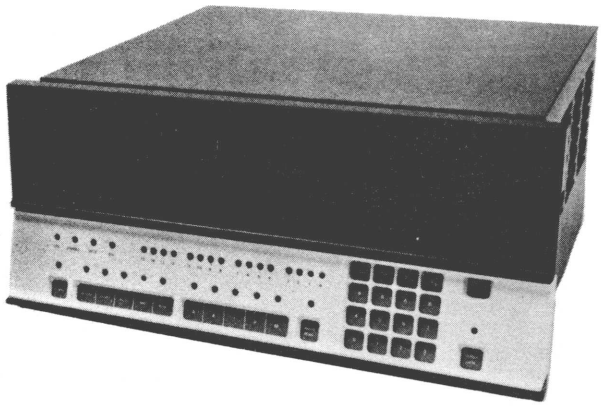
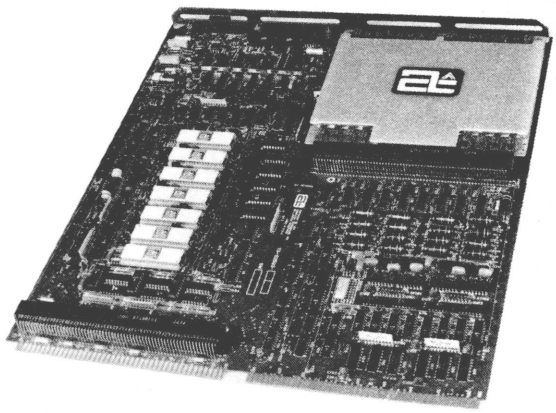
91-20400-00A2
OCTOBER **1974**

NAKED MINI® LSI SERIES

COMPUTER HANDBOOK



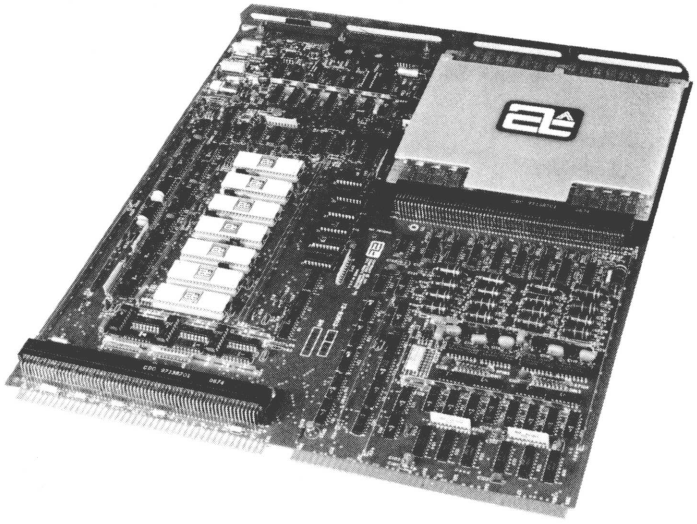


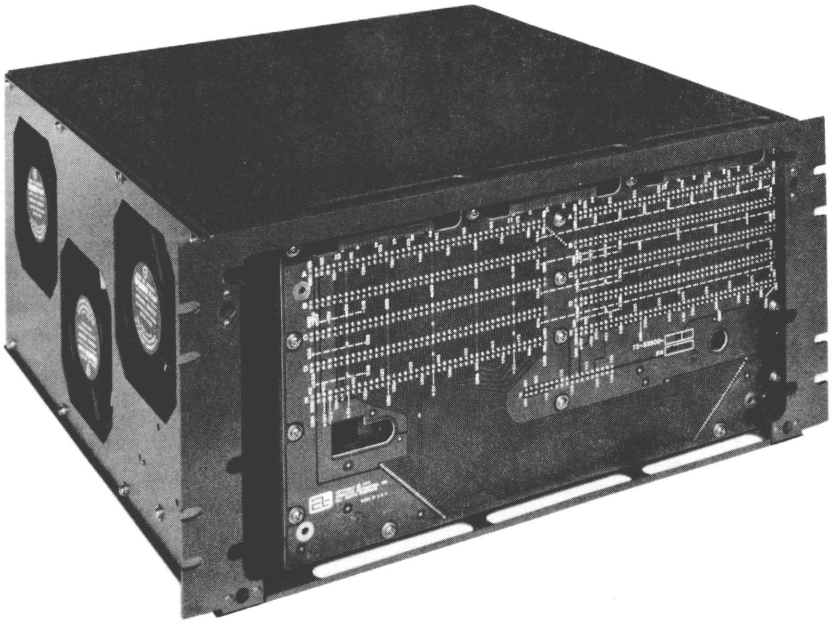
Patent Pending

ComputerAutomation
Naked Mini® Division
18651 Von Karman, Irvine, Calif. 92664
Tel. 714-833-8830 TWX 910-595-1767

COPYRIGHT 1973, COMPUTER AUTOMATION, INC.

NAKED MINI LSI-1





NAKED MINI LSI-2/20

ALPHA LSI

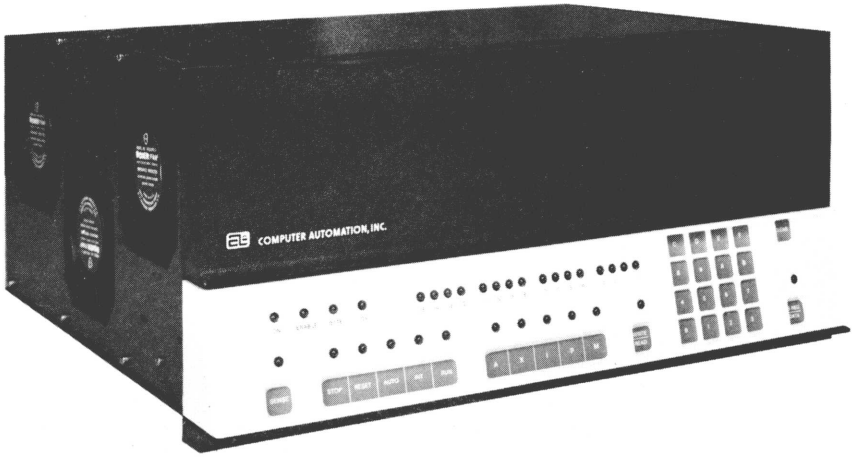


TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF TABLES

TABLE OF CONTENTS (Cont'd)

LIST OF TABLES (Cont'd)

Section 1

GENERAL DESCRIPTION

1.1 INTRODUCTION

The ALPHA LSI and NAKED MINI®LSI (hereafter referred to as ALPHA LSI when discussed together) are general purpose, stored program digital computers. They are extensions of the successful and proven 16-bit computer family from Computer Automation, Inc.

1.1.1 The ALPHA LSI Family

The ALPHA LSI is not just one computer that can be packaged with or without a chassis, power supply and console. Instead, it is an integrated family of compatible components -- two central processors; three kinds of memories in fourteen sizes and three speeds; peripheral controllers; computer options, general purpose interfaces; etc. -- which can be combined in a multitude of configurations to match a wide range of needs.

Several central processors are available and are referred to as the NAKED MINI LSI type 1 (LSI-1) and the NAKED MINI LSI type 2 (LSI-2). The LSI-1 and LSI-2 Processors feature the same basic architecture, instruction set and I/O capabilities. They differ in terms of performance wherein the LSI-2 is faster than the LSI-1 and also features additional instructions. Both processors are plug-to-plug compatible and, except for timing differences, programs will execute properly in either Processor without change (except when the additional instructions applicable to LSI-2 only are used).

Several memories are available: Core 980, Core 1200, Core 1600, and semiconductor - SC1200. The numbers define the full cycle time of the memory in nanoseconds and each memory type can be interleaved.

The user can mix memories of varying speeds, sizes, and technologies with either processor to obtain the best price/performance margin possible.

1.1.2 Upward Compatibility

The ALPHA LSI is upward software and I/O compatible with earlier 16-bit computers from Computer Automation. Upward software compatibility means that virtually all programs written for the earlier 16-bit computers will run without change on the

ALPHA LSI. However, due to the expanded and improved instruction set of the ALPHA LSI, programs written for these computers may not run on the earlier computers.

1.1.3 General Features

The ALPHA LSI computer family features a 16-bit word format and 168 basic instructions (188 in the LSI-2). The instruction set is divided into seven major classes (eight with LSI-2) which provide memory-to-register and register-to-register data movement as well as conditional jump, single and double-register shift, register change, machine control and Input/Output instructions. The computer utilizes eight addressing modes (12 in the LSI-2) for effective and efficient management of memory resources.

The ALPHA LSI computer has fully buffered I/O structure coupled with five levels of interrupts and five I/O modes which permit high speed, low speed, synchronous and asynchronous data transfers to take place.

The ALPHA LSI may readily accommodate additional memory modules and I/O by adding expansion chassis to the basic system. An optional Memory Banking feature permits the user to extend the upper limit of Memory from 32K words to 256K words.

1.2 THE NAKED MINI LSI CONCEPT

The NAKED MINI LSI-1 computer consists of the Processor and first memory module on one printed circuit (PC) board. The NAKED MINI LSI-1 is a complete stand alone computer without a chassis, motherboard, power supply or operators console.

The NAKED MINI LSI-1 computer is designed to be used as a system component along with other system components. It depends on the system power supply for a power source, the system control panel for operational control signals, and the system enclosure for structural and environmental support.

The NAKED MINI LSI-2 computer consists of the Processor (full PC board) and one or more memory modules, a motherboard and a chassis. Like the LSI-1, the NAKED MINI LSI-2 Processor depends on the system power supply for power and a system control panel for operational control signals.

1.3 THE ALPHA LSI

Take a NAKED MINI LSI-1 or -2 computer and add a power supply module, a motherboard, a chassis and an operator's console and you get the ALPHA LSI computer. The Motherboard interconnects the NAKED MINI LSI computer with additional I/O and memory modules, the power supply, and the operator's console.

1.4 CHARACTERISTICS

The characteristics of the ALPHA LSI are explained in subsequent sections of this manual. The following is an overview of the characteristics of this computer.

1.4.1 Processor and Memory

Some of the significant characteristics of the Processor and Memory are:

Parallel processing of full 16-bit words and 8-bit bytes.

Seven 16-bit hardware registers, one 8-bit Status register.

Memory word size of 16 bits, with each word addressable as a full 16-bit word or as two separate 8-bit bytes.

Memory capacity is 1,024 words minimum, expandable to 32,768 words per bank maximum. (Up to 262,144 words with optional Memory Banking.)

Computer cycle time is 1.6 microseconds for LSI-1; 150 nanoseconds for LSI-2.

Direct Memory Access (standard) provides data transfer rates up to 1,020,000 words per second in a single memory bank or 1,666,667 words per second with interleaved memory banks.

Binary 2's complement arithmetic processing.

Automatic memory scan (standard).

Hardware Multiply and Divide (standard).

1.4.2 Instruction Set

These computers have a very powerful instruction set consisting of 168 basic instructions divided into seven classes (188 instructions and 8 classes with the LSI-2 Processor). The instruction classes are:

1. Memory Reference.

   Access Memory in either full Word or Byte mode and perform logical and arithmetic operations involving data in Memory and data in hardware registers. The hardware Multiply, Divide and Normalize instructions are included in this class.

2. Stack (LSI-2 only)

   Similar to the Memory Reference class of instructions except they operate on words maintained in "stacks" in Memory. The number, size, and location of stacks in use at any time are unlimited, as are the number of stacks in use by any code module, and the number of code modules using any given stack.

3. Byte Immediate.

   Similar to the Memory Reference class in that logical and arithmetic operations are performed involving data in hardware registers. The memory data, however, is contained within the instruction word so that it is immediately available for processing without requiring an operand cycle to fetch it from Memory.

4. Conditional Jump.

   Test conditions within the Processor and perform conditional branches depending on the results of the tests performed. Jump may be as much as ± 64 locations from the location of the conditional jump instruction.

5. Shift.

   Include singleregister logical, arithmetic, and rotate shifts; double register logical and rotate shifts.

6. Register Change.

   Provide logical manipulation of data within hardware registers.

7. Control.

   Enable and disable interrupts; suppress status, control word, or byte mode data processing; and perform other general control functions.

8. Input/Output.

   Provide communications between the computer and external devices They include conventional I/O instructions plus Block Transfer and Automatic Input/Output instructions. I/O may be to/from register or directly to/from Memory.

1.4.3 Registers

Following are descriptions of hardware registers of interest to the operator and programmer. Except for the I and P registers, all others are under program control.

1. A Register. A 16-bit register used for arithmetic, logical and input/output operations.
2. X Register. A 16-bit register that holds the index value for memory address modification. It is also used for input/output and certain arithmetic and logic operations.
3. OV (Overflow). A one-bit register set by arithmetic logic when an overflow occurs. It is also used for extended shift operation. It can be tested and modified by software.
4. BM (Byte mode). A one-bit register that specifies either word or byte mode. It is set and cleared by software.
5. EIN (Enable Interrupts). A one-bit register that, when set, enables interrupts of processor operation. It is set and cleared by software.
6. I Register. A 16-bit register that holds the instruction currently being processed by the computer.

7. P Register. A 16-bit register that holds the program location counter. It addresses each instruction and increments as each instruction is executed. For skip or jump instructions (modifying normal program sequence), P is loaded with the next instruction to be executed.

1.4.4 Memory Addressing

1.4.4.1 Memory Reference Addressing

An important feature of these machines is the ability to access full 16-bit words and 8-bit bytes (half words) in Memory. Memory may be as small as 1K x 16-bit words, and as large as 32K x 16-bit words. Since Memory may contain 32K words, and since each word contains two bytes, provisions are made for addressing up to 64K bytes.

Instructions which access Memory may operate in either Word or Byte mode. Memory Reference instructions are sixteen bits in length (one-word instructions), with the eight least-significant bits, plus three control bits, dedicated to memory addressing. The eight least-significant bits address 256 words or bytes. The ALPHA LSI computer uses the three control bits to specify several addressing modes. These addressing modes are discussed briefly below and are explained in detail in section 3. The addressing modes used are Scratchpad, Relative Forward, Relative Backward, Indexed, and Indirect.

1. Scratchpad

   Scratchpad addressing accesses the first 256 words in Memory in Word mode, or the first 256 bytes in Byte mode. The first 256 words in Memory are referred to as "Scratchpad" memory, because these are common words which can be addressed words which can be addressed directly by instructions located anywhere in Memory.

2. Relative.

   In Word mode, relative addressing can address an area of Memory extending from the instruction address forward 256 words (+256) or backward 255 words (-255). In Byte mode, the range is forward 512 bytes. Bytes cannot be directly addressed relative backward.

3. Indexed.

   The Index (X) register can be added to the address field of Memory Reference instructions to form an effective memory word or byte address.

4. Indirect.

   Indirect addressing uses scratchpad or relative addressing to access a word in Memory which contains the address of a memory operand. The word that contains a memory address rather than an operand is called an address pointer. In Word mode, multi-

level indirect addressing is possible; i.e., one address pointer may contain the address of another address pointer rather than the address of an operand. In Byte mode, only one level of indirect addressing is possible.

Indirect addressing may also be used in conjunction with indexing. When indexed indirect addressing is specified, the indirect operation is performed first and then the contents of the X register are added to the contents of the address pointer. This process is called Post Indexing.

1.4.4.2 Stack Addressing

All stack accesses are controlled by a stack pointer. Stacks may be accessed in the conventional "PUSH" and "POP" fashion utilizing automatic hardware predecrement and postincrement respectively, of the stack pointer. Stack contents can also be accessed directly or with indexing through the stack pointer without altering the stack pointer value.

1.4.5 I/O Structure

The ALPHA LSI series computers are highly flexible system components designed for easy application to control, communications, and monitoring tasks. These computers are extremely easy to program using assembly language. Organization of the Processor enables the computer to obtain high memory efficiency, avoiding the problem of "core burning", so prevalent in many computers. Memory utilization is further enhanced by the powerful and flexible I/O instruction set. The I/O structure is simple and efficient, sharply reducing the amount of I/O logic required by units interfacing with the Processor.

1.4.5.1 Control Modes

Two type of I/O instructions, Select and Sense, provide control information to and from an interface. The Select instructions establish operating modes, control interrupts or initialize the interface. The Sense instructions permit the Processor to obtain the operational status of an interface.

1.4.5.2 Input Output Modes

The ALPHA LSI computer features five distinct I/O modes which, when combined with an extensive set of I/O instructions, provides a very powerful and easy to use I/O

structure. These modes are:

1. Programmed I/O via Registers
2. Programmed I/O via Memory
3. Automatic I/O under Interrupts
4. Block I/O
5. DMA

Transfers can be made to or from the A or X registers or directly to or from Memory, whichever is more convenient. Both word and byte data can be handled directly, with byte data being packed automatically, if desired, without the need for time and space-consuming programmed routines.

1. Programmed Input/Output via Registers

   For greater convenience in handling data that must be examined immediately upon input, or is the result of computations that must be output immediately, programmed I/O transfers the data directly to and from the operating registers of the Processor. Furthermore, programmed I/O instructions can be combined with Sense and Skip instructions to allow testing of controller or peripheral status prior to making a transfer.

2. Programmed Input/Output via Memory

   This mode capitalizes on the power of the Automatic I/O instructions to transfer data to or from Memory without disturbing the working registers of the Processor. Any size block of data may be transferred into or out of Memory.

3. Automatic Input/Output under Interrupt Control

   This mode permits an interface to transfer data to or from Memory at its own data rate with minimal disturbance of the main program. When all data has been transferred, the interface develops an End-of-Block interrupt. This, in turn, causes an interrupt subroutine to be entered which performs the necessary housekeeping associated with End-of-Block operations.

4. Block Input/Output

   For high speed transfer rates, Block I/O transfers data blocks of any length. Data is exchanged directly between Memory and the peripheral interface with the index register providing the word count. During execution of Block I/O instructions, the computer is totally dedicated to the Block I/O transfer and cannot respond to interrupts until the entire block has been transferred.

5. Direct Memory Access (DMA)

   For very high speed transfer rates, DMA transfers data directly to and from Memory. Since this data transfer does not require the Processor, the Processor can be performing other operations while interleaving with DMA on a cycle stealing basis. Multiple DMA controllers may use the DMA feature simultaneously (interleaved cycles) up to the full memory transfer rate. When more than one memory module is installed, the modules may be two way interleaved to provide data transfer at twice the individual memory data rates.

1.4.5.3 Vectored Interrupts.

The LSI series computers feature vectored hardware priority interrupts, wherein each peripheral controller supplies its own unique interrupt address to any location in Memory. There are five standard interrupt levels (two internal and three external). The third external level, with control lines, can accommodate a virtually unlimited number of vectored interrupts.

1.4.6 Processor Options

Four general options are offered with the ALPHA LSI computer. They are: Power Fail/Restart; the Teletype/CRT Interface; Real Time Clock, and Autoload.

The Power Fail/Restart option mounts directly on the NAKED MINI LSI computer PC board. The other three options mount on an option board which plugs into a special connector (in piggyback fashion) on the NAKED MINI LSI computer PC board. None of these options interface directly with the motherboard.

1. Teletype/CRT Modem Interface.

   Interfaces a modified ASR-33 Teletype, CRT terminal, or modem to the computer. This is a fully-buffered interface that includes remote Teletype motor on/off control. In addition to the standard TTY baud rate (110 baud), nine user selectable baud rates, ranging from 75 to 9600 bauds, are provided for driving a CRT terminal. Either Half or Full-duplex operation is selectable on command.

2. Power Fail/Restart.

   This option includes the hardware necessary to detect low input power conditions and bring the computer to an orderly halt until normal input power is restored. When normal power is restored, this option will generate an orderly restart. The Power Fail/Restart option allows

completely unattended operation of the computer at locations where power conditions are unreliable.

3. Real Time Clock.

The Real Time Clock option features a crystal controlled internal clock which may be wired to produce clock rates of 100 microseconds, 1 millisecond, 10 milliseconds, or twice the input AC line frequency, (8.33 or 10 milliseconds -60 Hz and 50 Hz, respectively). The 10 millisecond (crystal derived) rate is standard. An external clock source may also be used. The Real Time Clock provides time-of-day information to the computer and may be used to time periodic events that must be controlled by the computer.

4. Multi-Device Autoload

The Multi-Device Autoload option consists of a Read-Only Memory (ROM) programmed with a complete binary loader which is capable of loading binary programs from any one of several input devices. The Autoload hardware reads from the ROM when the Console AUTO switch is activated.

1.4.7 Plug-In Options

Locations are provided within the ALPHA LSI computer chassis for the installation of processor options, peripheral interfaces, and memory modules. The options are mounted on printed circuit boards which plug into the locations within the computer chassis. Some of the available plug-in processor options are:

1. Digital I/O interfaces: up to 64 bits.
2. Relay I/O interfaces: up to 32 isolated relays.
3. Modem interfaces: asynchronous and synchronous.
4. Memory Banking controller: extends upper limit of Memory to 262,144 words.
5. Read Only Memory (ROM).
6. Priority Interrupt module.

1.4.8 Peripheral Equipment

The following is a partial list of the various types of peripheral equipment for which interfaces to the ALPHA LSI have been developed. This list does not imply that these are the only devices for which interfaces can be developed. The interface structure of these computers is such that virtually any peripheral device can be interfaced to the computer.

1. ASR-33 Teletypewriter
2. High speed Paper Tape Readers and Punches
3. Line Printers
4. Card Readers
5. Open reel and cassette Magnetic Tape Units
6. Magnetic Disks
7. CRT terminals
8. Communications interfaces

1.5 DATA HANDLING CHARACTERISTICS

1.5.1 Data Word Format

Processor registers and memory locations are capable of storing data words consisting of 16 binary digits or "bits". A word may be handled as a single 16-bit field or as two 8-bit bytes. The following paragraphs describe the word format of the computer. Byte format is described later in this section.

1.5.1.1 Bit Identification

A data word may contain a single number, or it may contain a string of individual binary bits, with each bit having a unique meaning. For purposes of explanation and identification, each bit within a word is uniquely identified. The identification is accomplished by numbering each bit within a word from right to left. The bit on the extreme right

of the word is bit 0, and the bit on the extreme left is bit 15. Figure 1-1 illustrates the format of a 16-bit data word with the bit number shown above the bit position.

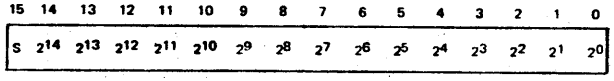


Figure 1-1. Data Word Bit Identification

1.5.1.2 Bit Values

The ALPHA LSI is a binary computer; therefore numeric information stored in the computer and processed by the computer must be in binary format. Figure 1-1 illustrates the binary value of a one- bit (1) in each bit position of the 16-bit data word. These values are expressed as powers of two. For example, a 1 in bit 3 has the value of 2^3 or 8. The single exception to this rule is bit 15 which is the sign bit.

1.5.1.3 Signed Numbers

The ALPHA LSI is capable of performing arithmetic operations with signed numbers. Binary two's complement notation is used to represent and process numeric information. Bit 15 of a data word indicates the algebraic sign of the number contained within that word.

1.5.1.4 Positive Numbers

A positive number is identified by a 0 in bit 15, and the binary equivalent of the magnitude of the positive number is stored in bits 0 to 14. The largest positive signed number which can be stored in a 16-bit word is +32,767 .

1.5.1.5 Negative Numbers

A negative number is identified by a 1 in bit 15 of the data word. A negative number is represented by the binary two's complement of the equivalent positive number. A negative number must follow the mathematical rule where:

$$0 - (+n) = -n$$

For example:

$$0 - (+5) = -5$$

Negative numbers must also be constructed such that:

$$(+n) + (-n) = 0$$

The binary two's complement of some numeric value may be constructed by subtracting the binary representation of the absolute magnitude of that value from 0.

Note that the formation of a binary two's complement negative number from the equivalent positive number automatically sets the sign bit to a one. The largest negative number that can be stored in a 16-bit word is $-32{,}768_{10}$.

1.5.2 Data Byte Format

A 16-bit data word is capable of storing two 8-bit bytes. Since most data transfers between mini computers and peripheral devices are in the form of bytes rather than words, the ALPHA LSI computer provides the capability of addressing individual bytes as well as full data words. Figure 1-2 illustrates the storage of two bytes within one computer word.

Bit positions within bytes are identified much the same as in 16-bit words. Figure 1-2 also illustrates the numbering of data bits within a byte. The bits are numbered 0 through 7, where bit 0 is the least-significant bit (LSB), and bit 7 is the most-significant bit (MSB) of the byte.

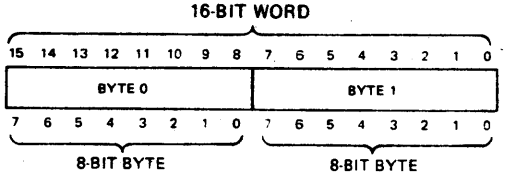


Figure 1-2. Byte Storage, Two Bytes Per Word

1.5.2.1 Byte Mode Processing

There are two control instructions in the computer which control Word mode processing and Byte mode processing. One of the instructions causes the computer to enter Byte mode processing, and the other causes the computer to enter Word mode processing.

In Word mode, all Memory Reference instructions access full words in Memory. In Byte mode, all Memory Reference instructions (except IMS, MPY, DVD, NRM, JMP, and JST) access one byte within a word. The method of addressing individual bytes is discussed in a subsequent part of this section. The present discussion is concerned with computer operations while in Byte mode as contrasted with computer operations in Word mode.

Byte mode affects the address and operand cycles of the computer only. All other computer functions operate the same as in Word mode. In Byte mode, the computer operand cycle reads a single byte from Memory instead of a full word. The following paragraphs illustrate Byte mode operations for Memory Reference instructions.

1.5.2.2 Register Load

In Word mode, the full word is loaded into the selected register. In Byte mode, the selected byte is loaded into the lower eight bits of the selected register and the upper eight bits are set to zero. Note that the location of the byte within the memory word does not determine the location the byte will occupy in the register being loaded.

1.5.2.3 Arithmetic Operations

For arithmetic purposes, bytes are handled as positive numbers only. The reason is that a byte occupies the lower eight bits of a register, or a data bus, and the upper eight bits contain zeros.

1.5.2.4 Data Packing

One of the most useful features of Byte mode processing is in the packing and unpacking of data in Memory. Since most of the peripheral devices used with mini computers are byte oriented, high-speed data transfers between the computer and the peripheral device generally require data to be packed one byte per word. Such an arrangement is illustrated in figure 1-3. In this illustration, the upper eight bits of each data word to be transmitted to a peripheral device contain zeros. A full 16-bit word is transmitted to the device, but the device discards the upper eight bits and accepts only the lower eight bits. Data received from a byte oriented peripheral device during high-speed data transfers is packed in Memory one byte per word in the same format described previously (figure 1-3). If a software subroutine were required to pack the data two bytes per word, in the format illustrated in figure 1-4, it would waste memory space and time in performing the formatting required for high-speed data transfers.

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

Figure 1-3. Data in Memory, One Byte Per Word

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

Figure 1-4. Data in Memory, Two Bytes Per Word

The capability of the ALPHA LSI computer to address individual bytes in Memory allows high-speed data transfers using the memory format shown in figure 1-4 for both transmission and reception of data. Bytes may be addressed sequentially and transmitted or received sequentially, just as words are transmitted or received sequentially in conventional unpacked data transfers. This arrangement saves memory space since none of the memory word is wasted, and it saves time since no software routines are required to pack and unpack data for internal processing.

1.5.3 Memory Address Formats

Maximum memory capacity (exclusive of Memory Banking control) in the ALPHA LSI computer is 32,768 words which means a byte capacity of 65,536 bytes. A fifteen bit address is required to address 65,536 bytes. The following paragraphs discuss the formats of the addresses that must be presented to Memory for addressing both words and bytes. This discussion is concerned only with address formats. Section 3 of this manual discusses the memory address modes which form these addresses.

1.5.3.1 Word Addressing

Figure 1-5 illustrates the format of an address presented to Memory to address a full word. This is the format that is used to address instructions or full data words. The address is contained in bits 0 - 14, and bit 15 contains a zero.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WORD ADDRESS: 15 BITS														

Figure 1-5. Basic Word Address Format

1.5.3.2 Byte Addressing

Figure 1-6 illustrates the format used to address a byte within a data word. Bits 1-15 contain the address of the memory word, and bit 0 specifies which byte within the word is to be addressed.

Bit 0 = 0 specifies Byte 0 (Most Significant Byte).

Bit 0 = 1 specifies Byte 1 (Least Significant Byte).

If the computer is set for Byte mode, all operand addresses presented to Memory are assumed to be byte addresses. The computer assumes that the address is in the format shown in figure 1-6. If the computer is set for Word mode processing, all addresses presented to Memory are assumed to be word addresses in the format shown in figure 1-5. These assumptions apply to operand cycles only. They do not apply to instruction cycles or indirect addressing cycles.

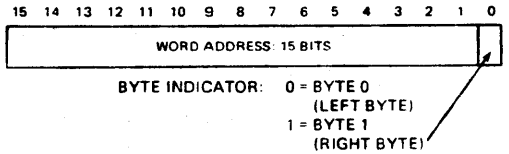


Figure 1-6. Byte Address Format

1.5.3.3 Indirect Addressing

The ALPHA LSI computer is capable of performing single level indirect addressing for addressing bytes, and multi-level indirect addressing for addressing words. Indirect addressing uses direct addressing to read a word in Memory, called an address pointer, which contains the address of another word. In Byte mode, the address pointer contains the address of the byte to be addressed. The format of the address in the address pointer is the same as that shown in figure 1-6.

In Word mode, the format of the address in the address pointer is that shown in figure 1-7. Bits 0 - 14 contain the address of another word in Memory. Bit 15 is a multi-level indicator. If bit 15 contains a 1, the address in bits 0 - 14 is the address of another indirect address pointer. The number of levels of indirect addressing which may be used is limited only by memory size.

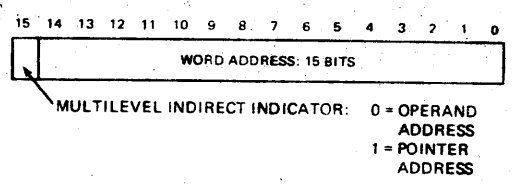


Figure 1-7. Indirect Address Pointer Format

Section 2

INTEGRATION

2.1 INTRODUCTION

This section provides detailed information pertaining to the mounting, cooling, and interconnection of either the ALPHA LSI or NAKED MINI LSI-1 and -2 computers.

2.2 ALPHA LSI INTEGRATION

The following paragraphs discuss mounting, cooling, installation of PC boards, and ac power application for the ALPHA LSI computer.

2.2.1 Mounting (Figure 2-1)

The ALPHA LSI computer is designed to be mounted in a standard 19-inch rack or cabinet. Figure 2-1 provides outline and mounting dimensions to facilitate installation of the computer.

2.2.2 Cooling (Figure 2-2)

The ALPHA LSI Computer is designed to operate over a temperature range of 0° C to 50° C. When the computer is installed in an enclosure, the installation requirements depend on the ventilating system employed such that the thermal requirements of the computer are maintained.

There are three installation criteria which provide the minimum cooling conditions allowable for the ALPHA LSI computer.

1. Closed Ventilation System
2. Side Ventilation System
3. Top/Bottom Ventilation System

In the closed ventilating system, it is assumed the ambient temperature will be maintained by the thermal interface. The minimum size enclosure must provide adequate air flow paths for the computer's internal fans.

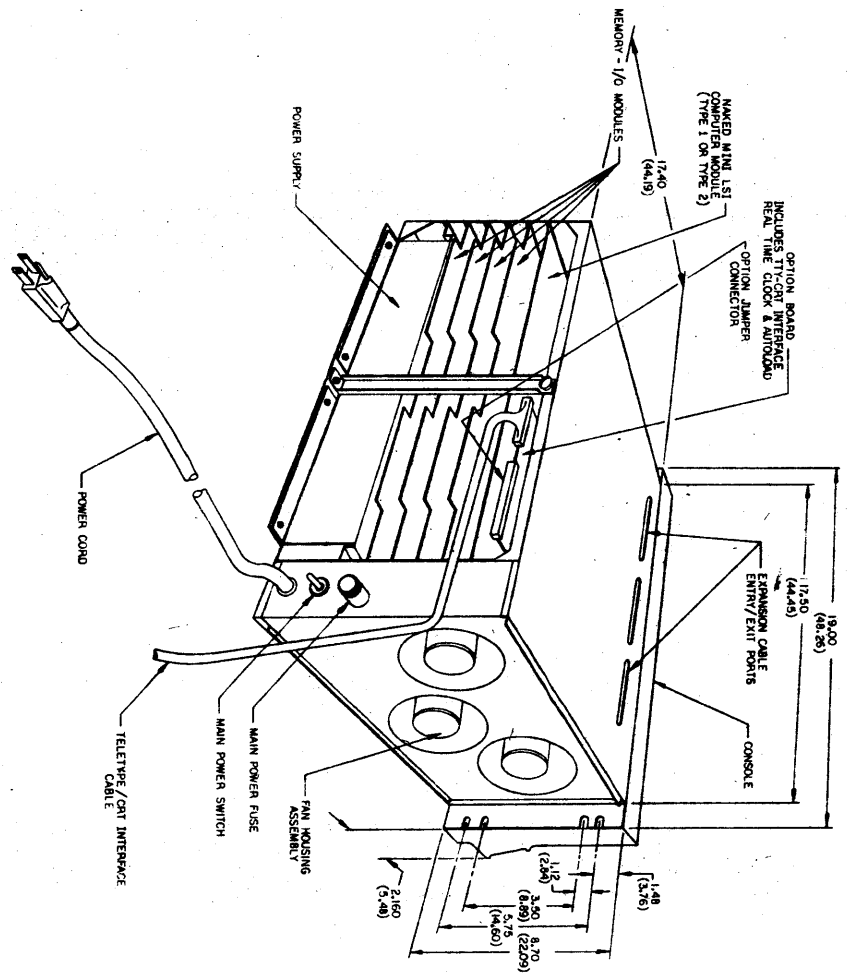


Figure 2-1. ALPHA LSI Outline and Mounting Diagram

Top/Bottom Vent System

Side Vent System

Closed Vent System

Figure 2-2. ALPHA LSI Ventilating Systems

ENCLOSURE TABULATION						
AIR FLOW DIRECTION	VENTILATION SYSTEM	"A"	"B"	"C"	"D"	"E" Sq. In.
A/F#1	CLOSED	12.0	---	---	---	---
A/F#2	TOP/BOTTOM	8.75	---	---	5.0	44.0
A/F#3	SIDE	8.75	7.0	16.0	---	112.0

The side ventilating system establishes the minimum enclosure size and rectangular surface for the minimum size opening. This provides for a safety guard if necessary.

The top/bottom ventilating system defines the minimum airflow paths for a cabinet with stacked equipment or an individual console enclosure.

Figure 2-2 defines the minimal dimension parameters for each of these ventilating systems.

2.2.3 Joining Two Half PC Boards

Most I/O modules occupy only half a PC board slot in the computer. When several half board options are used, it is recommended that half boards be joined together to form full boards. In those cases where an odd multiple of half board I/O modules is used, a blank Filler PC board is available from Computer Automation, Inc. to join with the last half board. (Refer to section 12 for further details on the Filler PC board, Part No. 10053-00).

Half board modules are joined together by means of a stiffener kit which is supplied with each half board module (CAI part no. 95-20389-00). Each stiffener kit consists of the following parts:

1. Two 14-inch stiffener bars
2. Twelve 4-40 x .500 inch nylon screws
3. One nylon board extractor with roll pin
4. One interface connector

When joining two half boards together, two stiffener kits are required.

The stiffener bars are installed on the component side of each printed circuit board. One stiffener bar is located parallel to the computer interface contacts on each module. Another stiffener bar is located at the back edge of each module parallel to the peripheral interface contact strips. Finally, two stiffener bars (one for each module) are located on the adjacent edges of each module (what would be the center of a full PC board).

Stiffener bars are installed in the following manner:

1. First determine the physical placement of the module in the computer, that is, the relative placement of the module with regard to the priority string.

2. Next, install the center stiffener bars. The nylon screw is inserted through from the solder side of the board. Tighten the screws.

3. Install a stiffener bar on the front and rear edges of both modules. Do not tighten the screws.

4. Next, find a level work surface. Stand both modules in a vertical position with the front edge down. Ensure that the contact edge of each module is touching the table surface and that the modules are butted together. Tighten the nylon screws on the front edge. Now tighten the screws on the back edge.

5. Finally, examine the board extractors on one of the processor boards in the computer. Find the similar extractor mounting holes on each module. Mount the extractor on each side of the module and insert the roll pin.

This completes the joining operation. The PC board is now ready to install in the computer. When all boards are installed, be sure to install the board retainer at the rear of the computer.

2.2.4 Option Board Installation

The Option PC board (option board) mounts in piggyback fashion to the left half (as viewed from the front) of either the LSI-1 or LSI-2 processor module. Support standoffs are provided with the processor modules. All loose hardware (screws, lockwashers, washers and rear-edge connectors) is provided with the option board.

The option board has three edge connectors. Connector P1 interfaces with J1 on the processor module. Connector J1 is the option jumper connector and connector J2 is the Teletype interface connector. Detailed information about the use of connectors J1 and J2 is provided in section 6 of this manual.

To install the option board, proceed as follows:

1. Take the option board and insert J1 and J2 through the slots in the rear stiffener of the processor module.

2. Position connector P1 for insertion into connector J1 on the processor module.

3. Gently push the option board into processor connector J1 aligning the four mounting holes with the processor module standoffs.

4. Install a screw, lockwasher, and washer in each standoff and tighten.

5. Install rear edge connectors per instructions in section 6.

2.2.5 Module Installation, Processor Chassis Only

CAUTION

Do not remove or install any PC boards or cables while power is applied to the computer.

The ALPHA LSI motherboard slot organization is shown in figure 2-3. All modules, except the processor module which is restricted to the top slot (slot A), can be placed in any location within the processor chassis. In the placement of these modules, however, consideration must be given to priority chains. These priority chains, namely Interrupt, DMA, and Memory Banking, must be maintained. DMA and memory modules provide for the propagation of all priorities. The I/O modules provide for Interrupt priority, but may not provide for DMA and Memory Banking. If I/O modules are placed above DMA or memory modules, the priority input and output pins relating to DMA and Memory Banking must be jumpered. The priority input and output pins for DMA and Memory Banking are given in the chart below.

	PRIORITY IN		PRIORITY OUT	
	MNEMONIC	PIN	MNEMONIC	PIN
DMA	DPIN-	209	DPOT-	210
Memory Banking	MBIN	237	MBOT	238

NOTE

Some I/O modules have the priority input and output pins brought out to plated holes to facilitate jumpering. If these plated holes are not provided, the jumpers should be soldered directly to the connector pin etch.

Interrupt priority is determined by physical location of the interface module within the chassis. The priority line begins with slot B200 and weaves through the motherboard as shown in figure 2-3. It is routed through each I/O controller so it can inhibit the lower priority devices when requesting service. Therefore, all I/O modules must be placed in consecutive priority level slots to provide continuity in the priority chain. If the priority chain is broken, down-stream interrupts may not be serviced. If they are serviced, they will be serviced improperly.

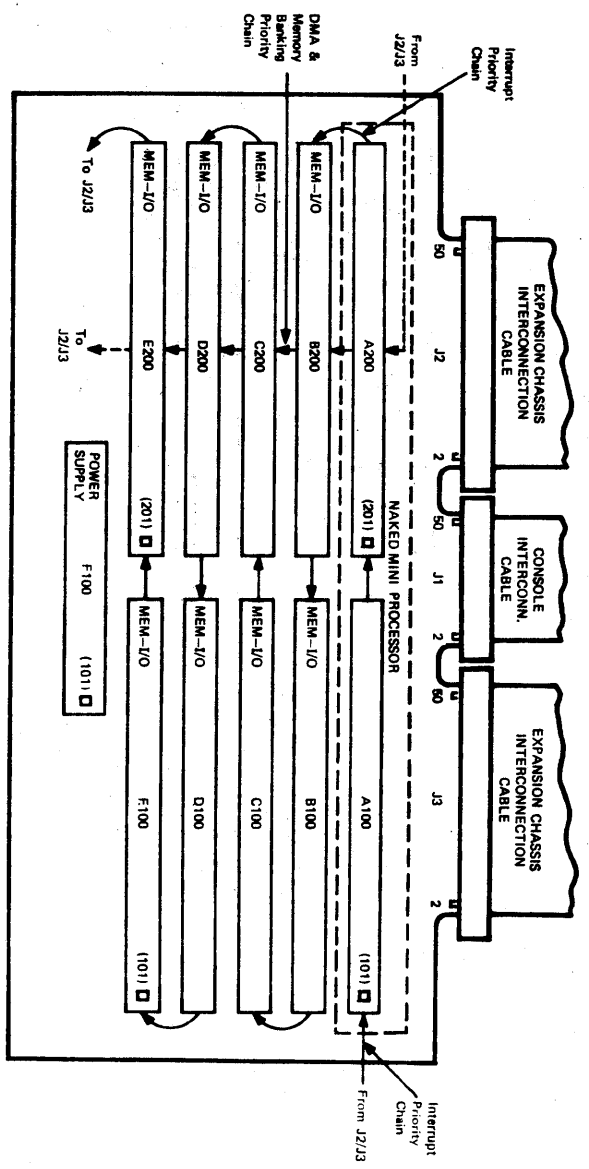
As with Interrupt priority, DMA priority is determined by the physical location of the DMA controller. The DMA priority chain runs down the 200-series side connectors only, the highest priority being in slot B200 and the lowest in slot B200. Half board DMA controllers must be installed in 200-series connectors only.

The Memory Banking chain runs down the 200-series side connectors only. If half board memory modules are used, they must be installed in 200-series connectors only.

If no specific module placement scheme is required, the general rules below may be applied to facilitate module installation. If these rules are followed, no particular problems should occur.

NOTE

Install all modules with component side up.



(Rear View)

Figure 2-3. Motherboard Priority String

1. Install processor module in the top slot (slot A).

2. Install memory modules next. The various sizes and types of memory modules can be intermixed in any order. (Refer to section 7 for Memory Interleaving and Banking information.) Half board memory modules must be installed in 200-series connectors only.

3. Install DMA controllers after memory modules.

4. Install I/O modules last.

Documentation is provided for each type of I/O interface module. This document defines the software and cabling requirements of the interface module. Refer to the appropriate interface description to resolve any questions about the interface module.

CAUTION

All I/O interface modules must have the rear-edge cable connector installed prior to operation of the Processor. If the connector is not installed, a default device address of zero will be assigned to the module, causing improper instruction execution. Device address zero is reserved exclusively for Processor use. For details concerning assignment of a unique device address to each I/O interface module, refer to the associated interface description which is packed with each module.

2.2.6 Expansion (Figure 2-4)

In the event insufficient slots are provided in the processor chassis for a given application, the Maxi-Bus may be expanded via one or more expansion chassis. The expansion chassis is identical to the processor chassis (same motherboard, etc.) but includes a Buffer PC board (buffer board) to regenerate Maxi-Bus signals, and also ribbon cables of the appropriate length for interconnecting between chassis. (The length of the cables depends upon whether the interconnection is from the processor chassis to the first expansion chassis, or between subsequent expansion chassis.)

To facilitate the computer system expansion, Maxi-Bus expansion connectors J2 and J3 are provided on the motherboard immediately above slot A. (Refer to figure 8-11 for the pin assignments of connectors J2 and J3.) Connectors J2 and J3 are connected to buffer board connectors J2 and J1, respectively, in the first expansion chassis. If further expansion is required, connectors J4 and J3 at the bottom of the buffer board are connected to J2 and J1, respectively, of the next buffer board. The interconnect cables should be routed through slots located at the front, bottom and top of each chassis.

The Buffer PC board is mounted with the component side facing the expansion chassis motherboard. Emanating from the center of the component side of each buffer board are two ribbon cables (W1 and W2). When facing the front of the chassis, the cable on the right, W2, interfaces with connector J2 on the expansion chassis motherboard. The cable on the left, W1, interfaces with connector J3.

Expansion may extend to a maximum of three chassis. As expansion chassis are installed, a speed degradation will occur. Memory modules located in expansion chassis will exhibit an apparent slower system access and cycle time (200 ns for each expansion chassis). Similarly, I/O modules located in a second expansion chassis or beyond may require that the processor timing circuit be altered to provide additional phase stretching during I/O operations (refer to paragraph 6.6.5). (A minimum I/O stretch period of 100 ns is recommended for each "subsequent" expansion chassis beyond the "first" expansion chassis.) This timing circuit is modified simply by changing an option-jumper connector which configures all jumper-controlled processor options in the machine. This option-jumper connector mounts to the rear-edge of the processor option board. Note that whenever any stretch is inserted, all I/O timing throughout the system is slowed down by the stretch period.

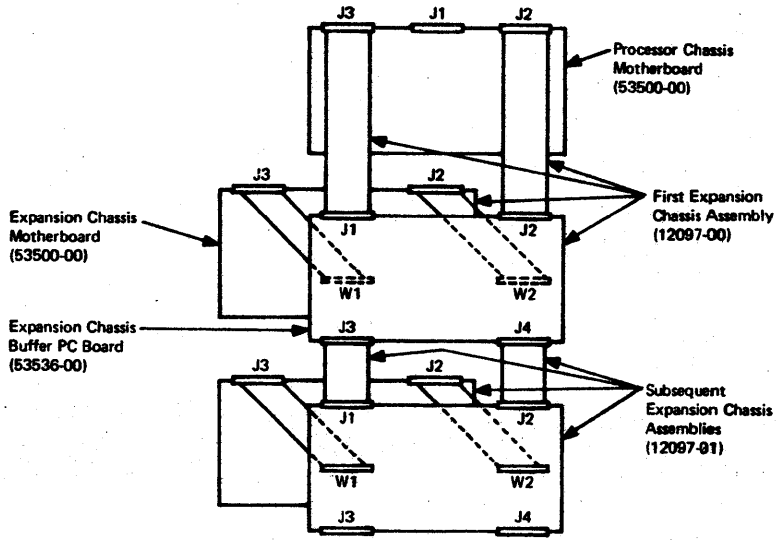


Figure 2-4. Expansion Chassis Cabling Scheme

2.2.6.1 Module Installation, Processor and Expansion Chassis

In general, the processor chassis module installation rules described in paragraph 8.2.5 (referring to priority chains, placement of half board DMA and memory modules, etc.) are also applicable to module installation in expansion chassis. In addition to these general rules, the following rule applicable to installation of DMA controllers in expansion chassis must be adhered to.

> DMA controllers cannot communicate with memory or I/O modules located in up-stream chassis. They can, however, communicate with these modules if they are installed in any slot within the same chassis, or within any down-stream chassis.

NOTE

> Expansion chassis must be installed below the processor chassis.

If no specific module placement scheme is required, the general rules below may be applied to facilitate module installation in the processor and expansion chassis.

1. Install the processor module in slot A of the processor chassis.
2. Install DMA controllers immediately below the processor module.

NOTE

> If the LSI-1 Processor is being used and a DMA module is in an expansion chassis, it will not be able to communicate with the Memory on the LSI-1 Processor board. For this reason, another memory module with which the DMA controller can communicate must be placed within the same chassis, or a chassis down-stream from the DMA controller.

3. Install all memory modules next.
4. Install all I/O modules last.

2.2.7 AC Power Application

Computers intended for use with 110 Vac are shipped with a line cord containing a standard 3-prong ac plug. Computers intended for use with 220/240 Vac are shipped with a line cord, but without a plug due to the various plug configurations possible when using 220/240 Vac. In these instances, the customer must install an appropriate ac plug. Color coding for the wires contained in the ac line cord are as follows:

Black	Hot line-fused
White	Neutral line-unfused
Green	Ground

Before plugging the ac line cord into a power source, be sure that the main power switch, located on the back of the chassis, is in the OFF position. Plug the ac line cord into the power source.

CAUTION

Connect ac line cord to properly grounded 3-prong receptacle only.

NOTE

When ac power is applied, the fans will operate when the main power switch is in the ON position. Ensure that they are operating.

2.2.8 110 to 220/240 Power Line Conversion

The ALPHA LSI computer may be powered from either 110 Vac or 220/240 Vac. To convert from 110 to 220/240, or 220/240 to 110, follow the procedure outline below and perform the appropriate step 4 for the conversion desired. Step 4a is for converting from 110 to 220/240 and step 4b is for converting from 220/240 to 110.

Step 1 Turn power off and remove line cord from ac power source.

Step 2 Remove Console from front of chassis.

Step 3 Disconnect ac power connector P1 from the power supply. Power connector P1 is connected to the power supply through an opening in the motherboard.

Step 4a 110 Vac to 220/240 Vac

Step 4a1 Using a Molex removal tool, remove pin 3 from power connector P1. (The pins are numbered on the wiring side of the connector.) Insulate the pin with a piece of electrical tape and tie back to cable.

Step 4a2 Remove pin 6 from power connector P1 and insert in pin 3 of P1.

Step 4a3 Install a 220/240 Vac plug on the line cord.

Step 4a4 Change line fuse from 7A, 125 V to 3A, 250 V.

Step 4a5 Proceed to step 5.

Step 4b 220/240 Vac to 110 Vac

Step 4b1 Using a Molex removal tool, remove pin 3 from power connector P1 and insert in pin 6 of P1. (The pins are numbered on the wiring side of the connector.)

Step 4b2 Take the pin which is tied back to the power cable (contains a blue and a black wire) and insert in pin 3 of P1.

Step 4b3 Install a 110 Vac plug on the line cord.

Step 4b4 Change line fuse from 3A to 7A.

Step 5 Reconnect power connector P1 to the power supply.

Step 6 Install the Console.

Step 7 Connect the line cord to the appropriate source of ac power. Then turn power on and test the computer.

2.3 NAKED MINI LSI INTEGRATION

The following paragraphs discuss mounting, cooling and interconnection of the NAKED MINI LSI-1 and -2 computers.

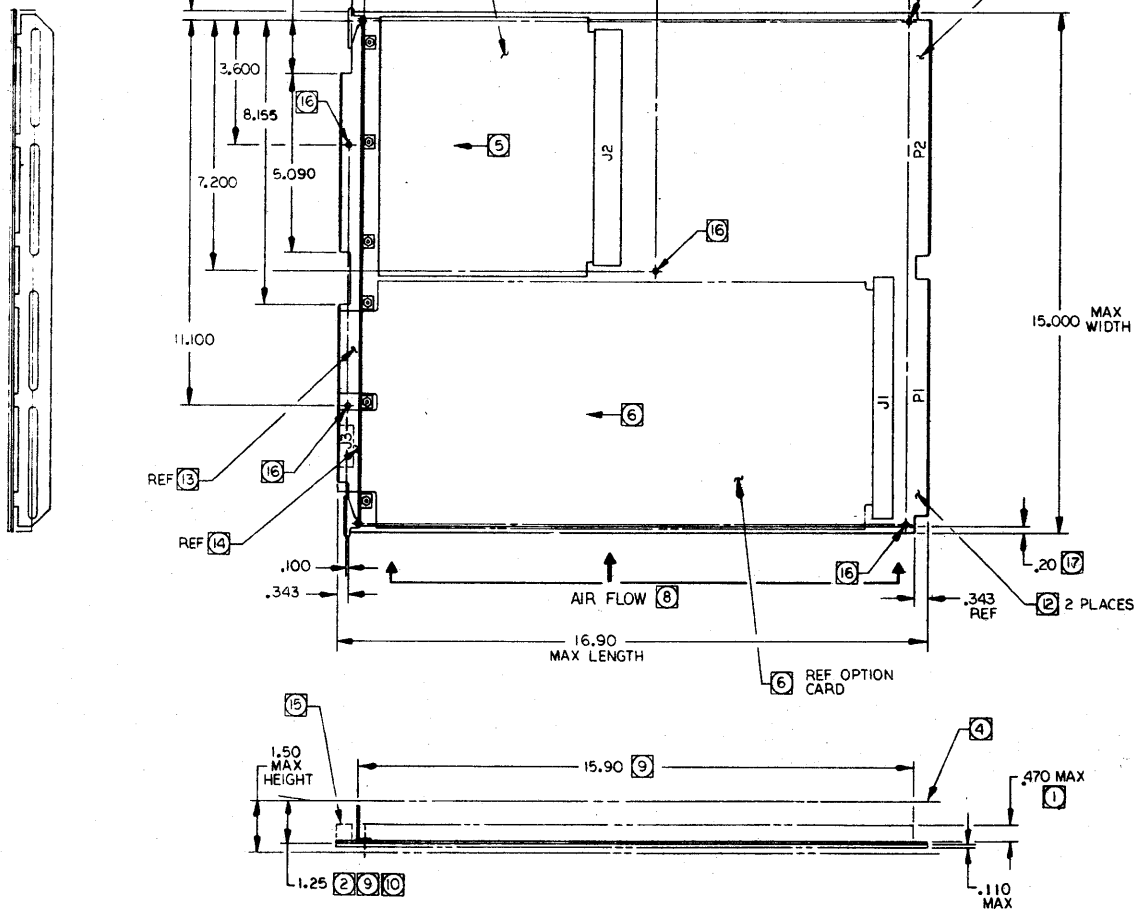
2.3.1 Mounting

There are two mounting considerations: one for LSI-1 and one for LSI-2.

2.3.1.1 LSI-1 Mounting Considerations (Figure 2-5)

The LSI-1 computer may be mounted in any plane as long as the cooling requirements are satisfied. The computer may be hard mounted with mobile or fixed interface connector or slide mounted with fixed interface connectors.

Five mounting holes are provided for hard mounting. Two holes are at the front of the module near the corners, two are at the back of the module and one hole is located in the center of the module. It is recommended that standoffs be used when hard mounting the computer.



(17) THE COMPUTER MAY BE MOUNTED UTILIZING SLIDE IN RAILS. THE AREA PROVIDED AT THE EDGE OF EACH SIDE OF THE MAIN CARD IS FREE OF ETCH AND COMPONENTS TO DIMENSION INDICATED.

(16) THE COMPUTER MAY BE HARD MOUNTED UTILIZING THESE FIVE (5) MOUNTING HOLES.

(15) MATES WITH CONNECTOR (SPECTRA STRIP P/N SS800-034) OR EQUIVALENT.

(4) MATES WITH CONNECTOR (VIKING P/N 3VH25/1JN-5) OR EQUIVALENT.

(13) MATES WITH CONNECTOR (WINCHESTER P/N 8BDJ185) OR EQUIVALENT.

(12) MATES WITH CONNECTOR (VIKING P/N 2VK43D/1-12) OR EQUIVALENT.

11. THE NAKED MINI ALPHA LSI COMPUTER SHALL BE KEPT FREE OF EXCESSIVE FORIGN MATERIAL (OIL, DUST, SALTS, ETC).

(10) OBSTRUCTIONS AND AIR LEAKS SHALL BE ALLOWABLE TO THE EXTENT THAT A MINIMUM OF 150 FPM OF AIR IS EXHAUSTED FROM ANY PART ACROSS THE OUTPUT SIDE OF THE COMPUTER MAIN CARD AS INDICATED.

(9) AIR SHALL BE SUPPLIED AT THE VOLUME OF 20 CFM MINIMUM WITH A MAXIMUM PRESSURE DROP OF .2 INCHES OF WATER THROUGH AN AIR CORRIDOR AS INDICATED.

(8) AIR FLOW SHALL BE IN THE INDICATED DIRECTION ONLY.

7. THE NAKED MINI ALPHA LSI COMPUTER MAY BE MOUNTED IN ANY PLANE PROVIDING NOTES 8,9 & 10 ARE ADHERED TOO.

(6) THE OPTION CARD IS ACCESSABLE BY THE REMOVAL OF FOUR (4) #4 SCREWS AND PULLING THE CARD FROM THE CONNECTOR IN THE DIRECTION SHOWN.

(5) THE MEMORY CARD IS ACCESSABLE BY THE REMOVAL OF TWO (2) #4 SCREWS AND PULLING THE CARD FROM THE CONNECTOR IN THE DIRECTION SHOWN.

(4) THE MEMORY AND OPTION CARD ARE ACCESSABLE FROM THE SURFACE INDICATED.

(3) CLEAR AREA (ETCH OR FEED THRUS) .350 DIA MINIMUM AROUND MOUNTING HOLES BOTH SIDES.

(2) MAXIMUM HEIGHT OF MEMORY CARD & OPTION CARD (OPTION CARD NOT SHOWN).

(1) MAXIMUM COMPONENT HEIGHT OF MAIN CARD.

NOTES: UNLESS OTHERWISE SPECIFIED

Figure 2-5. NAKED MINI LSI-1 Outline and Mounting Diagram

For slide mounting, a clear area of 0.200 inch is provided along each side of the module to accommodate various types of PC board guides. The PC board guide should be able to handle a PC board thickness of 0.062 inch. The LSI-1 computer module should be supported along all four edges. The interface connectors along the front of the module should be hard mounted to the users structure and some type of support should be provided at the rear of the module.

2.3.1.2 LSI-2 Mounting

The LSI-2 is mounted in the same manner as the ALPHA LSI. Refer to paragraph 2.2.1 and figure 2-1.

2.3.2 Cooling

The cooling requirements for the LSI-1 and LSI-2 are discussed below.

2.3.2.1 LSI-1 Cooling

The LSI-1 computer is designed to operate over a temperature range of 0° C to 50° C. Cooling air must flow from the processor side of the module to the memory side of the module. Notes 8, 9, and 10 of figure 2-5 must be adhered to.

2.3.2.2 LSI-2 Cooling

The LSI-2 chassis has a fan housing with three fans. These fans provide adequate cooling for the computer.

2.3.3 Interconnection

The interconnection requirements of the LSI-1 and LSI-2 are discussed below.

2.3.3.1 NAKED MINI LSI-1 Interconnection

The LSI-1 interconnections consist of bringing power to the module, strapping all of the signals from P1 to P2 (with the exception listed below), and interfacing the system control console to P1.

There are ten special signals that interface with the P1 connector that are not part of the Maxi-Bus. Eight of these signals are dedicated console interface signals while the other two are dedicated power supply signals. Under no circumstances should these signals be strapped across to the P2 connector. These dedicated signals and their pin assignments are listed below.

Signal	Pin	Dedicated to
SSW-	P1-9	Console
IF-	P1-10	Console
TTLF-	P1-11	Power Supply
+5H	P1-12	Power Supply
AL-	P1-33	Console
BM-	P1-34	Console
OV-	P1-37	Console
START-	P1-38	Console
SERV-	P1-83	Console
CINT-	P1-84	Console

Table 8-2 lists Maxi-Bus and power signals, along with associated pin assignments.

2.3.3.2 NAKED MINI LSI-2 Interconnections

All LSI-2 interconnections are made at the motherboard. Motherboard connector J1 provides the console interface while connector F100 provides the power interface. Console interface information is available in section 10 while power supply interface information is available in section 11.

To convert the LSI-2 from 110 Vac to 220/240 Vac, refer to paragraph 2.2.8.

NOTE

The NAKED MINI LSI-2 consists of a processor module, memory module(s), chassis, motherboard and fan housing. In addition to dc power, the user must provide fan power of 110 Vac at 0.6 amps to pin 1 and 2 of connector P1 of the fan housing.

Section 3

CONSOLES

3.1 PROGRAMMING CONSOLE

The ALPHA LSI Programming Console provides the switches and indicators required to operate, display and control the computer. This section describes the controls and indicators on the Console, provides operating procedures, and defines machine modes.

3.1.1 Switches and Indicators

For the convenience of the user, the switches and indicators have been grouped into the following sections:

1. Status
2. Control
3. Entry and Display

Figure 3-1 illustrates the ALPHA LSI Console. All console switches, except the Console Enable switch, are momentary contact touch switches and all indicators are light-emitting diodes (LED's). The switches and indicators are listed and explained in table 3-1.

NOTE

Due to the momentary contact nature of the Console switches, the information entered via these switches is volatile since it is stored electrically rather than mechanically. The information will be lost during a power outage. All pertinent information can be restored, however, upon power resumption through use of the Power Fail/Restart option and appropriate software to restore the Status word. (Refer to Power Fail/Restart, section 6, and Status Control instructions, section 4.)

Table 3-1. Console Switches and Indicators

SWITCH OR INDICATOR	PURPOSE
System Status Section	
ON Indicator	On when power is applied, off when power is removed. The main power switch is located on the rear of the computer.
ENABLE Slide Switch and Indicator	The console enable/disable slide switch is located in a recess on the edge of the console. When the switch is on, the ENABLE indicator is on. Likewise, when the switch is off the indicator is off. When in the ENABLE state, all switches and indicators are enabled. When in the disabled state, the only functions that are effective are: 1. The SENSE switch and indicator. 2. The console sense register, console sense register display, hex entry keyboard for the console sense register, console interrupt, and interrupt indicator.
BYTE Indicator	On when the Processor is in Byte mode. Off when the Processor is in Word mode.
OV Indicator	On when the Processor Overflow flag is on. Off when the Overflow flag is off.
SENSE Switch and Indicator	The SENSE Switch toggles the SENSE indicator. The SENSE indicator may be tested by program instructions. The Sense test will be true if the SENSE indicator is on.
System Control Section	
STOP Switch and Indicator	The STOP switch toggles the STOP indicator. The indicator is on when the Stop mode is established. When the indicator is off the Run Enable mode is established. When the Stop mode is established and the Console is enabled (ENABLE indicator on), data entry and display operations may be performed. In addition, the Processor will fetch and execute one program instruction each time the RUN switch is pressed. When in the Run Enable mode, data entry and display operations may not be performed. The Run mode is enabled but not entered until the RUN switch is pressed.

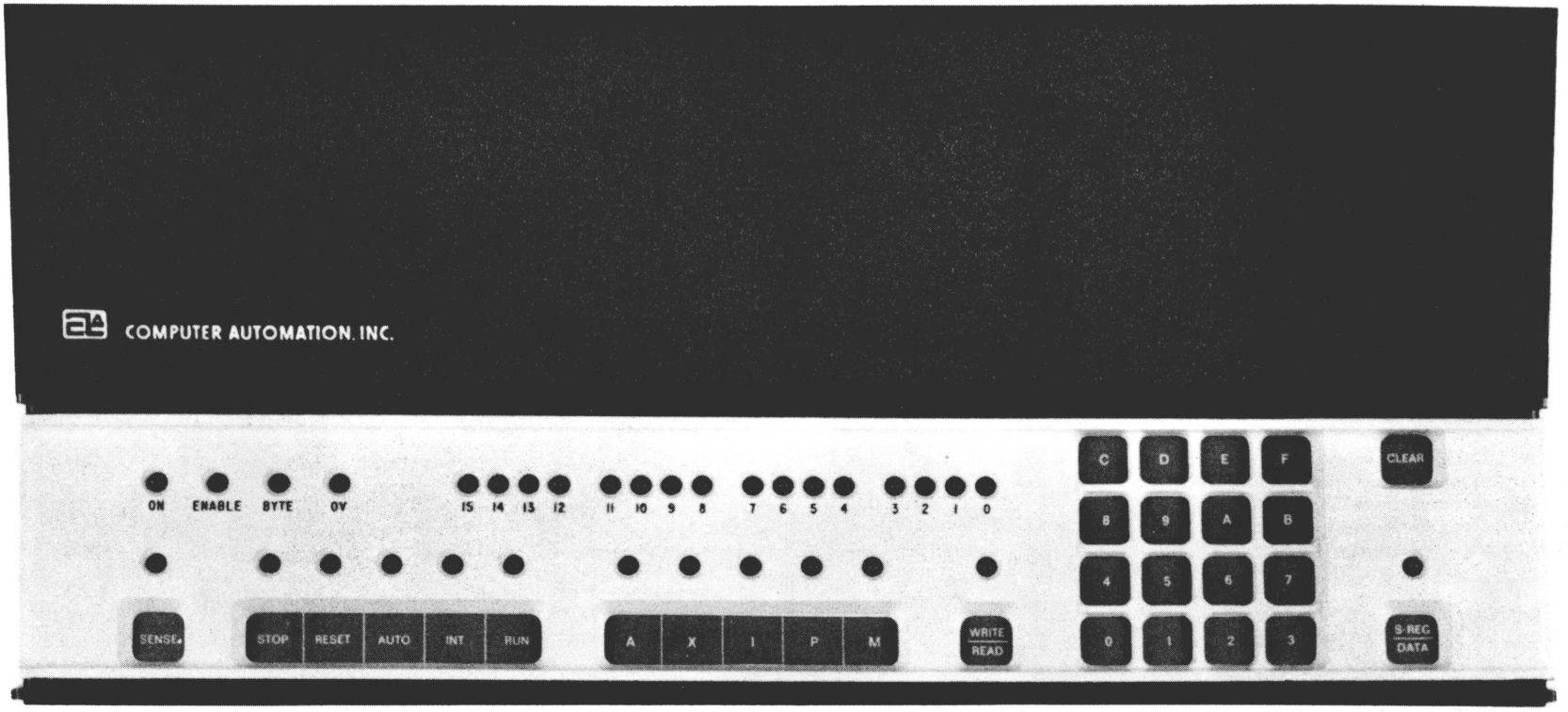


Figure 3-1. ALPHA LSI Console

Table 3-1. Console Switches and Indicators (Cont'd)

SWITCH OR INDICATOR	PURPOSE
RESET Switch and Indicator	The indicator is on when the RESET switch is on and remains on only as long as the switch is pressed. The RESET switch generates a system reset signal which causes the Processor and all interfaces to be initialized. The RESET switch should not normally be used to stop the computer. If RESET is pressed while the computer is running, the instruction currently being executed may not complete. The STOP switch should normally be used to halt the computer. The only time that RESET should be used to halt the computer is in the case where the Processor is hung up in a non-escapable one instruction loop (e.g., multi-level indirect address instruction with closed address chain). The RESET switch should not be used after entering data via the Console or any flags and indicators turned on during data entry will be turned off.
AUTO Switch and Indicator	The AUTO switch is used to initiate an Autoload sequence if the Autoload option is installed. The AUTO switch is enabled only during the Run Enable mode. Depressing the Switch establishes the Run mode and initiates the Autoload sequence. The indicator turns on when the switch is pressed and remains on until the Autoload sequence is completed. With no Autoload option installed, depression of AUTO will still cause the processor to run starting at location :0000. However, no loading occurs.
INT Switch and Indicator	The INT switch is used to initiate a Console interrupt. The switch is enabled only during the Run mode. The indicator turns on when the switch is pressed and remains on until the Processor honors the Console interrupt request.
RUN Switch and Indicator	The RUN switch is used to establish the Run mode when the STOP indicator is off. When the STOP indicator is on, the RUN switch causes one instruction to be fetched and executed when pressed. The WRITE/READ and register indicators (A,X,I,P and M) are turned off whenever RUN is pressed. The RUN indicator is turned on when in the Run mode.
Entry/Display Section	
Register Display Indicators (0 thru 15)	The 16 Register Display indicators display the contents of either the Console Data register or the Console Sense

Table 3-1. Console Switches and Indicators (Cont'd)

SWITCH OR INDICATOR	PURPOSE
	register depending on the state of the S REG/DATA indicator. When the S REG/DATA indicator is off, the contents of the Console Data register are displayed. The Console Data register contains either: 1) the most recent contents of the A, X, I or P register or Memory as requested by the Register Select switches; 2) the last processor output to the Console Data register; or 3) the last keyboard entry to the Console Data register. When the S REG/DATA indicator is on, the contents of the 4-bit Console Sense register are displayed on the Register Display indicators. The Console Sense register contains either the last keyboard entry to the sense register or the last processor output via the Status Output command. The upper 12 Register Display indicators are turned off when displaying the Console Sense register.
Register Select Switches and Indicators (A, X, I, P and M)	The five Register Select switches determine which one of four processor registers or memory data is to be involved in a read/write operation. Each switch has a corresponding indicator which turns on when a given switch is pressed. The indicators are interlocked such that only one indicator is on at a time. The A, X, I and P switches cause a transfer to occur between the target register and the Console Data register. The M switch causes a transfer between the addressed memory location addressed by P Register and Console Data register to occur and also causes the P counter to increment after the transfer. This feature permits manual scanning or loading of sequential memory locations by repeated pressing of the M switch.
WRITE/READ Switch and Indicator	The WRITE/READ switch is used in conjunction with the Register Select switches. When the WRITE/READ indicator is on, the contents of the Console Data register will be written into the target register or addressed memory location when the appropriate Register Select switch is pressed. When the WRITE/READ indicator is off, the contents of the selected register or addressed

Table 3-1. Console Switches and Indicators (Cont'd)

SWITCH OR INDICATOR	PURPOSE
	memory location are copied into the Console Data register and displayed.
Hexadecimal Entry Keyboard (0 thru F)	The Hexadecimal Entry Keyboard consists of 16 switches which are used to enter data into either the 16-bit Console Data register or the 4-bit Console Sense register as determined by the S REG/DATA switch and indicator. When the S REG/DATA indicator is off, each depression of a key causes a corresponding 4-bit binary hex code to be entered into the four least-significant bits (LSB's) of the Console Data register with the previously entered data shifted four places to the left. The Console Data register will be statically displayed as long as the S REG/DATA indicator is off and the computer program does not alter the contents of the Console Data register. When the S REG/DATA indicator is turned on, each depression of a hex entry key causes the corresponding binary hex code to be entered into the four-bit Console Sense register. The Console Sense register is statically displayed in the four least significant Register Display indicators so long as S REG/DATA is in the on state and the computer program does not modify the contents of the Console Sense register. The upper 12 Register Display indicators are extinguished.
S REG/DATA Switch and Indicator	The S REG/DATA switch toggles the S REG/DATA indicator which determines whether the Console Data register or the Console Sense register is to be connected to the hex entry keyboard and the Register Display indicators. If the S REG/DATA indicator is off, the hex entry keyboard is used to enter data into the Console Data register and the Register Display indicators are connected to the Console Data register. If the S REG/DATA indicator is on, the keyboard and display are connected to the Console Sense register.
CLEAR Switch	The CLEAR switch, when pressed, clears data from the Console Data register. The switch does not affect the Console Sense register.

3.1.2 Machine Modes

There are four machine modes which are controlled from the Console. These modes are:

1. Stop Mode
2. Step Mode
3. Run Enable Mode
4. Run Mode

Mode selection is made by use of the RUN and STOP switches. The RUN and STOP indicators define the current machine mode as follows:

STOP	RUN	MODE
on	off	Stop
on	on	Step
off	off	Run Enable
off	on	Run

3.1.2.1 Stop Mode

The Stop mode unconditionally halts program execution and enables the Entry and Display section of the Console. The Stop mode is manually entered from either the Run mode or the Run Enable mode when the STOP switch is pressed. While in the Stop mode, the Entry and Display section of the Console is enabled.

3.1.2.2 Step Mode

The Step mode is a transient condition in which a single instruction is executed. The Stop mode is re-entered upon completion of the instruction. A single instruction is executed each time the RUN switch is pressed while the STOP indicator is on. Interrupts are not serviced while in Step mode.

3.1.2.3 Run Enable Mode

The Run Enable mode is an intermediate mode between the Stop and Run modes. Either the Run or Stop mode may be entered from the Run Enable mode. Conversely, the Run Enable mode can be entered from the Run mode by execution of a programmed halt. The Run Enable mode can be entered from the Stop mode by turning off the STOP indicator. While in the Run Enable mode, the Entry and Display section of the Console is disabled.

3.1.2.4 Run Mode

The Run mode can be entered only from the Run Enable mode. When entered, the Run mode permits the user's program to execute. The Run mode can be established manually from the Console; semi-automatically by means of the Autoload option; or, automatically by means of the Power Fail/Restart option.

The Run mode is entered manually from the Run Enable mode by pressing the Console RUN switch. If the Autoload and Power Fail/Restart options are installed, the Run mode is entered from the Run Enable mode when the AUTO switch is pressed. The Power Fail/Restart option automatically establishes the Run mode upon application of adequate power regardless of processor or console status prior to the power failure.

3.1.3 Console Operation

The ALPHA LSI Console is used for initial start-up, program debug, and trouble-shooting. The primary functions executed at the Console are register display and register change, and the display and entry of memory data. The following paragraphs discuss detailed procedures for performing these operations.

3.1.3.1 Console Preparation

There are several common steps that must be performed before any console operation may be attempted. These steps prepare the Console and the computer for console operations. The initial steps are:

1. **Power On** — The main power switch for the computer is at the rear of the chassis. Place the power switch in the up position (ON). The ON indicator on the Console will light and the chassis blowers will run.

2. **Enable Console** — Enable the Console by moving the Console Enable slide switch (located in the recess on the side of the Console) to the enable position. The ENABLE indicator is on when the Console is enabled.

3. **Press STOP** — The computer may come up in the Run mode because of a previously loaded program. Pressing STOP causes the computer to leave the Run mode.

NOTE

In some cases the RUN indicator may remain on after the STOP switch is pressed. This condition may exist when the computer is attempting to execute certain I/O instructions. This does not indicate a malfunction of the computer. When this occurs, step 4 of this procedure will correct the condition.

4. **Press RESET** — Pressing RESET puts the computer in Word mode and initializes the computer and peripheral interfaces. It forces the termination of any incomplete instructions.

3.1.3.2 Console Data Entry Procedure

The Console Data Entry procedure is used to store data into selected registers or memory locations from the ALPHA LSI Console. The general procedure is to enter the data into the Console Data register via the hex keyboard and then transfer the data to a target register or addressed memory location via the Register Select switches. The detailed procedure is as follows:

1. **Ready Console** — Prepare the Console and the computer for console operations as described in paragraph 3.1.3.1.

2. **Turn S REG/DATA Indicator off** — Enables Console Data register entry, display and transfer.

3. **Turn WRITE/READ Indicator on** — Enables writing into a selected target register or memory location.

4. **Memory Address ——→P** — Before writing into memory locations, the memory address where data is to be stored is entered into the Console Data register and the P switch is pressed to transfer the contents of the Console Data register to P. This step is not required to enter data into the A, X, I or P registers only.

5. **Data ——→ Target Register or Memory** — The data is entered into the Console Data register. The appropriate register select switch is pressed to transfer the contents of the Console Data register to the target register or addressed memory location.

6. **Sequential Memory Stores** — The P register is automatically incremented each time M is pressed. To store data in sequential memory locations, go back to step 5 for each succeeding word. To store data in a new location, go back to step 4.

3.1.3.3 Console Display Procedure

The Console Display procedure is used to display the contents of selected registers or memory locations. The general procedure is to transfer the data from a register or memory location to the Console Data register by use of the appropriate Register Select switch. The detailed procedure is as follows:

1.	Ready Console	Prepare the Console and the computer for console operations as described in paragraph 3.1.3.1.
2.	Turn S REG/DATA Indicator off	Enables Console Data register, entry, display and transfer.
3.	Turn WRITE/READ Indicator on	Enables writing desired address into P register. (Required only prior to displaying memory locations.)
4.	Memory Address ——→P	The address of the memory location to be displayed is entered into the Console Data register and the P switch is pressed. (Required only prior to displaying memory locations.)
5.	Turn WRITE/READ Indicator off	Enables reading from a selected register or memory location.
6.	Target Register or Memory ——→ Console	When the appropriate Register Select switch is pressed, the contents of the selected target register or memory location are copied into the Console Data register and displayed.
7.	Sequential Memory Displays	The P register is incremented each time M is pressed. Therefore, to display data in sequential memory locations, go back to step 6.

CAUTION

The following caution is applicable when stepping through a program on the LSI-2 computer:

If the computer is halted (execution of HLT instruction) within the range of a SIN instruction, any Console operation will cause execution of the remaining instructions within the SIN range before the Console is serviced.

3.1.3.4 Program Execution

Programs to be executed may be entered into Memory by a number of different means. Short programs may be entered using the Console Data Entry procedure described in paragraph 3.1.3.2. Longer programs may be entered using the Autoload feature or various loader programs. Regardless of the means used to get a program into Memory, the method used to execute that program is generally the same. The Program counter (P register) must be set to the starting address of the program, and the computer Run mode must be entered. The following steps are used to start program execution from the Console:

1.	Ready Console	Prepare the Console and the computer for console operations as described in paragraph 3.1.3.1.
2.	Start Address ——→P	Enter the starting address of the program to be executed in the P register.

NOTE

Enter any required starting information associated with the program in the A, X or Sense register as appropriate.

3.	Press STOP	This enables Run mode, but does not cause the computer to enter Run mode.
4.	Press RUN	Pressing the RUN switch causes the computer to enter the Run mode. The computer will continue to run until it executes a Halt instruction, or until the STOP switch is pressed.

3.1.4 Unattended Operation

If for any reason the computer is left unattended when executing a program, it is recommended that the Console be disabled by placing the Console Enable switch to the Disable position.

3.2 OPERATOR CONSOLE

3.2.1 Introduction

The Operator Console provides minimum facilities for the control and display of processor operations. It can be used in systems having at least one of the following options: Power Fail/Restart (PFR), Autoload (AL) or Automatic Start-up (ASU).

The Operator Console is connected to console interface connector J1 on the motherboard and receives its power, +5VDC and ground, through the motherboard. The console provides switches to reset the system, to interrupt the processor, and to start the processor or initiate autoload, depending on the options installed. Indicators are provided to indicate power on, system running, and overflow.

3.2.2 Switches and Indicators

All switches are of the momentary-contact type activated in the down position. All indicators are LED's. Switch and indicator operation is summarized in table 3.2.

Table 3.2 Switch/Indicators - Operator Console

Switch/Indicator	Function
ENABLE Switch	Activation of this switch provides a ground-true signal that enables all other switches on the Operator Console. ENABLE must be held down while any other switch is activated and not released until the activated switch is released.
RESET Switch	The RESET switch, when activated, forces system Reset (RST-) ground true initializing the processor and all interfaces.
START Switch	In systems having the Autoload option, this switch, when activated, generates the Autoload signal (AL-, ground-true) starting the Autoload sequence. For this operation, signals must be strapped as described in paragraph 3.2.3. In systems without Autoload option, AL- starts the processor operting from location :0000 by initiating a power-up sequence providing that signals are strapped as described in paragraph 3.2.3.
INTerrupt Switch	When activated, this switch generates the Console Interrupt signal (CINT-, ground true) commanding the processor to interrupt normal processing. Once the processor has serviced this interrupt, the Console Interrupt Enable Mask (CIE) is not reenabled for 1.5 ms, under software control.
Power ON Indicator	This indicator, when "on," indicates that power (+5VDC) is applied to the Operator Console.
RUN Indicator	This indicator, when "on," indicates that the processor is in Run mode. This LED is energized as a result of Memory Start, MST-, from the processor.
OVerflow Indicator	This indicator, when "on," indicates data overflow in the processor. It is energized by the OV flip-flop.

3.2.3 Strapping Requirements

Since the Operator Console does not have a SENSE switch or Sense Register, jumpers (or switches) must be installed to replace these functions. The requirements vary with two system configurations:

1. Systems Without Autoload Option. To start processor operation upon activation of the START switch as explained in table 3.2, AL- must be jumpered to QATLD- on the option board or at processor connector J1. Alternately, AL- can be jumpered to PFD- on the motherboard.
2. Systems with Autoload Options. With this option, the activation of START initiates an Autoload sequence. To perform an autoload and execute from a loader device, the Sense Switch signal (SSW-, pin 2) and Enable Data Sense Word (ENDSW-, pin 28) must be strapped to ground at option board connector J1. Also, data sense signals DS00 - DS03 must be strapped at the option board connector J1 for proper selection of the loader device. These signals are on the following pins of J1 (see figure 6-2):

   DS00-, pin 34
   DS01-, pin 33
   DS02-, pin 36
   DS03-, pin 31

   The device is selected by strapping the appropriate pin(s) according to table 3.3.

Table 3.3 Device Selection

LOADER DEVICE	LOAD MODE	(J1 pin no.) STRAP TO GND	EQUIVALENT HEX ADDRESS
TTY/P.T Reader	ABS	None	:0
Hi Speed P.T.	ABS	34	:1
Mag Tape	ABS	33	:2
Cassette	ABS	33,34	:3
Disc	ABS	36	:4
TTY/P.T Reader	REL	31	:8
Hi Speed P.T.	REL	31,34	:9
Mag Tape	REL	31,33	:A
Cassette	REL	31,33,34	:B
Disc	REL	31,36	:C

To perform an Autoload and execute without a loader device, all data sense signals (bits) must be grounded (:F). This causes an unconditional exit to location :31 (see paragraph 6.5.6).

Section 4

INSTRUCTIONS AND DIRECTIVES

4.1 INTRODUCTION

This section deals with the various instructions and directives recognized by the assembler. The Beta assembler translates programs which are written in a symbolic language (mnemonics, etc.) into an object language (machine code - see appendices C and D) which may be loaded into the ALPHA LSI computer. Outputs from the assembler consist of the program object code (typically a punched paper tape) and the program assembly listing. The Beta assembler is a two-pass assembler. A symbol table for the program is compiled on the first pass and the program object code and assembly listing are produced on the second pass.

4.1.1 Instruction and Directive Classes

The instruction and directive classes are listed below in figure 4-1. They are discussed in this section.

CLASS	1	SINGLE-WORD MEMORY REFERENCE INSTRUCTIONS
CLASS	2	DOUBLE-WORD MEMORY REFERENCE INSTRUCTIONS
CLASS	3	STACK INSTRUCTIONS
CLASS	4	BYTE IMMEDIATE INSTRUCTIONS
CLASS	5	CONDITIONAL JUMP INSTRUCTIONS
CLASS	6	SHIFT INSTRUCTIONS
CLASS	7	REGISTER CHANGE AND CONTROL INSTRUCTIONS
CLASS	8	INPUT/OUTPUT INSTRUCTIONS
CLASS	9	JUMP ON CONDITION INSTRUCTIONS
CLASS	10	ASSEMBLER CONTROL DIRECTIVES
CLASS	11	DATA AND SYMBOL DEFINITION DIRECTIVES
CLASS	12	PROGRAM LINKAGE DIRECTIVES
CLASS	13	SUBROUTINE DEFINITION DIRECTIVES
CLASS	14	LISTING FORMAT AND ASSEMBLER INPUT DIRECTIVES
CLASS	15	USER DEFINED OPERATION CODE DIRECTIVES

Figure 4-1. Instruction and Directive Classes

4.1.2 Symbolic Notation

The symbolic source code input to the Beta assembler consists of individual symbolic statements. All of the statements taken together make up a program which is to be translated.

All instructions and certain directives generate an object code. Other directives serve only to control the assembly process.

A source statement represents either an instruction or a directive. It contains four fields - the Label field, the Operation Code (Op Code) field, the Operand field and the Comments field. Adjacent fields are separated by one or more spaces which allows free-form symbolic input to the assembler. A space in the first character position of a source statement indicates no label present. The listing output from the assembler is formatted for ease in reading, with the Op Code, Operand and the Comments fields beginning at fixed positions on the listing. Source statements on paper tape are terminated with a carriage return. Line feeds and "rubouts" are ignored. All source statements are limited to 72 characters.

The instructions and directives acceptable to the BETA assembler are described in detail in the remainder of this section. The following conventions apply:

1. Square brackets [ ] enclose elements which are optional and may be included or omitted as required.
2. Two or more elements separated by a vertical bar (|) indicates a choice must be made from the enclosed elements.
3. A right square bracket followed by dots ( ] ...) indicates that the enclosed element may be repeated an arbitrary number of times.

4.1.3 Assembler Source Statement Fields

The following paragraphs discuss the four assembler source statement fields. The relative positions of the fields are shown below in figure 4-2.

LABEL FIELD	OP CODE FIELD	OPERAND FIELD	COMMENTS FIELD

Figure 4-2. Source Statement Format.

4.1.3.1 Label Field

The Label field may contain a name which can be referenced by other instruction statements. It is identified by an alphabetic (A-Z) character in the first position of the source statement. This first character may be followed by as many as five alphanumeric (A-Z, 0-9) or colon (:) characters. This field is terminated by one or more spaces.

At assembly time, the label is assigned the current value and relocation attribute of the Program counter (P register). The same name may not appear in the Label field of more than one source statement in a given program (except when used with the SET directive).

4.1.3.2 Op Code Field

The Op Code field contains a legally defined symbolic instruction or directive. In addition, user-defined Op codes may appear in this field. The Op Code field consists of not less than one nor more than four characters, and is terminated by one or more spaces. The Op Code field of a source instruction statement must be present.

4.1.3.3 Operand Field

The various instructions and directives may or may not require operands. In any case, the syntax of the Operand field depends on the type of instruction or directive with which it is associated. The Operand field syntax description is contained in the discussions of the instructions and directives. If the Operand field is present, it contains an expression consisting of one of the following:

1. The currency symbol ($), representing the current program location.
2. A single symbolic term.
3. A single numeric term.
4. A combination of symbolic terms, numeric terms and/or the currency symbol joined by the arithmetic operators plus (+) or minus (-).
5. A text string.
6. A literal (=xx).

The value assigned the currency symbol by the assembler is the value of the assembler's Working Location Counter at the time the currency symbol is encountered. The value is absolute if an absolute assembly is being performed and relative if a relocatable assembly is being performed. The currency symbol allows the programmer to reference memory locations relative to the instruction being written rather than assigning labels to the referenced location.

Symbolic terms (names) may be absolute or relative, depending on the assembly mode under which they have been defined.

Numeric terms are always absolute. They consist of decimal, octal and hexadecimal numbers. Decimal numbers can be any value in the range -32768 through +32767. The first digit of the number must be non-zero. Octal numbers can be any octal value in the range 0 through 0177777. The first - or leading - digit of the number must be zero to specify octal numbers. Hexadecimal numbers can be any hexadecimal value in the range :0 through :FFFF. The number must be preceded by a colon (:). Although octal and hexadecimal numbers may be signed, they are normally used to generate a bit pattern or reference a particular memory location rather than to generate a signed numeric value.

Combinations of terms (including the currency symbol) can be achieved by using the arithmetic operators plus (+) and minus (-). The value of the final expression will be in the range :0 thru :FFFF. Combinations of relative and absolute terms are governed by additional restrictions (see paragraph 4.1.5).

Text strings consist of any sequence of characters surrounded by single quotes ('). Inclusion of a single quote within the character string is accomplished using two adjacent single quotes. The object code generated consists of 8-bit ASCII character codes, packed two characters per word, or one 8-bit ASCII character in the LS byte of an instruction (e.g., the operands of Immediate instructions). When a DATA directive is used, the text string may consist of one or two characters. When one character is specified, the 8-bit code appears in the LSB byte of the computer word, with the MS byte set to zero.

If two characters are specified, the code for the first character is put in the MS byte of the computer word and the code for the second character is put in the LS byte of the computer word. When the TEXT directive is used, the text string may consist of as many as 57 characters. The characters are packed two per word, with the code for the first character appearing in the MS byte of the computer word and the code for the second character appearing in the LS byte of the computer word. Trailing character positions are filled with blanks (:A0) - e.g., TEXT 'A' would generate a value of :C1A0 for the specified computer word.

Literals are designated by preceding the expression in the operand with an equal (=) sign (literals are only valid for class 1 instructions). This affects the entire expression, not just one term in the expression. When a literal is encountered by the assembler, a word is reserved in the scratchpad area of Memory to hold the computed value of the expression in the Operand field. Memory addressing is then generated to access the scratchpad location.

4.1.3.4 Comments Field

The Comments field follows the Operand field or, for those instructions which do not require operands, the Op Code field. This field generally contains programmer's notes, cryptic messages, helpful hints, etc. Comments appear on the assembly listing, but do not generate object code.

4.1.4 Arithmetic Operations and Overflow

The ALPHA LSI computer performs two's complement arithmetic. All additions and subtractions are performed on full 16-bit values. Thus, addition operations involving byte values place the 8-bit data in the least significant 8 bits of the adder and set the most significant 8 bits to zero (e.g., AXI :50 would add :0050 to the 16-bit X register). Subtraction operations involving byte values similarly obtain the 16-bit two's complement of the data (e.g., SXI :50 would add :FFB0 to the 16-bit X register).

Arithmetic overflow occurs when the result of an arithmetic operation exceeds the range -32768 through +32767. Specifically, this involves the carry from bit 14 to bit 15 of the adder, and the carry out of bit 15 (CO). If the carry from bit 14 to 15 is not the same as the carry from 15 to CO (0 and 1 or 1 and 0), an arithmetic overflow has occurred and the overflow (OV) indicator is set. The operation is described below in figure 4-3.

1. Carry In and Carry Out — No Overflow

	CO	S				
	1	1	111	1111	1111	011←carries
-5 =		1	111	1111	1111	1011
+ (-5) =		1	111	1111	1111	1011
-10 =		1	111	1111	1111	0110

2. No Carry In and No Carry Out — No Overflow

	CO	S				
	0	0				101←carries
+5 =		0	000	0000	0000	0101
+ (+5) =		0	000	0000	0000	0101
+10 =		0	000	0000	0000	1010

3. Carry In and No Carry Out — Overflow

	CO	S				
	0	1	111	1111	1111	111←carries
+32767 =		0	111	1111	1111	1111
+ (+1) =		0	000	0000	0000	0001
32768 =		1	000	0000	0000	0000

4. Carry Out and No Carry In — Overflow

	CO	S				
	1	0				←carries
-32768 =		1	000	0000	0000	0000
+ (-1) =		1	111	1111	1111	1111
-32769 =		0	111	1111	1111	1111

Figure 4-3. Arithmetic Overflow

4.1.5 Relocatability

Relative and absolute programming modes are controlled by the REL and ABS directives. The default condition of the assembler is the Relative (REL) mode. The programmer should note that the ORG directive modifies the contents, but not the relocation attribute, of the assembler's Working Location Counter.

An absolute program (or section of coding) can only be loaded and executed in the memory locations specified by the user at assembly time, whereas a relative (or relocatable) programs may be loaded and executed in any memory area specified by the user at load time. Out-of-range memory references are resolved through the use of the scratchpad area in the base page (the first 256 words of Memory). The user should refer to the LAMBDA Object Loader documentation.

Multiple-term expressions are reduced by the assembler to a single expression which may be relocatable or absolute, according to the following rule:

R = (Number of added relocatable terms) - (Number of subtracted relocatable terms)

If R = 1, the expression is relocatable; if R = 0, the expression is absolute; and if R is not equal to 0 or 1, the expression is illegal.

Relocatable expressions are modified by the load bias (established at program load time) when the LAMBDA Object loader is executed:

Relocated Expression Value = Assembled Expression Value + Load Bias

In addition, the location of the entire program (or block of coding) is offset by the same load bias:

Relocated Program Location = Assembled Program Location + Load Bias.

4.2 MEMORY REFERENCE INSTRUCTIONS

4.2.1 Word Mode Operations and Instruction Format

Word mode Memory Reference operations access full 16-bit memory operands. The default mode of the computer is the Word mode - i.e., when no mode control instruction has been executed, the computer is in the Word mode. SWM is the mode control instruction which places the computer in the Word mode. In addition, the SIN, SIA and SIX instructions force the computer into the Word mode. The SIN instruction forces the Word mode for the number of succeeding instructions specified by its associated operand. The SIA and SIX instructions unconditionally force the Word mode. The format for the Word mode Memory Reference instructions is shown in figure 4-4.

[LABEL] OPCODE [* | @ | *@] EXPRESSION [COMMENTS]

No Operator = Direct Address
* = Indirect Addressing (multi-level)
@ = Indexed Addressing
*@ = Indirect Post-indexed Addressing (multi-level)

Figure 4-4. Word Mode Memory Reference Instruction Format

All (16-bit) word address pointers (defined by DATA statements) consist of fifteen bits of address in the least significant 15 bits. The most significant bit (bit 15) specifies indirect addressing if equal to 1 or direct addressing if equal to 0.

4.2.1.1 Word Mode Direct Addressing

Word mode direct addressing allows any Memory Reference instruction to access the first 256 words of Memory (the base page/scratchpad area) as well as 512 memory locations about the instruction itself (relative to P). Relative to P forward addressing includes 256 words forward (toward higher memory) of the instruction and relative to P backwards

addressing includes the instruction itself and 255 memory locations backward from the instruction. When direct addressing is desired, the expression in the Operand field should not be preceded by an * or @ character. When the assembler encounters a direct reference to an out of range memory location, it automatically generates an address pointer in the scratchpad area and references the associated memory location indirectly through the pointer.

4.2.1.2 Word Mode Indirect Addressing

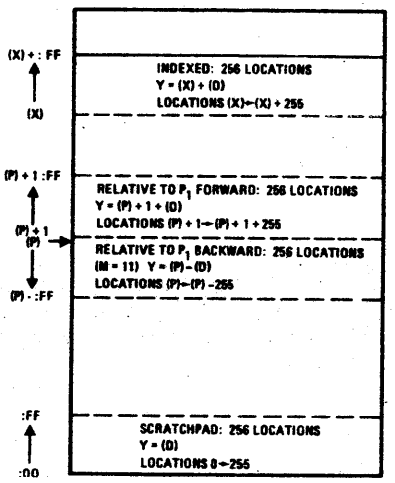
Word mode indirect addressing allows any Memory Reference instruction to access any memory location through an address pointer in the scratchpad area or an address pointer in the 512 memory locations about the instruction itself (relative to P). Relative to P forward indirect addressing allows the address pointer to reside in any memory location up to 256 words forward (toward higher memory) of the instruction and relative to P backwards indirect addressing allows the address pointer to be in any memory location 255 words or less prior to the instruction. When indirect addressing is desired, the expression in the Operand field should be preceded by an asterisk (*). Multi-level indirect addressing is accomplished by accessing address pointers in which the most significant bit (bit 15) is set. The memory operand is not accessed until an address pointer with the most significant bit reset (= 0) is encountered. Indirect address pointers can be defined by the programmer through the use of the DATA directive by preceding the expression in the Operand field with an asterisk (*).

4.2.1.3 Word Mode Direct Indexed Addressing

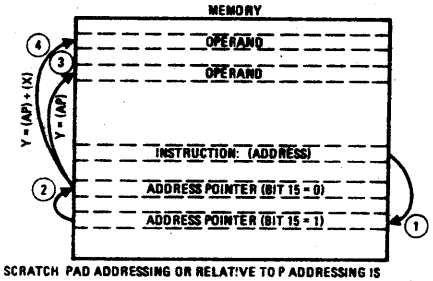
Word mode direct indexed addressing allows any Memory Reference instruction to access memory locations by algebraically summing the signed contents of the X register and any offset value in the range 0 through 255. The offset value is defined by the expression in the Operand field. When direct indexed addressing is desired, the expression in the Operand field should be preceded by an @ symbol. When the assembler encounters an expression with a value greater than 255 in the Operand field of a direct indexed Memory Reference instruction, it automatically generates an address pointer in the scratchpad area and references the associated memory location indirect postindexed, through the pointer.

4.2.1.4 Word Mode Indirect Postindexed Addressing

Word mode indirect postindexed addressing allows any Memory Reference instruction to access memory locations by algebraically summing the contents of the X register and the contents of an address pointer in the scratchpad area. If the most significant bit of the address pointer is set, it contains the address of another address pointer, which in turn may contain the address of another pointer, and so forth. When an address



Direct Adressing



(1) SCRATCH PAD ADDRESSING OR RELATIVE TO P ADDRESSING IS USED TO ADDRESS AN ADDRESS POINTER

(2) BITS 0-14 OF THE ADDRESS POINTER CONTAIN A MEMORY ADDRESS. IF BIT 15 OF THE ADDRESS POINTER CONTAINS A 1-BIT, THE MEMORY ADDRESS IN BITS 0-14 IS THE ADDRESS OF ANOTHER ADDRESS POINTER.

(3) IF BIT 15 OF THE ADDRESS POINTER CONTAINS A 0-BIT, THE ADDRESS IN BITS 0-14 IS THE ADDRESS OF THE MEMORY OPERAND.

(4) IF INDEXING IS SPECIFIED BY THE INSTRUCTION, THE ADDRESS IN BITS 0-14 IS ADDED TO THE CONTENTS OF THE X REGISTER TO FORM THE EFFECTIVE OPERAND ADDRESS.

Indirect Adressing

Figure 4-5. Word Mode Adressing Summary

pointer with the most significant bit (bit 15) set to zero is found, the contents of the X register are added to it to form the effective memory address. The memory operand is then accessed. When indirect postindexed addressing is desired, the expression in the Operand field should be preceded by an asterisk (*) and an @ symbol.

Because the Scan Memory (SCM) instruction always uses indirect postindexed addressing, the assembler automatically generates the necessary machine code and does not allow @ or * operators on the associated operand expression. The operand expression for this instruction should reference a user-defined address pointer in the base page.

4.2.1.5 Word Mode Summary

A summary of Word mode addressing is shown in figure 4-5.

4.2.2 Byte Mode Operations and Instruction Format

Byte mode Memory Reference operations access 8-bit byte operands. The Byte mode is established by execution of the Set Byte Mode (SBM) instruction. Byte mode is inhibited (the computer is forced into the Word mode) by execution of the SIN, SWM, SIA and SIX instructions. The SIN instruction inhibits Byte mode operations for the number of succeeding instructions specified by its associated operand. The SWM, SIA and SIX instructions unconditionally force the computer into the Word mode. The format for Byte mode Memory Reference instructions is shown below in figure 4-6.

[LABEL] OP CODE [*|@|*@] EXPRESSION [COMMENTS]

No Operator = Direct Address
* = Indirect Addressing (One Level)
@ = Indexed Addressing
*@ = Indirect Postindexed Addressing (One Level)

Figure 4-6. Byte Mode Memory Reference Instruction Format

All (16-bit) byte address pointers (BAC directive) consist of fifteen bits of word address in the most significant 15 bits. The least significant bit (bit 0) specifies the most significant 8 bits (MS byte) of the addressed word if equal to 0, or the least significant 8 bits (LS byte) if equal to 1. Only one level of byte memory reference indirect addressing, specified in the instruction itself, is possible. Byte operands affecting the register are always right-justified, i.e., bytes cannot be loaded into, added to or stored from the MS bytes of the A and X registers.

The IMS, MPY, DVD, NRM, JMP and JST instructions are not affected by the Byte mode. They always use full 16-bit word operands.

4.2.2.1 Byte Mode Direct Addressing

Byte mode direct addressing allows any byte Memory Reference instruction to access the first 256 bytes (128 words) of Memory as well as 512 byte locations forward (toward higher memory) of the instruction itself. When direct addressing is desired, the expression in the Operand field should not be preceded by an * or @ character. When the assembler encounters a direct reference to an out of range byte location, it automatically generates a byte address pointer in the scratchpad area and references the associated byte location indirectly through the pointer.

4.2.2.2 Byte Mode Indirect Addressing

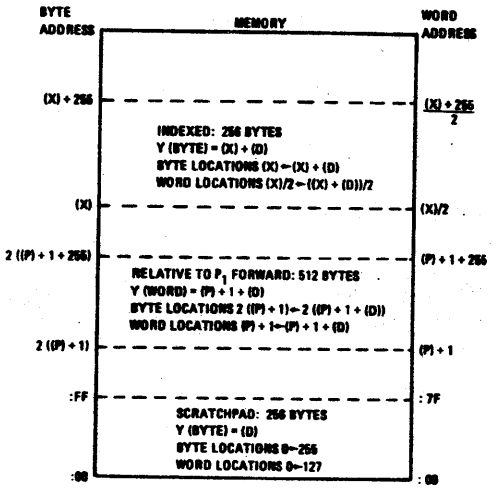
Byte mode indirect addressing allows any byte Memory Reference instruction to access any byte location through a byte address pointer in the scratchpad area or a byte address pointer in the memory locations about the instruction itself (relative to P). Relative to P forward indirect addressing allows the byte address pointer to reside in any memory location up to 256 words forward (toward higher memory) of the instruction and relative to P backwards indirect addressing allows the byte address pointer to be in any memory location 255 words or less prior to the instruction. When indirect addressing is desired, the expression in the Operand field should be preceded by an asterisk (*). Byte address pointers to be used by indirect byte Memory Reference instructions can be defined by the programmer by using the BAC directive. Since a byte address pointer utilizes all 16 bits to specify a given byte location, indirect byte addressing is limited to one level.

4.2.2.3 Byte Mode Direct Indexed Addressing

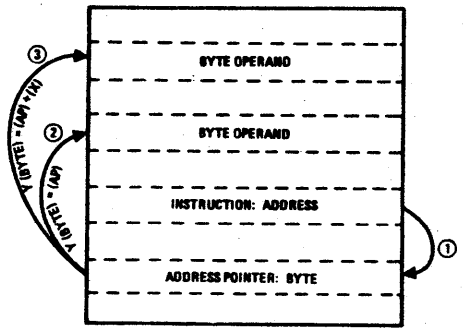
Byte mode direct indexed addressing allows any byte Memory Reference instruction to access byte locations by summing the contents of the X register and any base value in the range 0 through 255. The base value is defined by the expression in the Operand field. When direct indexed addressing is desired, the expression in the Operand field should be preceded by an @ symbol. When the assembler encounters an expression with a value greater than 255 in the Operand field of a direct indexed byte Memory Reference instruction, it automatically generates a byte address pointer in the scratchpad area and references the associated byte memory location indirect postindexed through the byte address pointer.

4.2.2.4 Byte Mode Indirect Postindexed Addressing

Byte mode indirect postindexed addressing allows any byte Memory Reference instruction to access byte locations by summing the contents of the X register and the contents of a byte address pointer in the scratchpad area. When indirect postindexed byte addressing is desired, the expression in the Operand field should be preceded by an asterisk (*) and an @ symbol.



Direct Addressing



① SCRATCHPAD OR RELATIVE ADDRESSING IS USED TO ADDRESS A FULL WORD ADDRESS POINTER.

② IF INDEXING IS NOT REQUIRED, THE ADDRESS POINTER CONTAINS THE EFFECTIVE 16-BIT BYTE ADDRESS.

③ IF INDEXING IS REQUIRED, THE BYTE ADDRESS IN THE ADDRESS POINTER IS ADDED TO THE VALUE IN THE X REGISTER TO FORM THE EFFECTIVE BYTE ADDRESS.

Indirect Addressing

Figure 4-7. Byte Mode Addressing Summary

Because the Scan Memory Byte (SCMB) instruction always uses indirect postindexed addressing, the assembler automatically generates the necessary machine code and does not allow @ or * operators on the associated operand expression. When performing byte scans, the operand expression for this instruction should reference a user defined byte address pointer in the base page.

4.2.2.5 Byte Mode Summary

A summary of Byte mode addressing is shown in figure 4-7.

4.2.3 Arithmetic Memory Reference Instructions

ADD ADD TO A. Adds contents of effective memory location to contents of A register. OV is set if arithmetic overflow occurs.

ADDB ADD BYTE TO A. Adds contents of effective byte location to contents of A register. OV is set if arithmetic overflow occurs.

SUB SUBTRACT FROM A. Subtracts contents of effective memory location from contents of A register. OV is set if arithmetic overflow occurs.

SUBB SUBTRACT BYTE FROM A. Subtracts contents of effective byte location from contents of A register. OV is set if arithmetic overflow occurs.

4.2.4 Logical Memory Reference Instructions

AND AND TO A. Logically AND's contents of effective memory location with contents of A register. Result replaces contents of A register.

ANDB AND BYTE TO A. Logically AND's contents of effective byte location with contents of LS byte of A register. Result replaces contents of LS byte of A register. MS byte of A register is reset to zero.

IOR INCLUSIVE OR TO A. Inclusively OR's contents of effective memory location with contents of A register. Result replaces contents of A register.

IORB INCLUSIVE OR BYTE TO A. Inclusively OR's contents of effective byte location with contents of LS byte of A register. Result replaces contents of LS byte of A register. MS byte of A register remains unchanged.

XOR EXCLUSIVE OR TO A. Exclusively OR's contents of effective memory location with contents of A register. Result replaces contents of A register.

XORB EXCLUSIVE OR BYTE TO A. Exclusively OR's contents of effective byte location with contents of LS byte of A register. Result replaces contents of LS byte of A register. MS byte of A register remains unchanged.

4.2.5 Data Transfer Memory Reference Instructions

LDA LOAD A. Loads contents of effective memory location into A register.

LDAB LOAD A BYTE. Loads contents of effective byte location into LS byte of A register. MS byte of A register is reset to zero.

LDX LOAD X. Loads contents of effective memory location into X register.

LDXB LOAD X BYTE. Loads contents of effective byte location into LS byte of X register. MS byte of X register is reset to zero.

STA STORE A. Stores contents of A register in effective memory location.

STAB STORE A BYTE. Stores contents of LS byte of A register in effective byte location.

STX STORE X. Stores contents of X register in effective memory location.

STXB STORE X BYTE. Stores contents of LS byte of X register in effective byte location.

EMA EXCHANGE MEMORY AND A. Simultaneously stores contents of A register in effective memory location and loads contents of effective memory location into A register.

EMAB EXCHANGE MEMORY BYTE AND A. Simultaneously stores contents of LS byte of A register in effective byte location and loads contents of effective byte location into LS byte of A register. MS byte of A register is reset to zero.

4.2.6 Program Transfer Memory Reference Instructions

CMS COMPARE MEMORY TO A AND SKIP IF HIGH OR EQUAL. Compares contents of effective memory location with contents of A register. If A register is greater than contents of memory location, a one word skip occurs. If A register is equal to contents of memory location, a two word skip occurs. If A register is less than contents of memory location, next sequential instruction is executed.

CMSB COMPARE BYTE AND SKIP IF HIGH OR EQUAL. Compares contents of effective byte location with contents of A register. If A register is greater than contents of byte location, a one word skip occurs. If A register is equal to contents of byte location, a two word skip occurs. If A register is less than contents of byte location, next sequential instruction is executed. All 16 bits of A register are compared to contents of effective byte location, so MS byte of A register should be equal to zero.

IMS INCREMENT MEMORY AND SKIP ON ZERO RESULT. Contents of effective memory location are incremented by one. If increment causes result to become zero, a one word skip occurs. If not, next sequential instruction is executed. OV is set if arithmetic overflow occurs.

NOTE

IMS is often used as an interrupt instruction in which case, when the increment causes a zero result, an ECHO signal is generated and sent to the interrupting device. The interrupting device uses the ECHO signal to develop an EOB (End-of-Block) interrupt. Under these conditions a skip does not occur and OV is unaffected. (See paragraph 5.3).

JMP JUMP UNCONDITIONAL. P register is loaded with the address of effective memory location causing an unconditional branch to that address.

JST JUMP AND STORE. Contents of P register (address of JST instruction +1) are stored in effective memory location and P register is then loaded with address of effective memory location +1, causing an unconditional branch to that address.

NOTE

JST is often used as an interrupt instruction. When used as such, all interrupts under EIN/DIN control are automatically disabled upon instruction execution. (See paragraph 5.3). In this case, the P register content is not the address of JST instruction +1.

SCM SCAN MEMORY. Compares contents of A register with contents of memory location in data buffer defined by address pointer in scratchpad (base address of data buffer - 1) added to contents of X register (buffer length). If a match is found, Scan is terminated and next sequential instruction is executed. X register is decremented once for each word scanned. Thus, data buffer is scanned in descending order, beginning with highest memory location and ending with lowest (base address). When a match is found, X register contains number of words remaining to be scanned. Remainder of data buffer can be scanned simply by executing SCM instruction again. If a match is not found when X register reaches zero, a one word skip occurs and instruction terminates.

SCMB SCAN MEMORY BYTE. Compares contents of A register with contents of memory byte locations in data buffer defined by byte address pointer in scratchpad (byte base address of pointer - 1) added to contents of X register (data buffer length in bytes). If a match is found, Scan is terminated and next sequential instruction is executed. X register is decremented once for each byte scanned. Thus, data buffer is scanned, by byte, in descending order, beginning with highest memory byte location and ending with lowest (base address). Remainder of data buffer can be scanned simply by executing SCMB instruction again. If a match is not found when X register reaches zero, a one word skip occurs and instruction terminates. All 16 bits of A register are compared to contents of effective byte location, so MS byte of A register should be equal to zero.

NOTES

1. The SCM and SCMB instructions are interruptable. Upon completion of interrupt processing, Scan resumes operation at the point where the interrupt occurred.
2. The Set Byte Mode (SBM) instruction must be executed prior to the execution of the SCMB instruction.

4.3 DOUBLE-WORD MEMORY REFERENCE INSTRUCTIONS

4.3.1 Format

The Double-Word Memory Reference instructions require two consecutive memory locations and allow direct and indirect addressing. Indexed addressing is not allowed and is, in fact, not useful, since these instructions manipulate both the A and X registers. The format for Double-Word Memory Reference instructions is shown in figure 4-8.

[LABEL] OP-CODE [*]EXPRESSION 1[,EXPRESSION 2] [COMMENTS]

No Operator = Direct Address
* = Indirect Addressing (multi-level)
EXPRESSION 1: any absolute or relative expression defining the effective memory location.
EXPRESSION 2: an optional instruction count in the range 0 thru 31 for NRM.

Figure 4-8. Double-Word Memory Reference Format

4.3.2 Instructions

DVD DIVIDE. Divides contents of the A and X registers by contents of memory location addressed by Expression 1. This address pointer (Expression 1) may be direct or indirect and occupies second word of double-word DVD instruction.

Prior to execution of instruction, A and X registers contain signed 30 bit dividend (as shown in figure 4-9), and addressed memory location contains signed full-word divisor. Both dividend and divisor must be positive.

Quotient is placed in X register (sign plus 15 bits) and fractional remainder in A register (sign plus 15 bits). OV is set if a divide fault occurs (Divisor ≤ most significant half of dividend). If no divide fault occurs, OV is returned to original state (prior to DVD instruction). Note that least significant half of dividend is 15 bits, left justified.

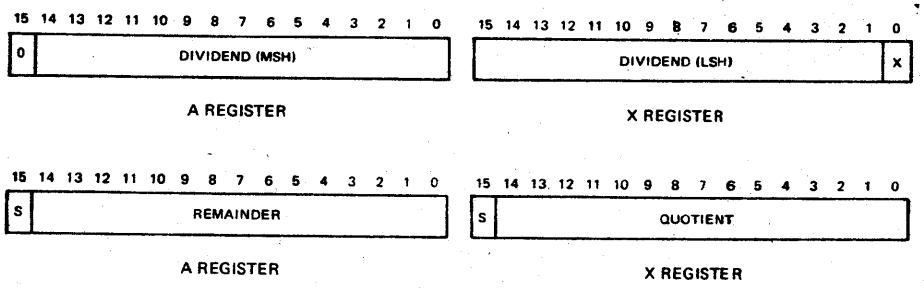


Figure 4-9. Divide

MPY MULTIPLY AND ADD. Multiplies contents of X register by contents of the memory location addressed by Expression 1 and then adds contents of A register to product. Address pointer (Expression 1) may be direct or indirect and occupies second word of double-word MPY instruction.

Prior to execution of MPY instruction, X register contains signed full-word multiplicand, addressed memory location contains full-word multiplier, and A register contains "offset" to be added. (Refer to figure 4-10.) Multiplier and offset must be positive or zero. Multiplicand may be either positive, negative or zero. Result is placed in A and X registers (sign plus 30 bits). Note that least significant half of result is a 15-bit left justified value consistent with format of least significant half of dividend.

In all cases OV will be reset (= 0) at completion of a full multiply. The contents of OV prior to execution of MPY will be returned in the least significant bit (bit 0) of the X register.

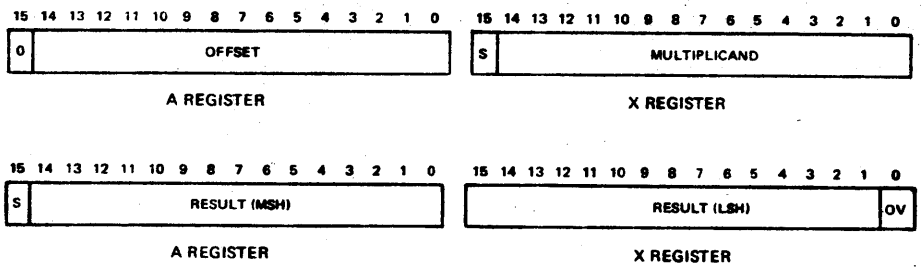


Figure 4-10. Multiply and Add

NRM NORMALIZE A AND X. Contents of A and X registers are arithmetically shifted left (see figure 4-11) until bit 15 of A register is not equal to bit 14 or until maximum shift count specified (Expression 2) is exhausted. Exponent (count cell), addressed by Expression 1, is a two's complement number which is decremented (incremented in two's complement) once for each shift until normalization occurs. Address of exponent may be direct or indirect and occupies second word of double-word NRM instruction. No indication is given if arithmetic overflow occurs when exponent is decremented.

NRM instruction treats A and X registers as a combined 31-bit, plus sign, register.

OV is reset (= 0) if normalization occurs; otherwise it is set (= 1). In either case, exponent will be decremented once for each shift performed.

A full 31-bit normalize is performed if no instruction count (Expression 2) is specified. Otherwise, specified count will determine maximum shifts performed. A normalize operation with a count of zero (Expression 2) provides a test for normalization without affecting contents of A and X registers.

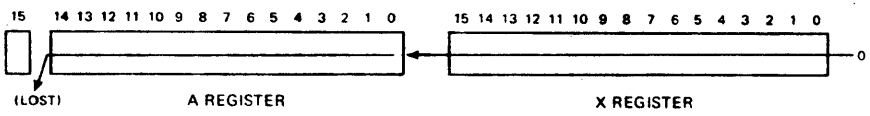


Figure 4-11. NRM Shift Path

4.4 STACK, DOUBLE WORD INSTRUCTIONS (LSI-2 only)

Stack instructions permit the programmer to enter or retrieve a full 16-bit word from a stack. A stack is a group of continuous memory locations whose length is variable up to 32,768 words. A stack is organized on a last-in-first-out basis whereby the last word entered into the stack will be the first word retrieved from the stack.

A stack can start at any address and fills from upper memory toward lower memory (decreasing addresses). The stack instructions themselves do not provide any stack boundary limit testing features. The user must provide boundary limit testing as overhead associated with using Stack instructions.

All stack accesses are controlled by a stack pointer for each stack. The stack pointer is a 15-bit address which points to the most recently accessed location in the stack. The contents of the stack pointer are referred to as the stack element address--SEA. The stack pointer may be located anywhere in Memory.

Stack instructions occupy two consecutive words in memory and operate in Word mode only, independent of processor status. The first word contains the instruction while the second word contains the address of the stack pointer. The format for Stack instructions is shown below in figure 4-12.

With the stack pointer and the stack pointer address, indirection is not possible since the Processor ignores bit 15. If bit 15 of the stack pointer is a 1, the stack pointer will be treated as a negative number when indexing (see paragraph 4.4.1.2).

[LABEL] OPCODE OPERAND [,AM] [COMMENTS]

AM = No Operator = Direct Access
- = PUSH (stack pointer decremented prior to access)
+ = POP (stack pointer incremented after access)
@ = Indexed (single level)

Figure 4-12. Stack Instruction Format

The Label and Comment fields are optional with this class of instruction.

The Op Code field must be present. The legal op codes for Stack instructions are defined in paragraphs 4.4.2 through 4.4.6 inclusive.

The Operand field consists of one or two expressions. The first expression represents a memory address and must be present. The second expression (AM) is optional and, when included, must be separated from the first by a comma. This expression represents the addressing mode of the Stack instruction. Figure 4-12 gives a list of valid expression characters and their associated addressing modes, and 4.4.1 describes them in greater detail.

These instructions generate two 16-bit words. The first word is the Stack instruction Op code. The second word is the absolute address of the stack pointer.

4.4.1 Addressing Modes (Figure 4-13)

To provide flexibility in stack management, four addressing modes are provided with Stack instructions.

4.4.1.1 Direct Access to Stack

In the Direct Access mode, the second word of the instruction (stack pointer address -- SPA) is used to fetch the stack pointer from Memory. In this mode, the stack pointer contains the effective stack element address (SEA) and is used to access the stack element for entry, retrieval, or testing of data.

4.4.1.2 Indexed Access to Stack

In the Indexed Access mode, the SPA in the second word of the instruction is used to fetch the stack pointer from Memory. The contents of the signed X register are then algebraically summed with the stack pointer to form the effective SEA. After the summation, bit 15 is treated as a 0 for accessing the stack element. This allows access to the nth element in the stack relative to the last stack entry when the X register contains n. For example, if X = 0, the most recent stack entry is accessed while if X = 1, the next most recent entry is accessed.

4.4.1.3 Auto-Postincrement Access to Stack (POP)

In the Auto-Postincrement mode, the SPA is used to obtain the stack pointer. In this mode, the stack pointer contains the effective SEA and provides direct access to the stack element. Upon completion of the stack access, the stack pointer is incremented and restored to its memory location. This mode of addressing appears to remove (POP) the most recent entry from the stack when used with a load type instruction.

4.4.1.4 Auto-Predecrement Access to Stack (PUSH)

In this mode, the stack pointer is accessed via the SPA, decremented by one, and restored. The stack element is then accessed using the decremented contents of the stack pointer. This mode of addressing appears to insert (PUSH) a new entry onto the stack when used with a store type instruction.

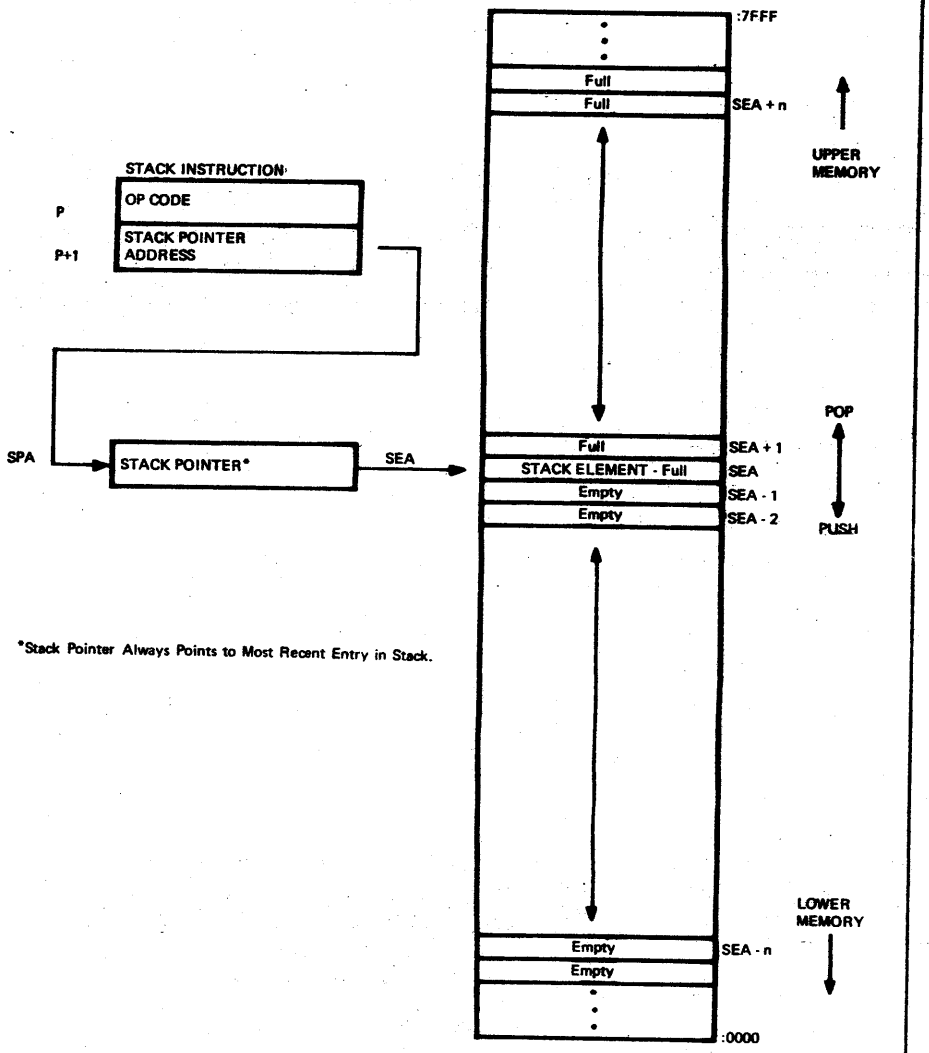


Figure 4-13. Stack Organization and Management

4.4.2 Arithmetic Stack Instructions

ADDS ADD STACK ELEMENT TO A. Adds contents of stack element to contents of A register. OV is set if arithmetic overflow occurs.

SUBS SUBTRACT STACK ELEMENT FROM A. Subtracts contents of stack element from contents of A register. OV is set if arithmetic overflow occurs.

4.4.3 Logical Stack Instructions

ANDS AND STACK ELEMENT TO A. Logically AND's contents of stack element with contents of A register. Result replaces contents of A register.

IORS INCLUSIVE OR STACK ELEMENT TO A. Inclusively OR's contents of stack element with contents of A register. Result replaces contents of A register.

XORS EXCLUSIVE OR STACK ELEMENT TO A. Exclusively OR's contents of stack element with contents of A register. Result replaces contents of A register.

4.4.4 Data Transfer Stack Instructions

EMAS EXCHANGE STACK ELEMENT AND A. Simultaneously stores contents of A register in stack element and loads contents of the stack element into A register.

LDAS LOAD STACK ELEMENT INTO A. Loads contents of stack element into A register.

LDXS LOAD STACK ELEMENT INTO X. Loads contents of stack element into X register.

STAS STORE A IN STACK ELEMENT. Stores contents of A register in stack element.

STXS STORE X IN STACK ELEMENT. Stores contents of X register in stack element.

4.4.5 Program Transfer Stack Instructions

CMSS COMPARE STACK ELEMENT TO A AND SKIP IF HIGH OR EQUAL. Compares contents of stack element with contents of A register. If A register is greater than contents of stack element, a one word skip occurs. If A register is equal to contents of stack element, a two word skip occurs. If A register is less than contents of stack element, next sequential instruction is executed.

IMSS INCREMENT STACK ELEMENT AND SKIP ON ZERO RESULT. Contents of stack element are incremented by one. If increment causes result to become zero, a one word skip occurs. If not, the next sequential instruction is executed. OV is set if arithmetic overflow occurs.

JMPS JUMP UNCONDITIONAL. P register is loaded with contents of stack pointer (SEA), causing an unconditional branch to the addressed stack element location. Next instruction is executed from location SEA.

JSTS JUMP AND STORE TO STACK ELEMENT. Contents of P register (P + 2) are stored in stack element and P register is then loaded with address of stack element plus one (SEA + 1). Next instruction is accessed from location SEA + 1.

4.4.6 Stack Control Instruction

SLAS STACK ELEMENT ADDRESS TO A. Loads contents of stack pointer into A register.

4.5 IMMEDIATE INSTRUCTIONS

4.5.1 Format

Immediate instructions are similar to Memory Reference instructions in that they perform logical and arithmetic operations involving memory data and operating registers. The memory data, however, is stored within the immediate instruction itself rather than in a separate operand word or byte. The operands of the instructions may be any absolute expression which is within the range 0 through :FF (i.e., any absolute expression which fits into eight bits). The Immediate instruction format is shown in figure 4-14.

[LABEL]	OP-CODE	EXPRESSION	[COMMENTS]

EXPRESSION: must be absolute and in the range : 0 thru : FF

Figure 4-14. Immediate Instruction Format

4.5.2 Instructions

AAI — ADD TO A IMMEDIATE. Operand is added to contents of A register. OV is set if arithmetic overflow occurs.

AXI — ADD TO X IMMEDIATE. Operand is added to contents of X register. OV is set if arithmetic overflow occurs.

SAI — SUBTRACT FROM A IMMEDIATE. Operand is negated (two's complemented) and added as a 16-bit word to A register. OV is set if arithmetic overflow occurs.

SXI — SUBTRACT FROM X IMMEDIATE. Operand is negated (two's complemented) and added as a 16-bit word to X register. OV is set if arithmetic overflow occurs.

CAI — COMPARE TO A IMMEDIATE. Operand is compared to contents of LS byte of A register. If unequal, a one word skip occurs. If equal, next sequential instruction is executed. Contents of A register are not disturbed. MS byte of A register does not take part in comparison.

CXI — COMPARE TO X IMMEDIATE. Operand is compared to contents of LS byte of X register. If unequal, a one word skip occurs. If equal, next sequential instruction is executed. Contents of X register are not distrubed. MS byte of X register does not take part in comparison.

LAP — LOAD A POSITIVE IMMEDIATE. Operand is loaded into LS byte of A register. MS byte of A register is set to zero.

LXP — LOAD X POSITIVE IMMEDIATE. Operand is loaded into LS byte of X register. MS byte of X register is set to zero.

LAM — LOAD A MINUS IMMEDIATE. The operand is negated (two's complemented) and loaded as a 16-bit word into the A register.

LXM — LOAD X MINUS IMMEDIATE. The operand is negated (two's complemented) and loaded as a 16-bit word into the X register.

4.6 CONDITIONAL JUMP INSTRUCTIONS

4.6.1 Format

Conditional Jump instructions test conditions within the computer and perform program branches depending on the results of the test. A jump occurs if the specified conditions are satisfied. All branches are direct and relative to the P register (location of the Conditional Jump instruction). The range of Conditional Jump instructions is:

Forward Jumps:	P + 1 through P + 64
Backward Jumps:	P through P - 63

4.6.2 Microcoding

A general code, JOC, for Jump On Condition, is provided so the programmer can microcode jump conditions. There are five different conditions which may be tested individually or in combination:

1. Sign of A (positive or negative)
2. Contents of A (zero or not zero)
3. Contents of X (zero or not zero)
4. Overflow indicator (set or reset)
5. SENSE indicator (on or off)

The conditions may be tested individually or in combination. Figure 4-15 shows the format for the JOC instruction:

[LABEL]	JOC	EXPRESSION 1, EXPRESSION 2	[COMMENTS]

EXPRESSION 1: must be absolute and in the range : 0 thru : 3F
EXPRESSION 2: must represent a location within -63 thru +64 computer words.

Figure 4-15. JOC Jump On Condition Format

JOC commands consist of two groups, the AND group and the OR group. The AND test group requires that all of the test conditions specified by bits 0 through 4 of Expression 1 be true for the jump to take place. The OR group requires that any one or more of the test conditions specified be true if the jump is to take place. Expression 1 consists of 6 bits (T0 through T5) as defined by figure 4-16. Bit T5 specifies which test group is used. Bits T0 through T4 specify inclusion of a specific test condition if equal to 1. If equal to 0, the associated test condition is not examined.

IR07-IR12 = T_0-T_5

JOC :XX,ADR

T_5 T_4 T_3 T_2 T_1 T_0

	AND GROUP (T_5 = 1)	OR GROUP (T_5 = 0)
T_4 = 1	X ≠ 0	X = 0
T_3 = 1	SENSE on	SENSE off
T_2 = 1	OV reset	OV set (resets OV)
T_1 = 1	A ≠ 0	A = 0
T_0 = 1	A positive	A negative

Figure 4-16. JOC Expression 1 Definitions

The following Conditional Jump instructions are special cases of the general JOC instruction. Since they are utilized more often than the general conditional jumps, they have been given their own mnemonics. Figure 4-17 illustrates the general format for the Conditional Jump instructions.

[LABEL] OP-CODE EXPRESSION [COMMENTS]

EXPRESSION: must represent a location within -63 thru +64 computer words.

Figure 4-17. Conditional Jump Format

4.6.3 Arithmetic Conditional Jump Instructions

JAG JUMP IF A GREATER THAN ZERO. Jump occurs if contents of A register are greater than zero.

JAP JUMP IF A POSITIVE. Jump occurs if contents of A register are greater than or equal to zero (A_{15} = 0).

JAZ JUMP IF A ZERO. Jump occurs if contents of A register are zero.

JAN JUMP IF A NOT ZERO. Jump occurs if contents of A register are not zero.

JAL JUMP IF A LESS THAN OR EQUAL TO ZERO. Jump occurs if contents of A register are less than or equal to zero.

JAM JUMP IF A MINUS. Jump occurs if contents of A register are less than zero (A_{15} = 1).

JXZ JUMP IF X ZERO. Jump occurs if contents of X register are zero.

JXN JUMP IF X NOT ZERO. Jump occurs if contents of X register are not zero.

4.6.4 Control Conditional Jump Instructions

JSS JUMP IF SENSE INDICATOR SET. Jump occurs if SENSE indicator is on.

JSR JUMP IF SENSE INDICATOR RESET. Jump occurs if SENSE indicator is off.

JOS JUMP IF OVERFLOW SET. Jump occurs if OV equal one. OV is reset to zero during jump.

JOR JUMP IF OVERFLOW RESET. Jump occurs if OV equal zero.

4.7 SHIFT INSTRUCTIONS

4.7.1 Operand Restrictions and Instruction Format

Shift instructions move bit patterns in the computer registers either right or left. Shifts may involve a single register (A or X), a single register and the overflow (OV) indicator, or both the A and X registers and the OV indicator. The Processor provides logical, arithmetic and rotate shifts. The operands (n) for single register and double register instructions can be any absolute value from 1 through 8 and 16, respectively. The single register shift instruction format is shown in figure 4-18 and the instruction format for double register (long) shifts is shown in figure 4-19.

[LABEL] OP-CODE EXPRESSION [COMMENTS]

EXPRESSION: must be absolute and in the range 1 thru 8.

Figure 4-18. Single Register Shift Format

[LABEL] OP-CODE EXPRESSION [COMMENTS]

EXPRESSION: must be absolute and in the range 1 thru 16.

Figure 4-19. Double Register (Long) Shift Format

4.7.2 Arithmetic Shift Instructions

The shift paths for the arithmetic shift instructions are illustrated below in figures 4-20 and 4-21.

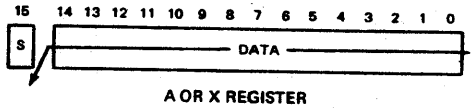


Figure 4-20. Arithmetic Left Shift

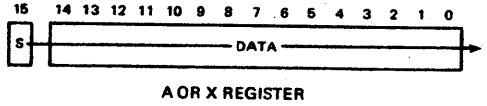


Figure 4-21. Arithmetic Right Shift

ALA ARITHMETIC SHIFT A LEFT. Contents of A register (bits 0-14) are shifted left n places. The sign bit (bit 15) is unchanged. Zeros are shifted into bit 0 and bits shifted out of bit 14 are lost.

ALX ARITHMETIC SHIFT X LEFT. Contents of X register (bits 0-14) are shifted left n places. The sign bit (bit 15) is unchanged. Zeros are shifted into bit 0 and bits shifted out of bit 14 are lost.

ARA ARITHMETIC SHIFT A RIGHT. Contents of A register are shifted right n places. The sign bit (bit 15) is unchanged and is shifted into and propagated through bit 14. Bits shifted out of bit 0 are lost.

ARX ARITHMETIC SHIFT X RIGHT. Contents of X register are shifted right n places. The sign bit (bit 15) is unchanged and is shifted into and propagated through bit 14. Bits shifted out of bit 0 are lost.

4.7.3 Logical Shift Instructions

The shift paths for the logical shift instructions are illustrated below in figures 4-22 and 4-23.

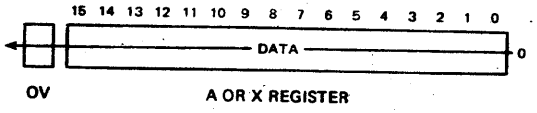


Figure 4-22. Logical Left Shift

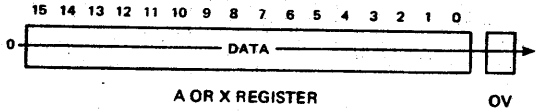


Figure 4-23. Logical Right Shift

LLA LOGICAL SHIFT A LEFT. Contents of A register are shifted left n places through OV. Zeros are shifted into bit 0. Bits are shifted from bit 15 of A into OV. Bits shifted out of OV are lost. A and OV act as a 17-bit register.

LLX LOGICAL SHIFT X LEFT. Confents of X register are shifted left n places through OV. Zeros are shifted into bit 0. Bits are shifted from bit 15 of X into OV. Bits shifted out of OV are lost. X and OV act as a 17-bit register.

LRA LOGICAL SHIFT A RIGHT. Contents of A register are shifted right n places through OV. Zeros are shifted into bit 15. Bits are shifted from bit 0 of A into OV. Bits shifted out of OV are lost. A and OV act as a 17-bit register.

LRX LOGICAL SHIFT X RIGHT. Contents of X register are shifted right n places through OV. Zeros are shifted into bit 15. Bits are shifted from bit 0 of X into OV. Bits shifted out of OV are lost. X and OV act as a 17-bit register.

4.7.4 Rotate Shift Instructions

The shift paths for the rotate shift instructions are illustrated below in figures 4-24 and 4-25.

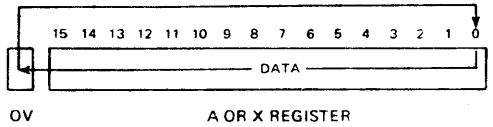


Figure 4-24. Rotate Left Shift

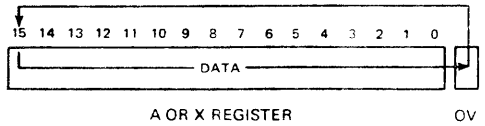


Figure 4-25. Rotate Right Shift

RLA ROTATE A LEFT WITH OVERFLOW. Contents of A register are shifted left n places through OV. OV is shifted into bit 0 and bit 15 is shifted into OV. No bits are lost when this shift is executed. A and OV act as a 17-bit register.

RLX ROTATE X LEFT WITH OVERFLOW. Contents of X register are shifted left n places through OV. OV is shifted into bit 0 and bit 15 is shifted into OV. No bits are lost when this shift is executed. X and OV act as a 17-bit register.

RRA ROTATE A RIGHT WITH OVERFLOW. Contents of A register are shifted right n places through OV. OV is shifted into bit 15 and bit 0 is shifted into OV. No bits are lost when this shift is executed. A and OV act as a 17-bit register.

RRX ROTATE X RIGHT WITH OVERFLOW. Contents of X register are shifted right n places through OV. OV is shifted into bit 15 and bit 0 is shifted into OV. No bits are lost when this shift is executed. X and OV act as a 17-bit register.

4.7.5 Double Register (Long) Logical Shift Instructions

The shift paths for the Long Logical Shift instructions are shown below in figures 4-26 and 4-27.

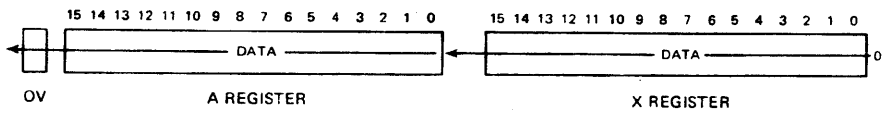


Figure 4-26. Long Left Shift

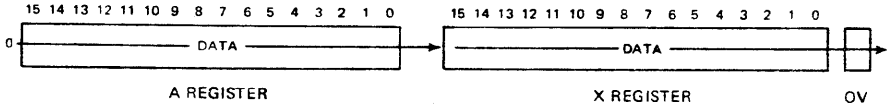


Figure 4-27. Long Right Shift

LLL LONG LOGICAL SHIFT LEFT. Contents of A and X registers are logically shifted left n places through OV. Zeros are shifted into bit 0 of X register. Bits shifted from bit 15 of X enter bit 0 of A, and from bit 15 of A they enter OV. Bits shifted out of OV are lost. A, X and OV act as a 33-bit register.

LLR LONG LOGICAL SHIFT RIGHT. Contents of A and X registers are logically shifted right n places through OV. Zeros are shifted into bit 15 of A register. Bits shifted from bit 0 of A enter bit 15 of X, and from bit 0 of X they enter OV. Bits shifted out of OV are lost. A, X and OV act as a 33-bit register.

4.7.6 Double Register (Long) Rotate Shift Instructions

Shift paths for the Long Rotate Shift instructions are shown below in figures 4-28 and 4-29:

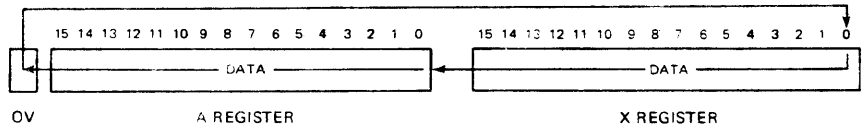


Figure 4-28. Long Rotate Left Shift

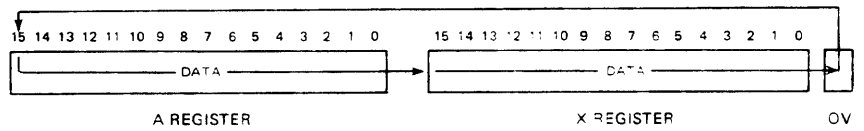


Figure 4-29. Long Rotate Right Shift

LRL LONG ROTATE LEFT. Contents of A and X registers are shifted left n places through OV. OV is shifted into bit 0 of X register. Bits shifted from bit 15 of X enter bit 0 of A, and from bit 15 of A they enter OV. No bits are lost when this shift is executed. A, X and OV act as a 33-bit register.

LRR LONG ROTATE RIGHT. Contents of A and X registers are shifted right n places through OV. OV is shifted into bit 15 of A register. Bits shifted from bit 0 of A enter bit 15 of X, and from bit 0 of X they enter OV. No bits are lost when this shift is executed. A, X and OV act as a 33-bit register.

4.8 REGISTER CHANGE INSTRUCTIONS

4.8.1 Format

Register change instructions perform arithmetic and logical operations involving the A register, the X register and/or the OV indicator. The Register Change instruction format is shown in figure 4-30.

[LABEL]	OP-CODE	[EXPRESSION]	[COMMENTS]

EXPRESSION: there is no expression in the Operand field except for the BAO and BXO instructions where it must be absolute and in the range 0 thru 15.

Figure 4-30. Register Change Format

4.8.2. A Register Change Instructions

ARM A REGISTER TO MINUS ONE. Sets contents of A register to -1 (:FFFF).

ARP A REGISTER TO PLUS ONE. Sets contents of A register to +1.

CAR COMPLEMENT A REGISTER. Performs one's complement on contents of A register.

DAR DECREMENT A REGISTER. Subtracts one from contents of A register. OV is set if arithmetic overflow occurs.

IAR INCREMENT A REGISTER. Adds one to contents of A register. OV is set if arithmetic overflow occurs.

NAR NEGATE A REGISTER. Performs two's complement on contents of A register. OV is set if arithmetic overflow occurs.

ZAR ZERO A REGISTER. Sets contents of A register to zero.

4.8.3 X Register Change Instructions

ZXR ZERO X REGISTER. Sets contents of X register to zero.

XRP X REGISTER TO PLUS ONE. Sets contents of X register to +1.

XRM X REGISTER TO MINUS ONE. Sets contents of X register to -1 (:FFFF).

CXR COMPLEMENT X REGISTER. Performs one's complement on contents of X register.

NXR NEGATE X REGISTER. Performs two's complement on contents of X register. OV is set if arithmetic overflow occurs.

IXR INCREMENT X REGISTER. Adds one to contents of X register. OV is set if arithmetic overflow occurs.

DXR DECREMENT X REGISTER. Subtracts one from contents of X register. OV is set if arithmetic overflow occurs.

4.8.4 OV Register Change Instructions

SOV SET OVERFLOW. Sets OV indicator (=1).

ROV RESET OVERFLOW. Resets OV indicator (=0).

COV COMPLEMENT OVERFLOW. Complements OV.

SAO SIGN OF A TO OVERFLOW. Bit 15 of A register is copied into OV. A register remains unchanged.

SXO SIGN OF X TO OVERFLOW. Bit 15 of X register is copied into OV. X register remains unchanged.

LAO LSB OF A TO OVERFLOW. Bit 0 of A register is copied into OV. A register remains unchanged.

LXO LSB OF X TO OVERFLOW. Bit 0 of X register is copied into OV. X register remains unchanged.

BAO BIT OF A TO OVERFLOW. Bit n of A register is copied into OV. A register remains unchanged. Bit n is specified in Operand field.

BXO BIT OF X TO OVERFLOW. Bit n of X register is copied into OV. X register remains unchanged. Bit n is specified in Operand field.

4.8.5 Multi-Register Change Instructions

ZAX ZERO A AND X. Sets contents of A and X registers to zero.

AXP A AND X REGISTERS TO PLUS ONE. Sets contents of A and X registers to +1.

AXM A AND X REGISTERS TO MINUS ONE. Sets contents of A and X registers to -1 (:FFFF).

TAX TRANSFER A TO X. Transfers contents of A register to X register. A register remains unchanged.

TXA TRANSFER X TO A. Transfers contents of X register to A register. X register remains unchanged.

EAX EXCHANGE A AND X. Exchanges contents of A and X registers.

ANA AND OF A AND X TO A. Contents of A and X registers are logically ANDed. Result replaces contents of A register. X register remains unchanged.

ANX AND OF A AND X TO X. Contents of A and X registers are logically ANDed. Result replaces contents of X register. A register remains unchanged.

NRA NOR OF A AND X TO A. Contents of A and X registers are logically NORed Result replaces contents of A register. X register remains unchanged.

NRX NOR OF A AND X TO X. Contents of A and X registers are logically NORed Result replaces contents of X register. A register remains unchanged.

CAX COMPLEMENT OF A TO X. Performs one's complement on contents of A register and places result in X register. A register remains unchanged.

CXA COMPLEMENT OF X TO A. Performs one's complement on contents of X register and places result in A register. X register remains unchanged.

NAX NEGATE A TO X. Performs two's complement on contents of A register and places result in X register. A register remains unchanged. OV is set if arithmetic overflow occurs.

NXA NEGATE X TO A. Performs two's complement on contents of X register and places result in A register. X register remains unchanged. OV is set if arithmetic overflow occurs.

IAX INCREMENT A TO X. Adds one to contents of A register and places result in X register. A register remains unchanged. OV is set if arithmetic overflow occurs.

IXA INCREMENT X TO A. Adds one to contents of X register and places result in A register. X register remains unchanged. OV is set if arithmetic overflow occurs.

IPX INCREMENT P TO X. Adds two to current program counter (address of IPX) and places result in X register. P is then incremented for the next instruction fetch. Example:


```
                .
                .
(P)             IPX             Place P+2 in X
(P+1)           JMP     ROUT    Jump to routine with address of GO in X
(P+2)   GO      EQU     $
                .
                .
                .
        ROUT    EQU     $       Subroutine starts here
                .
                .
                .
                JMP     @0      Return to GO
```

DAX DECREMENT A TO X. Subtracts one from contents of A register and places result in X register. A register remains unchanged. OV is set if arithmetic overflow occurs.

DXA DECREMENT X TO A. Subtracts one from contents of X register and places result in A register. X register remains unchanged. OV is set if arithmetic overflow occurs.

4.8.6 Extended Multi-Register Change Instructions (LSI-2 Only)

BCA BIT CLEAR A. The contents of the X register are ones complemented and then logically ANDed with the contents of the A register. The result replaces A and the original value of X is left unchanged.

BCX BIT CLEAR X. The contents of the X register are ones complemented and then logically ANDed with the contents of the A register. The result replaces X and the original value of A is left unchanged.

BSA BIT SET A. Contents of X register are logically ORed with contents of A register. Result is placed in A register and X register remains unchanged.

BSX BIT SET X. Contents of A register are logically ORed with contents of X register. Result is placed in X register and A register remains unchanged.

EIX Execute instruction pointed to by X. Instruction contained in location addressed by contents of X register is executed immediately following EIX instruction. Next sequential instruction following EIX instruction is skipped.

Note the following:

1. If the executed instruction is a multi-word instruction, the second and succeeding words of the instruction must be located at the second location after the EIX instruction (EIX+2).

2. If the executed instruction modifies the P register, the modification is relative to location EIX+1.

3. If the executed instruction is a SCM or conditional I/O instruction, the location following the EIX instruction (EIX+1) should be coded with a JMP $-1. This is required for recovery purposes in the event of an interrupt or the lack of a true sense response.

4. EIX is not interruptable.

4.8.7 Console Register Instructions

IAH INPUT CONSOLE DATA REGISTER TO A AND HALT. Contents of Console Data register are loaded into A register. Computer then halts.

IXH INPUT CONSOLE DATA REGISTER TO X AND HALT. Contents of Console Data register are loaded into X register. Computer then halts.

ICA INPUT CONSOLE DATA REGISTER TO A. Contents of Console Data register are loaded into A register.

ICX INPUT CONSOLE DATA REGISTER TO X. Contents of Console Data register are loaded into X register.

IIH INPUT CONSOLE DATA REGISTER TO I AND HALT. Contents of Console Data register are loaded into I register. Computer then halts.

IMH INPUT CONSOLE DATA REGISTER TO MEMORY AND HALT. Contents of Console Data register are stored into memory location following IMH instruction. Computer halts with P register set to location following modified memory location.

IPH INPUT CONSOLE DATA REGISTER TO P AND HALT. Contents of Console Data register are loaded into P register. Computer then halts. When RUN is depressed, execution of the program will begin at address just input to P register.

ISA INPUT CONSOLE SENSE REGISTER TO A. Four-bit contents of Console Sense register are loaded into least significant 4 bits of A register. Most significant 12 bits of A register are set to zero.

ISX INPUT CONSOLE SENSE REGISTER TO X. Four-bit contents of Console Sense register are loaded into least significant 4 bits of X register. Most significant 12 bits of X register are set to zero.

OAH OUTPUT A TO CONSOLE DATA REGISTER AND HALT. Contents of A register are loaded into Console Data register. Computer then halts.

OXH OUTPUT X TO CONSOLE DATA REGISTER AND HALT. Contents of X register are loaded into Console Data register. Computer then halts.

OCA OUTPUT A TO CONSOLE DATA REGISTER. Contents of A register are loaded into Console Data register.

OCX OUTPUT X TO CONSOLE DATA REGISTER. Contents of X register are loaded into Console Data register.

OLH OUTPUT LOCATION TO CONSOLE DATA REGISTER AND HALT. Location of OLH instruction is loaded into Console Data register. Computer then halts.

OMH OUTPUT MEMORY TO CONSOLE DATA REGISTER AND HALT. Contents of memory location following OMH instruction are loaded into Console Data register. Computer halts with P register set to location following output memory location (OMH instruction +2).

OPH OUTPUT P TO CONSOLE DATA REGISTER AND HALT. Contents of P register (address of OPH instruction +1) are loaded into Console Data register. Computer then halts.

4.9 CONTROL INSTRUCTIONS

4.9.1 Format

Control instructions are used for general status manipulation in the computer. The general format for these instructions is shown in figure 4-31.

[LABEL]	OP-CODE	[EXPRESSION]	[COMMENTS]

There is no expression in the Operand field, except for the SIN and STOP instructions.
For SIN, the expression must be absolute and in the range 1 thru 6.
For STOP, the expression must be absolute and in the range 1 thru 255.

Figure 4-31. Control Format

4.9.2 Processor Control Instructions

HLT HALT. Halts the computer.

NOP NO OPERATION. Performs no active function. Normally used to reserve space for other instructions.

STOP HALT WITH OPERAND. Halts computer with specified operand occupying least significant 8 bits of I (instruction) register. Operand may be any absolute expression in the range 0 through 255. As an example, STOP 5 would halt with : 0805 in I register.

WAIT WAIT FOR INTERRUPT. Executes as JMP $. Program loops on one location waiting for an interrupt. After interrupt is serviced, return is made to WAIT instruction to await further interrupts.

4.9.3 Mode Control Instructions

SBM SET BYTE MODE. Conditions computer to address byte (8 bit) operands rather than word operands when executing Memory Reference instructions (see paragraph 4.2.2).

SWM SET WORD MODE. Conditions computer to address word (16 bit) operands rather than byte operands when executing Memory Reference instructions (see paragraph 4.2.1). "Reset" condition of computer is Word mode.

4.9.4 Status Control Instructions

The format of the 8-bit Computer Status word is shown in figure 4-32:

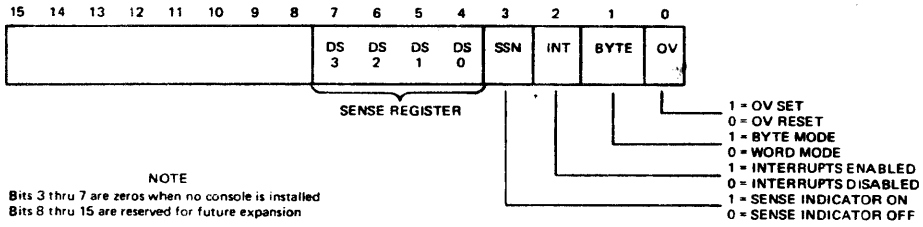


Figure 4-32. Computer Status Word Format

SIN STATUS INHIBIT. Inhibits interrupts and places computer in Word mode for number of succeeding instructions specified by operand. Operand may be any absolute expression in range 1 through 6. As an example, execution of SIN 4 instruction will force Word mode operation for four succeeding instructions and will inhibit interrupt acknowledgement until after completion of five succeeding instructions since interrupts are serviced at end of instruction execution.

NOTE

The following should be noted when using the SIN instruction in the LSI-2 computer.

1. Do not place a HLT instruction within a SIN instruction range.

2. Do not attempt to step through a SIN range when the computer is in Step mode. If an instruction sequence which falls within a SIN range must be examined, press the RESET pushbutton first to clear the SIN counter. The sequence can then be stepped through. Note that the computer will revert to the Word mode.

SIA STATUS INPUT TO A. Computer Status word is loaded into LS byte of A register. Resets OV and places computer in Word mode. State of interrupts is unchanged. MS byte of A register is set to zero.

SIX STATUS INPUT TO X. Computer Status word is loaded into MS byte of X register. Resets OV and places computer in Word mode. State of interrupts is unchanged. MS byte of X register is set to zero.

SOA STATUS OUTPUT FROM A. Least significant byte of A register is loaded into computer Status register. This instruction does not alter Interrupt Enable flag.

SOX STATUS OUTPUT FROM X. Least significant byte of X register is loaded into computer status register. This instruction does not alter Interrupt Enable flag.

4.9.5 Interrupt Control Instructions

EIN ENABLE INTERRUPTS. Enables recognition of external interrupts by the computer. Interrupts will not be serviced for a minimum of one instruction time following EIN and possibly as long as three instruction times (maximum).

DIN DISABLE INTERRUPTS. Prevents Processor from responding to any interrupts. A special jumper option on processor option board allows Power Fail, Console and Trap interrupt operation independent of DIN.

CIE CONSOLE INTERRUPT ENABLE. Enables Console interrupts. Console interrupts are generated each time INT switch is pressed when computer is in RUN mode. Console interrupts are also under control of EIN/DIN instructions. A special jumper option on processor option board allows Console interrupts to be enabled independent of EIN/DIN instructions. Console interrupts are disabled when a Console interrupt or TRAP is serviced.

CID CONSOLE INTERRUPT DISABLE. Disables Console interrupts.

PFE POWER FAIL INTERRUPT ENABLE. When option placing Power Fail interrupt outside EIN and DIN control is selected, Power Fail Interrupt Enable (PFE) instruction allows recognition of Power Fail interrupts. If Power Fail interrupts were disabled at issuance of PFE, PFE does not take effect until after two succeeding instructions have been executed.

PFD POWER FAIL INTERRUPT DISABLE. When option placing Power Fail interrupts outside EIN and DIN control is selected, Power Fail Interrupt Disable (PFD) instruction inhibits recognition of Power Fail interrupts.

TRP TRAP. Generates an interrupt to Console interrupt location if interrupts are enabled or if special jumper option placing Power Fail, Console and Trap interrupts outside EIN/DIN control is in use. In latter case, there is no enable or disable instruction associated with Trap interrupts. Console interrupt is disabled when TRAP is serviced. Interrupts will not be serviced for a minimum of one instruction time following TRP.

4.10 INPUT/OUTPUT INSTRUCTIONS

Input/Output instructions are either single word or multiple word instructions. All single word instructions use the same format (see figure 4-33). Multiple word formats are described separately in paragraphs 4.10.4 and 4.10.5. All I/O instructions have 8 bits available for addressing a particular peripheral device and a particular register or function within a device. These 8 bits are arbitrarily divided into a 5-bit Device Address field to address one of 32 devices and a 3-bit Function Code field to specify one of 8 registers or functions within a device. The device address and function code may be expressed as either one or two self-defined (i.e., numeric expressions) or absolute expressions. If a single expression is used, it must be in the range : 0 through : FF and it represents both the device address and function code. If two expressions are used, the first must be the device address in the range : 0 through : 1F and the second must be the function code in the range: 0 through : 7.

[LABEL] OP-CODE EXPRESSION 1[,EXPRESSION 2] [COMMENTS]

If EXPRESSION 2 is not present, EXPRESSION 1 must be absolute and in the range : 0 through : FF.
If EXPRESSION 2 is present, EXPRESSION 1 must be absolute and in the range : 0 through : 1F.
EXPRESSION 2 must be absolute and the range : 0 through : 7.

Figure 4-33. Single Word Input/Output Instruction Format

Both Word and Byte I/O instructions are available. Whether a full 16-bit word or an 8-bit byte is transferred depends upon the instruction used and is not affected by the word/byte addressing mode flip-flop (SWM/SBM) used by Memory Reference instructions.

4.10.1 Control Input/Output Instructions

The Control I/O instructions are divided into Sense and Select instructions. Sense instructions are used to test the status of a function within the addressed peripheral device. Select instructions are used to control the operation of specific functions within the addressed peripheral device. The functions tested or controlled depend upon the individual peripheral device. Control I/O instructions use the Single Word I/O instruction format shown in figure 4-33.

4.10.1.1 Sense Instructions

SEN SENSE AND SKIP ON RESPONSE. Tests specified function in addressed peripheral device. If a true response is obtained, next sequential instruction is skipped. If a false response is obtained, next sequential instruction is executed.

SSN SENSE AND SKIP ON NO RESPONSE. Tests specified function in addressed peripheral device. If a false response is obtained, next sequential instruction is skipped. If a true repsonse is obtained, next sequential instruction is executed.

4.10.1.2 Select Instructions

SEL SELECT FUNCTION. Transmits specified function code to addressed peripheral device along with a Select Control signal. All zeros are placed on Data bus. Any action generated is a function of peripheral device interface design.

SEA SELECT AND PRESENT A. Transmits specified function code to addressed peripheral device along with a Select Control signal. Contents of A register are placed on Data bus. Any action generated is a function of peripheral device interface design.

SEX SELECT AND PRESENT X. Transmits specified function code to addressed peripheral device along with a Select Control signal. Contents of X register are placed on Data bus. Any action generated is a function of peripheral device interface design.

NOTE

When a Select type instruction is used to turn off interrupts that may be pending, it should be preceded by a SIN 1 instruction to disable Processor recognition of the pending interrupt. This is necessary since the Processor examines interrupt requests prior to the Select taking effect and will therefore respond to the interrupt even though it is no longer pending.

4.10.2 Word Input/Output Instructions

Word I/O instructions transmit 16 bits of data at a time. They are divided into Unconditional and Conditional instructions. Conditional instructions are automatically repeated until a true sense response is obtained, at which time the data transmission occurs and the next instruction in sequence is executed. Response to an interrupt may occur "within" a conditional I/O instruction - i.e., during a false sense response an interrupt can be acknowledged and the computer will return to execution of the conditional I/O instruction after servicing the interrupt. If a word input is requested from an 8-bit device, the upper 8 bits will be input as zeros. If an output is performed to an 8-bit device, the upper 8 bits will be ignored by the device.

4.10.2.1 Unconditional Word Input/Output Instructions

INA — INPUT TO A REGISTER. Unconditionally transfers a full 16-bit data word from addressed peripheral device to A register.

INAM — INPUT TO A REGISTER MASKED. Unconditionally transfers a full 16-bit data word from addressed peripheral device to Processor and logically ANDs data word with contents of A register. Result replaces contents of A register.

INX — INPUT TO X REGISTER. Unconditionally transfers a full 16-bit data word from addressed peripheral device to X register.

INXM — INPUT TO X REGISTER MASKED. Unconditionally transfers a full 16-bit data word from addressed peripheral device to Processor, and logically ANDs data word with contents of X register. Result replaces contents X register.

OTA — OUTPUT A REGISTER. Unconditionally transfers full 16-bit contents of A register to addressed peripheral device.

OTX — OUTPUT X REGISTER. Unconditionally transfers full 16-bit contents of X register to addressed peripheral device.

OTZ — OUTPUT ZERO. Unconditionally transfers a 16-bit word containing all zeros to addressed peripheral device.

4.10.2.2 Conditional Word Input/Output Instructions

RDA — READ WORD TO A REGISTER. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit data word is transferred from addressed device to A register.

RDAM — READ WORD TO A REGISTER MASKED. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit data word is transferred from addressed device to Processor and logically ANDed with contents of A register. Result replaces contents of A register.

RDX — READ WORD TO X REGISTER. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit data word is transferred from addressed device to X register.

RDXM — READ WORD TO X REGISTER MASKED. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit data word is transferred from addressed device to Processor and logically ANDed with contents of X register. Result replaces contents of X register.

WRA — WRITE FROM A REGISTER. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, full 16-bit contents of A register are transferred to addressed device.

WRX — WRITE FROM X REGISTER. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, full 16-bit contents of X register are transferred to addressed device.

WRZ — WRITE ZERO. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, a 16-bit word containing all zeros is transferred to addressed device.

4.10.3 Byte Input Instructions

Byte Input instructions input 8 bits of data to the LS byte of a target register leaving the MS byte unchanged. They are divided into Unconditional and Conditional instructions. Conditional instructions are automatically repeated until a true sense response is obtained, at which time the data transmission occurs and the next instruction in sequence is executed. Response to an interrupt may occur "within" a Conditional Byte Input instruction - i.e., during a false sense response an interrupt can be acknowledged and the computer will return to execution of the conditional instruction after serviceing the interrupt. Byte Input instructions use the Single Word Input/Output instruction format as shown in figure 4-33.

4.10.3.1 Unconditional Byte Input Instructions

IBA — INPUT BYTE TO A REGISTER. Unconditionally transfers an 8-bit data byte from addressed peripheral device to LS byte of A register. MS byte of A register remains unchanged.

IBAM — INPUT BYTE TO A REGISTER MASKED. Unconditionally transfers an 8-bit data byte from addressed peripheral device to Processor and logically ANDs data byte with contents of LS byte of A register. Result replaces LS byte of A register and MS byte of A register remains unchanged.

IBX INPUT BYTE TO X REGISTER. Unconditionally transfers an 8-bit data byte from addressed peripheral device to LS byte of X register. MS byte of X register remains unchanged.

IBXM INPUT BYTE TO X REGISTER MASKED. Unconditionally transfers an 8-bit data byte from address peripheral device to Processor and logically ANDs data byte with contents of LS byte of X register. Result replaces LS byte of X register and MS byte of X register remains unchanged.

4.10.3.2 Conditional Byte Input Instructions

RBA READ BYTE TO A REGISTER. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred from addressed device to LS byte of A register and MS byte of A register remains unchanged.

RBAM READ BYTE TO A REGISTER MASKED. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred from addressed device to Processor and logically ANDed with contents of LS byte of A register. Result replaces LS byte of A register and MS byte of A register remains unchanged.

RBX READ BYTE TO X REGISTER. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred from addressed device to LS byte of X register. MS byte of X register remains unchanged.

RBXM READ BYTE TO X REGISTER MASKED. Tests specified function in addressed peripheral device. If a false response is received, instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred from addressed device to Processor and logically ANDed with contents of LS byte of X register. Result replaces LS byte of X register and MS byte of X register remains unchanged.

4.10.4 Block Input/Output Instructions

The two instructions in this class provide for high-speed, full 16-bit data word transfers between Memory and the addressed peripheral device. The Processor is totally dedicated to these instructions until the specified block of data has been completely transferred - i.e., no interrupts may be serviced until the instructions have been executed to completion.

The Block Transfer instructions are double-word instructions. The second word of the instruction contains the base address minus one of the associated memory data buffer. The X register contains the (positive) number of words to be transferred - i.e., the length of the data buffer. The memory location of each word transferred is obtained by summing the base address minus one and the contents of the X register. As each data word is transmitted, the X register is decremented by one. Thus, the data buffer is output or input in descending order, beginning with the highest memory location and ending with the lowest memory location (base address plus length -1). When the X register is decremented to zero, the next instruction in sequence is executed.

The format for the Block Transfer instructions is shown in figure 4-34.

[LABEL]	OP-CODE	EXPRESSION 1 [EXPRESSION 2]	[COMMENTS]
[LABEL]	DATA	EXPRESSION 3	[COMMENTS]

If EXPRESSION 2 is not present, EXPRESSION 1 must be absolute and in the range : 0 thru : FF.
If EXPRESSION 2 is present, EXPRESSION 1 must be absolute and in the range : 0 thru : 1F.
EXPRESSION 2 must be absolute and in the range : 0 thru : 7.
EXPRESSION 3 is an absolute or relocatable expression giving the base address -1 of the buffer.

Figure 4-34. Block Input/Output Instruction Format

The expressions in the Operand field of these instructions must be either self-defining (i.e., numeric expressions) or absolute expressions. If only one expression is present, it must be in the range : 0 through : FF. The high-order 5 bits represent the peripheral device address and the low-order 3 bits represent the function code. If two expressions are present, the first must be in the range : 0 through : 1F and the second must be in the range : 0 through : 7. The first expression represents a peripheral device address, and the second expression represents a function code.

The expression in the Operand field of the DATA statement must not be an indirect address (no*). It represents the memory location less one (low-order memory location) of the data buffer.

BIN BLOCK IN. Tests specified function in addressed peripheral device and transfers a full 16-bit data word from addressed device to memory data buffer each time a true sense response is received. Instruction executes until all data words have been input. Interrupts are not acknowledged until completion of instruction.

BOT **BLOCK OUT. Tests specified function in addressed peripheral device and transfers a full 16-bit data word from memory data buffer to addressed device each time a true sense response is received. Instruction executes until all data words have been output. Interrupts are not acknowledged until completion of instruction.**

4.10.5 Automatic Input/Output Instructions

The Automatic Input/Output instructions (Auto I/O) provide data transfers directly between Memory and peripheral devices without affecting the A and X registers. These multiple word instructions effectively constitute complete I/O subroutines, thus facilitating their use as interrupt instructions. They increment a (negative) data word or byte counter, increment a data word or byte pointer and transfer a data word or byte between Memory and a peripheral device.

Each Auto I/O instruction occupies three words in Memory. The first word contains the instruction itself, the second word contains the two's complement (negative) of the word or byte count for the data buffer, and the third word contains an address pointer specifying the address minus one, of the first (lower-order memory) location in the memory data buffer. The data buffer is input or output in order of ascending memory locations (low-order to high-order). The format for these instructions is shown in figure 4-35.

[LABEL]	OP-CODE	EXPRESSION 1 [, EXPRESSION 2]	[COMMENTS]
[LABEL]	DATA	EXPRESSION 3	[COMMENTS]
[LABEL]	{ BAC or DATA }	EXPRESSION 4	[COMMENTS]

If EXPRESSION 2 is not present, EXPRESSION 1 must be absolute and in the range : 0 thru : FF.
IF EXPRESSION 2 is present, EXPRESSION 1 must be present and in the range : 0 thru : 1F.
EXPRESSION 2 must be absolute and in the range : 0 thru : 7.
EXPRESSION 3 is the negative word or byte count of the data buffer.
EXPRESSION 4 is an absolute or relocatable expression defining the base address -1 of the buffer.

Figure 4-35. Automatic Input/Output Instruction Format

The expressions in the Operand fields of the first two statements must be either self-defined (i.e., numeric expressions) or absolute expressions. If only one expression is present in the operand field of the instruction, it must be in the range : 0 through : FF. The high-order 5 bits represent the device address and the low order 3 bits represent the function code. If two expressions are present, the first must be in the range : 0 through : 1F, and the second must be in the range : 0 through : 7. The first expression represents a peripheral device address, and the second expression represents a function code.

The absolute expression for the second word represents the negative (two's complement) data word or byte count for the data buffer. This word is incremented once prior to each data word or byte transfer and must be preset each time a block of data is to be transferred.

The expression in the Operand field of the third word of the instruction is an address pointer specifying the byte or word address minus one, of the data buffer starting location. This word is incremented once prior to each data word or byte transferred and must be preset each time a block of data is to be transferred.

Operation of Auto I/O instructions differs depending upon usage. When used as an in-line program instruction, the Auto I/O instruction sequence is as shown in figure 4-36. Each time the instruction is executed, the word/byte count and address pointer are incremented, one word or byte of data is transferred, and then the incremented word count is examined. If the word count has not yet reached zero, the next instruction executed is from location P+4. If the word count reached zero, the next instruction executed is at location P+3 (End-of-Block exit location). Since Auto I/O instructions do not sense for the peripheral device to be ready prior to data transfer, a Sense (SEN) instruction should be used prior to each execution (one word transferred) of the instruction, i.e., to transfer a block location, P+4 would normally contain a jump back to a Sense instruction prior to location P.

P	Automatic I/O Instruction
P+1	Word/Byte Counter (negative)
P+2	Word/Byte Address Pointer (start address -1)
P+3	End-of-Block Exit (Word Count = 0)
P+4	Next Instruction (Word Count ≠ 0)

Figure 4-36. In-line Auto I/O Instruction Sequence

Auto I/O instructons may also be used under interrupt control at an interrupt location to implement a Direct Memory channel. In this application, the Auto I/O instruction is executed once each time the peripheral device indicates that it is ready for a data transfer by interrupting to the location containing the Auto I/O instruction. Since the Auto I/O instructions do not alter any processor registers, no jumping to an interrupt subroutine to save registers, status, and return location is required. The Auto I/O instruction is, in itself, a one instruction subroutine. When executed under interrupts, the skips after execution are suppressed. Instead, if the word count has not reached zero after a data transfer, control is passed directly back to the main-line program at the point it was interrupted. If the word count did reach zero, a special signal (ECHO-) is sent to the peripheral device to indicate that it should stop requesting further data transfers. The Auto I/O instruction transfers control back to the main-line program whether the ECHO- signal is true or false. Upon receipt of ECHO-, the peripheral device stops data transfer requests, performs any stop action required (e.g., CRC checking or generation for magnetic tape), and then generates an End-of-Block interrupt so the program can process the data block input or prepare another block for output. Although the End-

of-Block interrupt can be vectored to any location by the peripheral controller, it is standard practice for the controller to vector this interrupt to four locations beyond the data transfer interrupt location. Figure 4-37 illustrates the typical usage of Auto I/O instructions under interrupts.

Data Transfer Interrupt Location	I	Automatic I/O Instruction
	I+1	Word/Byte Counter (negative)
	I+2	Word/Byte Address Pointer (start address -1)
	I+3	Unused
End-of-Block Interrupt Location	I+4	JST EOBSUB (Jump and Store to End-of-Block subroutine)

Figure 4-37. Interrupt Location Auto I/O Instruction Sequence

AIB AUTOMATIC INPUT BYTE TO MEMORY. Increments byte counter and address pointer, and unconditionally transfers an 8-bit data byte from addressed peripheral device to updated byte location in memory data buffer, which is addressed by address pointer. When byte count is incremented to zero, normal one-word skip after data transfer does not take place, or when used as an interrupt instruction, an ECHO signal is sent to addressed device.

AIN AUTOMATIC INPUT WORD TO MEMORY. Increments word counter and address pointer, and unconditionally transfers a full 16-bit data word from addressed peripheral device to updated word location in memory data buffer, which is addressed by address pointer. When word count is incremented to zero, normal one-word skip after data transfer does not take place, or when used as an interrupt instruction, an ECHO signal is set to addressed device.

AOB AUTOMATIC OUTPUT BYTE FROM MEMORY. Increments byte counter and address pointer, and unconditionally transfers an 8-bit byte from updated byte location in memory data buffer, which is addressed by address pointer, to addressed peripheral device. When byte count is incremented to zero, normal one-word skip after data transfer does not take place, or when used as an interrupt instruction, an ECHO signal is sent to addressed device.

AOT AUTOMATIC OUTPUT WORD FROM MEMORY. Increments word counter and address pointer, and unconditionally outputs a full 16-bit data word from updated word location in memory data buffer, which is addressed by address pointer, to addressed peripheral device. When word count is incremented to zero, normal one-word skip after data transfer does not take place, or when used as an interrupt instruction, an ECHO signal is sent to addressed device.

4.11 ASSEMBLER CONTROL DIRECTIVES

The assembler control directives provide for conditional assembly of source statements and establish and/or alter the contents and relocatability of the P register. If a label is present on any of these control directives, it is generally assigned the current value and relocation attribute of the assembler's Working Location Counter. These directives do not generate computer instruction words.

4.11.1 Conditional Assembly Controls

The IFF (If False) and IFT (If True) directives are provided to conditionally assemble subsequent lines of source code. The format for these two instructions is shown in figure 4-38.

[LABEL]	OP-CODE	EXPRESSION	[COMMENTS]

EXPRESSION: must be an absolute value of zero (False) or non-zero (True)

Figure 4-38. Begin Conditional Assembly Directives Format

The absolute expression must be previously defined (but not as an external). The last line affected must be an ENDC directive which signals the end of the conditional assembly. The ENDC directive has the following format:

[LABEL]	ENDC	[COMMENT]

There is no expression in the operand field.

Figure 4-39. End Conditional Assembly Directive Format

IFF and IFT directives must not be nested - i.e., no other IFF or IFT directive can appear between a given IFF or IFT directive and its associated ENDC directive. If the value of the absolute expression is zero, it is defined as false. If it is not equal to zero, it is defined as true. If the value of the expression satisfies the condition of the directive (false for IFF and true for IFT), the source lines between the directive and its associated ENDC directives are assembled. If the conditions are not met, the source lines are skipped (not assembled). The program END directive must not appear between an IFF or IFT directive and its associated ENDC directive.

must precede all data generating statements. If a label is present, it is assigned a zero value and a relative relocation attribute. No machine instructions are generated.

EXTR — EXTERNAL REFERENCE-SCRATCHPAD. Declares external symbols referenced by current program. Object loader links these declared external symbols through scratchpad (first 256 words of memory) at load time. Each name or symbol appearing in Operand field and also referenced by current program is output to object loader at load time. Since they are not defined within current program, these symbols must not be used in multi-term expressions. References to an EXTR-defined symbol must be direct, since assembler automatically generates indirect references through scratchpad. If a label is present, it is assigned current value and relocation attribute of the assembler's Working Location Counter. No machine instructions are generated.

REF — EXTERNAL REFERENCE-POINTER. Defines current location as linkage for reference to external symbol contained in the Label field. At load time, address assigned to external symbol is stored in memory location of REF directive.

4.14 SUBROUTINE DEFINITION DIRECTIVES

The following directives are provided primarily for documentation purposes. They are used for calling and delimiting subroutines in assembler output listings. The formats are described below in figure 4-44.

[LABEL]	CALL	EXPRESSION	[COMMENTS]
LABEL	ENT		[COMMENTS]
[LABEL]	RTN	EXPRESSION	[COMMENTS]

Figure 4-44. Subroutine Definition Directive Formats

No Operand field is allowed for ENT. The expression for RTN may be any expression defining the location of a subroutine return pointer (normally the label for the subroutine ENT).

CALL — SUBROUTINE CALL. Causes assembler to generate a Jump and Store instruction to location specified by expression. It is provided primarily for documentation purposes to facilitate recognition of subroutine Call instructions.

ENT — SUBROUTINE ENTRY. Reserves a word to hold return address from a subroutine call (JST). Assembler generates a HLT instruction for this directive. Any source statement which causes one word to be reserved could be used in its place.

RTN — SUBROUTINE RETURN. Generates an indirect Jump via symbol in Operand field (JMP *Expression). Note that expression is direct.

4.15 LISTING FORMAT AND ASSEMBLER INPUT CONTROLS

The following controls are provided for the purpose of formatting assembler output listings. With the exception of the TITL directive, these controls are simply special characters in the first column or position of a source line. The format for the TITL directive is shown below in figure 4-45.

TITL (one blank) ANY COMBINATION OF ALPHANUMERIC CHARACTERS NOT EXCEEDING 51 CHARACTERS IN LENGTH

Figure 4-45. Title Directive Format

No label field is allowed for TITL.

TITL — PAGE EJECT WITH TITLE. Generates a Top-of-Form to assembler listing device. Page number is then printed, followed (on same line) by character string specified in Operand field. Same character string is printed with page number at top of each page until a new TITL directive is encountered. If these directives are to be used throughout a program, first TITL directive should appear as first source line of program, ahead of comments, user defined op code definitions, and origin statements.

. (Period) — PAGE EJECT. Generates a Top-of-Form to assembler listing device. This control must appear as first character of a source statement. Remainder of input line will be ignored. If a TITL directive has been previously processed, the title will be printed at the Top-of-Form as described under TITL. If no TITL has been processed, a Top-of-Form is generated and a page number is printed.

* (Asterisk) — COMMENT LINE. Allows source line comments to be exactly duplicated on assembler listing device. This control must appear as first character of source statement. All characters following asterisk on source statement are duplicated on output listing. Comment lines may appear anywhere in a program.

↑ (Up Arrow) — PAUSE. Causes assembler to halt. Assembly is continued by pressing RUN pushbuttom. This control is most useful when paper tape input is used. Up-arrow must appear as first character of a source line. Remainder of input line will be ignored.

4.16 USER DEFINED OPERATION CODE DIRECTIVE

User defined operation code directives allow the user to name or define his own instruction mnemonics for the current assembly. If included in a program, user defined op code directives must precede all source statements other than comments or TITL directives. The user is referred to the applicable Assembler Reference manual for a detailed discussion of their usage.

Section 5

INPUT/OUTPUT AND INTERRUPT OPERATIONS

5.1 INTRODUCTION

5.1.1 Discussion of Input/Output Operations

Interfacing with the standard peripheral devices generally consists of operations which can be treated as members of three major categories - Control, Sense, and Data Transmission. The precise definitions of the various instructions, function codes and status words depend on the design of the individual peripheral interfaces.

5.1.1.1 Control

Control instructions prepare peripheral devices for data transmission. The instructions, Select (SEL) and Select and Present (SEA and SEX), initialize, establish operating codes, and control the status of the addressed peripheral device. The format for Control instruction follows:

[LABEL] INST DA,FC

where:

INST = mnemonic of Control instruction (SEL, SEA, SEX)
DA = assigned address of device interface (:01 thru :1F)
FC = any one of eight function codes (:0 thru :7)

The SEL instruction commands the addressed peripheral device to perform some function (initialization, etc.) according to the function code. SEL is used where no further information, other than the function code, is required, so zeros are placed on the Data bus.

The SEA and SEX instructions command the peripheral device to perform some function where additional information, other than the function code, is required. For example, if the device interface controller contains a status or address register which must be set during initialization, the required information is first loaded into the A or X register. Upon execution of the appropriate Select and Present instruction (SEA/SEX), the contents of the A or X register are placed on the Data bus. An example of the use of a Select and Present instruction is when the Teletype controller is initialized for Full-duplex operation (SEA/SEX 7,4 with appropriate register, A or X, = 1).

5.1.1.2 Sense

Once a peripheral device has been prepared for transmission of data with the proper commands, it is necessary to determine whether the device is ready to accept or send the data. This is accomplished using the Sense and Skip on Response (SEN) and Sense and Skip on No Response (SSN) instructions. One or the other of these instructions should immediately precede an unconditional data transmission sequence such that an appropriate Sense response is detected prior to the data transfer.

```
    INST        OPERANDS

    SEN         DA,FC
    JMP         $-1
    Data Transmission
    .
    .
    .
or:
    SSN         DA,FC
    Data Transmission
    .
    .
```

Figure 5-1. Sense Routines

Refer to figure 5-1. In the first example, the Sense instruction is executed until a true response is detected and the Jump instruction is skipped. The data transmission is then performed. In the second example, the Sense instruction is executed only once. If a false response is detected, the data transmission instruction is skipped.

5.1.1.3 Data Transmission

Unconditional data transmission is accomplished using the Input to Register (INA and INX) and Output from Register (OTA, OTX and OTZ) instructions. (Refer to figure 5-2).

```
    INST        OPERANDS

    SEN         DA,FC
    JMP         $-1
    INA         DA,FC
    .
or:
    SEN         DA,FC
    JMP         $-1
    OTA         DA,FC
    .
    .
```

Figure 5-2. Unconditional Data Transmission

When the Sense response is true, the Jump instruction is skipped and the data transmission instruction is executed.

Conditional data transmission is accomplished by combining Sense operations with data transmission using the Read to Register (RDA, RDX, RBA and RBX) and Write from Register (WRA, WRX and WRZ) instructions. (Refer to figure 5-3.)

```
        INST        OPERANDS
        RBA         DA,FC
        .
or:     .
        WRX         DA,FC
        .
        .
```

Figure 5-3. Conditional Data Transmission

These instructions are executed repeatedly until a true Sense response is received. The data transmission then occurs and the next instruction in sequence is executed. The Sense and unconditional data transfer operations can be combined in a conditional data transfer instruction only when the function codes for the two operations are the same. The conditional data transmission instructions are interruptable.

Block data transmissions are performed using the Block Input to Memory (BIN) and Block Output from Memory (BOT) instructions. (Refer to figure 5-4.)

```
        LABEL   INST    OPERANDS
                LXP     COUNT
                BIN     DA,FC
                DATA    BUF - 1
                .
or:             .
                LXP     COUNT
                BOT     DA,FC
                DATA    BUF - 1
                .
                .
        BUF     RES     COUNT
                .
                .
```

Figure 5-4. Block Data Transmission

These instructions are executed repeatedly, transmitting one word of data each time a true Sense response is received, until all data has been transmitted. The data is transmitted in reverse order (in order of decreasing addresses). The next instruction in sequence is then executed. The function code associated with these instructions is the same as the function code used by the incorporated Sense. The block data transmission instructions are not interruptable.

In-line automatic data transmissions are performed using the Automatic Input to Memory (AIN and AIB) and Automatic Output from Memory (AOT and AOB) instructions. (Refer to figure 5-5.)

```
        LABEL   INST    OPERANDS
        SENSE   SEN     DA,FC
                JMP     $ - 1
                AIN     DA,FC
                DATA    Negative Data Count (Word)
                DATA    BUF - 1 (Word)
                JMP     EOB
                JMP     SENSE
                .
                .
or:
        SENSE   SEN     DA,FC
                JMP     $ - 1
                AOB     DA,FC
                DATA    Negative Data Count (Byte)
                BAC     BUF - 1 (Byte)
                JMP     EOB
                JMP     SENSE
                .
                .
        BUF     RES     COUNT
                .
                .
```

Figure 5-5. In-line Auto I/O Data Transmission

These instructions unconditionally transmit one word/byte of data each time they are executed and are therefore preceded by an appropriate Sense command. In addition, the Base Address pointer and the Negative Data Count are incremented, with the Data Count eventually becoming zero and generating an exit to the End-of-Block processing routine (EOB). Automatic I/O instructions may be used under interrupts, in which case the Sense instruction is not required and the exits are replaced by a return to the mainline program. A final interrupt to a different (End-of-Block) location is generated by the peripheral controller when the buffer is completely transferred.

5.1.2. Interrupt Operations

Interrupts constitute a means of reacting quickly to random, external stimuli without consuming valuable processing time in a continuous polling environment. Peripheral devices which are to be operated under interrupt control are assigned reserved memory locations anywhere in Memory. These interrupt addresses are generated by the individual peripheral controllers and generally have jumper selectable locations within the first 512 locations of Memory. Appendix B includes a table of standard interrupt address assignments.

When an interrupt is recognized, the instruction at the associated interrupt location is executed. If the instruction does not modify the program counter, control is immediately restored to the mainline program. Otherwise, processing continues at the location specified by the new contents of the P register. Although any of the instructions in the ALPHA LSI's repertoire could be used in the reserved locations as interrupt instructions, only certain of them are generally useful - IMS, JMP, JST and the Auto I/O instructions. With LSI-1 processors, any memory reference instruction performing relative to P backwards addressing should not be used as an interrupt instruction (the instruction would reference the location one lower that the location actually programmed; i.e., $9 instead of $8).

Before a given peripheral device can be operated under interrupt control, the interrupts for that device must be enabled. This enables the device to generate an interrupt request when the associated event occurs. In addition, Processor interrupts must be enabled. This is accomplished using the EIN instruction and allows the Processor to respond to the interrupt request of the peripheral device.

5.1.2.1. Non-Input/Output

The Increment Memory and Skip on Zero (IMS) instruction is used in interrupt programming as a counter or timer for external events. As interrupt instructions, increment results of zero do not generate skips. They generate, instead, a signal (ECHO) to the peripheral interface which caused the interrupt. Usually this signal is used by the device to generate a second interrupt to another reserved location at which a Jump and Store (JST) instruction to a counter/timer maintenance subroutine would be located.

The JST instruction is used in interrupt programming as a means of transferring control to an interrupt subroutine in a manner such that return to the mainline program at the interrupted location can be accomplished upon completion of the operations required by the interrupt. JST is the only instruction which disables Processor interrupts when it is used as an interrupt instruction. Before returning to the mainline program, the Processor interrupts should be re-enabled.

5.1.2.2 Input/Output

The Automatic Input to Memory (AIN and AIB) and Automatic Output from Memory (AOT and AOB) instructions were specifically designed as interrupt instructions. Used to transfer blocks of data between Memory and the peripheral devices, these instructions contain their own word/byte count and memory word/byte address. They do not affect the A and X registers, the OV indicator or the P register when transferring data as interrupt instructions. As each data word/byte is transmitted, the associated pointer and counter are automatically incremented.

5.1.2.3 End-of-Block Interrupts

When either the IMS or Auto I/O instructions are used as interrupt instructions, increment results of zero (any memory location for IMS and the negative word/byte count for the Auto I/O instructions) produce ECHO signals which are typically used by the various peripheral devices to generate End-of-Block interrupt requests to different reserved interrupt locations.

5.2 NON-INTERRUPT INPUT/OUTPUT EXAMPLES

The examples shown in figures 5-6 through 5-10 are discussed in the paragraphs that follow.

```
LABEL      INST    OPERANDS    COMMENTS

Optional   SEL     4,4         Initialize Line Printer
           .
           LDA     CHAR        A = Char to Print
           SEN     4,1         Sense Line Printer Ready
           JMP     $-1         (Not Ready)
           OTA     4,1         Unconditionally Output A
           .
           .
```

Figure 5-6. Initialization and Unconditional Output to Line Printer

```
LABEL      INST    OPERANDS    COMMENTS

Optional   SEL     7,4         Initialize Teletype
           .
           SEN     7,3         Sense Teletype Ready (not busy)
           JMP     $-1         (Not Ready)
           SEL     7,2         Command Step Read
           SEN     7,1         Sense Character Buffer Full
           JMP     $-1         (Not Full)
           INA     7,0         Unconditionally Input Character to A
           .
           .
```

Figure 5-7. Unconditional Character Read from Teletype Paper Tape Reader

```
LABEL      INST    OPERANDS    COMMENTS

           .
           .
Optional   SEL     4,4         Initialize Line Printer
           .
           .
           LXP     :0C         Top of Form Character
           WRX     4,1         Output to Line Printer When Ready
           .
           .
```

Figure 5-8. Initialization and Conditional Control of Line Printer

```
LABEL       INST    OPERANDS    COMMENTS
                .
                .
Optional    SEN     7,3         Sense Teletype Ready (not busy)
            JMP     $-1         (Not Ready)
                .
                .
            SEL     7,0         Enable Auto Echo
                .
                .
            RBA     7,1         Input a Teletype Character to A When Ready
            LLA     8           Shift to Most Significant 8 Bits
            RBA     7,1         Input Another character to Least
                                Significant 8 Bits
            SEL     7,4         Disable Auto Echo
                .
                .
```

Figure 5-9. Conditional Input from Teletype Keyboard with Auto Echo

```
LABEL       INST    OPERANDS    COMMENTS
                .
                .
Optional    SEL     4,4         Initialize Line Printer
                .
                .
            LXP     COUNT       X = Word Buffer Length
            BOT     4,1         Block Output to Line Printer
            DATA    BUF-1       Character Buffer Address Less One
                .
                .
BUF         RES     COUNT       Data Buffer
```

Figure 5-10. Uninterruptable Block Output to Line Printer


```
LABEL       INST    OPERANDS    COMMENTS
                .
                .
Optional    SEN     5,3         Sense Card Reader Ready
            JMP     $-1         (Not Ready)
            SEL     5,4         Initialize Card Reader
            SEL     5,3         Command Card Reader Read Card
LOOP        SEN     5,0         Sense Input Character Ready
            JMP     $-1         (Not Ready)
            AIB     5,0         Automatic Input Character to Buffer
            DATA    -80         Buffer Byte Count
            BAC     BUF-1       Buffer Byte Address
            JMP     $+2         Zero Counter Results - Exit
            JMP     LOOP        Loop on Non-Zero Counter Results
                .
                .
BUF         RES     40          80 Character (Byte) Data Buffer
```

Figure 5-11. Automatic Byte Input from Card Reader

5.2.1 Control Instructions

The SEL instruction is the most widely used control instruction for peripheral devices. It is used both for initializing the devices, as in figures 5-6, 5-7, 5-8, 5-10 and 5-11, and for causing the peripheral devices to perform specific functions, as in figures 5-7, 5-9 and the second SEL instruction in figure 5-11. Special characters are sometimes used for control functions (e.g., the Line Printer Top of Form character in figure 5-3).

NOTE

When a Select type instruction is used to turn off interrupts that may be pending, it should be preceded by a SIN 1 instruction to disable Processor recognition of the pending interrupt. This is necessary since the Processor examines interrupt requests prior to the Select taking effect and will therefore respond to the interrupt even though it is no longer pending.

The SEN instruction is used to test whether the specified data source or destination in the addressed peripheral device is ready to transmit or receive data. Sometimes both the peripheral device and a particular buffer within the device must be ready for data transmission, as in figures 5-7 and 5-11. In many cases, the Sense function can be incorporated into the Conditional I/O instructions, as in figures 5-8 and 5-9.

5.2.2 Unconditional Instructions

Unconditional Input instructions consist of both word and byte instructions. While the Word input instructions replace all 16 bits of the register (figure 5-7), the byte input instructions affect only the least significant 8 bits of the register. When byte-orientated peripheral devices are used, these instructions allow the programmer to pack the input data before storing it in Memory.

The Unconditional Output instructions are word-oriented instructions. Since byte-oriented peripheral devices accept only the least significant 8 bits of data output from a register, there is no need for byte Output instructions.

5.2.3. Conditional Instructions

The Conditional I/O instructions incorporate both the Sense and data transmission functions into one instruction. These instructions make sense, of course, only when the function codes for the Sense and data transmission operations are the same.

The Conditional Input instructions consist of both word and byte instructions. While the word input instructions replace all 16 bits of the register, the byte input instructions affect only the least significant 8 bits of the register. When byte-oriented peripheral devices are used, these instructions allow the programmer to pack the input data before storing it in Memory, as in figure 5-9.

The Conditional Output instructions are word-oriented instructions. Since byte-oriented peripheral devices accept only the least significant 8 bits of data output from a register, there is no need for byte-output instructions.

Interrupts may be acknowledged during the execution of a Conditional I/O instruction.

5.2.4 Block I/O Instructions

The Block I/O instructions allow high speed data transmissions between Memory and peripheral devices. They essentially access each data buffer memory location by summing the contents of the X register and the data buffer pointer (buffer address - 1) in the second word of the instruction. Each time the addressed peripheral device generates a true Sense response, data is transmitted and the X register is decremented. Thus, the data is transmitted from, or to, the end of the buffer (higher memory locations) first. The last word transmitted accesses the start (lowest memory location) of the buffer. Interrupts may be acknowledged only after the X register has been decremented to zero and the instruction has been completed - i.e., when all data words have been input or output.

These instructions access word memory operands only (see figure 5-10). They do not affect the contents of the A register.

5.2.5 Automatic I/O Instructions

Although the Auto I/O instructions have been designed specifically as interrupt instructions, they may also be used in non-interrupt, in-line programming. They are three word instructions, with the second word containing the negative (two's complement) word or byte count and the third word containing a word or byte address pointer (buffer address - 1). Since they are unconditional transfer instructions, the specified data source or destination in the addressed peripheral device must generate true Sense responses before data transmission occurs. Each data transmission increments both the data counter and the address pointer. Non-zero data counter increment results generate a one-word skip. Zero increment results cause the next instruction in sequence (the instruction after the address pointer which is skipped by non-zero increment results) to be executed (see figure 5-11).

5.3 INTERRUPT STRUCTURE AND EXAMPLES

5.3.1 General Interrupt Handling

External interrupts cause the computer to execute one instruction outside of the mainline program. If the instruction does not modify the P register, the computer continues with the mainline program after executing the interrupt instruction. If the interrupt instruction modifies the P register (either a JST or JMP), the computer continues processing at the location specified by the new value in the P register.

If a peripheral device is to operate under interrupt control, reserved locations in Memory are assigned to the device. The computer then executes the instruction at the reserved location when the peripheral device generates an interrupt to the Processor. Each device may be assigned one or more reserved locations. For example, a device moving blocks of data to or from the computer may generate one interrupt for each word or byte of data moved and a second interrupt when the entire block of data has been moved. The interrupt for each word or byte would require one location and the interrupt indicating the end of the block of data would require another.

Before any interrupt can be recognized by the Processor, several conditions must be met:

1. Interrupts must be enabled, in general. If any interrupts are to be recognized, the Enable Interrupts (EIN) instruction must be executed.

2. The specific peripheral device interrupt must be enabled. Specific interrupts are enabled by setting an interrupt enable flag in the peripheral device interface controller. Enable flags are generally set by executing a Select (SEL) instruction with a device address and function code specifying which interrupt is to be enabled. Using interrupt enable flags, the programmer can selectively enable and disable interrupts.

3. The interrupt condition must exist (i.e., the device must be ready to accept or transmit data). Many peripheral devices "remember" interrupt

conditions generated prior to enabling the interrupt enable flags. Care should be taken to reset the peripheral device interrupts before enabling the enable flag so that false interrupts do not occur immediately after enabling the interrupts.

4. No higher priority interrupt must be waiting. Each peripheral interface or computer option has a definite priority assignment. Interrupts are processed according to priority if more than one interrupt is pending.

5. The computer must be in the RUN mode. Interrupts cannot be recognized when the computer is halted, or during DMA operations.

5.3.2 Examples of Initialization and Enabling Sequences

Initialization and interrupt enabling take place prior to the generation and use of the interrupts. The examples below involving a Line Printer and the Real Time Clock are typical of initialization sequences.

```
INST      OPERANDS      COMMENTS

SEN       4,1           Wait for Line Printer Buffer ready
JMP       $-1           (Not Ready)
SEL       4,7           Reset Interrupt Enable flags
SEL       4,5           Enable Word Interrupt Enable flag
SEL       4,6           Enable EOB Interrupt Enable flag
EIN                     Enable Processor interrupts
.
.
```

Figure 5-12. Line Printer Interrupt Initialization Sequence

In addition to being reset by the SEL 4,7 instruction (figure 5-12), the interrupt enable flags may also be reset by the Line Printer Initialization instruction (SEL 4,4). Note that the Word interrupt enable flag is enabled before the End-of-Block (EOB) interrupt enable flag. When specific actions in a peripheral device are additionally required to generate interrupts (e.g., a card reader must read a card), the instruction (SEL) causing the action must be executed before the interrupt can take place. The sequence in figure 5-12 is used in conjunction with an AOT or AOB instruction in the Word interrupt location and a JST instruction to an EOB routine at the EOB interrupt location.

In addition to being reset by the SEL 8,3 instruction (figure 5-13), the interrupt enable flags may also be reset by the Real Time Clock Initialization instruction (SEL 8,4). Note that the Sync interrupt enable flag is armed before the Time and Sync interrupt enable flags are enabled. This sequence is used in conjunction with an IMS instruction in the Word interrupt location and a JST instruction to a Sync maintenance routine in the Sync interrupt location.


```
INST      OPERAND       COMMENTS

SEL       8,3           Reset RTC Interrupt Enable flags
SEL       8,2           Arm RTC Sync Interrupt Enable flag
SEL       8,0           Enable RTC Time and Sync Interrupt Enable
                        flag
EIN                     Enable Processor Interrupts
.
.
.
```

Figure 5-13. Real Time Clock Interrupt Initialization Sequence

5.3.3 Examples of Interrupt Instructions

The contents of the interrupt locations associated with the above examples are illustrated in figures 5-14 and 5-15.

```
LABEL/
LOCATION    INST      OPERANDS      COMMENTS

:42(Word)   AOB       4,1           Automatic Output Byte Instruction
            DATA      -80           Negative Character Buffer Length
                                    (Byte Counter)
            BAC       BUF-1         Byte Address Pointer (Start-1)
            .
            .
:46(EOB)    JST       SUB           Jump to End-of-Block Routine,
            .                       Disable Processor Interrupts
            .
Main Memory
SUB
            ENT
            .
            .
            RTN       SUB
            .
            .
BUF         RES       40
            .
            .
```

Figure 5-14. Line Printer Interrupt Instructions

Since the byte counter and address pointer are modified during the data transmission, they must be preset each time a line of characters is to be printed prior to execution of the initialization sequence discussed in paragraph 5.3.1. When all characters have been transferred, the instruction at location :46 is executed and control is transferred to the EOB routine beginning at SUB. This routine might output a carriage return

character to cause the line to be printed, or perform any other line termination processing required. The last character of the buffer might be a carriage return (see Line Printer Driver documentation in Software manual).

```
 LABEL/
LOCATION    INST    OPERANDS    COMMENTS

 :18 (Time) IMS     COUNT       Increment RTC Counter COUNT

 :1A (Sync) JST     SYNC        Transfer to Sync Subroutine,
                                Disable Processor Interrupts


Main Memory
  SYNC      ENT                 Save Main Program Return Location
            SIN     1           Inhibit Status (Guarantee Word Mode)
                                to Save A Register
            STA     ASAVE       Save A Register
            SIA
            STA     STATUS      Save Status
            STX     XSAVE       Save X Register
            LAM     100         Reset
            STA     COUNT       RTC Counter COUNT
            .
            .
                                Perform Specified Maintenance Function
            .
            .
            LDX     XSAVE       Restore X Register
            LAP     3
            AND     STATUS      Byte and OV Bits to A Register
            LRA     1           Restore OV
            JAZ     $+2         Test Byte Mode
            SBM                 Restore Byte Mode
            SIN     1           Inhibit Status (Guarantee Word Mode)
                                to Restore A Register
            LDA     ASAVE       Restore A Register
            EIN                 Enable Processor Interrupts
            RTN     SYNC        Return to Mainline Program
```

Figure 5-15. Real Time Clock Interrupt Instructions

Each acknowledgement of a Time interrupt causes the RTC counter COUNT to be incremented. When COUNT is incremented to zero, recognition of the Sync interrupt (at location : 1A) generates execution of the SYNC interrupt subroutine.

Interrupts are automatically disabled by execution of the JST instruction, but the addressing mode and the state of the overflow indicator are unchanged. Because the computer might be in the Byte addressing mode when the interrupt occurs, the Word mode is forced for one instruction so the full 16-bit contents of the A register can be saved and the instruction address will be treated as a word address. When this is done, the computer status is input, which also sets the addressing mode to the Word mode and resets the overflow indicator. The Status and the contents of the X register are then saved. The Real Time Clock counter COUNT is reset to a negative value as part of the required maintenance operations.

Restoration of the contents of the X register begins the exit sequence of the subroutine. The computer status is then restored and Byte mode inhibited for one instruction to ensure restoration of the full 16-bit contents of the A register. The interrupts are then re-enabled and the subroutine is exited prior to acknowledgement of any other interrupt (since the EIN instruction inhibits recognition of interrupts for the duration of the RTN SYNC instruction).

The save/restore sequences discussed here should be used at the beginning and end of any interrupt subroutine to which a JST instruction at an interrupt location refers. The Real Time Clock counter COUNT should also be set to a negative value before the initialization sequence discussed in paragraph 5.3.1 is executed.

5.4 INTERRUPT LATENCY

Recognition of an interrupt request from a peripheral device by the computer is not always instantaneous. The conditions discussed below delay acknowledgement of interrupts.

5.4.1 Interrupt Service

Interrupt acknowledgement occurs "between" the execution of instructions - i.e., just after the completion of a given instruction. The Conditional Input/Output instructions allow recognition of interrupts before their completion as long as false (not ready) Sense responses are obtained from the specified data source or destination. After the interrupt is serviced, processing is resumed with the Conditional Input/Output instruction. The Scan Memory (SCM) instruction similarly allows recognition of interrupts after each specified word or byte of Memory is compared to the contents of the A register. If interrupts were off prior to issuing an instruction, the EIN delays recognition of any interrupt until after the execution of from one (minimum) to three (maximum) instructions. This allows return from interrupt subroutines to the mainline program before acceptance of another interrupt. The Block Input/Output (BIN and BOT) instructions, the Status Inhibit (SIN) instruction and all shift instructions must be completed before recognition of an interrupt may occur. Since their use in mainline programs may constitute non-trivial delays in the recognition of interrupts, the programmer should use such instructions with care. In addition, when Direct Memory Access (DMA) operations are in progress, recognition of interrupts is delayed for the duration of data block transmission.

5.4.2 Priority Resolution

Occasionally, multiple interrupt requests occur. When this happens, the interrupt having the highest priority is acknowledged first, then the next, and so forth down to the interrupt having the lowest priority. To avoid responding to the same interrupt twice, one to three mainline program instructions will always be executed between each recognition of an interrupt. The number of instructions expected depends on the Processor type and the duration of the instructions executed. The standard interrupt priorities are listed in figure 8-4.

Section 6

PROCESSOR OPTIONS

6.1 INTRODUCTION

This section describes how to use the various features of the Teletype/CRT/Modem controller, Real Time Clock (RTC), and Autoload (AL) options, and the Basic Variables (BV) package which are contained on the Processor Option board (Figure 6-1). These features are selectable by means of external jumpers on connectors located on the rear edge of the board. In addition, the Power Fail/Restart option contained in the Processor is also described.

The most common operating modes require no external jumpers. Unjumpered mating connectors are supplied with the Processor Option board.

6.2 REAR EDGE CONNECTORS (Figures 6-2 and 6-3)

The rear edge of the Processor Option board has two connectors designated J1 and J2. Connector J1 is used to select various operating modes via external jumpers while connector J2 is used to interface to a Teletype, CRT, or Modem.

J1 is designed to accept a 50-pin two-row edge connector. Identifying pin numbers silkscreened on the board apply to the Viking type 2VH25/1JN5 connector which is numbered 1-50 with the odd numbers (1-49) in one row and even numbered pins (2-50) in the other. In some cases, connector type 3VH25 is used. Pin designations of this connector are A1 thru A25 in one row and pins B1 thru B25 in the other. Corresponding pins of the two types of connector are shown in figure 6-2 along with signals and related options (in parenthesis).

J2 is designed to accommodate a 36-pin Winchester connector (8BDJ185). The pin assignments, signals, and related option (in parenthesis) for connector J2 are shown in figure 6-3.

NOTE

All reserved pins listed in figures 6-2 and 6-3 are not to be used for any purpose.

Connector J1 mounts on the board with the row having pins A1 thru A25 (or 1 thru 49) interfacing with the component side of the board. The contacts for J2 are designated A through V and 1 through 18. Pins A through V interface with the component side of the option board while pins 1 through 18 interface with the solder side.

Connector J1 should be installed with connector pins A1 and B1 (or 1 and 2) to the right when viewed from the rear of the computer. Connector J2 has the signals brought out in such a way that when interfacing with an ASR-33 teletype, the connector may be installed right-side up or up-side down with no ill effects. When used with terminals other than a Teletype, J2 must be installed with pins A and 1 to the right as viewed from the rear of the computer.

6.3 TELETYPE/CRT/MODEM CONTROLLER

The Teletype/CRT/Modem (TTY/CRT) option interfaces a CRT, Modem, or modified ASR-33 Teletype to the ALPHA LSI computer. It performs all of the data and control signal conversion required for the computer to control the user terminal. An ASR-33 Teletype provides four Input/Output features in one package: Keyboard Input, Page Printer, Paper Tape Reader and Paper Tape Punch. A CRT provides keyboard entry and display.

The interface contains a data buffer register which performs parallel-to-serial data conversion for transferring data from the computer to the user terminal and serial-to-parallel conversion when transferring data from the user terminal to the computer. In addition, the interface has provisions for interrupt generation for both Word and End-of-Block interrupts.

The TTY/CRT Interface option has been assigned a standard device address of 7.

Output from the computer is printed on the TTY page printer or displayed on the CRT. If the TTY punch is turned on, the output is also punched. The TTY punch and page printer cannot be separately controlled by the computer. The TTY operator must turn the punch on or off as desired.

Input to the computer is accomplished via the TTY/CRT keyboard or the TTY Paper Tape Reader. They are controllable separately from the computer. The Paper Tape Reader can read bytes one at a time or continuously. Automatic Echo is a feature which allows any input to be echoed back to the TTY/CRT for printing or display.

The Teletype or CRT can be operated in either Half-duplex or Full-duplex mode. The Initialize instruction (SEL 7,4) puts the controller in the Half-duplex mode. Execution of the Select and Present instructions (SEA 7,4 or SEX 7,4) with the register contents equal to 1 puts the controller in Full-duplex mode.

The TTY/CRT controller has provisions for ten different baud rates, a variable length word (with or without parity), and either one or two stop bits. Additionally, the user can select a current loop data path for teletypes, a TTL compatible data path, or an EIA RS232C/CCITT data path for various terminals. The user should consult the terminal manufacturers literature to determine the exact interface requirements of the terminal.

6.3.1 Baud Rate Selection

The TTY/CRT controller uses a variable format counter to provide internal clock timing for the data channel. Two counter inputs (SLCT1 and SLCT2) determine the count pattern to be employed. Eight counter outputs are brought out to connector J1. One of these outputs (CP006, CP013, CP026, CP052, CP104, CP208, CP416 or CP568) can be jumpered to the TCLK terminal to provide the appropriate clock period.

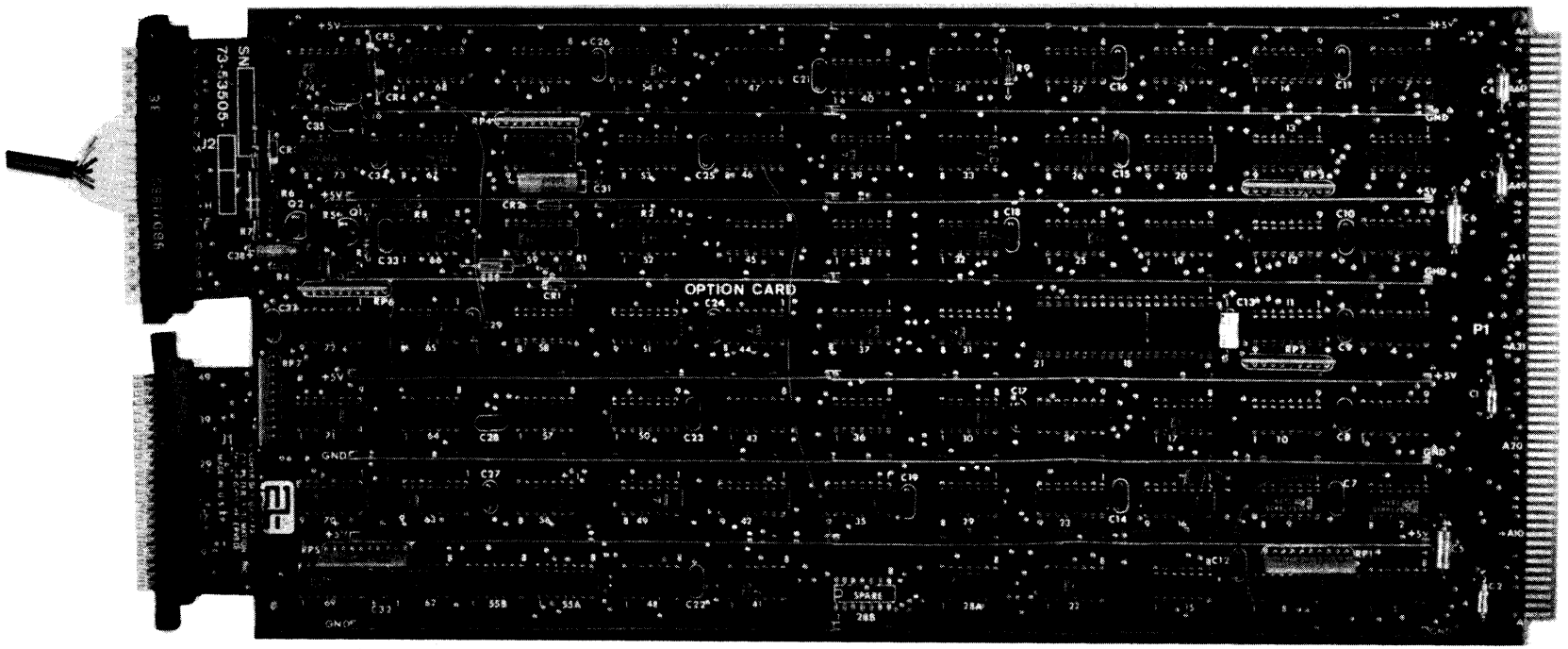


Figure 6-1. Processor Option Board

OPTION BOARD
CONNECTOR J1
(ACCEPTS VIKING
3VH25/1JN5)

	A			B	
(RTC) TTLF	1	1*	2*	1	SSW– (BV)
(TTY) SLCT1	2	3	4	2	OFST– (BV)
(AL) PFAL–	3	5	6	3	MAI– (BV)
(TTY) CP006	4	7	8	4	TCLK-(TTY)
(TTY) SLCT2	5	9	10	5	REMOTE AL– (AL)
(RTC) CLKIN–	6	11	12	6	INH– (RTC)
(TTY) CP026	7	13	14	7	CP013 (TTY)
(TTY) CP104	8	15	16	8	CP052 (TTY)
(TTY) CP568	9	17	18	9	CP416 (TTY)
(TTY) CP208	10	19	20	10	RMDIS– (AL)
(RESERVED)	11	21	22	11	MEC (RTC)
GND	12	23	24	12	GND
GND	13	25	26	13	GND
(TTY) ORIN	14	27	28	14	ENDSW– (BV)
(TTY) TTYOF–	15	29	30	15	(RESERVED)
(BV) DS03–	16	31	32	16	SMDAT– (TTY)
(BV) DS01–	17	33	34	17	DS00– (BV)
(BV) OPT–	18	35	36	18	DS02– (BV)
(BV) RST–	19	37	38	19	CINT– (BV)
(RTC) 1KHZ	20	39	40	20	10KHZ (RTC)
(BV) STR1	21	41	42	21	STR2 (BV)
(BV) STR3	22	43	44	22	STR4 (BV)
(TTY) PS	23	45	46	23	DTDAT (TTY)
(TTY) WLS1	24	47	48	24	WLS2 (TTY)
(TTY) PI	25	49	50	25	SBS (TTY)

*Pin numbering system if type 2VH25/1JN5 connector installed.

Figure 6-2. Option Board Connector J1 Pin Assignments

OPTION BOARD
CONNECTOR J2
(ACCEPTS WINCHESTER
8BDJ18S)

(RESERVED)	A	1	IRDRA	
GND	B	2	DTDAT	
GND	C	3	EIAT–	
(TTY) ORIN	D	4	RTS	
(AL) AL–	E	5		
(BV) CINT–	F	6		
TDAT	H	7	(RESERVED)	(TTY)
RCV	J	8	MOT+	
GND	K	9	MOT–	
MOT–	L	10	GND	
MOT+	M	11	RCV	
(RESERVED)	N	12	TDAT	
	P	13	CINT– (BV)	
	R	14	AL– (AL)	
EIAR–	S	15	ORIN (TTY)	
CTS	T	16	GND	
SMDAT–	U	17	GND	
IRDRA	V	18	(RESERVED)	

(TTY)

Figure 6-3. Option Board Connector J2 Pin Assignments

The SLCT1 and SLCT2 signals are static control signals that are either grounded or left open. Ground is available on pins 23 thru 26 of connnector J1. The grounding configurations for selecting the various baud rates are shown in table 6-1.

Table 6-1. Baud Rate Selection

BAUD RATE	SLCT1 (pin 3)	SLCT2 (pin 9)	JUMPER
75	GND	OPEN	Pin 8 to 17
110 (standard)	OPEN	OPEN	none
134.5	OPEN	GND	none
150	GND	OPEN	Pin 8 to 18
300	GND	OPEN	Pin 8 to 19
600	GND	OPEN	Pin 8 to 15
1200	GND	OPEN	Pin 8 to 16
2400	GND	OPEN	Pin 8 to 13
4800	GND	OPEN	Pin 8 to 14
9600	GND	OPEN	Pin 8 to 7

6.3.2 Word Length Selection

The user may select either 5-, 6-, 7- or 8-bit character lengths for the controller to process. Character length selection is controlled by WLS1 and WLS2 (pins J1-47 and J1-48 respectively). These signals are static control signals that are either grounded or left open. Ground is available on pins 23 through 26. The grounding configurations for word length selections are shown in table 6-2.

Table 6-2. Word Length Selections

WORD LENGTH	WLS1 (pin 47)	WLS2 (pin 48)
5-bits	GND	GND
6-bits	OPEN	GND
7-bits	GND	OPEN
8-bits (standard)	OPEN	OPEN

6.3.3 Parity Selection

The user can choose to have parity error processing with parity error sensed by the SEN 7,6 instruction. Two signals control parity in the controller. Parity Inhibit (PI, J1-49) controls parity. When PI is open, parity is disabled. When PI is grounded, the parity generation and check functions are enabled and a parity bit is inserted into the transmitted word. When parity is enabled, the Parity Select signal (PS, J1-45) determines whether even or odd parity is generated by the transmit function and checked by the receive function. When PS is open, even parity is selected. When PS is grounded, odd parity is selected.

6.3.4 Stop Bit Selection

All terminal equipment requires either one or two stop bits. The Stop Bit Select signal (SBS, J1-50) provides this selection capability. When SBS is grounded, one stop bit is inserted in the transmitted word. When SBS is open, two stop bits are inserted in the transmitted word. Note that the selection of two stop bits when programming a 5-bit word generates 1.5 stop bits.

6.3.5 Alternate Interrupt Locations

When using the TTY/CRT controller in the Half-duplex mode, the standard TTY/CRT interrupt locations of : 0002 and : 0006 may be changed to : 0022 and : 0026, respectively by jumpering TTYOF- (J1-29) to MEC (J1-22). Note that this feature is automatically overridden when operating in the Full-duplex mode.

6.3.6 Data Interface Selection

The user has a choice of three types of data interface that can be used with a terminal device. These interface types are current loop, RS232C/CCITT and TTL/DTL compatible.

6.3.6.1 Current Loop Interface (Figure 6-3)

The Current Loop interface utilizes a 3-wire ground common interface which is characterized by the presence or absence of a 20 milliamp dc signalling current. The current loop interface converts logic signals to current signals and vice-versa as follows:

Mark = 20 mA current flow
Space = no current flow

The controller current loop transmit signal is TDAT, while the controller receive signal is RCV-. TDAT is available on connector J2 at pins H and 12. RCV- enters the controller at J2 pins J and 11. A logic ground reference between the controller and the terminal device is required and is available on J2 pins K and 10.

The controller current loop receive and transmit circuits have a 1500 ohm, 1 watt resistor in series with their respective lines. These resistors are used to set the current level on each line to 20 mA dc. The current loop receive line also has a built-in rolloff filter which limits baud rates to 150 baud maximum for use with teletypes. For faster current-loop devices, the filter capacitor may be removed.

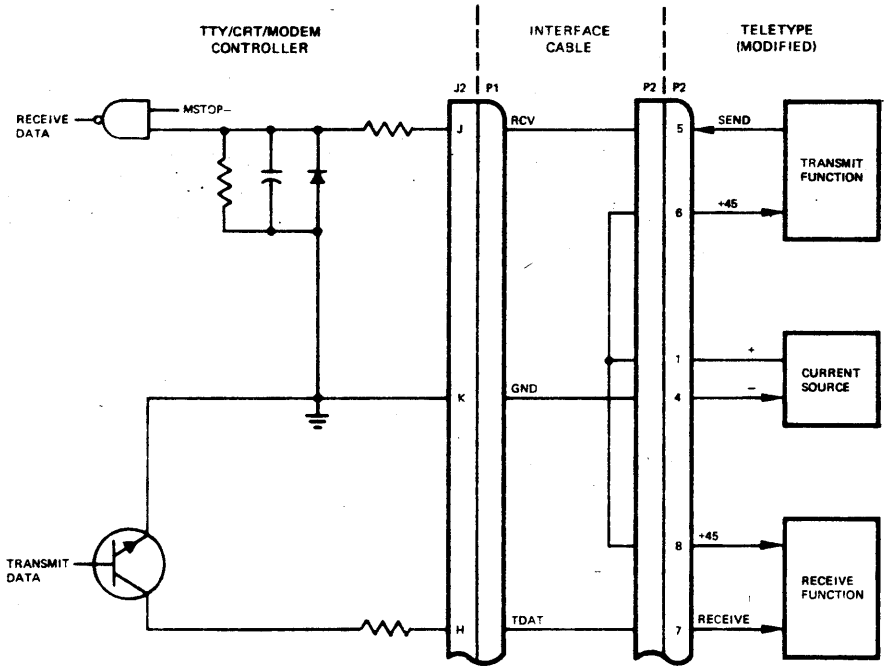


Figure 6-4. Current Loop Interface

6.3.6.2 EIA RS232C/CCITT Interface (Figure 6-4)

The EIA RS232C/CCITT EIA interface uses signal levels which vary between plus and minus seven volts. The interface provides two control signals in addition to receive/transmit data signals. The interface signal levels are as follows:

Data:	Mark = -7 Vdc
	Space = +7 Vdc
Control:	True = +7 Vdc
	False = -7 Vdc

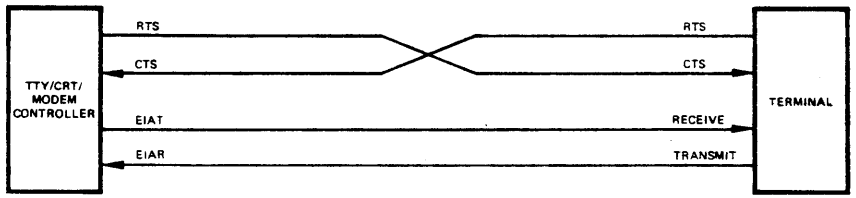
The controller EIA receive signal is designated EIAR- and is available on J2 pin S. The EIA transmit signal is designated EIAT- and is available on J2 pin 3. The two EIA control signals are Request to Send (RTS) and Clear to Send (CTS). RTS is available at J2 pin 4 while CTS enters the interface at J2 pin T.

The RTS and CTS lines from both the controller and terminal devices are defined for operation with a modem. When operating without a modem (direct interface as shown in figure 6-5a), the RTS and CTS lines must be crossed.

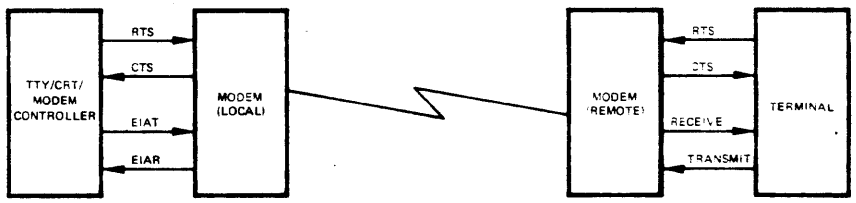
With the RTS and CTS control lines crossed, Half-duplex switching from Receive mode to Transmit mode and vice-versa is controlled by the controller RTS line. When the controller RTS line is true, the terminal device transmits to the controller. When the controller RTS line is false, the controller transmits to the terminal device. During Full-duplex operation, the RTS line of both the controller and the terminal device must be true for simultaneous transmission.

When operating with a Half-duplex modem, carrier keying by means of the RTS signal is not used to switch from Transmit to Receive modes. Instead, End-of-Message (EOM) character detection within the support software is used. When operating with a Full-duplex modem, no special disciplines are required.

The RTS signal is generated by the controller Motor On/Off flip-flop. The Motor On/Off flip-flop has delay circuitry which disables the controller Sense multiplexer for 600 ms after receipt of a Motor On command. When using the Motor On/Off flip-flop with an EIA device, the delay circuitry must be disabled. The delay circuits are disabled by grounding the ORIN- input, J1 pin 27 or J2 pins D and 15. Note that RTS and Motor On are in opposite sense. That is, a Motor On instruction turns RTS off.



a. Interface Without Modem



b. Interface With Modem

Figure 6-5. EIA RS232C/CCITT Interface

6.3.6.3 TTL/DTL Compatible Interface (Figure 6-5)

The TTL/DTL Compatible (TTL) interface uses signal levels which vary from 0 to +5 volts dc. The interface signal levels are as follows:

Mark = 0.0 to +0.45 Vdc
Space = 2.4 to +5.0 Vdc

The TTL receive signal is SMDAT- which is available at J1 pin 32 and J2 pin U. SMDAT- should be driven by an open-collector driver in the terminal device. The controller represents only one load to the driver. The controller provides a 1K ohm pull-up resistor to +5 Vdc. The TTL transmit signal is DTDAT and is available on J1 pin 46 and J2 pin 2. DTDAT is driven by the controller with an open-collector driver which is capable of 50 milliamps dc drive current. The terminal device must provide a pull-up resistor to the terminal VCC supply which must not exceed 100 volts dc.

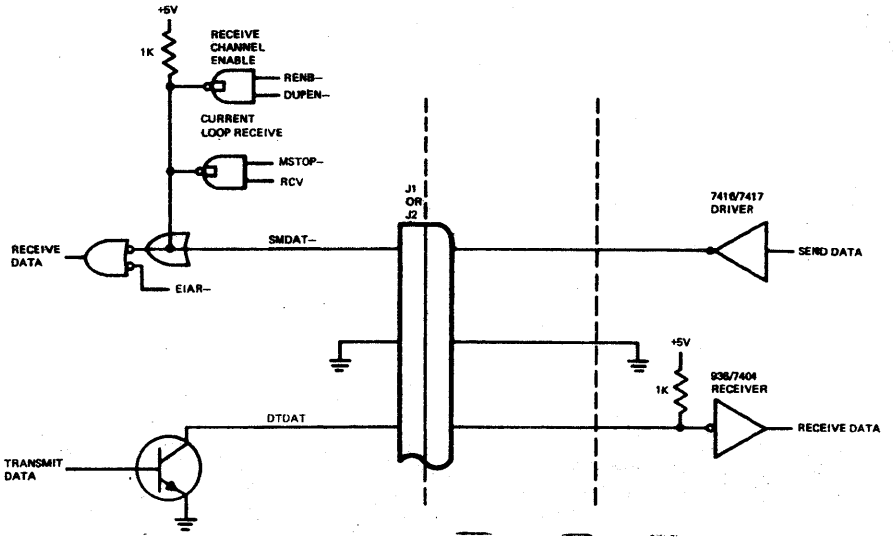


Figure 6-6. TTL/DTL Interface

6.3.7 Special Teletype Controls

The Teletype/CRT controller contains provisions which permit user generated software to control Paper Tape Reader and drive motor turnon and turnoff in specially modified ASR-33 Teletype units.

The reader control signal is designated IRDRA and is available at J2 pins V and 1. The motor control signals are referred to as MOT+ and MOT- and are available at J2 pins M and 8, and L and 9, respectively.

6.3.8 Half-Duplex Usage

Half-duplex controller operations involve either input from, or output to, the terminal device, but not simultaneously. Use of the Auto Echo feature causes input from the device to be automatically "echoed" back for printing or display, thus eliminating the necessity for echoing characters back under software control.

The following figures are examples of typical Half-duplex teletype I/O sequences:

```
LABEL   INST    OPERANDS    COMMENTS

        SBM                 Set Byte Addressing Mode
        SEL     7,4         Initialize TTY Interface
LOOP    LDAB    *DATA       Load Byte/Character into LS Byte
                            of A Register
        IMS     DATA        Increment Byte Address Pointer
        WRA     7,1         Output Byte when TTY is Ready
        IMS     COUNT       Increment Negative Number of
                            Characters to be Transferred
        JMP     LOOP        Continue Data Output if Non-zero
                            Increment Results
        SEN     7,1         Wait for last character to be printed
        JMP     $-1
        SWM                 Restore Word Addressing Mode
        .                   Exit
        .
        .
```

Figure 6-7. Half-Duplex Program-Controlled Data Output

```
LABEL    INST    OPERANDS    COMMENTS

         SBM                 Set Byte Addressing Mode
         SEL     7,0         Enable Auto Echo to Print Data
                             Being Input
         SEL     7,3         Start the Paper Tape Reader in a
                             Continuous Read Mode
LOOP     RBA     7,1         Input Byte when TTY is Ready
         STAB    *DATA       Store Character in Data Buffer in
                             Memory
         IMS     DATA        Increment Byte Address Pointer
         IMS     COUNT       Increment Negative Number of
                             Characters to be Transferred
         JMP     LOOP        Continue Data Input if Non-zero
                             Increment Results
         SEL     7,4         Initialize the TTY Interface to Stop
                             the Paper Tape Reader and Disable
                             the Auto Echo
         .
         .
         SWM                 Restore Word Addressing Mode
         .
         .
```

Figure 6-8. Program-Controlled TTY Reader Input

The standard Word interrupt location for Half-duplex operation is : 0002. The controller interrupts to this location when the Word Transfer mask is set, interrupts are enabled, and the terminal device is ready for either input or output. A jumper option allows this interrupt location to be relocated to location : 0022. The standard End-of-Block interrupt location for Half-duplex operation of the terminal device is location : 0006. The controller interrupts to this location when the Block Transfer mask is set, interrupts are enabled,and an ECHO signal (from completion of an Auto I/O interrupt sequence) is received from the Processor. A jumper option allows this interrupt location to be relocated to location : 0026. An additional jumper option allows Processor mounted option interrupts to be offset by : 0100 locations. The standard Half-duplex controller interrupts can thus be relocated to locations : 0102 and : 0106 or : 0122 and : 0126.

6.3.9 Half-Duplex Controller Instructions

SEL 7,0 ENABLE AUTO ECHO. Places controller in Read mode and causes all inputs to be echoed back to source terminal for printing or display. Initialize instruction (SEL 7,4) turns Auto Echo off.

SEL 7,1 SELECT KEYBOARD. Places controller in Read mode.

SEL 7,2 STEP READ. Places controller in Read mode and causes character under Paper Tape Reader read station to be read. Paper tape is then advanced one character position. Reader switch must be in START position.

SEL 7,3 CONTINUOUS READ. Places controller in Read mode and causes TTY Paper Tape Reader to read continuously until reader is stopped or tape runs out. Reader switch must be in START position.

SEL 7,4 INITIALIZE CONTROLLER. Places controller in Half-duplex and Write modes, and resets all control flags. Static marking condition will be present.

SEL 7,5 ENABLE WORD TRANSFER INTERRUPTS. Sets appropriate interrupt mask to enable generation of a Word interrupt each time Buffer Ready condition occurs

SEL 7,6 ENABLE END-OF-BLOCK INTERRUPT. Sets appropriate interrupt mask to enable generation of an EOB interrupt upon reception of ECHO signal from Processor. Instruction must be executed after SEL 7,5 or immediate EOB interrupt will occur.

SEL 7,7 DISABLE INTERRUPTS. Disable both Word and EOB interrupts by resetting both interrupt enable masks.

SEN 0,4 SENSE TTY CONTROLLER INSTALLED. Tests for presence of TTY controller on Option board. If controller is installed, next sequential instruction is skipped. If controller is not installed, next sequential instruction is executed. (Used by diagnostic programs.)

SEN 7,1 SENSE BUFFER READY. Tests for Buffer Ready condition. If buffer is ready, next sequential instruction is skipped. If buffer is not ready, next sequential instruction is executed.

SEN 7,2 SENSE WORD TRANSFER INTERRUPTS ENABLED. Tests if Word interrupts are enabled. If they are, next sequential instruction is skipped. If they are not, next sequential instruction is executed.

SEN 7,3 SENSE CONTROLLER NOT BUSY. Tests busy state of controller. If controller is not busy processing a character, next sequential instruction is skipped. If controller is busy, next sequential instruction is executed.

SEN 7,4 SENSE CLEAR TO SEND. Tests CTS line from a CRT or modem. If signal is true, next sequential instruction is skipped. If signal is false, next sequential instruction is executed. (This feature is available only with EIA RS232C/CCITT interface option.)

SEN	7,5	SENSE TTY MOTOR ON. Tests if TTY motor is on. If it is on, next sequential instruction is skipped. If it is off, next sequential instruction is executed.
SEN	7,6	SENSE PARITY ERROR. Tests for occurrence of parity error during most recent input operation. If a parity error occurred, next sequential instruction is skipped. If a parity error did not occur, next sequential instruction is executed. (Requires prior strapping of parity option at rear-edge connector.)
SEN	7,7	SENSE FULL DUPLEX MODE ENABLED. Tests if controller is in Full-duplex mode. If it is, next sequential instruction is skipped. If it is not, next sequential instruction is executed.
OTZ	7,6	TURN MOTOR ON. Turns TTY motor on and places controller in Write mode. Turning motor on introduces a 600 ms delay for all controller Sense responses and interrupts to allow motor to come up to speed. (This feature is only available if TTY has been modified for remote motor on/off control.)

NOTE

Motor is unconditionally turned on whenever a Power-up or System reset occurs.

OTZ	7,6	CLEAR REQUEST TO SEND. When used with a CRT or modem, this instruction turns off RTS signal and places controller in Write mode. (This feature is available only with EIA RS232C/CCITT interface option.)
OTZ	7,7	TURN MOTOR OFF. Turns TTY motor off and places controller in Write mode.
OTZ	7,7	REQUEST TO SEND. When used with a CRT or modem, this instruction turns on RTS signal and places controller in Write mode. (This feature is available only with EIA RS232C/CCITT interface option.)
OTA OTX	7,0 7,0	OUTPUT A OR X REGISTER TO CONTROLLER. Unconditionally transfers contents of LS byte of specified register to controller and causes character to be transmitted to terminal device.
WRA WRX	7,1 7,1	WRITE FROM A OR X REGISTER TO CONTROLLER. Tests for Output buffer empty condition. If buffer is empty, contents of LS byte of specified register are transferred to controller and subsequently transmitted to terminal device. If buffer is not empty, instruction is continuously repeated until it becomes empty.

AOT	7,0	OUTPUT WORD FROM MEMORY TO CONTROLLER, AUTOMATICALLY. Contents of LS byte of memory location addressed by updated AOT address pointer are unconditionally transferred to controller and subsequently transmitted to terminal device. (Refer to Auto I/O instructions in section 4.)
AOB	7,0	OUTPUT BYTE FROM MEMORY TO CONTROLLER, AUTOMATICALLY. Contents of memory byte location addressed by updated AOB address pointer are unconditionally transferred to controller and subsequently transmitted to terminal device. (Refer to Auto I/O instructions in section 4.)
BOT	7,1	OUTPUT BLOCK FROM MEMORY TO CONTROLLER. Places controller in Write mode and tests for Output buffer empty condition. When buffer is empty, contents of LS byte of effective memory location are transferred to controller, and subsequently transmitted to terminal device. Word count is decremented by one. Instruction is repeated continuously until word count is decremented to zero. (Refer to Block I/O instructions in section 4.)
INA INX	7,0 7,0	INPUT WORD FROM CONTROLLER TO A OR X REGISTER. Unconditionally transfers contents of Input buffer to LS byte of specified register. MS byte of specified register is set to zero.
IBA IBX	7,0 7,0	INPUT BYTE FROM CONTROLLER TO A OR X REGISTER. Unconditionally transfers contents of Input buffer to LS byte of specified register. MS byte of register is unaffected.
RDA RDX	7,1 7,1	READ WORD FROM CONTROLLER TO A OR X REGISTER. Tests for Input buffer full condition. If buffer is full, contents are transferred to LS byte of specified register. MS byte of specified register is set to zero. If buffer is not full, instruction is continuously repeated until it becomes full.
RBA RBX	7,1 7,1	READ BYTE FROM CONTROLLER TO A OR X REGISTER. Tests for Input buffer full condition. If buffer is full, contents are transferred to LS byte of specified register. MS byte of specified register is unaffected. If buffer is not full, instruction is continuously repeated until it becomes full.
AIN	7,0	INPUT WORD FROM CONTROLLER TO MEMORY, AUTOMATICALLY. Unconditionally transfers contents of Input buffer to LS byte of memory location addressed by updated AIN address pointer. MS byte of memory location is set to zero. (Refer to Auto I/O instructions in section 4.)

AIB 7,0 INPUT BYTE FROM CONTROLLER TO MEMORY, AUTOMATICALLY. Unconditionally transfers contents of Input buffer to memory byte location addressed by updated AIB address pointer. (Refer to Auto I/O instructions in section 4.)

BIN 7,1 INPUT BLOCK FROM CONTROLLER TO MEMORY. Tests for Input buffer full condition. When buffer is full, contents are transferred to LS byte of effective memory location. MS byte of memory location is set to zero and word count is decremented by one. Instruction is repeated continuously until word count is decremented to zero. Refer to Block I/O instructions in section 4.)

6.3.10 Full-Duplex Usage

Full-duplex controller operations allow simultaneous input and output. The interface contains two data buffers in this mode - one for input and one for output. Use of the Auto Echo feature causes input from the device to be automatically "echoed" back for printing or display, thus eliminating the necessity for echoing characters back under software control. When this feature is used, normal output data and echoed data can be intermixed but care should be taken to assure that the resulting sequence of output characters makes sense.

Full-duplex operation also allows use of a special "loop-back" diagnostic feature. This mode is entered by executing the Select and Present instructions SEA 7,4 or SEX 7,4 with the appropriate register (A or X) contents equal to 3. This feature connects the Output buffer to the Input buffer, allowing immediate comparison of transmitted data and received data. Figure 6-9 is an example of full-duplex data-input under interrupts.

In the example, a 20-character "question" is transferred to the TTY. A one-character "answer", entered at the keyboard is also printed but not before printing of the question is complete.

If printing of the question is not completed when the answer is entered, the -1 byte count is incremented to zero and the processor issues an ECHO-. Upon receiving ECHO-, the controller generates an EOB interrupt to location :26. Location :26 contains a JST to the EOB routine (ENDA). The program then waits for completion of the output byte transfer and the EOB interrupt. When it occurs, the A register is cleared and the EOB routine for byte input initializes the output interrupt sequence for output. The answer is then printed completing the example.

	Standard Location	Offset Location	Priority
Output Word Transfer Interrupt	:0002	:0102	4
Output EOB Interrupt	:0006	:0106	2
Input Word Transfer Interrupt	:0022	:0122	3
Input EOB Interrupt	:0026	:0126	1

The jumper option for offsetting interrupt locations to :0022 and :0026 (or :0122 and :0126) in the Half-duplex has no effect on the interrupt locations for Full-duplex operation. Note that the EOB interrupts have priority over the word interrupts.

6.3.11 Full-Duplex Controller Instructions

SEL 7,0 ENABLE AUTO ECHO. Causes all inputs to be echoed back to source terminal for printing or display. Initialize instructions (SEL/SEA/SEX 7,4) turns Auto Echo off.

SEL 7,1 SELECT KEYBOARD. Turns off Paper Tape Reader if on, without affecting any other operation.

SEL 7,2 STEP READ. Causes character under Paper Tape Reader read station to be read. Paper tape is then advanced one character position. Reader switch must be in START position.

SEL 7,3 CONTINUOUS READ. Causes TTY Paper Tape Reader to read continuously until reader is stopped or tape runs out. Reader switch must be in START position.

SEL 7,4 INITIALIZE CONTROLLER TO HALF-DUPLEX. Places controller in Half-duplex and Write modes, and resets all control flags. Static marking condition will be present.

SEA 7,4
SEX 7,4
(A or X = 1) INITIALIZE CONTROLLER TO FULL-DUPLEX. Either instruction (with appropriate register =1) will place controller in Full-duplex mode and reset all control flags.

SEA 7,4
SEX 7,4
(A or X = 3) INITIALIZE CONTROLLER TO FULL-DUPLEX DIAGNOSTIC. Either instruction (with appropriate register = 3) will place controller in Full-duplex mode and reset all control flags. In addition, the Output buffer is connected to the Input buffer. Any character which is output will be received by the Input buffer.

SEL 7,5 ENABLE OUTPUT WORD TRANSFER INTERRUPT. Sets appropriate interrupt mask to enable generation of an Output Word interrupt each time Output buffer empty condition occurs.

SEA 7,5
SEX 7,5
(A or X = 1) ENABLE INPUT WORD TRANSFER INTERRUPTS. Sets appropriate interrupt mask to enable generation of Input Word interrupt each time Input buffer full condition occurs.

SEL 7,6 ENABLE OUTPUT END-OF-BLOCK INTERRUPT. Sets appropriate interrupt mask to enable generation of Output EOB interrupt upon reception of ECHO signal from Processor, generated as a result of

```
LABEL/
LOCATION    INST.     OPERANDS      COMMENTS

:2          AOB       7,1           Automatic byte output
            DATA      -20           Negative byte count
            BAC       OBUF-1        Address of output buffer-1
            .
            .
            .
:6          ZAR                     End-of-block termination
            .
            .
:22         AIB       7,0           Automatic byte input
            DATA      -1            Negative byte count
            DATA      IBUF-1        Address of input buffer-1
            .
            .
:26         JST       ENDA          End-of-block termination
            .
            .
Main Memory .
START       LAP       1             Set A to +1
GO          SEA       7,4           Set full duplex
            SEL       7,5           Enable word output mask
            SEL       7,6           Enable EOB output mask
            SEA       7,5           Enable word input mask
            SEA       7,6           Enable EOB input mask
            EIN                     Enable interrupts
            WAIT                    Wait for interrupts
            .
            .
            .
            .
ENDA        ENT                     Entry for input done
            EIN                     Enable interrupt
            JAN       $             Wait for line output interrupts
            DIN                     Disable interrupts
            LAM       1             Setup automatic output or input character
            STA       :3
            LDA       IBUFA
            STA       :4
            LDA       DONE
            STA       6
            ZAR
            JMP       GO            Go do it
            .
            .
            .
FINISH      ENT                     Done!
            SEL       7,7           Turn off all masks
            LAM       20            Re-setup output and input instructions
            STA       :3
            LDA       OBUFA         For next time
            STA       :4
            LDA       ZAR
            STA       :6
            LDA       IBUFA
            STA       :24
            LAM       1
            STA       :23
            .
            .
            .
            .
IBUFA       BAC       IBUF-1
OBUFA       BAC       OBUF-1
ZAR         ZAR
DONE        JST       FINISH
            .
            .
            .
IBUF        DATA      $-$
OBUF        'SOURCE INPUT IS - "
            DATA      :8A8D  CR and LF
```

Figure 6-9. Full-Duplex Auto-Input Under Interrupt

Output Word interrupt. Instruction must be executed after SEL 7,5 or immediate Output EOB interrupt will occur.

SEA 7,6
SEX 7,6
(A or X = 1)

ENABLE INPUT END-OF-BLOCK INTERRUPT. Either instruction (with appropriate register = 1) will set appropriate mask to enable generation of Input EOB interrupt upon reception of ECHO signal from Processor, generated as a result of Input Word interrupt. Instruction must be executed after SEA/SEX 7,5 or an immediate Input EOB interrupt will occur.

SEL 7,7

DISABLE OUTPUT WORD TRANSFER AND END-OF-BLOCK INTERRUPTS. Disables both Output Word and EOB interrupts by resetting corresponding interrupt enable masks.

SEA 7,7
SEX 7,7
(A or X = 1)

DISABLE INPUT WORD TRANSFER AND END-OF-BLOCK INTERRUPTS. Either instruction (with appropriate register = 1) will disable both Input Word and EOB interrupts by resetting corresponding interrupt enable masks.

SEN 0,4

SENSE TTY CONTROLLER INSTALLED. Tests for presence of TTY controller on Option board. If controller is installed, next sequential instruction is skipped. If controller is not installed, next sequential instruction is executed. (Used by diagnostic programs.) The buffer is full, next sequential instruction is skipped.

SEN 7,0

SENSE INPUT BUFFER FULL. Tests for Input buffer full condition. If buffer is not full, next sequential instruction is executed.

SEN 7,1

SENSE OUTPUT BUFFER EMPTY. Tests for Output buffer empty condition. If buffer is empty, next sequential instruction is skipped. If buffer is not empty, next sequential instruction is executed.

SEN 7,2

SENSE OUTPUT WORD TRANSFER INTERRUPTS ENABLED. Tests if Output Word interrupts are enabled. If they are, next sequential instruction is skipped. If they are not, next sequential instruction is executed.

SEN 7,3

SENSE CONTROLLER NOT BUSY. Tests busy state of controller. If controller is not busy processing a character, next sequential instruction is skipped. If controller is busy, next sequential instruction is executed.

SEN 7,4

SENSE CLEAR TO SEND. Tests CTS line from a CRT or modem. If signal is true, next sequential instruction is skipped. If signal is false, next sequential instruction is executed. (This feature is available only with EIA RS232C/CCITT interface option.)

SEN 7,5

SENSE TTY MOTOR ON. Tests if TTY motor is on. If it is on, next sequential instruction is skipped. If it is off, next sequential instruction is executed.

SEN 7,6

SENSE PARITY ERROR. Tests for occurrence of parity error during most recent input operation. If a parity error occurred, next sequential instruction is skipped. If a parity error did not occur, next sequential instruction is executed. (Requires prior strapping of parity option at rear-edge connector.)

SEN 7,7

SENSE FULL DUPLEX MODE ENABLED. Tests if controller is in Full-duplex mode. If it is, next sequential instruction is skipped. If it is not, next sequential instruction is executed.

OTZ 7,6

TURN MOTOR ON. Turns TTY motor on. Turning motor on introduces a 600 ms delay for all controller Sense responses and interrupts to allow motor to come up to speed. (This feature is only available if TTY has been modified for remote motor on/off control.)

NOTE

Motor is unconditionally turned on whenever a Power-up or System reset occurs.

OTZ 7,6

CLEAR REQUEST TO SEND. When used with a CRT or modem, this instruction turns off RTS signal. (This feature is available only with EIA RS232C/CCITT interface option.)

OTZ 7,7

TURN MOTOR OFF. Turns TTY motor off.

OTZ 7,7

REQUEST TO SEND. When used with a CRT or modem, this instruction turns on RTS signal. (This feature is available only with EIA RS232C/CCITT interface option.)

OTA 7,0
OTX 7,0

OUTPUT A OR X REGISTER TO CONTROLLER. Unconditionally transfers contents of LS byte of specified register to controller Output buffer and causes character to be transmitted to terminal device.

WRA 7,1
WRX 7,1

WRITE FROM A OR X REGISTER TO CONTROLLER. Tests for Output buffer empty condition. If buffer is empty, contents of LS byte of specified register are transferred to controller Output buffer and subsequently transmitted to terminal device. If buffer is not empty, instruction is continuously repeated until it becomes empty.

AOT 7,0

OUTPUT WORD FROM MEMORY TO CONTROLLER, AUTOMATICALLY. Contents of LS byte of memory location addressed by updated AOT address pointer are unconditionally transferred to controller Output buffer and subsequently transmitted to terminal device. (Refer to Auto I/O instructions in section 4.)

AOB	7,0	OUTPUT BYTE FROM MEMORY TO CONTROLLER, AUTOMATICALLY. Contents of memory byte location addressed by updated AOB address pointer are unconditionally transferred to controller Output buffer and subsequently transmitted to terminal device. (Refer to Auto I/O instructions in section 4.)
BOT	7,1	OUTPUT BLOCK FROM MEMORY TO CONTROLLER. Tests for Output buffer empty condition. When buffer is empty, contents of LS byte of effective memory location are transferred to controller Output buffer and subsequently transmitted to terminal device. Word count is decremented by one. Instruction is repeated continuously until word count is decremented to zero. (Refer to Block I/O instructions in section 4.)
INA INX	7,0 7,0	INPUT WORD FROM CONTROLLER TO A OR X REGISTER. Unconditionally transfers contents of controller Input buffer to LS byte of specified register. MS byte of specified register is set to zero.
IBA IBX	7,0 7,0	INPUT BYTE FROM CONTROLLER TO A OR X REGISTER. Unconditionally transfers contents of controller Input buffer to LS byte of specified register. MS byte of register is unaffected.
RDA RDX	7,0 7,0	READ WORD FROM CONTROLLER TO A OR X REGISTER. Tests for Input buffer full condition. If buffer is full, contents are transferred to LS byte of specified register. MS byte of specified register is set to zero. If buffer is not full, instruction is continuously repeated until it becomes full.
RBA RBX	7,0 7,0	READ BYTE FROM CONTROLLER TO A OR X REGISTER. Tests for Input buffer full condition. If buffer is full, contents are transferred to LS byte of specified register. MS byte of specified register is unaffected. If buffer is not full, instruction is continuously repeated until it becomes full.
AIN	7,0	INPUT WORD FROM CONTROLLER TO MEMORY, AUTOMATICALLY. Unconditionally transfers contents of controller Input buffer to LS byte of memory location addressed by updated AIN address pointer. MS byte of memory location is set to zero. (Refer to Auto I/O instruction in section 4.)
AIB	7,0	INPUT BYTE FROM CONTROLLER TO MEMORY, AUTOMATICALLY. Unconditionally transfers contents of controller Input buffer to memory byte location addressed by updated AIB address pointer. (Refer to Auto I/O instructions in section 4.)
BIN	7,0	INPUT BLOCK FROM CONTROLLER TO MEMORY. Tests for Input buffer full condition. When buffer is full, contents are transferred to LS byte of effective memory location. MS byte of memory location is set to zero and word count is decremented by one. Instruction is repeated continuously until word count is decremented to zero. (Refer to Block I/O instructions in section 4.)

6.4 REAL TIME CLOCK

The Real Time Clock (RTC) option provides a means to determine elapsed time and/or creating a time-of-day clock, with software. The RTC keeps time by responding to electrical pulses of a known frequency, such as the output of a crystal oscillator or the input frequency of an ac power source. The standard configuration uses a 20 MHz crystal oscillator as the basic timing source. The 20 MHz clock is applied to a counter chain to produce 10 kHz, 1 kHz and 100 Hz clock sources (timing increments of 100 µs, 1 ms and 10 ms, respectively). In addition, a 120 Hz clock source is available (100 Hz when the computer is used with 50 Hz power source). The desired clock source is selected by a jumper wire. An external timing source may be applied to the RTC option if some source other than the crystal oscillator or twice the ac line frequency is desired. This allows the use of almost any timing period.

6.4.1 Clock Source Selection

With no jumper installed, the RTC option operates from a built in 100 Hz timing source. The user can select four other timing sources (10 kHz, 1kHz, twice the ac line frequency (TTLF) or a TTL compatible external timing source).

The RTC option represents only one TTL load to the external timing source. The external timing source must be a TTL compatible logic signal with rise and fall times of less than 50 ns. With regard to duty cycle, the only requirement is that the signal be ground true, with a minimum of 100 ns.

When the user desires to select an alternate timing source (other than the standard 100 Hz source), the 100 Hz clock source must be inhibited by grounding the INH- input. Clock source selection can be accomplished at connector J1 using table 6-3.

Table 6-3. Clock Source Selection

CLOCK SOURCE	INH- (pin 12)	JUMPER CONNECTIONS
100 Hz (standard)	OPEN	none
1,000 Hz	GND	Pin 39 to pin 11
10,000 Hz	GND	Pin 40 to pin 11
TTLF	GND	Pin 1 to pin 11
EXTERNAL*	GND	User Timing source to pin 11

*External timing source must be TTL/DTL compatible.

6.4.2 Discussion of Usage

If RTC interrupts are enabled, the RTC generates a Time interrupt to the Processor each time a clock pulse is detected from the clock source. This interrupt is usually

serviced by an IMS instruction at the interrupt location. Increment results of zero cause the generation of an ECHO signal to the RTC, which in turn generates a Sync interrupt to the Processor. The Sync interrupt is normally serviced by a JST instruction to an interrupt subroutine. The RTC has been assigned a device address of 8.

In the programming example shown in figure 6-10, an external device must be sampled once a second, using a 10 ms clock source.

```
      LABEL/
      LOCATION   INST      OPERANDS     COMMENTS

      (Time)
      : 0018 or  IMS       COUNT        Increment Timing Counter
      : 0118

      (Sync)
      : 001A or  JST       SYNC         Jump and Store to Interrupt
      : 011A                            Subroutine, Disable Interrupts.

                 .
Initialization   .
      INIT       LAM       100          Set Timing Count to -100.
                 STA       COUNT
                 SEL       8,4          Initialize RTC and Clear
                                        Unserviced Interrupt Requests.
                 SEL       8,2          Arm Sync-Allow Sync Interrupts
                                        when ECHO is Received.
                 SEL       8,0          Enable RTC-Allow Generation
                 .                      of Time and Sync Interrupts (Since
                                        Sync is Armed).
Interrupt Subroutine
      SYNC       ENT                    Reserved Location for Storage
                 .                      of P Register
                 .
                                        Save Contents of Registers, Status,
                                        etc. (see paragraph 5.3)
                 .
                 .
                 LAM       100          Reset Timing Counter to -100.
                 STA       COUNT
                 .
                 .
                 EIN                    Enable Interrupts.
                 RTN       SYNC         Return to Mainline Program.
                 .
                 .
      COUNT      DATA      0
```

Figure 6-10. RTC Interrupt Programming Example

The timing counter COUNT becomes zero after being incremented 100 times, i.e., after 100 Time interrupts, each 10 ms apart. The RTC responds to the resulting ECHO signal by generating a Sync interrupt which is serviced by the interrupt subroutine SYNC. The timing counter COUNT is reset to -100 and the external device is sampled.

6.4.3 Summary

6.4.3.1 RTC Interrupt Locations

Time Interrupt location: : 0018 (offset = : 0118)
Sync Interrupt location: : 001A (offset = : 011A)

6.4.3.2 RTC Instructions

SEL	8,0	ENABLE RTC. Allows Time and Sync interrupts to be generated (if Sync is armed).
SEL	8,2	ARM SYNC. Allows generation of Sync interrupts if RTC is enabled and ECHO received.
SEL	8,3	CLEAR RTC INTERRUPTS. Resets both Time and Sync interrupt requests. Does not disable or disarm interrupts, but instead removes interrupt request history from RTC.
SEL	8,4	INITIALIZE RTC. Disarms, disables, and clears interrupt requests.
SEL	8,7	DISARM SYNC. Prevents Sync interrupts from being generated without disabling Time interrupts.
SEN	0,2	SENSE RTC INSTALLED. Tests if RTC option is installed on Option board. If it is, next sequential instruction is skipped. If it is not, next sequential instruction is executed.

6.5 AUTOLOAD

6.5.1 Description

The Autoload option consists of a 256-word read-only memory (ROM) preprogrammed with a binary loader and the necessary logic to execute the loader. The autoload program is a complete binary program loader for TTY and high-speed paper tape (not just a bootstrap) and includes appropriate input format and data error checking.

For bulk storage devices, Autoload provides a first record bootstrap. Autoload requires the presence of the power fail/restart (PF/R) or automatic startup (ASU) processor option.

Autoload uses main memory locations :30 through :3B for scratchpad. A program occupying these addresses cannot be properly loaded using autoload.

The autoload sequence is initiated by depressing the console AUTO switch or, in configurations not using a console, by momentarily grounding a pin on the option board (see 6.5.7) . Upon execution, a binary program is automatically loaded into computer main memory from any one of the following input devices:

1. Teletype paper tape reader
2. High-speed paper tape reader
3. Nine-track magnetic tape unit
4. Cassette tape
5. Moving head disk

If more than one magnetic tape, cassette or disk drive device is used in the system, autoload will load from the device designated as device zero.

When selecting autoload from the console, the computer must be in the Run Enable mode (STOP indicator off) to enable the AUTO switch. AUTO is interlocked with the RUN switch so that Run mode is selected as autoload is initiated. A remote autoload command (grounding a pin on the option board) can be initiated at any time.

The presence of the autoload option can be sensed using the sense instruction with device address zero and function code zero. This instruction is used primarily by diagnostic and executive programs. The sense instruction takes the following form:

SEN 0,0 SENSE AUTOLOAD INSTALLED. Tests if autoload option is installed. If so, next sequential instruction is skipped. If autoload is not installed, the next sequential instruction is executed.

6.5.2 Device and Mode Selection

The input device and load mode (absolute or relocatable) is selected at the console sense register. In computer configurations not having a console, the sense register and certain console switch functions can be duplicated by the use of jumpers on the option board (secondary console); see paragraphs 6.6.3 and 6.6.4. A hex code entered into the sense register selects the following device and load mode:

	TTY	HSPT	MAG TAPE	CASSETTE	DISK
Load Absolute	:0	:1	:2	:3	:4
Load Relocatable	:8	:9	:A	:B	:C

If relocation is desired, the user enters the start address in the X register. If "load and execute" is desired, the SENSE switch is set (ON); for "load only", the SENSE switch must be reset (OFF).

6.5.3 Autoload Sequence

When autoload is initiated, the processor is placed in word mode, interrupts are disabled, and the power up sequence of the PF/R or ASU option generates a reset and starts the computer running at location :0000. Autoload ROM address space is :0000 through :00FF. Autoload logic causes all instruction cycles to fetch instructions from ROM (main memory disabled) and all data cycles to access memory. The first instruction is fetched from ROM location :0000. As the loader is executed, the program being read from the input device is treated as data and stored in memory.

6.5.4 Termination of Autoload

The action performed at the end of a successful load is dependent on the type of input device used and the position of the SENSE switch.

6.5.4.1 TTY and High-Speed Paper Tape Reader

Control is transferred to the start address of the loaded program if (1) the SENSE switch is set, and (2) a valid start address was on the tape. If the SENSE switch is reset or if no valid start address was on the tape, autoload halts with :0800 in the I register with the X register containing the next location available for loading. The start address in the A register will be negative (:FFFF) if a valid start address was not present.

6.5.4.2 Magnetic Tape, Cassette and Disk

Control is transferred to the start address of the loaded program if the SENSE switch was set. If the SENSE switch was reset, autoload halts with :0800 in the I register.

6.5.5 Error Detection

The standard autoload program detects checksum and format errors on paper tape devices. If an invalid checksum or format error is detected, the program halts with :0801 in the I register. The program may be restarted with the depression of AUTO. If an error occurs while attempting to load from paper tape, it is possible to backup the tape one record and press AUTO to continue. However, it is recommended that loads exhibiting errors be completely repeated.

If an error occurs while attempting to load from magnetic tape, cassette or disk, autoload will halt with :0801 in the I register, and may be retried by pressing AUTO.

6.5.6 Accessing Autoload ROM

The autoload ROM normally contains 256 words, but can be expanded to 512 words for special autoload sequences or for use as a high-speed read-only memory. To use it as a normal read-only memory or to read out the contents of ROM, the SEL 0,1 instruction is used. When enabled under program control, the ROM occupies addresses :7800 through :7FFF, modulo 512 (:7800 through :79FF, :7A00, :7BFF, etc.). Any memory access in this range is automatically disabled when autoload ROM is enabled.

An I/O instruction, SEL 0,0 is used to disable the autoload ROM. When disabling ROM, one additional access to ROM is required before the ROM is actually disabled. This allows a program resident in ROM to turn itself off and then jump to main memory.

A diagnostic feature allows verification of the autoload sequence. This is accomplished by setting the sense register to :F (all bits on), programming a halt at location 31, then initiating autoload. If autoload is sequencing properly, the program will exit to location :31 and halt.

6.5.7 Remote Autoload Initiation

A momentary grounding (i.e., a switch closure to ground) of pin 10 on connector J1 on the option card, or pins E or 14 on connector J2, causes the system to reset and an autoload program to be initiated. The signal must be ground-true for a minimum of 100 ns. This feature should be used only in conjunction with the secondary console sense register (paragraph 6.6.3).

6.5.8 Automatic Autoload (Upon restoration of power)

An autoload sequence can be automatically initiated upon restoration of power by jumpering J1 pins 20 (RMDIS-) to J1 pin 5 (PFAL-). This feature is particularly useful when using volatile memories without battery backup power in unattended operation. With this feature, memory is automatically reloaded with an operational program from a peripheral storage device after power is restored. This feature should be used only in conjunction with the secondary console sense register (paragraph 6.6.3).

6.5.9 Autoload Operation Summary (Console Operation)

Following is a summary of the procedures required to load programs into memory using autoload. For details of console operation, see section 3.

a) Enable console.
b) Press STOP to halt the computer (STOP indicator on).
c) Press RESET
d) If relocation desired, enter start address into X register.
e) Enter proper hex code for device and load mode into sense register.
f) If load and execute desired, set SENSE switch (on); if load and halt desired, reset SENSE switch (off).
g) Ready the load device.
h) Press STOP to enable RUN mode (STOP indicator off).
i) Press AUTO.

6.6 BASIC VARIABLES PACKAGE

The Basic Variables package permits the user to operate high priority (Processor) interrupts independent of EIN/DIN control, offset interrupts, extend I/O transfer timing, and perform certain console functions in the absence of a Console.

6.6.1 Independent Processor Interrupt Operation

In normal operation, the Power Fail, Console and Trap interrupts (referred to as Processor generated interrupts) will not be recognized by the Processor if interrupts are not enabled (DIN instruction has disabled recognition of ALL interrupts). The EIN instruction must be executed before any interrupts can be processed.

By grounding the OPT- signal (J1 pin 35), the Processor generated interrupts can obtain immediate recognition by the Processor when they are enabled.

With J1-35 grounded, the PFE and PFD instructions control the Power Fail/Restart interrupt while the CIE and CID instructions control Console interrupts. There are no control instructions for a Trap interrupt other than the TRP instruction itself.

6.6.2 Interrupt Offset

All interrupts (except Power-up) generated within the Processor and the Processor Option board may be relocated (offset) from the scratchpad area of Memory by : 100 locations to allow for more efficient utilization of the scratchpad area.

Two types of offset are available on connector J1. The high priority Processor interrupts (Power Fail, Console, and Trap) and the high priority user generated interrupts (IL1 and IL2) can be offset by grounding the OFST - signal (J1 pin 4). Likewise, the low priority Teletype/CRT controller and Real Time Clock option interrupts can be offset by grounding the MAI - signal (J1 pin 6).

6.6.3 Secondary Console Sense Register

The Basic Variables package contains four jumpers which permit the user to simulate the Console Sense register and develop a Console Sense word in the absence of a Console. The jumper inputs are DS00- (J1 pin 34), DS01- (J1 pin 33), DS02- (J1 pin 36) and DS03 (J1 pin 31). DS00- is the least significant bit of the simulated register, while DS03- is the most significant bit. Grounding a particular jumper input introduces a logic 1 into the corresponding bit position of the Console Sense word. A logic 0 is introduced when a given input is left open.

The entire simulated register is enabled by grounding the ENDSW- signal which is available at J1 pin 28. Note that all control logic required to respond to the ISA (:5801) and ISX (:5A01) instructions is also provided with this feature. This feature cannot be used when a Console is installed.

6.6.4 Secondary Console Switch Functions

Secondary console SENSE, RESET and INT switch signals which duplicate the functions of the Console are available to the user. The SSW- signal (J1 pin 2) duplicates the SENSE switch, RST- (J1 pin 37) duplicates the RESET switch and CINT- (J1 pin 38 and J2 pins F and 13) duplicates the INT switch. These switch functions are generated by taking the input pin to ground (momentarily). RST must be ground-true a minimum of 5 μs. SSW- must remain at ground when the SENSE switch is active. These signals are collector-ORed with the corresponding console signals.

6.6.5 I/O Timing Extension

The Basic Variables package features an I/O stretch capability which permits the user to slow down the I/O transfer timing when driving the Maxi-Bus through multiple expansion chassis or over long distances. Four strap connections (STR1, STR2, STR3 and STR4) permit the user to specify 16 different increments of stretch. The LSI-1 uses stretch increments of 200 ns while the LSI-2 has stretch increments of 100 ns. Based on these increments, the LSI-1 stretch can range from 0 to 3000 ns while the LSI-2 stretch can range from 0 to 1500 ns.

Note that whenever any stretch is inserted, all I/O timing throughout the system is slowed down. This can have an adverse effect on speed critical I/O devices and in general reduces processor throughput. The stretch strapping scheme for both the LSI-1 and LSI-2 is shown in table 6-4. Ground is available on pins 23 through 26 of connector J1.

Table 6-4. I/O Stretch Selection

STRETCH ADDITION (Nsec)		STRAP CONFIGURATION			
LSI-1	LSI-2	STR4 (J1-44)	STR3 (J1-43)	STR2 (J1-42)	STR1 (J1-41)
0	0	OPEN	OPEN	OPEN	OPEN
200	100	OPEN	OPEN	OPEN	GND
400	200	OPEN	OPEN	GND	OPEN
600	300	OPEN	OPEN	GND	GND
800	400	OPEN	GND	OPEN	OPEN
1000	500	OPEN	GND	OPEN	GND
1200	600	OPEN	GND	GND	OPEN
1400	700	OPEN	GND	GND	GND
1600	800	GND	OPEN	OPEN	OPEN
1800	900	GND	OPEN	OPEN	GND
2000	1000	GND	OPEN	GND	OPEN
2200	1100	GND	OPEN	GND	GND
2400	1200	GND	GND	OPEN	OPEN
2600	1300	GND	GND	OPEN	GND
2800	1400	GND	GND	GND	OPEN
3000	1500	GND	GND	GND	GND

6.7 POWER FAIL/RESTART

6.7.1 General

Power Fail/Restart (PFR) is an optional feature of the ALPHA LSI computer. It allows the computer to operate from unreliable ac power sources without the requirement of human monitors. A low power condition or a temporary power outage is detected in time for the operating program to prepare for the power loss. When power returns to normal, the computer is automatically restarted without loss of data or operating position. Thus, unattended operation is possible.

6.7.2 Power Fail

When a power failure is detected, a Power Fail interrupt is generated to the Processor. If the Power Fail interrupt is enabled, the Processor is interrupted to a reserved location in Memory (location : 001C or : 011C if offset). The Processor executes the instruction (usually a JST to a software power down routine) at that location. The Processor has 0.9 ms to complete the power down routine, once it is started, before the PFR option halts the computer and protects Memory from transient power conditions.

6.7.3 Restart

When PFR detects power restoration to an acceptable level, a power up sequence is started. PFR re-enables Memory, sets the P register to : 0000, and generates a Run signal to the computer. The computer then executes the instruction (normally a JMP to a software power up routine) at location : 0000. The computer always undergoes this sequence when power is applied. The software power up routine must be completed within 0.9 milliseconds to allow enough time to process a Power Fail interrupt if one should occur immediately after power up.

CAUTION

When the Power Fail/Restart option is installed, the computer will start running at location : 0000 when power is applied whether the computer was running or not (i.e., independent of Console setting) prior to removal of power. To avoid false starts, it is customary for the power down subroutine to save a flag indicating that the computer was in fact running before power failed.

6.7.4 Interrupt Control Option

A hardware wiring option may place the Power Fail interrupt outside EIN/DIN control. Under this option, it is necessary to execute the PFE or PFD instructions to enable or disable the Power Fail interrupt. Without the option, the EIN or DIN instructions must be executed and PFE and PFD have no effect.

6.7.5 Programming Examples

Figure 6-11 shows examples of simple Power Fail/Restart software routines. In these examples, the contents of the A and X registers, the computer status and the mainline program location at the time of the Power Fail interrupt are saved during the power down sequence and restored during the power up sequence. Note that

the Power Fail interrupt is outside EIN/DIN control in this example. If the Power Fail interrupt were inside EIN/DIN control, the power up routine would not have to include a PFE instruction and the power down routine would not have to include a PFD instruction.

```
LABEL/
LOCATION        INST    OPERANDS    COMMENTS

: 0000          JMP     UP          Power Up Interrupt Location. Contains
                                    Unconditional Jump to Power Up
                                    Subroutine.

Interrupt Location
: 001C or       JST     DOWN        Power Down Interrupt Location.
: 011C                              Contains a Jump and Store to Power
                                    Down Subroutine. Using JST Auto-
                                    matically Saves Contents of P Register
                                    and Disables Interrupts.

DOWN            ENT                 Reserved Location for Storage of P
                                    Register when JST Instruction at Power
                                    Down Interrupt Location is Executed.

                PFD                 Disable Further Power Fail Interrupts.

                SIN     1           Inhibit Byte Mode if Set.

                STA     ASAVE       Save A Register.

                SIA                 Read Computer Status Word to A
                                    Register, Set Word Mode, and Reset
                                    OV Indicator.

                STA     STATUS      Save Computer Status Word.

                ICA                 Input Console Data Register to A
                                    Register

                STA     CSAVE       Save Contents of Console Data Register.

                STX     XSAVE       Save X Register.

                IMS     PSTP        Save a Flag Indicating Computer Was
                                    Stopped by a Power Failure.

                WAIT                Wait for Power Down to Complete.
                .
                .
                .
UP              ZAR                 JMP Instruction at Power Up Interrupt
                                    Location Enters Here.
```

Figure 6-11. Power Fail/Restart Software Routines

LABEL	INST	OPERANDS	COMMENTS
	EMA	PSTP	Check Flag to See if Computer Was Stopped By a Power Failure. Reset Flag.
	JAN	$+2	
	HLT		No - Do Not Restart.
	LDX	XSAVE	Restore X Register.
	LDA OCA	CSAVE	Restore Contents of Console Data Register.
	LDA	STATUS	Load Computer Status into A Register then set Computer Status (Sense Switch, Data Switches, OV Indicator and Address Mode).
	SIN	5	Inhibit Byte Mode if Set.
	SOA		
	LDA	ASAVE	Restore A Register.
	PFE		Enable Power Fail.
	EIN		Enable Interrupts.
	JMP	*DOWN	Restart Main Program by Executing an Indirect Jump to Location Specified by Saved Contents of P Register.
ASAVE	DATA 0		A Register Save Location.
CSAVE	DATA 0		Console Register Save Location.
XSAVE	DATA 0		X Register Save Location.
STATUS	DATA 0		Computer Status Word Save Location.
PSTP	DATA 0		Flag Indicating Processor Was By a Power Failure.

Figure 6-11. Power Fail/Restart Software Routines (Continued)

6.8 AUTOMATIC START-UP (ASU)

Automatic Start-up is an optional feature that, like PFR, automatically starts the processor after a power failure. It is for use in applications where it is not required to save the processor conditions as they were prior to power failure. Operation is similar to that of PFR except that a power fail interrupt is not generated.

6.8.1 Restart

When ASU detects power restoration to an acceptable level, a power up sequence is started. ASU re-enables Memory, sets the P register to :0000, and generates a Run signal to the computer. The computer then executes the instruction (normally a JMP to a software power-up routine) at location :0000. The computer always undergoes this sequence when power is applied. The software power-up routine must be completed within 0.9 milliseconds to allow enough time to process a power fail interrupt if one should occur immediately after power up.

CAUTION

When the ASU option is installed, the computer will start running at location :0000 when power is applied whether the computer was running or not (i.e., independent of Console setting) prior to removal of power.

Section 7

MEMORY INTERLEAVING AND BANKING

7.1 INTRODUCTION

All LSI Series computers include provisions for Memory Interleaving and Memory Banking.

7.1.1 Memory Interleaving

Memory Interleaving allows memory modules to be paired so that even and odd addresses are assigned in different memory modules. Since a relatively high percentage of memory accesses are normally sequential, this feature allows alternate memory accesses to address different memory modules. The result of alternate module accesses is that the asynchronous Maxi-Bus can support a much higher data rate than would be possible without alternate accesses. DMA transfer rates for both LSI-1 and LSI-2, and execution times for LSI-2, can be improved substantially by use of interleaving. Execution time for LSI-1 is limited by computer speed rather than memory access time. Therefore, execution time in LSI-1 is not affected by interleaving.

7.1.2 Memory Banking

Memory Banking allows an optional Memory Bank controller to switch memory modules off and on so that up to 256K (K=1024) words of Memory can be used. Each memory module is individually controllable. A maximum of 32K words can be enabled at any given time. Switching between memory modules occurs in a single instruction time.

7.2 INTERCONNECTIONS

Each memory module includes a 16-pin integrated circuit socket(memory control connector) near the rear edge of the board for jumpering interleaving controls and for connection to an optional Memory Bank controller. Jumpering and cabling is done by using a standard 16-pin socket header. Pin-outs for the memory control connector are given in figure 7-1.

Four signals are used to control interleaving and banking. Memory modules operate in their normal mode when no connection is made to any of the four control signals.

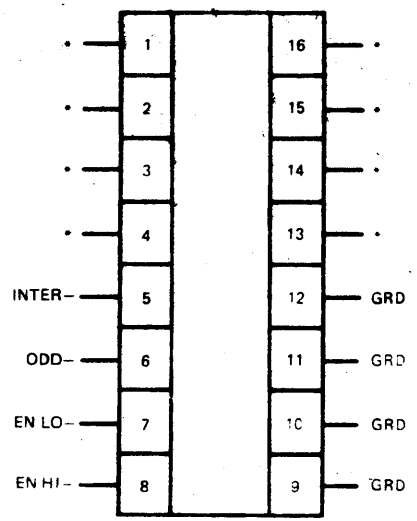
7.2.1 Memory Interleaving

When pin 5 (INTER-) is jumpered to pin 12 (GND), the memory module is set up to interleave and store even addresses only. When pin 6 (ODD-) is jumpered to pin 11 (GND) along with the pin 12 jumper, the memory module is set up to interleave and store odd

addresses only. Memory modules are always interleaved in pairs--one jumpered for even (pin 5 to pin 12) and one for odd (pin 5 to pin 12 and pin 6 to pin 11).

7.2.2 Memory Banking

Two enable signals allow the Memory Bank controller to switch memory modules on and off. The Memory Bank controller uses either high (+5 Volts) true enabling or low (0 Volts) true enabling, depending upon the particular system configuration. For low true enabling, the Memory Bank controller bank enable signal is connected to pin 7 (EN LO) on the memory control connector, and pin 8 (EN HI) is strapped to pin 9 (GND). For high true enabling the Memory Bank controller bank enable signal is connected to pin 8 (EN HI). Pins 9 and 10 may be used as a ground return when cabling to the Memory Bank controller.



*Reserved – No Connection Allowed

Figure 7-1. Memory Control Connector

7.3 USAGE AND INSTALLATION

The following paragraphs describe the usage and installation rules for Memory Interleaving and Memory Banking.

7.3.1 Memory Interleaving (Figure 7-2)

Memory modules are always interleaved in pairs of equal capacity or equal groups. When interleaving two equal sized modules, e.g., two 8K memory modules, one is strapped for even interleaving and one is strapped for odd interleaving. The two modules that are to be interleaved together must be installed in "adjacent" card slots with the odd strapped module closest to the Processor. Memories are considered "adjacent" as long as there is no intervening memory module and as long as the MBIN/MBOT, DPIN/DPOT and PRIN/PROT chains are properly chained through any intervening Input/Output or DMA controllers. (The last slot of the main chassis or expansion chassis is considered "adjacent" to the first slot in the next expansion chassis.)

If more than two equal sized memory modules are to be interleaved, they are treated in pairs with each pair strapped for one module interleaved odd and one module interleaved even. Each pair of modules is then installed with the odd strapped module first in each pair. If there is not an even number of equal sized memory modules to strap in pairs, the left over module(s) may be installed in any position as long as paired groups are not split. See figure 7-2 for examples of memory module installation.

Memory modules of unequal sizes may be interleaved together only when two or more memory modules are grouped together as the even half of a pair, and their total capacity is exactly equal to the capacity of the single module used as the odd half of the pair. For example, one 8K, one 4K and two 2K modules may be interleaved together if the 4K and two 2K modules are all strapped for even interleaving and paired as a group with the 8K module (see figure 7-2).

7.3.2 Memory Banking (Figure 7-3)

Memory Banking operation, memory installation rules, and cabling rules are discussed in the following paragraphs.

7.3.2.1 Operation

The operation of Memory Banking can best be understood by considering memory modules to be organized in a two dimensional matrix as shown in figure 7-3. Normally memory modules occupy unique address spans within the computer's total addressing range of 32K words. Memory Banking allows multiple memory modules to occupy the same address span at different times. A maximum of 32 memory modules may be attached to a Processor. Modules are organized as a matrix of Primary modules and Alternate modules. A maximum of 32K words of Memory may be assigned as Primary modules. The

A. Two 8K Modules

Processor
8K ODD
8K EVEN
———
———

B. Four 4K Modules

Processor
4K ODD
4K EVEN
4K ODD
4K EVEN

C. Three 8K Modules

Processor
8K ODD
8K EVEN
8K NORMAL
———

OR

Processor
8K NORMAL
8K ODD
8K EVEN
———

D. One 8K, one 4K, two 2K Modules

Processor
8K ODD
4K EVEN
2K EVEN
2K EVEN

Figure 7-2. Interleaved Memory Installation

remaining memory modules are Alternate modules. At power up time, and following a system RESET or Memory Bank controller initialization, the Primary Modules are all enabled and the Alternate modules are all disabled. The enabled modules can always be operated as though they were the only modules installed.

In the example of figure 7-3, there are four Primary modules, two 4K's and two 8K's. Following initialization, the computer therefore operates as a normal 24K computer using these modules. The two 4K modules are interleaved in this example and designated as Primary modules 00 odd and 00 even (P00 ODD and P00 EVEN). The two 8K modules are not interleaved in the example and are designated Primary modules 10 and 20 (P10 and P20). There are seven Alternate modules in this example. Each Alternate module can be assigned as the Alternate module for only one Primary module. For example, modules A11, A12 or A13 are the first, second and third alternates for Primary module 10. Under software control, the Memory Bank controller can disable P10 and enable A11, A12 or A13. Thus, a total of 32K words of Memory is available between addresses 8K and 16K, but only 8K of the 32K is available at any given time.

In addition to providing for memory expansion beyond 32K, Memory Banking provides a rapid context switching capability. For example, if module P20 contains an operating program which uses four sets of data (i.e., four users) at different times, modules P10, A11, A12 and A13 could each contain one set of data. Now the operating program can switch between data sets (users) in a single instruction. Detailed programming information is provided with the Memory Banking controller.

7.3.2.2 Memory Installation

When planning an installation using Memory Banking, a plan drawing similar to figure 7-3 should be prepared and each physical module assigned to a Primary module or Alternate module position according to the following rules:

1. There may be, at most, 32K words of Primary modules.

2. Primary module capacities and corresponding Alternate module capacities must be identical (e.g., P10, A11, A12 and A13) or Primary modules may be grouped, the sum of which has the same capacity as the corresponding Alternate module (e.g., P00 ODD plus P00 EVEN matches A02).

3. There may not be an Alternate module for which there is no corresponding Primary module.

4. A Primary module cannot be paired with an Alternate module of a different capacity, or with a group of smaller capacity modules, even if the smaller alternates sum to the same capacity as the Primary module. An exception is allowed for single alternates smaller than the primary, but only for the last primary (e.g., A22).

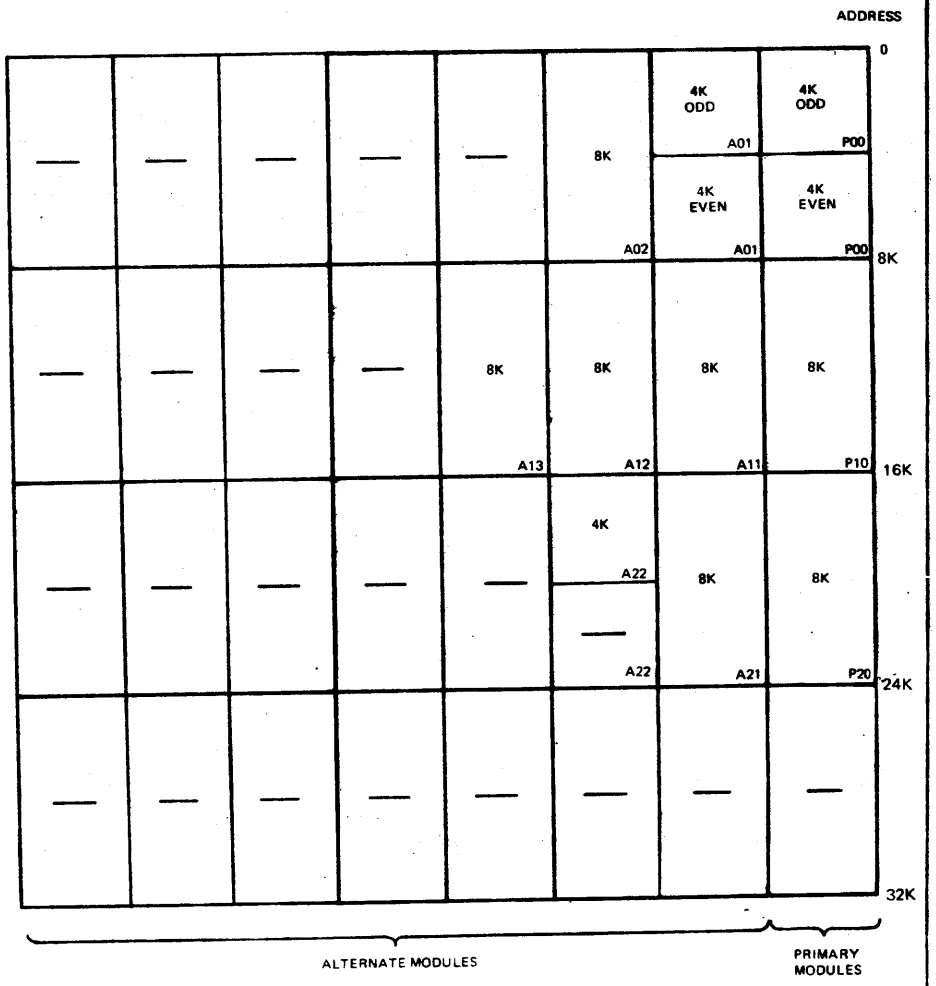


Figure 7-3. Memory Banking Example

5. When interleaved modules are banked, they must be banked in pairs (e.g., P00 consists of two interleaved 4K modules). Modules to be banked may be interleaved and an interleaved pair may be banked with a single module whose size is equal to the interleaved pair (e.g., A01 and P00 are composed of two interleaved 4K modules while A02 is a single 8K module).

6. After module positions are assigned, they must be installed in the following order beginning at the Processor:

   a. All alternates to Primary module 00 (the order of the alternates is immaterial).

   b. Primary Module 00.

   c. Remaining alternates and primaries with each set of alternates preceding their primary.

7. Any interleaved modules must obey the rules for interleaving given in paragraph 7.3.1.

7.3.2.3 Cabling

After modules are installed, they are cabled to the Memory Bank controller by connecting either the EN HI or EN LO memory control connector pin of each memory module to a control output of the Memory Bank controller. The following rules apply to cabling:

1. All Primary modules use EN LO.

2. All Alternate modules use EN HI.

3. Each interleaved module pair must have the appropriate EN lines connected together to a single Memory Bank controller output.

Cabling in this fashion guarantees that the Primary modules are selected at power up and initialization time since the Memory Bank controller resets with all outputs low.

Section 8

MAXI-BUS CHARACTERISTICS

8.1 INTRODUCTION

This section describes the signals and electrical characteristics of the NAKED MINI LSI Computer Maxi-Bus. Additionally, the distribution of the Maxi-Bus and the ALPHA LSI computer motherboard are discussed.

The Maxi-Bus consists of 58 lines (plus power and ground) that are used to convey address, data, and control information to or from the Processor, Memory, DMA controllers, and I/O controllers (see figure 8-1).

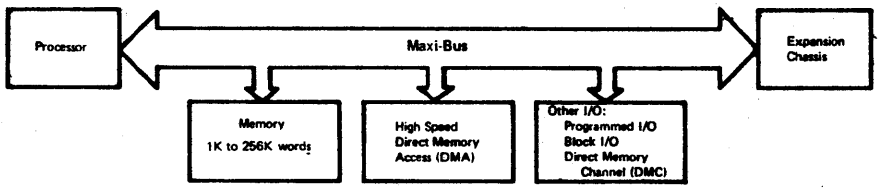


Figure 8-1. Maxi-Bus Configuration

The Maxi-Bus provides a common transfer path for all system modules. Maxi-Bus transfers involving Memory are asynchronous wherein the amount of time that signals from a source device spend on the Maxi-Bus depends upon the access and cycle time of the addressed memory module and not upon a fixed clock interval. All Maxi-Bus operations between the Processor and I/O controllers are synchronous and therefore do not require timing generation within I/O controllers.

All address and data signals, as well as memory control signals from a source device, must be driven by 32 mA tri-state drivers. Certain control signals that can be driven simultaneously by more than one device must use 32 mA open-collector drivers. Standard TTL receivers can be used by all devices. Only one receiver per line per module is permitted and the maximum receiver loading must not exceed 1.8 mA per module.

Address and data lines are shared by Memory and I/O devices. During communication intervals involving Memory, all bus drivers on these lines must be tri-state. During communication intervals involving standard I/O devices, bus drivers may be either tri-state or open collector.

8.2 MAXI-BUS COMPONENTS (Figure 8-2)

The ALPHA LSI computer Maxi-Bus consists of three major components: the Address bus (A), the Data bus (D), and the Control bus (C).

8.2.1 Address Bus (A)

The Address bus consists of 16 lines (AB00- through AB15-)that are time shared by the Processor and DMA controllers.

The Processor and DMA controllers use the 15 bits of the A bus to address memory locations. The 16th A bus bit (MSB) is used to specify word or byte memory operation. During I/O operations, the Processor uses the low order 8-bits of the A bus to convey device address and function code information to I/O devices. The high order 8-bits contain random information and are not normally used. The format of the low order 8-bits during I/O operations is as follows:

AB07-	Device Address bit 4	P Field
AB06-	Device Address bit 3	
AB05-	Device Address bit 2	
AB04-	Device Address bit 1	
AB03-	Device Address bit 0	
AB02-	Function Code bit 2	F Field
AB01-	Function Code bit 1	
AB00-	Function Code bit 0	

NOTE

The eight lines devoted to the device address and function code are arbitrarily divided into groups of five and three, respectively. They can be divided differently to increase or decrease the number of device addresses and function codes. For example, six lines can be devoted to the device address and only two to the function code. This would increase the number of device addresses to 64 and reduce the number of function codes to 4.

Throughout the remainder of this design guide, all examples which involve I/O addresses assume the arbitrary five and three division.

8.2.2 Data Bus (D)

The D bus consists of 16 bidirectional lines (DB00- through DB15-) that are time shared by the Processor, Memory, and I/O Interface controllers.

The Processor uses the D bus to read data from or write data into Memory. Likewise, the Processor uses the D bus to transfer data to or from an I/O controller.

A DMA controller uses the D bus to read data from or write data into Memory.

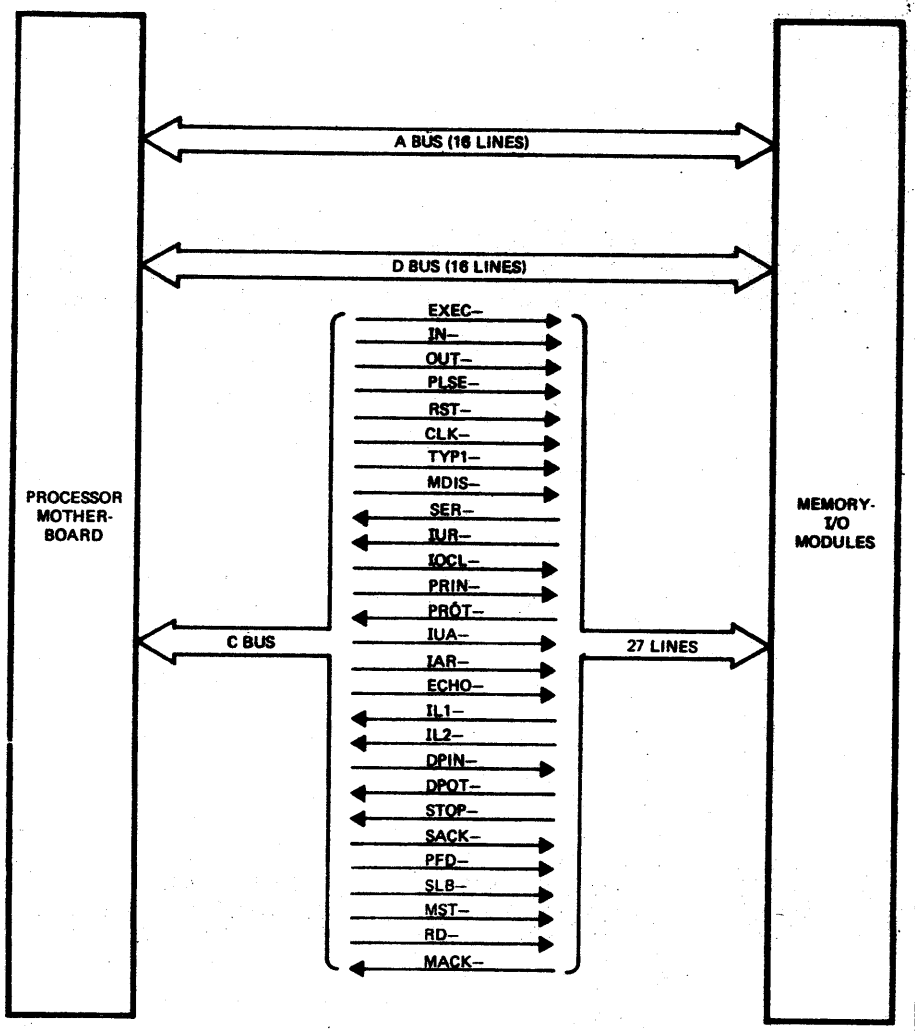


Figure 8-2. Maxi-Bus Components

I/O controllers use the D bus to convey an interrupt address to the Processor during interrupt processing.

8.2.3 Control Bus (C)

The C bus consists of 27 unidirectional control lines which define the specific action that an interface device is to perform. Nineteen lines are outputs from the Processor to Memory and I/O controllers while eight lines are inputs from either Memory or I/O controllers to the Processor. The 27 C bus lines are subdivided into four broad categories: I/O command, utility signals, interrupt signals, and DMA signals. Except as noted below, all Processor generated or received signals may also be generated or received by DMA controllers during DMA operations.

8.2.3.1 I/O Commands

There are three signals in this category: EXEC-, IN- and OUT-. These signals define the type of I/O operation in process.

EXEC- Execute. Processor generated signal that indicates the current instruction is a Select or Select and Present instruction. EXEC- is used typically to set or reset controls in the addressed I/O controller.

IN- Input. Processor generated signal that indicates the current instruction is an Input instruction and that the addressed I/O controller should place input data on the D bus.

OUT- Output. Processor generated signal that indicates the current instruction is an Output instruction and that the Processor has placed output data on the D bus for the addressed I/O controller to accept.

8.2.3.2 Utility Signals

There are five signals in this category: PLSE-, RST-, CLK-, TYP1-, and SER-.

PLSE- Pulse. Processor generated signal which is used as a strobe pulse to load registers during an output transfer, set or reset controls during a Select instruction, reset data transfer controls during an input transfer, and to reset Interrupt Stimulus Store controls upon recognition of an interrupt.

RST- System Reset. Processor or Console generated signal which is used to reset all controls in ALL interfaces to a known starting configuraton. RST- is generated by the Processor in response to a power failure condition, an Autoload initiation sequence, or when the Console RESET switch is depressed. Note - not driven by DMA controllers.

CLK- Clock. Processor generated, 1 megahertz, free-running square wave signal that may be used as a timing reference by I/O controllers. It is not synchronized to Processor operations. Note that only the Processor generates this signal. DMA controllers may not generate this signal.

TYP1- Type 1 Processor Installed. This signal is ground-true when the LSI-1 Processor is installed and open when the LSI-2 Processor is installed. This signal permits DMA controllers to determine which Processor is installed and perform hog mode transfers if necessary. The TYP1- signal is strung through the "200" side of the motherboard only (see paragraph 6-4).

SER- Sense Response. Signal generated by addressed I/O controller which, when true, indicates a true response to an interrogation by the Processor of some function associated with the controller or device it controls. Interrogation is made when a Sense or Conditional I/O instruction is issued.

MDIS- Memory Disable. Processor generated signal which is active during power up and power down sequences to assure no spurious memory cycles will occur during power transitions.

8.2.3.3 Interrupt Signals

There are nine signals associated with interrupt generation and processing. These signals are: IUR-, IOCL-, PRIN-, PROT-, IUA-, IAR-, ECHO-, IL1-, and IL2.

IUR- Interrupt Request. Multiplexed interrupt request line which multiple I/O controllers use to request interrupt service. Interrupts requested via this line are recognized on a priority basis. If two or more interfaces request interrupt service at the same time, recognition is given to the highest priority interface via the priority string (PRIN- and PROT-).

IOCL- I/O Clock. Processor generated signal which is used by I/O controllers to synchronize IUR interrupt requests into the Processor. IOCL has a minimum duration of 150 ns; however, the duration varies with internal Processor operation. When an interrupt is recognized by the Processor, IOCL is inhibited to prevent the generation of additional IUR interrupt requests. IOCL remains inhibited until the Processor completes execution of the interrupt instruction. DMA controllers may not generate this signal.

PRIN- and PROT- Priority In and Priority Out. PRIN- and PROT- form an interrupt priority chain which is strung serially through all I/O controllers and memory modules. PRIN- is the name given to the priority chain where it enters an interface. If low, it allows the interface to generate interrupts. Each interface generates a PROT- signal to indicate that neither it nor other upstream devices are generating an interrupt. The PROT- signal from each I/O controller is the PRIN- signal for the next downstream controller.

IUA- Interrupt Acknowledge. Processor generated signal which goes true upon recognition of any interrupt and remains true during execution of the interrupt instruction. DMA controllers may not generate this signal.

IAR- Interrupt Address Request. Processor generated signal which is used to request an interrupt address from an I/O controller in response to an interrupt request. DMA controllers may not generate this signal.

ECHO- Echo. Signal generated by the Processor when an Auto I/O instruction has transferred all data, or by an IMS instruction when the count overflows. ECHO- is typically used by the I/O controller to request an interrupt. This interrupt vectors to a user-determined location in Memory which normally contains a JST instruction to a subroutine. The subroutine performs the necessary housekeeping associated with an End-of-Block or elapsed count operation. DMA controllers may not generate this signal.

IL1- and IL2- Interrupt Lines 1 and 2. I/O controller generated high priority interrupt request lines which interrupt to locations :0002 and :0006, respectively. They are higher priority than the IUR line. IL1 has priority over IL2. IL1 and IL2 do not require interrupt vectoring by the interface as does IUR.

8.2.3.4 DMA Signals

Nine signals are associated with DMA control and processing. These signals are: DPIN-, DPOT-, STOP-, SACK-, PFD-, SLB-, MST-, RD- and MACK-.

DPIN- and DPOT- DMA Priority In and DMA Priority Out. DPIN and DPOT form a DMA priority chain which is strung serially through all DMA controllers and memory modules. DPIN- is the name given to the priority chain where it enters a DMA controller. If low, it allows the controller to access Memory. Each controller generates a DPOT- signal to indicate that neither it nor other upstream controllers are communicating with Memory. The DPOT- signal from each controller is the DPIN- signal for the next downstream controller. The DPIN- and DPOT- signals are strung through the "200" side of the motherboard only (see paragraph 8.7).

STOP- Stop Processor. DMA controller generated signal which stops the Processor upon completion of its current machine cycle to permit the DMA controller to gain control of the I/O bus. STOP- may be generated at any time and may remain active for any length of time.

SACK- Stop Acknowledge. Processor generated signal which informs DMA controllers that the Processor has relinquished control of the I/O bus to the DMA controllers. SACK- will remain true until STOP- is removed.

PFD- Power Failure Detected. Power supply generated signal which, when active, forces any DMA operations to terminate in order to permit the Processor to shut down the system in an orderly manner.

SLB- Select Least Significant Byte. Processor or DMA controller generated signal which is used for Byte Mode memory accesses. When SLB- is low, the least significant byte (bits 0 through 7) of the addressed memory word is accessed. When SLB- is high, the most significant byte (bits 8 through 15) of the addressed memory word is accessed. SLB- is used to disable Memory during Autoload operations by forcing it low while AB15- is high (Word mode).

MST- Memory Start. Processor or DMA controller generated signal which is used to initiate a memory cycle.

RD- Read Mode. Processor or DMA controller generated signal which, when low, indicates the current memory cycle is a Read/Restore cycle. When high, RD- indicates that the current memory cycle is a Clear/Write cycle.

MACK- Memory Acknowledge. Memory generated signal that is used to inform the Processor or DMA controller that data is available on the Data bus during a Read operation, or that data has been accepted during a Write operation.

8.3 I/O TRANSFER TIMING

I/O transfer timing is the period during an I/O instruction when data is transferred between the Processor and an I/O controller. (Refer to figure 8-3.)

NOTE

Unless otherwise noted, all timing intervals indicated in timing diagrams are given in nanoseconds. All timing intervals discussed in text are nominal.

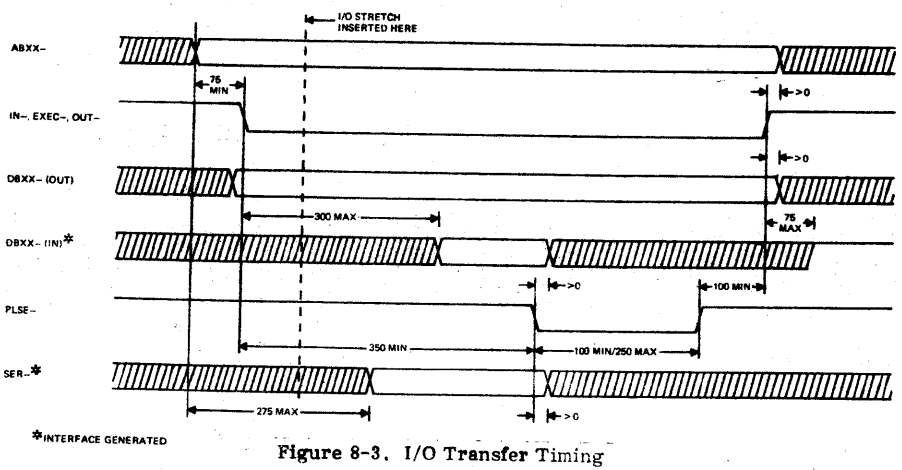


Figure 8-3. I/O Transfer Timing

8.3.1 I/O Bus Considerations

The A bus is active for non-I/O as well as I/O instructions. To guard against responding to a non-I/O instruction, the I/O control signals (EXEC-, IN-, or OUT-) should be used when interpreting the A bus. The SER- signal is the only exception and may be driven independent of EXEC-, IN-, or OUT-.

Data should never be placed on the D bus by an I/O controller except in the presence of IN- or IAR-.

8.3.2 Sense Instruction Timing

No Maxi-Bus control signals are generated by the Processor during a Sense instruction. The addressed I/O controller uses the function code information to determine which one of eight possible functions are to be tested. The sense information is sent to the Processor via the SER- line. If the Processor is looking for a Sense response, the SER- signal is gated into the Processor. Otherwise it is ignored. The user has 275 ns to stabilize the Sense response after receipt of the Device Address signals.

8.3.3 Select Instruction Timing

During Select or Select and Present instructions, the EXEC- signal is generated a minimum of 75 ns after the A bus stabilizes. The D bus is selected for output as a result of EXEC- and becomes stable a maximum of 150 ns after the leading edge of EXEC-. If a command register is used, the information on the D bus can be presented to the register by EXEC- and clocked in with PLSE-. The D bus contains all zeros during the SEL instruction and is equal to the contents of the Processor A or X register during the SEA or SEX instructions, respectively.

The PLSE- signal is developed a minimum of 350 ns after EXEC-. PLSE- is generally used to clock all control flip-flops in the I/O controller. Either the leading or trailing edge of PLSE- may be used to set or reset control flip-flops.

8.3.4 Input Timing

All input sequences, regardless of the Input instruction type, appear basically the same to an I/O controller. For all Input instructions, the IN- signal is generated a minimum of 75 ns after the A bus stabilizes. The D bus is selected for input as a result of IN-. The IN- signal is used by the controller to gate data onto the D bus. Data must be present and stable on the D bus no later than 300 ns after IN- goes low.

The PLSE- signal is developed a minimum of 350 ns after IN- goes low. PLSE- is typically used to reset the buffer ready control in the I/O controller. Either the leading or trailing edge of PLSE- may be used to reset the buffer ready control. Note, however, that data on the D bus must remain stable until the leading edge of PLSE- and must be removed no later than 75 ns after the trailing edge of IN-.

If the Input instruction issued is conditional, the Sense response (SER-) must be stable no later than 275 ns after the A bus stabilizes to guarantee detection of SER- by the Processor. If SER- is high from the 275 ns point to the leading edge of PLSE-, the entire input sequence is repeated for a Conditional Input or Block Input, without issuing PLSE, until the SER-line goes low. If SER- is low at the 275 ns point, the operation is terminated after the present cycle and PLSE- is generated to indicate the Processor has accepted the data. If SER- changes state between the 275 ns point and the leading edge of PLSE-, the Processor may or may not detect SER-.

All Sense responses are ignored by the Processor when executing Unconditional Input instructions.

8.3.5 Output Timing

All Output instruction sequences, regardless of the Output instruction type, appear basically the same to an I/O controller. During an Output instruction, the OUT- signal is generated a minimum of 75 ns after the A bus stabilizes. The D bus is selected for output as a result of OUT-. Once selected, the D bus stabilizes in a maximum of 150 nanoseconds after leading edge of OUT-.

The PLSE- signal is generated a minimum of 350 ns after OUT- goes low. PLSE- serves two functions. The first is to clock output data into a receiving register of the I/O controller. The second function is to reset the Output buffer empty control in the I/O controller.

If the Output instruction is conditional, the Sense response must be stable no later than 275 ns after the A bus stabilizes to guarantee detection of SER- by the Processor. If SER- is high from the 275 ns point to the leading edge of PLSE-, the entire output sequence is repeated, without issuing PLSE, until the SER- line goes low. If SER- is low at the 275 ns point, the operation is terminated after the present cycle and PLSE- is generated to indicate the availability of data to the controller. If SER- changes state between the 275 ns point and the leading edge of PLSE-, the Processor may or may not detect SER-.

Any Sense responses that are generated during an Unconditional Output instruction are ignored by the Processor.

8.3.6 Automatic Input and Output Timing

The Automatic Input and Output instructions have essentially the same transfer timing as all other I/O instructions. The only difference is that when used as interrupt instructions, Auto I/O instructions develop an ECHO- signal to the controller when the last word of byte of data has been transferred. The ECHO- signal occurs a minimum of 350 ns after IN- or OUT- during the last transfer. ECHO- is typically used by the interface to develop an End-of-Block interrupt. These instructions are unconditional and do not require a Sense response.

8.3.7 I/O Instruction List

For the convenience of the user, table 8-1 provides a list of the Processor I/O instructions. The instructions are grouped into four major categories (Sense, Select, Input and Output). The Input and Output categories are further divided into Unconditional, Automatic, Conditional and Block sub-categories. The Conditional and Block sub-categories require a Sense response while the Unconditional and Automatic sub-categories do not.

Table 8-1. I/O Instruction List

FUNCTION	MNEMONIC	MACHINE CODE (HEX)
Sense	SEN	49XX
	SSN	48XX
Select	SEL	46XX
	SEA	44XX
	SEX	46XX
Unconditional Input	INA	58XX
	INX	5AXX
	IBA	78XX
	IBX	7AXX
	INAM	5CXX
	INXM	5EXX
	IBAM	7CXX
	IBXM	7EXX
Automatic Input	AIN	50XX
	AIB	54XX
Conditional Input	RDA	59XX
	RDX	5BXX
	RBA	79XX
	RBX	7BXX
	RDAM	5DXX
	RDXM	5FXX
	RBAM	7DXX
	RDXM	7FXX
Block Input	BIN	71XX
Unconditional Output	OTA	6CXX
	OTX	6EXX
	OTZ	68XX
Automatic Output	AOT	60XX
	AOB	64XX
Conditional Output	WRA	6DXX
	WRX	6FXX
	WRZ	69XX
Block Output	BOT	75XX

XX = device address and function code

8.4 INTERRUPT CHARACTERISTICS

Minicomputers perform in a wide variety of applications where they communicate with many different types of devices. These devices operate at widely varying speeds and generate events that occur randomly rather than at evenly spaced time intervals. If the events do occur at evenly spaced time intervals, these intervals may be relatively far apart. For these reasons, a versatile and efficient computer needs a priority interrupt system.

If a computer does not have a priority interrupt system, the computer must poll all of the external devices which may require service. The polling must be at frequent enough intervals so that events are serviced within a reasonable time after they occur. Polling consumes considerable time, and may not allow much processing time between the handling of external events.

A priority interrupt system relieves the computer of the polling responsibility. The computer may continue processing data between external events, and may take time out from main program processing to handle external events as they occur.

The ALPHA LSI computers feature five levels of interrupts. Each interrupt level uses an interrupt request line to obtain attention from the Processor. Upon obtaining this attention, the source of the interrupt vectors the Processor to an interrupt location in Memory. The interrupt location contains an interrupt instruction which defines the specific action that the Processor is to take in processing the interrupt.

The five interrupt request lines are designated Power Fail Interrupt (PFI), Console/TRAP Interrupt (CINT), Interrupt Line 1 (IL1), Interrupt Line 2 (IL2), and Interrupt Request (IUR). A priority level exists between each of these lines wherein PFI has the highest priority, CINT is second, IL1 is third, IL2 is fourth and IUR is lowest in priority. PFI, CINT, IL1 and IL2 are self-vectoring lines (the user does not have to supply the interrupt address). The IUR line is shared by multiple devices and features a priority chain to resolve priority when two or more devices issue an IUR interrupt request at the same time. Each of the multiple interrupt sources that share the IUR line cause the Processor to be vectored to distinct locations that can be anywhere in Memory.

8.4.1 Interrupt Lines

The characteristics of each of the five interrupt request lines are discussed in the following paragraphs.

8.4.1.1 Power Fail Interrupt

The PFI line services the power down interrupt only. PFI is the highest priority interrupt line in the interrupt system and is not accessible to the user via the Processor Maxi-Bus.

8.4.1.2 Console (TRAP) Interrupt

The CINT line services the Console and Trap interrupts only. CINT is the second highest priority interrupt line and is not accessible to the user via the Processor Maxi-Bus.

8.4.1.3 Interrupt Line 1

IL1 vectors all interrupts to memory location : 0002. IL1 does not provide external priority resolution when servicing multiple devices. IL1 is the third highest priority interrupt line and is accessible to the user via the Processor Maxi-Bus.

8.4.1.4 Interrupt Line 2

IL2 vectors all interrupts to memory location : 0006. IL2 is the fourth highest priority interrupt line and is accessible to the user via the Processor Maxi-Bus. Like IL1, IL2 does not provide external priority resolution to service multiple devices.

8.4.1.5 Interrupt Request

The IUR line vectors interrupts to the Processor from a virtually unlimited number of devices. The IUR line has a priority string associated with it. The priority string ensures that a device with a higher priority will be serviced before a lower priority device when two or more IUR requests occur at the same time. When the interrupting device has priority, it must furnish an interrupt address to the Processor upon request. In general, IUR interrupt addresses are user defined. There is a recommended list of addresses, however (refer to appendix B).

8.4.2 Processor Generated Interrupts

The ALPHA LSI computer generates two standard and six optional interrupts. In addition, two optional pseudo interrupts are generated. Each of these interrupts are discussed briefly in the following paragraphs in order of priority.

8.4.2.1 Power Fail/Restart Interrupt (Optional)

The Power Fail/Restart (PF/R) option generates a power down interrupt to location :001C whenever a low power condition exists. The power down interrupt has the highest priority of any interrupt serviced by the Processor. When power is restored to an acceptable level, the PF/R logic causes the P register to be set to location :0000 and the RUN mode is established to restart the system. Although location :0000 is the power up location, it is not a true interrupt location, but rather a pseudo interrupt location since no interrupt processing is required to get to location :0000.

8.4.2.2 Autoload (Optional)

The Autoload option utilizes the PF/R logic to develop a pseudo interrupt to location :0000 of a special Autoload read-only-memory as a starting point for the Autoload sequence.

8.4.2.3 Console Interrupt and Trap (Standard)

A Console interrupt can be developed when the Processor is in the RUN mode and the INT switch on the Console is depressed. A Trap interrupt is developed when the TRP instruction is executed. Both the Console and Trap interrupts share the second highest interrupt priority and they both interrupt to location :001E.

8.4.2.4 Real Time Clock (Optional)

The Real Time Clock (RTC) option generates a clock and sync interrupt. The Clock and Sync interrupts share the first highest priority on the IUR line. The Clock interrupt is vectored to location :0018 while the Sync interrupt is vectored to location :001A.

8.4.2.5 Teletype/CRT/Modem Controller (Optional)

The processor mounted TTY/CRT/Modem controller generates both Word and End-of-Block (EOB) interrupts via the IUR line. The Word interrupt is vectored to location :0002 while the EOB interrupt is vectored to location :0006. These interrupt vectors are the same interrupt vectors that are used by the IL1 and IL2 lines. Since IL1 and IL2 do not provide priority resolution and are of a higher priority than these interrupts, the TTY Word and EOB interrupts should be displaced to alternate locations when IL1 and IL2 are used. A jumper option permits the Word and EOB interrupts to be displaced to locations :0022 and :0026, respectively. When used in the Full Duplex mode, the TTY controller generates four interrupts (locatons :0002, :0006, :0022, and :0026). The TTY interrupts share the second highest priority on the IUR line.

8.4.3 Offsetting Processor Generated Interrupts

Figure 8-4 lists, in the order of their absolute priority, the standard interrupt locations for all Processor generated interrupts. These interrupt locations are all located in the

ABSOLUTE PRIORITY		INTERRUPT ADDRESS
	1 POWER FAIL (PFI)	:001C (:011C)
	2 TRAP INTERRUPT (CINT)	:001E (:011E)
	3 CONSOLE INTERRUPT (CINT)	:001E (:011E)
	4 INTERRUPT LINE 1 (IL1)	:0002 (:0102)
	5 INTERRUPT LINE 2 (IL2)	:0006 (:0106)
IUR CHAIN	6 RTC SYNC INTERRUPT (IUR)	:001A (:011A)
	7 CLOCK INTERRUPT (IUR)	:0018 (:0118)
	8 TTY END-OF-BLOCK (IUR)	:0006 (:0106); OPTIONAL :0026 (:0126)
	9 TTY WORD (IUR)	:0002 (:0102), OPTIONAL :0022 (:0122)
	10 SLOT B200	
	11 SLOT B100	
	12 SLOT C100	
	13 SLOT C200	
	14 SLOT D200	
	15 SLOT D100	
	16 SLOT E100	
	17 SLOT E200	
	18 EXPANSION CHASSIS SLOT A100	
	19 EXPANSION CHASSIS SLOT A200	
	20 EXPANSION CHASSIS SLOT B200	

Slots B through E accommodate plug-in modules (either memory or I/O). All I/O modules may use the IUR line and must provide an interrupt address. Modules with multiple interrupt capabilities must have internal priority resolution and multiple addresses. The continuity of the priority chain must not be broken. If broken, interrupts below the break may not be recognized or may be recognized erroneously.

Figure 8-4. ALPHA LSI Interrupt Organization

scratchpad area of Memory. A jumper option permits the user to offset these locations by :100 locations to place them outside the scratchpad area. This allows for more efficient utilization of the scratch area. IUR interrupts generated by non-processor mounted options may be individually offset to place them outside the scratch area.

NOTE

The power up restart and autoload start up location (location :0000) is not affected by the offset jumper option.

8.4.4 Peripheral Generated Interrupts

Peripheral interface controllers may request interrupt service via the IL1-, IL2- or IUR- request lines. The techniques used to develop these interrupt requests are discussed in detail in section 9 of this manual.

8.4.5 Interrupt Transfer Timing (Figure 8-5)

For the purpose of priority resolution, all interrupts must be synchronized prior to being generated. Synchronization can occur only during a mainline program instruction. This is to ensure that when executing the interrupt instruction, no other interrupt can intervene. When synchronization is obtained, the PROT- signal from the interrupting device goes high (false) to disable all down-stream IUR interrupts. When interupts of higher priority than IUR are serviced, the Processor makes the PROT- signal high to disable all IUR interrupts.

If interrupts are enabled, the Processor recognizes an interrupt request when the current mainline program instruction has finished execution. When recognition of an interrupt is given, the Interrupt Acknowledge signal (IUA) is issued by the Processor and IOCL is turned off to inhibit any change in interrupt request status until the current interrupt operation is complete.

Approximately 2 μs after IUA- goes low, the Processor generates the Interrupt Address Request signal (IAR-) and selects the D bus for input. IAR- is used by the interrupting controller to generate the interrupt address. The IAR- signal is low for approximately 750 ns. During this interval, the user generated interrupt address must be available within 300 ns of IAR- and remain stable until the leading edge of PLSE-. PLSE- is used in the more complex interrupt structures to reset the Interrupt Stimulus Store control.

IUA- will remain low until the interrupt instruction completes execution. The duration (IUA low) is a function of the number of machine cycles that are required to execute the interrupt instruction. When IUA- goes high, IOCL is re-enabled permitting subsequent interrupts to be generated.

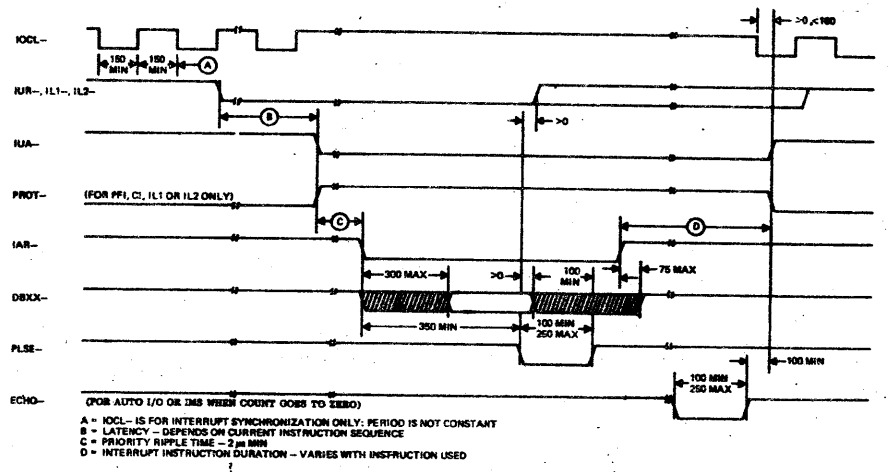


Figure 8-5. Interrupt Transfer Timing

8.4.6 Interrupt Operation Control

Two levels of control are associated with IL1, IL2, and IUR interrupt processing--primary and secondary.

The primary control level is provided by the Enable Interrupt flip-flop (EIN) in the Processor. The EIN flip-flop is accessible to the programmer and can be enabled or disabled on command. When enabled, EIN allows recognition of any interrupt. Likewise, when EIN is disabled, interrupts will not be recognized.

The secondary control level is provided by an interrupt enable flip-flop in each I/O controller. The interrupt enable flip-flop enables or disables the interrupt structure of the I/O controller. Like the EIN flip-flop discussed above, the interrupt enable flip-flop in each controller can be enabled or disabled by means of a Select instruction addressed to the specific I/O controller with the appropriate function code.

This dual system of interrupt control can be very useful to a programmer. With this system, the programmer can control interrupts in general with the EIN flip-flop, yet enable or disable interrupts from selected devices as conditions dictate.

Interrupts developed via the PF and CINT lines are somewhat different in that they can be generated outside EIN control. In normal operation (that is, when operating under EIN control), the Power Fail, Console and Trap interrupts require that EIN be enabled. Most interrupt subroutines disable interrupts during execution of the subroutines causing high priority interrupts such as Power Fail to wait until EIN is re-enabled. A special jumper on the option board permits all interrupts generated on the PF and CINT lines to be recognized regardless of the state of EIN.

When the jumper option is employed, two new instructions (PFE and PFD) are used to control the Power Fail circuits. The PFE instruction must have been issued before a Power Fail interrupt can be generated. Likewise, the PFD instruction disables the generation of a Power Fail interrupt.

The Console interrupt is controlled by the CIE and CID instructions in the same way as in normal operation. The Trap interrupt is generated in the same manner as in normal operation. The only difference between normal operation and the jumper option is that EIN does not have to be set to generate the Console and Trap interrupts.

Another useful programming feature is the SIN instruction. The SIN instruction permits the programmer to suppress recognition of all interrupts (and Byte mode operation) for up to six instructions.

Once an interrupt structure is enabled, an interrupt can be generated in five basic steps:

Step 1 Stimulus Generation--The user generates the interrupt stimulus in response to some event or condition.

Step 2 Interrupt Request Generation--The interrupt structure of the I/O controller, if enabled, stores the interrupt stimulus and generates an interrupt request.

Step 3 Interrupt Recognition--The Processor upon receipt of the interrupt request waits for the current instruction to complete execution, and if system interrupts are enabled (EIN set), issues an interrupt address request.

Step 4 Interrupt I/O Address Generation--The interrupt structure of the I/O controller responds to the interrupt address request by placing the interrupt address on the D bus lines (except for IL1 and IL2 interrupt).

Step 5 Interrupt Instruction Execution--The Processor fetches and executes the instruction from the interrupt location.

8.4.7 Interrupt Request Line Trade Offs

The user has a choice of three interrupt request lines, IL1, IL2 and IUR. The trade offs associated with each of these lines are discussed below.

The IL1 and IL2 interrupt structures are the simplest structures to implement in terms of hardware since they do not require interrupt address logic, Processor synchronization logic, or down-stream priority disable logic. All of these functions are provided in the Processor. The IL1 and IL2 lines are intended for single device applications where high speed devices require the highest available priority to minimize interrupt latency.

The IUR line is for multiple devices where each device competes for service via the priority chain. The priority of an I/O controller can be changed by simply removing the controller from the computer chassis and relocating it in a higher or lower priority card slot. An IUR generating controller has greater flexibility in terms of address vectoring. If an address vector must be changed, the address may be offset from its base location ot another location by means of address select lines.

8.5 DMA OPERATIONS

The ALPHA LSI computer has a direct memory access (DMA) port which permits specially built controllers (referred to as DMA controllers) to transfer data via the Maxi-Bus at very high speed to or from Memory or other controllers.

8.5.1 General Characteristics

8.5.1.1 Processor Provisions

The ALPHA LSI Processor is designed to surrender the Maxi-Bus to a DMA controller whenever a Stop command (STOP-) is received. Upon receipt of the STOP- signal, the Processor completes the current microcycle, stops, and sends a Stop Acknowledge (SACK-) signal to the requesting DMA controller(s). A DMA controller may hold STOP- active for as long as necessary to complete requested data transfers. But once the STOP- line is released, the Maxi-bus cannot be reacquired by the controller until SACK- goes high (see 8.5.2.1).

8.5.1.2 Memory Operations

DMA controllers may communicate directly with Memory. The DMA controller must emulate the Processor by generating a memory address and appropriate control signals. Memory operations may be either Read (data accessed from Memory) or Write (data written into Memory). Data cannot be read, modified and rewritten in one cycle. When communicating with a single memory module, data transfer rates of up to 625,000 words per second can be achieved with the standard 1.6 μs Memories. When more than one memory module is used in the computer, DMA transfer rates of up to twice the basic speed of the memory modules can be achieved by making alternate memory accesses in different modules. Memory interleaving straps allow even and odd addresses to be in separate memory modules so that sequential addressing automatically alternates between modules.

In addition to word transfer capabilities, byte transfers may be performed by a DMA controller. All byte packing and unpacking is done automatically by the memory modules with all byte data transferred on the lower eight D bus lines (the upper eight D bus lines are ignored during byte transfers).

All memory modules contain data and address registers to permit asynchronous operation. During a Write operation, the source device furnishes an address and data along with a memory start signal. As soon as the address and data is stored in its registers, the

memory issues an acknowledge signal and releases the bus even though it has not actually finished the Write operation. During a Read operation, the memory accesses the addressed location, places the data on the D bus, and then issues the memory acknowledge signal. When the source device recognizes the memory acknowledge signal, it removes the start signal releasing the Maxi-Bus. Any memory restore operation or overhead interval does not tie up the Maxi-Bus and therefore frees the Processor or DMA controller to perform another operation.

8.5.1.3 I/O Operations

A DMA controller may emulate the I/O instructions of the Processor. The DMA controller may issue Input, Output, Sense, Select, and Select and Present instructions. It may perform conditional and unconditional I/O. All I/O instructions and control lines of the Maxi-Bus that are used by the Processor for I/O operations are available to a DMA controller when the Processor is stopped.

8.5.1.4 Limitations

A DMA controller is not permitted to use the interrupt processing capabilities of the Processor. Interrupts are reserved for use by the Processor only. I/O controllers that are under control of a DMA controller must have their interrupt facilities disabled.

When multiple DMA controllers are employed in a system, they must compete for control of the Maxi-Bus on a priority basis. DMA Priority lines are strung serially through the 200 series connectors of the ALPHA LSI motherboard. Therefore, DMA controllers must be either full boards or half boards that are installed in the 200 series connectors of the ALPHA LSI motherboard.

When using the standard expansion chassis buffer board, a DMA controller must be in either the same chassis or in a chassis that is closer to the Processor than a memory module or I/O controller that it must communicate with. This is because the expansion buffer board treats unidirectional lines (such as the A bus lines) as originating from the Processor end of a chain of expansion chassis. Therefore, unidirectional signals which normally originate from the Processor cannot be transmitted to an up-stream memory module or I/O controller.

8.5.2 DMA Timing

The following paragraphs define DMA transfer timing. All timing intervals shown in timing diagrams are in nanoseconds and all timing intervals discussed in the text are nominal. Times determined by memory access and cycle intervals are shown for the standard 1.6μs memory modules and may be different for other memory modules.

8.5.2.1 Maxi-Bus Acquisition Timing (Figure 8-6)

Two signals are involved with Maxi-Bus acquisition: STOP- and SACK-. When a DMA controller is ready to make a transfer, it drives the STOP- line low (ground-true). The Processor, upon seeing STOP- low, immediately begins preparing to vacate the Maxi-Bus. After performing the required internal housekeeping associated with stopping, the Processor drives the SACK- signal low (ground-true). The time interval from the leading edge of STOP- to the leading edge of SACK- can be as much as 4800 ns for the LSI-1 Processor.

Once SACK- goes low, the DMA controller is free to commence the transfer operation. Typically, DMA controllers operate on a request basis wherein they make one transfer for each request received from an associated peripheral. If the DMA controller receives another request prior to completion of the current transfer (Burst mode), it will keep STOP- active. Otherwise it releases the STOP- line when the current operation is completed, as signaled by the trailing edge of the Memory Acknowledge (MACK-) signal.

After releasing the STOP- line, the DMA controller may not attempt to reacquire the Maxi-Bus before SACK- goes high. The LSI-1 Processor can take up to 2400 ns to raise SACK- and restart programmed operation. Once SACK- goes high, the DMA controller is forced to wait out the DMA acquisition period before acquiring the Maxi-Bus again. Therefore, the worst case latency period is 5600 ns for LSI-1 Processor. The LSI-2 Processor DMA latency is a function of the type of memory module used. The LSI-2 Processor DMA latency times are as follows:

Core 980 = 1405 ns
Core 1200 = 1825 ns
Core 1600 = 2575 ns
SC 1200 = 3025 ns

Latency time may be longer if a higher priority DMA controller is also requesting the Maxi-Bus.

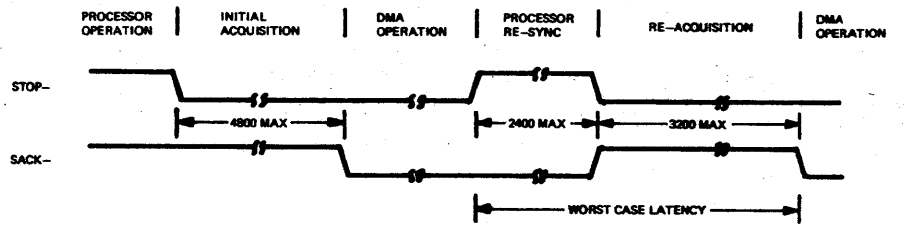


Figure 8-6. Maxi-Bus Acquisition Timing

8.5.2.2 Memory Transfer Timing (Figure 8-7)

Memory modules of various speeds, sizes and technologies may be intermixed in a system. The standard 4K core memory has a cycle time of 1600 ns which provides a maximum data transfer rate of 625,000 words/bytes per second.

A memory cycle is divided into an access interval and an overhead interval. The access interval is the period when data is transferred to or from Memory. The overhead interval is used for internal memory operations. For core memories, the overhead interval is used to restore the contents of the word just read, or to write the word just transferred. For non-destructive readout memories, the overhead interval consists primarily of logic recovery time. For dynamic MOS memories, the overhead also includes cycles stolen by Memory to refresh dynamic storage. During the overhead interval, the Maxi-Bus is available for other operations.

For DMA applications requiring data transfer rates in excess of 625,000 words/bytes per second, memory interleaving can be employed. When alternate memory cycles address different memory modules, each memory's overhead interval can be used to access another memory module, yielding transfer rates up to twice that possible with a single memory module. Each memory module features static control lines at the rear of the module which permit the memory module to operate in the interleaved mode. Each memory module can be configured to respond to either even or odd memory addresses. This feature allows sequentially addressed memory locations to automatically alternate between memory modules.

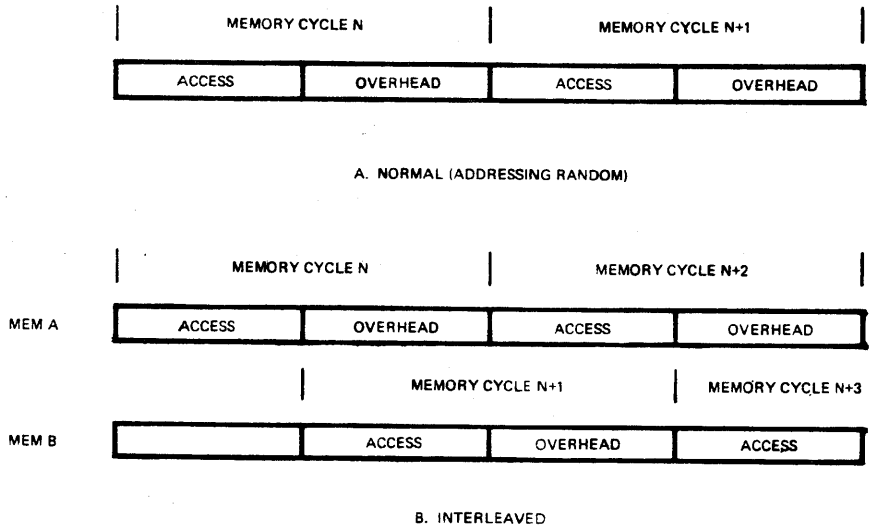


Figure 8-7. Memory Addressing Comparisons

8.5.2.2.1 DMA Read Access Timing (Figure 8-8). A DMA read access sequence is started by the DMA controller placing the desired memory address on the A bus. A minimum of 75 ns is required for A bus settling and address recognition for all memory modules before the DMA controller drives the Memory Start (MST-) signal low. The Read (RD-) signal must be driven low no later than 25 ns after MST- goes low.

The addressed memory module begins execution of a memory cycle when MST- goes low, and after it has finished any previous operation. When the addressed location has been accessed (approximately 450 ns for standard 1600 ns memories), the contents of the addressed memory location are placed on the D bus and the MACK- signal is issued. The information on the D bus will remain stable until MST- is removed.

Upon receipt of MACK-, the DMA controller is free to disengage the A bus. After allowing for settling time on the D bus, the DMA controller strobes the contents of the D bus into a receiving register and removes MST- and RD-. The memory module removes MACK- on the trailing edge of MST- and disengages the D bus on the trailing edge of MST- or RD-, whichever goes away first. The DMA controller must disengage the A bus prior to, or coincident with, removal of MST-. The DMA controller may not initiate another memory cycle until MACK- has been removed.

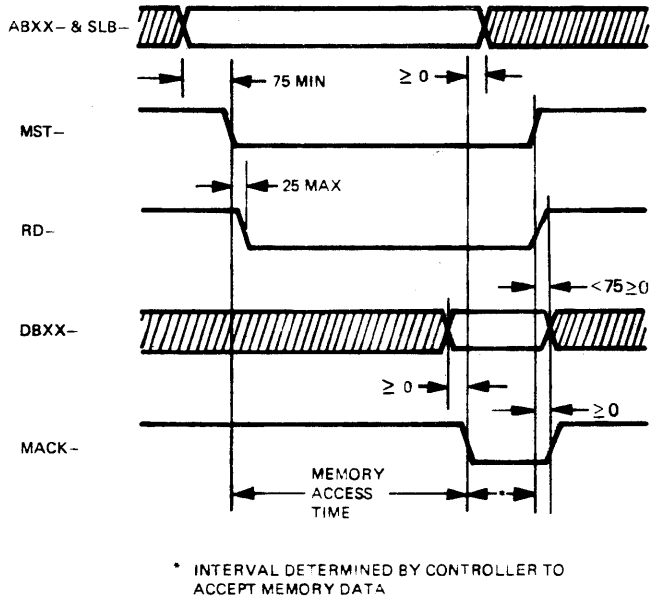


Figure 8-8. Read Access Timing

8.5.2.2.2 DMA Write Access Timing (Figure 8-9). A write access sequence is similar to a read access sequence except that the RD- signal is held high and the write data is presented to the addressed memory at the same time MST- is generated.

A write access is started by placing the memory address on the A bus. After a minimum of 75 ns the MST- signal is driven low. The RD- signal is held high and the write data is gated onto the D bus no later than 25 ns after MST- goes low. The memory module indicates acceptance of the write data by driving the MACK- signal low.

The DMA controller must disengage the A bus and the D bus and remove MST- when MACK- goes low. MACK- is removed on the trailing edge of MST- at the memory module.

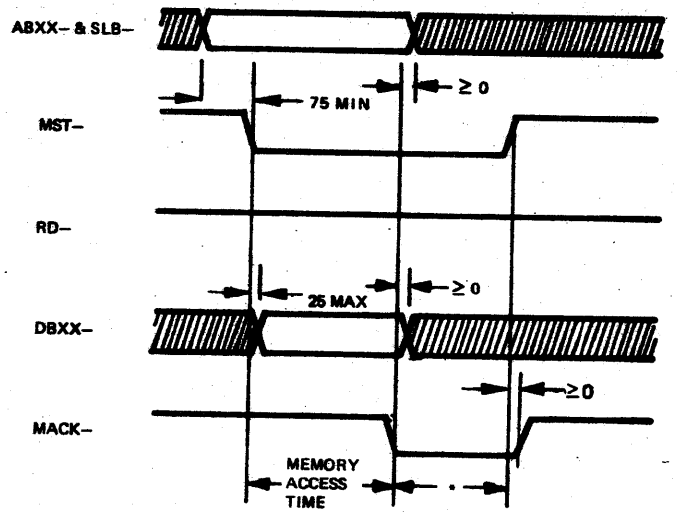


Figure 8-9. Write Access Timing

8.5.2.3 I/O Transfer Timing

A DMA controller may transfer data to or from another controller by emulating the Processor's operations on the I/O control signals. A single exception to standard I/O transfer sequencing involves generation of MACK- during I/O transfers under DMA control that do not involve the use of Memory. In this case, the DMA controller must generate

MACK- for a minimum of 100 ns prior to completion of the I/O transfer. This allows other DMA controllers in the system to synchronize any pending Maxi-Bus requests and properly auction DMA priority (see paragraph 9.5.2.2).

8.6 ELECTRICAL CHARACTERISTICS

The Maxi-Bus is best classified as a hybrid tri-state open-collector (wire-OR) bus, unterminated.

Most processor drivers are tri-state power elements, capable of sinking 32 mA at 0.4 Vdc maximum and sourcing 2.0 mA at 2.4 Vdc minimum. In a few isolated cases, open-collector TTL drivers (32 mA sink at 0.4 Vdc) are used.

Processor receivers present one standard TTL load to the line (-1.6 mA at 0.8 Vdc, 40 μA at 2.4 Vdc). Depending on the nature of the particular signal, pullup resistors to +5 Vdc are used.

Open-collector drivers in I/O and memory modules are permitted on those bus lines for which pullup resistors are provided. Minimum required drive capability is 32 mA at 0.4 Vdc max. Tri-state drivers electrically equivalent to the processor bus drivers are also allowed, as long as the logic design of the system guarantees that no two tri-state drivers connected to the same bus line are simultaneously enabled. Receivers on I/O and memory modules may be any standard 74 series TTL device. Only one such receiver per module is permitted. Maximum loading shall not exceed 1.6 mA per module.

Logic Levels (Negative-true)

logic "1": +0.4 Vdc max.
logic "0": +2.4 Vdc min.

Table 8-2 summarizes the Maxi-Bus driver, receiver and pullup circuits.

8.7 MOTHERBOARD ORGANIZATION

Any slot (other than the slot dedicated to the NAKED MINI LSI Processor) can accept either an I/O or memory module.

Figure 8-11 provides an illustration of the system motherboard. The motherboard provides for six slots used as follows:

Slot	Purpose
A	NAKED MINI LSI Processor
B	Universal (Memory or I/O)
C	Universal (Memory or I/O)
D	Universal (Memory or I/O)
E	Universal (Memory or I/O)
F	Power Supply

In any given slot, either a full board (15" x 16.5") or two half boards (each 7.5" x 16.5") may be installed. One slot contains two connectors. The connector on the right (rear-facing) is referred to as the 100 series connector and contains pins numbered 100 through 186; similarly, the connector on the left if referred to as the 200-series connector and contains pins numbered 200 through 286.

With the exception of the priority chains, memory bank control, and two special processor power supply signals, all signals are wired in a U fashion through all half board connectors. All exceptions are described below (shown in figure 2-3).

8.7.1 Interrupt Priority

The daisy chained interrupt priority string (PRIN-, PROT-) is wired in S fashion beginning at the 100-series connector of slot A, across to the 200-series connector, then in reverse direction across the two B slot connectors, etc., until all slots are connected. Both ends of the chain are connected to the expansion connectors. Both PRIN- and PROT- on processor connector A100 are used to carry special signals to the Console; the actual origin of the priority chain is slot A200.

8.7.2 Memory Bank Control, DMA Priority

The Memory Bank control (MBIN, MBOT), DMA priority (DPIN-, DPOT-) and TYP1- lines daisy chain down the 200-series connectors only. Therefore, memory modules and DMA controllers must be either full boards or half boards installed on the 200 series side only.

8.7.3 Processor Power Supply Signals

Two lines from the power supply, TTLF (Twice the Line Frequency) and +5 H (Hang Power) are wired directly between the power supply slot and processor slot A100.

8.8 EXPANSION AND CONSOLE INTERCONNECT

To facilitate expansion of the computer system beyond the first chassis and to provide for interconnect to the ALPHA LSI Console, connectors are supplied on the motherboard immediately above slot A. Two connectors, J2 and J3, are provided for Maxi-Bus expansion, and one connector, J1, is provided to interconnect the Console. Figure 8-10 shows the pin assignments for connectors J2 and J3, and figure 10-7 in section 10 shows the pin assignments for J1.

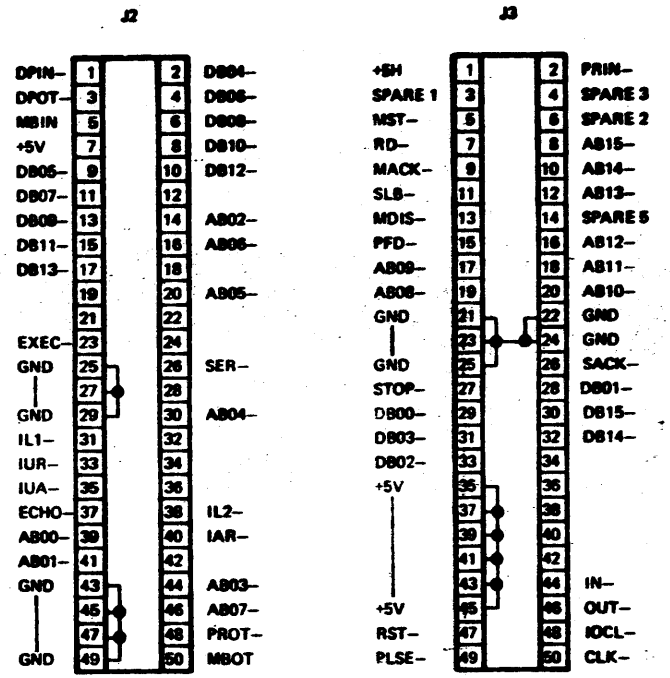


Figure 8-10. Maxi-Bus Expansion Connector, Pin assignments

8.9 NAKED MINI LSI MAXI-BUS REQUIREMENTS

In applications where the NAKED MINI LSI computer is used without the system motherboard and is instead connected to I/O and/or memory modules via user-supplied cabling, printed circuit board, etc., the line length of each signal must be limited to 18 inches.

The user designed Maxi-Bus interface cabling must be designed to minimize crosstalk, reflections, etc., so as to preserve signal integrity. Recommendations as to line termination are available upon request. In general, consultation with Computer Automation is recommended to ensure system performance.

8-10 TWO-MODULE OPTIONS

Any option requiring more than one PC board may not use the motherboard for interconnection. Unique interconnections may be made via a jumper cable installed on the rear-edge of the two boards.

Table 8-2. Maxi-Bus Load, Drive and Termination Summary

			DEVICE TYPE(S) (REFER TO NOTE 1)						
	SIGNAL	PIN	CPU	MEMORY	I/O CONT	DMA CONT	CONSOLE	BUFFER	OPT.BD
	GND	1							
	GND	2							
	+12V	3							
	+12V	4							
	+12V	5							
	+12V	6							
	-12V	7							
	-12V	8							
NOTE 2	DPIN-	9		J]	J]	5			
NOTE 2	DPOT-	10				4		*	
NOTE 4	EBSEL-	11							
NOTE 3		12							
	+5V	13							
	+5V	14							
	MST-	15	1,6	5		1	5	5	5
	AL-	16							
	MACK-	17	5,6	1		5		1	2
	RD-	18	2,6	5		2		5	
NOTE 3	TYP1-	19	OPNorGRD					2	
	SLB-	20	2,6	5		1		5	2
	PFD-	21	5,6			5		*	
	MDIS-	22	3	5				5	
	AB08-	23	1	5		1		5	
	AB09-	24	1	5		1		5	
	AB10-	25	1	5		1		5	
	AB11-	26	1	5		1		5	5
	GND	27							
	GND	28							
	AB12-	29	1	5		1		5	5
	AB13-	30	1	5		1		5	5
	AB14-	31	1	5		1		5	5
	AB15-	32	1	5		1		5	
NOTE 3	DB16-	33				5,6			
NOTE 3	DB17-	34							
	STOP-	35	5,6			2		2	
	SACK-	36	3			5		5	
NOTE 2	MBIN	37		5	J]	J]			
NOTE 2	MBOT	38	4	4				*	
	DB00-	39	1,5,6	1,5	2,5	1,5	2,5	1,5	1,2,5
	DB01-	40	1,5,6	1,5	2,5	1,5	2,5	1,5	1,2,5
	DB02-	41	1,5,6	1,5	2,5	1,5	2,5	1,5	1,2,5
	DB03-	42	1,5,6	1,5	2,5	1,5	2,5	1,5	1,2,5
	+5V	43							

NOTES: 1. DEVICE TYPES ARE AS FOLLOWS-
1) TRI-STATE DRIVER, 32ma (8835 or EQUIV.)
2) 32 MA OPEN-COLLECTOR DRIVER (7438 or EQUIV.)
3) 32 MA TTL DRIVER (7437 OR EQUIV.)
4) 16 MA TTL DRIVER (7400 OR EQUIV.)
5) TTL RECEIVER (7404 OR EQUIV.)
6) PULL-UP RESISTOR (1 KOHM)
J) JUMPER
*) STRAIGHT THRU SIGNAL (NO DEVICES IN SIGNAL PATH)

Table 8-2. Maxi-Bus Load, Drive and Termination Summary (Cont'd)

		DEVICE TYPE(S) (REFER TO NOTE 1)						
SIGNAL	PIN	CPU	MEMORY	I/O CONT	DMA CONT	CONSOLE	BUFFER	OPT. BD
+5V	44							
DB04-	45	1,5,6	1,5	2,5	1,5	2,5	1,5	1,5
DB05-	46	1,5,6	1,5	2,5	1,5	2,5	1,5	1,5
DB06-	47	1,5,6	1,5	2,5	1,5	2,5	1,5	1,5
DB07-	48	1,5,6	1,5	2,5	1,5	2,5	1,5	1,5
DB08-	49	1,5,6	1,5	2,5	1,5	2,5	1,5	1
DB09	50	1,5,6	1,5	2,5	1,5	2,5	1,5	1
DB10-	51	1,5,6	1,5	2,5	1,5	2,5	1,5	1
DB11-	52	1,5,6	1,5	2,5	1,5	2,5	1,5	1
DB12-	53	1,5,6	1,5	2,5	1,5	2,5	1,5	1
DB13-	54	1,5,6	1,5	2,5	1,5	2,5	1,5	1
DB14-	55	1,5,6	1,5	2,5	1,5	2,5	1,5	1
DB15-	56	1,5,6	1,5	2,5	1,5	2,5	1,5	1
EXEC-	57	1,6		5	5	5	5	5
IN-	58	1,6		5	5	5	5	5
GND	59							
GND	60							
IOCL-	61	1,6		5	5	5	5	5
OUT-	62	1,6		5	5	5	5	5
CLK-	63	3		5	5	5	5	
SER-	64	5,6		2	2		2	2
IUR-	65	5,6		2	2		2	2
IL1-	66	5,6		2	2		2	
IAR-	67	1,6		5	5	5	5	5
IL2-	68	5,6		2	2		2	
RST-	69	2,5,6		5	5	2,5	2	5
IUA-	70	1,6		5	5		5	
PLSE-	71	1,6		5	5	5	5	5
ECHO-	72	1,6		5	5		5	5
+5V	73							
+5V	74							
AB03-	75	1,6	5	5	1,5	5	5	5
AB04-	76	1,6	5	5	1,5	5	5	5
AB05-	77	1,6	5	5	1,5	5	5	5
AB06-	78	1,6	5	5	1,5	5	5	5
AB07-	79	1,6	5	5	1,5	5	5	5
AB00-	80	1,6	5	5	1,5	5	5	5
AB01-	81	1,6	5	5	1,5	5	5	5
AB02-	82	1,6	5	5	1,5	5	5	5
PRIN-	83			5	5			5
PROT-	84	4	J]	4	4		*	4
GND	85							
GND	86							

2. DPIN-, DPOT-, MBIN-, MBOT-, AND TYP1- ARE STRUNG THROUGH THE 200 SERIES CONNECTORS ONLY. THESE PIN POSITIONS ARE UNASSIGNED ON THE 100 SERIES CONNECTORS AND ARE RESERVED FOR FUTURE EXPANSION.

3. THESE PINS CARRY SPECIAL SIGNALS ON SLOT A100 AND ARE RESERVED FOR FUTURE EXPANSION ON THE REMAINING 100 AND 200 SERIES CONNECTORS.

4. EBSEL-, PIN 211, IS USED FOR TEST ONLY.

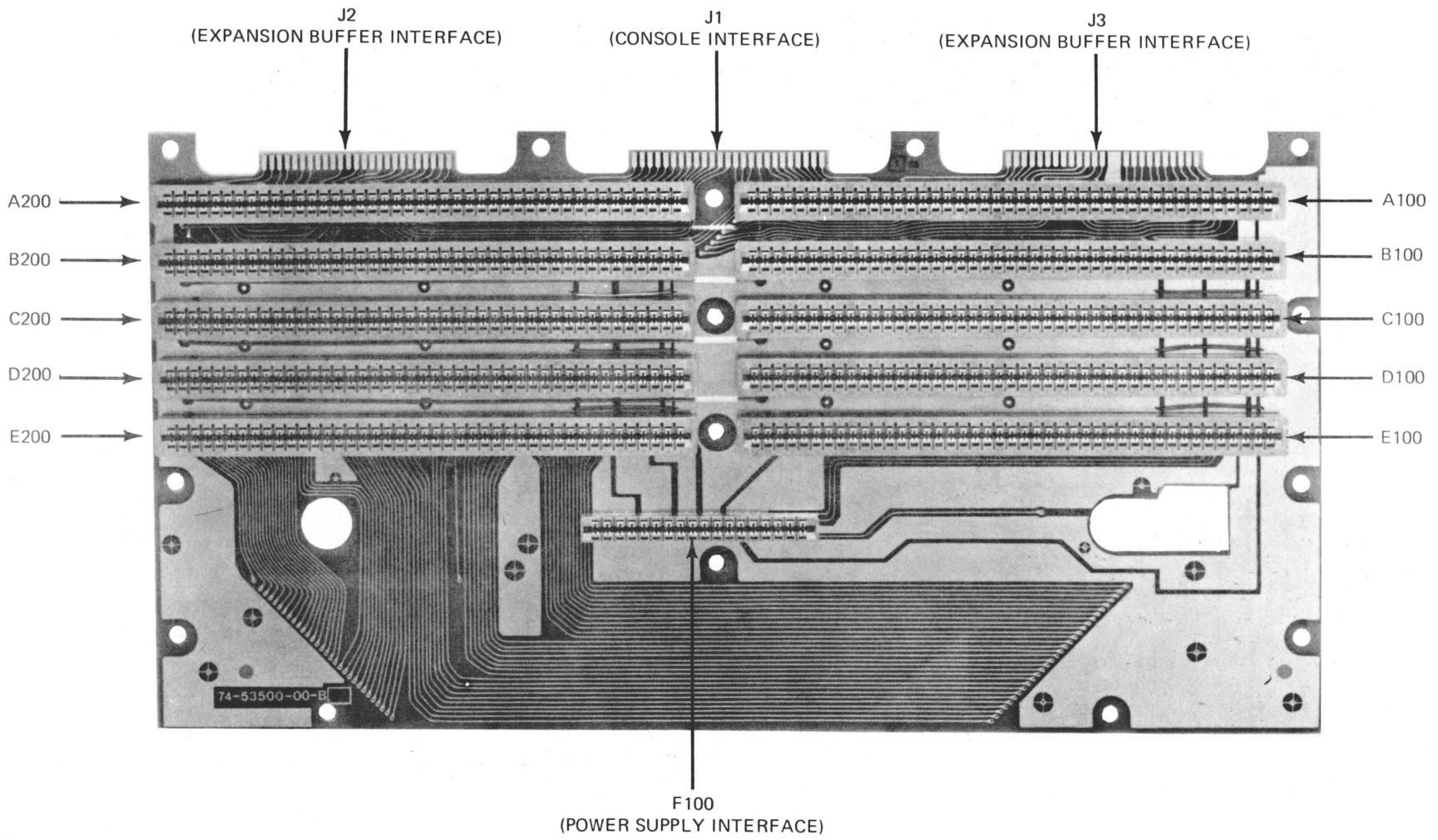


Figure 8-11. ALPHA LSI Motherboard Slot Organization (Rear View)

Section 9

DEVICE INTERFACE CONTROLLER, DESIGN TECHNIQUES

9.1 INTRODUCTION

This section describes how to design a device interface (I/O) controller that will be compatible with the I/O structure of the ALPHA LSI computer. The logic circuits described here are from Computer Automation, Inc. standard interface products that are successfully performing at user installations throughout the world.

9.2 I/O CONTROL IMPLEMENTATION

The following paragraphs describe I/O controller design requirements for compatibility with the I/O structure of the Processor.

9.2.1 Device Address Decoder (Figure 9-1)

The Device Address decoder is a comparator circuit which compares the five-bit Device Address field of an I/O instruction with the user assigned device address.

The example A address decoder uses an exclusive OR (EX OR) gate and an inverter for each of the five device address bits to be decoded. The outputs of the inverters are tied together to form a wired-AND address decoder output signal, DAXX.

Address decoding is controlled by the five Peripheral Select signals (PS0- through PS4-). These signals are brought in from the device interface connector to corresponding EX OR gates. If a true (low) address bit is to be decoded, the corresponding address select signal must be externally wired to ground (ground = true). Likewise, if a false address bit is to be decoded, the address select signal must be left open permitting the pull-up resistor to provide the false (high) address select signal.

When the device address bit agrees with the address select signal, the output of the EX OR gate is low. All five device address bits must agree with the user defined address selection. If agreement is obtained, the decoder output signal DAXX goes high enabling recognition of I/O instructions.

Example B shows an address decoder which decodes Device Address 6. This type of decoder is used only in dedicated applications and does not provide the flexibility that the example A decoder offers. Refer to appendix B for standard device address assignments.

CAUTION

Device Address : 00 should not be used. This address is reserved for Processor mounted options, the Console and certain control instructions. Using it will cause improper operation of the Processor. Furthermore, a device interface connector containing properly installed device address jumpers must be applied to the rear-edge connector at all times. If it is not, a default address of : 00 will be assigned to the module, causing the same problem referred to above.

9.2.2 Function Decoder (Figure 9-2)

The Function decoder uses an MSI chip, or a network comprised of SSI chips, to decode the contents of the Function field of the Address bus. The result is a function code (1 of 8 maximum) which performs some function in the selected I/O controller.

The choice of chips depends upon the user's application. Figure 9-2 shows three examples, A, B and C, of how to implement the Function decoder. When decoding three or less functions, example C may be the most efficient. However, if chip count is a factor, example A or B is probably more efficient. In any case, where more than three functions are to be decoded, example A or B is probably the most efficient.

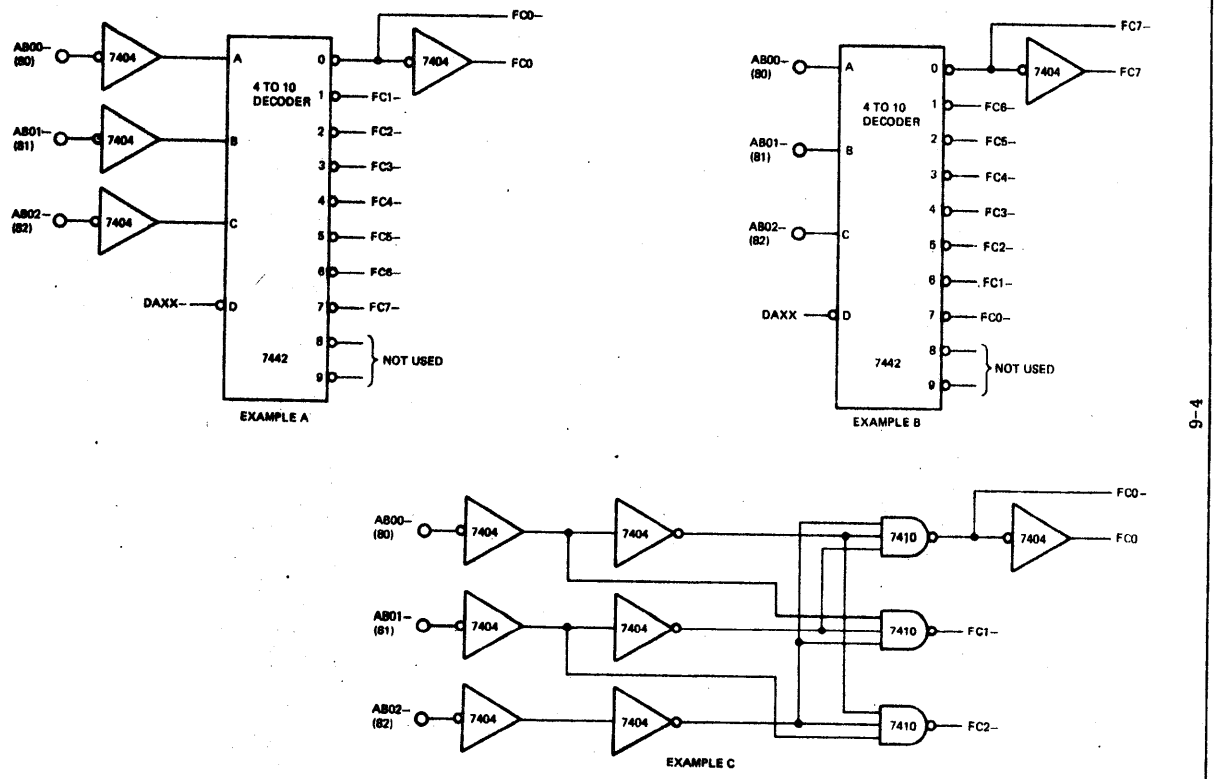
9.2.2.1 Example A

Example A uses a TTL 7442 MSI chip which is a 4 to 10 Decoder. Inputs A, B and C are the 2^1, 2^2, and 2^3 inputs respectively. Input D is the 2^4 input. When high, input D enables decoded output 8 or 9. However, only the first eight outputs of the decoder (0 through 7) are normally used, since eight is the maximum capacity of the three Function field lines in its normal configuration. D input is the enable input for the first eight decoded outputs, and utilizes the DAXX- signal for this purpose. When the device address is decoded, the DAXX- signal goes low, thus enabling the Function decoder.

Input lines from the Function field of the A bus are first unloaded by inverter gates and then applied to the decoder. As an example, if all Function field lines were false (high, implying Function Code 0), lows would be applied to inputs A, B and C. The decode of all low inputs would be zero thus causing FC0- to go low. (Decoded outputs of a 7442 are always low.) If a high signal is required, it can be obtained by using a simple inverter gate, such as the TTL 7404 illustrated.

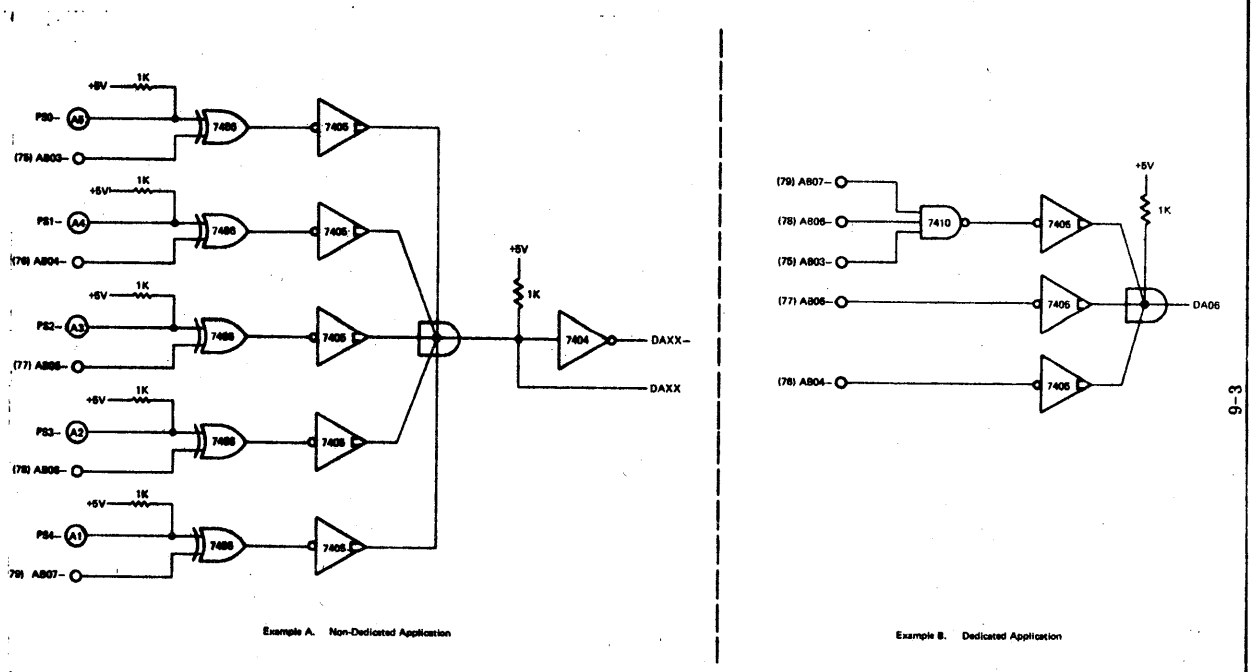
9.2.2.2 Example B

Example B is the same as example A, except that the outputs are reversed (output 7 = FC0, output 6 = FC1, etc.). However, example B can only be used where the Function field lines will not be applied to any other circuit on the same I/O controller.



9-4

Figure 9-2. Function Decoder Configurations (Typical)



9-3

Figure 9-1. Device Address Decoding Techniques

This complies with the rule that each controller represents no more than one load to each I/O line.

9.2.2.3 Example C

Example C can decode only three function codes. TTL 7410 3-input NAND gates are the decoders. The three Function field signals are applied to the appropriate NAND gates to produce FC0- through FC2-. If the decoded device address is to enable the function codes, TTL 7420 NAND gates can be used, with the DAXX signal applied to the fourth input of each gate.

9.2.3 Select, Input or Output Instruction Decoding (Figure 9-4)

Similar to the Function decoder, the Select, Input or Output (I/O) instructions can be decoded by an MSI chip or a network of SSI chips. Figure 9-4 shows two methods, example A and B, of implementing this circuit. When the various instructions are fully decoded using the Function field signals of the A bus, the Function decoder is not generally needed.

9.2.3.1 Example A

Example A shows a TTL 7442 4 to 10 Decoder used as a Select, Input or Output instruction decoder. The decoder also decodes the contents of the A bus Function field, but only for the specific type of I/O instruction with which it is being used. Assume the decoder is used as a Select instruction decoder. The contents of the Function field are applied to the A, B and C inputs to produce the appropriate function code--any one of up to eight associated with the Select instruction. The decoder is enabled by NANDing DAXX (device address decoded), EXEC and PLSE. The Select instruction and associated functions are decoded by the one circuit. Refer to paragraph 6.4 for Select instruction timing.

9.2.3.2 Example B

Example B shows a decode network of SSI chips. This circuit can offer greater efficiency than the 7442 chip, depending upon the application. For example, if three types of I/O instructions (Select, Input and Output) are used by a controller, and less than three functions are associated with each type instruction, it is probably more efficient to use decoders of this type, each utilizing the outputs of a single Function decoder.

9.2.4 Initialization Implementation (Figure 9-3)

Initialization circuitry establishes a known static state within an I/O controller. Initialization is started by executing a Select instruction with a function code dedicated to initial-

isation (nominally Function Code 4) or when the RST- signal goes low (upon depression of the RESET switch on the Console, or during a power fail/restart situation). Figure 9-3 shows a circuit configuration for implementing initialization. When the device address and function code of the Select instruction are decoded, the DAXX and FC4 signals go high to prime the 3-input NAND gate. EXEC goes high during the Select instruction, enabling the gate to produce the INZX- and INZX signals. These signals are distributed throughout the controller to reset or set flip-flops, data registers, counters, etc., to establish the known static state.

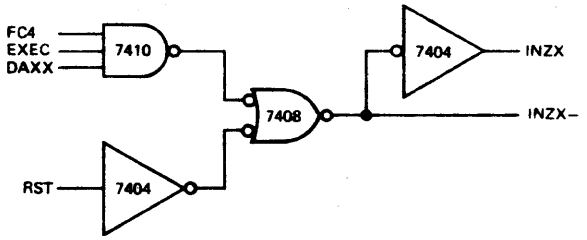


Figure 9-3. Initialization Circuit

9.2.5 Sense Instruction Implementation (Figure 9-5)

The Sense instruction circuit can be implemented using an MSI chip or a network comprised of SSI chips. As in the Function and I/O instruction decoders, application determines the most efficient method. An MSI chip can accommodate up to eight sense conditions, and provide its own function decoding. (Function code determines sense condition to be interrogated.)

The SSI network can be implemented more efficiently where three or less sense conditions are to be interrogated. However, the circuit requires inputs from a Function decoder. Both positive and negative, internal and external signals can be sensed. An example of each is described below and illustrated in figure 9-5.

9.2.5.1 Positive Sensing

Example A shows positive sensing using a TTL 74151A MSI chip. The 74151A is an 8 to 1 Multiplexer that provides internal function code decoding and an enable input (EN). It also provides both true and complement outputs. The top four inputs (0 through 3) are shown accepting External Sense (ES0 through ES3) signals from the external device. Pull-up resistors should be connected to each external input line (10K typical). Internal Sense (IS4 through IS7) signals are applied to inputs 4 through 7. When the device address is decoded, the multiplexer is enabled by DAXX- at the EN input. The outputs of the A bus Function field unloading gates are applied to the decode input of the multiplexer (AD0, 1, and 2). The appropriate sense signal, as determined by the function code, is then applied to the two outputs.

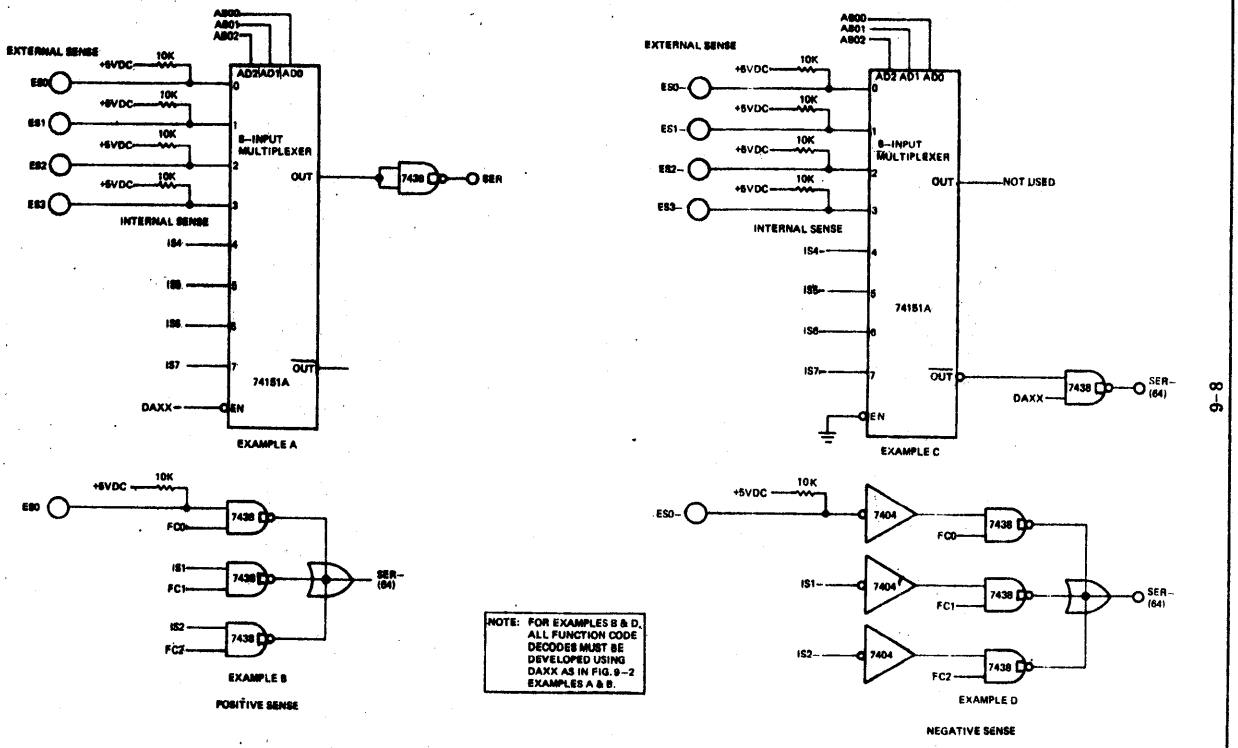


Figure 9-5. Positive and Negative Sense, Circuit Configurations

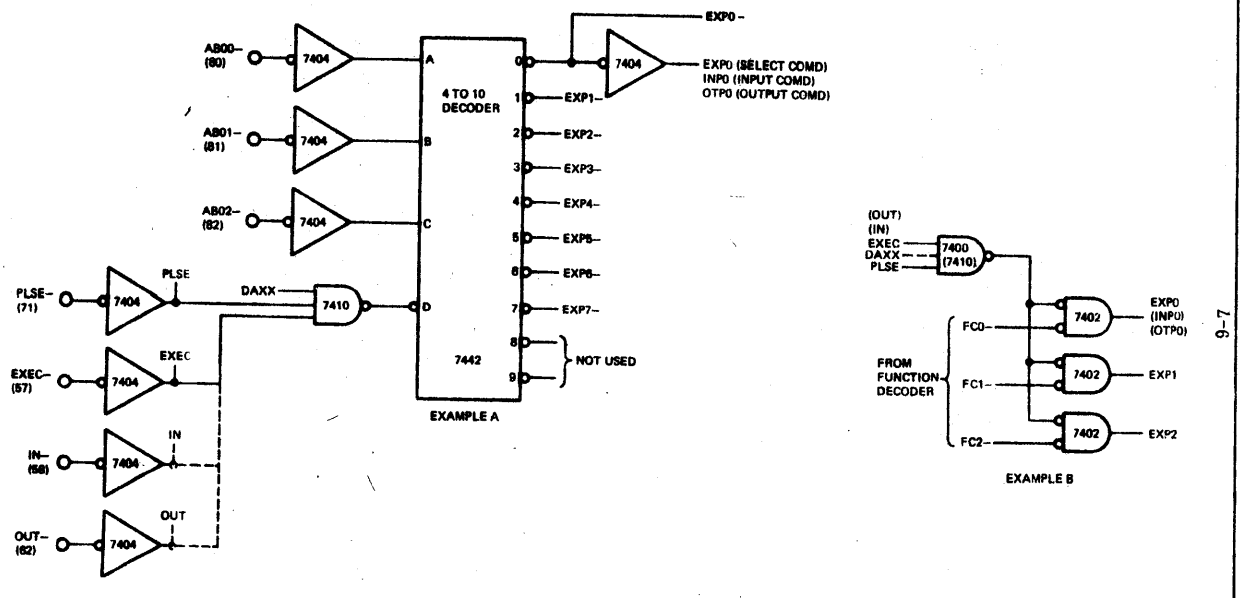


Figure 9-4. Select, Input, or Output Instruction Decode Configurations

Only the high output (OUT) is used in this case. The signal is inverted and applied to the Sense Response line (SER-) by the 7438 driver. When the OUT signal is high, the SER- line goes low. When the OUT signal is low, the SER- line stays high.

Example B shows positive sensing using SSI chips. Both external and internal sensing is again illustrated. A separate Function decoder is required to provide the necessary function codes. NAND gates combine the sense lines with the associated function codes. The outputs of the NAND gates are connected in a wire-ORed configuration to the SER- line.

9.2.5.2 Negative Sensing

Example C shows negative sensing using the 74151A MSI chip. Negative sensing is similar to positive sensing, except that the low output (OUT-) of the chip is employed rather than the high output, the EN input is grounded to permanently enable the chip, and DAXX is used to gate the multiplexer output onto the SER- line. As with positive sensing, all external sense lines should be provided with pull-up resistors.

Example D shows negative sensing using SSI chips. The negative-true signals are inverted and applied to 7438 2-input NAND gate drivers. Function code signals enable the appropriate driver. The outputs of the drivers may be connected in a wire-ORed configuration before being applied to the SER- line.

9.3 DATA TRANSFER CONTROL IMPLEMENTATION (Figure 9-6)

The efficient transfer of data between the Processor and I/O controller is controlled by the various buffer control circuits shown in figure 9-6. An Output Buffer Empty circuit controls the transfer of data from the Processor to the interface (examples A and B). An Input Buffer Full circuit controls the transfer of data from the interface to the Processor (examples C and D).

9.3.1 Example A

Example A shows an Output Buffer Empty latch (OBE) comprised of two TTL 7400 negative input OR gates. The latch is initially set upon execution of the Initialize instruction for the controller. The INZX signal goes high and is applied through the NOR gate to the set side of the latch, causing it to set. The OBE signal thus goes high and is applied to the Sense multiplexer from which it can be interrogated by Sense or Conditional Output instructions using the appropriate function code. The OBE signal can also cause an interrupt through implementation of interrupt logic. When data is transferred to the controller Output buffer, the DAXX, OUT and PLSE signals go high, enabling the NAND gate whose output is applied to the reset side of the latch. The latch now resets, inhibiting response to further interrogations by the Processor. When the data has been transmitted, a signal should be generated

to indicate completion of the transfer. (Data Transmitted--DXMT). DXMT is applied to the same NOR gate as INZX, causing the latch to set again and indicate that the buffer is ready for more data at the next Processor interrogation.

9.3.2 Example B

The circuit in example B does the same thing as example A. The only difference is a TTL 7474 D type flip-flop is used, rather than the dual NOR gate latch. INZX- direct sets the flip-flop. The high OBE signal is then available for interrogation. When data is transferred to the Output buffer, the flip-flop is direct reset. When DXMT- goes true, the flip-flop is once again set to indicate the buffer is ready to accept more data.

9.3.3 Example C

Example C shows a latch configuration of an Input Buffer Full circuit (IBF). The latch is reset by INZX upon initialization of the controller. After data has been transferred to the Input buffer, a signal should be generated to indicate the completion of the transfer (Data Received--DRCV). DRCV- sets the latch, causing IBF to go high. The IBF signal is then applied to the Sense multiplexer where it can be interrogated by the Processor with Sense or Conditional Input instructions. IBF can also cause an interrupt when implemented in the interrupt logic. When the data is transferred to the Processor, the DAXX, IN and PLSE signals go high, resetting the latch.

9.3.4 Example D

Example D shows an Input Buffer Full circuit using a TTL 7474 D type flip-flop. The flip-flop is direct reset upon initialization. The flip-flop is set when data is received (DRCV goes high). The flip-flop is then direct reset when the data is transferred to the Processor (DAXX, IN and PLSE go true).

9.4 PERIPHERAL DEVICE INTERRUPT IMPLEMENTATION

The design requirements for various interrupt structures compatible with the ALPHA LSI computers are now discussed.

9.4.1 Interrupt Address Rationale

In general, interrupts are vectored to a location within the first 256 words of Memory. The main advantage for having interrupts vectored to this area of Memory is in the housekeeping associated with certain interrupt instructions. An Auto I/O instruction, for instance, must have the word/byte count and address pointer redefined after it has been moved. An IMS instruction must have the count value redefined after it has

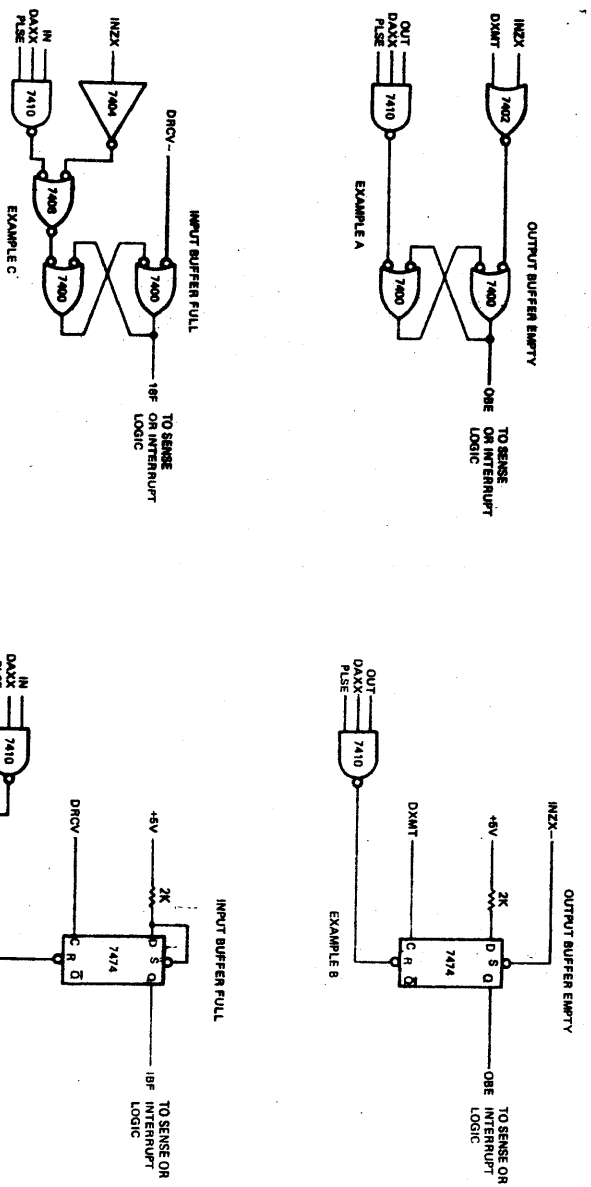


Figure 9-6. Data Transfer Control

overflowed. If the interrupt instructions are in the first 256 words of Memory, direct addressing can be used from anywhere in Memory to update the instruction parameters in anticipation of the next interrupt pass.

In applications where the use of the first 256 words of Memory for interrupts makes programming difficult, all interrupts can be offset : 100 locations into the next 256 words of Memory.

The number of memory locations that are reserved for interrupts varies with each I/O controller. If the I/O controller is intended to move data under Auto I/O interrupt control, four locations should be reserved for the Auto I/O instruction and two locations for the End-of-Block (EOB) interrupt. If a simple transfer of control is required, only two locations are required for a JST instruction. If external events are being counted, four locations must be reserved--two for the IMS instruction and two for the EOB interrupt.

If multiple interrupts are developed by an interface, these interrupts are organized into a family. Referring to appendix A, the Real Time Clock option has a four word interrupt family and the 103 Data Set Controller has a 16-word family. Family size is strictly a function of the number of interrupts an interface develops and the number of locations required by each interrupt instruction.

To preserve compatibility throughout the ALPHA computer family, I/O controllers are designed to interrupt to an even numbered address. If an I/O controller develops multiple interrupts, the base addresses of these interrupts are partitioned either two or four locations apart. The standard base addresses are : 0XX2, : 0XX6, : 0XXA and : 0XXE. These standard base addresses leave locations : 0XX0 and : 0XX8 available for special interrupts, if required.

The Auto I/O instruction requires three locations while the IMS and JST instructions require one location each. The unused reserved locations may be used for address pointers.

9.4.2 Single Interrupt Implementation Using IUR- (Figure 9-7)

This structure features an Interrupt Enable flip-flop (INTE), an Interrupt Stimulus Store flip-flop (INTS), an Interrupt Pending flip-flop (IP1), priority determination logic, priority out disable logic and an interrupt address generator.

The INTE flip-flop is a J-K type device which is synchronously set or reset by an addressed Select instruction. Function Code M (FCM) sets INTE while Function Code R (FCR) resets INTE. The INTS flip-flop is a D-type positive-edge triggered circuit. When enabled, INTS sets on the positive excursion of the external stimulus signal (EXTS).

An optional feature is an edge detector consisting of an Exclusive-OR gate and an inverter. The edge detector permits the use of either a high or low stimulus signal. The polarity of EXTS is defined by RPOL (Request Polarity). If EXTS is a low signal when active, RPOL is grounded. Likewise, if EXTS is a high signal when active, RPOL is left open and the pull-up resistor provides the positive-logic level signal. When both EXTS and RPOL are of the same polarity, the output of the edge detector will be high causing INTS to set, if enabled. Once both INTE and INTS are set, an interrupt request is generated. The Interrupt Pending flip-flop is enabled when INTE and INTS are both set. When enabled, IP1 sets on the negative excursion of the processor I/O clock (IOCL).

Once IP1 is set, the structure must have priority before an IUR interrupt request can be generated. If up-stream devices are not generating interrupts, PRIN- (Priority In, pin 83) will be low. Both PRIN and IP1 are ANDed to produce the Interrupt Request Pending (ME) signal. ME is used to develop the Interrupt Request (IUR-) signal and disable down-stream interrupts by causing PROT- (Priority Out, pin 84) to go high.

When the Processor recognizes the interrupt request, it responds by issuing the Interrupt Address Request (IAR) signal. If ME is still high (a higher priority interrupt may have been generated at the same time as this one, causing PRIN- to go high, disabling ME), IAR causes the interrupt address to be generated.

The Interrupt Address generator develops a unique vectored interrupt address. The base address that is developed is : 0XX2. The Interrupt Address Select lines (E4- through E256-) permit the user to displace the base address anywhere in the first 512 words of Memory. Grounding a particular address select line adds a corresponding decimal value to all base addresses. For example, grounding E32- adds 32 decimal locations to all interrupt addresses.

This type of address generation permits the user to redefine interrupt locations with a minimum of effort. In the event the user is limited by the number of pins available, specific data bus drivers can be used instead of the structure shown.

When ME and IAR are high (ADRR), the Data bus drivers are enabled and the interrupt address is transferred to the Processor. The Processor directs the contents of the D bus to the Memory Address register. After the Memory Address register is loaded, the PLSE signal is generated. The PLSE signal, NANDed with ADRR, will cause INTS to reset.

At the end of the last cycle of the interrupt instruction, IOCL is re-enabled. With INTS reset and IOCL enabled, IP1 resets on the negative excursion of IOCL terminating the IUR interrupt request.

The only feature of the interrupt structure not mentioned previously is the initialize feature. Generally, all controllers have an initialize circuit which generates the INZX signal. INZX sets or resets all control flip-flops to a known condition. In this case, INTE and INTS are reset by INZX. INZX is typically generated in response to an addressed Select instruction with a function code of 4, or by the Processor generated System Reset signal, RST-.

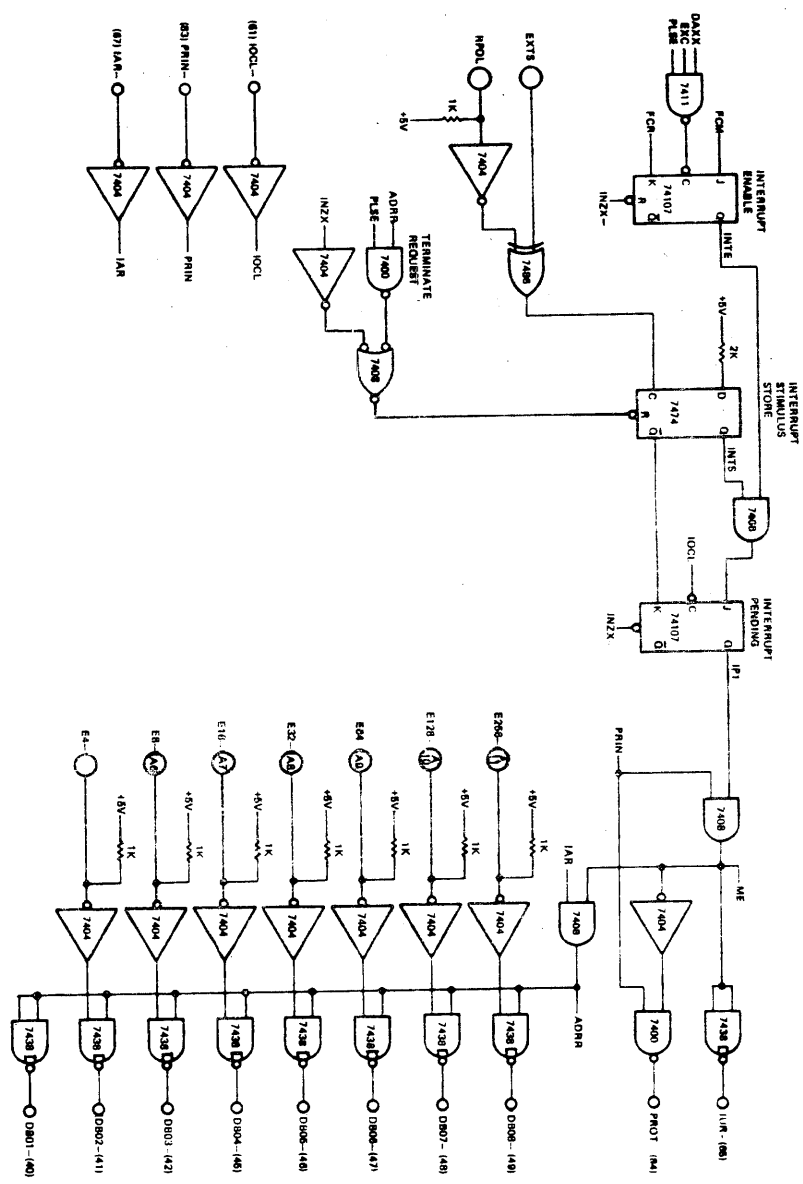


Figure 9-7. Single Interrupt Implementation Using IUR-

9.4.3 End-of-Block Interrupt Implementation Using IUR (Figure 9-10)

The interrupt structure shown in figure 9-8 develops two interrupts on the IUR- request line.

The structure is similar to the IUR structure described in paragraph 9.4.2 except that an Echo Interrupt flip-flop (ECHO1) is added. The interrupt request is developed as a result of ORing IP1 and ECHO1, and two base addresses are developed (: 0XX2 for IP1 and : 0XX6 for ECHO1).

ECHO1 is enabled by IP1 and PRIN. If the structure has priority at the instant an ECHO signal is developed by the Processor (upon determining the last word/byte of a data block has been transferred), ECHO1 sets when ECHO is received. ECHO1 is reset, if IP1 is reset, if the structure has priority when IAR and PLSE are received.

Note that IP1 is set for the entire period of the interrupt instruction and that ECHO1 is set only as long as required to obtain recognition from the Processor.

9.4.4 Reentrant Interrupt Implementation (Figure 9-8)

Reentrant interrupt programming permits an interrupt of higher priority to interrupt an interrupt subroutine. Interrupts of lower priority are not recognized. Reentrant interrupt programming requires that the Priority Out Disable latch be implemented in the user's interface hardware. When the latch is implemented, the generation of an interrupt sets the latch, which in turn disables the generation of PROT- to downstream devices.

The reentrant interrupt feature disables all lower priority interrupts for the duration of an entire interrupt subroutine. The reentrant interrupt circuit is shown in figure 9-8. The circuit prevents the PROT- signal from being transmitted to the next lower priority controller until the subroutine has been completed. The PROT disable latch is initially set when the interrupt request is acknowledged with the IAR signal from the Processor. IAR is ANDed with ME to produce Address (ADRR) which enables the interrupt address drivers and also sets the PROT Disable latch. PROTD- thus goes low, disabling the 3-input NAND gate which normally produces the PROT- signal when ME- goes false (high). Inhibiting the generation of PROT- prevents priority from being passed on to lower priority controllers until the latch is reset.

The latch can be reset by issuing a Select instruction with a function code dedicated to resetting the latch, or by initializing the controller. When the Select instruction is decoded, the DEXP (combination of DAXX, EXEC and PLSE signals) signal goes high. DEXP is NANDed with the appropriate function code (FCX) and is applied through a negative input OR gate to the reset side of the latch. The latch is thus reset and PROT- is passed on to lower priority devices (if PRIN- is low).

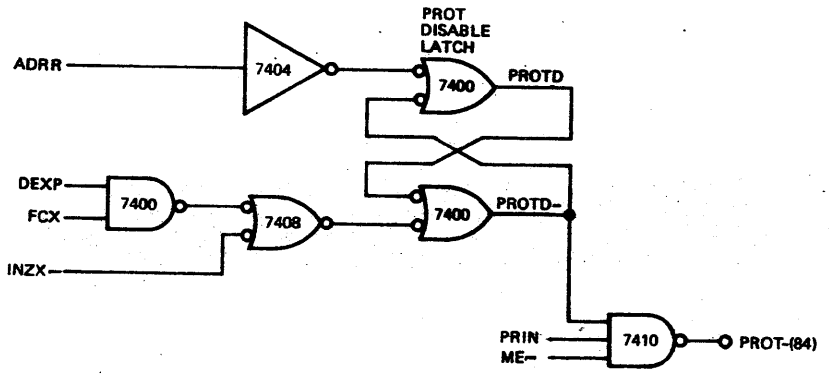


Figure 9-8. Reentrant Interrupt Implementation

9.4.5 Single Interrupt Implementation Using IL1- or IL2- (Figure 9-9)

The structure shown in figure 9-9 consists of an Interrupt Enable (INTE) flip-flop and and interrupt request driver. The INTE flip-flop is used to enable the driver. When the external stimulus is applied, an interrupt request is generated. This structure demands that the external stimulus remain active until some positive action takes place to move data or transfer control (the issuance of the IN-, OUT- or EXEC- control signals with the proper device address).

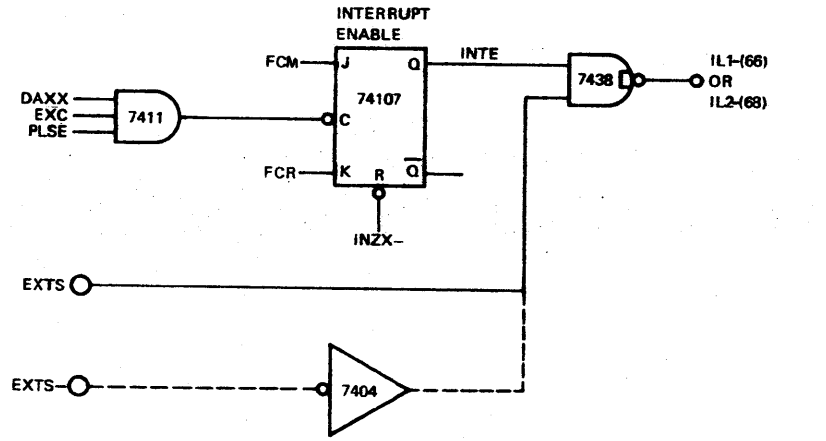


Figure 9-9. Simple IL1-/IL2- Interrupt Structure

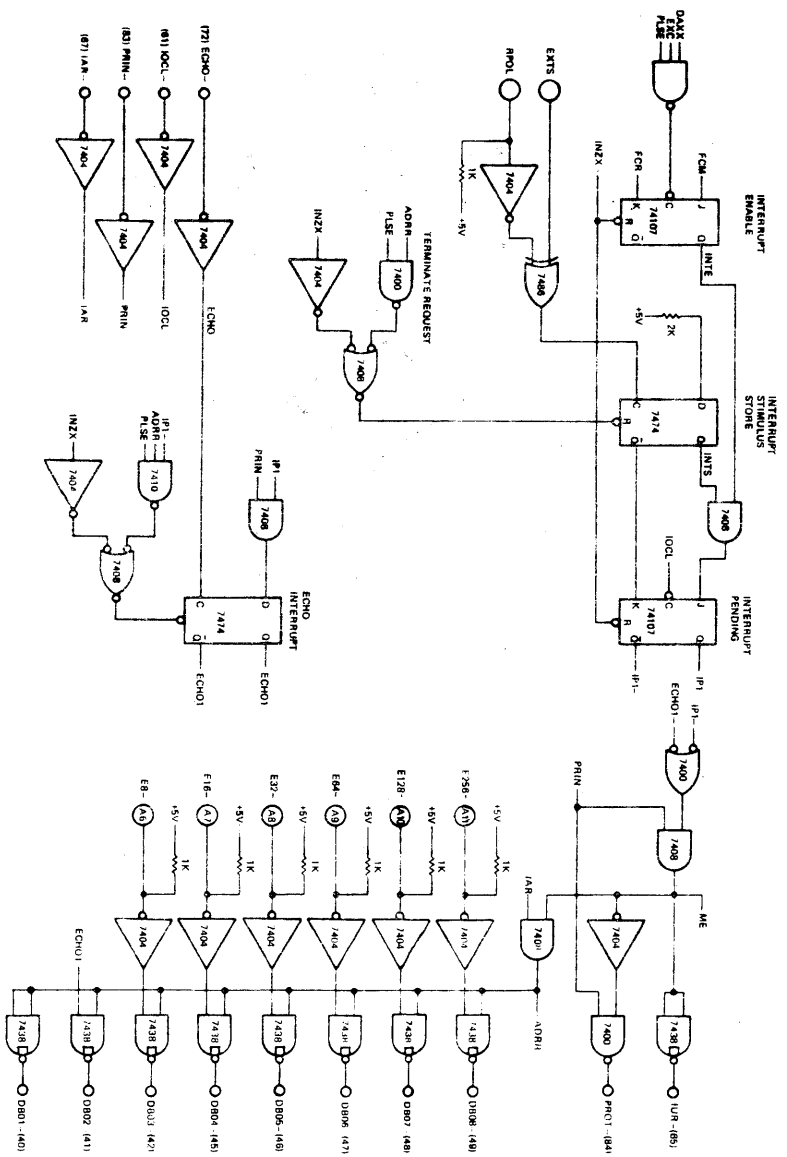


Figure 9-10. End-of-Block Interrupt Implementation

9.4.6 End-of-Block Interrupt Implementation Using IL1 and IL2 (Figure 9-12)

The interrupt structure shown in figure 9-12 develops two interrupts which utilize the IL1- and IL2- request lines. Since this interrupt structure is designed to accommodate any ECHO signal generating instruction (the four Auto I/O instructions and the IMS instruction), no other devices may be attached to the IL1- and IL2- request lines. These lines are totally dedicated to this structure.

This structure is essentially the same as the IUR- structure described in paragraphs 9.4.2 and 9.4.3. The most significant difference is that the request flip-flops are distributed directly to the IL1- and IL2- drivers. The operation of this structure is essentially the same as the IUR structures, except during request termination. Once the interrupt request is generated, the request must be recognized by the Processor. The Processor recognizes the highest priority interrupt first and all other requests in their order of priority. Since there are three higher priority interrupts above IL1 (Power Fail, Trap, and Console) and four above IL2- (the three just mentioned and IL1), the interrupt structure must be able to detect no higher priority interrupt activity before terminating the request. The only thing that the Power Fail, Trap, and Console interrupts have in common is that during the interrupt address request interval, they all cause bit 4 of the D bus to be low. If DB04- is low during IAR, the IL1 request will not reset but will remain active since the Processor has not honored the request. When no higher priority exists after generating the interrupt request, INTS is reset on the leading edge of the PLSE signal and terminates the interrupt request. To avoid retriggering the INTS flip-flop, the interrupt stimulus should remain in the active condition until an addressed I/O instruction (Select, Input or Output) causes the source of the stimulus to reset.

9.5 DIRECT MEMORY ACCESS IMPLEMENTATION

DMA controllers generally have three basic phases of operation. These phases are initialization, execution, and termination. This section provides a general overview of each of these phases. A simple overview flow chart is shown in figure 9-11.

9.5.1 Initialization

The initialization phase is used to transfer task parameters from an operating program to the DMA controller. Typically, the task parameters define operating modes, data transfer paths, the total number of transfers to be made, the starting memory address (if Memory is involved) and search parameters for items such as a disk or tape unit. The complexity of the task parameters is directly related to the complexity of the DMA controller and the various tasks it can perform. Depending on the DMA controller design, the task parameters can be transferred from Memory to the DMA controller's registers either by use of normal I/O instructions or by means of a task control block which is read from Memory by the DMA controller.

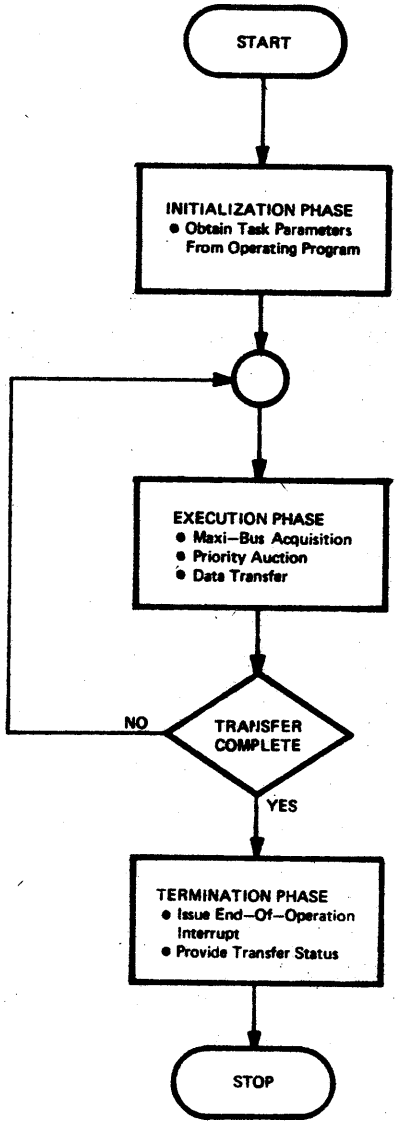


Figure 9-11. DMA Operational Phases

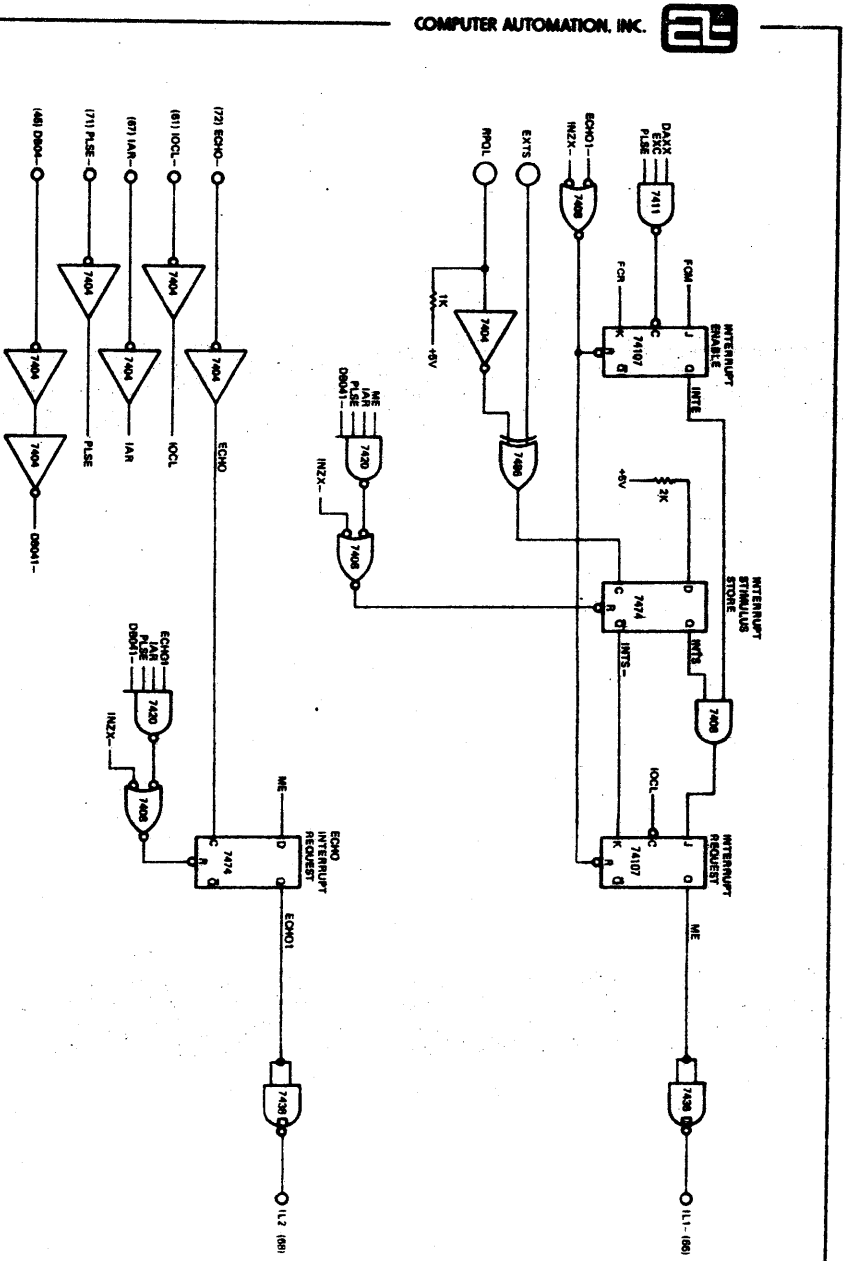


Figure 9-12. End-of-Block Interrupt Implementation Using IL1- and IL2-

Once the task parameters have been transferred, the DMA controller may begin data transfer execution.

9.5.2 Execution (Figures 9-13 through 9-15)

The execution phase is entered upon completion of initialization. When the associated peripheral logic is ready to transfer data, it generates a DMA transfer request. The DMA controller executes the DMA request in three stages. These stages are Maxi-Bus acquisition, priority auction, and data transfer. Figure 9-13 shows a typical implementation of the Maxi-Bus acquisition and priority auction logic. Figure 9-14 shows the state counter and decoder implementation. Figure 9-15 depicts the timing for both a Memory Write and Memory Read operation.

9.5.2.1 Maxi-Bus Acquisition

Maxi-Bus acquisition is initiated upon receipt of a data transfer request. The Maxi-Bus acquisition logic consists of three control elements: a Request Store flip-flop (RQ), a Request Sync flip-flop (REQF) and a STOP- driver.

The data transfer request is stored in the Request flop-flop. RQ remains set until the data transfer stage is entered.

If no DMA operations are currently in progress (processor Stop Acknowledge signal, SACK-, high), the Request Sync flip-flop is asynchronously set which causes STOP- to go low requesting use of the Maxi-Bus. If a DMA operation is in progress (SACK- low), the Request Sync flip-flop must be set synchronously with Memory Acknowledge (MACK-) to assure proper bus operation.

9.5.2.2 Priority Auction

Priority auction is required only if multiple DMA controllers are employed in the same system. Priority auction permits multiple DMA controllers to compete for use of the Maxi-Bus by means of the DMA priority string (DPIN- and DPOT-). DPIN- is the name given to the priority chain as it enters a controller and DPOT- is the name given to the priority chain as it leaves each controller. The DPOT- of one controller is the DPIN- of the next lowest priority controller. A DMA controller has priority if its DPIN- line is low. The number of DMA controllers which may be used within the system is limited only by priority ripple time on the priority string. Nominally, 200 ns are allocated to priority ripple. Where more than 200 ns is required for priority ripple, each DMA controller must be designed to abstain from beginning a transfer operation until sufficient time has elapsed for priority ripple.

Priority auction occurs at two times: after the leading edge of SACK- and, if another request has been received, after the data transfer (after the trailing edge of MACK-). If only one DMA controller is installed in the system, or if only one DMA controller is allowed to be active at a time in multiple DMA configurations, then priority ripple time need not be allocated.

Within the DMA controller, priority auction is controlled by a DMA Start flip-flop (START). START is enabled by REQF (which indicates that a synchronized data transfer request is pending) and is clocked by the leading edge of SACK- during initial Maxi-Bus acquisition, or by the trailing edge of MACK- during sequential DMA operations. When set, START inhibits downstream DMA Priority (DPOT-, high) and starts the DMA State counter.

When two or more DMA controller START controls are set simultaneously, the highest priority controller inhibits priority to the down-stream controllers. The down-stream controllers, upon seeing DPIN- high, reset their START flip-flop and DMA State counter aborting the data transfer. An aborted transfer remains pending until all higher priority DMA requests have been serviced.

Priority auction terminates when the auction interval (normally 200 ns) has been timed out.

9.5.2.3 Data Transfer

When the data transfer interval is entered, the DMA controller is free to initiate data transfers to or from Memory or another I/O controller. All data transfer timing is controlled by the DMA controller per paragraph 10.2.2 for memory transfers and per section 7 for transfers to/from another I/O controller. For each data transfer, the DMA controller must generally decrement a Word or Byte counter and increment an Address counter if transferring data to/from Memory. These overhead operations generally take place immediately after a data transfer to assure that address information is stable during the next data transfer. When a data transfer is completed, the DMA controller enters the Priority Auction stage if more data transfers remain or enters the termination phase if all transfers are complete.

The RST signal should never be used to clear the DMA Data Transfer logic since RST is an asynchronous signal and may occur in the middle of a memory cycle. To guarantee that the DMA Data Transfer logic is initiated in the proper state when power is first applied, the MDIS- signal should be used as shown in figure 9-13 and 9-14.

9.5.3 Termination

A DMA controller should provide for two types of termination: normal and abnormal. A normal termination occurs when the Word counter decrements to zero with no errors detected. An abnormal termination occurs if an error condition exists. Since DMA transfer operations can be terminated for a variety of reasons, termination flags should be used to store the reason for a termination.

When a termination condition exists (either normal or abnormal) subsequent DMA transfer requests are inhibited, Maxi-Bus control is returned to the Processor, and an End-of-Operation (EOP) interrupt is developed by the DMA controller. In some cases,

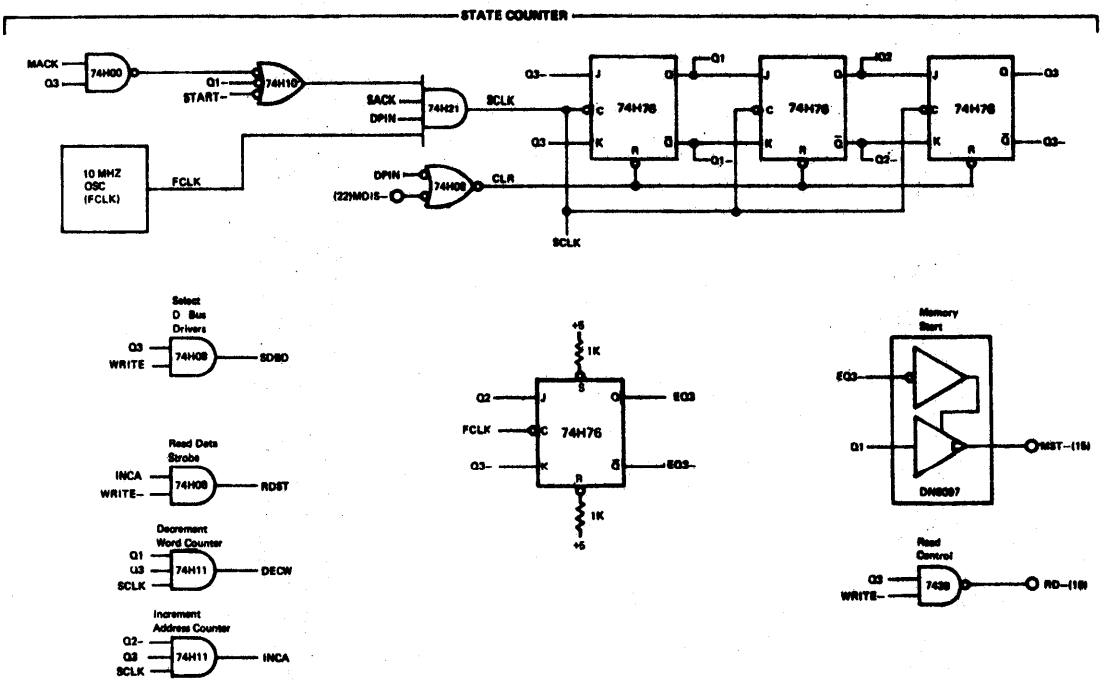


Figure 9-14. State Counter and Decoder

9-24

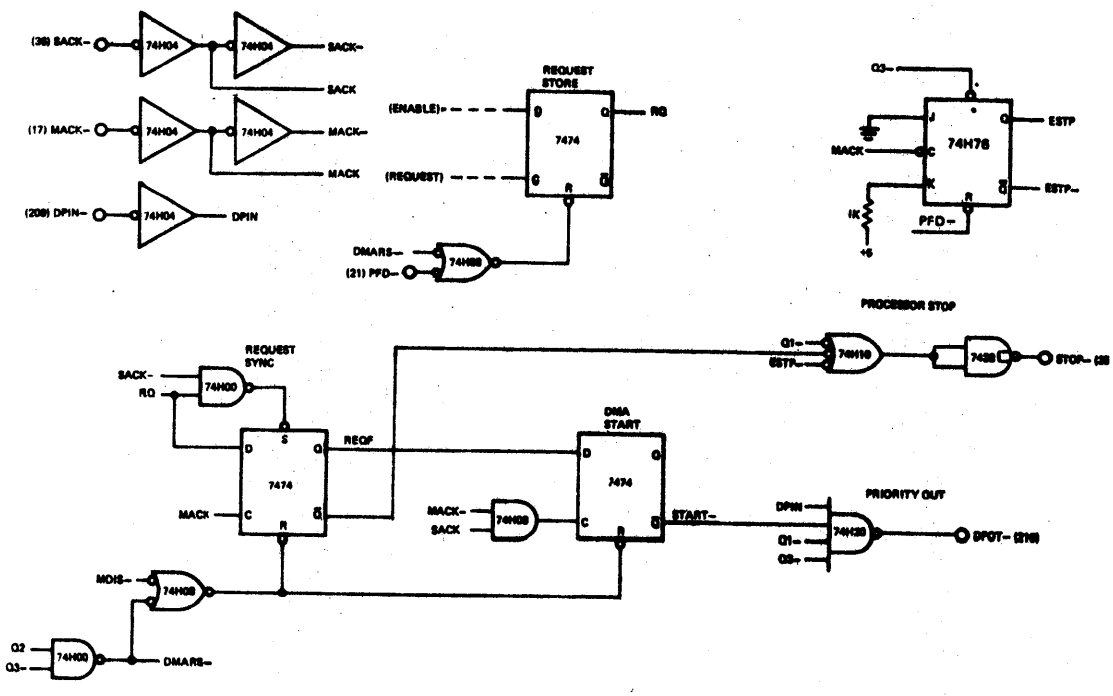


Figure 9-13. Maxi-Bus Acquisition and Priority Auction Controls

9-23

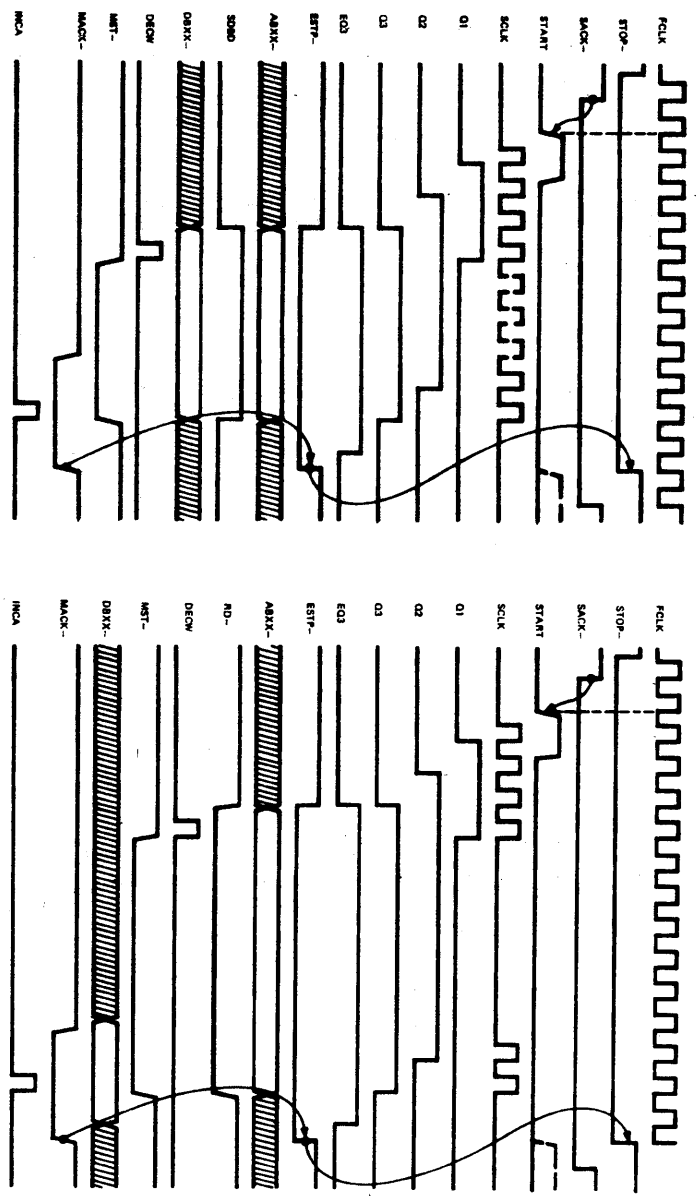


Figure 9-15. DMA Transfer Timing

it may be desirable to have the Processor periodically examine DMA controller status rather than generate a termination interrupt.

Typically, the EOP interrupt service routine will input the termination flags and any other pertinent status, and determine if the complete transfer was acceptable. If the data transfer was not acceptable, the software may retry the transfer operation if it deems it necessary.

It is the responsibility of all DMA controllers to terminate with the current bus operation and not request further bus operations in the event of a power failure (PFD- low). This is necessary to allow the Power Fail/Restart circuitry to interrupt the Processor so that a software power down subroutine can be executed. Normally a DMA controller will set a termination flag in the event of a power failure during active operation so that software will be aware of an incomplete operation.

9.5.4 Basic DMA Controller Architecture

A typical DMA controller interfaces between Memory and a high speed peripheral device. It must be able to emulate the Processor in terms of controlling Memory and making block data transfers of any length. A typical DMA controller must be able to perform the following operations:

1. Provide initialization sequencing by programmed I/O or DMA transfer.
2. Stop the Processor to seize control of the Maxi-Bus.
3. Initiate a memory cycle.
4. Define either a Read or Write operation.
5. Provide temporary data storage and asynchronous data transfer to/from the associated peripheral.
6. Maintain the memory address for the current transfer and increment the address for the next transfer.
7. Maintain a count of the number of remaining transfers.
8. Provide error detection.
9. Terminate transfer operations (surrender Maxi-Bus to Processor) after the last transfer or upon an error indication.
10. Provide End-of-Operation interrupt or status response.

A basic DMA controller features a Control section, a Word/Byte counter, an Address register/counter and a Data channel as shown in figure 9-16.

9.5.4.1 Control Section

The Control section consists of Initialization logic, a Mode Control register, Maxi-Bus acquisition controls, DMA Priority logic and a 3-bit State counter and decoder.

The initialization logic is used to set up the DMA controller for subsequent operation. It generates load signals for the mode control flags, the Word counter and the Address

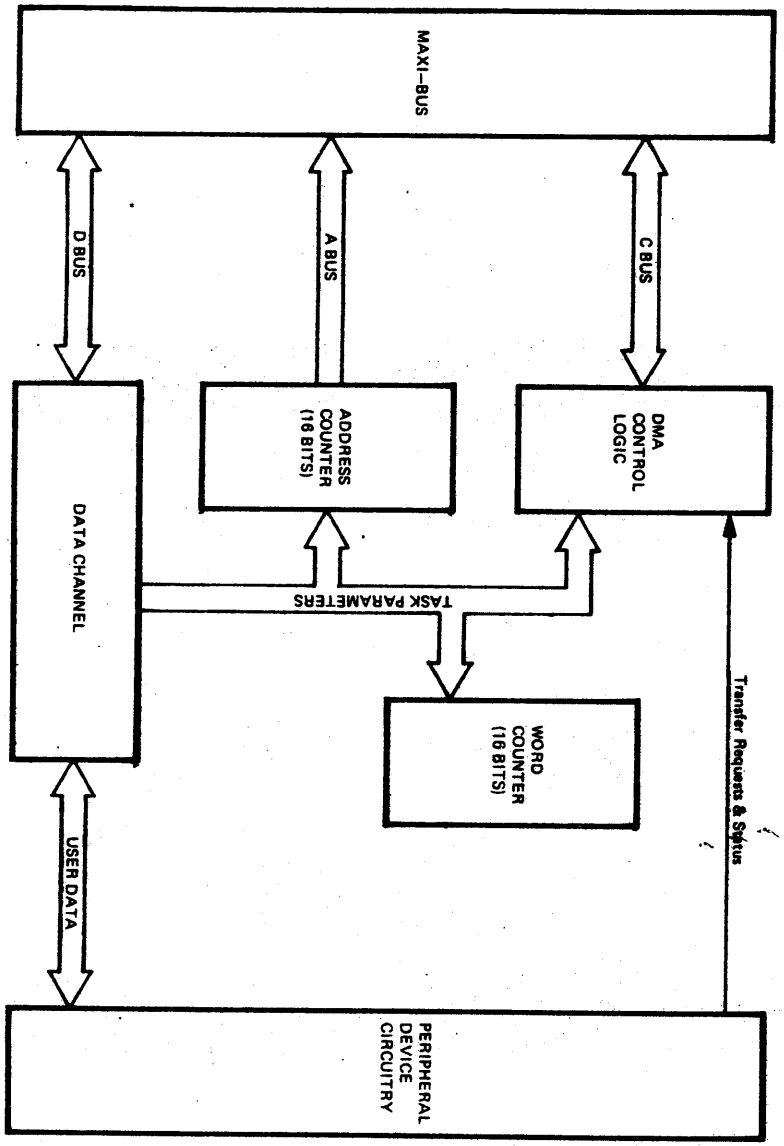


Figure 9-16. Basic DMA Controller Architecture

register. Two techniques can be used to implement the Initialization logic. One technique involves the use of programmed I/O to set flags and load registers. An alternate technique involves the use of a sequencer and the DMA control logic to access a task control block in Memory.

The Maxi-Bus acquisition controls issue the Processor STOP- signal in response to a DMA request.

The DMA Priority logic permits DMA operations between multiple DMA controllers. During each DMA cycle, the DMA priority is auctioned so that a higher priority DMA controller can transfer data.

The 3-bit State counter is used to time all operations during a data transfer. The decoder network decodes specific states of the counter to generate a Memory Start (MST-) signal, increment or decrement registers and gate data and address information to Memory.

The Mode Control register has a minimum of 1-bit storage for the Read/Write mode flag. If the user wishes to implement the Byte mode, a Byte mode flag is required to distinguish word transfers from byte transfers. The register may be expanded to accommodate other user defined flags as deemed necessary.

9.5.4.2 Word/Byte Counter

The Word/Byte counter is a 16-bit parallel-loaded binary counter. During initialization, the word/byte count that corresponds to the total number of words or bytes to be transferred is parallel loaded into the register. During execution, the Word/Byte counter is decremented with each DMA transfer to or from Memory. The counter also requires a word count equal-to-zero detection feature. This feature monitors the count during each transfer such that when the word count reaches zero, subsequent DMA requests are inhibited and termination operations are performed (typically an End-of-Operation interrupt).

9.5.4.3 Address Counter

The Address counter is a 16-bit parallel-loaded binary counter. During initialization, the starting address of the memory area being accessed is parallel loaded into the low order 15 bits of the counter. The MSB of the counter is set false for Word mode and true for Byte mode. During execution, the Address counter is incremented for each transfer (after MACK- is received). During Byte mode operations, the Select Least Significant Byte (SLB) flag is used as the LSB of the address count. When SLB- is low, the least significant byte of the transferred data word is read from or written into Memory. Likewise, when SLB- is high, the most significant byte of the transferred data word is used. SLB- must be high (or not used) during Word mode operation.

9.5.4.4 Data Channel

The Data channel is a temporary storage element that serves as a staging area for DMA data transfers to or from Memory. The complexity of the channel is determined by two factors. The first factor is DMA latency. DMA latency is defined as the time required, under worst case conditions, for the Processor to surrender the Maxi-Bus to a DMA controller. This worst case time for the NAKED MINI/LSI with the standard 1600 ns Memory is 5.6 μs (this is the maximum time that the Processor requires to do internal housekeeping and generate a Stop Acknowledge (SACK-) signal. The second factor that determines Data channel complexity is the user's maximum data transfer rate when writing into Memory.

Using the 5.6 μs DMA latency as a constant, the number of buffers that would be required for temporary data storage in the Data channel is directly related to how many word transfers could be attempted prior to gaining control over Memory. For instance, if the user has a data transfer rate of 750 kilowords per second, 1.3 μs would be required for each data transfer. With a latency of 5.6 μs and a transfer rate of 1.3 μs, a minimum of four words would be transferred and the transfer of a fifth word would have started before Memory was under control. Thus, five buffers would be required for a 750 kiloword transfer rate. Furthermore, the memory capability would have to operate in the interleaved mode. The number of buffers required for various transfer rates are summarized in the following chart:

Data Transfers Up To	Number of Buffers Required
178,571 words/bytes/sec	1
357,142 words/bytes/sec	2
535,713 words/bytes/sec	3
714,284 words/bytes/sec interleaved	4
892,855 words/bytes/sec interleaved	5
1,071,426 words/bytes/sec interleaved	6
1,249,997 words/bytes/sec interleaved	7

The user can avoid the necessity of multiple buffers by use of a Hog Mode flip-flop. This flip-flop keeps the STOP- line active and disables down-stream priority even though transfer requests are not occurring at a sufficient rate to sustain "Burst" mode. In the Burst mode, every memory cycle is dedicated to DMA transfers, i.e., 16-bit word transfer rate of 625 kHz (single memory module). The TYP1- signal on the mother board permits the DMA controller to sense which Processor is installed and perform Hog mode transfers if necessary. TYP1- is ground when the LSI-1 is installed and is open when the LSI-2 is installed.

9.6 PRIORITY AND MEMORY BANKING PROPAGATION

It is the users' responsibility to propagate the Interrupt priority, DMA priority, and Memory Banking chains regardless of whether or not a module is associated with any of these chains. If a module is not associated with any of these chains, the corresponding chain signals (namely PRIN- and PROT- for Interrupt priority, DPIN- and DPOT- for DMA priority, and MBIN and MBOT for Memory Banking) must be propagated through

the module for use by down-stream modules. These signals should be jumpered together within the module. The ALPHA LSI motherboard input and output pins for Interrupt and DMA priorities, and Memory Banking, are given in the chart below.

	INPUT MNEMONIC	PIN	OUTPUT MNEMONIC	PIN
Interrupt Priority	PRIN-	183 & 283	PROT-	184 & 284
DMA Priority	DPIN-	209	DPOT-	210
Memory Banking	MBIN	237	MBOT	238

Modules associated with Interrupt or DMA priority, or Memory Banking, should use TTL gates for unloading and driving the corresponding chain signals. It is imperative that the propagation delays internal to the modules be minimized. A total of two microseconds is allowed for signal propagation through all modules in a chain. The implementation of expansion chassis Buffer board look-ahead propagation limits the longest signal propagation path to the maximum number of modules that can be installed in two chassis (20 half board modules). Signal propagation delays should therefore be held to less than 100 ns average per module.

9.7 I/O BUS LOADING RULES

For loading rules, see maxi-bus electrical characteristics, paragraph 8.6.

9.8 POWER AND GROUND SYSTEM CONCEPTS

The power supply that is furnished with the ALPHA LSI computer produces three voltages: +5 Vdc, +12 Vdc and -12 Vdc. The +5 volt supply is used to provide the Vcc voltage for all integrated circuits in the Processor, Memory and I/O modules. The +12 and -12 volt supplies are used by the Processor and memory modules and are available to all I/O modules if needed. Typically, the +12 and -12 volt supplies provide power for analog and communications type interfaces. All three regulated voltages share a common ground system referred to as logic ground.

Power (+5, +12 and -12 Vdc) and logic ground are distributed from the system power module through the motherboard to all plug-in modules. Within a module, +5V and ground are distributed by means of bus bars. The power and ground pins on the motherboard are organized such that each bus bar can pick up a separate set of pins.

A typical half board module has a density of 72 integrated circuits which are organized in six rows of 12 chips. A typical full board module has a density of 144 IC's organized in 12 rows of 12 chips. Bus bars are mounted in between each row of chips and on the outside edges of a board. A half board module has seven bus bars while a full board

module has 13. Odd numbered bus bars are ground, even numbered bus bars are +5 Vdc.

Most 14-pin chips use pin 14 for Vcc (+Vdc in this case) and pin 7 for logic ground. A typical 16-pin chip uses pin 16 for Vcc and pin 8 for logic ground. By alternating the pin 1 orientation of each row of chips, two rows of chips can share a common +5 or ground bus bar. The Vcc pins of all chips in adjacent rows are routed to the nearest +5 bus bar mounting pad. Likewise, all ground pins in adjacent rows are routed to the nearest ground bus bar mounting pad.

The bus bar is designed such that when it is installed there is a .030 inch gap between the underside of the bus bar and the printed circuit board. This is to permit etched circuitry to pass beneath the bus bar without shorting. (Refer to figure 12-3.)

Table 9-1 lists all power and ground pin assignments that exist in the 100 and 200 series connectors of a typical motherboard slot.

Table 9-1. Power and Ground Pin Assignments

PIN	SIGNAL	PIN	SIGNAL
1,2	Ground	43,44	+5 Vdc
3,4,5,6	+12 Vdc	59,60	Ground
7,8	-12 Vdc	73,74	+5 Vdc
13,14	+Vdc	85,86	Ground
27,28	Ground		

There are two ground systems in the ALPHA LSI computer. They are logic ground and chassis ground. It is recommended that the user avoid tying these two ground systems together. The chassis ground system usually has more noise than the logic ground system can tolerate. In the event it is necessary to tie the two systems together, they should be tied together at only one point in the users' system. For personnel protection, the chassis ground system is tied to earth-ground via the third wire in the ac line cord.

9.9 FILTERING TECHNIQUES

Integrated circuits introduce switching transients into the +5 Vdc power supply which must be filtered out. It is recommended that both high frequency and low frequency filtering be employed. The low frequency filter consists of a 2.2 μF, 10%, 20 Vdc tantalum capacitor between +5V and ground for each row of 12 chips. The high frequency filter consists of a .022 μf, 25 Vdc ceramic capacitor between +5V and ground for every four chips in a given row of chips. Thus, a typical half board module would have 6 tantalum capacitors and 18 ceramic capacitors for transient filtering. Where a large number of MSI devices and Fairchild 9602 one-shots are used, it is recommended that a .022 μF ceramic capacitor be used for each device.

The -12 Vdc supply is used by the inhibit drivers in Memory. The inhibit drivers introduce approximately .5 volts of transient noise into the -12 Vdc power supply. If the user cannot tolerate this much noise, an inductive type filter is recommended.

9.10 STANDARD INTERFACE CONNECTOR

The standard interface connector is a Viking 3VH50/1JN5 or equivalent. This connector features two rows of 50 contacts designated A1 through A50 and B1 through B50. Contacts A1 through A50 interface with the contact strip on the solder side of the PC board. Contacts B1 through B50 interface with the component side of the board. The interface connector should be installed with pins B1 and A1 to the left as viewed from the rear of the computer.

9.11 NORMAL INTERFACE PINS

The interface pin assignments normally used by CAI for device address and interrupt address jumpers are listed in table 9-2.

Table 9-2. Normal Interface Pins

PIN	SIGNAL	PIN	SIGNAL
A01	PS4-	B01	+5Vdc
A02	PS3-	B02	+5Vdc
A03	PS2-	B03	GND
A04	PS1-	B04	GND
A05	PS0-	B05	GND
A06	E8-	B06	GND
A07	E16-	B07	GND
A08	E32-	B08	GND
A09	E64-	B09	GND
A10	E128-	B10	GND
A11	E256-	B11	GND

Section 10

CONSOLE INTERFACE REQUIREMENTS

10.1 INTRODUCTION

A Console, be it the standard ALPHA/LSI Programming Console or a user designed Console, is an I/O device with a special set of dedicated I/O instructions having special mnemonics.

The Console is assigned Device Address 0 (DA0) and shares this device address with the Power Fail/Restart option, the Autoload option and the Console interrupt and Trap controls of the Processor.

The Console communicates with the Processor via the Maxi-Bus and uses a special set of control signals (not considered part of the Maxi-Bus) to stop, step, and start the Processor.

This section provides a detailed discussion of interface signals, transfer timing, data formats, etc. This section also discusses the minimum requirements of a Console and how to add features to the minimum configuration Console.

10.2 CONSOLE - PROCESSOR INTERFACE (Figure 10-1)

The Console interfaces to the Processor via the Maxi-Bus, plus special control lines not generally considered to be part of the Maxi-Bus. The special lines and the associated functions are described below. The signals are all ground-true.

SERV- Console Service. The SERV- signal is issued by the Console to command the Processor to service the Console. The SERV- line may be considered an interrupt line with priority over all interrupts, but superseded by DMA operations. The Processor responds to SERV- by performing a Console Control word (CCW) input (actually, an instruction fetch from the Console instead of Memory). The CCW determines the required servicing.

IF- Instruction Fetch. The IF- signal, issued by the Processor, envelops the instruction fetch cycle. In response to SERV-, the Processor performs an instruction fetch cycle, which in this case is a CCW fetch instead of the usual memory read cycle. The Console uses IF- to differentiate the CCW input cycle from a status word input cycle; both use Device Address and Function Code 0. If SERV- is issued coincident with the leading edge of IF- or later, the instruction fetch cycle will cause an instruction to be accessed from Memory and subsequently executed before SERV- will be honored. SERV- must lead IF- by at least 1.6 μs to guarantee the next IF- cycle will be a CCW input cycle.

START- Start Processor. Signal START- is issued by the Console to command the Processor to resume processing. START- must be a minimum of 1.6 μs wide. The Processor resumes processing on the trailing edge of START-. Signal SERV- must precede the trailing edge of START- by at least 1.6 μs to guarantee the Processor will immediately perform a CCW input instead of a memory read cycle when processing is resumed.

CINT- Console Interrupt. CINT- is issued by the Console to interrupt normal processing. Signal CINT-, once issued, must be held true until signal IAR- (Interrupt Address Request) is true.

SSW- Sense Switch. Signal SSW- issued by the Console, tracks the console SENSE switch. No synchronization is required. If the SENSE switch is set, signal SSW- is true.

AL- Autoload. Signal AL- is issued by the Console to command the optional Autoload logic to perform an autoload sequence. The autoload sequence is initialized on the leading edge of AL- and commences on the trailing edge of AL-. The AL- pulse width must be 100 ns minimum.

OV- Overflow. The OV- signal is issued by the Processor. OV- tracks the Overflow flip-flop internal to the Processor.

BM- Byte Mode. The BM- signal is issued by the Processor. BM- tracks the Byte Mode flip-flop internal to the Processor.

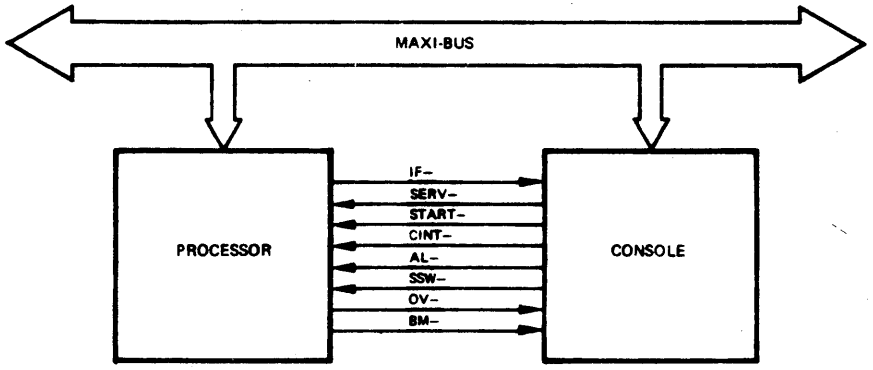


Figure 10-1. Processor/Console Interface

10.3 CONSOLE TRANSFER TIMING

There are four basic functions (beyond normal I/O functions) that a console can perform. These are: establishment of the Stop mode, register entry and display, Step mode operation, and establishment of the Run mode. The timing requirements for each of these functions are discussed in the following paragraphs.

10.3.1 Establishment of Stop Mode (Figure 10-2)

During the Run mode, the Processor Instruction Fetch signal (IF-) is ground-true when the Processor is fetching an instruction from Memory and is high during the execution of the instruction. The Console uses the trailing edge of the IF- signal to synchronize the generation of a Console Service Request (SERV-).

The Stop mode is initiated by operator activation of the console STOP switch. With the STOP switch active, the SERV- signal is enabled. SERV- goes true during the execution period of the current instruction and remains true for the next instruction fetch.

Upon seeing the SERV- signal active, the Processor fetches the next instruction from the Console rather than from Memory. When the Processor fetches the instruction from the Console, it addresses Device Address 0 and Function Code 0 and issues the IN- control signal. The Console, upon seeing IF- low, Device Address and Function Code 0 and IN- low, places a Stop CCW word on the Data bus.

The Processor vectors the Stop CCW word to its instruction register and executes the instruction. The CCW instruction algorithms cause the Processor to halt.

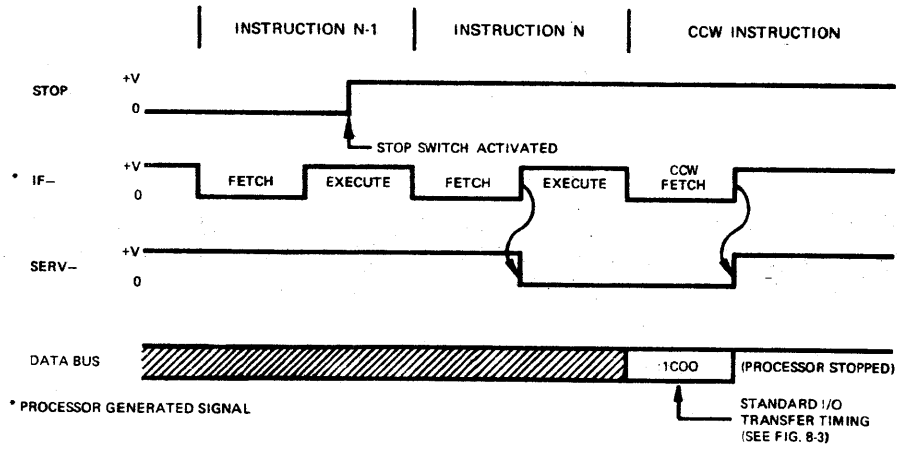


Figure 10-2. Establishment of Stop Mode

10.3.2 Register Entry and Display (Figure 10-3)

The register entry and display sequence can be performed only when the Processor is stopped. The sequence is initiated by activation of a Register Select switch on the Console. The switch activation causes both SERV- and START- (Processor Start) to go low, simultaneously. Approximately 1600 ns later, the Processor resumes operation on the trailing edge of START-.

Upon resumption of operation, the Processor recognizes that the SERV- signal is active and fetches the next instruction from the Console. The Console, upon seeing IF-, Device Address and Function Code 0, and IN- low, places the CCW on the Data bus. The Processor executes the CCW instruction and transfers data between the Console and the target register or Memory (as defined by bits 0 thru 15 of the CCW). Upon completion of the transfer, the Processor stops.

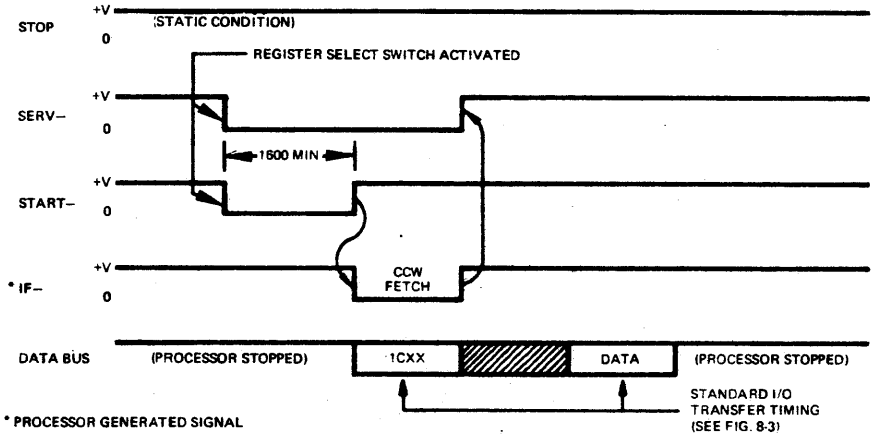


Figure 10-3. Register Entry/Display Sequence

10.3.3 Step Mode Operation (Figure 10-4)

The Step mode causes the Processor to fetch one instruction from Memory, execute the instruction and then stop. The Step mode operation can be performed only when the Processor is stopped and the console RUN switch is activated. Activation of the RUN switch causes the START- signal to go low. Approximately 1600 ns later, the Processor resumes operation on the trailing edge of START-.

The Processor, upon resumption of operation, fetches the next instruction from Memory (as defined by the current value of the P register) and executes it. The Console, upon seeing the trailing edge of IF-, generates SERV-. Upon completion of the execution of the instruction fetched from Memory, the Processor fetches a Stop CCW from the Console, executes the instruction, and then stops.

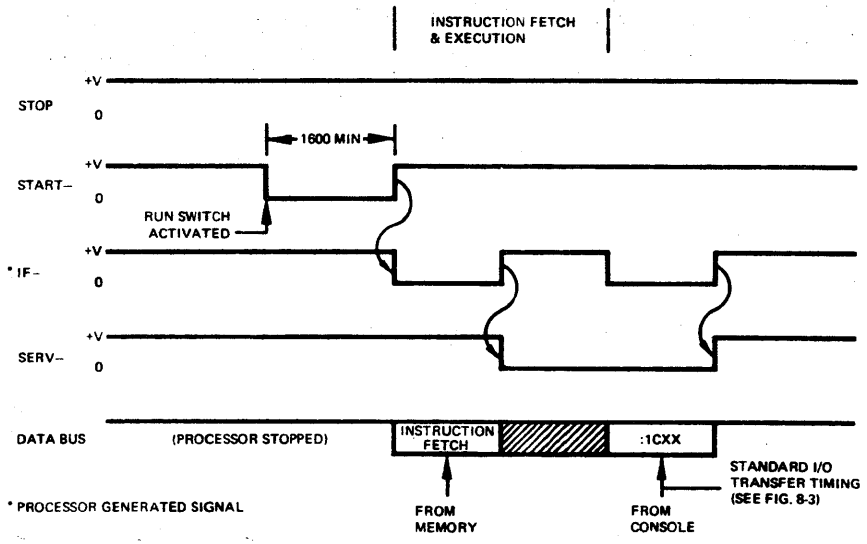


Figure 10-4. Step Mode Sequence

10.3.4 Establishment of Run Mode (Figure 10-5)

The Run mode is established by deactivation of the console STOP switch and activation of the console RUN switch. Activation of the RUN switch causes the START- signal to go low. Approximately 1600 ns later, the Processor resumes operation on the trailing edge of START-.

10.4 CONSOLE WORD FORMATS (Figure 10-6)

The NAKED MINI LSI uses four different word formats to convey information between the Console and the Processor. These word formats are as follows:

1. Computer Status Word
2. Console Sense Word
3. Console Data Word
4. Console Control Word

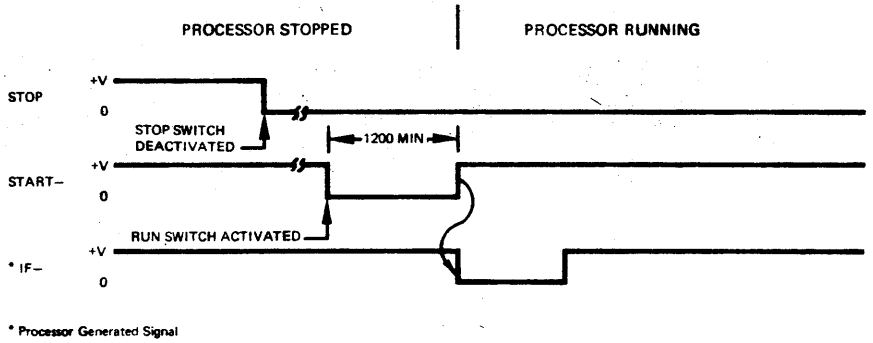


Figure 10-5. Establishment of Run Mode

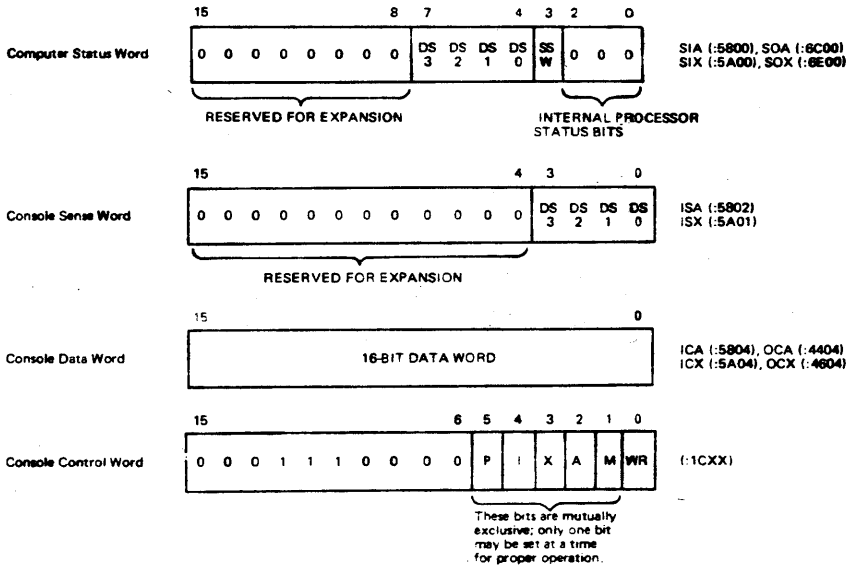


Figure 10-6. Console Word Formats

10.4.1 Computer Status Word

The Computer Status word permits the program to store volatile Sense register data during a power failure and to restore the Sense register data during restart operations. This capability is required with the standard ALPHA LSI Console since the sense data is stored in a volatile storage register. If non-volatile toggle switches are used, this capability is not required.

The Computer Status word is transferred between the Console and the Processor when IF- is false, using special unconditional Input or Output instructions with a device address and function code of 0. During an SIA or SIX instruction, the Console copies the state of the SENSE switch (SSW) into bit 3 of the word and the contents of the Sense register (DS0 thru DS3) into bits 4 thru 7, respectively. The internal processor status (bits 0, 1 and 2) is generated concurrently within the Processor. Upon input, the Computer Status word is loaded into either the A or X register. Note that the Console can drive only bits 3 thru 7 during an SIA or SIX instruction.

During an SOA or SOX instruction, bit 3 of the Computer Status word contains the new state of the SENSE switch and bits 4 through 7, respectively, contain the new state of DS0 thru DS3.

10.4.2 Console Sense Word

The Console Sense word is transferred from the Console to the Processor in response to an unconditional Input instruction with Device Address 0 and Function Code 1. During an input operation (ISA or ISX instructions), the contents of the console Sense register, DS0 through DS3, are copied into data bits 0 through 3 of the Maxi-Bus, respectively. All other bits of the word are transferred as zeroes. No Output instructions are issued by the Processor in conjunction with the Console Sense word.

10.4.3 Console Data Word

The Console Data word is a full unsigned (absolute) 16-bit data word that is transferred between the Processor and Console in response to an unconditional Input or Output instruction with Device Address 0 and Function Code 4.

During routine input operations (ICA or ICX instructions), the Console Data word is input to the Processor A or X register. Likewise, during routine output operations (OCA or OCX instructions), the Console Data Word is transferred from the Processor to the Console.

During a console service sequence, the Console Data word can be transferred to or from the Processor A, X, I or P registers as well as Memory.

10.4.4 Console Control Word

The Console Control word (CCW) is an instruction word rather than a data word. The CCW is generated by the Console during a console service sequence. The operation code of the CCW resides in bits 15 through 6 while bits 5 through 0 are modifiers.

The NAKED MINI LSI is designed to respond to eleven different CCW codes. These codes are listed below:

CCW CODE	FUNCTION
:1C00	Stop Processor
:1C02	Read Data from Memory, Increment P and Halt
:1C03	Write Data into Memory, Increment P and Halt
:1C04	Output Data from A Register and Halt
:1C05	Input Data to A Register and Halt
:1C08	Output Data from X Register and Halt
:1C09	Input Data to X Register and Halt
:1C10	Output Data from I Register and Halt
:1C11	Input Data to I Register and Halt
:1C20	Output Data from P Register and Halt
:1C21	Input Data to P Register and Halt

Note that bits 1 through 5 are mutually exclusive, namely, only one bit may be true at a time.

10.5 MINIMUM CONSOLE REQUIREMENTS

A minimal user designed Console should have facilities to stop, reset and start the Processor as well as have system performance indicators.

10.5.1 Stopping the Processor

Stopping the Processor requires the issuance of a Console Service Request (SERV-) and the furnishing of a Stop Processor CCW to the Processor upon recognition of SERV-.

The Processor will not recognize the Console Service Request until completion of the current instruction. Upon completion of the current instruction, the Processor recognizes the Console Serivce Request by initiating a CCW instruction fetch from the Console rather than the normal instruction fetch from Memory. The CCW transfer timing is discussed in paragraph 10.3.

The users Console should be designed to furnish the CCW word during an input sequence with Device Address 0 and Function Code 0 ONLY when the instruction fetch signal (IF-) is true. Once the CCW is transferred to the Processor, the internal microprogram algorithm of the Processor brings the Processor to a stopped condition.

10.5.2 Resetting the System

Resetting the system is accomplished by forcing the System Reset signal (RST-) groundtrue for a minimum of 5 μs. This can be accomplished with a switch or a TTL compatible open-collector signal capable of driving 32 mA. It is not necessary to synchronize or debounce this signal.

10.5.3 Starting the System

The system is started by issuance of the Start Processor signal (START-). START- is a ground-true signal that must have a minimum duration of 1.6μs. START- should be driven with a 32 mA open collector TTL driver.

10.5.4 Visual Indicators

Visual indicators should be provided for ease in determining the operational status of the system. Indicators should be provided on the debounced STOP switch signal and the system RESET signal. A RUN indicator can be provided by use of a 500 μs retriggerable one-shot that is triggered by the Memory Start signal MST-. As long as the system is running, the Run one-shot will be retriggered each time Memory is accessed and will time out 500 μs after the last memory access following departure from the Run mode. The RUN indicator should light whenever the Run one-shot is set. The Byte Mode signal (BM-) and the Overflow signal (OV-) are available for display. If these signals are applied to lamp drivers and indicators, an additional performance monitor can be obtained.

10.6 OPTIONAL CONSOLE FEATURES

The minimal Console discussed in the previous paragraph can be expanded to include several additional features which are discussed in the following paragraphs.

10.6.1 Data Entry and Display

The data entry and display feature provides the capability to enter data from the Console into the Processor registers or Memory. Likewise, data from the Processor registers, Memory, or a program can be stored and displayed for operator observation.

The data entry and display feature requires that the Console generate the Console Data word. Generation of the Console Data word requires a 16-bit register and 16 32 mA open-collector drivers to drive DB00- through DB15-. The entry switches can be applied via the storage register to the drivers. The drivers should be enabled only upon receipt of an Input instruction with Device Address 0 and Function Code 4 (ICA or ICX).

If the user desires to accept data from the Processor, the Console must have 16 Data bus receivers and a 16-bit holding register. The holding register must be clocked only when a Select and Present instruction with Device Address 0 and Function Code 4 is received (OCA or OCX).

Display indicators may be tied to the outputs of the storage register and should light when a corresponding bit is true.

10.6.2 Register and Memory Display and Modification

This feature permits the operator to transfer the Console Data word between the Console and the Processor A, X, I or P register or Memory.

This feature requires that, in addition to other bits, the Console be able to drive DB00- through DB05- during a Console Control word transfer. Bits 1 through 5 of the CCW must be mutually exclusive, i.e. only one bit may be true at a time.

The Console logic should be designed such that when a register select signal for bits 1 through 5 of the CCW is generated, the SERV- and START- signals are generated simultaneously. Furthermore, the generation of any CCW word, other than the Stop Processor CCW (:1C00), must be enabled only when the Stop mode is established. This is to avoid possible alteration of volatile data in a user's program during Run mode.

10.6.3 Sense Register Entry and Display

The Sense register entry and display feature permits the operator to generate a Console Sense word. The generation of a Console Sense word requires that a 4-bit Sense register be applied to four 32 mA open-collector data bus drivers (DB00- through DB03-). The drivers should be enabled only upon receipt of an Input instruction having Device Address 0 and Function Code 1.

10.6.4 SENSE Switch Feature

In addition to the four sense lines discussed above, the Processor will accept a SENSE switch signal (SSW-) that may be tested by program instructions. The SSW- signal must be ground-true when the SENSE switch is active.

10.6.5 Console Interrupt Feature

The Console interrupt feature permits the operator to interrupt normal processing. Console interrupts generate signal CINT- which is sent to the Processor. The only timing restriction on CINT- is that it must remain active until the Processor recognizes the CINT request (recognition is obtained when the Interrupt Address Request (IAR-) signal goes ground-true).

10.6.6 Autoload Initiation Controls

The Autoload initiation controls permit the operator to command the Autoload option to perform an autoload sequence. Autoload initiation should only be permitted when the system is in the Run Enable mode (STOP and RUN switches are reset or off). Autoload initiation will take place whenever the AL- signal is forced ground-true. The signal must be ground-true for a minimum of 100 ns to guarantee a response from the Autoload option.

The user may use the AL- signal to set a flip-flop which, in turn, may drive an auto-load indicator. A Select instruction with a device address and function code of 0 can be used to reset the flip-flop when loading is complete.

10.6.7 Step Mode Feature

The Step mode feature permits the operator to manually step through a program one instruction at a time. The Step mode is an extension of the Stop mode wherein, if the RUN switch is activated while in the Stop mode, the Processor will go into the Run mode, execute one instruction, recognize a console service request, process the request and then stop. Step mode timing is discussed in paragraph 10.3.

10.7 USER CONSOLE INTERCONNECTION (Figure 10-7)

A user designed Console can interface to the Processor in two different ways. If the user has the motherboard assembly, the Console can be interfaced at connector J1. If the motherboard is not employed in the users system, the Console can be interfaced directly to connector P1 of the Processor. (Intercabling must be limited to 18 inches.)

Motherboard connector J1 will accept a 50-pin 3M connector (Part number 3451-0000). This connector is designed to accommodate a SCOTCHFLEX™ ribbon cable (3M part number 3365-50). A PC board transition adapter (3M part number 3456) is also available for the console end of the ribbon cable. Note that power and ground are available at J1 in addition to all signals required for a Console. The pin assignments for connector J1 are shown on figure 10-7.

In systems that do not have a motherboard, refer to paragraph 2.3.3 of this manual.

10.8 OPTION CARD CONSOLE ACCOMMODATIONS

The NAKED MINI LSI Option board provides console skeleton logic. Included in the logic are the following capabilities:

1. Secondary Console Sense register. Grounding four jumper pins introduces corresponding logic 1 bits in the Console Sense register word for ISA and ISX instructions. Satisfies requirements of paragraph 10.6.3.

2. Secondary Console SENSE switch. A ground jumper on the pin simulates the console SENSE switch in a set state for conditional jump instructions only. Satisfies requirements of paragraph 10.6.4.

3. Secondary Console Interrupt switch. A momentary ground jumper simulates a Console interrupt. This jumper option is also available at the TTY interface connector. Satisfies requirements of paragraph 10.6.5.

CONSOLE CONNECTOR
(3M 3415-0000)

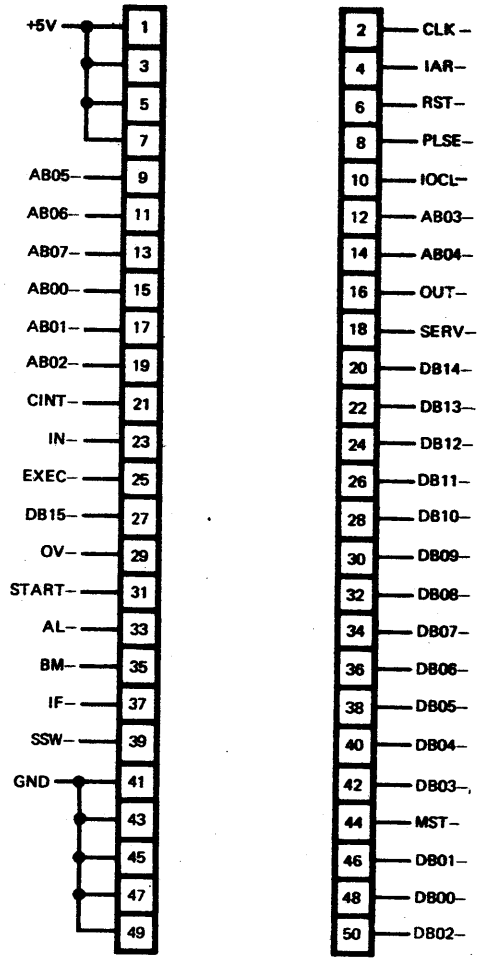


Figure 10-7. Motherboard/Console Connector (J1) Pin Assignments

4. Secondary Autoload switch. A momentary ground jumper simulates the console Autoload (AL-) signal and results in the execution of the autoload sequence. This jumper option is also available on the TTY interface connector. (Jumper is active at all times and will first reset the computer if pressed while the computer is running.) Satisfies requirement of paragraph 10.6.6.

5. Secondary Reset switch. A momentary ground jumper simulates the console Reset (RST-) signal. Satisfies requirements of paragraph 10.5.2.

Each of the above capabilities and their implementation are described in Section 6 of this manual.

Table 10-1. Console Special Signal Load/Drive Summary

SIGNAL	CPU	CONSOLE
SSW-	5,6	2
IF-	2,6	5
AL-	5,6	2
BM-	2,6	5
OV-	2,6	5
START-	2,5,6	2
SERV-	2,5,6	2
CINT	5,6	2

Device types are as follows:

2 = 32 mA open-collector driver (7438 or equivalent)
5 = TTL receiver (7400 or equivalent)
6 = Pullup resistor (1 Kohm)

Section 11

POWER SUPPLY INTERFACE REQUIREMENTS

11.1 INTRODUCTION

This section discusses the requirements of a user furnished power supply. Among the items discussed are DC power requirements, power monitor facilities, an optional ac line synchronized timing source and interconnection requirements. Refer to figure 11-1 for a top and bottom view of the ALPHA LSI power supply.

11.2 DC POWER REQUIREMENTS

The user designed power supply must produce four voltages: +5Vdc, +12Vdc, -12Vdc, and +5H (hangpower). The +5 volt supply provides the Vcc voltage for most integrated circuits in the processor, memory and I/O modules. The +12 and -12 volt supplies are used by the processor and memory modules and by the MOS LSI integrated circuits. Certain analog and communications options use +12 and -12Vdc. The +5H hangpower supply is used exclusively by the Processor; a detailed discussion of the +5H supply is provided in paragraph 11.3. All four dc voltages share a common ground system referred to as logic ground.

11.2.1 Estimating DC Current Requirements

Before a user can design a power supply, the current requirements of each dc supply must be determined. The current load of most standard modules built by Computer Automation, Inc. are listed in table 11-1. The load currents listed are worst case for each module. The user can determine actual power requirements for his system configuration by summing the load currents for each standard module (and multiples thereof) along with the load currents of any user designed controllers.

11.2.2 Overvoltage and Reverse Voltage Protection

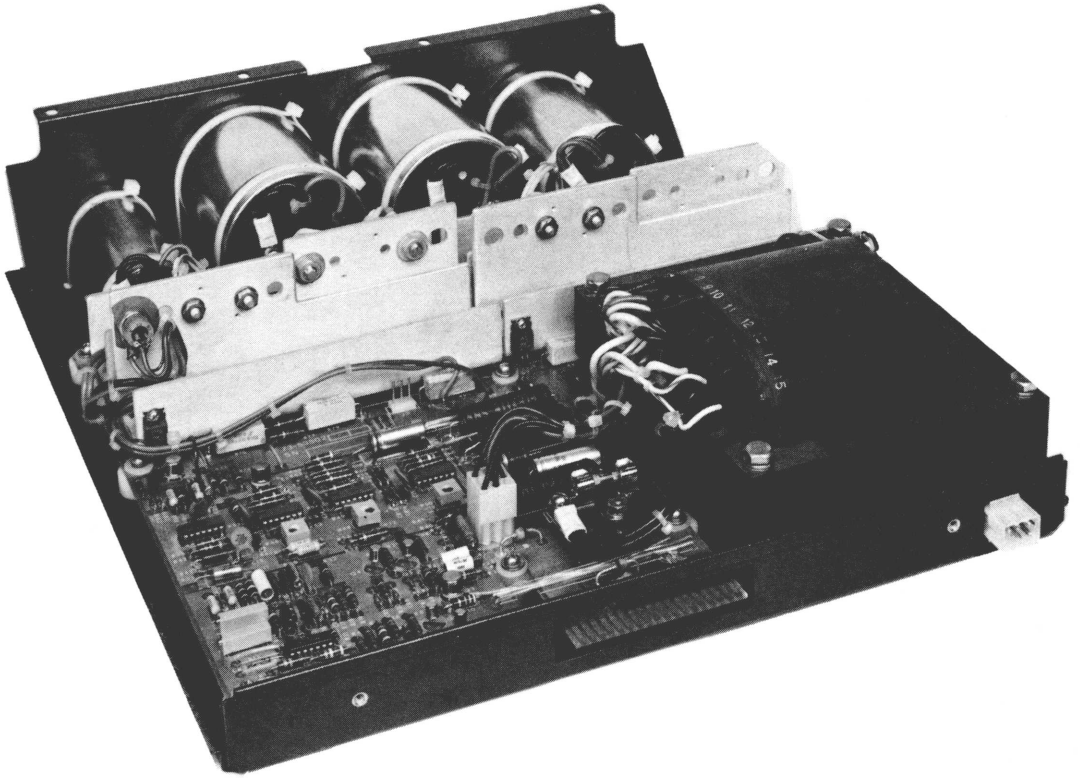
It is redommended that the +5Vdc power supply employ overvoltage and reverse voltage protection devices. The overvoltage device must prevent the +5Vdc output from exceeding +6.5 volts in the event of a power supply failure or an accidental application of a high voltage potential from an external source. Each supply output should have circuitry to prevent damage to its load or the supply itself in the event that one supply is shorted to another or to ground.

Table 11-1. Standard Module Load Currents

ASSY. NO.	COMMON NAME		DC CURRENT (AMPS) +5V	DC CURRENT (AMPS) +12V	DC CURRENT (AMPS) -12V
53537-XX	NAKED MINI LSI-1 With 2/4/8K Core 1600 Memory	-ACTIVE	5.3	.65	5.22
		STANDBY	5.0	0.2	0.47
53538-XX	NAKED MINI LSI-1 With 1/2/4K Semiconductor Memory	-ACTIVE	3.5	1.2	0.52
		STANDBY	3.5	0.7	0.52
53506-00	LSI-2 Processor (Including 53507 Module)		7.35	-----	-----
53504-XX	2/4/8K Core 1600 Memory Module	-ACTIVE	2.0	0.5	2.6
		STANDBY	1.85	0.25	0.20
53526-XX	4/8K Core 980 or 16K Core 1200 Memory Module	-ACTIVE	3.0	.65	4.9 (Note 1)
		STANDBY	2.5	0.2	0.15
53514-XX	1/2/4/8K Semiconductor Memory Module	-ACTIVE	1.6	1.2	0.20
		STANDBY	1.6	0.7	0.20
53501-00	ALPHA LSI Console		1.7	-----	-----
53505-XX	Option Card (maximum configuration)		2.15	0.03	0.14
53213-00	16-bit I/O Module		1.5	-----	-----
53214-XX	32-bit Relay Contact Output Module		1.5	-----	-----
53215-00	32-bit Relay Contact Input Module		1.5	-----	-----
53216-00	64-bit Output Module		1.5	-----	-----
53219-00	64-bit Input Module		1.5	-----	-----
53220-00	16 Channel Priority Interrupt Module		1.5	-----	-----
53222-00	I/O Drive Module		1.5	-----	-----
53223-00	Utility I/O Interface		1.5	-----	-----
53224-XX	Magnetic Tape Interface		3.0	-----	-----
53227-XX	103/202 Data Set Controller		1.5	0.05	0.05
53236-XX	Dual TTY/CRT Interface		1.5	0.05	0.05
53240-XX	Cassette Interface		2.5	0.19*	0.19* (Note 2)
53513-00	Synchronous Modem Controller		1.20	0.15	0.15
53512-XX	Asynchronous Modem Multiplexer		3.0	0.30	0.30
53530-00	Moving Head Disk Controller no. 1		3.5	-----	-----
53531-00	Moving Head Disk Controller no. 2		3.0	-----	-----

NOTES
1. When two modules are interleaved, power requirements are 170% of single module requirements.
2. (*) A maximum of four cassette drives.

Top View



Bottom View

Figure 11-1. ALPHA LSI Power Supply

11.2.3 Ripple and Noise Requirements

The regulator and output filter design of each power supply must be adequate to limit ripple, noise and voltage transients to 50 mV peak-to-peak.

11.2.4 Turnon/Turnoff Overshoot

Turnon/turnoff overshoot should not exceed two percent (2%) of the nominal voltage output of each dc power supply.

11.2.5 Regulation Requirements

Each dc power supply should maintain a regulation envelope of ±2 percent of nominal output voltage from 0 to 100 percent of full rated load over the expected range of input line voltage and over a temperature range of 0°C to 50°C.

These regulation requirements must be maintained at the processor module. Remote sensing must be employed when voltage drops in the power supply wiring are of sufficient magnitude to cause voltage regulation to exceed ±2 percent when the load current is varied from no load to full load.

11.2.6 DC Power Storage

The +5Vdc, +12Vdc and -12Vdc power supplies must have sufficient storage in the regulation to insure regulated output for at least 2ms after a power failure has been detected (refer to paragraph 11.3 for details on power fail detection).

11.3 POWER MONITOR FACILITIES (Figures 11-2 and 11-3)

The Power Monitor Facilities must develop a +5H (hangpower) voltage and a ground-true Power Failure Detected signal (PFD-) for the exclusive use of the Processor. These provisions are required whether the Processor Power Fail/Restart option is used or not.

11.3.1 +5H (Hangpower) Regulator

The +5H power supply must provide auxilliary +5Vdc power for use by the Processor to assure proper startup and shutdown. The +5 H supply must be the first dc voltage to come into regulation upon application or restoration of ac line power and the last dc voltage to drop out of regulation upon loss or removal of ac line power.

The +5H supply must provide 200 mA of dc current at +5 Vdc and regulate this voltage to within ±5 percent of nominal. Ripple and noise must be within 50 mV peak-to-peak. The +5H supply must be in regualtion at all times that the +5 Vdc and ±12 Vdc supplies are above 10 percent of their specified values.

11.3.2 Power Fail Detector

The Power Fail detector must sense when the nominal ac line voltage falls below its minimum sustaining level. When this minimum sustaining level is sensed, the Power Fail detector must generate a ground-true PFD- signal for use by the Processor.

The Power Fail detector must also have a timing function that turns off the +5, +12 and -12Vdc regulators a minimum of 2 ms after PFD- goes low.

When the ac line voltage rises above the minimum sustaining level, the Power Fail detector must turn on the +5, +12 and -12Vdc regulators after allowing for a charge buildup in the storage capacitors of each regulator. The PFD- signal must remain in the ground-true state for a minimum of 2 ms after the +5, +12 and -12 Vdc regulators have reached 98 percent of their nominal values.

The PFD- signal driver must have a minimum drive capability of 20 mA dc and must be collector-ORable. The driver may be implemented with either discrete elements or with an integrated circuit. The logic levels for PFD- are as follows:

True = 0.0 to +0.45 Vdc
False = +2.4 to +5.0 Vdc

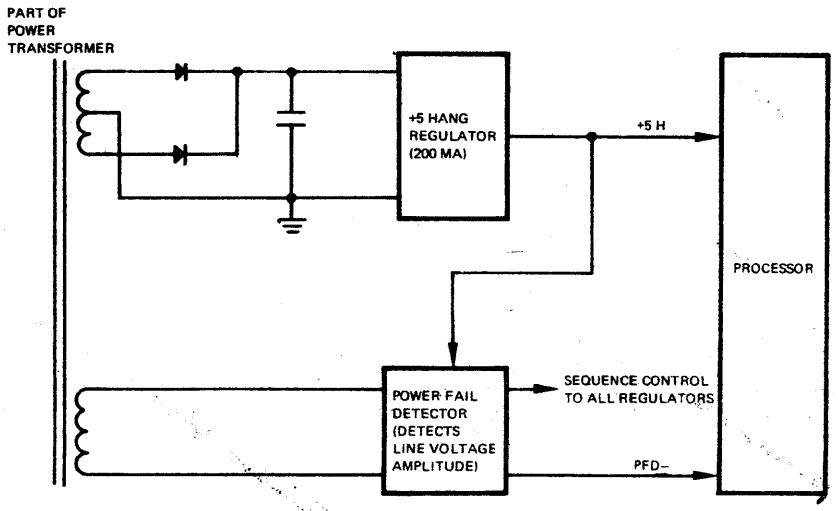


Figure 11-2. Power Monitor Block Diagram

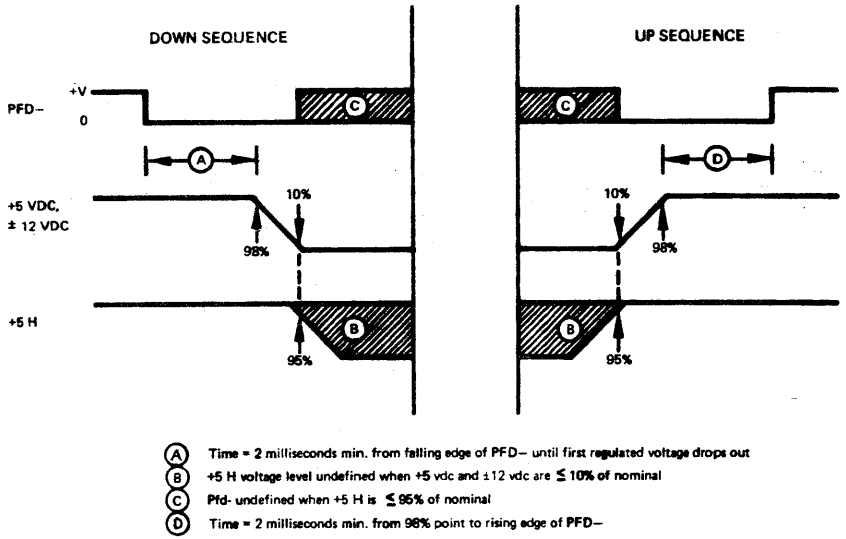


Figure 11-3. Power Monitor Timing Requirements

11.4 AC LINE SYNCHRONIZED TIMING SOURCE (OPTIONAL)

The Processor Real Time Clock (RTC) option has provisions for a timing source input which is twice the ac line frequency. The RTC option represents only one TTL load to the timing source. The timing source output must be a TTL compatible logic signal with rise and fall times of less than 50 ns. With regard to the duty cycle of the signal, the only requirement is that the signal be ground-true a minimum of 100 ns. The Processor refers to this timing signal as TTLF- (Twice the Line Frequency). The logic levels for TTLF- are as follows:

True = 0.0 to +0.45 Vdc
False = +2.4 to +5.0 Vdc

11.5 INTERCONNECTION REQUIREMENTS (Figures 11-4 and 11-5)

The user furnished power supply may be interfaced to the computer system in two ways: at the motherboard or directly at the Processor.

11.5.1 Motherboard Interface Requirements

The user may interface to the motherboard at slot F100. The motherboard distributes power and ground to all plug-in modules via the F100 connector. The F100 connector is a 36-pin connector with two rows of 18 pins. When viewed from the rear of the computer, pin 101 is to the right on the upper row of contacts. The odd numbered contacts (101 through 135) are in the upper row while the even numbered contacts are in the lower row.

When interfacing to slot F100, the user must provide a special PC board transition adaptor. A detailed drawing of this adapter, showing critical dimensions, is provided in figure 11-4. The interface pin assignments are shown in figure 11-5.

11.5.2 NAKED MINI LSI Power Connections

The user may distribute power directly to the NAKED MINI LSI computer. The Processor has two connectors, designated P1 and P2, which must be powered. Refer to table 8-2 for the appropriate power and ground pin assignments.

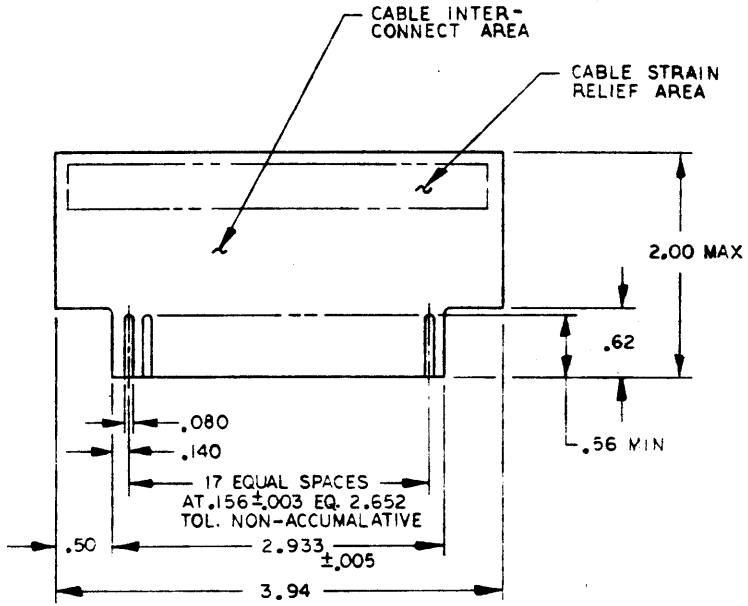


Figure 11-4. User Power supply Transition Adapter

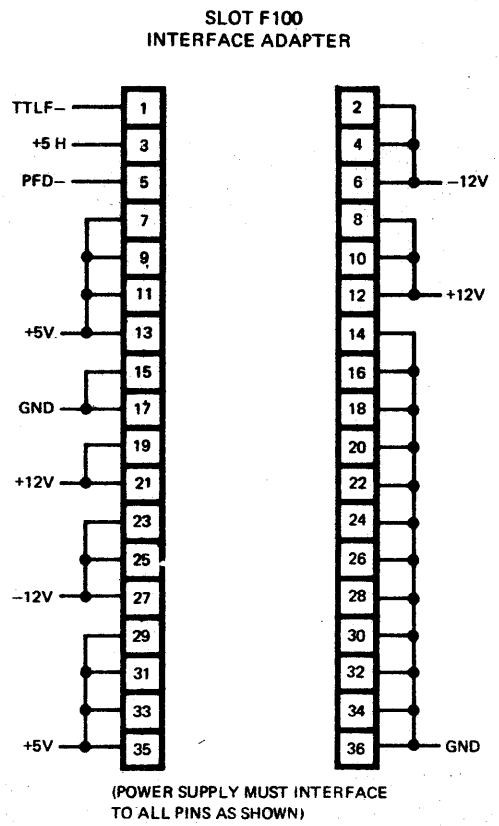


Figure 11-5. Motherboard Power Adapter Pin Assignments

Section 12

INTERFACE CONTROLLER MECHANICAL CONSIDERATIONS

12.1 INTRODUCTION

This section discusses the mechanical design of a printed circuit (PC) board which can be installed in an ALPHA LSI computer chassis.

Either full or half PC boards may be used. When half boards are used, two half boards are joined together to form a full board.

All boards use bus bars to distribute power and ground to circuits. The bus bars minimize the ground and power etch runs, leaving more space on the board for signal etched circuit routing. The bus bar design permits etched circuitry to be routed beneath the bus bar with no danger of shorting.

Fiberglass or metal stiffeners are used on all full boards to eliminate sag and provide improved structural integrity.

12.2 CHASSIS CONSTRAINTS

The computer chassis is designed to accommodate a PC board which has a width of 15 inches. All PC boards are installed in the horizontal position. When installed, the chassis provides four-way support for the PC board. The PC board guides support both sides of the board, the motherboard connectors support the front, and a board retainer supports the rear edge.

The thickness of the PC board is determined by the motherboard connectors. A typical board is .062 inch thick. The motherboard connector permits variations in thickness ranging from .054 to .071 inch.

All components, stiffeners, bus bars, etc. are mounted on one side of the board. This side of a board is referred to as the "component side" while the other side is referred to as the "solder side". Boards are always installed with the component side up.

The chassis PC board guides are spaced on .75 inch centers. The height of components on the component side of a board and the lead protrusion on the solder side of a board must be minimized to permit unimpeded airflow and easier insertion and removal of PC boards. All components should be no higher than .47 inch maximum. Lead protrusion should be held to .062 inch maximum.

The PC board guides are an integral part of the computer chassis which is metal. To prevent short circuits on a board, the user should not permit any etched circuit runs closer than .200 inch from either edge of a board.

12.3 PRINTED CIRCUIT BOARD CONSIDERATIONS (Figures 12-1 thru 12-3)

Figures 12-1 and 12-2 show the critical dimensions, hole patterns for bus bars, and stiffener and integrated circuit layout organization for a full and a half board, respectively.

The motherboard interface dimensions are extremely critical and must be adhered to rigorously.

The rear edge of the full board has room for two interface connectors. The 1.250 inch dimension from each edge is the area reserved for the board extractors (Part No. 40-06100-00). The .800 inch dimension at the center is the area reserved for the board retainer. The remaining area along the rear edge is connector area. The 6.350 inches dimension is the maximum allowable area that the mating connector can occupy. The overall length of a connector cannot exceed this dimension.

The rear edge of a half board has room for only one interface connector. A distance of 1.210 inches must be reserved for a modified board extractor (Part No. 00-00296-00). This leaves 5.080 inches of useable connector area remaining. The 5.080 dimension is the inside contact dimension of the standard 100-pin interface connector.

Half boards must provide for a board extractor at both rear corners although only one is installed depending upon which way the board is strapped to a second half board.

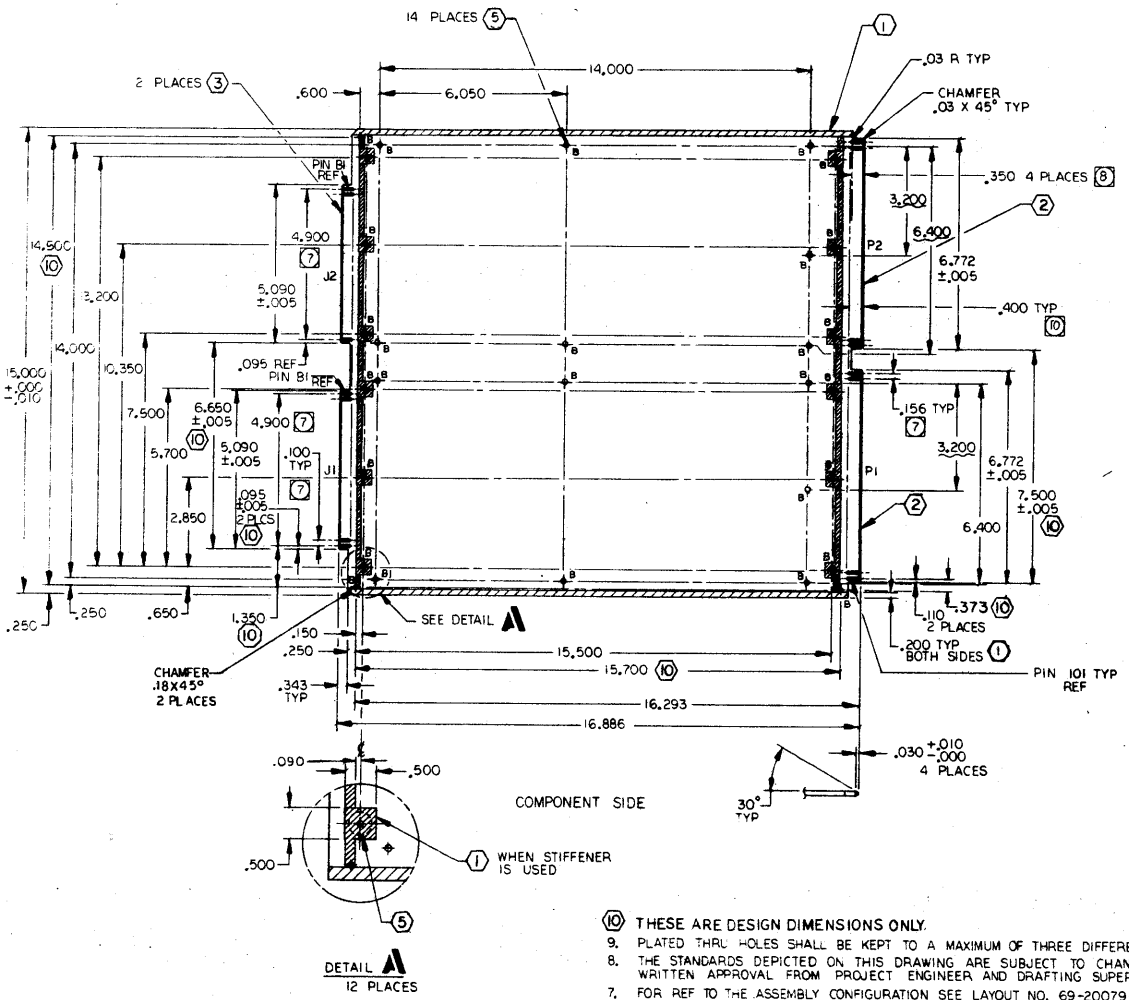
Figure 12-3 shows the standard PC board hardware. All dimensions are provided for layout planning purposes. Connector data on the motherboard connector and various rear edge interface connectors is also provided.

12.4 WIRE-WRAP BREADBOARD PC BOARD (Figure 12-4)

A wire-wrap breadboard PC board (half board configuration) is available from Computer Automation, Inc. (Part number 13234-00). This board features 72 IC sockets with wire-wrap posts, ground and power busses, and filters. The board can be useful for prototype development and checkout prior to making a formal PC board design.

12.5 FILLER BOARD PC BOARD (Figure 12-5)

A filler board PC board (half board configuration) is available from Computer Automation Inc. (Part number 10053-00). This board can be joined with a half board I/O module to form a full board as recommended in section 2, paragraph 2.2.3. The filler board does not pass the priority chains. Therefore, it must be the last board in the chain.



HOLE SCHEDULE					
HOLE	FINISHED DIMENSION	REMARKS	QTY	SYMBOL	USAGE
A	.035 +.004 -.001	PLATED THRU	ACTUAL COUNT	REF	FOR I.C.'S, BUS BARS AND COMPONENT LEADS UNDER .028 DIA MAXIMUM
B	.140 +.004 -.001	NO PLATING	3		TOOLING HOLES PART OF STD BOARD CONFIGURATION.
			12	▨	STIFFENER HOLES PART OF STD BOARD CONFIGURATION.
			14	⊕	PIGGY BACK CARD & CONNECTOR MTG HOLES NON-STANDARD.

THIS TYPE OF FORMAT TO BE USED ON ALL DETAIL FAB DRAWINGS

FOR COMPLETE INFORMATION REGARDING RECOMENDED HOLE SIZES, SEE FIGURE "F" DRM SECTION 98-10089-04

(10) THESE ARE DESIGN DIMENSIONS ONLY.

9. PLATED THRU HOLES SHALL BE KEPT TO A MAXIMUM OF THREE DIFFERENT SIZES.
8. THE STANDARDS DEPICTED ON THIS DRAWING ARE SUBJECT TO CHANGE UPON WRITTEN APPROVAL FROM PROJECT ENGINEER AND DRAFTING SUPERVISOR.
7. FOR REF TO THE ASSEMBLY CONFIGURATION SEE LAYOUT NO. 69-20079-00.
6. AREA UNDER STIFFENER (DWG'S 72-10048-00 & 72-20046-00) SHALL BE FREE OF COMPONENTS.
(5) WHEN USING STANDARD #4 HARDWARE (.312 O.D. WASHER) ETCH FREE AREA SHALL BE: .400 DIA ON SOLDER SIDE, .250 DIA ON COMPONENT SIDE. WHEN USING NON-STANDARD #4 HARDWARE (.250 O.D. WASHER) ETCH FREE AREA SHALL BE .320 DIA ON SOLDER SIDE, .250 DIA ON COMPONENT SIDE AND PROJECT ENGINEER SHALL BE CONSULTED.
4. ETCH SHALL BE NO CLOSER THAN .050 TO ANY EDGE, CUTOUT, HOLE, ETC.
(3) INTENDED TO MATE WITH CONNECTOR 17-49075-00 OR EQUIV.
(2) INTENDED TO MATE WITH CONNECTOR 17-10035-01 OR EQUIV.
(1) SHADED AREA SHALL BE FREE OF FEED THRU HOLES AND ETCH.

DESIGN INFORMATION NOTES

(10) THIS AREA TO BE FREE OF SOLDER BOTH SIDES.

9. SHEETS AS SPECIFIED BELOW (SH. 1 THRU 7) COMPRISE A COMPLETE SET OF DOCUMENTS FOR FABRICATION OF A PCB.
   - SHEET 2 PAD MASTER.
   - 3 COMPONENT SIDE A/W.
   - 4 SOLDER SIDE A/W.
   - 5 COMPONENT SIDE SILKSCREEN MASTER.
   - 6 SOLDER SIDE SOLDER MASK.
   - SHEET 7 GROUND PLANE (IF REQUIRED).

(8) CONTACT FINGER PLATING AREA.
(7) THESE DIMENSIONS ARE ESTABLISHED FROM THE ARTWORK.
6. STAMP REVISION NO., COLOR BLACK, CHARACTER HEIGHT .090 MINIMUM.
5. SILKSCREENING TO BE WHITE, COMPONENT SIDE PER SHEET 5.
4. ALL PLATED THRU HOLES TO CONFORM WITH 85-20017-00, SEC. 3.5.5.
3. FINISH: (A) SOLDER PLATE REMAINDER OF BOARD PER 85-20017-00, SEC. 3.6.2.-5.
   (B) SOLDER MASK BOTH SIDES OF BOARD PER 85-20017-00, SEC. 3.8, (DARK GREEN)
   (C) FINGERS TO BE GOLD OVER NICKEL PER 85-20017-00, SEC. 3.6.1.3 & 3.6.4
2. MATERIAL: .063 THICK COPPER-CLAD, 2 SIDES, GLASS EPOXY LAMINATE GF (FR4), 2 OZ AFTER PLATING.
1. FABRICATE PER THIS DRAWING AND C.A.I. SPECIFICATION 85-20017-00.

FABRICATION NOTES: THESE NOTES WILL APPEAR ON ALL DETAIL FAB DRAWINGS

Figure 12-1. Full Board Design Guide

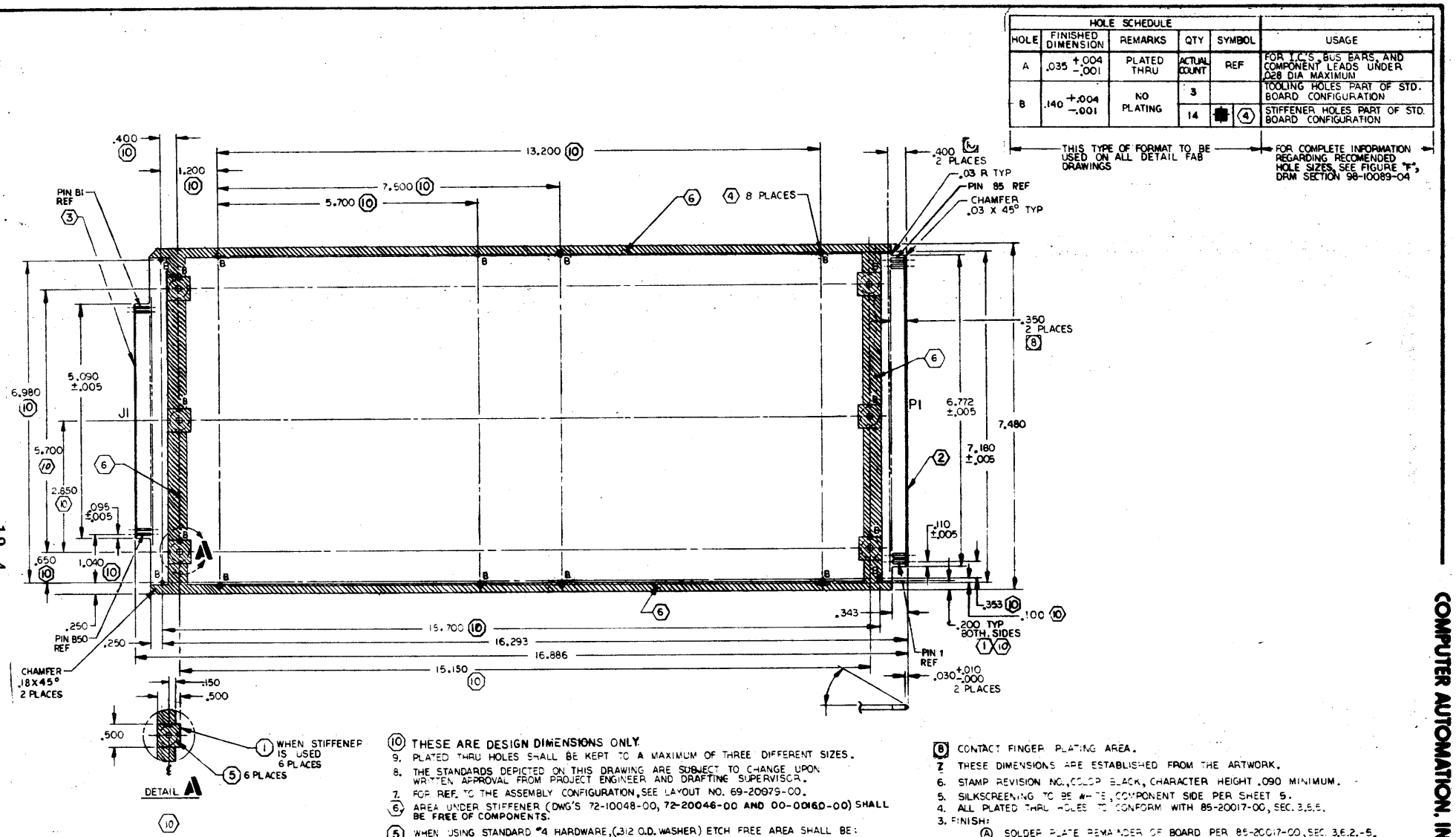
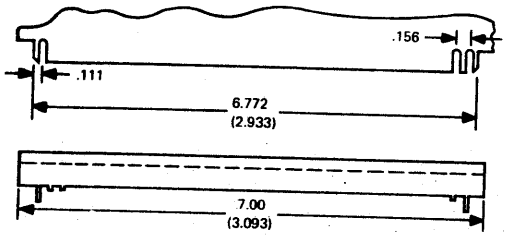


Figure 12-2. Half Board Design Guide

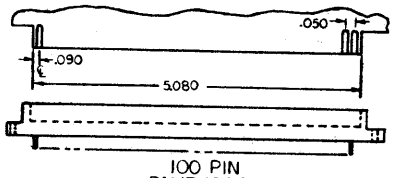
COMPUTER AUTOMATION, INC.

MOTHERBOARD CONNECTOR &
(POWER SUPPLY CONNECTOR)

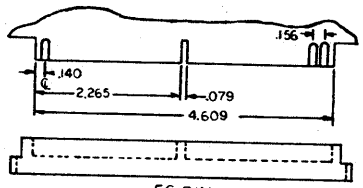


86 PIN PN 17-10035-01
(36 PIN PN 17-10035-02)

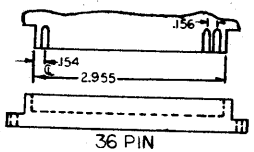
REAR EDGE CONNECTOR VARIATIONS



100 PIN
PN 17-10001-50P0

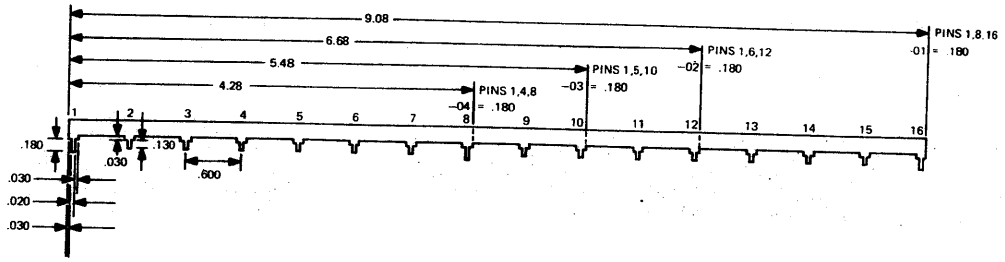


56 PIN
PN 17-10002-28P1

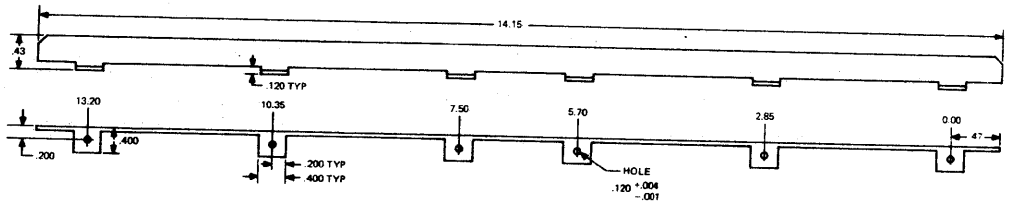


36 PIN
PN 17-10002-18P1

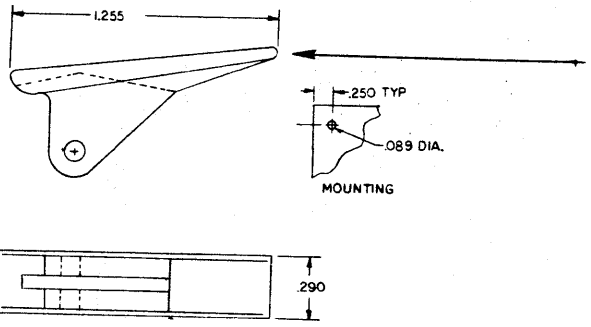
BUS BAR (PIN 72-10054-XX)



STIFFENER (PIN 72-10048-00)



FULL BOARD CARD EXTRACTOR (PN40-06100-00N0)



NOTE: The half-board card extractor (PN00-00296-00) is the same as the full-board extractor, except .130 inches of material are removed from the tip of the extractor.

Figure 12-3. Standard PC Board Hardware

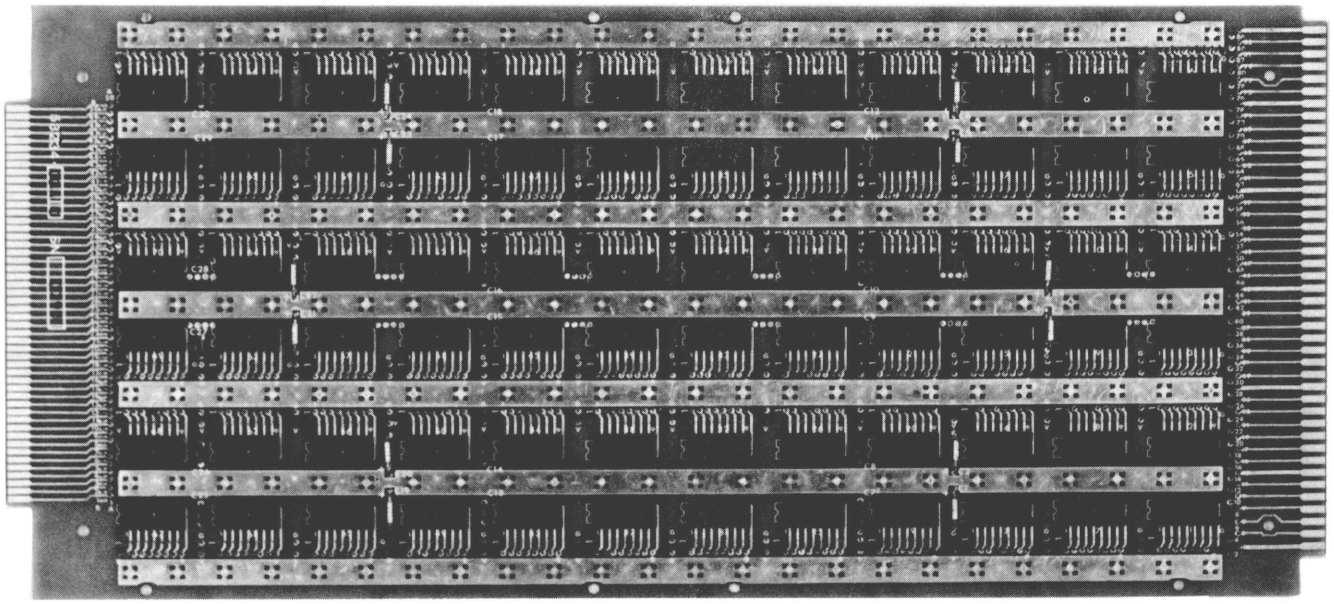


Figure 12-4. Wire-Wrap Breadboard PC Board

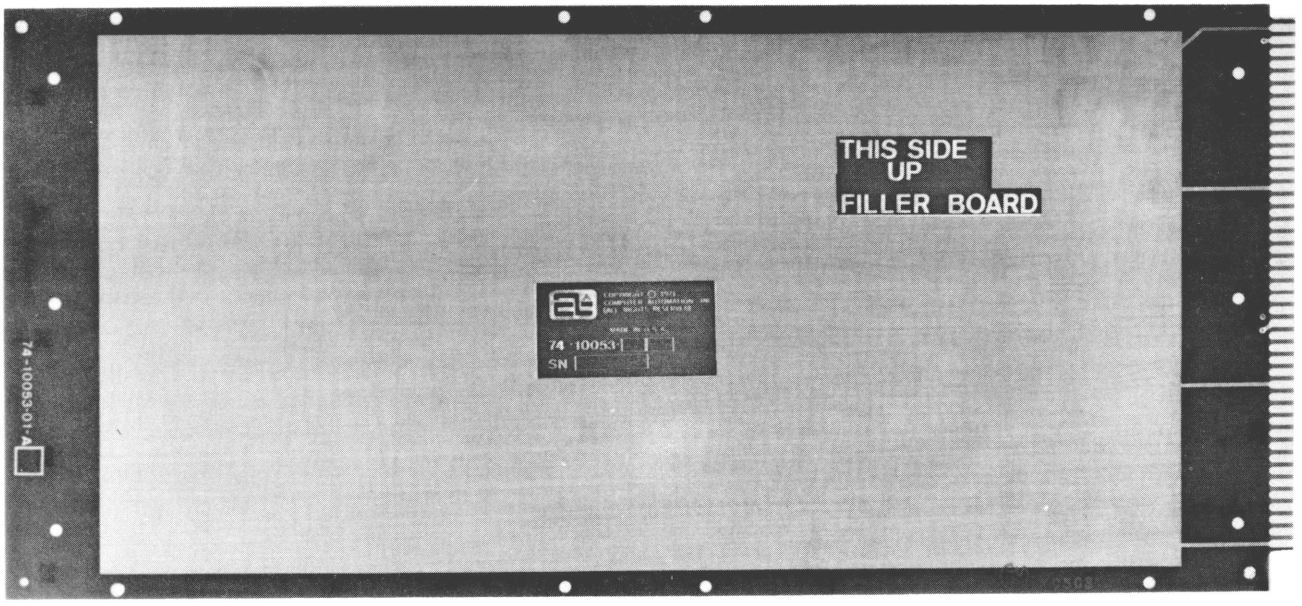


Figure 12-5. Filler Board PC Board

Appendix A

HEXADECIMAL TABLES

Tables A-1 and A-2 are quick reference conversion tables that have been included for the convenience of the user.

Table A-1. Hexadecimal-Decimal Conversions

This table is designed to facilitate conversion of positive hexadecimal integers in standard single-precision or double-precision format to decimal equivalents. The fourth and eighth digit positions therefore contain only values in the range : 0 through : 7.

	DECIMAL EQUIVALENTS							
HEXADECIMAL	DIGIT 8	DIGIT 7	DIGIT 6	DIGIT 5	DIGIT 4	DIGIT 3	DIGIT 2	DIGIT 1
1	134217728	8388608	524288	32768	4096	256	16	1
2	268435456	16777216	1048576	65536	8192	512	32	2
3	402653184	25165814	1572864	98304	12288	768	48	3
4	536870912	33554432	2097152	131072	16384	1024	64	4
5	671088640	41943040	2621440	163840	20480	1280	80	5
6	805306368	50331648	3145728	196608	24576	1536	96	6
7	939524096	28720256	3670016	229376	28672	1792	112	7
8		67108864	4194304	262144		2048	128	8
9		75497472	4718592	294912		2304	144	9
A		83886080	5242880	327680		2560	160	10
B		92274688	5767168	360448		2816	176	11
C		100663296	6291456	393216		3072	192	12
D		109051904	6815744	425984		3328	208	13
E		117440512	7340032	458752		3584	224	14
F		125829120	7864320	491520		3840	240	15

Hexadecimal to decimal conversion is accomplished by summing the decimal equivalents of the hexadecimal digits. Decimal to hexadecimal conversion involves locating the next lower decimal number and its hexadecimal equivalent and then taking the difference. Each difference is treated similarly until the entire hexadecimal number is developed.

Table A-2. 8-BIT ASCII Teletype Codes

Symbol	Hexadecimal Code	Symbol	Hexadecimal Code
@	C0	¤	A0
A	C1	!	A1
B	C2	"	A2
C	C3	#	A3
D	C4	$	A4
E	C5	%	A5
F	C6	&	A6
G	C7	'	A7
H	C8	(	A8
I	C9	)	A9
J	CA	*	AA
K	CB	+	AB
L	CC	,	AC
M	CD	-	AD
N	CE	.	AE
O	CF	/	AF
P	D0	0	B0
Q	D1	1	B1
R	D2	2	B2
S	D3	3	B3
T	D4	4	B4
U	D5	5	B5
V	D6	6	B6
W	D7	7	B7
X	D8	8	B8
Y	D9	9	B9
Z	DA	:	BA
[	DB	;	BB
\	DC	<	BC
]	DD	=	BD
↑	DE	>	BE
←	DF	?	BF
NULL	00	CR	8D
BELL	87	LF	8A
		RUBOUT	FF

Appendix B

RECOMMENDED DEVICE AND INTERRUPT ADDRESSES

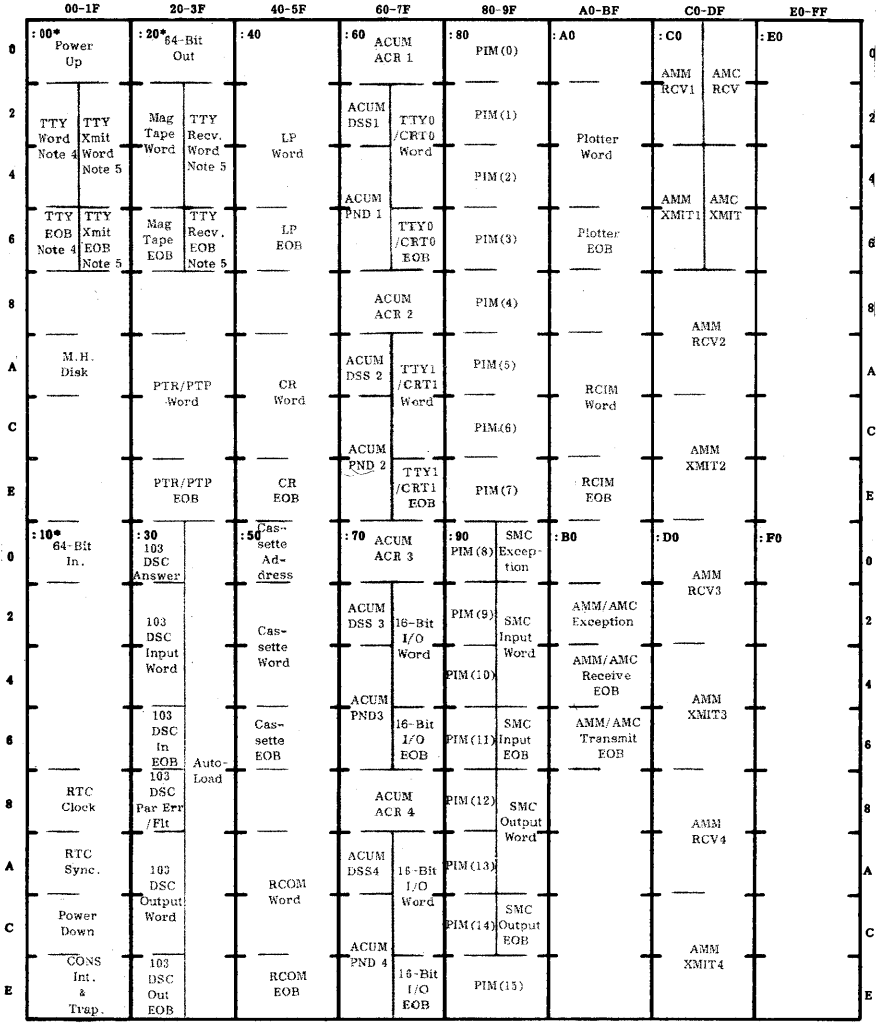
Table B-1 and B-2 list recommended Device and Interrupt Addresses to prevent possible conflict during future expansion to other I/O modules.

Table B-1. Recommended Device Addresses

DEVICE	DEVICE ADDRESSES (HEXADECIMAL) STANDARD	ACTUAL
Refer to Table B-3	00	
	01	
Dual TTY/CRT (TTY1/CRT1)	02	
Dual TTY/CRT (TTY0/CRT0)	03	
Line Printer (LP)	04	
Card Reader (CR)	05	
Paper Tape Punch (PTP)	06(17)	
Paper Tape Reader (PTR)	06	
Processor TTY* (TTY)	07	
Real Time Clock* (RTC)	08	
Magnetic Tape (Mag Tape)	09	
	0A	
	0B	
Automatic Calling Unit Mux (ACUM)	0C	
Synchronous Modem Controller (SMC)	0D	
Asynchronous Modem Multiplexer (AMM)	0E	
Disc	0F	
Cassette	10	
Floppy Disc	11	
16-Bit I/O (A/D System)	12	
	13	
Plotter	14	
	15	
32-Bit Relay In (RCIM)	16	
Punch Alternate	17	
16-Bit Input/Output (16-Bit I/O)	18	
64-Bit Input (64-Bit In)	19	
64-Bit Output (64-Bit Out)	1A	
Priority Interrupt Module (PIM)	1B	
32-Bit Relay Out (RCOM)	1C	
103 Data Set Controller (103 DSC)	1D	
Memory Bank Controller	1E	
	1F	

* Processor mounted options. Device Address non-alterable.
( ) Indicates suggested alternate.

Table B-2. Recommended Interrupt Address Map



1. *Address cannot be changed.
2. EOB = End-of-Block
3. Split Address blocks and/or a/ indicate that this is the standard I/O vector for more than one device. Only one device may use a vector in any given system.
4. Half Duplex
5. Full Duplex

Table B-3. Device Address 0 Command Summary

FUNCTION CODE	SELECT COMMANDS	SENSE COMMANDS	INPUT COMMANDS	OUTPUT COMMANDS
0	Autoload off	Autoload Option Installed	SIA (:5800), SIX (:5A00)	SOA (:6C00), SOX (:6E00)
1	Enable Autoload ROM	Real-Time Clock Option Installed	ISA (:5801) ISX (:5A01)	SIN 0 (:6801)
2	PFE (:4002)			SIN 1 (:6802)
3	PFD (:4003)			SIN 2 (:6803)
4	OCA (:4404) OCX (:4604)	TTY/CRT/Modem Option Installed	ICA (:5804) ICX (:5A04)	SIN 3 (:6804)
5	CIE (:4005)			SIN 4 (:6805)
6	CID (:4006)			SIN 5 (:6806)
7	TRP (:4007)			SIN 6 (:6807)

Appendix C

INSTRUCTION SET BY CLASS

This appendix contains the ALPHA LSI instruction set in class order. For each instruction, reference is made to one of the assembler syntax formats listed below.

[LABEL] OP-CODE [* |@| *@] EXPRESSION [COMMENTS]

No Operator = Direct Addressing
* = Indirect Addressing (multi-level)
@ = Indexed Addressing
*@ = Indirect Post-Indexed Addressing (multi-level)

Figure C-1. Class 1 - Single-Word Memory Reference Instruction Format

[LABEL] OP-CODE [*] EXPRESSION 1 [,EXPRESSION 2] [COMMENTS]

No Operator = Direct Addressing
* = Indirect Addressing (multi-level)
EXPRESSION 1 represents an address to be stored in the second word of the instruction.
EXPRESSION 2 is an optional absolute instruction count in the range 0 through 31 for NRM.

Figure C-2. Class 2 - Double-Word Memory Reference Instruction Format

[LABEL] OP-CODE OPERAND [,AM] [COMMENTS]

AM = No Operator = Direct access
- = PUSH (stack pointer decremented)
+ = POP (stack pointer incremented)
@ = Indexed (single level)

Figure C-3. Class 3 - Stack Instruction Format (LSI-2 only)

[LABEL] OP-CODE EXPRESSION [COMMENTS]

EXPRESSION must be absolute and in the range : 0 through : FF. This format is also used by the STOP and SCM instructions.

Figure C-4. Class 4 - Byte Immediate Instruction Format

[LABEL] OP-CODE EXPRESSION [COMMENTS]

EXPRESSION must represent a location within -63 through +64 words.

Figure C-5. Class 5 - Conditional Jump Instruction Format

[LABEL] OP-CODE EXPRESSION [COMMENTS]

EXPRESSION must be absolute and in the range 1 through 8 (single register) or 1 through 16 (double register). This format is also used by the SIN instruction with an upper range limit of 6.

Figure C-6. Class 6 - Register Shift Instruction Format

[LABEL] OP-CODE [EXPRESSION] [COMMENTS]

EXPRESSION: there are no expressions in the operand field, except for BAO and BXO instructions, where it must be value in the range 0 through 15.

Figure C-7. Class 7 - Register Change and Control Instruction Format

[LABEL] OP-CODE EXPRESSION 1 [,EXPRESSION 2] [COMMENTS]

Both EXPRESSION 1 and expression 2 must be absolute.
If EXPRESSION 2 is present, EXPRESSION 1 must be in range : 0 through : 1F.
If EXPRESSION 2 is not present, EXPRESSION 1 must be in the range : 0 through : FF.

Figure C-8. Class 8 - Input/Output Instruction Format

[LABEL]	[JOC]	EXPRESSION 1[,EXPRESSION 2] [COMMENTS]
		EXPRESSION 1 must be absolute and in the range :0 through :3F. EXPRESSION 2 must represent a location within -63 through +64 words.

Figure C-9. Class 9 - JOC Jump-On-Condition Instruction Format

INSTRUCTION SET BY CLASS

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
MEMORY REFERENCE (Class 1)			
Arithmetic			
ADD	Add to A Register	8800	4-12
ADDB	Add Byte to A	8800	4-12
SUB	Subtract from A Register	9000	4-12
SUBB	Subtract Byte from A	9000	4-12
Logical			
AND	AND to A	8000	4-12
ANDB	AND Byte with A	8000	4-12
IOR	Inclusive OR to A	A000	4-12
IORB	Inclusive OR Byte with A	A000	4-12
XOR	Exclusive OR to A	A800	4-12
XORB	Exclusive OR Byte with A	A800	4-13
Data Transfer			
LDA	Load A	B000	4-13
LDAB	Load A with Byte	B000	4-13
LDX	Load X	E000	4-13
LDXB	Load X with Byte	E000	4-13
STA	Store A	9800	4-13
STAB	Store Byte from A	9800	4-13
STX	Store X	E800	4-13
STXB	Store Byte from X	E800	4-13
EMA	Exchange A and Memory	B800	4-13
EMAB	Exchange A and Memory Byte	B800	4-13

INSTRUCTION SET BY CLASS (Cont'd)

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
Program Transfer			
JMP	Unconditional Jump	F000	4-14
JST	Jump and Store P Counter	F800	4-14
IMS	Increment Memory, Skip on Zero	D800	4-14
SCM	Scan Memory	CD00	4-14
SCMB	Scan Memory Byte	CD00	4-15
CMS	Compare A with Memory, Skip	D000	4-13
CMSB	Compare A with Memory Byte, Skip	D000	4-14
DOUBLE WORD MEMORY REFERENCE (Class 2)			
DVD	Divide	1970	4-16
MPY	Multiply and Add	1960	4-16
NRM	Normalize A and X	1940	4-17
STACK CLASS (Class 3) (LSI-2 only)			
Arithmetic			
ADDS	Add Stack Element to A	1438	4-21
SUBS	Subtract Stack Element from A	1458	4-21
Logical			
ANDS	AND Stack Element to A	1418	4-21
IORS	Inclusive OR Stack Element to A	1498	4-21
XORS	Exclusive OR Stack Element to A	14B8	4-21
Data Transfer			
EMAS	Exchange Stack Element and A	14F8	4-21
LDAS	Load Stack Element into A	14D8	4-21
LDXS	Load Stack Element into X	1698	4-21
STAS	Store A in Stack Element	1478	4-21
STXS	Store X in Stack Element	16B8	4-21
Program Transfer			
CMSS	Compare Stack Element to A and Skip if High or Equal	1658	4-22
IMSS	Increment Stack Element and Skip on Zero Result	1678	4-22

INSTRUCTION SET BY CLASS (Cont'd)

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
JMPS	Jump Unconditional to Stack Element	16D8	4-22
JSTS	Jump and Store to Stack Element	16F8	4-22
Stack Control			
SLAS	Stack Location to A	1618	4-22
BYTE IMMEDIATE (Class 4)			
AAI	Add to A Register Immediate	0B00	4-23
AXI	Add to X Register Immediate	C200	4-23
SAI	Subtract from A Register Immediate	0D00	4-23
SXI	Subtract from X Register Immediate	C300	4-23
CAI	Compare to A Immediate, Skip if Not Equal	C000	4-23
CXI	Compare to X Immediate, Skip of Not Equal	C100	4-23
LAP	Load A Positive Immediate	C600	4-23
LXP	Load X Positive Immediate	C400	4-23
LAM	Load A Minus Immediate	C700	4-23
LXM	Load X Minus Immediate	C500	4-23
CONDITIONAL JUMP (Class 5 or 9)			
Microcoded (Class 9)			
JOC	Jump on Condition Specified	2000	4-24
Arithmetic (Class 5)			
JAG	Jump if A Greater than Zero	3180	4-25
JAP	Jump if A Positive	3080	4-25
JAZ	Jump if A Zero	2100	4-25
JAN	Jump if A Not Zero	3100	4-25
JAL	Jump if A Less Than or Equal to Zero	2180	4-25
JAM	Jump if A Minus	2080	4-25
JXZ	Jump if X Zero	2800	4-26
JXN	Jump if X Not Zero	3800	4-26
Control (Class 5)			
JSS	Jump if SENSE Indicator ON	3400	4-26
JSR	Jump if SENSE Indicator OFF	2400	4-26
JOS	Jump if OV Set	2200	4-26
JOR	Jump if OV Reset	3200	4-26

INSTRUCTION SET BY CLASS (Cont'd)

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
SHIFT CLASS (Class 6)			
Single Register			
Arithmetic			
ARA	Arithmetic Right A	10D0	4-27
ARX	Arithmetic Right X	10A8	4-27
ALA	Arithmetic Left A	1050	4-27
ALX	Arithmetic Left X	1028	4-27
Logical			
LRA	Logical Right A	13D0	4-28
LRX	Logical Right X	13A8	4-28
LLA	Logical Left A	1350	4-28
LLX	Logical Left X	1328	4-28
Rotate			
RRA	Rotate Right A with OV	11D0	4-29
RRX	Rotate Right X with OV	11A8	4-29
RLA	Rotate Left A with OV	1150	4-29
RLX	Rotate Left X with OV	1128	4-29
Double Register			
Logical			
LLL	Long Logical Left	1B00	4-30
LLR	Long Logical Right	1B80	4-30
Rotate			
LRL	Long Rotate Left with OV	1900	4-31
LRR	Long Rotate Right with OV	1980	4-31
REGISTER CHANGE (Class 7)			
Accumulator			
ZAR	Zero A Register	0110	4-31
ARP	Set A Register to Positive 1	0350	4-31

INSTRUCTION SET BY CLASS (Cont'd)

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
ARM	Set A Register to Minus 1	0010	4-31
CAR	Complement (1's) A Register	0210	4-31
NAR	Negate A Register	0310	4-31
IAR	Increment A Register	0150	4-31
DAR	Decrement A Register	00D0	4-31
Index			
ZXR	Zero X Register	0108	4-32
XRP	Set X Register to Positive 1	0528	4-32
XRM	Set X Register to Minus 1	0008	4-32
CXR	Complement (1's) X Register	0408	4-32
NXR	Negate X Register	0508	4-32
IXR	Increment X Register	0128	4-32
DXR	Decrement X Register	00A8	4-32
Overflow			
SOV	Set Overflow	1400	4-32
ROV	Reset Overflow	1200	4-32
COV	Complement Overflow	1600	4-32
SAO	Sign of A to OV	1340	4-32
SXO	Sign of X to OV	1320	4-32
LAO	Least Significant Bit of A to OV	13C0	4-32
LXO	Least Significant Bit of X to OV	13A0	4-32
BAO	Bit of A to OV	1340	4-32
BXO	Bit of X to OV	1320	4-32
Multi-Register			
ZAX	Zero A and X Register	0118	4-33
AXP	Set A and X Registers to Positive 1	0358	4-33
AXM	Set A and X Registers to Minus 1	0018	4-33
TAX	Transfer A to X	0048	4-33
TXA	Transfer X to Z	0030	4-33
EAX	Exchange A and X	0428	4-33
ANA	AND of A and X to A	0070	4-33
ANX	AND of A and X to X	0068	4-33
NRA	NOR of A and X to A	0610	4-33
NRX	NOR of A and X to X	0608	4-33
CAX	Complement A (1's) and put in X	0208	4-33
CXA	Complement X (1's) and put in A	0410	4-33
NAX	Negate A and put in X	0308	4-33
NXA	Negate X and put in A	0510	4-33

INSTRUCTION SET BY CLASS (Cont'd)

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
IAX	Increment A and put in X	0148	4-33
IXA	Increment X and put in A	0130	4-34
IPX	Increment P and put in X	0090	4-34
DAX	Decrement A and put in X	00C8	4-34
DXA	Decrement X and put in A	00B0	4-34
Extended Multi-Register (LSI-2 only)			
BCA	Bit Clear A	06CA	4-34
BCX	Bit Clear X	06C8	4-34
BSA	Bit Set A	068A	4-34
BSX	Bit Set X	0688	4-34
EIX	Execute Instruction Pointed to By X	0218	4-34
Console Register			
IAH	Input Console Data Register to A and Halt	1C05	4-35
ICA	Input Console Data Register to A	5804	4-35
ICX	Input Console Data Register to X	5A04	4-35
IIH	Input Console Data Register to I and Halt	1C11	4-35
IMH	Input Console Data Register to Memory and Halt	1C03	4-35
IPH	Input Console Data Register to P and Halt	1C21	4-35
ISA	Input Console Sense Register to A	5801	4-35
ISX	Input Console Sense Register to X	5A01	4-35
IXH	Input Console Data Register to X and Halt	1C09	4-35
OAH	Output A to Console Data Register and Halt	1C04	4-35
OCA	Output A to Console Data Register	4404	4-36
OCX	Output X to Console Data Register	4604	4-36
OLH	Output Location to Console Data Register and Halt	1C10	4-36
OMH	Output Memory to Console Data Register and Halt	1C02	4-36
OPH	Output P to Console Data Register and Halt	1C20	4-36
OXH	Output X to Console Data Register and Halt	1C08	4-35
Processor			
NOP	No operation	0000	4-36
HLT	Halt	0800	4-36
STOP	Halt with Operand	0800	4-36
WAIT	Wait for Interrupts	F600	4-37

INSTRUCTION SET BY CLASS (Cont'd)

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
Mode Control			
SBM	Set Byte Operand Mode	0E00	4-37
SWM	Set Word Operand Mode	0F00	4-37
Status			
SIN	Status Inhibit	6800	4-37
SIA	Status Input to A	5800	4-38
SIX	Status Input to X	5A00	4-38
SOA	Status Output from A	6C00	4-38
SOX	Status Output from X	6E00	4-38
Interrupts			
EIN	Enable Interrupts	0A00	4-38
DIN	Disable Interrupts	0C00	4-38
CIE	Console Interrupt Enable	4005	4-38
CID	Console Interrupt Disable	4006	4-38
PFE	Power Fail Interrupt Enable	4002	4-38
PFD	Power Fail Interrupt Disable	4003	4-38
TRP	Trap	4007	4-39
INPUT/OUTPUT (Class 8)			
Control			
SEL	Select	4000	4-40
SEA	Select and Present A	4400	4-40
SEX	Select and Present X	4600	4-40
SEN	Sense and Skip on Response	4900	4-40
SSN	Sense and Skip on No Response	4800	4-40
Unconditional Word			
INA	Input Word to A	5800	4-41
INAM	Input Word to A Masked	5C00	4-41
INX	Input Word to X	5A00	4-41
INXM	Input Word to X Masked	5E00	4-41
OTA	Output A	6C00	4-41
OTX	Output X	6E00	4-41
OTZ	Output Zero's	6800	4-41

INSTRUCTION SET BY CLASS (Cont'd)

Instruction Mnemonic	Description	Instruction Skeleton in Hex	Page
Conditional Word			
RDA	Read Word to A	5900	4-41
RDAM	Read Word to A Masked	5D00	4-41
RDX	Read Word to X	5B00	4-41
RDXM	Read Word to X Masked	5F00	4-42
WRA	Write A	6D00	4-42
WRX	Write X	6F00	4-42
WRZ	Write Zero's	6900	4-42
Unconditional Byte			
IBA	Input Byte to A	7800	4-42
IBAM	Input Byte to A Masked	7C00	4-42
IBX	Input Byte to X	7A00	4-43
IBXM	Input Byte to X Masked	7E00	4-43
Conditional Byte			
RBA	Read Byte to A	7900	4-43
RBAM	Read Byte to A Masked	7D00	4-43
RBX	Read Byte to X	7B00	4-43
RBXM	Read Byte to X Masked	7F00	4-43
Block			
BIN	Input Block to Memory	7100	4-44
BOT	Output Block from Memory	7500	4-45
Automatic			
AIN	Automatic Input Word to Memory	5000	4-47
AOT	Automatic Output Word from Memory	6000	4-47
AIB	Automatic Input Byte to Memory	5400	4-47
AOB	Automatic Output Byte from Memory	6400	4-47

Appendix D

INSTRUCTION SET IN ALPHABETICAL ORDER

This appendix contains the ALPHA LSI instruction set in alphabetical order by instruction mnemonic. Those instructions which contain variable fields have been appended with an asterisk (*). Those applying to LSI-2 only have been prefixed with an asterisk.

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
AAI	0B00*	Add to A Immediate; Direct	4-23
ADD	8800*	Add to A; Direct, Scratchpad	4-12
ADD	8900*	Add to A; Indirect, AP in Scratchpad	4-12
ADD	8A00*	Add to A; Direct, Relative to P Forward	4-12
ADD	8B00*	Add to A; Indirect, AP Relative to P Forward	4-12
ADD	8C00*	Add to A; Direct, Indexed	4-12
ADD	8D00*	Add to A; Indirect, Indexed, AP in Scratchpad	4-12
ADD	8E00*	Add to A; Direct, Relative to P Backward	4-12
ADD	8F00*	Add to A; Indirect, AP Relative to P Backward	4-12
ADDB	8800*	Add Byte; Direct, Scratchpad	4-12
ADDB	8900*	Add Byte; Indirect, AP in Scratchpad	4-12
ADDB	8A00*	Add Byte 0; Direct, Relative to P Forward	4-12
ADDB	8B00*	Add Byte; Indirect, AP Relative to P Forward	4-12
ADDB	8C00*	Add Byte; Direct, Indexed	4-12
ADDB	8D00*	Add Byte; Indirect, Indexed, AP in Scratchpad	4-12
ADDB	8E00*	Add Byte 1; Direct Relative to P Forward	4-12
ADDB	8F00*	Add Byte; Indirect, AP Relative to P Backward	4-12

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
*ADDS	1438	Add Stack Element to A; Direct	4-21
*ADDS	1439	Add Stack Eleement to A: Indexed	4-21
*ADDS	143A	Add Stack Element to A; Auto-Postincrement	4-21
*ADDS	143B	Add Stack Element to A; Auto-Predecrement	4-21
AIB	5400*	Automatic Input Byte to Memory	4-47
AIN	5000*	Automatic Input Word to Memory	4-47
ALA	1050*	Arithmetic Shift A Left	4-27
ALX	1028*	Arithmetic Shift X Left	4-27
ANA	0070	AND of A and X to A	4-33
AND	8000*	AND to A; Direct, Scratchpad	4-12
AND	8100*	AND to A; Indirect, AP in Scratchpad	4-12
AND	8200*	AND to A; Direct, Relative to P Forward	4-12
AND	8300*	AND to A; Indirect, AP Relative to P Forward	4-12
AND	8400*	AND to A; Direct, Indexed	4-12
AND	8500*	AND to A; Indirect, Indexed, AP in Scratchpad	4-12
AND	8600*	AND to A; Direct, Relative to P Backward	4-12
AND	8700*	AND to A; Indirect, AP Relative to P Backward	4-12
ANDB	8000*	AND Byte to A; Direct, Scratchpad	4-12
ANDB	8100*	AND Byte to A; Indirect, AP in Scratchpad	4-12
ANDB	8200*	AND Byte 0 to A; Direct, Relative to P Forward	4-12
ANDB	8300*	AND Byte to A; Indirect, AP Relative to P Forward	4-12
ANDB	8400*	AND Byte to A; Direct, Indexed	4-12
ANDB	8500*	AND Byte to A; Indirect, Indexed, AP in Scratchpad	4-12

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
ANDB	8600	AND Byte 1 to A; Direct, Relative to P Forward	4-12
ANDB	8700*	AND Byte to A; Indirect, AP Relative to P Backward	4-12
*ANDS	1418	AND Stack Element to A; Direct	4-21
*ANDS	1419	AND Stack Element to A; Indexed	4-21
*ANDS	141A	AND Stack Element to A; Auto-Postincrement	4-21
*ANDS	141B	AND Stack Element to A; Auto-Predecrement	4-21
ANX	0068	AND of A and X to X	4-33
AOB	6400*	Automatic Output Byte from Memory	4-47
AOT	6000*	Automatic Output Word from Memory	4-47
ARA	10D0*	Arithmetic Shift A Right	4-27
ARM	0010	Set A to Minus 1	4-31
ARP	0350	Set A to Plus 1	4-31
ARX	10A8*	Arithmetic Shift X Right	4-27
AXI	C200*	Add to X Immediate	4-23
AXM	0018	Set A and X to Minus 1	4-33
AXP	0358	Set A and X to Plus 1	4-33
BAO	1340*	Bit of A to Overflow	4-32
*BCA	06CA	Bit Clear A	4-34
*BCX	06C8	Bit Clear X	4-34
BIN	7100*	Block Input to Memory	4-44
BOT	7500*	Block Output from Memory	4-45
*BSA	0C8A	Bit Set A	4-34

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
*BSX	0688	Bit Set X	4-34
BXO	1320*	Bit of X to Overflow	4-32
CAI	C000*	Compare to A Immediate, Skip if Unequal	4-23
CAR	0210	Complement A (1's)	4-31
CAX	0208	Complement A (1's) and Put in X	4-33
CID	4006	Console Interrupt Disable	4-38
CIE	4005	Console Interrupt Enable	4-38
CMS	D000*	Compare Memory to A and Skip if High or Equal; Direct, Scratchpad	4-13
CMS	D100*	Compare Memory to A and Skip if High or Equal; Indirect, AP in Scratchpad	4-13
CMS	D200*	Compare Memory to A and Skip if High or Equal: Direct, Relative to P Forward	4-13
CMS	D300*	Compare Memory to A and Skip if High or Equal; Indirect, AP Relative to P Forward	4-13
CMS	D400*	Compare Memory to A and Skip if High or Equal; Direct, Indexed	4-13
CMS	D500*	Compare Memory to A and Skip if High or Equal; Indirect, Indexed, AP in Scratchpad	4-13
CMS	D600*	Compare Memory to A and Skip if High or Equal; Direct, Relative to P Backward	4-13
CMS	D700*	Compare Memory to A and Skip if High or Equal; Indirect, Relative to P Backward	4-13
CMSB	D000*	Compare Byte and Skip if High or Equal; Direct, Scratchpad	4-14
CMSB	D100*	Compare Byte and Skip if High or Equal: Indirect, AP in Scratchpad	4-14

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
CMSB	D200*	Compare Byte 0 and Skip if High or Equal; Direct, Relative to P Forward	4-14
CMSB	D300*	Compare Byte and Skip if High or Equal; Indirect, AP Relative to P Forward	4-14
CMSB	D400*	Compare Byte and Skip if High or Equal; Direct, Indexed	4-14
CMSB	D500*	Compare Byte and Skip if High or Equal; Indirect, Indexed, AP in Scratchpad	4-14
CMSB	D600*	Compare Byte 1 and Skip If High or Equal; Direct, Relative to P Forward	4-14
CMSB	D700*	Compare Byte and Skip if High or Equal; Indirect, AP Relative to P Backward	4-14
*CMSS	1658	Compare Stack Element to A and Skip if High or Equal; Direct	4-22
*CMSS	1659	Compare Stack Element to A and Skip if High or Equal; Indexed	4-22
*CMSS	165A	Compare Stack Element to A and Skip if High or Equal; Auto-Postincrement	4-22
*CMSS	165B	Compare Stack Element to A and Skip if High or Equal; Auto-Predecrement	4-22
COV	1600	Complement Overflow	4-32
CXA	0410	Complement X (1's) and Put in A	4-33
CXI	C100*	Compare to X Immediate, Skip if Unequal	4-23
CXR	0408	Complement X (1's)	4-32
DAR	00D0	Decrement A	4-31
DAX	00C8	Decrement A and Put in X	4-34
DIN	0C00	Disable Interrupts	4-38
DVD	1970*	Divide	4-16

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
DXA	00B0	Decrement X and Put in A	4-34
DXR	00A8	Decrement X	4-32
EAX	0428	Exchange A and X	4-33
EIN	0A00	Enable Interrupts	4-38
*EIX	0218	Execute Instruction Pointed to by X	4-34
EMA	B800*	Exchange Memory and A; Direct, Scratchpad	4-13
EMA	B900*	Exchange Memory and A; Indirect, AP in Scratchpad	4-13
EMA	BA00*	Exchange Memory and A; Direct, Relative to P Forward	4-13
EMA	BB00*	Exchange Memory and A; Indirect, AP Relative to P Forward	4-13
EMA	BC00*	Exchange Memory and A; Direct, Indexed	4-13
EMA	BD00*	Exchange Memory and A; Indirect, Indexed, AP in Scratchpad	4-13
EMA	BE00*	Exchange Memory and A; Direct, Relative to P Backward	4-13
EMA	BF00*	Exchange Memory and A: Indirect, AP Relative to P Backward	4-13
EMAB	B800*	Exchange Memory Byte and A; Direct, Scratchpad	4-13
EMAB	B900*	Exchange Memory Byte and A; Indirect, AP in Scratchpad	4-13
EMAB	BA00*	Exchange Memory Byte 0 and A; Direct, Relative to P Forward	4-13
EMAB	BB00*	Exchange Memory Byte and A; Indirect, AP Relative to P Forward	4-13
EMAB	BC00*	Exchange Memory Byte and A; Direct, Indexed	4-13

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
EMAB	BD00*	Exchange Memory Byte and A; Indirect, Indexed, AP in Scratchpad	4-13
EMAB	BE00*	Exchange Memory Byte 1 and A; Direct, Relative to P Forward	4-13
EMAB	BF00*	Exchange Memory Byte and A; Indirect, AP Relative to P Backward	4-13
*EMAS	14F8	Exchange Stack Element and A; Direct	4-21
*EMAS	14F9	Exchange Stack Element and A; Indexed	4-21
*EMAS	14FA	Exchange Stack Element and A; Auto-Postincrement	4-21
*EMAS	14FB	Exchange Stack Element and A; Auto-Predecrement	4-21
HLT	0800	Halt	4-36
IAH	1C05	Input Console Data Register to A and Halt	4-35
IAR	0150	Increment A	4-31
IAX	0148	Increment A and Put in X	4-33
IBA	7800*	Input Byte to A (Unconditionally)	4-42
IBAM	7C00*	Input Byte to A, Masked (Unconditionally)	4-42
IBX	7A00*	Input Byte to X (Unconditionally)	4-43
IBXM	7E00*	Input Byte to X, Masked (Unconditionally)	4-43
ICA	5804	Input Console Data Register to A	4-35
ICX	5A04	Input Console Data Register to X	4-35
IIH	1C11	Input Console Data Register to I and Halt	4-35
IMH	1C03	Input Console Data Register to Memory and Halt	4-35
IMS	D800*	Increment Memory and Skip on Zero Result; Direct, Scratchpad	4-14

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
IMS	D900*	Increment Memory and Skip on Zero Result; Indirect, AP in Scratchpad	4-14
IMS	DA00*	Increment Memory and Skip on Zero Result; Direct, Relative to P Forward	4-14
IMS	DB00*	Increment Memory and Skip on Zero result; Indirect, AP Relative to P Forward	4-14
IMS	DC00*	Increment Memory and Skip on Zero Result; Direct, Indexed	4-14
IMS	DD00*	Increment Memory and Skip on Zero Result; Indirect, Indexed, AP in Scratchpad	4-14
IMS	DE00*	Increment Memory and Skip on Zero Result; Direct, Relative to P Backward	4-14
IMS	DF00*	Increment Memory and Skip on Zero Result; Indirect, AP Relative to P Backward	4-14
*IMSS	1678	Increment Stack Element and Skip on Zero; Direct	4-22
*IMSS	1679	Increment Stack Element and Skip on Zero; Indexed	4-22
*IMSS	167A	Increment Stack Element and Skip on Zero; Auto-Postincrement	4-22
*IMSS	167B	Increment Stack Element and Skip on Zero; Auto-Predecrement	4-22
INA	5800*	Input Word to A (Unconditionally)	4-41
INAM	5C00*	Input Word to A, Masked (Unconditionally)	4-41
INX	5A00*	Input Word to X (Unconditionally)	4-41
INXM	5E00*	Input Word to X, Masked (Unconditionally)	4-41
IOR	A000*	Inclusive OR to A; Direct, Scratchpad	4-12
IOR	A100*	Inclusive OR to A; Indirect, AP in Scratchpad	4-12
IOR	A200*	Inclusive OR to A; Direct, Relative to P Forward	4-12

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
IOR	A300*	Inclusive OR to A; Indirect, AP Relative to P Forward	4-12
IOR	A400*	Inclusive OR to A; Direct, Indexed	4-12
IOR	A500*	Inclusive OR to A; Indirect, Indexed, AP in Scratchpad	4-12
IOR	A600*	Inclusive OR to A; Direct, Relative to P Backward	4-12
IOR	A700*	Inclusive OR to A; Indirect, AP Relative to P Backward	4-12
IORB	A000*	Inclusive OR Byte to A; Direct, Scratchpad	4-12
IORB	A100*	Inclusive OR Byte to A; Indirect, AP in Scratchpad	4-12
IORB	A200*	Inclusive OR Byte 0 to A; Direct, Relative to P Forward	4-12
IORB	A300*	Inclusive OR Byte to A; Indirect, AP Relative to P Forward	4-12
IORB	A400*	Inclusive OR Byte to A; Direct, Indexed	4-12
IORB	A500*	Inclusive OR Byte to A; Indirect, Indexed, AP in Scratchpad	4-12
IORB	A600*	Inclusive OR Byte 0 to A; Direct, Relative to P Forward	4-12
IORB	A700*	Inclusive OR Byte to A; Indirect, AP Relative to P Backward	4-12
*IORS	1478	Inclusive OR Stack Element to A; Direct	4-21
*IORS	1479	Inclusive OR Stack Element to A; Indexed	4-21
*IORS	147A	Inclusive OR Stack Element to A; Auto-Postincrement	4-21
*IORS	147B	Inclusive OR Stack Element to A; Auto-Predecrement	4-21

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
IPH	1C21	Input Console Data Register to P and Halt	4-35
IPX	0090	Increment P and Put in X	4-34
ISA	5801	Input Console Data Switches to A	4-35
ISX	5A01	Input Console Data Switches to X	4-35
IXA	0130	Increment X and Put in A	4-34
IXH	1C09	Input Console Data Register to X and Halt	4-35
IXR	0128	Increment X	4-32
JAG		Jump if A Positive and Not Equal to Zero (A > 0)	4-25
	3180*	Forward Jump	
	31C0*	Backward Jump	
JAL		Jump if A Negative or Equal to Zero (A ≤ 0)	4-25
	2180*	Forward Jump	
	21C0*	Backward Jump	
JAM		Jump if A Negative (A<0)	4-25
	2080*	Forward Jump	
	20C0*	Backward Jump	
JAN		Jump if A Not Zero (A≠0)	4-25
	3100*	Forward Jump	
	3140*	Backward Jump	
JAP		Jump if A Positive or Equal to Zero (A≥0)	4-25
	3080*	Forward Jump	
	30C0*	Backward Jump	
JAZ		Jump if A Zero (A=0)	4-25
	2100*	Forward Jump	
	2140*	Backward Jump	
JMP	F000*	Jump Unconditionally; Direct, Scratchpad	4-14
JMP	F100*	Jump Unconditionally; Indirect, AP in Scratchpad	4-14
JMP	F200*	Jump Unconditionally; Direct Relative to P Forward	4-14

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
JMP	F300*	Jump Unconditionally; Indirect AP Relative to P Forward	4-14
JMP	F400*	Jump Unconditionally; Direct, Indexed	4-14
JMP	F500*	Jump Unconditionally; Indirect, Indexed, AP in Scratchpad	4-14
JMP	F600*	Jump Unconditionally, Direct, Relative to P Backward	4-14
JMP	F700*	Jump Unconditionally; Indirect, AP Relative to P Backward	4-14
*JMPS	16D8	Jump to Stack Element Unconditional; Direct	4-22
*JMPS	16D9	Jump to Stack Element Unconditional; Indexed	4-22
*JMPS	16DA	Jump to Stack Element Unconditional; Auto-Postincrement	4-22
*JMPS	16DB	Jump to Stack Element Unconditional; Auto-Predecrement	4-22
JOC	2000	Jump on Condition Specified	4-24
JOR		Jump if Overflow Reset (OV=0)	4-26
	3200*	Forward Jump	
	3240*	Backward Jump	
JOS		Jump if Overflow Set (OV=1)	4-26
	2200*	Forward Jump	
	2240*	Backward Jump	
JSR		Jump if Sense Switch Off (SS=0)	4-26
	2400*	Forward Jump	
	2440*	Backward Jump	
JSS		Jump if Sense Switch On (SS=1)	4-26
	3400*	Forward Jump	
	3440*	Backward Jump	
JST	F800*	Jump and Store; Direct, Scratchpad	4-14

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
JST	F900*	Jump and Store; Indirect, AP in Scratchpad	4-14
JST	FA00*	Jump and Store; Direct, Relative to P Forward	4-14
JST	FB00*	Jump and Store; Indirect, AP Relative to P Forward	4-14
JST	FC00*	Jump and Store; Direct, Indexed	4-14
JST	FD00*	Jump and Store; Indirect, Indexed, AP in Scratchpad	4-14
JST	FE00*	Jump and Store; Direct, Relative to P Backward	4-14
JST	FF00*	Jump and Store; Indirect, AP Relative to P Backward	4-14
*JSTS	16F8	Jump and Store to Stack Element: Direct	4-22
*JSTS	16F9	Jump and Store to Stack Element; Indexed	4-22
*JSTS	16FA	Jump and Store to Stack Element; Auto-Postincrement	4-22
*JSTS	16FB	Jump and Store to Stack Element; Auto-Predecrement	4-22
JXN		Jump if X Non-Zero (X≠0)	4-26
	3800*	Forward Jump	
	3840*	Backward Jump	
JXZ		Jump if X Equal to Zero (X=0)	4-26
	2800*	Forward Jump	
	2840*	Backward Jump	
LAM	C700*	Load A Minus Immediate	4-23
LAO	13C0	LSB of A to OV	4-32
LAP	C600*	Load A Positive Immediate	4-23
LDA	B000*	Load A; Direct, Scratchpad	4-13
LDA	B100*	Load A; Indirect, AP in Scratchpad	4-13

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
LDA	B200*	Load A; Direct, Relative to P Forward	4-13
LDA	B300*	Load A; Indirect, AP Relative to P Forward	4-13
LDA	B400*	Load A; Direct, Indexed	4-13
LDA	B500*	Load A; Indirect, Indexed, AP in Scratchpad	4-13
LDA	B600*	Load A; Direct, Relative to P Backward	4-13
LDA	B700*	Load A; Indirect, AP Relative to P Backward	4-13
LDAB	B000*	Load A Byte; Direct, Scratchpad	4-13
LDAB	B100*	Load A Byte; Indirect; AP in Scratchpad	4-13
LDAB	B200*	Load A Byte 0; Direct, Relative to P Forward	4-13
LDAB	B300*	Load A Byte; Indirect, AP Relative to P Forward	4-13
LDAB	B400*	Load A Byte; Direct, Indexed	4-13
LDAB	B500*	Load A Byte; Indirect, Indexed, AP in Scratchpad	4-13
LDAB	B600*	Load A Byte 1; Direct, Relative to P Forward	4-13
LDAB	B700*	Load A Byte; Indirect, AP Relative to P Backward	4-13
*LDAS	14D8	Load Stack Element into A; Direct	4-21
*LDAS	14D9	Load Stack Element into A; Indexed	4-21
*LDAS	14DA	Load Stack Element into A; Auto-Postincrement	4-21
*LDAS	14DB	Load Stack Element into A; Auto-Predecrement	4-21
LDX	E000*	Load X; Direct, Scratchpad	4-13
LDX	E100*	Load X; Indirect, AP in Scratchpad	4-13
LDX	E200*	Load X; Direct, Relative to P Forward	4-13

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
LDX	E300*	Load X, Indirect, AP Relative to P Forward	4-13
LDX	E400*	Load X; Direct, Indexed	4-13
LDX	E500*	Load X; Indirect, Indexed, AP in Scratchpad	4-13
LDX	E600*	Load X; Direct, Relative to P Backward	4-13
LDX	E700*	Load X; Indirect; AP Relative to P Backward	4-13
LDXB	E000*	Load X Byte; Direct, Scratchpad	4-13
LDXB	E100*	Load X Byte; Indirect, AP in Scratchpad	4-13
LDXB	E200*	Load X Byte 0; Direct, Relative to P Forward	4-13
LDXB	E300*	Load X Byte; Indirect, AP Relative to P Forward	4-13
LDXB	E400*	Load X Byte; Direct, Indexed	4-13
LDXB	E500*	Load X Byte; Indirect, Indexed, AP in Scratchpad	4-13
LDXB	E600*	Load X Byte 1; Direct, Relative to P Forward	4-13
LDXB	E700*	Load X Byte; Indirect, AP Relative to P Backward	4-13
*LDXS	1698	Load Stack Element into X; Direct	4-21
*LDXS	1699	Load Stack Element into X; Indexed	4-21
*LDXS	169A	Load Stack Element into X; Auto-Postincrement	4-21
*LDXS	169B	Load Stack Element into X; Auto-Predecrement	4-21
LLA	1350*	Logical Shift A Left	4-28
LLL	1B00*	Long Logical Left Shift	4-30
LLR	1B80*	Long Logical Right Shift	4-30
LLX	1328*	Logical Shift X Left	4-28
LRA	13D0*	Logical Shift A Right	4-28

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
LRL	1900*	Long Rotate Left	4-31
LRR	1980*	Long Rotate Right	4-31
LRX	13A8*	Logical Shift X Right	4-28
LXM	C500*	Load X Minus Immediate	4-23
LXO	13A0	LSB of X to OV	4-32
LXP	C400*	Load X Positive Immediate	4-23
MPY	1960*	Multiply and Add	4-16
NAR	0310	Negate A Register	4-31
NAX	0308	Negate A and Put in X	4-33
NOP	0000	No Operation	4-36
NRA	0610	NOR of A and X to A	4-33
NRM	1940*	Normalize A and X	4-17
NRX	0608	NOR of A and X to X	4-33
NXA	0510	Negate X and Put in A	4-33
NXR	0508	Negate X Register	4-32
OAH	1C04	Output A to Console Data Register and Halt	4-35
OCA	4404	Output A to Console Data Register	4-36
OCX	4604	Output X to Console Data Register	4-36
OLH	1C10	Output Location to Console Data Register and Halt	4-36
OMH	1C02	Output Memory to Console Data Register and Halt	4-36
OPH	1C20	Output P to Console Data Register and Halt	4-36
OTA	6C00*	Output A Register (Unconditionally)	4-41

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
OTX	6E00*	Output X Register (Unconditionally)	4-41
OTZ	6800*	Output Zero (Unconditionally)	4-41
OXH	1C08	Output X to Console Data Register (Unconditionally)	4-35
PFD	4003	Power Fail Interrupt Disable	4-38
PFE	4002	Power Fail Interrupt Enable	4-38
RBA	7900*	Read Byte to A Register	4-43
RBAM	7D00*	Read Byte to A Register, Masked	4-43
RBX	7B00*	Read Byte to X Register	4-43
RBXM	7F00*	Read Byte to X Register, Masked	4-43
RDA	5900*	Read Word to A Register	4-41
RDAM	5D00*	Read Word to A Register, Masked	4-41
RDX	5B00*	Read Word to X Register	4-41
RDXM	5F00*	Read Word to X Register, Masked	4-42
RLA	1150*	Rotate A Left with OV	4-29
RLX	1128*	Rotate X Left with OV	4-29
ROV	1200	Reset Overflow	4-32
RRA	11D0*	Rotate A Right with OV	4-29
RRX	11A8*	Rotate X Right with OV	4-29
SAI	0D00*	Subtract from A Immediate	4-23
SAO	1340	Sign of A to OV	4-32
SBM	0E00	Set Byte Mode	4-37
SCM	CD00*	Scan Memory; Indirect, Indexed, AP in Scratchpad	4-14

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
SCMB	CD00*	Scan Memory Byte; Indirect, Indexed, AP in Scratchpad	4-15
SEA	4400*	Select and Present A	4-40
SEL	4000*	Select Function	4-40
SEN	4900*	Sense and Skip on Response	4-40
SEX	4600*	Select and Present X	4-40
SIA	5800	Status Input to A	4-38
SIN	6800	Status Inhibit	4-37
SIX	5A00	Status Input to X	4-38
*SLAS	1618	Stack Element Address to A; Direct	4-22
*SLAS	1619	Stack Element Address to A; Indexed	4-22
*SLAS	161A	Stack Element Address to A; Auto-Postincrement	4-22
*SLAS	161B	Stack Element Address to A; Auto-Predecrement	4-22
SOA	6C00	Status Output from A	4-38
SOX	6E00	Status Output from X	4-38
SOV	1400	Set Overflow	4-32
SSN	4800*	Sense and Skip and No Response	4-40
STA	9800*	Store A; Direct, Scratchpad	4-13
STA	9900*	Store A; Indirect, AP in Scratchpad	4-13
STA	9A00*	Store A; Direct, Relative to P Forward	4-13
STA	9B00*	Store A; Indirect, AP Relative to P Forward	4-13
STA	9C00*	Store A; Direct, Indexed	4-13
STA	9D00*	Store A; Indirect, Indexed, AP in Scratchpad	4-13

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
STA	9E00*	Store A; Direct, Relative to P Backward	4-13
STA	9F00*	Store A; Indirect, AP Relative to P Backward	4-13
STAB	9800*	Store A Byte; Direct, Scratchpad	4-13
STAB	9900*	Store A Byte; Indirect, AP in Scratchpad	4-13
STAB	9A00*	Store A Byte 0; Direct, Relative to P Forward	4-13
STAB	9B00*	Store A Byte; Indirect, AP Relative to P Forward	4-13
STAB	9C00*	Store A Byte; Direct, Indexed	4-13
STAB	9D00*	Store A Byte; Indirect, Indexed, AP in Scratchpad	4-13
STAB	9E00*	Store A Byte 1; Direct, Relative to P Forward	4-13
STAB	9F00*	Store A Byte; Indirect, AP Relative to P Backward	4-13
*STAS	1478	Store A in Stack Element; Direct	4-21
*STAS	1479	Store A in Stack Element; Indexed	4-21
*STAS	147A	Store A in Stack Element; Auto-Postincrement	4-21
*STAS	147B	Store A in Stack Element; Auto-Predecrement	4-21
STOP	0800*	Halt with Operand	4-36
STX	E800*	Store X; Direct, Scratchpad	4-13
STX	E900*	Store X; Indirect, AP in Scratchpad	4-13
STX	EA00*	Store X; Direct, Relative to P Forward	4-13
STX	EB00*	Store X; Indirect, AP Relative to P Forward	4-13
STX	EC00*	Store X; Direct, Indexed	4-13
STX	ED00*	Store X; Indirect, Indexed, AP in Scratchpad	4-13

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
STX	EE00*	Store X; Direct, Relative to P Backward	4-13
STX	EF00*	Store X; Indirect; AP Relative to P Backward	4-13
STXB	E800*	Store X Byte; Direct, Scratchpad	4-13
STXB	E900*	Store X Byte; Indirect, AP in Scratchpad	4-13
STXB	EA00*	Store X Byte 0; Direct, Relative to P Forward	4-13
STXB	EB00*	Store X Byte; Indirect, AP Relative to P Forward	4-13
STXB	EC00*	Store X Byte; Direct, Indexed	4-13
STXB	ED00*	Store X Byte; Indirect, Indexed, AP in Scratchpad	4-13
STXB	EE00*	Store X Byte 1; Direct, Relative to P Forward	4-13
STXB	EF00*	Store X Byte; Indirect, AP Relative to P Backward	4-13
*STXS	16B8	Store X in Stack Element; Direct	4-21
*STXS	16B9	Store X in Stack Element; Indexed	4-21
*STXS	16BA	Store X in Stack Element; Auto-Postincrement	4-21
*STXS	16BB	Store X in Stack Element; Auto-Predecrement	4-21
SUB	9000*	Subtract from A; Direct, Scratchpad	4-12
SUB	9100*	Subtract from A; Indirect, AP in Scratchpad	4-12
SUB	9200*	Subtract from A; Direct, Relative to P Forward	4-12
SUB	9300*	Subtract from A; Indirect, AP Relative to P Forward	4-12
SUB	9400*	Subtract from A; Direct, Indexed	4-12
SUB	9500*	Subtract from A; Indirect, Indexed, AP in Scratchpad	4-12

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
SUB	9600*	Subtract from A; Direct, Relative to P Backward	4-12
SUB	9700*	Subtract from A; Indirect, AP Relative to P Backward	4-12
SUBB	9000*	Subtract Byte; Direct, Scratchpad	4-12
SUBB	9100*	Subtract Byte; Indirect, AP in Scratchpad	4-12
SUBB	9200*	Subtract Byte 0; Direct, Relative to P Forward	4-12
SUBB	9300*	Subtract Byte; Indirect, AP Relative to P Forward	4-12
SUBB	9400*	Subtract Byte; Direct, Indexed	4-12
SUBB	9500*	Subtract Byte; Indirect, Indexed, AP in Scratchpad	4-12
SUBB	9600*	Subtract Byte 1; Direct, Relative to P Forward	4-12
SUBB	9700*	Subtract Byte; Indirect, AP Relative to P Backward	4-12
*SUBS	1458	Subtract Stack Element from A; Direct	4-21
*SUBS	1459	Subtract Stack Element from A; Indexed	4-21
*SUBS	145A	Subtract Stack Element from A; Auto-Postincrement	4-21
*SUBS	145B	Subtract Stack Element from A; Auto-Predecrement	4-21
SWM	0F00	Set Word Mode	4-37
SXI	C300*	Subtract from X Immediate	4-23
SXO	1320	Sign of X to OV	4-32
TAX	0048	Transfer A to X	4-33
TRP	4007	Trap	4-39
TXA	0030	Transfer X to A	4-33

INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
WAIT	F600	Wait for Interrupts	4-37
WRA	6D00*	Write from A	4-42
WRX	6F00*	Write from X	4-42
WRZ	6900*	Write Zeros	4-42
XOR	A800*	Exclusive OR to A; Direct, Scratchpad	4-12
XOR	A900*	Exclusive OR to A; Indirect, AP in Scratchpad	4-12
XOR	AA00*	Exclusive OR to A; Direct, Relative to P Forward	4-12
XOR	AB00*	Exclusive OR to A; AP Relative to P Forward, Indirect	4-12
XOR	AC00*	Exclusive OR to A; Direct, Indexed	4-12
XOR	AD00*	Exclusive OR to A; Indirect, Indexed, AP in Scratchpad	4-12
XOR	AE00*	Exclusive OR to A; Direct, Relative to P Backward	4-12
XOR	AF00*	Exclusive OR to A; Indirect, AP Relative to P Backward	4-12
XORB	A800*	Exclusive OR Byte; Direct, Scratchpad	4-13
XORB	A900*	Exclusive OR Byte; Indirect, AP in Scratchpad	4-13
XORB	AA00*	Exclusive OR Byte 0; Direct, Relative to P Forward	4-13
XORB	AB00*	Exclusive OR Byte; Indirect, AP Relative to P Forward	4-13
XORB	AC00*	Exclusive OR Byte; Direct, Indexed	4-13
XORB	AD00*	Exclusive OR Byte; Indirect, Indexed, AP in Scratchpad	4-13
XORB	AE00*	Exclusive OR Byte 1; Direct, Relative to P Forward	4-13

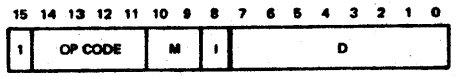
INSTRUCTION SET IN ALPHABETICAL ORDER (Cont'd)

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
XORB	AF00*	Exclusive OR Byte; Indirect, AP Relative to P Backward	4-13
*XORS	14B8	Exclusive OR Stack Element to A; Direct	4-21
*XORS	14B9	Exclusive OR Stack Element to A; Indexed	4-21
*XORS	14BA	Exclusive OR Stack Element to A; Auto-Postincrement	4-21
*XORS	14BB	Exclusive OR Stack Element to A; Auto-Predecrement	4-21
XRM	0008	Set X to Minus 1	4-32
XRP	0528	Set X to Plus 1	4-32
ZAR	0110	Zero A Register	4-31
ZAX	0118	Zero A and X Registers	4-33
ZXR	0108	Zero X Register	4-32

Appendix E
INSTRUCTION SET IN NUMERICAL ORDER

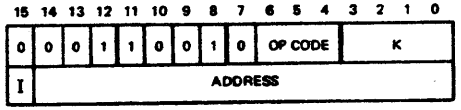
This appendix contains the ALPHA LSI instruction set in machine code in numerical order. For each instruction, reference is made to one of the machine code formats listed below. Instructions with variable fields (D, K, etc.) are followed by asterisks (*). Those instructions which apply to LSI-2 only are prefixed with an asterisk.

D = Address Field (0 to 255)
I = Direct/Indirect Address Bit
M = Address Mode Code
Y = Effective Address



M	I	Word Mode (Word Operand)	Byte Mode (Byte Operand)
00	0	Y = (D), Words : 00-:FF	Y = (D), Bytes : 00-:FF
01	0	Y = (D) + (P) + 1	Y = (D) + (P) 1, Byte 0
10	0	Y = (D) + (X)	Y = (D) + (X)
11	0	Y = (P) - (D)	Y = (D) + (P) + 1, Byte 1
00	1	AP = (D), AP = (AP) , Y = (AP)	AP = (D), Y = (AP)
01	1	AP = (D) + (P) + 1, AP = (AP) , Y + (AP)	AP = (D) + (P) + 1, Y=(AP)
10	1	AP = (D), AP = (AP) , Y = (AP) + (X)	AP = (D), Y = (AP) + (X)
11	1	AP = (P) - (D), AP = (AP) , Y = (AP)	AP = (P) - (D), Y = (AP)

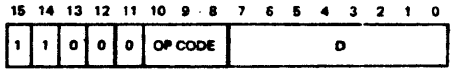
Figure E-1. Single-Word Memory Reference Instruction Machine Code Format



Op Code = 100 for NRM 0 through 15
= 101 for NRM 16 through 31
= 110 for MPY
= 111 for DVD

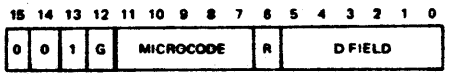
I = Indirect Addressing
1 = Indirect Address
0 = Direct Address
K = Instruction Count

Figure E-2. Double-Word Memory Reference Instruction Machine Code Format



D = 8-Bit Immediate Operand

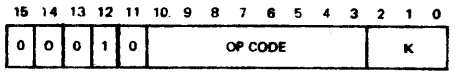
Figure E-3. Byte Immediate Instruction Machine Code Format



Bits	Field	Definition
12	G	Test Group Indicator: G = 1 for AND Group G = 0 for OR Group
7-11	Conditions	Microcode of Test Conditions:
6	R	Jump Direction: R = 0 for Forward Jump R = 1 for Backward Jump
0-5	D Field	Jump Distance (-63 to +64)

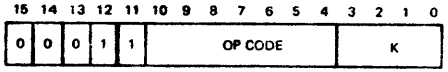
Bit	AND Group	OR Group
7	A Positive	A Negative
8	A ≠ 0	A = 0
9	OV Reset	OV Set (Resets OV)
10	Sense Indicator on	Sense Indicator off
11	X ≠ 0	X = 0

Figure E-4. Conditional Jump Instruction Machine Code Format



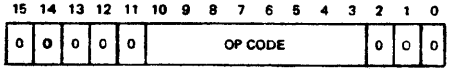
K = Shift Control Count, Shift Will Move 1 + K Bit Positions.
Op Code = Shift Control Code Which Selects Source, Type of Shift, and Location of Results

Figure E-5. Single-Register Shift Instruction Machine Code Format



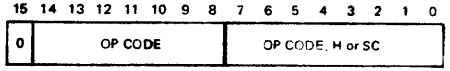
Op Code = Shift Control Code Which Selects the Type of Long Shift to be Executed
K = Shift Count. Shift Will Move 1 + K Bit Positions

Figure E-6. Double-Register Shift Instruction Machine Code Format



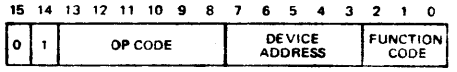
Op Code = The Register Change Control Code which specifies the Source, Operation, and Location of Results

Figure E-7. Register Change Instruction Machine Code Format



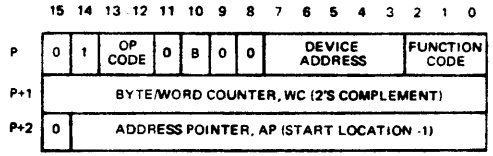
H = Halt ID Indicator
SC = Sin Instruction Count - 1

Figure E-8. Control Instruction Machine Code Format



Function Code = Specifies which device function or register
Device Address = The device's assigned address
Op Code = Operation Code Specifying One of the I/O Instructions

Figure E-9. Input/Output Instruction Machine Code Format



Opcode; 01 = Input, 10 = Output
B = 0: Word Transfer
B = 1: Byte Transfer
Byte/Word Counter = Number of Executions Until Skip or Echo
Byte/Word Address Pointer = Memory Location of I/O Transaction

Figure E-10. Automatic Input/Output Instruction Machine Code Format

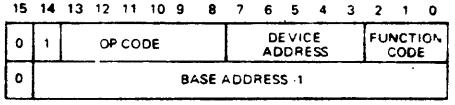
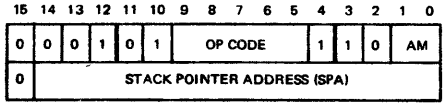


Figure E-11. Block Input/Output Instruction Machine Code Format



AM = Addressing Mode

00 = Direct Access to Stack
01 = Indexed Access to Stack
10 = Auto-increment Access to Stack (POP)
11 = Auto-decrement Access to Stack (PUSH)

Figure E-12. Stack Instruction Machine Code Format

INSTRUCTION SET IN NUMERICAL ORDER

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
0000	NOP	No Operation	7	4-36
0008	XRM	X Register to Minus One	7	4-32
0010	ARM	A Register to Minus One	7	4-31
0018	AXM	A and X Registers to Minus One	7	4-33
0030	TXA	Transfer X to A	7	4-33
0048	TAX	Transfer A to X	7	4-33
0068	ANX	AND of A and X to X	7	4-33
0070	ANA	AND of A and X to A	7	4-33
0090	IPX	Increment P to X	7	4-34
00A8	DXR	Decrement X Register	7	4-32
00B0	DXA	Decrement X to A	7	4-34
00C8	DAX	Decrement A to X	7	4-34

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
00D0	DAR	Decrement A Register	7	4-31
0108	ZXR	Zero X Register	7	4-32
0110	ZAR	Zero A Register	7	4-31
0118	ZAX	Zero A and X Registers	7	4-33
0128	IXR	Increment X Register	7	4-32
0130	IXA	Increment X to A	7	4-34
0148	IAX	Increment A to X	7	4-41
0150	IAR	Increment A Register	7	4-31
0208	CAX	Complement of A to X	7	4-33
0210	CAR	Complement A Register	7	4-31
0218	*EIX	Execute Instruction Pointed to By X	7	4-34
0308	NAX	Negate A to X	7	4-33
0310	NAR	Negate A Register	7	4-31
0350	ARP	A Register to Plus One	7	4-31
0358	AXP	A and X Registers to Plus One	7	4-33
0408	CXR	Complement X Register	7	4-32
0410	CXA	Complement of X to A	7	4-33
0428	EAX	Exchange A and X	7	4-33
0508	NXR	Negate X Regster	7	4-32
0510	NXA	Negate X to A	7	4-33
0528	XRP	X Register to Plus One	7	4-32
0608	NRX	NOR of A and X to X	7	4-33

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
0610	NRA	NOR of A and X to A	7	4-33
0688	*BSX	Bit Set X	7	4-34
068A	*BSA	Bit Set A	7	4-34
06C8	*BCX	Bit Clear X	7	4-34
06CA	*BCA	Bit Clear A	7	4-34
0800	HLT	Halt	8	4-36
0800	STOP*	Halt with Operand	8	4-36
0A00	EIN	Enable Interrupts	8	4-38
0B00	AAI*	Add to A Immediate	3	4-23
0C00	DIN	Disable Interrupts	8	4-38
0D00	SAI*	Subtract from A Immediate	3	4-23
0E00	SBM	Set Byte Mode	8	4-37
0F00	SWM	Set Word Mode	8	4-37
1028	ALX*	Arithmetic Shift X Left	5	4-27
0150	ALA*	Arithmetic Shift A Left	5	4-27
10A8*	ARX*	Arithmetic Shift X Right	5	4-27
10D0	ARA*	Arithmetic Shift A Right	5	4-27
1128	RLX*	Rotate X Left with Overflow	5	4-29
1150	RLA*	Rotate A Left with Overflow	5	4-29
11A8	RRX*	Rotate X Right with Overflow	5	4-29
11D0	RRA*	Rotate A Right with Overflow	5	4-29
1200	ROV	Reset Overflow	5	4-32
1320	BXO*	Bit of X to Overflow	5	4-32

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
1320	SXO	Sign of X to Overflow	5	4-32
1328	LLX*	Logical Shift X Left	5	4-28
1340	BAO*	Bit of A to Overflow	5	4-32
1340	SAO	Sign of A to Overflow	5	4-32
1350	LLA*	Logical Shift A Left	5	4-28
13A0	LXO	LSB of X to Overflow	5	4-32
13A8	LRX*	Logical Shift X Right	5	4-28
13C0	LAO	LSB of A to Overflow	5	4-32
13D0	LRA*	Logical Shift A Right	5	4-28
1400	SOV	Set Overflow	5	4-32
1418	*ANDS	AND Stack Element to A	12	4-21
1438	*ADDS	Add Stack Element to A	12	4-21
1458	*SUBS	Subtract Stack Element from A	12	4-21
1478	*STAS	Store A in Stack Element	12	4-21
1498	*IORS	Inclusive OR Stack Element to A	12	4-21
14B8	*XORS	Exclusive OR Stack Element to A	12	4-21
14D8	*LDAS	Load Stack Element into A	12	4-21
14F8	*EMAS	Exchange Stack Element and A	12	4-21
1600	COV	Complement Overflow	5	4-22
1618	*SLAS	Stack Element Address to A	12	4-22
1658	*CMSS	Compare Stack Element to A and Skip if High or Equal	12	4-22
1678	*IMSS	Increment Stack Element and Skip on Zero Result	12	4-22

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
1698	*LDXS	Load Stack Element into X	12	4-21
16B8	*STXS	Store X in Stack Element	12	4-21
16D8	*JMPS	Jump to Stack Element Unconditional	12	4-22
16F8	*JSTS	Jump to Stack Element and Store	12	4-22
1900	LRL*	Long Rotate Left	6	4-31
1940	NRM	Normalize A and X	2	4-17
1960	MPY	Multiply and Add	2	4-16
1970	DVD	Divide	2	4-16
1980	LRR*	Long Rotate Right	6	4-31
1B00	LLL*	Long Logical Shift Left	6	4-30
1B80	LLR*	Long Logical Shift Right	6	4-30
1C02	OMH	Output Console Data Register to Memory and Halt	9	4-36
1C03	IMH	Input Console Data Register to Memory and Halt	9	4-35
1C04	OAH	Output A to Console Data Register and Halt	9	4-35
1C05	IAH	Input Console Data Register to A and Halt	9	4-35
1C08	OXH	Output X to Console Data Register and Halt	9	4-35
1C09	IXH	Input Console Data Register to X and Halt	9	4-35
1C10	OLH	Output Location to Console Data Register and Halt	9	4-36
1C11	IIH	Input Console Data Register to I and Halt	9	4-35

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
1C20	OPH	Output P to Console Data Register and Halt	9	4-36
1C21	IPH	Input Console Data Register to P and Halt	9	4-35
2080-3F80 Fwd 20C0-3FC0 Bkwd	JOC*	Jump on Condition	4	4-24
2080 Fwd 20C0 Bkwd	JAM*	Jump if A Minus	4	4-25
2100 Fwd 2140 Bkwd	JAZ*	Jump if A Zero	4	4-25
2180 Fwd 21C0 Bkwd	JAL*	Jump if A Less Than One	4	4-25
2200 Fwd 2240 Bkwd	JOS*	Jump if Overflow Set	4	4-26
2400 Fwd 2440 Bkwd	JSR*	Jump if Sense Switch Reset	4	4-26
2800 Fwd 2840 Bkwd	JXZ*	Jump if X Zero	4	4-26
3080 Fwd 30C0 Bkwd	JAP*	Jump if A Positive	4	4-25
3100 Fwd 3140 Bkwd	JAN*	Jump if A Not Zero	4	4-25
3180 Fwd 31C0 Bkwd	JAG*	Jump if A Greater Than Zero	4	4-25
3200 Fwd 3240 Bkwd	JOR*	Jump if Overflow Reset	4	4-26
3400 Fwd 3440 Bkwd	JSS*	Jump if Sense Switch Set	4	4-26
3800 Fwd 3840 Bkwd	JXN*	Jump if X Not Zero	4	4-26

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
4000	SEL*	Select Function	9	4-40
4002	PFE	Power Fail Enable	9	4-38
4003	PFD	Power Fail Disable	9	4-38
4005	CIE	Console Interrupt Enable	9	4-38
4006	CID	Console Interrupt Disable	9	4-38
4007	TRP	Trap	9	4-39
4400	SEA*	Select and Present A	9	4-40
4404	OCA	Output A to Console Register	9	4-36
4600	SEX*	Select and Present X	9	4-40
4604	OCX	Output X to Console Register	9	4-36
4800	SSN*	Sense and Skip On No Response	9	4-40
4900	SEN*	Sense and Skip On Response	9	4-40
5000	AIN*	Automatic Input Word to Memory	10	4-47
5400	AIB*	Automatic Input Byte to Memory	10	4-47
5800	INA*	Input to A Register	9	4-41
5800	SIA	Status Input to A	9	4-38
5801	ISA	Input Sense Register to A	9	4-35
5804	ICA	Input Console Register to A	9	4-35
5900	RDA*	Read Word to A Register	9	4-41
5A00	INX*	Input to X Register	9	4-41
5A00	SIX	Status Input to X	9	4-38
5A01	ISX	Input Sense Register to X	9	4-35
5A04	ICX	Input Console Register to X	9	4-35

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
5B00	RDX*	Read Word to X Register	9	4-41
5C00	INAM*	Input to A Register Masked	9	4-41
5D00	RDAM*	Read Word to A Register Masked	9	4-41
5E00	INXM*	Input to X Register Masked	9	4-41
5F00	RDXM*	Read Word to X Register Masked	10	4-42
6000	AOT*	Automatic Output Word from Memory	10	4-47
6400	AOB*	Automatic Output Byte from Memory	10	4-47
6800	OTZ*	Output Zero	9	4-41
6800	SIN*	Status Inhibit	8	4-37
6900	WRZ*	Write Zero	9	4-42
6C00	OTA*	Output A Register	9	4-41
6C00	SOA	Status Output from A	9	4-38
6D00	WRA*	Write from A Register	9	4-42
6E00	OTX*	Output X Register	3	4-41
6E00	SOX	Status Output from X	9	4-38
6F00	WRX*	Write from X Register	3	4-42
7100	BIN*	Block In	11	4-44
7500	BOT*	Block Out	11	4-45
7800	IBA*	Input Byte to A Register	9	4-42
7900	RBA*	Read Byte to A Register	9	4-43
7A00	IBX*	Input Byte to X Register	9	4-43
7B00	RBX*	Read Byte to X Register	9	4-43
7C00	IBAM*	Input Byte to A Register Masked	9	4-42

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
7D00	RBAM*	Read Byte to A Register Masked	9	4-43
7E00	IBXM*	Input Byte to X Register Masked	9	4-43
7F00	RBXM*	Read Byte to X Register Masked	9	4-43
8000	AND*	AND to A	1	4-12
8000	ANDB*	AND Byte to A	1	4-12
8800	ADD*	Add to A	1	4-12
8800	ADDB*	Add Byte to A	1	4-12
9000	SUB*	Subtract from A	1	4-12
9000	SUBB*	Subtract Byte from A	1	4-12
9800	STA*	Store A	1	4-13
9800	STAB*	Store A Byte	1	4-13
A000	IOR*	Inclusive OR to A	1	4-12
A000	IORB*	Inclusive OR Byte to A	1	4-12
A800	XOR*	Exclusive OR to A	1	4-12
A800	XORB*	Exclusive OR Byte to A	1	4-13
B000	LDA*	Load A	1	4-13
B000	LDAB*	Load A Byte	1	4-13
B800	EMA*	Exchange Memory and A	1	4-13
B800	EMAB*	Exchange Memory Byte and A	1	4-13
C000	CAI*	Compare to A Immediate	3	4-23
C100	CXI*	Compare to X Immediate	3	4-23
C200	AXI*	Add to X Immediate	3	4-23
C300	SXI*	Subtract from X Immediate	3	4-23

INSTRUCTION SET IN NUMERICAL ORDER (Cont'd)

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
C400	LXP*	Load X Positive Immediate	3	4-23
C500	LXM*	Load X Minus Immediate	3	4-23
C600	LAP*	Load A Positive Immediate	3	4-23
C700	LAM*	Load A Minus Immediate	3	4-23
CD00	SCM*	Scan Memory	1	4-14
CD00	SCMB*	Scan Memory Byte	1	4-15
D000	CMS*	Compare and Skip if High or Equal	1	4-22
D000	CMSB*	Compare Byte and Skip if High or Equal	1	4-14
D800	IMS*	Increment Memory and Skip on Zero Result	1	4-14
E000	LDX*	Load X	1	4-13
E000	LDXB*	Load X Byte	1	4-13
E800	STX*	Store X	1	4-13
E800	STXB*	Store X Byte	1	4-13
F000	JMP*	Jump Unconditional	1	4-14
F600	WAIT	Wait for Interrupts	1	4-37
F800	JST*	Jump and Store	1	4-14

Appendix F
ALPHA LSI EXECUTION TIMES

F.1 GENERAL

This appendix defines the execution time of each instruction in the ALPHA LSI instruction set. Two Processors and a variety of Memories, with varying access times, are offered with the ALPHA LSI. The variation in memory access time makes a tabulation of execution times difficult. For this reason time calculation algorithms are provided. These algorithms are useful with any memory access time by making the appropriate memory parameter substitution.

F.2 MEMORY PARAMETERS

Currently, four Memories are offered in the ALPHA LSI family; three of these are core Memories, while the fourth is a semiconductor Memory. Table F-1 lists the parameters of these Memories. All times listed are in nanoseconds.

Table F-1. LSI Family Memory Parameters

Memory Type	Configuration	C	RA	RO	WA	WO	M	M'	ROI	WOI
Core 980	Add on 4K, 8K	980	380	600	180	800	600	400	220	420
Core 1200	Add on 16K	1200	400	800	200	1000	600	400	300	500
Core 1600	Add on or integral 4K, 8K	1600	450	1150	250	1350	600	400	0	0
SC 1200	Add on 2K, 4K, 8K Integral 2K, 4K	1200	500	700	200	1000	600	400	0	0

Parameters in nanoseconds are:

C = Cycle Time
RA = Read Access
RO = Read Overhead
WA = Write Access
WO = Write Overhead
M = LSI-1 Effective Read Access
M' = LSI-1 Effective Write Access
ROI = Interleaved Effective Read Overhead
WOI = Interleaved Effective Write Overhead

F.3 LSI-1 EXECUTION TIME ALGORITHMS

The LSI-1 execution time algorithms are listed in table F-2. The algorithms are partitioned by class and subclass. Numerous instructions have two times listed with the reason for the dual listing given in parenthesis. All numeric values are in microseconds. The value of A (address calculation time) is derived from the list of addressing modes at the beginning of the table. The variables m and m' are derived from table F-1 and are in nanoseconds.

The letter i stands for indirect address levels. Where indirect addressing is used, the value (3.2 + m)i must be added for each level of indirect addressing that is employed.

The letter n denotes a shift. The value 1.6n or 3.2n must be added to the basic execution time of shift instructions for each bit shifted.

The letter w is used by the SCM and Block I/O instructions. The parenthetical expression which precedes the w is the time calculation on a per word basis.

Table F-2. LSI-1 Execution Time Algorithms

MEMORY REFERENCE CLASS

A = Address Calculation Time for Memory Reference Instructions:

DIRECT SCRATCHPAD	1.6 + m
DIRECT RELATIVE	1.6 + m
DIRECT INDEXED	3.2 + m
INDIRECT SCRATCHPAD	(3.2 + m) i
INDIRECT REALTIVE	1.6 + (3.2 + m) i
INDIRECT INDEXED	1.6 + (3.2 + m) i
ARITHMETIC	
ADD	6.4 + m + A
SUB	6.4 + m + A
LOGICAL	
AND	6.4 + m + A
IOR	6.4 + m + A
XOR	6.4 + m + A
DATA TRANSFER	
LDA	4.8 + m + A
LDX	4.8 + m + A
STA	4.8 + m' + A
STX	4.8 + m' + A
EMA	8.0 + m + m' + A

Table F-2. LSI-1 Execution Time Algorithms (Cont'd)

PROGRAM TRANSFER	
JMP	4.8 + A
JST (Non-Interrupt)	8.0 + m' + A
JST (Interrupt)	6.4 + m' + A
IMS	9.6 + m + m' + A
SCM	(12.8 + m + A) w
CMS	12.8 + m + A
DOUBLE WORD MEMORY REFERENCE CLASS	
DVD	118.4 + 3m + (3.2 + m) i
MPY	110.4 + 3m + (3.2 + m) i
NRM (count expires)	17.6 + 3m + m' + 9.6n + (3.2 + m) i
NRM (count does not expire)	20.8 + 3m + m' + 9.6n + (3.2 + m) i
BYTE IMMEDIATE CLASS	
AAI	4.8 + m
AXI	4.8 + m
SAI	4.8 + m
SXI	4.8 + m
CAI	6.4 + m
CXI	6.4 + m
LAP	4.8 + m
LXP	4.8 + m
LAM	4.8 + m
LXM	4.8 + m
CONDITIONAL JUMP CLASS	
MICROCODED (JOC)	
ALL Double Register Tests	14.4 + m
ALL Others	6.4 + m
ARITHMETIC	
JAG, JAP, JAZ, JAN, JAL, JAM, JXZ, JXN	6.4 + m

Table F-2. LSI-1 Execution Time Algorithms (Cont'd)

CONTROL	
JSS, JSR, JOS, JOR	6.4 + m
SHIFT CLASS	
ARITHMETIC SHIFTS	
ARA, ARX, ALA, ALX	3.2 + m + 1.6n
LOGICAL SHIFTS	
LRA, LRX, LLA, LLX	3.2 + m + 1.6n
ROTATE SHIFTS	
RRA, RRX, RLA, RLX	3.2 + m + 1.6n
DOUBLE REGISTER LOGICAL SHIFTS	
LLL, LLR	3.2 + m + 3.2n
DOUBLE REGISTER ROTATE SHIFTS	
LRL, LRR	3.2 + m + 3.2n
REGISTER CHANGE CLASS	
A REGISTER CHANGE	
ZAR, ARP, ARM, CAR, NAR, IAR, DAR	4.8 + m

Table F-2. LSI-1 Executive Time Algorithms (Cont'd)

X REGISTER CHANGE	
ZXR	4.8 + m
XRP	
XRM	
CXR	
NXR	
IXR	
DXR	
OVERFLOW REGISTER CHANGE	
SOV	4.8 + m
ROV	4.8 + m
COV	4.8 + m
SAO	6.4 + m
SXO	6.4 + m
LAO	6.4 + m
LXO	6.4 + m
BAO	6.4 + m + 1.6n
BXO	6.4 + m + 1.6n
MULTI-REGISTER CHANGE	
ZAX	6.4 + m
AXP	6.4 + m
AXM	6.4 + m
TAX	4.8 + m
TXA	4.8 + m
EAX	8.0 + m
ANA	4.8 + m
ANX	4.8 + m
NRA	6.4 + m
NRX	6.4 + m
CAX	4.8 + m
CXA	4.8 + m
NAX	4.8 + m
NXA	4.8 + m
IAX	4.8 + m
IXA	4.8 + m
IPX	4.8 + m
DAX	4.8 + m
DXA	4.8 + m
CONSOLE REGISTER	
ICA	5.6 + m
ICX	
ISA	
ISX	
OCA	
OCX	

Table F-2. LSI-1 Executive Time Algorithms (Cont'd)

CONTROL CLASS	
PROCESSOR CONTROLS	
NOP	4.8 + m
HLT (STOP)	
MODE CONTROLS	
SBM	4.8 + m
SWM	
STATUS CONTROLS	
SIN	5.6 + m
SIA	
SIX	
SOA	
SOX	
INTERRUPT CONTROLS	
EIN	4.8 + m
DIN	6.4 + m
CIE	5.6 + m
CID	5.6 + m
PFE	5.6 + m
PFD	5.6 + m
TRP	5.6 + m
INPUT/OUTPUT CLASS	
CONTROL	
SEL	5.6 + m
SEA	5.6 + m
SEX	5.6 + m
SEN	7.2 + m
SSN	7.2 + m
UNCONDITIONAL WORD	
INA	5.6 + m
INAM	7.2 + m
INX	5.6 + m
INXM	7.2 + m
OTA	5.6 + m
OTX	5.6 + m
OTZ	5.6 + m

Table F-2. LSI-1 Execution Time Algorithms (Cont'd)

CONDITIONAL WORD	
RDA	7.2 + m
RDAM	10.4 + m
RDX	7.2 + m
RDXM	10.4 + m
WRA	7.2 + m
WRX	7.2 + m
WRZ	7.2 + m
UNCONDITIONAL BYTE	
IBA	7.2 + m
IBAM	8.8 + m
IBX	7.2 + m
IBXM	8.8 + m
CONDITIONAL BYTE	
RBA	10.4 + m
RBAM	12.0 + m
RBX	10.4 + m
RBXM	12.0 + m
BLOCK	
BIN	11.2 + 2m + (7.2 + m) w
BOT	11.2 + 2m + (7.2 + m) w
AUTOMATIC	
AIN	23.2 + 2m + 3m'
AIN (Under Interrupts)	20.0 + 2m + 3m'
AOT	23.2 + 3m + 2m'
AOT (Under Interrupts)	20.0 + 3m + 2m'
AIB	23.2 + 2m + 3m'
AIB (Under Interrupts)	20.0 + 2m + 3m'
AOB	23.2 + 3m + 2m'
AOB (under Interrupts)	20.0 + 3m + 2m'

F.4 LSI-2 EXECUTION TIME ALGORITHMS

The LSI-2 execution time algorithms are listed in table F-3. The algorithms are partitioned by class and subclass as in table F-2.

The Memory Reference instruction address calculation times precede the instruction execution algorithms. Note that four different sets of address calculations are provided. The list of Memory Reference instructions have algorithms which list A_1, A_2, A_3, or A_4. The appropriate address calculation variable should be used as indicated.

The Stack instruction address calculation times precede the Stack instruction execution algorithms. Note that three different sets of address calculations are provided. The list of Stack instructions have algorithms which list S_1, S_2, or S_3. The appropriate address calculation variable should be used as indicated.

All Memories may be interleaved to achieve higher transfer rates. Core 1600 and SC1200 may be interleaved 100 percent to achieve twice the data transfer rate of a single memory module. Core 1200 and Core 980 may be interleaved to achieve a maximum transfer rate of 171 and 163 percent, respectively, of a single memory module. Interleaving is always effective for DMA operation.

Overlapping is effective for LSI-2 as indicated by the execution time equations. Terms of the form n/RO or m/WO mean that the larger of the two times indicated are to be used. When overlapping is achieved by alternate memory accesses in different memory modules, the overhead times are masked and the effective RO and WO become zero except for Core 980 and Core 1200 which have an overhead time even when interleaved.

As in table F-2, numerous instructions have several times listed to define variations of an instruction. The symbols i, n, and W are described in paragraph F.3.

Table F-3. LSI-2 Execution Time Algorithms

MEMORY REFERENCE CLASS

PROCESSOR MODE	ADDRESSING MODE	A_1	A_2
WORD	direct scratchpad	RA + 700/RO	RA + 800/RO
	direct relative forward	RA + 700/RO	RA + 800/RO
	direct relative backward	RA + 850/RO	RA + 950/RO
	direct indexed	RA + 700/RO	RA + 800/RO
	indirect scratchpad	2RA + 700/RO + 400/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 500/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 400/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 500/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 400/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 500/RO + (RA + 400/RO) (i-1)
	indirect relative indexed	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RP + (RA + 400/RO) (i-1)
BYTE	direct scratchpad	RA + 1000/RO	RA + 1100/RO
	direct relative	RA + 700/RO	RA + 800/RO
	direct indexed	RA + 1000/RO	RA + 1100/RO
	indirect scratchpad	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RO + (RA + 400/RO) (i-1)
	indirect indexed	2RA + 700/RO + 900/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)

PROCESSOR MODE	ADDRESSING MODE	A_3	A_4
WORD	direct scratchpad	RA + 1000/RO	RA + 1300/RO
	direct relative forward	RA + 1000/RO	RA + 1300/RO
	direct relative backward	RA + 1150/RO	RA + 1450/RO
	direct indexed	RA + 1000/RO	RA + 1300/RO
	indirect scratchpad	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)
	indirect relative indexed	2RA + 700.RO + 1200/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1500/RO + (RA + 400/RO) (i-1)
BYTE	direct scratchpad	RA + 1300/RO	RA + 1600/RO
	direct relative	RA + 1000/RO	RA + 1300/RO
	direct indexed	RA + 1300/RO	RA + 1600/RO
	indirect scratchpad	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1300/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1300/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1300/RO + (RA + 400/RO) (i-1)
	indirect indexed	2RA + 700/RO + 1200/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1500/RO + (RA + 400/RO) (i-1)

A_1 is used with ADD, SUB, AND, IOR, XOR, EMA, LDA, LDX, CMS and IMS.

A_2 is used with STA, STX and JST.

A_3 is used by JMP only.

A_4 is used by SCM only.

ARITHMETIC
ADD
SUB

LOGICAL
AND
IOR
XOR

$A_1 + RA + (400/RO)$

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

DATA TRANSFER	
LDA	A_1 + RA + 400/RO
LDX	A_1 + RA + 400/RO
STA	A_2 + WA + 250/RO
STX	A_2 + WA + 250/RO
EMA	A_1 + RA + 500/RO* + WA + 550/WO
PROGRAM TRANSFER	
JMP	A_3
JST (Non-Interrupt)	A_2 + WA + 550/WO
JST (Interrupt)	A_2 + WA + 700/WO
IMS	A_1 + RA + (500/RO)* + WA + 700/RO ≠ 0 in line no skip or + 1450/RO = 0 in line skip or + 850/RO ≠ 0 interrupt no echo or + 1600/RO = 0 interrupt echo
SCM	A_4 + RA + 550/RO + (RA + 1600/RO)(w-1)
CMS	A_1 + RA + 550/RO for A < Y or + 850/RO for A = Y or +1150/RO for A > Y

DOUBLE WORD MEMORY REFERENCE CLASS

DVD	2RA + 1000/RO + (RA + 400/RO) i + (2950 + 450n)/RO
MPY	2RA + 1000/RO + (RA + 400/RO) i + (3100** + 600n)/RO
NRM	2RA + 1000/RO + (RA + 400/RO) i + (1400 + 600n)/RO + WA + 1750/WO

STACK CLASS

ADDRESSING MODE	S_1	S_2	S_3
direct access	3RA + 2(400/RO) + 550/RO	S_1 + 100	S_1 + 300
indexed access	3RA + 2(400/RO) + 850/RO	S_1 + 100	S_1 + 300
auto-postincrement or auto-predecrement	3RA + 2(400/RO) + 500/RO* + WA + 400/WO	S_1 + 100	S_1 + 300

*not effected by interleave

S_1 is used with ADDS, SUBS, ANDS, IORS, XORS, EMAS, LDAS, LDXS, CMSS and IMSS.
S_2 is used with STAS, STXS, and JSTS.
S_3 is used by JMPS and SLAS.

ARITHMETIC ADDS SUBS **LOGICAL** ANDS IORS XORS	S_1 + RA + 400/RO
DATA TRANSFER	
LDAS	S_1 + RA + 400/RO
LDXS	S_1 + RA + 400/RO
STAS	S_2 + WA + 250/RO
STXS	S_2 + WA + 250/RO
EMAS	S_1 + RA + 500/RO* + WA + 550/WO
PROGRAM TRANSFER	
JMPS	S_3
JSTS	S_2 + WA + 550/WO
IMSS	S_1 + RA + 500/RO* + WA + 700/RO ≠ 0 in line, no skip or + 1450/RO = 0 in line, skip or + 850/RO ≠ 0 interrupt, no echo or + 1600/RO = interrupt, echo
CMSS	S_1 + RA + 550/RO A < Y or + 850/RO A = Y or + 1150/RO A > Y
STACK CONTROL	
SLAS	S_3

BYTE IMMEDIATE CLASS

AAI	RA + 1000/RO
AXI	RA + 700/RO
SAI	RA + 1000/RO
SXI	RA + 700/RO
CAI	RA + 1000/RO skip
CXI	RA + 850/RO no skip
LAP	RA + 700/RO
LXP	RA + 700/RO
LAM	RA + 700/RO
LXM	RA + 700/RO

* Not Affected By Interleave
** +300 for Negative Multiplier

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

CONDITIONAL JUMP CLASS	
MICROCODED JOC	
ARITHMETIC JAG JAL JAM JAP JAZ JXN JXZ	RA + 700/RO = No Jump RA + 1000/RO = Jump
CONTROL JOR JOS JSR JSS	

SHIFT CLASS	
ARITHMETIC SHIFTS ALA ALX ARA ARX	
LOGICAL SHIFTS LLA LLX LRA LRX	RA + 1150 + 150n/RO
ROTATE SHIFTS RLA RLX RRA RRX	

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

DOUBLE REGISTER LOGICAL SHIFTS LLL LLR DOUBLE REGISTER ROTATE SHIFTS LRL LRR	RA + 2350 + 150n/RO

REGISTER CHANGE CLASS	
A REGISTER CHANGE ZAR ARP ARM CAR NAR IAR DAR	RA + 1000/RO
X REGISTER CHANGE ZXR XRP XRM CXR NXR IXR DXR	RA + 1000/RO
OVERFLOW REGISTER CHANGE SOV ROV COV	RA + 850/RO
SAO SXO LAO LXO BAO BXO	RA + 1300 + 150n/RO n is number of bits away from 0 to 15

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

REGISTER CHANGE CLASS (Cont'd)	
MULTI-REGISTER CHANGE	
ZAX	RA + 1300/RO
AXP	RA + 1300/RO
AXM	RA + 1300/RO
TAX	RA + 1000/RO
TXA	RA + 1000/RO
EAX	RA + 1300/RO
ANA	RA + 1000/RO
ANX	RA + 1000/RO
NRA	RA + 1000/RO
NRX	RA + 1000/RO
CAX	RA + 1000/RO
CXA	RA + 1000/RO
NAX	RA + 1300/RO
NXA	RA + 1300/RO
IAX	RA + 1000/RO
IXA	RA + 1000/RO
IPX	RA + 1000/RO
DAX	RA + 1000/RO
DXA	RA + 1000/RO
BCA, BCX, BSA, BSX	RA + 1300/RO
EIX	RA + 500/RO + normal time of instruction executed
CONSOLE REGISTER	
ICA, ICX, ISA, ISX, OCA, OCX	RA + 1600/RO

CONTROL CLASS	
PROCESSOR CONTROLS	
HLT (STOP), NOP	RA + 1150/RO
MODE CONTROLS	
SBM, SWM	RA + 1000/RO

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

STATUS CONTROLS	
SIA, SIX, SIN, SOA, SOX	RA + 1600/RO
INTERRUPT CONTROLS	
CID, CIE	RA + 1600/RO
DIN, EIN	RA + 850/RO
PFE, PFE, TRP	RA + 1600/RO

INPUT/OUTPUT CLASS	
CONTROL	
SEN	RA + 1550/RO no skip RA + 1900/RO skip
SEA	RA + 1600/RO
SEL	RA + 1600/RO
SEX	RA + 1600/RO
SSN	RA + 1900/RO no skip RA + 1700/RO skip
UNCONDITIONAL WORD	
INA, INAM, INX, INXM, OTA, OTX, OTZ	RA + 1600/RO
CONDITIONAL WORD	
RDA, RDAM, RDX, RDXM, WRA, WRX, WRZ	RA + 2050/RO successful RA + 2000/RO unsuccessful repeat period
UNCONDITIONAL BYTE	
IBA, IBAM, IBX, IBXM	RA + 1600/RO

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

CONDITIONAL BYTE	
RBA, RBAM, RBX, RBXM	RA + 2050/RO successful RA + 2000/RO unsuccessful repeat period
BLOCK	
BIN	2RA + 400/RO + 1550/RO + WA + 850/WO + (WA + 2000/WO) (W-1)
BOT	3RA + 2 (400/RO) + 1300/RO + + (RA + 2050/RO) (W-1)
AUTOMATIC	
AIN/AIB	3RA + 3WA + 400/RO + 800/RO + 500/RO* + 550/WO + 1700/WO + 550/WO if inline, or + 400/WO if interrupt
AOT/AOB	4RA + 2WA + 400/RO + 800/RO + 500/RO* + 2(550/WO) + 1750/RO inline, or + 1600/RO if interrupt

NOTE

Time given assuming device sense response is present. If not present, BIN and BOT retest for ready every 850 ns.

* Not Affected By Interleave
** (1050/WO) if WC = 0

F.5 ALPHA LSI FAMILY INSTRUCTION EXECUTION TIMES

The execution times of the ALPHA LSI instruction set is listed in table F-7. The Memory Reference instruction address calculation times for the LSI-1 and LSI-2 are listed in tables F-4 and F-5, respectively. The LSI-2 Stack Instruction Address calculation times are listed in table F-6.

F.6 MAXIMUM I/O TRANSFER RATES

The maximum I/O transfer rates for the LSI-1 and LSI-2 computers are listed in table F-8.

Table F-4. LSI-1 Memory Reference Instruction Address Calculation Times

DIRECT SCRATCHPAD	2.2
DIRECT RELATIVE	2.2
DIRECT INDEXED	3.8
INDIRECT SCRATCHPAD	3.8i
INDIRECT RELATIVE	1.6 + 3.8i
INDIRECT INDEXED	1.6 + 3.8i

Table F-5. LSI-2 Memory Reference Instruction Address Calculation Times

MEMORY TYPE	PROCESSOR MODE	ADDRESSING MODE	A_1	A_2	A_3	A_4
CORE 1600	WORD	direct scratchpad	1.6	1.6	1.6	1.75
		direct relative forward	1.6	1.6	1.6	1.75
		direct relative backward	1.6	1.6	1.6	1.9
		direct indexed	1.6	1.6	1.6	1.75
		indirect scratchpad	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)
		indirect relative forward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)
		indirect relative backward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)
		indirect indexed	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
	BYTE	direct scratchpad	1.6	1.6	1.75	2.05
		direct relative	1.6	1.6	1.6	1.75
		direct indexed	1.6	1.6	1.75	2.05
		indirect scratchpad	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
		indirect relative forward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
		indirect relative backward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
		indirect indexed	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.55 + 1.6 (i-1)
CORE 1200	WORD	direct scratchpad	1.2	1.2	1.4	1.7
		direct relative forward	1.2	1.2	1.4	1.7
		direct relative backward	1.25	1.35	1.55	1.85
		direct indexed	1.2	1.2	1.4	1.7
		indirect scratchpad	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)
		indirect indexed	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
	BYTE	direct scratchpad	1.4	1.5	1.7	2.0
		direct relative	1.2	1.2	1.4	1.7
		direct indexed	1.4	1.5	1.7	2.0
		indirect scratchpad	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
		indirect indexed	2.5 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.8 + 1.2 (i-1)	3.1 + 1.2 (i-1)
CORE 980	WORD	direct scratchpad	1.08	1.18	1.38	1.68
		direct relative forward	1.08	1.18	1.38	1.68
		direct relative backward	1.23	1.33	1.53	1.83
		direct indexed	1.08	1.18	1.38	1.68
		indirect scratchpad	2.06 + .98 (i-1)	2.06 + .98 (i-1)	2.16 + .98 (i-1)	2.46 + .98 (i-1)
		indirect relative forward	2.06 + .98 (i-1)	2.06 + .98 (i-1)	2.16 + .98 (i-1)	2.46 + .98 (i-1)
		indirect relative backward	2.06 + .98 (i-1)	2.06 + .98 (i-1)	2.16 + .98 (i-1)	2.46 + .98 (i-1)
		indirect indexed	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
	BYTE	direct scratchpad	1.38	1.48	1.68	1.98
		direct relative	1.08	1.18	1.38	1.68
		direct indexed	1.38	1.48	1.68	1.98
		indirect scratchpad	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
		indirect relative forward	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
		indirect relative backward	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
		indirect indexed	2.36 + .98 (i-1)	2.46 + .98 (i-1)	2.66 + .98 (i-1)	2.96 + .98 (i-1)
SC1200	WORD	direct scratchpad	1.2	1.3	1.5	1.8
		direct relative forward	1.2	1.3	1.5	1.8
		direct relative backward	1.35	1.45	1.65	1.95
		direct indexed	1.2	1.3	1.5	1.8
		indirect scratchpad	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.7 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.7 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.7 + 1.2 (i-1)
		indirect indexed	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-10	3.0 + 1.2 (i-1)
	BYTE	direct scratchpad	1.5	1.6	1.8	2.1
		direct relative	1.2	1.3	1.5	1.8
		direct indexed	1.5	1.6	1.8	2.1
		indirect scratchpad	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-1)	3.0 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-1)	3.0 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-1)	3.0 + 1.2 (i-1)
		indirect indexed	2.6 + 1.2 (i-1)	2.7 + 1.2 (i-1)	2.9 + 1.2 (i-1)	3.2 + 1.2 (i-1)

A_1 is used with ADD, SUB, AND, IOR, XOR, EMA, LDA, LDX, CMS and IMS.

A_2 is used with STA, STX and JST.

A_3 is used by JMP only.

A_4 is used by SCM only.

Table F-6. Stack Instruction Address Calculation Times

MEMORY TYPE	ADDRESSING MODE	S_1	S_2	S_3
CORE 1600	direct access	4.8	4.9	5.1
	indexed access	4.8	4.9	5.1
	auto-postincrement or auto-predecrement	6.4	6.5	6.7
CORE 1200	direct access	3.6	3.7	3.9
	indexed access	3.65	3.75	3.95
	auto-postincrement or auto-predecrement	4.8	4.9	5.1
CORE 980	direct access	2.94	3.04	3.24
	indexed access	3.19	3.29	3.49
	auto-postincrement or auto-predecrement	3.92	4.02	4.22
SC 1200	direct access	3.6	3.7	3.9
	indexed access	3.75	3.85	4.05
	auto-postincrement or auto-predecrement	4.8	4.9	5.1

S_1 is used with ADDS, SUBS, ANDS, IORS, XORS, EMAS, LDAS, LDXS, DMSS and IMSS.

S_2 is used with STAS, STXS, and JSTS.

S_3 is used by JMPS and SLAS.

Table F-7. ALPHA LSI Family Instruction Execution Times

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
MEMORY REFERENCE					
Arithmetic					
ADD	A + 7	A_1 + 1.6	A_1 + 1.2	A_1 + 0.98	A_1 + 1.2
ADDB					
SUB	A + 7				
SUBB					
Logic,					
AND	A + 7				
ANDB					
IOR	A + 7				
IORB					
XOR	A + 7				
XORB					
Data Transfer					
LDA	A + 5.4				
LDAB					
LDX	A + 5.4				
LDXB					
STA	A + 5.2	A_2 + 1.6	A_2 + 1.2	A_2 + 0.98	A_2 + 1.2
STAB					
STX	A + 5.2				
STXB					
EMA	A + 9	A_1 + 3.2	A_1 + 2.4	A_1 + 1.96	A_1 + 2.4
EMAB		A_1 + 3.2	A_1 + 2.4	A_1 + 1.96	A_1 + 2.4
Program Transfer					
CMS	A + 13.4	A_1 + 1.6	A_1 + (1.2 or 1.55)	A_1 + (0.98 or 1.53)	A_1 + (1.2 or 1.65)
CMSB		A_1 + 1.6	A_1 + (1.2 or 1.55)	A_1 + (0.98 or 1.53)	A_1 + (1.2 or 1.65)
IMSN	A + 10.6	A_1 + (3.0 or 3.3)	A_1 + (2.2 or 2.85)	A_1 + (1.86 or 2.61)	A_1 + (2.1 or 2.85)
IMSI		A_1 + (3.0 or 3.45)	A_1 + (2.25 or 3.0)	A_1 + (2.01 or 2.76)	A_1 + (2.25 or 3.0)
JMP	A + 4.8	A_3	A_3	A_3	A_3
JSTN	A + 8.4	A_2 + 1.6	A_2 + 1.2	A_2 + 0.98	A_2 + 1.2
JSTI	A + 6.8	A_2 + 1.6	A_2 + 1.2	A_2 + 0.98	A_2 + 1.2
SCM	A + 13.4	A_4 + (1.6 + 2.05W)	A_4 + (1.2 + 2.0W)	A_4 + (0.98 + 1.98W)	A_4 + (1.2 + 2.1W)
SCMB		A_4 + (1.6 + 2.05W)	A_4 + (1.2 + 2.0W)	A_4 + (.098 + 1.98W)	A_4 + (1.2 + 2.1W)
DOUBLE WORD MEMORY REFERENCE					
DVD	3.8i + 120.2	13.35	12.74	12.44	12.9
MPY	3.8i + 112.2	15.75	15.1	14.84	15.3
NRM1	3.8i + 19.8 + 9.6n	7.05 + .6n	6.35 + .6n	6.07 + .6n	6.55 + .6n
NRM2	3.8i + 23 + 9.6n				

COMPUTER AUTOMATION, INC.

Table F-7. ALPHA LSI Family Instruction Set Execution Times

MNEMONIC	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
		STACK		
Arithmetic				
ADDS				
SUBS				
Logic				
ANDS	$S_1 + 1.6$	$S_1 + 1.2$	$S_1 + 0.98$	$S_1 + 1.2$
IORS				
XORS				
Data Transfer				
LDAS				
LDXS				
STAS	S + 1.6	S + 1.2	S + 0.98	S + 1.2
STXS				
EMAS	S + 3.2	S + 2.4	S + 1.96	S + 2.4
Program Transfer				
CMSS	$S_1 + 1.6$	$S_1 + (1.2$ or $1.55)$	$S_1 + (0.98$ or $1.53)$	$S_1 + (1.2$ or $1.65)$
IMSS	$S_1 + (3.0$ or $3.3)$	$S_1 + (2.2$ or $2.85)$	$S_1 + (1.86$ or $2.61)$	$S_1 + (2.1$ or $2.85)$
JMPS	S_3	S_3	S_3	S_3
JSTS	$S_2 + 1.6$	$S_2 + 1.2$	$S_2 + 0.98$	$S_2 + 1.2$
Stack Control				
SLAS	S_3	S_3	S_3	S_3

Table F-7. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
BYTE IMMEDIATE					
AAI	5.4	1.6	1.4	1.38	1.5
AXI	5.4	1.6	1.2	1.08	1.2
SAI	5.4	1.6	1.4	1.38	1.5
SXI	5.4	1.6	1.2	1.08	1.2
CAI	7	1.6	1.25 or 1.4	1.23 or 1.38	1.35 or 1.5
CXI	7	1.6	1.25 or 1.4	1.23 or 1.38	1.35 or 1.5
LAP	5.4	1.6	1.2	1.08	1.2
LXP	5.4	1.6	1.2	1.08	1.2
LAM	5.4	1.6	1.2	1.08	1.2
LXM	5.4	1.6	1.2	1.08	1.2
CONDITIONAL JUMP					
Microcoded					
JOC1	15				
JOC2	7				
Arithmetic					
JAG	7				
JAP	7				
JAZ	7				
JAN	7				
JAL	7	1.6	1.2 or 1.4	1.08 or 1.38	1.2 or 1.5
JAM	7				
JXZ	7				
JXN	7				
Control					
JOR	7				
JOS	7				
JSR	7				
JSS	7				
SHIFT					
Single Register					
Arithmetic Shifts					
ALA	3.8 + 1.6n	1.6 + .15n	1.55 + .15n	1.53 + .15n	1.65 + .15n
ALX					
ARA					
ARX					

COMPUTER AUTOMATION, INC.

Table F-7. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
SHIFTS (Cont'd)					
Logical Shifts LLA LLX LRA LRX Rotate Shifts RLA RLX RRA RRX	3.8 + 1.6n	1.6 + .15n	1.55 + .15n	1.53 + .15n	1.65 + .15n
Double Register Logical LLL LLR LRL LRR	3.8 + 3.2n	2.8 + .15n	2.75 + .15n	2.73 + .15n	2.85 + .15n
REGISTER CHANGE					
Accumulator ARM ARP CAR DAR IAR NAR ZAR Index ZXR XRP XRM CXR NXR IXR DXR	5.4	1.6	1.4	1.38	1.5

COMPUTER AUTOMATION, INC.

Table F-7. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
REGISTER CHANGE					
Overflow					
SOV, ROV, COV	5.4	1.6	1.25	1.23	1.35
SAO, SXO, LAO, LXO	7	1.75	1.7	1.68	1.8
BAO, BXO	5.4 + 1.6n	1.75 + 15n	1.7 + 1.5n	1.68 + 15n	1.8 + 15n
Multi-Register					
ZAX					
AXP	7	1.75	1.7	1.68	1.8
AXM					
TAX	5.4	1.6	1.4	1.38	1.5
TXA	5.4	1.6	1.4	1.38	1.5
EAX	8.6	1.75	1.7	1.68	1.8
ANA	5.4				
ANX	5.4				
NRA	7.0				
NRX	7.0	1.6	1.4	1.38	1.5
CAX	5.4				
CXA	5.4				
NAX	5.4	1.75	1.7	1.68	1.8
NXA	5.4	1.75	1.7	1.68	1.8
IAX, IXA, IPX, DAX, DXA	5.4	1.6	1.4	1.38	1.5
BCA					
BCX		1.75	1.7	1.68	1.8
BSA					
BSX					
EIX		1.6	1.2	0.98	1.2
Console Register					
ICA, ICX, ISA, ISX, OCA, OCX	6.2	2.05	2	1.98	2.1

COMPUTER AUTOMATION, INC.

Table F-7. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
REGISTER CHANGE (Cont'd)					
IAH, IIH, IMH, IPH, IXH, OAH, OLH, OMH, OPH, OXH	Indefinite				
CONTROL					
Processor					
NOP, HLT, STOP	5.4	1.6	1.55	1.53	1.65
WAIT	Indefinite	Indefinite	Indefinite	Indefinite	Indefinite
Mode Control					
SBM	5.4	1.6	1.4	1.38	1.5
SWM	5.4	1.6	1.4	1.38	1.5
Status					
SIA, SIN, SIX, SOA, SOX	6.2	2.05	2	1.98	2.1
Interrupts					
EIN	5.4	1.6	1.25	1.23	1.35
DIN	7	1.6	1.25	1.23	1.35
CIE, CID, PFE, PFD, TRP	6.2	2.05	2	1.98	2.1
INPUT/OUTPUT					
Control					
SEN	7.8	2 or 2.35	1.95 or 2.3	1.93 or 2.28	2.05 or 2.4
SSN	7.8	2.15 or 2.35	2.1 or 2.3	2.08 or 2.28	2.2 or 2.4
SEL, SEA, SEX	6.2	2.05	2	1.98	2.1

COMPUTER AUTOMATION, INC.

Table F-7. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
INPUT/OUTPUT (Cont'd)					
Unconditional Word					
INA	6.2	2.05	2	1.98	2.1
INAM	7.8				
INX	6.2				
INXM	7.8				
OTA	6.2				
OTX					
OTZ					
Conditional Word					
RDA	7.8	2.45 or 2.5	2.4 or 2.45	2.38 or 2.43	2.5 or 2.55
RDAM	11				
RDX	7.8				
RDAM	11				
WRA	7.8				
WRX					
WRZ					
Unconditional Byte					
IBA	7.8	2.05	2	1.98	2.1
IBAM	9.4				
IBX	7.8				
IBXM	9.4				
Conditonal Byte					
RBA	11	2.45 or 2.5	2.4 or 2.45	2.38 or 2.43	2.5 or 2.55
RBAM	12.6				
RBX	11				
RBXM	12.6				
Block					
BIN	12.4 + 7.6W	5 + 2.25W	4.2 + 2.2W	3.94 + 2.18W	4.3 + 2.2W
BOT	12.4 + 7.8W	4.95 + 2.5W	4.1 + 2.45W	3.64 + 2.43W	4.2 + 2.55W
Automatic					
AIB	25.6	9.95	7.9	6.98	8
AIBI	22.4				
AIN	25.6				
AINI	22.4				
AOB	25.8	10.2	8.15	7.23	8.35
AOBI	22.6	10.05	8	7.08	8.2
AOT	25.8	10.2	8.15	7.23	8.35
AOTI	22.6	10.05	8	7.08	8.2

Table F-8. ALPHA LSI Family Maximum Data Transfer Rates

I/O MODE	LSI-1	LSI-2			
	C1600 C1200 C980 SC1200	C1600	C1200	C980	SC1200
DMA (Non Interleaved)	same as LSI-2	625,000 w/s	833,333 w/s	1,020,000 w/s	833,333 w/s
DMA (Interleaved)	same as LSI-2	1,250,000 w/s	1,409,000 w/s	1,666,666 w/s	1,666,666 w/s
Block In	131,579 w/s	444,444 w/s	454,545 w/s	458,711 w/s	454,545 w/s
Block Out	131,579 w/s	400,000 w/s	408,163 w/s	411,522 w/s	392,156 w/s
Programmed In (Cond) Word	34,247 w/s	112,369 w/s	130,718 w/s	136,040 w/s	124,223 w/s
Byte	34,247 b/s	112,369 b/s	125,896 b/s	130,718 b/s	119,760 b/s
Programmed Out (Cond) Word	34,247 w/s	112,994 w/s	131,578 w/s	135,135 w/s	126,582 w/s
Byte	34,247 b/s	112,994 b/s	126,582 b/s	129,870 b/s	122,222 b/s
Programmed In (Memory)	24,631 w/b/s	71,942 w/b/s	85,106 w/b/s	92,678 w/b/s	82,987 w/b/s
Programmed Out (Memory)	24,631 w/b/s	72,727 w/b/s	82,440 w/b/s	90,570 w/b/s	80,645 w/b/s
DMC In	26,738 w/b/s	63,091 w/b/s	74,627 w/b/s	82,101 w/b/s	73,529 w/b/s
DMC Out	26,738 w/b/s	62,111 w/b/s	73,260 w/b/s	81,766 w/b/s	71,684 w/b/s

w/s = words per seconds
b/s = bytes per seconds
w/b/s = words or bytes per seconds

Appendix G

SOFTWARE SUMMARY

G.1 INTRODUCTION

This appendix contains short usage summaries of the standard system support software offered by Computer Automation, Inc.

Table G-1. Assembler Directives

ABS	Define Absolute Assembly
Asterisk (*)	Comment Line
BAC	Byte Address Constant
CALL	Subroutine Call
DATA	Data Definition (:Hex, 0 Octal, 'ASCII', Address)
END	End of Assembly
ENDC	End of Conditional Assembly
ENT	Subroutine Entry
EQU	Equate Symbol
EXTR	External Reference - Scratchpad
IFF	Conditional Assembly if False
IFT	Conditional Assembly if True
MACH	Set Machine Assembly Mode
NAM	External Name Definition
ORG	Define Origin
Period (.)	Page Eject without Title
REF	External Reference - Pointer
REL	Define Relocatable Assembly
RES	Reserve Storage
RTN	Subroutine Return
SAVE	Save Presently Existing Symbol Table
SET	Set Symbol Redefinable
STOP	Stop
TEXT	'ASCII Message'
TITL	Page Eject with Title
Up Arrow (↑)	Pause
WAIT	Wait for Interrupt

G.2 BOOTSTRAP

To Enter:

Set P = :nFF8
Set WRITE mode
Enter Data } Once per word
Depress M }

To Display:

Set P = :nFF8
Set READ mode
Depress M (Once per word)

Loc	TTY	HSPT
:nFF8	403B	4033
:nFF9	7939	7931
:nFFA	1357	1357
:nFFB	7939	7931
:nFFC	9C00	9C00
:nFFD	0128	0128
:nFFE	3145	3145
:nFFF	0800	0800

G.3 SOFTWARE OPERATION SUMMARY

G.3.1 Autoload

RESET
Enter option control value in Console Sense Register:

Mode \ Device	TTY	HSPT	MT	Cassette	Disk
Load Abs	:0	:1	:2	:3	:4
Load Rel	:8	:9	:A	:B	:C

To relocate (Load Rel), set X = load address
For Load and Go, set SENSE Switch
Ready Device
AUTO

G.3.2 Binary Loader (BLD)

Load BLD
Set P = first location of BLD
To relocate, set X = load address; enter :8 into Sense register
Ready tape in reader (TTY or HSPT)
RUN

G.3.3 Binary Dump/Verify (BDP/VER)

Load BDP/VER
Set P = first location of BDP/VER
Set A = Initial location
Set X = Last location
Enter option control value in Console Sense register:

Mode \ Device		Include EOF TTY	Include EOF HSPT	Suppress EOF TTY	Suppress EOF HSPT
Punch	Abs	:0	:1	:2	:3
	Rel	:8	:9	:A	:B
Verify	Abs	:4	:5	:6	:7
	Rel	:C	:D	:E	:F

For transfer address, set SENSE switch
RUN
If Halt (I = :0802), set A = transfer address, RUN

G.3.4 Object Loader (LAMBDA)

Load LAMBDA
Set P = first location of LAMBDA
Set A = Relocation Bias or zero
Set X = Base Page Bias or zero
Enter option control value in Console Sense register:

Load Mode \ Print Symbols	Defined and Undefined TTY	Defined and Undefined LP	Defined Only TTY	Defined Only LP	Undefined Only TTY	Undefined Only LP	Neither
Library	:0	:1	:2	:3	:4	:5	:6
Unconditional	:8	:9	:A	:B	:C	:D	:E

Ready tape in reader (TTY or HSPT)
RUN

G.3.5 BETA-4 Assembler

Load BETA-4
Set P = :0100
RUN
Enter option control number in Console Sense Register:

Punch Device \ Listing Device	TTY Complete Listing	TTY Error Only	Line Printer Complete Listing	Line Printer Error Only
TTY	:0	:1	:2	:3
HSP	:4	:5	:6	:7

To repeat Pass 2, add :8
To flag out-of-range memory reference instructions, set SENSE switch.
Ready source in reader (TTY or HSPT)
RUN

G.3.6 BETA-8 Assembler

Load BETA-8
Set P = :0100
RUN

Select Options

Enter \ For	SI=	LO=	BO=	SD=	P#=
B	BATCH	Error	Error	Error	Error
L	Error	Error	Library	Error	Error
X	Error	Error Only	N/A	Error	Error
0	Punch EOF	No Listing	No Binary	No Save	1
1	Keyboard	TTY	TTY	Memory	1
2	TTY	D.P.	Error	Unit 0	2
3	HSPT	Cent.	HS	Unit 1	1
4	Card Rdr.	Cent.	TTY	Unit 2	1
5	Card Rdr.	Cent.	TTY	Unit 3	1

G.3.7 OMEGA Conversational Assembler

Load OMEGA
Set P = : 0100
RUN

Command Summary (@ = space):

>AF.	Add keyboard lines to buffer after last line.
>An.	Add keyboard lines to buffer after line n.
>B.	Clear the buffer.
>CInLnOn.	Connect devices.
>CI0.	Punch EOF.
>DF.	Delete the last buffer line.
>Dn.	Delete buffer line n.
>Dn@m.	Delete buffer lines n through m.
>Eh.	Set end of buffer to h (hexadecimal) and intialize OMEGA.
>I.	Initialize OMEGA.
>LF.	List the last buffer line.
>Ln.	List buffer line n.
>Ln@m.	List buffer lines n through m.
>PLT@1@F.	Punch the buffer with leader and trailer.
>PL@n@m.	Punch buffer lines n through m with leader.
>P@n@m.	Punch buffer lines n through m.
>PT@n@m.	Punch buffer lines n through m with trailer.
>Qn.	Set ADD function terminator character to n.
>Rn.	Read source to line n and add to buffer.
>Sn.	Read source to line n-1, add to buffer, and skip line n.
>Sn@m.	Read source to line n-1, add to buffer and skip lines n through m.
>T.	Reset tape line count to zero.
>Tn.	Reset tape line count to n.
>XA.	Assemble.
>XE.	Assemble with ERROR only listing.
>XA2. or XE2.	Assemble starting with Pass 2.
>XLA. or XLE.	Suppress EOF for current assembly.

Device Selection

Input: (I)	Object: (O)	List: (L)
0 = none	0 = none	0 = none
1 = Teletype Keyboard	1 = Teletype Paper Tape	1 = Teletype
2 = Teletype Paper Tape	2 = Error	2 = Data Products Printer
3 = High Speed Paper Tape	3 = High Speed Paper Tape	3 = Centronics Printer
4 = Card Reader		
5 = Memory (assemble)		

G.3.8 Source Tape Preparation Program

Load STP
Set P = first location of STP
RUN

Command Summary (@ = space):

> AF.	Add keyboard lines to buffer after last line.
> An.	Add keyboard lines to buffer after line n.
> B.	Clear the buffer.
> CTT.	Connect teletype reader and teletype punch.
> CRT.	Connect high speed reader and teletype punch.
> CRP.	Connect high speed reader and high speed punch.
> CTP.	Connect teletype reader and high speed punch.
> DF.	Delete the last buffer line.
> Dn.	Delete buffer line n.
> Dn@m.	Delete buffer lines n through m.
> Eh.	Set end of buffer to h (hexadecimal).
> I.	Initialize STP (clear buffer and set T to zero).
> LF.	List the last buffer line.
> Ln.	List buffer line n.
> Ln@m.	List buffer lines n through m.
> PLT@1@F	Punch the buffer with leader and trailer.
> PL@n@m.	Punch buffer lines n through m with leader.
> P@n@m.	Punch buffer lines n through m.
> PT@n@m.	Punch buffer lines n through m with trailer.
> Qn.	Set ADD function termination character to n.
> Rn.	Read tape to line n and add to buffer.
> Sn.	Read tape to line n-1, add to buffer, and skip line n.
> Sn@m.	Read tape to line n-1, add to buffer, and skip lines n through m.
> T.	Reset tape line count to zero.
> Tn.	Reset tape line count to n.

G.3.9 Debug (DBG)

Debug is a 'binary relocatable' program and, as such, may be loaded any place in memory. Transferring to the first location in Debug (enter start location of Debug into the P register and depress RUN) will initialize Debug to accept any of the Debug commands summarized below.

Command Summary (@ = space):

>A.	Display pseudo A register.
>Av.	Set pseudo A register to value v.
>Ba.	Continue breakpoint to location a.
>Ba,b.	Continue breakpoint to location (a or b).
>Ba@b.	Breakpoint from location a to b.
>Ba@b,c.	Breakpoint from location a to location (b or c).
>Ca@b@c.	Copy locations a through b at c and following.
>Fa@b@v.	Fill locations a through b with value v.
>Ia.	Inspect location a.
>Ja.	Jump to location a.
>La@b.	List contents of locations a through b.
>Ma.	Modify memory starting at location a.
>O.	Display pseudo O register.
>Ov.	Set pseudo O register to value v.
>Pa@b.	Print locations a through b.
>Rn.	Display relocation register Rn.
>Rnv.	Set relocation register Rn to value v.
>Sa@b@v.	Search locations a through b for value v.
>Sa@b@v@m.	Search for value v using mask word m.
>T.	Enable console interrupt (TRAP).
>Tn.	Enable console interrupt and enable interrupts
>X.	Display pseudo X register.
>Xv.	Set pseudo X register to value v.

G.3.10 Concordance (CONC)

Load CONC
Set P = : x100 zero
RUN

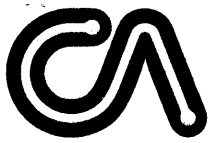
Select Options:

SI=	
R	Repeat listing
B	BATCH
1	Keyboard
2	TTY
3	HSR
4	CR
5	Unit 0
6	Unit 1
7	Unit 2
8	Unit 3

LO=	
L	List
1	TTY
2	D.P.
3	Cent.

G.3.11 OS Command Summary (DOS, MTOS and COS)

	COMMAND	RESPONSE
1.	/ASsign	unit=device [, unit=device...]
2.	/BAtch	device
3.	/BEgin	address [, parameters...]
4.	/CAncel	
5.	/COmment	
6.	/DAte *date	[mm/dd/yy]
7.	/EXec	program-name [, parameters...]
8.	/JOb *date, time	
9.	/LOad	program-name
10.	/LIst *date, time *lu pu	
11.	/NJob *JOB/NJOB time, current time	
12.	/REsume *time	[parameters...]
13.	/STatus *program-name, base page limits, memory limits, flag, time P register, A register, X register, CPU Status	
14.	/TIme *time	[hh:mm:ss]
15.	/TYpe	



ComputerAutomation
Naked Mini® Division
18651 Von Karman, Irvine, Calif. 92664
Tel. 714-833-8830 TWX 910-595-1767