-H7, using the Host Interface
10                                  ; logic. If P:$C000 bit 23 = 1 then it loads from 1,536 consecutive byte-wide P:
11                                  ; memory locations (starting at P:$C000).
13  0000C000                        BOOT    EQU     $C000                   ; The location in P: memory
14                                                                          ; where the external byte-wide
15                                                                          ; EPROM is expected to be mapped.
16
17  p:0000                          ORG     PL:$0                           ; Bootstrap code starts at P:$0
18
19  P:0000      62F400      START   MOVE            #$FFE9,R2               ; R2 = address of the Host
                00FFE9
20                                                                          ; Interface status register.
21  P:0002      61F400              MOVE            #BOOT,R1                ; R1 = starting P: address of
                00C000
22                                                                          ; external bootstrap byte-wide ROM.
23  P:0004      300000              MOVE            #0,R0                   ; R0 = starting P: address of
24                                                                          ; internal memory where program
25                                                                          ; will begin loading.
26
27  P:0005      07E18C              MOVE            P:(R1),A1               ; Get the data at P:$C000
28  P:0006      200037              ROL     A                               ; Shift bit 23 into the Carry flag
29  P:0007      0E0009              JCC     <INLOOP                 -       ; Perform load from Host Interface
30                                                                          ; if carry is zero.
31
32                                  ; IMPORTANT NOTE: This routine assumes that the L bit has been cleared before entering
33                                  ; this program and that M0 and M1 have been preloaded with $FFFF (linear addressing).
34                                  ; This would be the case after a reset. If this program is entered by changing the OMR
```

Figure A-2. Assembler Listing for Bootstrap Program (Sheet 1 of 3)

```
35                          ; to bootstrap operating mode, make certain that the L bit is cleared and registers M0
36                          ; and M1have been set to $FFFF. Also, make sure the BCR is set to $xxFx since
37                          ; EPROMS are slow and BCR is set to $FFFF after a reset. If the L bit was set before
38                          ; changing modes, the program will load from external program memory.
39
40  P:0008    0040F9                ORI     #$40,CCR            ; Set the L bit to indicate
41                                                                ; that the bootstrap program
42                                                                ; is being loaded from the
43                                                                ; external P: space.
44
45                          ; The first routine will load 1,536 bytes from the external P: memory space beginning
46                          ; at P:$C000 (bits 7-0). These will be packed into 512 24-bit words and stored in
47                          ; contiguous internal PRAM memory locations starting at P:$0.
48
49                          ; The shifter moves the 8-bit input data from register A2 into register A1 eight bits
50                          ; at a time. After assembling one 24-bit word (this takes three loops) it stores the
51                          ; result in internal PRAM and continues until internal PRAM is filled. Note that the
52                          ; first routine loads data starting with the least significant byte of P:$0 first.
53
54                          ; The second routine loads the internal PRAM using the Host Interface logic.
55                          ; If the host only wants to load a portion of the PRAM, the Host Interface bootstrap
56                          ; load program can be aborted and execution of the loaded program started, by setting
57                          ; the Host Flag (HF0) = 1 at any time during the load from the Host Processor.
58
59  P:0009    060082  INLOOP  DO      #512,_LOOP1         ; Load 512 instruction words.
              00001B
60
61                          ; This is the context switch
62
63  P:000B    0E6012                JLC     <_HOSTLD            ; Load from the Host Interface
64                                                                ; if the Limit flag is clear.
65
66                          ; This is the first routine. It loads from external P: memory.
67
68  P:000C    060380                DO      #3,_LOOP2           ; Each instruction has 3 bytes.
              000010
```

Figure A-2. Assembler Listing for Bootstrap Program (Sheet 2 of 3)

```
69  P:000E    07D98A              MOVE            P:(R1)+,A2      ; Get the 8 LSB from external
70                                                                ; P: memory.
71  P:000F    0608A0              REP     #8                      ; Shift 8 bit data into A1
72  P:0010    200022              ASR     A
73                      _LOOP2                                    ; Get another byte
74  P:0011    0C001B              JMP     < _STORE                ; then put the word in PRAM.
75
76                      ; This is the second routine. It loads from the Host Interface pins.
77
78  P:0012    0AA020    _HOSTLD   BSET    #0,X:$FFE0              ; Configure Port B as Host Interface
79  P:0013    0AA983    _LBLA     JCLR    #3,X:$FFE9, _LBLB       ; If HF0=1, stop loading data.
              000017
80  P:0015    00008C              ENDDO                           ; Must terminate the DO loop
81  P:0016    0C001C              JMP     <_BOOTEND
82
83  P:0017    0A6280    _LBLB     JCLR    #0,X:(R2), _LBLA        ; Wait for HRDF to go high
              000013
84                                                                ; (meaning 24-bit data is present)
85  P:0019    54F000              MOVE            X:$FFEB,A1      ; Put 24-bit host data in A1
              00FFEB
86
87  P:001B    07588C    _STORE    MOVE            A1,P:(R0)+      ; Store 24-bit result in PRAM.
88
89                      _LOOP1                                    ; and return for another 24-bit word
90
91                      ; This is the exit handler that returns execution to normal expanded mode
92                      ; and jumps to the RESET location.
93
94  P:001C    0502BA    _BOOTEND  MOVEC           #2,0MR          ; Set the operating mode to 2
95                                                                ; (and trigger an exit from
96                                                                ; bootstrap mode).

97  P:001D    0000B9              ANDI    #$0,CCR                 ; Clear SR as if RESET and
98                                                                ; introduce delay needed for
99                                                                ; Op. Mode change.
100 P:001E    0C0000              JMP     <$0                     ; Start fetching from PRAM P:$0000
101
```


0 Errors
0 Warnings

Figure A-2. Assembler Listing for Bootstrap Program (Sheet 3 of 3)

APPENDIX B
APPLICATION EXAMPLES

The lowest cost DSP56001 based system is shown in Figure B-1. It uses no run time external memory and requires only two chips, the DSP56001 and a low cost EPROM. The EPROM read access time should be less than 780 nanoseconds when the DSP56001 is operating at a clock rate of 20.5 MHz.

A system with external data RAM memory requires no glue logic to select the external EPROM from bootstrap mode. $\overline{PS}$ is used to enable the EPROM and $\overline{DS}$ is used to enable the high speed data memories as shown in Figure B-2.

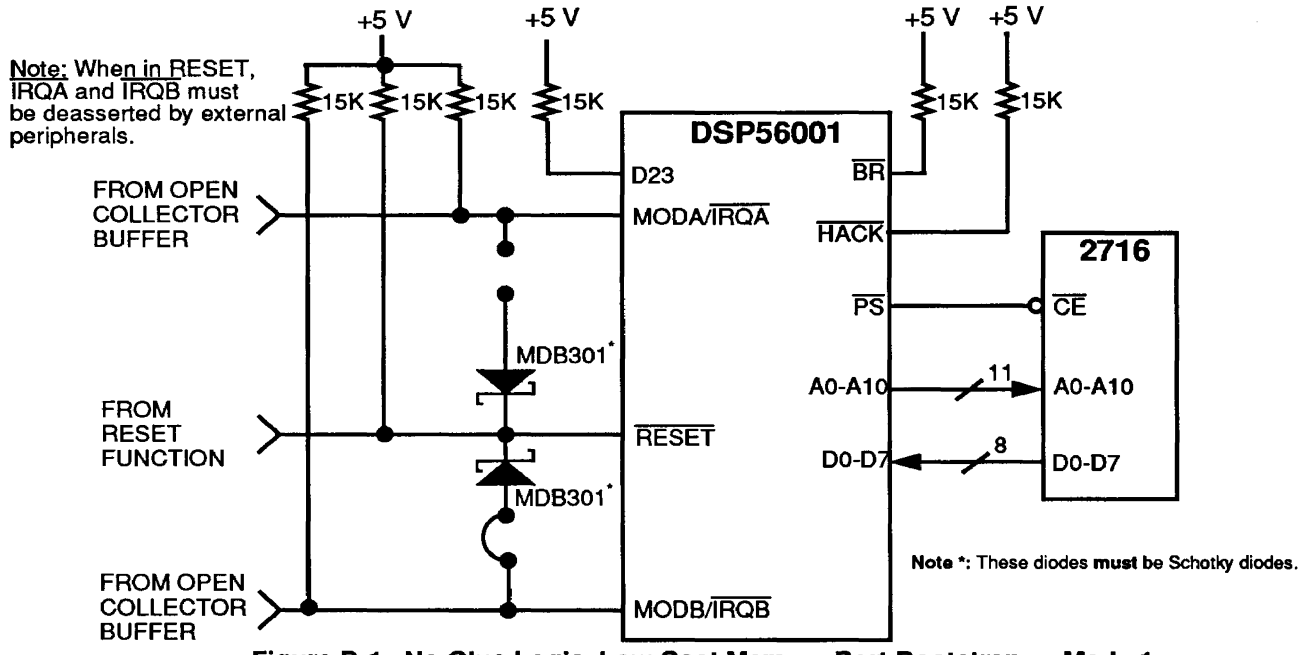


Figure B-1. No Glue Logic, Low Cost Memory Port Bootstrap — Mode 1

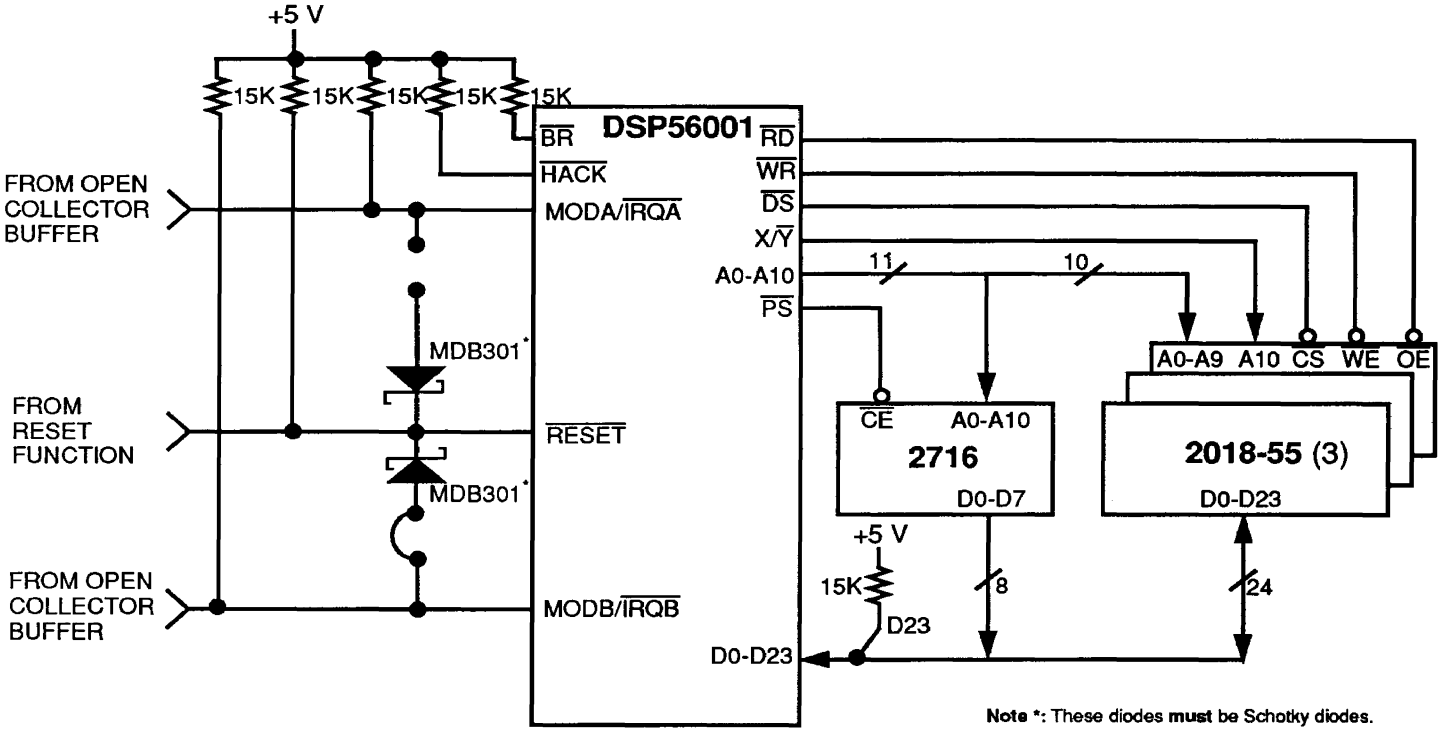


Figure B-2. Port A Bootstrap with External Data RAM — Mode 1

Figure B-3 shows the DSP56001 bootstrapping via the Host Port from an MC68000.

Systems with external program memory can load the on-chip PRAM without using the bootstrap mode. In Figure B-4, the DSP56001 is operated in mode 2 with external program memory at location $E000. The programmer can overlay the high speed on-chip PRAM with DSP algorithms by using the MOVEM instruction.

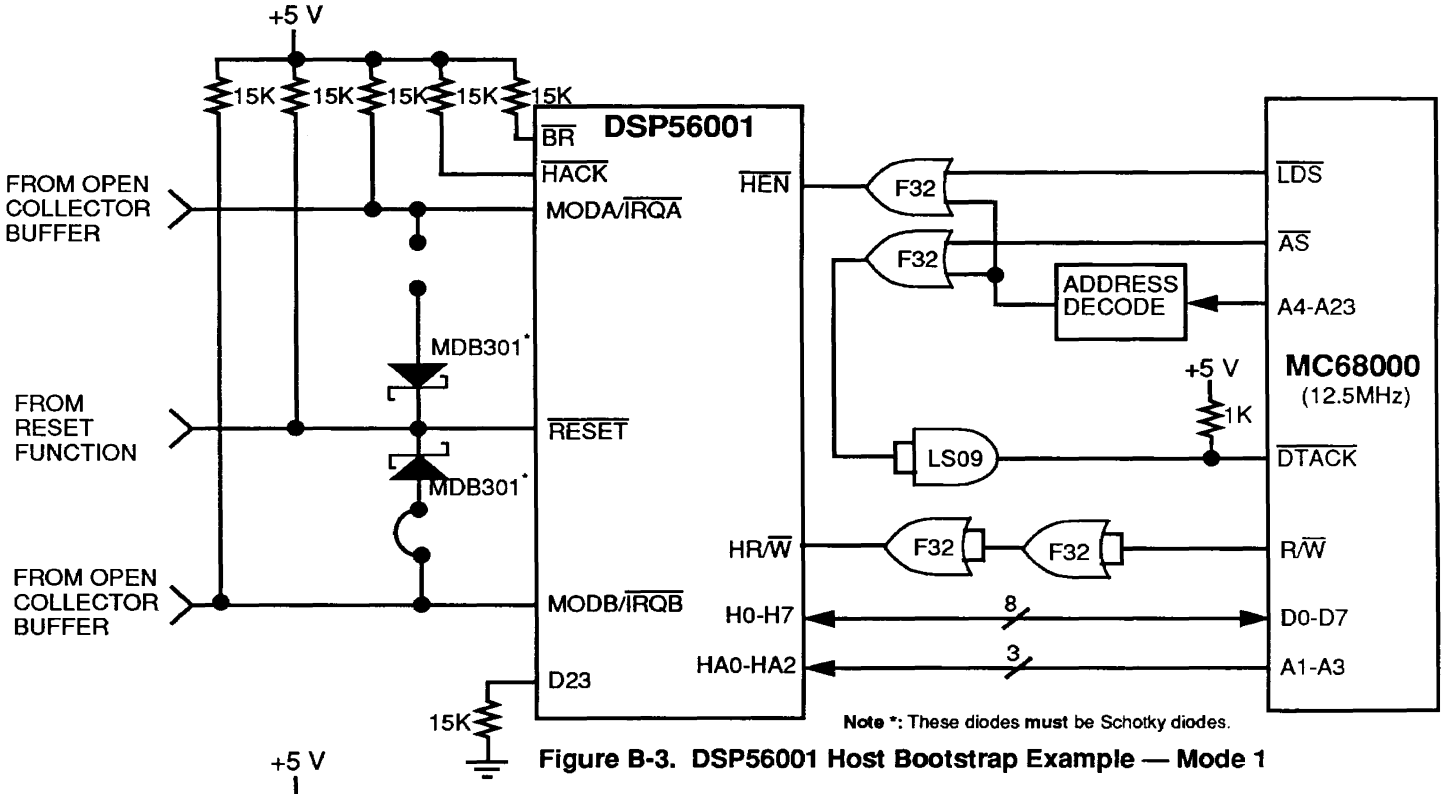


Figure B-3. DSP56001 Host Bootstrap Example — Mode 1

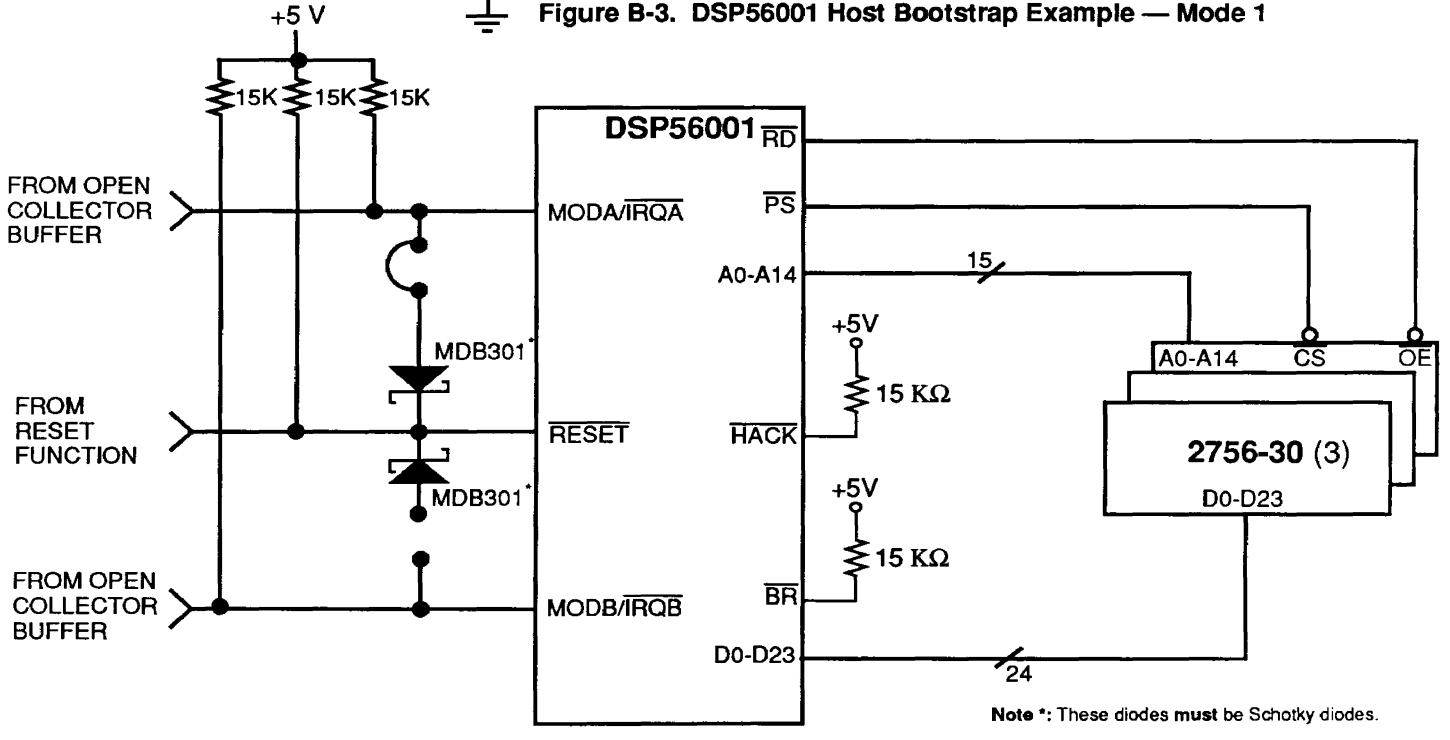


Figure B-4. 32K Words of External Program ROM — Mode 2

Figure B-5 shows an alternative clock oscillator circuit used in the Graphic Equalizer application note (APR2). The 330Ω resistor provides additional current limiting in the crystal. Figure B-6 shows a circuit which waits until Vcc on the DSP is at least 4.5 V before initiating a 3.75 ms minimum (150,000T) oscillator stabilization delay required for the on-chip oscillator (only 50T is required for an external oscillator). This insures that the DSP is operational and stable before releasing the reset signal.

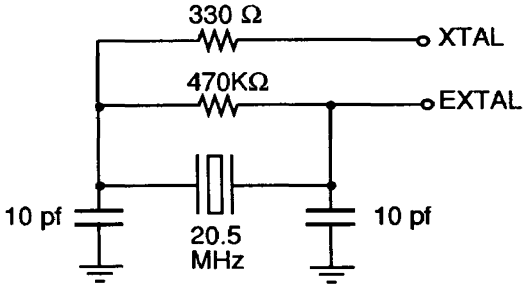
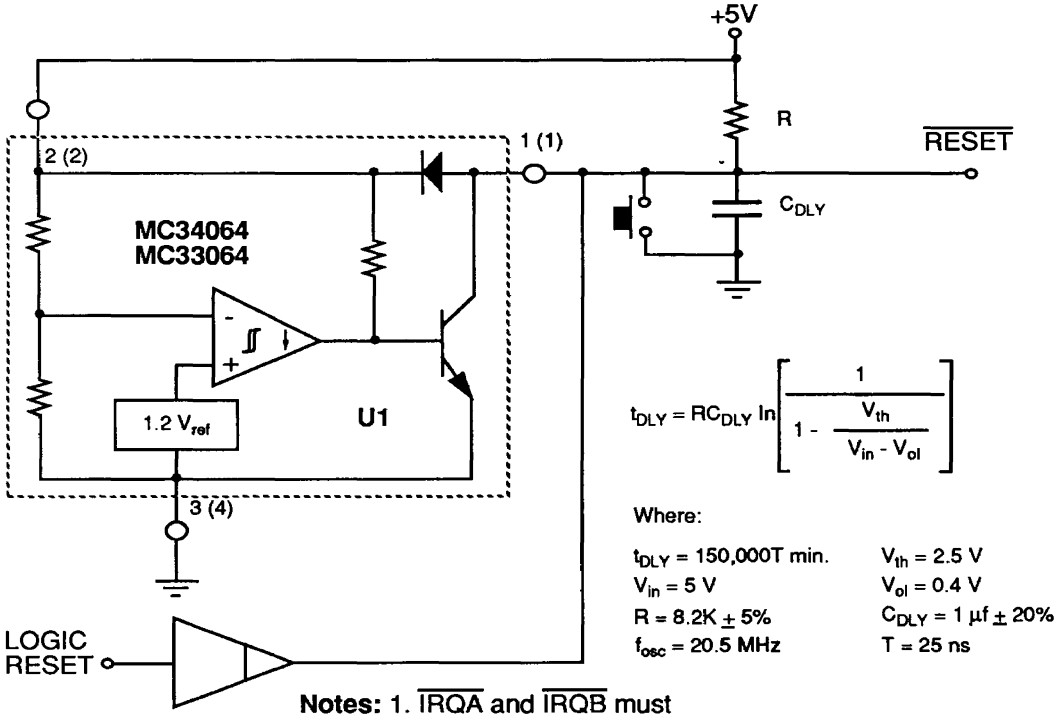


Figure B-5. Alternative Clock Circuit from the Graphic Equalizer (APR2)



Notes: 1. $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$ must be hardwired.

2. MODA and MODB must be hard wired.

Figure B-6. Reset Circuit Using MC34064/MC33064

Figure 7 illustrates how to connect a 35 ns static RAM with a 27 MHz. DSP56001. The important parameters are $T_{DW} \leq 15$ ns, $T_{DOE} \leq 15$ ns, and $T_{AA} = 35$ ns maximum. A 7.5 ns PLD is used to minimize decoding delays. Note that timing parameter #125 = tcl-4ns = 12.5 ns does not meet the SRAM requirement of 15 ns. The OR gate used to deselect $\overline{WR}$ should be an "F" part which provides 3–6.3 ns of additional delay. This additional delay provides the extra time needed for the $\overline{WR}$ signal to stay comfortably within the MCM6264 timing requirements. This example maps the static RAM into the ranges X:$1000-1FFF and Y:$1000-1FFF. The PLD equation is:

$\overline{RAM\_SELECT}$ = $\overline{PS}$ & !$\overline{DS}$ & !A15 & !A14 & !A13 & !A12

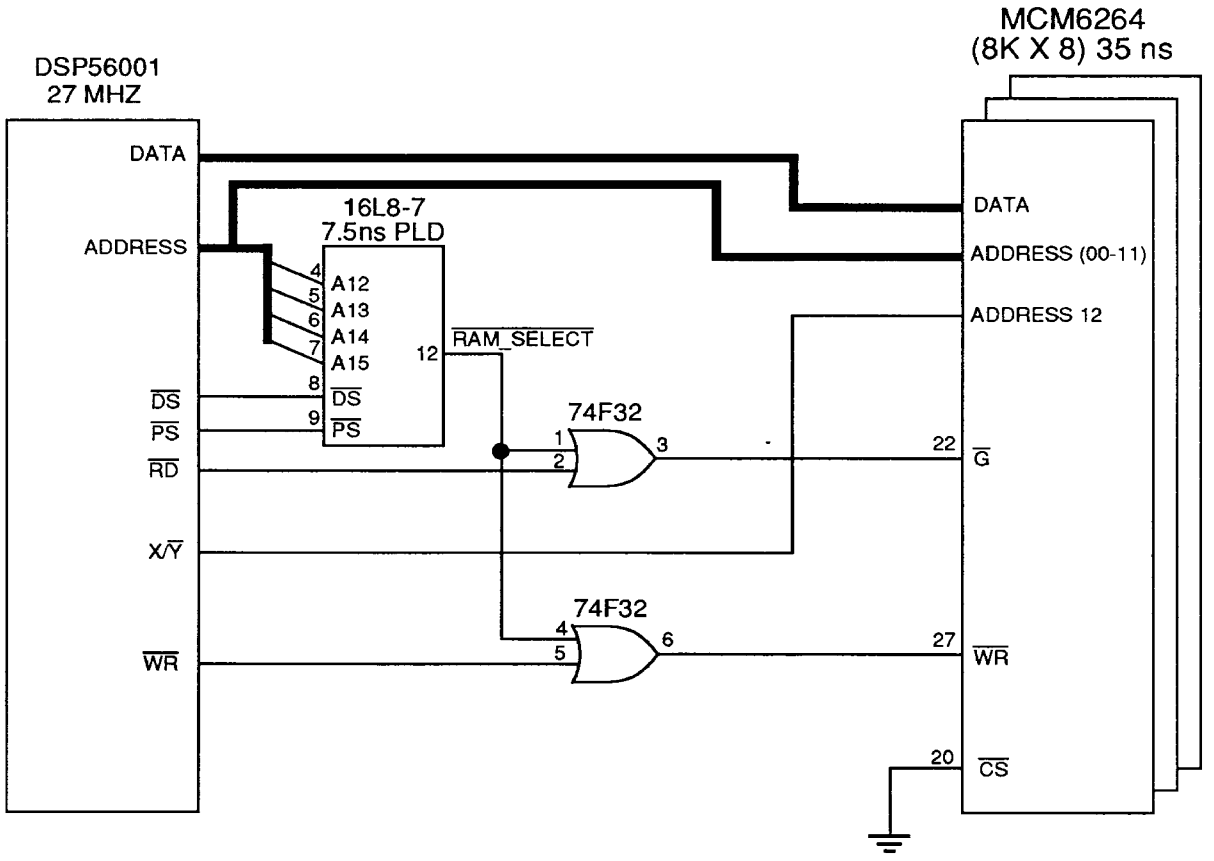


Figure B-7. 27 MHz DSP56001 with 35 ns SRAM

Figure B-8 shows the DSP56001 connected to the bus of an IBM-PC computer. The PAL equations and other details of this circuit are available in "An ISA BUS INTERFACE FOR THE DSP56001" which is provided on request by the Motorola DSP Marketing Department (512-891-2030).

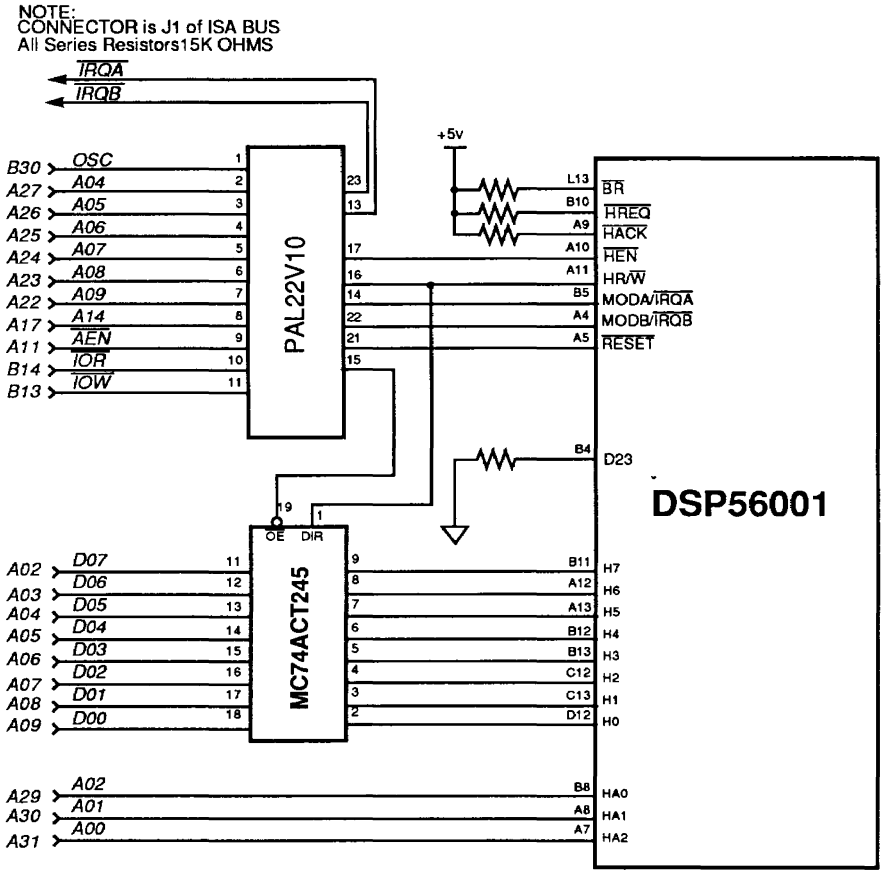


Figure B-8. DSP56001-to-ISA Bus Interface Schematic

APPENDIX C
MU-LAW / A-LAW EXPANSION TABLES

```
        ORG   X:$100
;
M_00    DC    $7D7C00   ; 8031
M_01    DC    $797C00   ; 7775
M_02    DC    $757C00   ; 7519
M_03    DC    $717C00   ; 7263
M_04    DC    $6D7C00   ; 7007
M_05    DC    $697C00   ; 6751
M_06    DC    $657C00   ; 6495
M_07    DC    $617C00   ; 6239
M_08    DC    $5D7C00   ; 5983
M_09    DC    $597C00   ; 5727
M_0A    DC    $557C00   ; 5471
M_0B    DC    $517C00   ; 5215
M_0C    DC    $4D7C00   ; 4959
M_0D    DC    $497C00   ; 4703
M_0E    DC    $457C00   ; 4447
M_0F    DC    $417C00   ; 4191
M_10    DC    $3E7C00   ; 3999
M_11    DC    $3C7C00   ; 3871
M_12    DC    $3A7C00   ; 3743
M_13    DC    $387C00   ; 3615
M_14    DC    $367C00   ; 3487
M_15    DC    $347C00   ; 3359
M_16    DC    $327C00   ; 3231
M_17    DC    $307C00   ; 3103
M_18    DC    $2E7C00   ; 2975
M_19    DC    $2C7C00   ; 2847
M_1A    DC    $2A7C00   ; 2719
M_1B    DC    $287C00   ; 2591
M_1C    DC    $267C00   ; 2463
M_1D    DC    $247C00   ; 2335
M_1E    DC    $227C00   ; 2207
M_1F    DC    $207C00   ; 2079
M_20    DC    $1EFC00   ; 1983
M_21    DC    $1DFC00   ; 1919
M_22    DC    $1CFC00   ; 1855
M_23    DC    $1BFC00   ; 1791
M_24    DC    $1AFC00   ; 1727
M_25    DC    $19FC00   ; 1663
M_26    DC    $18FC00   ; 1599
M_27    DC    $17FC00   ; 1535
M_28    DC    $16FC00   ; 1471
M_29    DC    $15FC00   ; 1407
M_2A    DC    $14FC00   ; 1343
M_2B    DC    $13FC00   ; 1279
M_2C    DC    $12FC00   ; 1215
M_2D    DC    $11FC00   ; 1151
M_2E    DC    $10FC00   ; 1087
M_2F    DC    $0FFC00   ; 1023
M_30    DC    $0F3C00   ;  975
M_31    DC    $0EBC00   ;  943
M_32    DC    $0E3C00   ;  911
M_33    DC    $0DBC00   ;  879
M_34    DC    $0D3C00   ;  847
M_35    DC    $0CBC00   ;  815
M_36    DC    $0C3C00   ;  783
M_37    DC    $0BBC00   ;  751
M_38    DC    $0B3C00   ;  719
M_39    DC    $0ABC00   ;  687
M_3A    DC    $0A3C00   ;  655
M_3B    DC    $09BC00   ;  623
M_3C    DC    $093C00   ;  591
M_3D    DC    $08BC00   ;  559
M_3E    DC    $083C00   ;  527
M_3F    DC    $07BC00   ;  495
M_40    DC    $075C00   ;  471
M_41    DC    $071C00   ;  455
M_42    DC    $06DC00   ;  439
M_43    DC    $069C00   ;  423
M_44    DC    $065C00   ;  407
M_45    DC    $061C00   ;  391
M_46    DC    $05DC00   ;  375
M_47    DC    $059C00   ;  359
M_48    DC    $055C00   ;  343
M_49    DC    $051C00   ;  327
M_4A    DC    $04DC00   ;  311
M_4B    DC    $049C00   ;  295
M_4C    DC    $045C00   ;  279
M_4D    DC    $041C00   ;  263
M_4E    DC    $03DC00   ;  247
M_4F    DC    $039C00   ;  231
M_50    DC    $036C00   ;  219
M_51    DC    $034C00   ;  211
M_52    DC    $032C00   ;  203
M_53    DC    $030C00   ;  195
M_54    DC    $02EC00   ;  187
M_55    DC    $02CC00   ;  179
M_56    DC    $02AC00   ;  171
M_57    DC    $028C00   ;  163
M_58    DC    $026C00   ;  155
M_59    DC    $024C00   ;  147
M_5A    DC    $022C00   ;  139
M_5B    DC    $020C00   ;  131
M_5C    DC    $01EC00   ;  123
M_5D    DC    $01CC00   ;  115
M_5E    DC    $01AC00   ;  107
M_5F    DC    $018C00   ;   99
M_60    DC    $017400   ;   93
M_61    DC    $016400   ;   89
M_62    DC    $015400   ;   85
M_63    DC    $014400   ;   81
M_64    DC    $013400   ;   77
M_65    DC    $012400   ;   73
M_66    DC    $011400   ;   69
M_67    DC    $010400   ;   65
M_68    DC    $00F400   ;   61
M_69    DC    $00E400   ;   57
M_6A    DC    $00D400   ;   53
M_6B    DC    $00C400   ;   49
M_6C    DC    $00B400   ;   45
M_6D    DC    $00A400   ;   41
M_6E    DC    $009400   ;   37
M_6F    DC    $008400   ;   33
M_70    DC    $007800   ;   30
M_71    DC    $007000   ;   28
M_72    DC    $006800   ;   26
M_73    DC    $006000   ;   24
M_74    DC    $005800   ;   22
M_75    DC    $005000   ;   20
M_76    DC    $004800   ;   18
M_77    DC    $004000   ;   16
M_78    DC    $003800   ;   14
M_79    DC    $003000   ;   12
M_7A    DC    $002800   ;   10
M_7B    DC    $002000   ;    8
M_7C    DC    $001800   ;    6
M_7D    DC    $001000   ;    4
M_7E    DC    $000800   ;    2
M_7F    DC    $000000   ;    0
```

Figure C-1. Mu-Law/A-Law Expansion Table Contents (Sheet 1 of 2)

```
A_80    DC    $158000  ;  688
A_81    DC    $148000  ;  656
A_82    DC    $178000  ;  752
A_83    DC    $168000  ;  720
A_84    DC    $118000  ;  560
A_85    DC    $108000  ;  528
A_86    DC    $138000  ;  624
A_87    DC    $128000  ;  592
A_88    DC    $1D8000  ;  944
A_89    DC    $1C8000  ;  912
A_8A    DC    $1F8000  ; 1008
A_8B    DC    $1E8000  ;  976
A_8C    DC    $198000  ;  816
A_8D    DC    $188000  ;  784
A_8E    DC    $1B8000  ;  880
A_8F    DC    $1A8000  ;  848
A_90    DC    $0AC000  ;  344
A_91    DC    $0A4000  ;  328
A_92    DC    $0BC000  ;  376
A_93    DC    $0B4000  ;  360
A_94    DC    $08C000  ;  280
A_95    DC    $084000  ;  264
A_96    DC    $09C000  ;  312
A_97    DC    $094000  ;  296
A_98    DC    $0EC000  ;  472
A_99    DC    $0E4000  ;  456
A_9A    DC    $0FC000  ;  504
A_9B    DC    $0F4000  ;  488
A_9C    DC    $0CC000  ;  408
A_9D    DC    $0C4000  ;  392
A_9E    DC    $0DC000  ;  440
A_9F    DC    $0D4000  ;  424
A_A0    DC    $560000  ; 2752
A_A1    DC    $520000  ; 2624
A_A2    DC    $5E0000  ; 3008
A_A3    DC    $5A0000  ; 2880
A_A4    DC    $460000  ; 2240
A_A5    DC    $420000  ; 2112
A_A6    DC    $4E0000  ; 2496
A_A7    DC    $4A0000  ; 2368
A_A8    DC    $760000  ; 3776
A_A9    DC    $720000  ; 3648
A_AA    DC    $7E0000  ; 4032
A_AB    DC    $7A0000  ; 3904
A_AC    DC    $660000  ; 3264
A_AD    DC    $620000  ; 3136
A_AE    DC    $6E0000  ; 3520
A_AF    DC    $6A0000  ; 3392
A_B0    DC    $2B0000  ; 1376
A_B1    DC    $290000  ; 1312
A_B2    DC    $2F0000  ; 1504
A_B3    DC    $2D0000  ; 1440
A_B4    DC    $230000  ; 1120
A_B5    DC    $210000  ; 1056
A_B6    DC    $270000  ; 1248
A_B7    DC    $250000  ; 1184
A_B8    DC    $3B0000  ; 1888
A_B9    DC    $390000  ; 1824
A_BA    DC    $3F0000  ; 2016
A_BB    DC    $3D0000  ; 1952
A_BC    DC    $330000  ; 1632
A_BD    DC    $310000  ; 1568
A_BE    DC    $370000  ; 1760
A_BF    DC    $350000  ; 1696

A_C0    DC    $015800  ;   43
A_C1    DC    $014800  ;   41
A_C2    DC    $017800  ;   47
A_C3    DC    $016800  ;   45
A_C4    DC    $011800  ;   35
A_C5    DC    $010800  ;   33
A_C6    DC    $013800  ;   39
A_C7    DC    $012800  ;   37
A_C8    DC    $01D800  ;   59
A_C9    DC    $01C800  ;   57
A_CA    DC    $01F800  ;   63
A_CB    DC    $01E800  ;   61
A_CC    DC    $019800  ;   51
A_CD    DC    $018800  ;   49
A_CE    DC    $01B800  ;   55
A_CF    DC    $01A800  ;   53
A_D0    DC    $005800  ;   11
A_D1    DC    $004800  ;    9
A_D2    DC    $007800  ;   15
A_D3    DC    $006800  ;   13
A_D4    DC    $001800  ;    3
A_D5    DC    $000800  ;    1
A_D6    DC    $003800  ;    7
A_D7    DC    $002800  ;    5
A_D8    DC    $00D800  ;   27
A_D9    DC    $00C800  ;   25
A_DA    DC    $00F800  ;   31
A_DB    DC    $00E800  ;   29
A_DC    DC    $009800  ;   19
A_DD    DC    $008800  ;   17
A_DE    DC    $00B800  ;   23
A_DF    DC    $00A800  ;   21
A_E0    DC    $056000  ;  172
A_E1    DC    $052000  ;  164
A_E2    DC    $05E000  ;  188
A_E3    DC    $05A000  ;  180
A_E4    DC    $046000  ;  140
A_E5    DC    $042000  ;  132
A_E6    DC    $04E000  ;  156
A_E7    DC    $04A000  ;  148
A_E8    DC    $076000  ;  236
A_E9    DC    $072000  ;  228
A_EA    DC    $07E000  ;  252
A_EB    DC    $07A000  ;  244
A_EC    DC    $066000  ;  204
A_ED    DC    $062000  ;  196
A_EE    DC    $06E000  ;  220
A_EF    DC    $06A000  ;  212
A_F0    DC    $02B000  ;   86
A_F1    DC    $029000  ;   82
A_F2    DC    $02F000  ;   94
A_F3    DC    $02D000  ;   90
A_F4    DC    $023000  ;   70
A_F5    DC    $021000  ;   66
A_F6    DC    $027000  ;   78
A_F7    DC    $025000  ;   74
A_F8    DC    $03B000  ;  118
A_F9    DC    $039000  ;  114
A_FA    DC    $03F000  ;  126
A_FB    DC    $03D000  ;  122
A_FC    DC    $033000  ;  102
A_FD    DC    $031000  ;   98
A_FE    DC    $037000  ;  110
A_FF    DC    $035000  ;  106
```

Figure C-1. Mu-Law/A-Law Expansion Table Contents (Sheet 2 of 2)

APPENDIX D
SINE WAVE TABLE

This sine wave table is normally used by FFT routines which use bit reversed address pointers. This table can be used as it is for up to 512 point FFTs; however, for larger FFTs, the table must be copied to a different memory location to allow the reverse-carry addressing mode to be used (see **Section 5.3.2.3 REVERSE-CARRY MODIFIER (Mn=$0000)** in the **DSP56000/DSP56001 Digital Signal Processor User's Manual** for additional information).

```
        ORG   Y:$100
;
S_00    DC    $000000  ; +0.0000000000
S_01    DC    $03242B  ; +0.0245412998
S_02    DC    $0647D9  ; +0.0490676016
S_03    DC    $096A90  ; +0.0735644996
S_04    DC    $0C8BD3  ; +0.0980170965
S_05    DC    $0FAB27  ; +0.1224106997
S_06    DC    $12C810  ; +0.1467303932
S_07    DC    $15E214  ; +0.1709619015
S_08    DC    $18F8B8  ; +0.1950902939
S_09    DC    $1C0B82  ; +0.2191012055
S_0A    DC    $1F19F9  ; +0.2429800928
S_0B    DC    $2223A5  ; +0.2667128146
S_0C    DC    $25280C  ; +0.2902846038
S_0D    DC    $2826B9  ; +0.3136816919
S_0E    DC    $2B1F35  ; +0.3368898928
S_0F    DC    $2E110A  ; +0.3598949909
S_10    DC    $30FBC5  ; +0.3826833963
S_11    DC    $33DEF3  ; +0.4052414000
S_12    DC    $36BA20  ; +0.4275551140
S_13    DC    $398CDD  ; +0.4496113062
S_14    DC    $3C56BA  ; +0.4713967144
S_15    DC    $3F174A  ; +0.4928981960
S_16    DC    $41CE1E  ; +0.5141026974
S_17    DC    $447ACD  ; +0.5349975824
S_18    DC    $471CED  ; +0.5555701852
S_19    DC    $49B415  ; +0.5758082271
S_1A    DC    $4C3FE0  ; +0.5956993103
S_1B    DC    $4EBFE9  ; +0.6152315736
S_1C    DC    $5133CD  ; +0.6343932748
S_1D    DC    $539B2B  ; +0.6531729102
S_1E    DC    $55F5A5  ; +0.6715589762
S_1F    DC    $5842DD  ; +0.6895405054
S_20    DC    $5A827A  ; +0.7071068287
S_21    DC    $5CB421  ; +0.7242470980
S_22    DC    $5ED77D  ; +0.7409511805
S_23    DC    $60EC38  ; +0.7572088242
S_24    DC    $62F202  ; +0.7730104923
S_25    DC    $64E889  ; +0.7883464098
S_26    DC    $66CF81  ; +0.8032075167
S_27    DC    $68A69F  ; +0.8175848722
S_28    DC    $6A6D99  ; +0.8314697146
S_29    DC    $6C2429  ; +0.8448535204
S_2A    DC    $6DCA0D  ; +0.8577286005
S_2B    DC    $6F5F03  ; +0.8700870275
S_2C    DC    $70E2CC  ; +0.8819212914
S_2D    DC    $72552D  ; +0.8932244182
S_2E    DC    $73B5EC  ; +0.9039893150
S_2F    DC    $7504D3  ; +0.9142097235
S_30    DC    $7641AF  ; +0.9238795042
S_31    DC    $776C4F  ; +0.9329928160
S_32    DC    $788484  ; +0.9415441155
S_33    DC    $798A24  ; +0.9495282173
S_34    DC    $7A7D05  ; +0.9569402933
S_35    DC    $7B5D04  ; +0.9637761116
S_36    DC    $7C29FC  ; +0.9700313210
S_37    DC    $7CE3CF  ; +0.9757022262
S_38    DC    $7D8A5F  ; +0.9807853103
S_39    DC    $7E1D94  ; +0.9852777123
S_3A    DC    $7E9D56  ; +0.9891765118
S_3B    DC    $7F0992  ; +0.9924796224
S_3C    DC    $7F6237  ; +0.9951847792
S_3D    DC    $7FA737  ; +0.9972904921
S_3E    DC    $7FD888  ; +0.9987955093
S_3F    DC    $7FF622  ; +0.9996988773
S_40    DC    $7FFFFF  ; +0.9999998808
S_41    DC    $7FF622  ; +0.9996988773
S_42    DC    $7FD888  ; +0.9987955093
S_43    DC    $7FA737  ; +0.9972904921
S_44    DC    $7F6237  ; +0.9951847792
S_45    DC    $7F0992  ; +0.9924796224
S_46    DC    $7E9D56  ; +0.9891765118
S_47    DC    $7E1D94  ; +0.9852777123
S_48    DC    $7D8A5F  ; +0.9807853103
S_49    DC    $7CE3CF  ; +0.9757022262
S_4A    DC    $7C29FC  ; +0.9700313210
S_4B    DC    $7B5D04  ; +0.9637761116
S_4C    DC    $7A7D05  ; +0.9569402933
S_4D    DC    $798A24  ; +0.9495282173
S_4E    DC    $788484  ; +0.9415441155
S_4F    DC    $776C4F  ; +0.9329928160
S_50    DC    $7641AF  ; +0.9238795042
S_51    DC    $7504D3  ; +0.9142097235
S_52    DC    $73B5EC  ; +0.9039893150
S_53    DC    $72552D  ; +0.8932244182
S_54    DC    $70E2CC  ; +0.8819212914
S_55    DC    $6F5F03  ; +0.8700870275
S_56    DC    $6DCA0D  ; +0.8577286005
S_57    DC    $6C2429  ; +0.8448535204
S_58    DC    $6A6D99  ; +0.8314697146
S_59    DC    $68A69F  ; +0.8175848722
S_5A    DC    $66CF81  ; +0.8032075167
S_5B    DC    $64E889  ; +0.7883464098
S_5C    DC    $62F202  ; +0.7730104923
S_5D    DC    $60EC38  ; +0.7572088242
S_5E    DC    $5ED77D  ; +0.7409511805
S_5F    DC    $5CB421  ; +0.7242470980
S_60    DC    $5A827A  ; +0.7071068287
S_61    DC    $5842DD  ; +0.6895405054
S_62    DC    $55F5A5  ; +0.6715589762
S_63    DC    $539B2B  ; +0.6531729102
S_64    DC    $5133CD  ; +0.6343932748
S_65    DC    $4EBFE9  ; +0.6152315736
S_66    DC    $4C3FE0  ; +0.5956993103
S_67    DC    $49B415  ; +0.5758082271
S_68    DC    $471CED  ; +0.5555701852
S_69    DC    $447ACD  ; +0.5349975824
S_6A    DC    $41CE1E  ; +0.5141026974
S_6B    DC    $3F174A  ; +0.4928981960
S_6C    DC    $3C56BA  ; +0.4713967144
S_6D    DC    $398CDD  ; +0.4496113062
S_6E    DC    $36BA20  ; +0.4275551140
S_6F    DC    $33DEF3  ; +0.4052414000
S_70    DC    $30FBC5  ; +0.3826833963
S_71    DC    $2E110A  ; +0.3598949909
S_72    DC    $2B1F35  ; +0.3368898928
```

Figure D-1. Sine Wave Table Contents (Sheet 1 of 3)

```
S_73    DC    $2826B9  ; +0.3136816919
S_74    DC    $25280C  ; +0.2902846038
S_75    DC    $2223A5  ; +0.2667128146
S_76    DC    $1F19F9  ; +0.2429800928
S_77    DC    $1C0B82  ; +0.2191012055
S_78    DC    $18F8B8  ; +0.1950902939
S_79    DC    $15E214  ; +0.1709619015
S_7A    DC    $12C810  ; +0.1467303932
S_7B    DC    $0FAB27  ; +0.1224106997
S_7C    DC    $0C8BD3  ; +0.0980170965
S_7D    DC    $096A90  ; +0.0735644996
S_7E    DC    $0647D9  ; +0.0490676016
S_7F    DC    $03242B  ; +0.0245412998
S_80    DC    $000000  ; +0.0000000000
S_81    DC    $FCDBD5  ; -0.0245412998
S_82    DC    $F9B827  ; -0.0490676016
S_83    DC    $F69570  ; -0.0735644996
S_84    DC    $F3742D  ; -0.0980170965
S_85    DC    $F054D9  ; -0.1224106997
S_86    DC    $ED37F0  ; -0.1467303932
S_87    DC    $EA1DEC  ; -0.1709619015
S_88    DC    $E70748  ; -0.1950902939
S_89    DC    $E3F47E  ; -0.2191012055
S_8A    DC    $E0E607  ; -0.2429800928
S_8B    DC    $DDDC5B  ; -0.2667128146
S_8C    DC    $DAD7F4  ; -0.2902846038
S_8D    DC    $D7D947  ; -0.3136816919
S_8E    DC    $D4E0CB  ; -0.3368898928
S_8F    DC    $D1EEF6  ; -0.3598949909
S_90    DC    $CF043B  ; -0.3826833963
S_91    DC    $CC210D  ; -0.4052414000
S_92    DC    $C945E0  ; -0.4275551140
S_93    DC    $C67323  ; -0.4496113062
S_94    DC    $C3A946  ; -0.4713967144
S_95    DC    $C0E8B6  ; -0.4928981960
S_96    DC    $BE31E2  ; -0.5141026974
S_97    DC    $BB8533  ; -0.5349975824
S_98    DC    $B8E313  ; -0.5555701852
S_99    DC    $B64BEB  ; -0.5758082271
S_9A    DC    $B3C020  ; -0.5956993103
S_9B    DC    $B14017  ; -0.6152315736
S_9C    DC    $AECC33  ; -0.6343932748
S_9D    DC    $AC64D5  ; -0.6531729102
S_9E    DC    $AA0A5B  ; -0.6715589762
S_9F    DC    $A7BD23  ; -0.6895405054
S_A0    DC    $A57D86  ; -0.7071068287
S_A1    DC    $A34BDF  ; -0.7242470980
S_A2    DC    $A12883  ; -0.7409511805
S_A3    DC    $9F13C8  ; -0.7572088242
S_A4    DC    $9D0DFE  ; -0.7730104923
S_A5    DC    $9B1777  ; -0.7883464098
S_A6    DC    $99307F  ; -0.8032075167
S_A7    DC    $975961  ; -0.8175848722
S_A8    DC    $959267  ; -0.8314697146
S_A9    DC    $93DBD7  ; -0.8448535204
S_AA    DC    $9235F3  ; -0.8577286005
S_AB    DC    $90A0FD  ; -0.8700870275
S_AC    DC    $8F1D34  ; -0.8819212914
S_AD    DC    $8DAAD3  ; -0.8932244182
S_AE    DC    $8C4A14  ; -0.9039893150
S_AF    DC    $8AFB2D  ; -0.9142097235
S_B0    DC    $89BE51  ; -0.9238795042
S_B1    DC    $8893B1  ; -0.9329928160
S_B2    DC    $877B7C  ; -0.9415441155
S_B3    DC    $8675DC  ; -0.9495282173
S_B4    DC    $8582FB  ; -0.9569402933
S_B5    DC    $84A2FC  ; -0.9637761116
S_B6    DC    $83D604  ; -0.9700313210
S_B7    DC    $831C31  ; -0.9757022262
S_B8    DC    $8275A1  ; -0.9807853103
S_B9    DC    $81E26C  ; -0.9852777123
S_BA    DC    $8162AA  ; -0.9891765118
S_BB    DC    $80F66E  ; -0.9924796224
S_BC    DC    $809DC9  ; -0.9951847792
S_BD    DC    $8058C9  ; -0.9972904921
S_BE    DC    $802778  ; -0.9987955093
S_BF    DC    $8009DE  ; -0.9996988773
S_C0    DC    $800000  ; -1.0000000000
S_C1    DC    $8009DE  ; -0.9996988773
S_C2    DC    $802778  ; -0.9987955093
S_C3    DC    $8058C9  ; -0.9972904921
S_C4    DC    $809DC9  ; -0.9951847792
S_C5    DC    $80F66E  ; -0.9924796224
S_C6    DC    $8162AA  ; -0.9891765118
S_C7    DC    $81E26C  ; -0.9852777123
S_C8    DC    $8275A1  ; -0.9807853103
S_C9    DC    $831C31  ; -0.9757022262
S_CA    DC    $83D604  ; -0.9700313210
S_CB    DC    $84A2FC  ; -0.9637761116
S_CC    DC    $8582FB  ; -0.9569402933
S_CD    DC    $8675DC  ; -0.9495282173
S_CE    DC    $877B7C  ; -0.9415441155
S_CF    DC    $8893B1  ; -0.9329928160
S_D0    DC    $89BE51  ; -0.9238795042
S_D1    DC    $8AFB2D  ; -0.9142097235
S_D2    DC    $8C4A14  ; -0.9039893150
S_D3    DC    $8DAAD3  ; -0.8932244182
S_D4    DC    $8F1D34  ; -0.8819212914
S_D5    DC    $90A0FD  ; -0.8700870275
S_D6    DC    $9235F3  ; -0.8577286005
S_D7    DC    $93DBD7  ; -0.8448535204
S_D8    DC    $959267  ; -0.8314697146
S_D9    DC    $975961  ; -0.8175848722
S_DA    DC    $99307F  ; -0.8032075167
S_DB    DC    $9B1777  ; -0.7883464098
S_DC    DC    $9D0DFE  ; -0.7730104923
S_DD    DC    $9F13C8  ; -0.7572088242
S_DE    DC    $A12883  ; -0.7409511805
S_DF    DC    $A34BDF  ; -0.7242470980
S_E0    DC    $A57D86  ; -0.7071068287
S_E1    DC    $A7BD23  ; -0.6895405054
S_E2    DC    $AA0A5B  ; -0.6715589762
S_E3    DC    $AC64D5  ; -0.6531729102
S_E4    DC    $AECC33  ; -0.6343932748
S_E5    DC    $B14017  ; -0.6152315736
S_E6    DC    $B3C020  ; -0.5956993103
S_E7    DC    $B64BEB  ; -0.5758082271
S_E8    DC    $B8E313  ; -0.5555701852
S_E9    DC    $BB8533  ; -0.5349975824
S_EA    DC    $BE31E2  ; -0.5141026974
S_EB    DC    $C0E8B6  ; -0.4928981960
S_EC    DC    $C3A946  ; -0.4713967144
S_ED    DC    $C67323  ; -0.4496113062
S_EE    DC    $C945E0  ; -0.4275551140
S_EF    DC    $CC210D  ; -0.4052414000
S_F0    DC    $CF043B  ; -0.3826833963
S_F1    DC    $D1EEF6  ; -0.3598949909
S_F2    DC    $D4E0CB  ; -0.3368898928
S_F3    DC    $D7D947  ; -0.3136816919
S_F4    DC    $DAD7F4  ; -0.2902846038
```

Figure D-1. Sine Wave Table Contents (Sheet 2 of 3)

```
S_F5    DC    $DDDC5B  ; -0.2667128146
S_F6    DC    $E0E607  ; -0.2429800928
S_F7    DC    $E3F47E  ; -0.2191012055
S_F8    DC    $E70748  ; -0.1950902939
S_F9    DC    $EA1DEC  ; -0.1709619015
S_FA    DC    $ED37F0  ; -0.1467303932
S_FB    DC    $F054D9  ; -0.1224106997
S_FC    DC    $F3742D  ; -0.0980170965
S_FD    DC    $F69570  ; -0.0735644996
S_FE    DC    $F9B827  ; -0.0490676016
S_FF    DC    $FCDBD5  ; -0.0245412998
```

Figure D-1. Sine Wave Table Contents (Sheet 3 of 3)