MIP-16

MODULAR INFORMATION PROCESSOR

SYSTEM REFERENCE MANUAL

SA*
SANDERS
ASSOCIATES, INC.

MODULAR INFORMATION PROCESSOR SYSTEM REFERENCE MANUAL FOR MIP-16 SYSTEMS

SANDERS ASSOCIATES, INC.
95 CANAL STREET
NASHUA, NEW HAMPSHIRE

First Edition (February 1972)

Specifications contained herein are subject to change without notice. Such revisions will be reported in subsequent editions or newsletters

Communications should be addressed to:

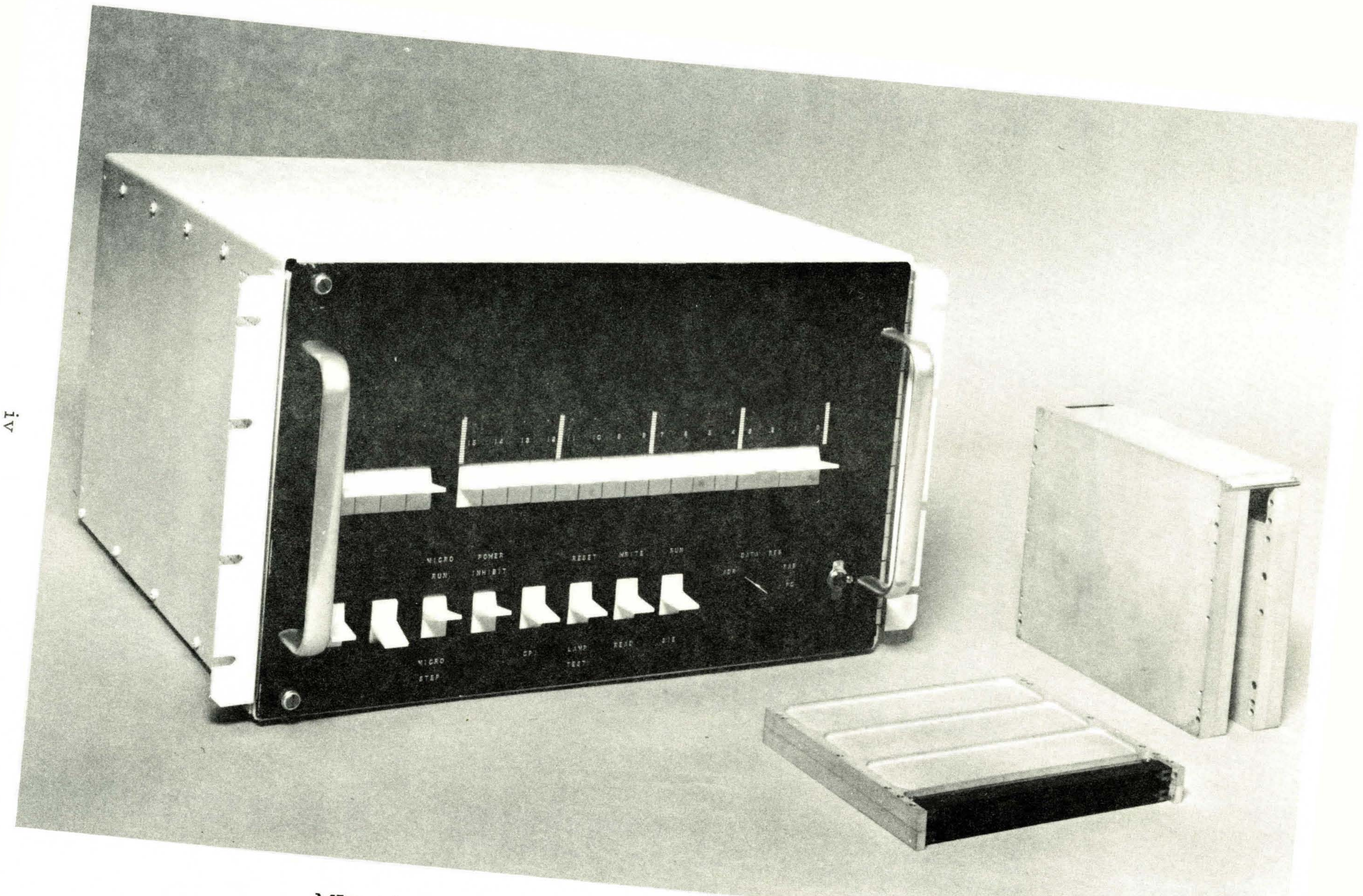
Systems Engineering
Group 1-2706
Sanders Associates, Inc.
95 Canal Street
Nashua, New Hampshire 03060

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES



MIP-16 LAB AND SEVERE ENVIRONMENT COMPUTERS

SECTION 1
INTRODUCTION

1.0 INTRODUCTION

Sanders Associates Modular Information Processor family of 16 bit computers has been designed to meet the needs of real time control, data acquisition, and analysis. Its small size, ruggedness, light weight and low power permit it to be used where these factors are at a premium, yet it provides extremely high performance at low cost. This performance permits programmable software techniques to be applied to applications which have, in the past, required special purpose designs. Areas of potential application include:

Optical Countermeasures (OCM)
Electronic Countermeasures (ECM)
Electronic Support Measures (ESM)
Radar Signal Processing
Engine Control and Health Monitoring
Airframe Data Management
Communication and Control
Navigation Aids
Fire Control
Test Equipment
Antisubmarine Warfare

The MIP-16 family features a modular design which permits a computer to be tailored to fit a particular application. Processor performance, memories, peripherals, and packaging may be selected to provide a cost effective solution to a particular problem.

Both lab and severe environment configurations are available. These are extensible so that a selection of processors, memories and peripherals may be used. The high density severe environment package may be as small as 70 cubic inches and weigh as little as 3 pounds, yet it will operate at temperatures up to 120°C and at vibration levels of 30g's. The lab environment package is a portable, self-contained unit which may be rack or table top mounted.

Two models of the processor are available. The model 116 provides a cost effective processor with a maximum speed of 5 million instructions per second. The model 216 provides a maximum speed of 14 million instructions per second. Both models are software compatible and use the same I/O devices. Both achieve this performance over the full military temperature range and both use off-the-shelf MSI TTL circuit technology to provide high reliability and low cost.

This level of performance is provided by an advanced system architecture utilizing pipelining and instruction fetch overlap. The processors are microprogrammed with an alterable microstore which permits instructions to be added for particular applications.

The MIP-16 family provides facilities for efficient, easy programming, The system architecture has provisions for the efficient implementation of high level languages in an operating system environment. Storage utilization is reduced through the use of a multiregister architecture and reentrant programs.

An instruction set of over 100 instructions is provided with capabilities for operation on byte, single precision fixed and floating point and double precision fixed and floating point data. Interrupt service overhead is minimized by the use of a hardware stack and a priority, vectored, interrupt system.

Programming aids are provided to assist in program development. These include an assembler, mathematical subroutines, diagnostics, and a debugging package.

The MIP-16 family is supported by a range of standard peripherals which include paper tape devices and disk storage. Provisions are made for the addition of a wide range of peripherals of diverse speed and transmission requirements.

The sections of this manual which follow provide comprehensive description of the facilities provided by the MIP-16 family. Additional processor options, peripherals and memory modules will be added in the future.

SECTION 2
SYSTEM ARCHITECTURE

2.0 INTRODUCTION

The MIP-16 family is designed to provide a broad spectrum of capabilities on a modular basis. The general modular architecture is illustrated in figure 2-1. The key system elements that make up a system for a particular application are:

1. Processor
2. Input/Output Facilities
3. Interrupt Facilities
4. Memories
5. Peripherals
6. Control Panel
7. Enclosures

The MIP-16 family provides the user a choice for these key system elements which he can use to configure a system suited for his particular application. In the sequel a description is given of each of these system elements.

2.1 PROCESSOR

The MIP-16 processor is a 16 bit, stored program, general purpose computer utilizing two's complement binary arithmetic. Instructions are variable length and can directly address 32,768 sixteen bit words or 65,536 eight bit bytes. A unified structure is used to address memory and device registers. The processor provides a large, powerful instruction set, 16 general purpose registers, multilevel priority interrupt system and input/output facilities.

The processor is microprogrammed with an extensible microstore. This permits microroutines to be added which are tailored for particular applications.

The MIP-16 processors provide facilities for efficient, easy programming of real time problems. Some of the key features of the processor are discussed below. A detailed description of the instruction set, however, is provided in Section 3 of this manual

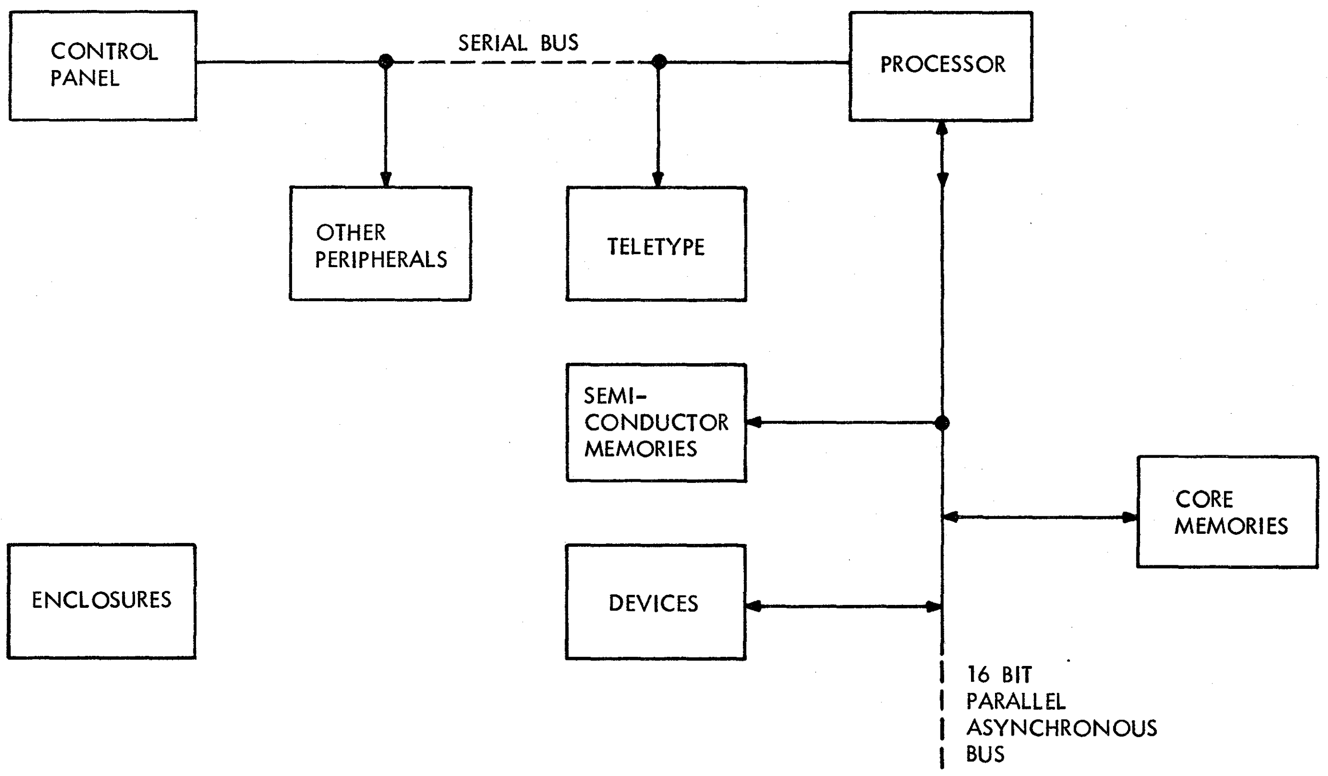


Figure 2-1 General Modular Processor.

2.1.1 PROCESSOR MODELS

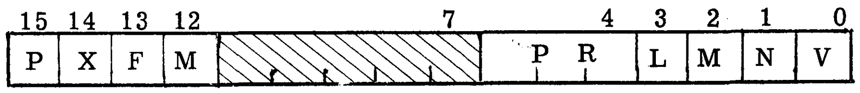
Two models of the processor are available. They utilize the same peripherals, memories, and instruction set but differ in internal organization and speed. The model 116 has a 5 MHz instruction rate and the model 216 has a 14 MHz rate and executes some instructions in fewer microcycles.

2.1.2 REGISTERS

The processor provides 16, 16 bit general purpose registers, a program status register and a program address register. The large number of general registers permit many operations to be performed without reference to memory and improves performance while reducing storage requirements. Register 15 is dedicated for use as a system stack pointer.

2.1.2.1 Program Status Register (PSR)

This sixteen bit register is used to indicate the result of previous operations, processor priority, and holds mask bits for internal interrupts. It is structured in the following format:



Bit 15-P	Priviledged Mode
Bit 14-X	Enables Fixed Point Interrupts
Bit 13-F	Enables Floating Point Interrupts
Bit 12-M	Memory Protect Enable
Bits 11-7	Reserved
Bit 6-4 PR	Processor Priority
Bit 3-L	Link-set if result produced a carry
Bit 2-M	Most-set to most significant bit of result
Bit 1-N	Not zero-set if result was not zero
Bit 0-V	Overflow-set if result produced overflow

When the priviledged mode bit is off an attempt to address peripheral devices or execute system control instructions will cause an internal interrupt. If the memory protect feature is attached and enabled (M=1), an attempt to access a protected location will cause an internal interrupt.

2.1.2.2 Program Address Register (PC)

This 16 bit register is used to address up to 65, 536 bytes for instruction fetch and data access.

2.1.3 INSTRUCTION SET

A total of 98 instructions are provided, broken into the following groups.

26	Fixed Point Arithmetic
11	Logical
15	Shift
17	Load-Store
20	Program Control
9	System Control

This large instruction set reduces the number of instructions required to perform a task and the amount of programming effort required.

Some particularly useful instructions are provided which are not usually found in machines of this class. These include:

a. Double Precision (32 bits)
b. One word immediate
c. Relative branch
d. Multiple Shift
e. Multiply and Divide
f. Load and Store Multiple

The one word immediate instructions permit an eight bit literal to be added to, compared with, or loaded into a register. This saves memory because an eight bit literal is sufficient for most operations of this type. Multiply and Divide can be performed in 1.3 and 2.6 μs respectively on the model 216 and in 3.6 and 7.2 μs on the model 116. Multiple shift instructions minimize the number of instructions required to perform scaling and packing and unpacking functions. Load and Store Multiple instructions are useful for saving and restoring the sixteen general purpose registers during subroutine calls.

2.1.4 ADDRESSING

The smallest unit of addressable data is the eight bit byte. Sixteen bit words and thirty-two bit double words may also be addressed.

Seven different memory addressing methods are available. These include:

a. Indirect Register
b. Indirect Incremented Register
c. Indirect Decremented Register
d. Absolute
e. Immediate
f. Relative
g. Indexed

The utilization of these registers has been carefully selected for maximum frequency use with minimum storage requirements. A sixteen bit address is used which eliminates the problems associated with paging and permits up to 65, 536 bytes to be addressed. Programs can be written which can be relocated without modification of internal addresses. The same addressing methods are used for both memory and peripheral device registers.

2.1.5 PROCESSOR OPTIONS

Options available for the processor include memory protection, fast multiply and additional general purpose register banks.

2.1.5.1 Memory Protection

The address space of 32,768 words is divided into 128 blocks of 256 words. Associated with each block are three bits: read protect, write protect, and write detect. A write detect bit is set if a word is written into its block. This is useful during page swapping to detect when a program has changed one of its blocks. The protect bits may be set only in priviledged mode. A logical block of 2048 words is defined as a group of eight 256 word blocks.

Memory protect is not disabled by priviledged mode. This affords some degree of supervisor self-protection.

Memory protect may be used to indicate blocks of memory that are not available to the system for read - write or write. Memory protect does not inhibit direct memory access. Protection for this is supplied by the supervisor when initiating the access.

2.1.5.2 Fast Multiply

This option performs a 16 x 16 bit multiply in 400 ns. Both sixteen bit rounded and thirty-two bit products are available. No special programming is required to use this option.

2.1.5.3 Register Banks

Up to three additional sets of 16, 16 bit general purpose registers may be installed. Bank selection is performed by the processor priority bits in the PSR. The correspondence is as follows:

Bank	Priority
0	1
1	2
2	5
3	7

The standard model 216 has Banks 0 and 3 installed while the 116 has only Bank 0. Bank 1 provides an extra set of program registers.

The major use of these banks is to reduce the overhead required to save and restore the registers in handling interrupts.

2.2 INTERRUPT FACILITIES

The processor provides a six level priority, vectored interrupt system. A push down stack is used to save machine state information consisting of the PC and PSR. Use of a stack facilitates programming of reentrant interrupt routines.

Multiple priority sources are identified on each level. In the event of the simultaneous request of interrupts the highest priority source on the highest priority level is serviced first. An interrupt request will only be honored if it is at a higher level than the processor priority as determined by the PSR. Table 2-1 lists the sources and their priority levels and transfer vector (T. V.) addresses.

Initiation of interrupt service begins by saving the PSR and PC on the stack. The source is used to address a transfer vector and a new PC and PSR are loaded in that order. The interrupt service routine is then entered. The end of the routine is indicated by a Return From Interrupt (RTI) Instruction which pops the stack and reloads the old PC and PSR.

Typically, the program operates at priority one in the PSR. Upon interrupt from level 6, for example, the new transfer vector will specifiy a new priority of level 6 which will block further interrupts at that level or below while still permitting interrupts at higher levels. When the PSR is restored by the RTI instruction the priority will be reset to one.

If the transfer vector PSR specifies a priority of zero,the old priority will not be changed when the PSR is loaded at the start of the interrupt sequence. This is used by internal interrupts to handle program interrupts such as overflow at the same level as the program which caused it. Of course, an internal interrupt such as power down can specifiy a priority of seven which will disable all external interrupts.

Since the priority is set by the transfer vector, levels may be grouped together. In the example above, the T. V. could specify level seven. This would have the effect of grouping levels 7 and 6 together.

The internal interrupts are so arranged so that only one can be active simultaneously. Since they cannot be masked they will always cause an interrupt which will reset the request. The internal interrupt routine may be reentrant however.

In the event of simultaneous internal and external interrupts, the internal interrupt takes precedence. If its T.V. specifies a new priority below that of the active external level the PC and PSR will be saved again and entry made to the external handler routine. At completion of that, the internal routine will be reentered.

This arrangement permits external interrupt routines to cause internal interrupts since the internal request generated at a lower level is saved before entry to the external routine.

2.3 INPUT-OUTPUT FACILITIES

Two types of data transmissions facilities are available. A high speed parallel asynchronous bus is used to link the processor with memories and high speed devices over short distances. The serial bus is utilized for differential transmission for

TABLE 2-1

INTERRUPT SOURCES AND PRIORITY LEVELS

Source	Source Description	Priority Level	Transfer Vector Address
15	Power Down	8	0000
14	Power Up	8	0004
13	Invalid Instruction	8	0008
12	Priviledged Instruction	8	000C
11	Protection	8	0010
10	I/O Error	8	0014
9	Fixed Point Overflow	8	0018
8	Floating Point Overflow	8	001C
7	Floating Point Underflow	8	0020
6	Floating Point Significance	8	0024
5	Fixed Point Divide	8	0028
4	Floating Point Divide	8	002C
3	Programmed Interrupt	8	0030
7	External 7	7	0040
6	External 6	7	0044
5	External 5	7	0048
4	External 4	7	004C
3	External 3	7	0050
2	External 2	7	0054
1	External 1	7	0058
0	External 0	7	005C
7-0	External 7-0	6	0060-007C
7-0	External 7-0	5	0080-009C
7-0	External 7-0	4	00A0-00BC
15-0	External 15-0	3	00C0-00FC
	Program	2	Used for alternate bank select
	Program	1	Main Program Level
	Retain Priority	0	Used to indicate no change of priority

communication of over long distance in a high noise environment to peripheral devices. Thirty-two interrupt sources are associated with the parallel bus and sixteen with the serial bus.

A unified programming and interrupt structure is used to control these facilities. Devices are addressed simply as memory locations. Transfer time dependence is transparent to the programmer. Identification of an interrupt source is carried out by the hardware and requires no programming assistance.

Detailed descriptions of these facilities are provided in Section 7 of this manual.

2.3.1 PARALLEL ASYNCHRONOUS BUS

This bus provides for transmission of data over a maximum distance of 5 feet. Direct memory access is available at an 8 MHz rate and instruction fetch at 14 MHz rate using queued memories. Asynchronous transmission is used so that memories and devices of differing transmission rates may be attached to the same bus. This permits a 1μ s core memory and a 70 ns bipolar memory to share the same bus facilities. Asynchronous transmission also accommodates the different delays associated with devices located along the fast bus. Total loop delay can be as much as 20 ns which is significant at transmission rates above 5 MHz.

2.3.2 SERIAL BUS

This bus is primarily used to communicate with slower character oriented devices over a maximum distance of fifty feet. It provides for word rates of 250 KHz and byte rates of 350 KHz. The control panel uses this bus for transmission in order to provide for remote location.

For aircraft type system applications this bus may be used for control and program fill of the computer from ground equipment without removal of the computer.

2.4 MEMORIES

Typical application requirements for memories include:

- Non-volatility
- Electrical Programmability
- Speed
- Size
- Cost

Semiconductor and core memories are provided to meet these application requirements. All memories are compatible between different processor models.

2.4.1 SEMICONDUCTOR MEMORIES

Semiconductor read/write memories are provided for use in applications requiring high speed, small to medium size and where volatility is not a problem. The memories are available in both lab and severe environment configurations.

2.4.1.1 1024 WORD, 200 ns, BIPOLAR READ/WRITE MEMORY (MODEL 700)

This memory utilizes 80 ns bipolar integrated circuit memories to provide a cost effective 1024 word memory with the following access times:

	Processor	
Temperature Range	Model 116	Model 216
Commercial	200ns	160ns
Military	250ns	200ns

Typical power dissipation is 35 watts and the memory is mounted on a single card.

2.4.1.2 1024 Word, Instruction Look Ahead Bipolar Read/Write Memory (Model 710)

This memory utilizes 80 ns bipolar integrated circuit memories to provide a high performance 1024 word memory. Instruction fetch times are reduced through the use of two instruction look ahead queuing. Typical power dissipation is 40 watts and the memory is mounted on a single card. Access times are given below.

Temperature Range	Data Fetch Processor Model 116	Data Fetch Processor Model 216	Instruction Fetch Processor Model 116	Instruction Fetch Processor Model 216
Commerical	200ns	160ns	200ns	70ns
Military	250ns	200ns	200ns	70ns

2.4.2 CORE MEMORIES

Core memories are provided for applications requiring medium to large size, medium speed and where non-volatility is required. A military and commercial version are available.

2.4.2.1 8192 Word, 1 μ s, Lab Environment, Core Memory System (Model 750)

This memory is designed specifically for lab application. It consists of a memory controller and a memory unit. One memory controller can control up to 4 memory units for a total capability of 32, 768 words. The unit is packaged on 2 circuit cards and the controller on 1 card.

A reduced 4096 word memory unit is also available.

2.4.2.2 4096 Word, 1 μ s, Severe Environment, Core Memory System(Model 760)

This system is designed specifically for military applications where small size and low power are prime requirements. The memory uses advanced design techniques in order to provide state-of-the-art performance. Low power and extremely wide operating temperatures are key performance features. Power strobing is used to reduce power consumption. At 1 μs the typical power consumption is 25 watts but this may be reduced by operating at a lower speed. At 3 μs, for example, power consumption drops to only 15 watts. The complete system requires only 3 cards. An 8192 word version is also available. It operates at similar power levels and requires only 4 cards.

2.5 PERIPHERALS

A complete line of standard peripherals is available. They are generally configured for lab environment but may be used with a severe environment system by proper I/O bus connections. The serial I/O bus is particularly applicable for this. In this manner a processor installed in a remote location can use peripherals.

Standard peripherals available include the following. Additional peripherals will be added in the future:

ASR-33 Teletype
300 cps Paper Tape Reader
75 cps Paper Tape Punch
Disk File
Real Time Clock

General programming techniques for peripherals are discussed in Section 4. Detailed programming and device descriptions of the peripherals are presented in Section 5.

2.6 CONTROL PANEL

The control panel is designed to control the operation of the computer when located up to fifty feet away in a high noise environment. This is useful in an environment where operator access is limited or where equipment removal is undesirable. The control panel can also read or write into any memory location or device register and can be used as an I/O device itself.

The reader is referred to Section 6 for a fuller discussion of the control panel.

2.7 ENCLOSURES

The MIP-16 family is designed to be used in both lab and severe environments. An optimum solution requires two different packaging concepts, however.

2.7.1 LABORATORY CONFIGURATION

MIP-16 systems intended for lab environments use a standard enclosure shown in Figure 2-2. This can be used to house both expansion modules as well as the central processor itself. This enclosure uses a standard card size of 8 x 11 inches with 160 pin connectors.

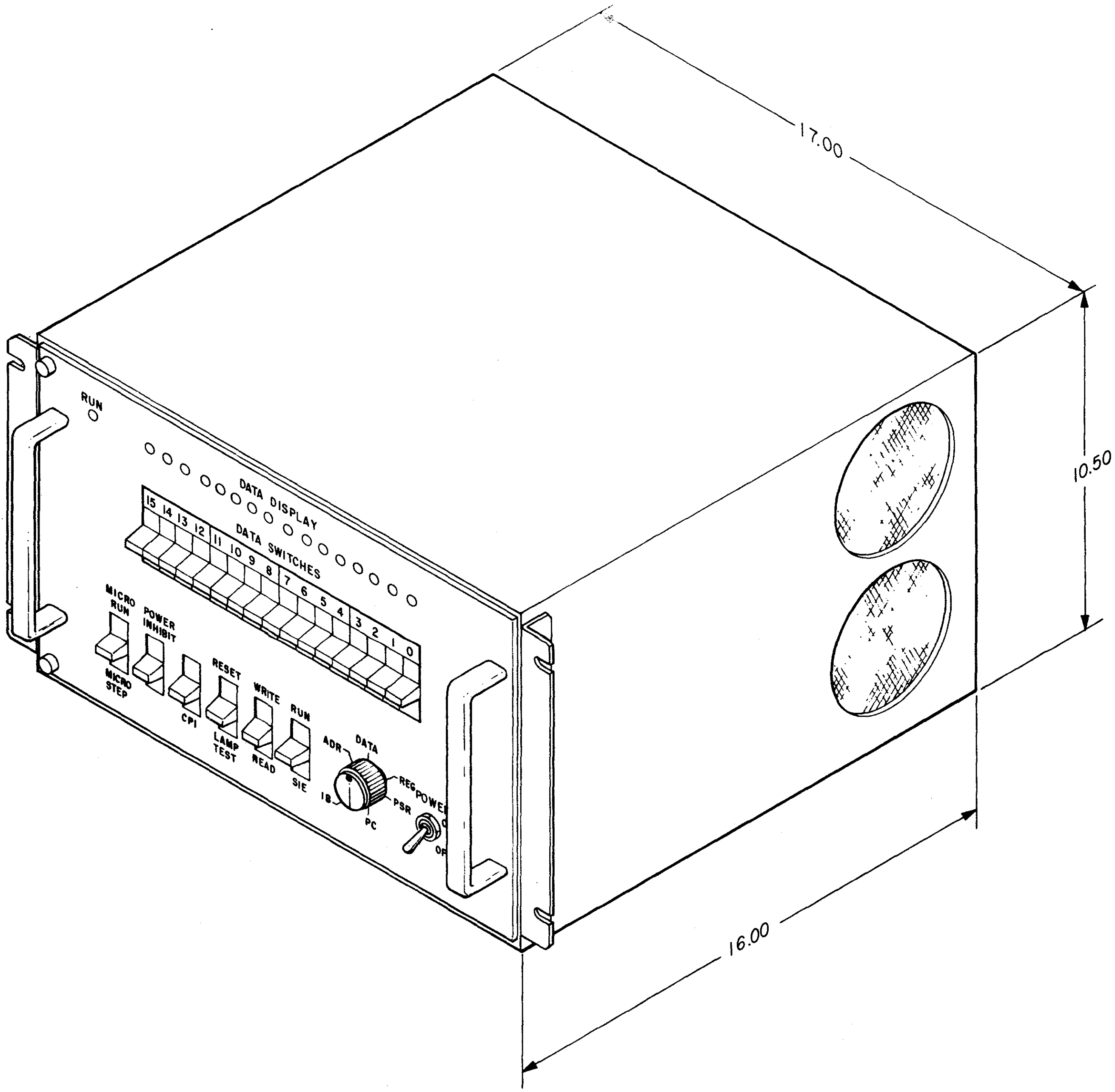


Figure 2-2 Processor Lab Unit.

The unit offers the following features:

Rack or table top mounting
Self contained power supplies and cooling
Lightweight
Portable
Wide range power inputs
Expandable

The enclosure is available in two basic configurations, although others can be provided.

The main frame configuration includes provisions for the following modules:

A model 216 or 116 Processor
Control Panel
Teletype Interface
Paper Tape Reader and Punch Interface
Real Time Clock
Disk File Controller
Four Expansion Cards
Two 1024 Word Bipolar Memory Modules
Core Memory Controller
8192 Word Core Memory

The core memory expansion configuration includes provisions for the following modules:

One Core Memory Controller
One to Four 8192 or 4096 Word Core Memory Units

2.7.2 SEVERE ENVIRONMENT CONFIGURATION

MIP-16 systems intended to be used in a severe environment use a modular enclosure. This enclosure is composed of variable number of double card plug clamshell modules which can be attached together to form a complete assembly. The basic module size is approximately .8 x 6.38 x 7.4 inches as is shown in Figure 2-3. Card size including a 160 pin connector is about 6 x 7 inches. Components are cooled by conduction to the sidewalls where a coldplate or finned heat exchanger may be attached.

The size, weight, temperature range, and operating vibration range depend on the number and type of modules used in the complete assembly. A typical assembly, shown in Figure 2-4, provides a guide, however.

This unit consists of modules and contains a model 116 processor, 4K core memory, and one expansion card. The system physical specifications are as follows:

Size:	2.6 x 6.38 x 7.4 inches (123 cubic inches)
Weight:	5.5 pounds
Power:	40 watts
Vibration:	30g's at 2000cycles
Operating Temperature:	-55°C to +110°C, at sidewall
Cooling:	Conduction

With semiconductor memories smaller systems can be constructed.

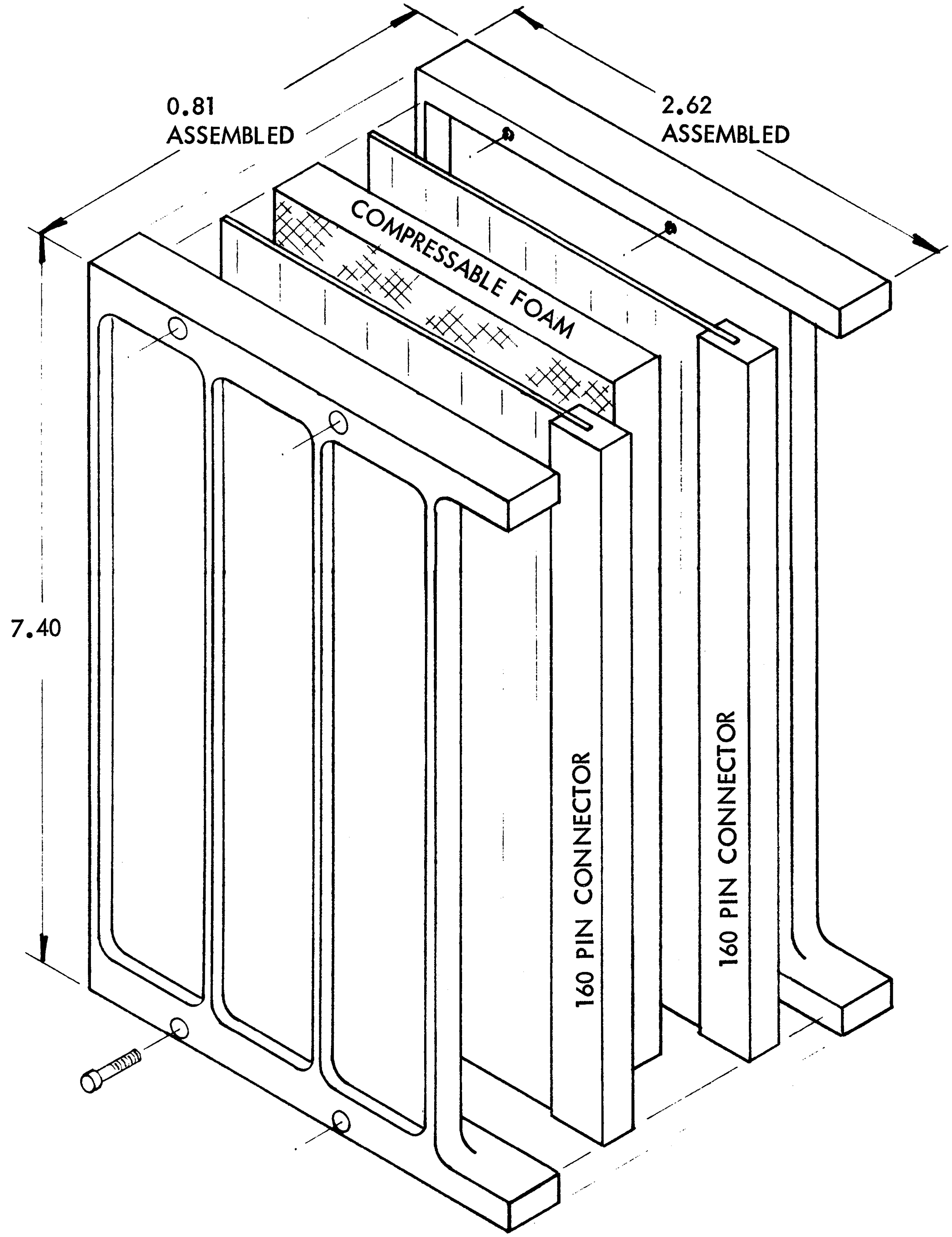


Figure 2-3 Clamshell Module.

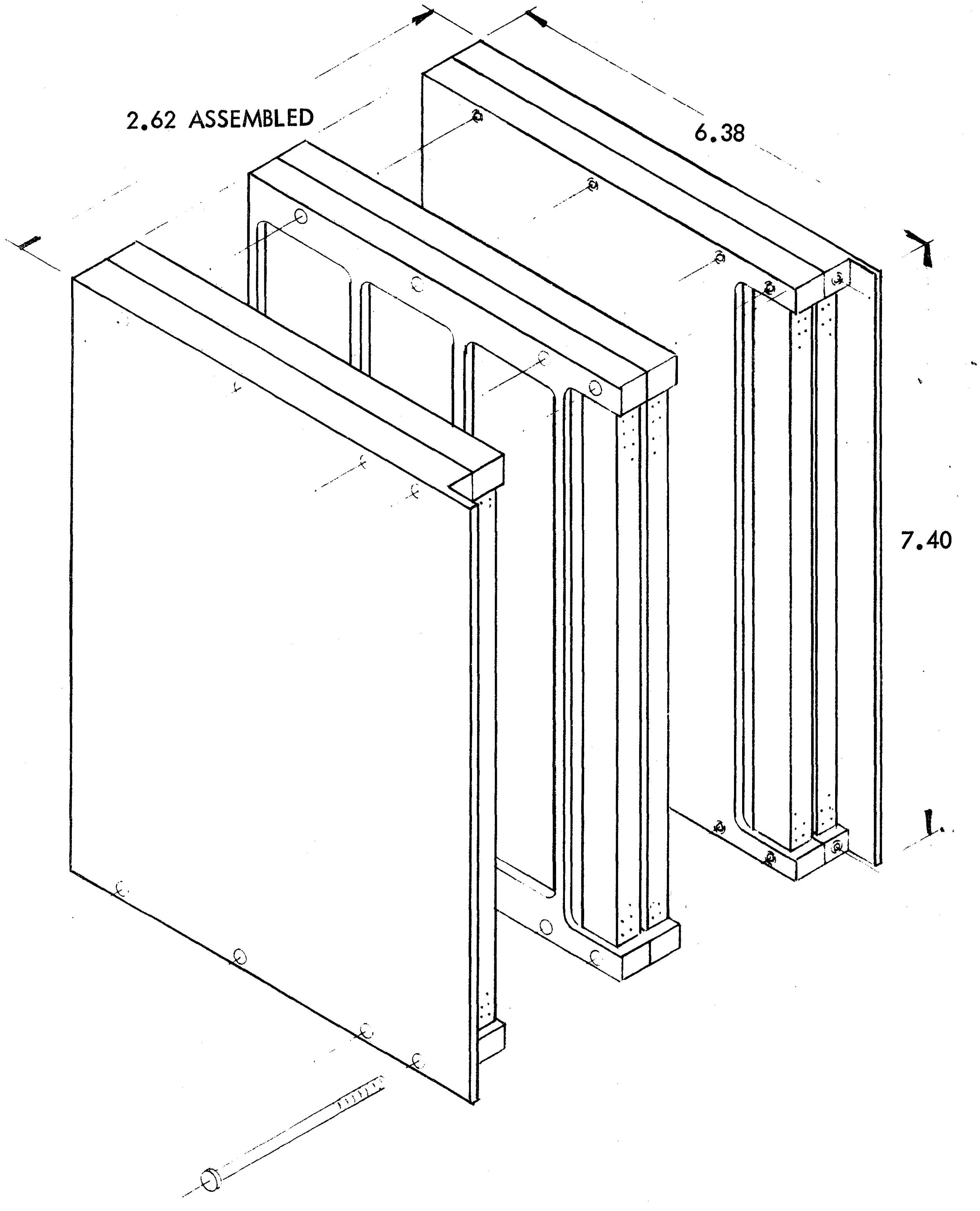


Figure 2-4 Three Module Assembly

SECTION 3

PROGRAMMING GUIDE

3.0 INTRODUCTION

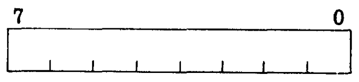
All instructions affecting the processor operations are either one word or two words in length. All instructions must start on an even byte address. There are six types of instructions: fixed-point arithmetic, logical, shifts, load and stores, program control, and system control.

3.1 DATA FORMATS

Data is stored in three types of formats: bytes, words, and double precision words. A word consists of two bytes.

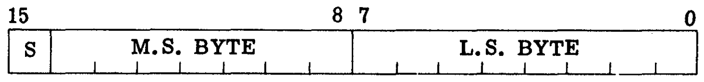
3.1.1 BYTE

A byte contains 8 bits of information. The bits are numbered from right to left with bit 0 considered the least significant bit and bit 7 the most significant bit.



3.1.2 WORD

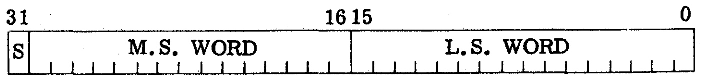
A word consists of two bytes and contains 16 bits which are numbered from 0 to 15. Bit 0 is the least significant (L. S.) bit.



A negative quantity is always stored in two's complement form. When a word is addressed it is always addressed by the most significant (M. S.) byte. The M. S. byte is always an even numbered byte.

3.1.3 DOUBLE PRECISION WORDS

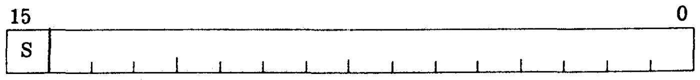
Consist of four bytes of data - 32 bits. The bits are numbered from right to left (0-31) with bit 0 considered the least significant bit.



A negative quantity is represented as a 32 bit two's complement number. A double precision word is addressed by the most significant byte of the M. S. word.

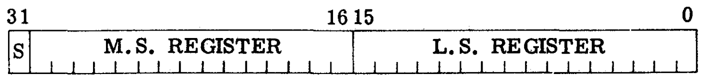
3.1.4 GENERAL PURPOSE REGISTERS

There are 16 general purpose registers (0-15). Each register consists of two bytes (16 bits) and the bits are numbered from 0 to 15. Bit 0 is the least significant bit.



Negative quantities are stored in two's complement form. When a register byte is referenced it is always the least significant (L. S.) byte - bits 0-7. In byte operations where a sign is considered bit 7 is treated as the sign bit.

All double precision operations involve general purpose registers as operands. Each operand is considered as a 32 bit two's complement quantity.



The programmer always references the most significant register, which can either be odd or even.

3.2 INSTRUCTION FORMATS

3.2.1 ONE WORD INSTRUCTIONS

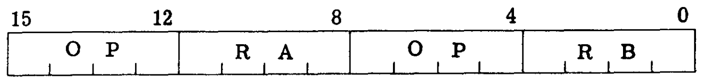
One word instructions are in one of four formats.

3.2.1.1 Register to Register Operations - RR

The assembler will recognize instructions of the form

OP RA, RB

as register to register operations. RA and RB are defined as one of the 16 general registers. The instruction word will contain 2 four bit fields for the operation codes (opcode) and 1 four bit field for each general register:



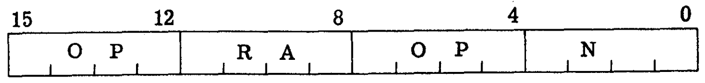
The RA and RB registers usually are the operands with RA being the destination register. With one word Input-Output instructions the definitions of RA and RB changed slightly. Register RA with input instructions is still the destination register but RB now contains the address of the source operand. With output instructions RB contains the address of the destination and RA becomes the source operand.

3.2.1.2 Single Register Operations - RN

The assembler will recognize instructions in the form

OP RA, N

as single register operations. RA is defined as the operand and is one of the 16 general registers. N specifies the number of times the operation is to be performed. The instruction word is similar to register-register operations:

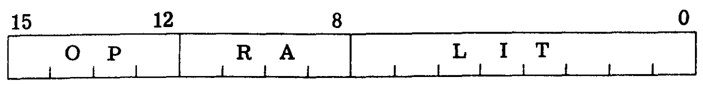


3.2.1.3 Immediate Operations - RI

Instructions of the form

OP RA, LIT

are recognized by the assembler as immediate operand instructions. The instruction word is:



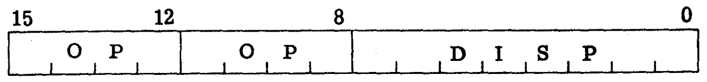
RA is again one of the 16 general registers. LIT is an eight bit literal. Register RA is always the destination if the operation requires one.

3.2.1.4 Relative Operations - RL

The assembler will recognize instructions of the form

```
OP     EXP
```

as a relative instruction and generates a memory word of the following format:



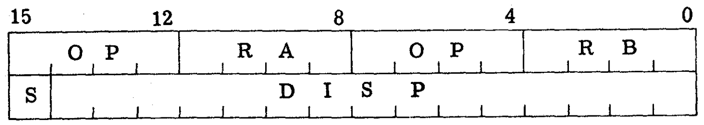
where DISP is the displacement of the expression (EXP) from the program counter (PC). The PC at execution time always points to one instruction ahead. DISP has a range of -126 to +128.

3.2.3 TWO WORD INSTRUCTIONS

The assembler recognizes instructions of the form

```
OP     RA, EXP(RB) or
OP     EXP(RB)
```

as two word instructions and will generate two memory words appearing as:



where RA is the general register affected by the operation and RB and DISP are used to calculate an effective byte address. The value of DISP depends on the value of RB and is either equal to the value of EXP or a displacement of the EXP from the Program Counter. The calculation of the effective byte address is discussed below. For immediate addressing modes in arithmetic logical, input-output, and jump and calls the (RB) field of the source code may be left off - the assembler will assume RB= 0.

3.3 BYTE ADDRESS CALCULATIONS

3.3.1 FOR ARITHMETIC, LOGICAL, AND INPUT-OUTPUT INSTRUCTIONS

If the RB field contains a

0 (immediate addressing) - The second word of the instruction contains the effective data.

1 (relative addressing) - The effective byte address is calculated by adding the contents of the PC (which at execution time is pointing to the second word) to the contents of the second word - DISP. The addition is performed in two's complement arithmetic. EA=(PC) +DISP

2 (absolute addressing) - The contents of the second word is the effective byte address.
EA = DISP

3-15 (indexed addressing) - The sum of the second word of the instruction and the contents of the specified RB register (3-15) constitutes the effective byte address.
EA = DISP +(Register B)

The effective byte address for logical and arithmetic operation must always be on a word boundary (it must be even). If it is odd the least significant bit of the EA is disregarded. The exception is memory byte operations in Load-Store instructions, where the byte address may be either odd or even.

3.3.2 FOR MEMORY JUMPS (JMPM) AND CALLS (CALM)

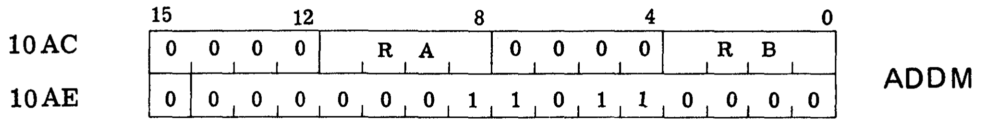
The effective byte address (in this case jump or call address) is calculated in the same manner as above except when the RB field is zero (0). If the RB field is zero a one level indirect jump or call address is calculated. The contents of the program counter and the contents of the second word are summed to form an intermediate address. The contents of this intermediate address is the jump or call address. The value of the RA field in JMPM is zero. EA (JA) = ((PC) + DISP)

3.3.3 LA INSTRUCTION

The above only has one address mode - immediate.

3.3.4 TWO WORD BYTE ADDRESS CALCULATION EXAMPLES

3.3.4.1 Example 1



The effective byte address is:

Immediate:	10AE if RB = 0 (note: 1B0 is the effective data)
Relative:	125E if RB =1 (1B0+10AE)
Absolute:	01B0 if RB =2
Indexed:	035C if RB =4 - if the contents of R4 = 01AC (1AC+1B0=35C)

3.3.4.2 Example 2

	15			12				8				4				0	
10AC	0	0	0	1		R	A		1	1	0	0		R	B		JMPM
10AE	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	

The effective jump address is:

Relative:	125E if RB = 1
Absolute:	01B0 if RB = 2
Indexed:	035C if RB = 4
Indirect:	1000 if RB = 0 and byte 125E and 125F contain 1000
	1B0 + 10AE = 125E

3.4 NOTATIONS

Below are notations that will be used in the operation code descriptions that follow:

For Operations:

$\vee$	-	Inclusive OR
$\oplus$	-	Exclusive OR
$\wedge$	-	AND
$\sim$	-	NOT
( )	-	Contents of
$\longrightarrow$	-	Is Stored In
EA	-	Effective Byte Address

For Condition Codes:

X	-	Set Conditionally
-	-	Not Affected
0	-	Cleared

For Fields:

EXP	-	Expression
RA	-	Register A
RB	-	Register B
PC	-	Program Counter
PSR	-	Program Status Register
PIA	-	Present Instruction Address
L. S.	-	Least Significant
M. S.	-	Most Significant

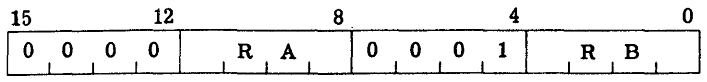
For Examples:

RA:	-	Means RA Contains

3.5 FIXED POINT ARITHMETIC INSTRUCTIONS

3.5.1 ADD REGISTER

ADD RA,RB



Operation: (RA) + (RB)$\longrightarrow$RA

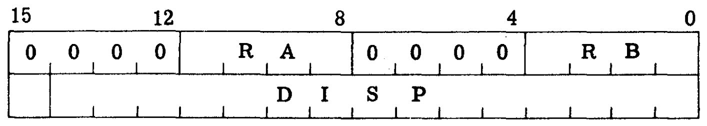
The contents of RA are added to the contents of RB and the result is placed in RA.

Condition Codes:

N: Set if result not zero; cleared if result zero
V: Set if overflow; cleared otherwise
L: Set equal to carry
M: Set to sign of result

3.5.2 ADD MEMORY

ADDM RA,EXP(RB)



Operation: (EA, EA + 1) + (RA)$\longrightarrow$RA

The contents of two bytes starting at the effective byte address are added to the contents of RA and the result is placed in RA.

Condition Codes:

N: Set if result not zero; cleared if result zero
V: Set if overflow; cleared otherwise
L: Set equal to carry
M: Set to sign of result

Example:

Initial Conditions:
PIA: 002E
R5: 01F4 R6: 000E
ADR at byte address 4E and 4F and
4E and 4F contain: 213D
LOC at byte address 40

a) ADDM R5, ADR (R2)

Assembles as:	0502
	004E
Result:	R5: 2331

b) ADDM R5, ADR(R1)

Assembles as:	0501
	001E
Result:	R5: 2331

c) ADDM R5, LOC(R6)

Assembles as:	0506
	0040
Result:	R5: 2331

d) ADDM R5, ADR(R0)

Assembles as:	0501
	004E
Result:	R5: 0242

3.5.3 DOUBLE PRECISION ADD

DADD RA,RB

15			12			8				4			0
0	1	1	1	R	A	+	1	0	0	1	0	R	B	+	1

Operation: (RA, RA +1) + (RB, RB +1) $\rightarrow$ RA, RA + 1

The 32 bit two's complement contents of RA and RA + 1 is added to the 32 bit two's complement contents of RB and RB+1. The result is placed in RA and RA + 1. RB and RB +1 are unaffected.

Condition Codes:

N:	Cleared only if result contents of RA and RA + 1 are both zero
V:	Set if arithmetic overflow
L:	Set equal to carry
M:	Set to most significant bit of result RA

Example:

R4:	054F	R5:	DE3A
R9:	00F2	R6:	5547

DADD R4, R9

Result: R4: 0642 R5: 3381

3.5.4 ADD LITERAL EXTENDED

ADDL RA,LIT

15			12		8	0
1	0	0	1	R A		L I T

Operation: (RA) + LIT $\rightarrow$ RA

The literal - sign extended - is added to the contents of RA and the result is placed in RA. -128≤LIT ≤127

Condition Codes:

N: Set if result is non-zero; cleared if zero
V: Set if overflow occurs; cleared otherwise
L: Set equal to carry
M: Set equal to most significant bit of result

Example:

R7: 0300
ADDL R7, -1
R7: 02FF

Programming Note: Useful in incrementing or decrementing a register.

3.5.5 SUBTRACT REGISTER

SUB RA,RB

15	12	8	4	0
0 0 0 0	R A	0 0 1 1	R B	

Operation: (RA) - (RB)$\rightarrow$RA

The contents of RB are subtracted from the contents of RA. The result is placed in RA.

Condition Codes:

N: Set if result is not zero; cleared if result zero
V: Set if arithmetic overflow; cleared otherwise
L: Set equal to the carry
M: Set to sign of the result

3.5.6 SUBTRACT MEMORY

SUBM RA,EXP(RB)

15	12	8	4	0
0 0 0 0	R A	0 0 1 0	R B	
	D I S P			

Operation: (RA) - (EA, EA +1) $\rightarrow$ RA

The contents of two bytes starting at the effective byte address are subtracted from the contents of RA, and the result is placed in RA.

Condition Codes:

N: Set if result is not zero; cleared if result zero
V: Set if arithmetic underflow; cleared otherwise
L: Set equal to the carry
M: Set to sign of the result

Examples:

Same conditions (but with SUBM opcode) as in the example for ADDM.
The results would be:

a) R5: E0B7
b) R5: E0B7
c) R5: E0B7
d) R5: 01A6

3.5.7 DOUBLE PRECISION SUBTRACT

	15 12	8	4	0
DSUB RA,RB	0 1 1 1	R A + 1	0 1 0 1	R B + 1

Operation: (RA, RA+1) - (RB, RB+1) → RA, RA+1

The 32 bit two's complement contents of RB and RB+1 are subtracted from 32 bit two's complement contents of RA and RA+1. The result is placed in RA and RA+1. RB and RB+1 are unaffected.

Condition Codes:

N:	Cleared only if both the result RA and RA +1 are zero
V:	Set if arithmetic overflow
L:	Set equal to carry
M:	Set equal to bit 15 of result RA

Example:

R3: 054F R4: DE3A
R8: 00F2 R9: 5547

DSUB R3, R8

Result: R3: 045D R4: 88F7

3.5.8 MULTIPLY

	15 12	8	4	0
MUL RA,RB	0 1 0 0	R A	0 1 1 0	R B

Operation: (RA) x (RB) → RA, RA+1

The contents of RB are multiplied by the contents of RA. The least significant 16 bits are placed in RA +1. The remaining most significant bits and sign are placed in RA, that is, RA and RA +1 contain a 32 bit two's complement product.

Examples:

R1: 0460   R5: 0500
MUL R1, R5
R1: 0017   R2: 6000

R3: FB40   R6: 0500
MUL R3, R6
R3: FFE8   R4: 4000

3.5.9 MULTIPLY AND ROUND

MULR   RA,RB

15	12	8	4	0
0 1 0 0	R A	0 1 1 0	R B	

Operation: (RA) x (RB) → RA

The 16 bit two's complement of registers RA and RB are multiplied together forming a 32 bit product. The result is left shifted one and bit 15 of the shifted result is used to round bits 16-31. Bits 16-31 are then placed in register RA. Overflow will occur if both register RA and RB contained 8000.

Condition Codes:

N: Set if result RA is not zero, cleared otherwise
V: Set if overflow (8000 x 8000)
L: Cleared
M: Set to sign of result RA

Example:

R3: 0600   R4: 0C00

MULR R3, R4

Result:

R3: 0090   R4: 0C00

3.5.10 INTEGER MULTIPLY

IMUL   RA,RB

15	12	8	4	0
0 1 0 0	R A	0 1 0 0	R B	

Operation: (RA) x (RB) → RA

The 16 bit two's complement contents of RA are multiplied by the 16 bit two's complement contents of RB. The result is a 32 bit product, but only the least significant 16 bits (0-15) are placed in RA. Bits 16-31 of the result are lost. RB and RA + 1 are unaffected.

Overflow will occur if the result is greater than 7FFF or less than 8000.

Condition Codes:

N:	Cleared only if RA x RB = 0
V:	Set if result >7FFF or< 8000
L:	Cleared
M:	Set to bit 15 of result RA

Example:

```
R3:  007A     R4:  0011

IMUL  R3, R4
```

Result:

```
R3:  081A     R4:  0011
```

3.5.11 DIVIDE

DIV RA,RB

15	12	8	4 0
0 1 0 0	R A	0 0 0 1	R B

Operation: (RA, RA + 1)/(RB) → RA, RA + 1

The contents of RA, RA + 1 are divided by the contents of RB. The dividend (RA, RA+1) is a 32 bit two's complement number. The quotient is placed in RA + 1 with appropriate sign. The absolute value of the remainder is placed in RA.

Condition Codes:

N:	Set if result not zero; cleared if result zero
V:	Set by an attempt to divide a number by a smaller number or zero
L:	Cleared
M:	Set to most significant bit of quotient

Examples:

```
R3:  0366    R4:  000D    R6:  13F8
   DIV   R5, R3
R5:  03D9    R6:  0082
```

3.5.12 MOVE REGISTER

MOV RA,RB

15	12	8	4 0
0 0 0 1	R A	0 0 0 0	R B

Operation: (RB) → RA

Contents of RB are moved to RA. RB is unaffected.

Condition Codes:

N: Set if RA not zero; cleared if RA = 0
V: Unaffected
L: Unaffected
M: Set to most significant bit of RA

3.5.13 MOVE REGISTER BYTE EXTENDED

MOVX RA,RB

15 12	8	4	0
0 1 0 0	R A	0 0 1 0	R B

Operation: L. S. 8 bits of RB → RA

The least significant byte of RB is moved to the least significant byte of RA. The bit 7 of the byte is propagated thru the most significant byte of RA. RB is unaffected.

Condition Codes:

N: Set if byte is not zero; cleared if it is zero
V: Unaffected
L: Unaffected
M: Set equal to bit 7 of transferred byte

3.5.14 MOVE LITERAL EXTENDED

MOVL RA,LIT

15 12	8	0
1 1 0 1	R A	L I T

Operation: LIT → RA

The eight bit literal (LIT) is moved into the least significant byte of RA. Bit 7 of the literal is extended into the most significant byte of RA.

Condition Codes:

N: Cleared only if the literal is zero
V: Unaffected
L: Unaffected
M: Set equal to most significant bit of the byte

3.5.15 NEGATE

NEG RA,RB

15 — 12	8	4	0
0 1 0 0	R A	0 1 0 0	R B

Operation: -(RB)$\rightarrow$RA

The two's complement of the contents of RB is placed in RA. The contents of RB are unaffected. Note: the two's complement of 8000 is 8000.

Condition Codes:

N:	Set if result is not zero; cleared if result is zero
V:	Set only if (RB) = 8000
L:	Set equal to carry
M:	Set to bit 15 of result RA

Example:

R3: 5A5A

NEG R3, R3

Result:

R3: B5B6

3.5.16 DOUBLE PRECISION NEGATE

DNEG RA,RB

15 — 12	8	4	0
0 1 1 1	R A + 1	0 1 0 0	R B + 1

Operation: -(RB, RB +1)$\rightarrow$RA, RA +1

The 32 bit contents of RB and RB +1 are two's complemented and the 32 bit result is placed in RA and RA + 1.

Condition Codes:

N:	Set if 32 bit result is not zero, cleared if result is zero
V:	Set if overflow (only if (RB), (RB)+1 = 8000 0000)
L:	Set equal to carry
M:	Set equal to bit 31 of result

3.5.17 ARITHMETIC COMPARE

CMP RA,RB

15 12	8	4	0
0 0 0 0	R A	1 0 1 1	R B

Operation: (RA) - (RB)

The two's complement contents of RA and RB are arithmetically compared. The contents of RA and RB are unaffected.

Condition Codes:

N:	Cleared if (RA) = (RB)
V:	Set if arithmetic overflow; cleared otherwise
L:	Set equal to carry
M:	Set equal to bit 15 of result

Example:

R3: 0F5A R4: 01B3 R5: 0F5A

CMP R3, R4
A branch will occur if one of the following branch instructions follows the compare:
BGT, BGE, BHI

CMP R4, R3
A branch will occur if one of the following branch instructions follows the compare:
BLT, BLE, BLOS

CMP R3, R5
A branch will occur when the compare preceeds the following branch instructions:
BGE, BLE, BLOS

3.5.18 ARITHMETIC COMPARE MEMORY

CMPM RA,EXP(RB)

15 12	8	4	0
0 0 0 0	R A	1 0 1 0	R B
	D I S P		

Operation: (RA) - (EA, EA + 1)

The two's complement quantities of RA and the effective word are arithmetically compared. Both operands are unaffected by the compare.

Condition Codes:

N: Cleared if (RA) = (EA, EA + 1)
V: Set if overflow
L: Set equal to carry
M: Set to bit 15 of compare result

3.5.19 DOUBLE PRECISION COMPARE

DCMP RA,RB

15	12	8	4	0
0 1 1 1	RA + 1	0 0 1 1	RB + 1	

Operation: (RA, RA+1) - (RB, RB+1)

The 32 bit two's complement contents of RA and RA+1 are compared to the 32 bit two's complement contents of RB and RB + 1. RA, RA + 1, RB and RB + 1 are not affected.

Condition Codes:

Same as DSUB

3.5.20 COMPARE BYTE LITERAL

CMPB RA,LIT

15	12	8	0
1 0 1 0	RA	LIT	

Operation: L. S. byte RA-LIT

Arithmetic compare between the least significant byte of RA and the literal. The most significant byte of RA is not involved in the compare.

Condition Codes:

N: Set if byte not equal to literal; cleared otherwise
V: Unaffected
L: Set equal to carry
M: Set to bit 7 of compare result

Programming Note: Useful in character compares.

3.5.21 SATURATED ADD

SADD RA,RB

15			12				8				4				0
0	1	0	0		R	A		0	1	1	0		R	B	

Operation: (RA) + (RB)$\rightarrow$RA
If overflow set; 7FFF$\rightarrow$RA or 8000$\rightarrow$RA
If overflow not set; RA=RA

The contents of Register A and Register B are added and the sum is placed in Register A. If overflow occurred on addition of positive numbers RA is set to 7FFF. If it occurred on addition of negative numbers RA is set to 8000. If no overflow occurred the sum is unaffected.

Condition Codes:

N:	Set if result not zero; cleared otherwise
V:	Cleared
L:	Set equal to zero
M:	Set to bit 15 of result RA

Examples:

R3: 74A5 R6: 0A27 R11: 3000

SADD R6, R3 SADD R3, R11

Results:

R6: 7ECC R3: 7FFF

3.5.22 SATURATED SUBTRACT

SSUB RA,RB

15			12				8				4				0
0	1	0	0		R	A		0	1	1	0		R	B	

Operation: (RA) - (RB)$\rightarrow$RA
If overflow set; 8000$\rightarrow$RA or 7FFF$\rightarrow$RA
If overflow not set; RA =RA

The contents of Register B are subtracted from the contents of Register A and the difference is placed in Register A. If overflow occurs when subtracting a positive quantity from a negative quantity RA is set to 8000. If it occurs when subtracting a negative quantity from a positive one RA is set to 7FFF. If no overflow occurs the difference is unaffected.

Condition Codes:

N: Set if result not zero; cleared otherwise
V: Cleared
L: Set equal to zero
M: Set to bit 15 of result RA

Examples:

R3: 8B5B   R6: 0A27   R11: 3000

SSUB R3, R6   SSUB R3, R11

Results:

R3: 8134   R3: 8000

3.5.23 MOVE POSITIVE

MOVP RA,RB

15			12		8			4		0
0	1	0	0	R A		0	0	1	0	R B

Operation: If

(RB) < 0; -(RB)⟶RA
(RB) ≥ 0; (RB)⟶RA

If the contents of Register B are positive they are moved unaffected to Register A. If they are negative the contents of Register B are two's complemented and placed in Register A. Register B is unaffected.

Condition Codes:

N: Set if result RA not zero; cleared otherwise
V: Set only if (RB) is 8000
L: Set to carry
M: Set to bit 15 of result

Example:

R1: FFA5   R2: 00A5

MOVP R1, R2   MOVP R2, R2

Results:

R2: 0058   R2: 00A5

3.5.24 MOVE NEGATIVE

MOVN RA,RB

15			12				8				4				0
0	1	1	1		R	A		0	1	1	0		R	B	

Operation: If
(RB) > 0; $-(RB) \rightarrow RA$
(RB) $\leq$ 0; $(RB) \rightarrow RA$

If the contents of Register B are negative they are moved unaffected to Register A. If they are positive the contents of Register B are twos complemented and placed in Register A. Register B is unaffected.

Condition Codes:
- N: Set if result RA not zero; cleared otherwise
- V: Unaffected
- L: Set equal to carry
- M: Set to bit 15 of result RA

Example:

Rl: FFA5	R2: 00A5
MOVN R3, Rl	MOVN R4, R2

Results: R3: FFA5 R4: FF5B

3.5.25 DOUBLE PRECISION MEMORY ADD

DADDM RA,EXP(RB)

15			12				8				4				0
0	1	1	1	R	A+1			0	0	0	1		R	B	
						D	I	S	P+2						

Operation: $(RA, RA+1) + (EA \text{ through } EA+3) \rightarrow RA, RA+1$

The double precision word (EA, EA+ 1, EA+ 2, EA+ 3) is added to the contents of Registers RA and RA+ 1. The result is placed in Registers RA and RA+ 1.

Condition Codes:
- N: Cleared only if result registers RA and RA+ 1 are both zero
- V: Set if arithmetic overflow
- L: Set equal to carry
- M: Set to bit 15 of result RA (MS register)

3.5.26 DOUBLE PRECISION MEMORY SUBTRACT

DSUBM RA,EXP(RB)

15	14	13	12	11–8	7	6	5	4	3–0
0	1	1	1	RA+1	1	0	1	0	RB
	DISP+2								

Operation: (RA, RA+ 1) - (EA through EA+ 1)$\rightarrow$RA, RA+ 1

The double precision word (EA, EA+ 1, EA+ 2, EA+ 3) is subtracted from the contents of Registers RA and RA+ 1. The result is placed in Registers RA and RA+ 1.

Condition Codes:

N:	Cleared only if result Registers RA and RA+ 1 are both zero
V:	Set if arithmetic overflow
L:	Set equal to carry
M:	Set to bit 15 of result RA (MS register)

3.6 LOGICAL INSTRUCTIONS

3.6.1 LOGICAL AND

AND RA,RB

15	14	13	12	11–8	7	6	5	4	3–0
0	0	0	1	RA	1	1	1	1	RB

Operation: (RB) $\wedge$ (RA) $\longrightarrow$RA

The 16 bit contents of RA and RB are logically ANDed together. The result is placed in RA. RB is unaffected. AND is useful in clearing a bit or bits.

Condition Codes:

N:	Set if RA not zero, cleared if zero
V:	Cleared
L:	Unaffected
M:	Set equal to most significant bit of result

Example:

```
R3:  47B5    R7:  3A57
AND  R3, R7
R3:  0215    R7:  3A57
```

3.6.2 MEMORY AND

ANDM RA,EXP(RB)

15	12	8	4	0
0 0 0 1	R A	1 1 1 0	R B	
	D I S P			

Operation: (EA, EA + 1) $\wedge$ (RA) $\longrightarrow$ RA

Two bytes of memory starting at the effective byte address are ANDed with the contents of RA. The result is placed in RA.

Condition Codes:

N: Set if RA not zero, cleared if zero
V: Cleared
L: Unaffected
M: Set equal to most significant bit of result

3.6.3 AND BYTE LITERAL

ANDB RA,LIT

15	12	8	0
1 1 0 0	R A	L I T	

Operation: L. S. Byte RA $\wedge$ LIT $\longrightarrow$ RA

The least significant byte of RA is ANDed with the literal and the result is placed in RA. The most significant byte of RA is unaffected.

Condition Codes:

N: Set if byte result not zero; cleared otherwise
V: Unaffected
L: Unaffected
M: Set to bit 7 of result

Programming Note: Useful to clear a bit or bits in the L. S. byte of a register.

3.6.4 INCLUSIVE OR

OR RA,RB

15	12	8	4	0
0 0 0 0	R A	0 1 1 1	R B	

Operation: (RB) $\vee$ (RA) $\longrightarrow$ RA

Contents of RB are inclusive OR'd with the contents of RA. The result is placed in RA. The contents of RB are unaffected. OR is useful in setting a bit or bits.

Condition Codes:

N:	Set if RA is not zero; cleared if RA is zero
V:	Unaffected
L:	Unaffected
M:	Set equal to most significant bit of RA

Example:

```
R3:  47B5    R7:  3A57
OR    R3, R7
R3:  7FF7    R7:  3A57
```

3.6.5 MEMORY INCLUSIVE OR

ORM RA,EXP(RB)

15 … 12	11 … 8	7 … 4	3 … 0
0 0 0 0	R A	0 1 1 0	R B
	D I S P		

Operation: (EA, EA + 1) v (RA) $\longrightarrow$ RA

The contents of the effective address is inclusive OR'd with the contents of RA. The result is placed in RA. The contents of the two bytes of the effective address are unaffected.

Condition Codes:

N:	Set if result RA is not zero; cleared otherwise
V:	Unaffected
L:	Unaffected
M:	Set to bit 15 of result RA

3.6.6 EXCLUSIVE OR

XOR RA,RB

15 … 12	11 … 8	7 … 4	3 … 0
0 0 0 0	R A	1 1 0 1	R B

Operation: (RB) $\oplus$ (RA) $\longrightarrow$ RA

The contents of RB are exclusive OR'd with the contents of RA. The result is placed in RA. The contents of RB are unaffected.

Condition Codes:

- N: Set if RA is not zero; cleared if RA is zero
- V: Unaffected
- L: Unaffected
- M: Set equal to most significant bit of RA

Example:

```
R3:  47B5    R5:  59CB
XOR  R3, R5
R3:  1E7E
```

Programming Note: The sequence RA exclusive OR'd RB, RB exclusive OR'd RA, RA exclusive OR'd RB results in the exchange of the contents of RA and RB without the use of temporary storage words.

Exclusive OR may be used to invert bits. Any field exclusive OR'd with itself becomes zero.

3.6.7 MEMORY EXCLUSIVE OR

XORM RA,EXP(RB)

15 12	8	4	0
0 0 0 0	R A	1 1 0 0	R B
	D I S P		

Operation: $(EA,\ EA\ \ 1) \oplus (RA) \longrightarrow RA$

The contents of the effective address is exclusive ORd with the contents of RA. The result is placed in RA. The contents of the EA are unaffected.

Condition Codes:

- N: Set if result RA not zero; cleared otherwise
- V: Unaffected
- L: Unaffected
- M: Set to bit 15 of result RA

3.6.8 ONE'S COMPLEMENT

COM RA,RB

15 12	8	4	0
0 0 0 1	R A	0 0 0 1	R B

Operation: $-(RB)-1 \longrightarrow RA$

One's complement RB - each bit of RB equal to 1 is cleared and each bit equal to zero is set to 1. The result is placed in RA. RB is unaffected.

Condition Codes:
- N: Cleared only if original RB was FFFF
- V: Unaffected
- L: Cleared
- M: Set to most significant bit of result RA

Example:

```
R3:  47B5
COM  R3, R3
R3:  B84A
```

3.6.9 ONE'S COMPLEMENT BYTE

COMB RA,RB

15 12	8	4	0
0 0 0 1	R A	0 0 1 1	R B

Operation: -(RB)-1 byte $\longrightarrow$ RA byte

The least significant byte of RB is one's complemented and placed in the least significant byte of RA. RB and the most significant byte of RA are unaffected.

Condition Codes:
- N: Cleared only if source RB was XXFF
- V: Unaffected
- L: Cleared
- M: Set to bit 7 of result RA

3.6.10 LOGICAL COMPARE

LC RA,RB

15 12	8	4	0
0 0 0 0	R A	1 0 0 1	R B

Operation: (RA) $\wedge$ (RB)

The contents of RA and RB are logically compared - they are ANDed together.

Condition Codes:
- N: Set if any bits match; cleared otherwise
- V: Unaffected
- L: Unaffected
- M: Set to most significant bit of the AND result

Programming Note: The logical compare is useful to compare bits of RA and RB. RA and RB are unaffected.

3.6.11 LOGICAL COMPARE MEMORY

LCM RA,EXP(RB)

15 12	8	4	0
0 0 0 0	R A	1 0 0 0	R B
	D I S P		

Operation: (RA) ∧ (EA, EA + 1)

Same as LC except RA is logically compared to the contents of the EA and EA + 1 instead of RB.

Condition Codes:

N:	Set if any bits match; cleared otherwise
V:	Unaffected
L:	Unaffected
M:	Set to bit 15 of the compare result

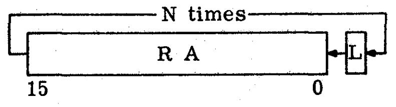
3.7 SHIFT INSTRUCTIONS

3.7.1 SHIFT LEFT LOGICAL AND INSERT LINK

SLL RA,N

15 12	8	4	0
0 1 1 0	R A	0 0 0 0	N-1

Operation:



Shift all 16 bits of Register A left N (≤16) places. On each shift, the link is shifted into bit 0 and bit 15 is shifted into the link. A shift of zero (as in all shift operations) is illegal.

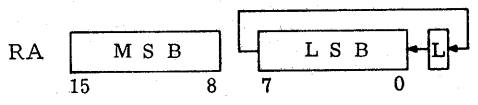
Condition Codes:
- N: Set if result not zero; cleared otherwise
- V: Unaffected
- L: Set to bit (16-N) of original data
- M: Set to bit 15 of result RA

3.7.2 SHIFT LEFT LOGICAL-LEAST SIGNIFICANT BYTE-AND INSERT LINK

SLLB RA,N

15 12	8	4	0
0 1 1 0	R A	0 0 0 1	N-1

Operation :

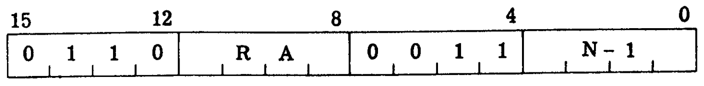


Shift the least significant 8 bits of Register A left N (≤ 8) places. On each shift the link is shifted into bit 0 and bit 7 of RA is shifted into the link.

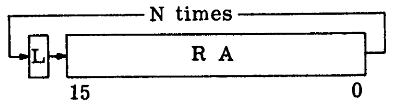
Condition Codes:
- N: Set if result LSB is not zero; cleared otherwise
- V: Unaffected
- L: Set to bit (8-N) of original data
- M: Set to bit 7 of result RA

3.7.3 SHIFT RIGHT LOGICAL AND INSERT LINK

SRL RA,N



Operation :



Shift all 16 bits of Register A right N (≤ 16) places. On each shift the link is shifted into bit 15 and bit 0 is shifted into the link.

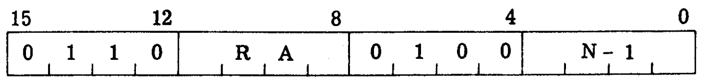
Condition Codes:

N: Set if results not zero; cleared otherwise
V: Unaffected
L: Set equal to bit (N-1) of original data
M: Set to bit 15 of result RA

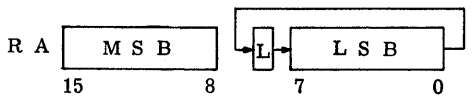
Programming Note: Useful in conjunction with SLL to reverse bits in a register.

3.7.4 SHIFT RIGHT LOGICAL-LEAST SIGNIFICANT BYTE-AND INSERT LINK

SRLB RA,N



Operation :



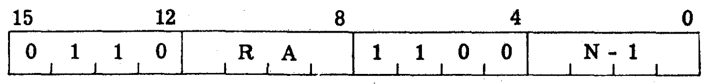
Shift the least significant 8 bits of Register A right N (≤ 8) places. On each shift the link is shifted into bit 7 and bit 0 is shifted into the link.

Condition Codes:

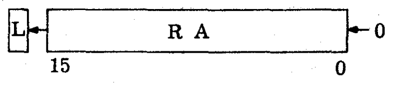
N: Set if result LSB is not equal to zero; cleared otherwise
V: Unaffected
L: Set equal to bit (N-1) of original data
M: Set to bit 7 of result RA

3.7.5 SHIFT LEFT LOGICAL

SLZ RA,N



Operation:



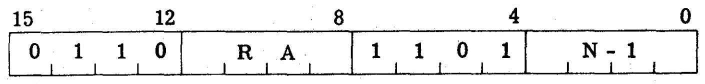
Shift all 16 bits of Register A left N (≤ 16) bits. On each shift a zero is shifted into bit 0 and bit 15 is shifted into link.

Condition Codes:

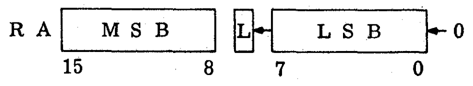
N:	Set if result is not zero; cleared otherwise
V:	Unaffected
L:	Set to bit (16-N) of original data
M:	Set to bit 15 of result RA

3.7.6 SHIFT LEFT LOGICAL-LEAST SIGNIFICANT BYTE

SLZB RA,N



Operation:



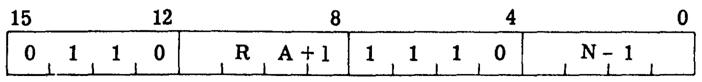
Shift the least significant 8 bits of Register A left N (≤ 8) times. On each shift a zero is shifted into bit 0 of RA and bit 7 is shifted into the link.

Condition Codes:

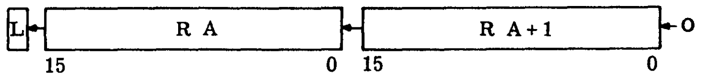
N:	Set if result LSB is not zero; cleared otherwise
V:	Unaffected
L:	Set to bit (8-N) of original data
M:	Set to bit 7 of result RA

3.7.7 SHIFT LEFT DOUBLE LOGICAL

DSL RA,N



Operation :



Shift the 32 bits of Register A and Register A 1 left N (16) places. On each shift a zero is shifted into bit 0 of RA 1 and bit 15 of RA is shifted into the link. If RA = 15 is specified in the symbolic language then the two registers shifted are R15 (RA) and RO (RA 1).

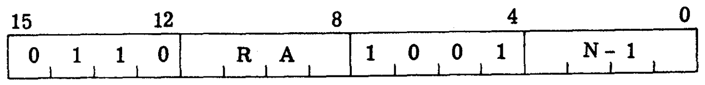
Note: Field 0-3 is alwavs N(Mod 16)

Condition Codes:

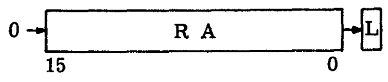
N: Set if result (RA, RA +1) is not zero; cleared otherwise
V: Unaffected
L: Set equal to bit (16-N) or original RA data
M: Set equal to bit 15 of result RA

3.7.8 SHIFT RIGHT LOGICAL

SRZ RA,N



Operation :



All 16 bits of Register A are right shifted N (≤16) places. On each shift bit 0 is shifted into the link and a zero is shifted into bit 15.

Condition Codes:

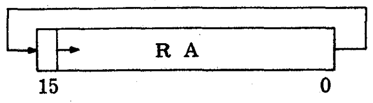
N: Set if result is not zero; cleared otherwise
V: Unaffected
L: Set to bit N-1 of result
M: Set to bit 15 of result RA

3.7.9 SHIFT RIGHT CIRCULAR

SRC RA,N

15 12	8	4	0
0 1 1 0	R A	1 1 1 1	N - 1

Operation:



Shift the 16 bits of Register A right circular N places. On each shift bit 15 is shifted into bit 14 and bit 0 is shifted into bit 15.

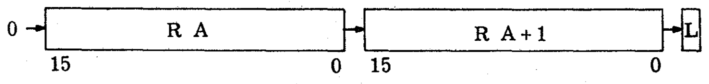
Condition Codes:
- N: Cleared only if original RA was zero; set otherwise
- V: Unaffected
- L: Unaffected
- M: Set to bit 15 of result RA

3.7.10 SHIFT RIGHT DOUBLE LOGICAL

DSR RA,N

15 12	8	4	0
0 1 1 0	R A	1 0 1 1	N - 1

Operation:

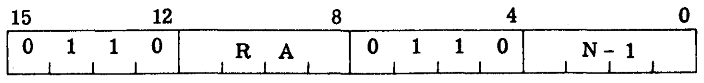


Shift the 32 bits of RA and RA +1 right N (≤ 16) places. On each shift bit 0 of RA + 1 is shifted into the link and a zero is shifted into bit 15 of RA. If RA =15 is specified the two registers operated on are R15 (RA) and RO (RA + 1).

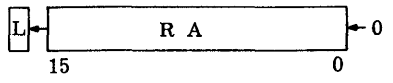
Condition Codes:
- N: Set if result (RA, RA +1) is zero; cleared otherwise
- V: Unaffected
- L: Set to bit (N-1) of original RA + 1 data
- M: Set to bit 15 of result RA

3.7.11 SHIFT LEFT ARITHMETIC

SLA RA,N



Operation:



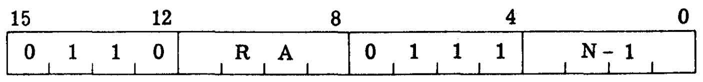
Shift all 16 bits of register A left N (≤ 16) places. On each shift bit 15 is shifted into the link and a zero is shifted into bit 0. Same as SLL except overflow is detected. (L ≠ M after shift).

Condition Codes:

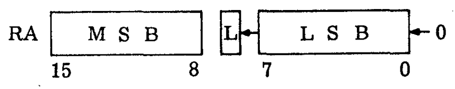
N:	Set if result RA not zero; cleared otherwise
V:	Set if arithmetic overflow (L ≠ M)
L:	Set to bit (16-N) of original RA data
M:	Set to bit 15 of result

3.7.12 SHIFT LEFT ARITHMETIC-LEAST SIGNIFICANT BYTE

SLAB RA,N



Operation:



Shift the least significant 8 bits of RA left N (≤ 8) places. On each shift bit 7 of RA is shifted into the link and a zero is shifted into bit 0. Same as SLB except that overflow is detected.

Condition Codes:

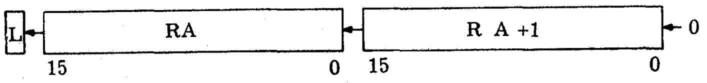
N:	Set if LSB of result RA is not zero; cleared otherwise
V:	Set if overflow (L ≠ M)
L:	Set to bit (8-N) of original RA data
M:	Set to bit 7 of result RA

3.7.13 SHIFT LEFT DOUBLE ARITHMETIC

DSLA RA,N

15	12	8	4	0
0 1 1 0	R A+1	1 0 0 0	N-1	

Operation :



Shift the 32 bits of RA and RA + 1 left N ($\leq$ 16) places. On each shift bit 15 of RA is shifted into the link and a zero is shifted into bit 0 of RA + 1. Same as DSL except that arithmetic overflow is detected.

Condition Codes:

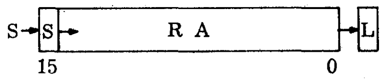
N:	Set if (RA) and (RA+1) are both not zero; cleared otherwise
V:	Set if over flow (L $\neq$ M)
L:	Set to bit (16-N) of original RA data
M:	Set to bit 15 of result RA

3.7.14 SHIFT RIGHT ARITHMETIC

SRA RA,N

15	12	8	4	0
0 1 0 1	R A	0 1 1 0	N-1	

Operation:



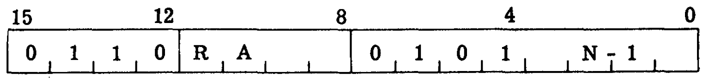
Register A is right shifted N ($\leq$16) places. On each shift bit zero of RA is shifted into the link and bit 15 is shifted into bit 14. Bit 15 same as before shift.

Condition Codes:

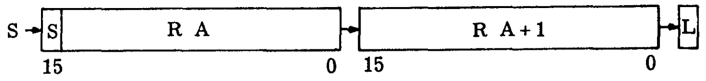
N:	Set if result RA is not zero; cleared otherwise
V:	Unaffected
L:	Set to bit (N-1) of original RA data
M:	Set to bit 15 of RA

3.7.15 SHIFT RIGHT DOUBLE ARITHMETIC

DSRA RA,N



Operation :



Shift RA and RA+ 1 right N (≤ 16) places. On each shift bit 0 of RA +1 is shifted into the link and bit 15 of RA is shifted into bit 14 of RA. Bit 15 of RA does not change.

Condition Codes:

N:	Set if both (RA) and (RA +1) not zero; cleared otherwise
V:	Unaffected
L:	Set to bit (N-1) of original RA +1 data
M:	Set to bit 15 of RA

SHIFT INSTRUCTION EXAMPLES:

Initial Condition: R5: 5A5A L = 1 (for each instruction)

Operation	Results	
SLL R5, 2	R5: 696A	L = 1
SLLB R5, 2	R5: 5A6A	L = 1
SRL R5, 2	R5: 5696	L = 1
SRLB R5, 2	R5: 5A56	L = 1
SLZ R5, 2	R5: 6968	L = 1
SLZB R5, 2	R5: 5A68	L = 1
SRZ R5, 2	R5: 1696	L = 1
SRC R5, 2	R5: 9696	L = 1

Initial Condition: R5: E4E3 R6: C2B1 L = 0

SLA R5, 2	R5: 938C	L = 1	
SLAB R5, 2	R5: E48C	L = 1	
SRA R5, 2	R5: F938	L = 1	
DSL R5, 2	R5: 938F	R6: 0AC4	L = 1
DSR R5, 2	R5: 3938	R6: F0AC	L = 0
DSLA R5, 2	R5: 9384	R6: 0AC4	L = 1
DSRA R5, 2	R5: F938	R6: F0AC	L = 0

3.8 LOAD-STORE INSTRUCTIONS

All Load-Store instructions may reference memory or any of the devices on either the serial or parallel bus.

3.8.1 LOAD REGISTER

L RA,RB

15			12		8				4		0
0	1	0	1	R A		0	0	0	0	R B	

Operation: ((RB), (RB) +1) → RA

Load Register A with two bytes starting with the byte addressed by Register B.

Condition Codes:
- N: Set only if input data is not zero
- V: Unaffected
- L: Unaffected
- M: Set to most significant bit of input data

3.8.2 LOAD REGISTER AND INCREMENT

LI RA,RB

15			12		8				4		0
0	1	0	1	R A		0	0	1	0	R B	

Operation: ((RB), (RB) +1)) — RA
(RB) + 2 → RB

RA is loaded with two bytes starting with the byte addressed by RB. RB is then incremented by two.

Condition Codes:
- N: Set only if input word is not zero
- V: Unaffected
- L: Unaffected
- M: Set to bit 15 of input word

Programming Note: Useful to sequentially access data in a buffer or stack.

3.8.3 LOAD REGISTER AND DECREMENT

LD RA,RB

15			12		8				4		0
0	1	0	1	R A		0	1	0	0	R B	

Operation: $((RB), (RB)+1)) \rightarrow RA$
$(RB) - 2 \rightarrow RB$

RA is loaded with two bytes starting with the byte addressed by RB. RB is then decremented by two.

Condition Codes:

N:	Set only if input word not zero
V:	Unaffected
L:	Unaffected
M:	Set to bit 15 of input word

3.8.4 LOAD REGISTER

LM RA,EXP(RB)

15 12	8	4	0
0 0 0 1	R A	0 0 1 0	R B
	D I	S P	

Operation: (EA, EA+1) $\longrightarrow$ RA

RA is loaded with 2 bytes starting at the effective byte address.

Condition Codes:
- N: Set only if input word not zero
- V: Unaffected
- L: Unaffected
- M: Set to bit 15 of input word

3.8.5 LOAD REGISTER BYTE

LMB RA,EXP(RB)

15 12	8	4	0
0 0 0 1	R A	0 1 0 0	R B
	D I	S P	

Operation: (EA) $\longrightarrow$ RA

The least significant byte of RA is loaded with the contents of the effective byte address. The most significant byte of RA is unaffected.

Condition Codes:
- N: Set only if input byte not zero
- V: Unaffected
- L: Unaffected
- M: Set to bit 7 of input byte

3.8.6 LOAD ADDRESS

LA RA,EXP

15 12	8	4	0
0 0 0 1	R A	0 1 0 1	R B
	D I	S P	

Operation: (PC) +DISP $\longrightarrow$ RA

The contents of the Program counter is added to the displacement and placed in Register A. This instruction is useful in programs that are meant to be relocatable.

Condition Codes:

N: Cleared only if DISP = 0
V: Unaffected
L: Unaffected
M: Unaffected

3.8.7 STORE REGISTER

S RA,RB

15 12	8	4	0
0 1 0 1	R A	0 1 1 0	R B

Operation: (RA)→(RB), (RB) + 1

The contents of RA are stored into memory starting at the byte address addressed by RB.

Condition Codes:

N: Set only if (RA) not zero
V: Unaffected
L: Unaffected
M: Set to bit 15 of RA

3.8.8 STORE REGISTER AND INCREMENT

SI RA,RB

15 12	8	4	0
0 1 0 1	R A	1 0 0 0	R B

Operation: (RA) →(RB), (RB) + 1
(RB) + 2 →RB

The 2 bytes in Register A are stored starting at the byte addressed by RB. RB is then incremented by two.

Condition Codes:

N: Set only if (RA) not zero
V: Unaffected
L: Unaffected
M: Set to bit 15 of RA

3.8.9 STORE AND DECREMENT

SD RA,RB

15 12	8	4	0
0 1 0 1	R A	1 0 1 0	R B

Operation: (RA) $\rightarrow$ (RB), (RB) + 1
(RB) - 2 $\rightarrow$ RB

The two bytes of Register A are stored starting at the byte addressed by Register B. The contents of RB are then decremented by 2.

Condition Codes:

N:	Set only if (RA) not zero
V:	Unaffected
L:	Unaffected
M:	Set to bit 15 of RA

3.8.10 STORE REGISTER

SM RA,EXP(RB)

15 12	8	4	0
0 0 0 1	R A	0 1 1 0	R B
	D I S P		

Operation: (RA) $\rightarrow$ EA, EA + 1

The contents of RA are stored into the effective byte address and effective byte address plus one. This instruction has no immediate addressing mode. RB = 0 is treated as indexing.

Condition Codes:

N:	Set only if RA not zero
V:	Unaffected
L:	Unaffected
M:	Set to bit 15 of RA

3.8.11 STORE REGISTER BYTE

SMB RA,EXP(RB)

15 12	8	4	0
0 0 0 1	R A	1 0 0 0	R B

Operation: L. S. Byte of RA $\rightarrow$ Effective byte address

Condition Codes:

N:	Set only if L. S. byte of RA not zero
V:	Unaffected
L:	Unaffected
M:	Set to bit 7 of RA

3.8.12 DOUBLE PRECISION LOAD

DLM RA,EXP(RB)

15	12	8	4	0
0 1 0 1		R A	0 0 0 1	R B
	D I S P			

Operation: (EA through EA + 3)→RA, RA + 1

The double precision contents of effective byte address are placed in Registers RA and RA + 1.

Condition Codes:
- N: Cleared only if (EA through EA + 3) are zero
- V: Unaffected
- L: Unaffected
- M: Set equal to bit 7 of (EA).

3.8.13 DOUBLE PRECISION STORE

DSM RA,EXP(RB)

15	12	8	4	0
0 1 0 1		R A	1 0 0 1	R B
	D I S P			

Operation: (RA, RA + 1)→EA through EA + 3

The contents of Registers RA and RA + 1 are stored in byte addresses EA through EA + 3.

Condition Codes:
- N: Cleared only if Registers RA and RA + 1 are both zero
- V: Unaffected
- L: Unaffected
- M: Set equal to bit 15 of Register RA

3.8.14 POP (LOAD MULTIPLE)

POP RA,N

15	12	8	4	0
0 1 0 1		R A	1 1 0 0	N

Operation: Words starting at (R15)-2 to (R15)-2N are loaded into register A to register A-N + 1.

The contents of the stack pointer (R15) is decremented by two. Register A is then loaded with 2 bytes starting with the byte addressed by R15. The count (N) is then decremented by one. If the count is not zero, R15 is again decremented by two and the next lower numbered register is loaded. This process is performed N times. R15 will point to last word loaded.

Condition Codes:
None affected

Example:

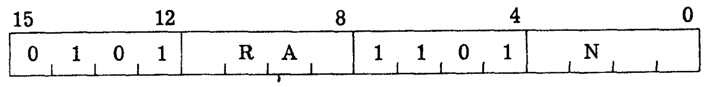
If R15 contains: 03FA then
POP R9, 4 will load

R9 with contents of bytes 3F8 and 3F9
R8 with contents of bytes 3F6 and 3F7
R7 with contents of bytes 3F4 and 3F5
R6 with contents of bytes 3F2 and 3F3 and
R15 will contain 03F2

Programming Note: Useful in restoring registers before exiting an interrupt handling routine.

3.8.15 PUSH (STORE MULTIPLE)

PUSH RA,N



Operation: (RA) to (RA) + N-1 are stored into bytes starting at (R15) to (R15)+2N-2

Register A is stored into two bytes starting with the one addressed by the stack pointer (R15). The contents of R15 is then incremented by two. The count (N) is then decremented by one. If the count is not zero the next higher numbered register is then stored into two bytes starting with the byte addressed by R15. The contents of R15 are again incremented by two. This process is performed N times. R15 will point to last word stored plus two.

Condition Codes:
None affected

Example:

If R15 contains: 03F2 then
PUSH R6, 4 will store

Contents of R6 into bytes 3F2 and 3F3
Contents of R7 into bytes 3F4 and 3F5
Contents of R8 into bytes 3F6 and 3F7
Contents of R9 into bytes 3F8 and 3F9 and
R15 will contain 03FA

Programming Note: Useful in saving registers in an interrupt handling routine.

3.8.16 LOAD MULTIPLE

LDM RA,N

15			12		8				4		0
0	1	1	1	RA		1	0	0	0	N	

Operation: Words starting at (R0)-2 to (R0)-2N are loaded into register A to register A-N + 1.

The contents of the stack pointer (R0) is decremented by two. Register A is then loaded with 2 bytes starting with the byte addressed by R0. The count (N) is then decremented by one. If the count is not zero, R0 is again decremented by two and the next lower numbered register is loaded. This process is performed N times. R0 will point to last word loaded.

Condition Codes:

None affected

Example:

If R0 contains: 03FA then
LDM R9, 4 will load

R9 with contents of bytes 3F8 and 3F9
R8 with contents of bytes 3F6 and 3F7
R7 with contents of bytes 3F4 and 3F5
R6 with contents of bytes 3F2 and 3F3 and
R0 will contain 03F2

3.8.17 STORE MULTIPLE

	15 12	8	4	0
STM RA,N	0 1 1 1	R A	1 0 0 1	N

Operation: (RA) to (RA) + N-1 are stored into bytes starting at (RO) to (R0)+2N-2

Register A is stored into two bytes starting with the one addressed by the stack pointer (R0). The contents of R0 is then incremented by two. The count (N) is then decremented by one. If the count is not zero the next higher numbered register is then stored into two bytes starting with the byte addressed by R0. The contents of R0 are again incremented by two. This process is performed N times. R0 will point to last word stored plus two.

Condition Codes:
None affected

Example:

If R0 contains: 03F2 then
STM R6, 4 will store

Contents of R6 into bytes 3F2 and 3F3
Contents of R7 into bytes 3F4 and 3F5
Contents of R8 into bytes 3F6 and 3F7
Contents of R9 into bytes 3F8 and 3F9 and
R0 will contain 03FA

3.9 PROGRAM CONTROL INSTRUCTIONS

3.9.1 SUBROUTINE CALLS AND UNCONDITIONAL JUMPS

3.9.1.1 SUBROUTINE CALL

CAL RA,RB

15 12	8	4	0
0 0 0 1	R A	1 0 1 1	R B

Operation: (PC) $\rightarrow$ RA, (RB) $\rightarrow$ PC

The contents of the program counter (return address) are saved in RA and then the program counter assumes the value (in bytes) contained in RB.

Condition Codes:
None affected

3.9.1.2 JUMP

JMP RB

15 12	8	4	0
0 0 0 1	0 0 0 0	1 1 0 1	R B

Operation: (RB) $\rightarrow$ PC

The contents of the PC are replaced by the contents of RB. The original contents of PC are lost.

Condition Codes:
None affected

3.9.1.3 SUBROUTINE CALL

CALM RA,EXP(RB)

15	12	8	4	0
0 0 0 1		R A	1 0 1 0	R B
	D I S P			

Operation: (PC) $\rightarrow$ RA, JA $\rightarrow$ PC

The contents of the PC (return address) are saved in RA. The calculated jump address (depends on RB field) now replaces the contents of PC.

Condition Codes:
None affected

3.9.1.4 JUMP

JMPM EXP(RB)

15			12				8				4				0
0	0	0	1	0	0	0	0	1	1	0	0	R B			
	D I S P														

Operation: JA $\longrightarrow$ PC

The contents of the PC are replaced by the calculated jump address.

Condition Codes:
None affected

Examples:
(1) JMPM ADR(R2): ADR at byte address 114
Assembles as: 4092
0114

Jump Address 114, i.e., 114 $\rightarrow$ PC

(2) JMPM ADR(R4)
Assembles as: 4094
0114

If R4: 0100 Jump Address is 214, i.e., 214 $\longrightarrow$ PC

(3) JMPM ADR(R1) instruction at byte address 100 and ADR at byte address 114
Assembles as: 4091
0012

Jump Address is 114, 114 $\longrightarrow$ PC

(4) JMPM ADR(R0) same condition as (3) and byte address ADR contains 02 and ADR + 1 contains 20
Assembles as: 4090
0012

Jump Address is (114, 115) = 0220, 220 $\longrightarrow$ PC

3.9.2 RELATIVE BRANCHES

In the description of relative branches the DISP (displacement) is equal to EXP-current instruction address-2. EXP can be an address or any valid expression.

3.9.2.1 BRANCH UNCONDITIONALLY

		15 … 12	8	… 0
BR	EXP	1 0 0 0	1 1 1 0	D I S P

Operation: PC + DISP $\longrightarrow$ PC

The program counter is replaced by the sum of the present PC (which is at present byte location plus two) and the value of the displacement sign extended. This permits relative branching in the range of +64 words or -63 words (+128 and -126 bytes).

Condition Codes:
None affected

The following branch instructions all examine the Program Status Register (PSR). Whether or not a certain bit (N, V, L, M) or combination of bits is set determines if the branch will occur. BL, BNL, BM, BNM, BNZ, BZ, BV and BNV all test just one bit in the PSR. These bits may be set or cleared by any instruction affecting the condition codes. The symbol Z will be used for ~ N, i.e., Z = 1 if result is zero, Z = 0 if result is not zero.

BGT, BLE, BGE and BLT are used to test the results of instructions in which the operands were signed (two's complement) values.

BHI and BLOS allow the programmer to test the result of an operation on unsigned operations--where the sixteen bits of each operand are considered as an unsigned quantity.

At execution time the PC is equal to the current instruction address plus two.

3.9.2.2 BRANCH IF LINK SET

		15 … 12	8	… 0
BL	EXP	1 0 0 0	0 1 1 0	D I S P

Operation: (1) if Link = 1; PC + DISP $\longrightarrow$ PC
(2) if Link = 0; step

If the link is set, the program counter is replaced by the sum of the present PC and the DISP sign extended. If the link is clear the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.3 BRANCH IF LINK CLEAR

	15 12	8	0
BNL EXP	1 0 0 0	0 0 0 1	D I S P

Operation: (1) if Link = 1; step
(2) if Link = 0; PC + DISP $\longrightarrow$ PC

If the Link is set, the next sequential instruction is executed. If the Link is clear, the Program Counter is replaced by the sum of the present PC and the DISP sign extended.

Condition Codes:
None affected

3.9.2.4 BRANCH IF MOST SET
BRANCH IF NEGATIVE

	15 12	8	0
BM EXP BN EXP	1 0 0 0	0 0 1 0	D I S P

Operation: (1) if M = 1; PC + DISP $\longrightarrow$ PC
(2) if M = 0; step

If the most significant result bit in the status word is set, the Program Counter is replaced by the sum of the present PC and the DISP sign extended. If it is clear, the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.5 BRANCH IF MOST CLEAR
BRANCH IF POSITIVE

	15 12	8	0
BNM EXP BP EXP	1 0 0 0	0 0 1 1	D I S P

Operation: (1) if M = 1; step
(2) if M = 0; PC + DISP $\longrightarrow$ PC

If the most significant result bit in the status word is clear, the Program Counter is replaced by the sum of the present PC and the DISP sign extended. If it is set the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.6 BRANCH IF NOT ZERO SET
BRANCH NOT EQUAL

BNZ EXP
BNE EXP

15 12	8	0
1 0 0 0	0 1 0 0	D I S P

Operation: (1) if N = 1; PC + DISP $\longrightarrow$ PC
(2) if N = 0; step

If the Not Zero indicator is set, the Program Counter is replaced by the sum of the present PC and the DISP sign extended. If Not Zero is clear, the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.7 BRANCH IF NOT ZERO CLEAR
BRANCH EQUAL

BZ EXP
BE EXP

15 12	8	0
1 0 0 0	0 1 0 1	D I S P

Operation: (1) if N = 1; step
(2) if N = 0; PC + DISP $\longrightarrow$ PC

If the Not Zero indicator is clear, the Program Counter is replaced by the sum of the present PC and the DISP sign extended. If it is set, the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.8 BRANCH IF OVERFLOW SET

BV EXP

15 12	8	0
1 0 0 0	0 1 1 0	D I S P

Operation: (1) if V = 1; PC + DISP $\longrightarrow$ PC
(2) if V = 0; step

If the overflow bit in the status word is set, the Program Counter is replaced by the sum of the present PC and the DISP sign extended. If it is clear, the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.9 BRANCH IF OVERFLOW CLEAR

BNV EXP

15			12				8	0
1	0	0	0	0	1	1	1	D I S P

Operation: (1) if V = 1; step
(2) if V = 0; PC + DISP $\longrightarrow$ PC

If the overflow indicator is not set, the Program Counter is replaced by the sum of the present PC and the DISP sign extended. If it is set, the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.10 BRANCH GREATER THAN

BGT EXP

15			12				8	0
1	0	0	0	1	0	0	0	D I S P

Operation: (1) if Z V (M⊕V) = 0; PC +DISP $\longrightarrow$ PC
(2) if Z V (M⊕V) = 1; step

The Program Counter is replaced by the sum of the present PC and the DISP if a previous result was greater than zero. Otherwise, the next instruction is executed.

Condition Codes:
None affected

3.9.2.11 BRANCH LESS THAN OR EQUAL

BLE EXP

15			12				8	0
1	0	0	0	1	0	0	1	D I S P

Operation: (1) if Z V (M⊕V) = 1; PC + DISP $\longrightarrow$ PC
(2) if Z V (M⊕V) = 0; step

The Program Counter is replaced by the sum of the present PC and the DISP sign extended, if a previous operation result was less than or equal to zero. Otherwise, the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.12 BRANCH GREATER THAN OR EQUAL

BGE EXP

15			12				8								0
1	0	0	0	1	0	1	0			D	I	S	P		

Operation: (1) if (M⊕V) = 0; PC + DISP $\longrightarrow$ PC
(2) if (M⊕V) = 1; step

The Program Counter is replaced by the sum of the present PC and the DISP sign extended, if a previous result was greater than or equal to zero. Otherwise, the next sequential instructions are executed.

Condition Codes:
None affected

3.9.2.13 BRANCH LESS THAN

BLT EXP

15			12				8								0
1	0	0	0	1	0	1	1			D	I	S	P		

Operation: (1) if M⊕V = 1; PC + DISP $\longrightarrow$ PC
(2) if M⊕V = 0; step

The Program Counter is replaced by the sum of the present PC and the DISP sign extended, if a previous result was less than zero. Otherwise, the next sequential instruction is executed.

Condition Codes:
None affected

3.9.2.14 BRANCH HIGH

BHI EXP

```
15          12          8                        0
| 1  0  0  0 | 1  1  0  0 |          D  I  S  P          |
```

Operation: (1) if Z V L = 0; PC + DISP $\rightarrow$ PC
(2) if Z V L = 1; step

The branch will occur if a previous operation, such as CMP, caused neither a carry or zero result. If the instructions were CMP R5, R6 a branch will occur if R5 had a higher unsigned value than R6.

Condition Codes:
None affected

3.9.2.15 BRANCH LOW OR SAME

BLOS EXP

```
15          12          8                        0
| 1  0  0  0 | 1  1  0  1 |          D  I  S  P          |
```

Operation: (1) if Z V L = 1; PC + DISP $\rightarrow$ PC
(2) if Z V L = 0; step

The branch will occur if a previous operation caused either a carry or zero result. If the operation were CMP R5, R6 a branch will occur if R5 $\leq$ R6 - both being considered as unsigned quantities.

Condition Codes:
None affected

3.9.2.16 BRANCH ON COUNT

BCT RA,EXP

```
15          12          8                        0
| 0  1  1  1 |    R  A    |          D  I  S  P          |
```

Operation: (RA) - 1 $\rightarrow$ RA
(1) if RA $\neq$ 0; (PC) + DISP $\rightarrow$ PC
(2) if RA = 0; step

The contents of Register A are decremented by one. If the result is not zero the PC is replaced by EXP (PC + DISP). If the result is zero the next sequential instruction is executed.

Condition Codes:

N:	Set if RA not equal to zero; cleared otherwise
V:	Set if arithmetic overflow
L:	Set equal to carry
M:	Set to bit 15 of RA

3.10 SYSTEM CONTROL INSTRUCTIONS

3.10.1 PROGRAMMED INTERRUPT

	15			12				8				4				0
TRP LIT	0	1	1	1	LIT_{MS}				1	1	0	0	LIT_{LS}			

Operation: (R0)→(R15) (R15) + 2 → R15
(PSR)→(R15) (R15) + 2 → R15
(PC)→(R15) (R15) + 2 → R15
2*LIT→L.S. Byte Of R0
30_{16} = Address of Transfer Vector

TRP is a programmed interrupt or supervisor call. The contents of R0, PSR and PC, in that order, are saved in a stack table pointed to by R15. R0 is then loaded with the LIT. A new PC and PSR, in that order, are obtained from the transfer vector at byte address 30_{16}. R15 at completion will be six greater than before TRP was executed.

Condition Codes:
As in new PSR

Programming Note: The actual trap routine number LIT (≤ 255) is divided into 2 four bit fields in the instruction.

The TRP handling routine examines R0 (containing the 8 bit literal - LIT) and determines if the pointer (LIT) to the branch table is valid. If it is, the branch to the specific TRP routine is taken. If not such TRP address exists the TRP handling routine will generate an error message.

3.10.2 RETURN FROM INTERRUPT

	15			12				8				4				0
RTI	0	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0

Operation: (R15)-2→R15 ((R15))→PC
(R15)-2→R15 ((R15))→PSR

RTI is a return from a machine interrupt. The original contents of PC and PSR, in that order, are restored from the stack table pointed to by R15. At completion R15 points to stack word containing original PC. To be used in priviledged mode only.

Condition Codes:
As contained in the PSR in the stack

3.10.3 OR INTO PROGRAM STATUS REGISTER

OSR LIT

```
 15        12         8         4         0
| 0 1 1 1 | 0 0 0 0 | 0 0 0 0 |   L I T |
```

Operation: (PSR) $\vee$ LIT $\rightarrow$ PSR

The four bit literal is OR'd with the status bits (L, M, N, V) of the PSR and the result placed into the status bits. The remaining bits of the PSR are unaffected.

Condition Codes:
As set by PSR

3.10.4 AND INTO PROGRAM STATUS REGISTER

ASR LIT

```
 15        12         8         4         0
| 0 1 0 1 | 0 0 0 0 | 1 1 1 1 |   L I T |
```

Operation: (PSR) $\vee$ LIT $\longrightarrow$ PSR

The four bit literal is ANDed with the status bits (L, M, N, V) of the PSR and the result is placed into the status bits. The remaining bits of the PSR are unaffected.

Condition Codes:
As set by NSR

3.10.5 LOAD PROGRAM STATUS REGISTER

LSR RA

```
 15        12         8         4         0
| 0 1 0 1 |   R A   | 1 1 1 0 | 0 0 0 0 |
```

Operation: (RA) $\longrightarrow$ PSR

The contents of RA replace the contents of the PSR. To be used in the priviledge mode only.

Condition Codes:
As set by LSR

3.10.6 STORE PROGRAM STATUS REGISTER

		15 12	8	4	0
SSR	RA	0 1 1 1	R A	0 1 1 1	0 0 0 0

Operation: (PSR) $\longrightarrow$ RA

The contents of the Program Status Register are stored in register RA.

Condition Codes:
Unaffected

3.10.7 HALT

	15 12	8	4	0
HALT	0 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0

The HALT instruction causes the processor to stop at the current address of the HALT (PC will be at current instruction address plus 2). To continue the RUN switch on the processor console must be placed in the HALT position, then it can be either placed in the RUN mode to continue executing the program or in the SIE position to single step through the program. To be used in priviledge mode only.

Condition Codes:
None affected

3.10.8 WAIT

	15 12	8	4	0
WAIT	0 1 1 1	0 0 0 0	1 1 1 0	0 0 0 0

The WAIT instruction causes the processor to wait at the current instruction address for an interrupt to occur. While in WAIT the processor does not access memory thus freeing the bus.

Condition Codes:
None affected

3.10.9 RESET

	15			12				8				4				0
RSET	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0

RESET issues a reset simultaneously to both the serial and parallel I/O busses. The effect of the RESET is the same as power up.

Condition Codes:
None affected

TABLE 3-1

PROCESSOR OPERATIONS

Mnemonic		Type	Opcode	Description	Condition Codes LMNV	MIP116/216 Microcycles	Reference Page
ADD	RA, RB	RR	0a1b	Add Registers	XXXX	1	3-7
ADDL	RA, LIT	RI	CaLL	Add literal extended to register	XXXX	1	3-8
ADDM	RA, EXP(RB)	MR	0a0b	Add memory to register	XXXX	3[1]	3-7
AND	RA, RB	RR	1aFb	AND registers	-XX-	1	3-20
ANDB	RA, LIT	RI	CaLL	AND literal with register	-XX-	1	3-21
ANDM	RA, EXP(RB)	MR	1aEb	AND memory with register	-XX-	3[1]	3-21
ASR	LIT	RR	50FL	AND literal into PSR	XXXX	1	3-53
BCT	RA, EXP	RL	7add	Branch on count	XXXX	3	3-50
BE	EXP	RL	85dd	Branch on equal (≡ BZ)	----	3[2]	3-47
BGE	EXP	RL	8Add	Branch if greater than or equal	----	3[2]	3-49
BGT	EXP	RL	88dd	Branch if greater than	----	3[2]	3-48
BHI	EXP	RL	8Cdd	Branch if high	----	3[2]	3-50
BL	EXP	RL	80dd	Branch if link set	----	3[2]	3-45
BLE	EXP	RL	89dd	Branch if less than or equal	----	3[2]	3-48
BLOS	EXP	RL	8Ddd	Branch if low or same	----	3[2]	3-50
BLT	EXP	RL	8Bdd	Branch if less than	----	3[2]	3-49
BM	EXP	RL	82dd	Branch if most set	----	3[2]	3-46
BN	EXP	RL	82dd	Branch if negative (≡ BM)	----	3[2]	3-46
BNE	EXP	RL	84dd	Branch if not equal (≡ BNZ)	----	3[2]	3-47
BNL	EXP	RL	81dd	Branch if link not set	----	3[2]	3-46
BNM	EXP	RL	83dd	Branch if most not set	----	3[2]	3-46
BNV	EXP	RL	87dd	Branch if overflow not set	----	3[2]	3-48
BNZ	EXP	RL	84dd	Branch if not zero set	----	3[2]	3-47
BP	EXP	RL	83dd	Branch if positive (≡ BNM)	----	3[2]	3-46
BR	EXP	RL	8Edd	Unconditional branch	----	3	3-45
BV	EXP	RL	86dd	Branch if overflow set	----	3[2]	3-47
BZ	EXP	RL	85dd	Branch if not zero clear	----	3[2]	3-47

TABLE 3-1 (Cont)

Mnemonic		Type	Opcode	Description	Condition Codes LMNV	MIP116/216 Microcycles	Reference Page
CAL	RA, RB	RR	1aBb	Call Subroutine	----	2	3-43
CALM	RA, EXP(RB)	MR	1aAb	Call Subroutine	----	3	3-43
CMP	RA, RB	RR	0aBb	Arithmetic register compare	XXXX	1	3-15
CMPB	RA, LIT	RL	AaLL	Arithmetic literal compare	XXXX	1	3-16
CMPM	RA, EXP(RB)	MR	0aAb	Arithmetic memory compare	XXXX	3[1]	3-15
COM	RA, RB	RR	1a1b	One's complement register	-XX-	1	3-23
COMB	RA, RB	RR	1a3b	One's complement register byte	-XX-	1	3-24
DADD	RA, RB	RR	7a2b	Double precision add	XXXX	2	3-8
DADDM	RA, EXP(RB)	MR	7a1b	Double precision memory add	XXXX	6/4	3-19
DCMP	RA, RB	RR	7a3b	Double precision arithmetic compare	XXXX	3	3-16
DIV	RA, RB	RR	4a1b	Divide registers	XXXX	34	3-12
DLM	RA, EXP(RB)	MR	5a1b	Double precision register load	-XX-	4	3-39
DNEG	RA, RB	RR	7a4b	Double precision two's complement	XXXX	3	3-14
DSL	RA, N	RN	6aEn	Double register logical left shift	X-XX	4N/N+3	3-29
DSLA	RA, N	RN	6a8n	Double register arithmetic left shift	XXXX	4N/N+3	3-32
DSM	RA, EXP(RB)	MR	5a9b	Double precision register store	-XX-	4	3-39
DSR	RA, N	RN	6aBn	Double register logical right shift	XXX-	4N/N+3	3-30
DSRA	RA, N	RN	6a5n	Double register arithmetic right shift	XXX-	N + 3	3-32
DSUB	RA, RB	RR	7a5b	Double precision subtract	XXXX	2	3-10
DSUBM	RA, EXP(RB)	MR	7aAb	Double precision memory subtract	XXXX	6/4	3-20
HALT		RR	70F0	Halt	----	1	3-54
IMUL	RA, RB	RR	4a4b	Integer multiply	0XXX	18	3-11
JMP	RB	RR	10Db	Jump	----	2	3-43
JMPM	EXP(RB)	MR	10Cb	Jump	----	3	3-44

TABLE 3-1 (Cont)

Mnemonic		Type	Opcode	Description	Condition Codes LMNV	MIP116/216 Microcycles	Reference Page
L	RA, RB	RR	5a0b	Load register	-XX-	3	3-34
LA	RA, EXP	MR	1a51	Load register with address	-XX-	2	3-36
LC	RA, RB	RR	0a9b	Logically compare registers	XXXX	1	3-24
LCM	RA, EXP(RB)	MR	0a8b	Logically compare memory with register	XXXX	3[1]	3-25
LD	RA, RB	RR	5a4b	Load register and decrement	XXXX	3	3-34
LDM	RA, N	RN	7a8n	Load multiple registers	----	N +3	3-41
LI	RA, RB	RR	5a2b	Load register and increment	XXXX	3	3-34
LM	RA, EXP(RB)	MR	1a2b	Load register	-XX-	3[1]	3-36
LMB	RA, EXP(RB)	MR	1a4b	Load register byte	-XX-	3[1]	3-36
LSR	RA	RR	5aE0	Load PSR	XXXX	1	3-53
MOV	RA, RB	RR	1a0b	Move register	-XX-	1	3-12
MOVL	RA, LIT	RI	DaLL	Move literal extended to register	-XX-	1	3-13
MOVN	RA, RB	RR	7a6b	Move register negative	XXXX	1	3-19
MOVP	RA, RB	RR	7a7b	Move register positive	XXXX	1	3-18
MOVX	RA, RB	RR	4a2b	Move register byte extended	-XX-	1	3-13
MUL	RA, RB	RR	4a0b	Multiply registers	0XX-	18	3-10
MULR	RA, RB	RR	4a6b	Multiply registers and round	0XXX	20	3-11
NEG	RA, RB	RR	4a4b	Two's complement register	XXXX	1	3-14
NOP		RL	8E00	No operation (BR *+2)	----	2	
OR	RA, RB	RR	0a7b	Inclusive OR registers	-XX-	1	3-21
ORM	RA, EXP(RB)	MR	0a6b	Inclusive OR with memory	-XX-	3[1]	3-22
OSR	LIT	RR	700L	Inclusive OR into PSR	XXXX	2	3-53
POP	RA, N	RN	5aCm	Pop through R15	----	N+ 3	3-39
PUSH	RA, N	RN	5aDm	Push through R15	----	N+ 3	3-40

TABLE 3-1 (Cont)

Mnemonic		Type	Opcode	Description	Condition Codes LMNV	MIP116/216 Microcycles	Reference Page
RSET		RR	2010	Reset	----	1	3-55
RTI		RR	7FDF	Return from interrupt	XXXX	3	3-52
S	RA, RB	RR	5a6b	Store register	-XX-	2	3-37
SADD	RA, RB	RR	4a6b	Saturated add	XXX-	4	3-17
SD	RA, RB	RR	5aAb	Store register and decrement	XXXX	3	3-37
SI	RA, RB	RR	5a8b	Store register and increment	XXXX	3	3-37
SLA	RA, N	RN	6a6n	Shift left arithmetic	XXXX	N +1	3-31
SLAB	RA, N	RN	6a7n	Shift byte left arithmetic	XXXX	N + 1	3-31
SLL	RA, N	RN	6a0n	Shift left logical and insert link	XXX-	N + 1	3-26
SLLB	RA, N	RN	6a1n	Shift byte left logical and insert link	XXX-	N + 1	3-26
SLZ	RA, N	RN	6aCn	Shift left logical	XXX-	N + 1	3-28
SLZB	RA, N	RN	6aDn	Shift byte left logical	XXX-	N + 1	3-28
SM	RA, EXP(RB)	MR	1a6b	Store register	-XX-	3	3-38
SMB	RA, EXP(RB)	MR	1a8b	Store register byte	-XX-	3	3-38
SRA	RA, N	RN	6a4n	Shift right arithmetic	XXXX-	N + 1	3-32
SRC	RA, N	RN	6aFn	Shift right circular	XXX-	N + 1	3-30
SRL	RA, N	RN	6a3n	Shift right logical and insert link	XXXX-	N + 1	3-27
SRLB	RA, N	RN	6a4n	Shift byte right logical and insert link	XXX-	N + 1	3-27
SRZ	RA, N	RN	6a9n	Shift right logical	XXX-	N + 1	3-29
SSR	RA	RR	7a70	Store PSR	----	1	3-54
SSUB	RA, RB	RR	4a6b	Saturated subtract	XXX-	4	3-17
STM	RA, N	RN	7a9b	Store multiple registers	----	N + 3	3-42
SUB	RA, RB	RR	023b	Subtract register	XXXX	1	3-9
SUBM	RA, EXP(RB)	MR	0a2b	Subtract memory	XXXX	3[1]	3-9
TRP	LIT	RN	7FCF	Programmed interrupt	XXXX	8	3-52
WAIT		RR	70E0	Wait	----	1	3-54

TABLE 3-1 (Cont)

Mnemonic		Type	Opcode	Description	Condition Codes LMNV	MIP116/216 Microcycles	Reference Page
XOR	RA, RB	RR	0aDb	Exclusive OR registers	-XX-	1	3-22
XORM	RA, EXP(RB)	MR	0aCb	Exclusive OR memory with register	-XX-	3[1]	3-23

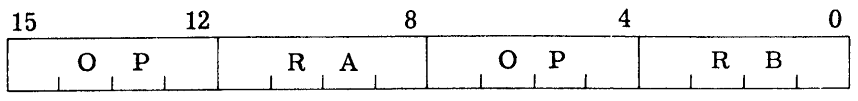
a		Register RA
b	-	Register RB
L	-	4 bit literal
LL	-	8 bit literal
n	-	Number of shifts minus one (N-1)
m	-	Number of registers to be loaded/stored
dd	-	Displacement
X	-	Condition code affected
-	-	Condition code unaffected
0	-	Condition code set to zero

1 - 2 cycles in the immediate mode

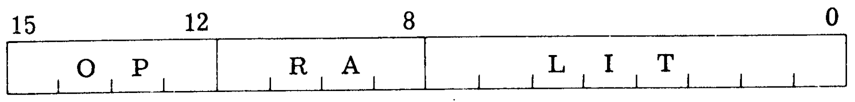
2 - 2 cycles if path is step

TABLE 3-2

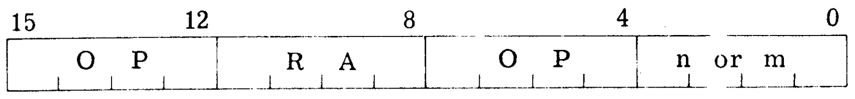
INSTRUCTION FORMATS



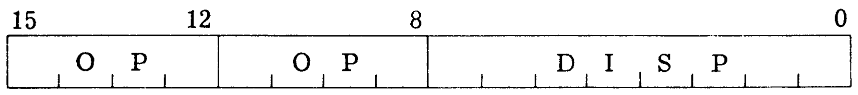
RR



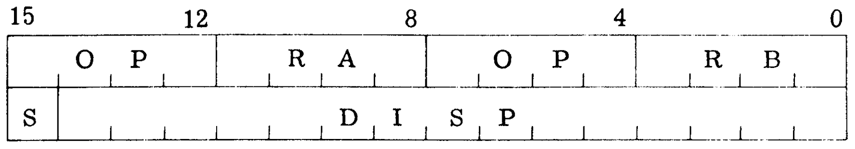
RI



RN



RL



MR

TABLE 3-3 POWERS OF 2

2	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 860 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

TABLE 3-4

DECIMAL AND HEXADECIMAL CONVERSIONS
POWERS OF 16

n	$n.16^7$	n	$n.16^6$	n	$n.16^5$	n	$n.16^4$	n	$n.16^3$	n	$n.16^2$	n	$n.16^1$	n	$n.16^0$
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	268,435,456	1	16,777,216	1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	536,870,912	2	33,554,432	2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	805,306,368	3	50,331,648	3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	1,073,741,824	4	67,108,864	4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	1,342,177,280	5	83,886,080	5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	1,610,612,736	6	100,663,296	6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	1,879,048,192	7	117,440,512	7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	2,147,483,648	8	134,217,728	8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	2,415,919,104	9	150,994,944	9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	2,684,354,560	A	167,772,160	A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	2,952,790,016	B	184,549,376	B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	3,221,225,472	C	201,326,592	C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	3,489,660,928	D	218,103,808	D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	3,758,096,384	E	234,881,024	E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	4,026,531,840	F	251,658,240	F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15

POWERS OF 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

To translate a decimal number to a hex number, find the next lower decimal number and its hex equivalent. Subtract this number from the decimal number. The next hex number is obtained from the difference. Continue until the entire hex number is developed.

To translate a hex number to its decimal equivalent, find the hex number and its decimal equivalent for each hex position. Sum these to obtain the decimal number.

SECTION 4
PROGRAMMING OF PERIPHERALS

4.0 INTRODUCTION

The MIP is designed to use a wide variety of peripherals ranging from teletypes to real time signal processing interfaces. A broad range of transfer rates, interrupt requirements and data transmission requirements may be accommodated. Programming is simplified by using a unified addressing structure and common interrupt programming techniques.

This flexibility is obtained by providing both a parallel and a serial data bus. The serial bus utilizes differential transmission and is suited for character type devices located up to fifty feet away in a high noise environment. It has a character rate of 500 KHz. The parallel bus is intended for high speed word oriented devices located within the processor enclosure. Transfer rates may be as high as 5 MHz on this bus for data and 5-14 MHz for instruction fetches.

Programming is simplified by using the same instructions for both busses as well as memory reference. Different addresses are used for devices on the serial or parallel busses. The upper 1024 byte addresses of page zero form the device address space of which the upper 256 byte addresses form the serial device address space.

Interrupt functions the same way for both busses. A device generates an interrupt request (one level for serial I/O; three levels for parallel I/O). The processor saves machine status and polls the highest level. It determines which of the requests has the highest priority and fetches a transfer vector to the appropriate handler. The transfer of control can take as little as 1 μs for parallel requests or 5 μs for serial requests.

Standard peripherals are available to provide input/output capability. These may be augmented by special application oriented peripherals. The standard peripherals include:

a.	ASR-33 Teletype	(serial bus)
b.	300 cps Paper Tape Reader	(serial bus)
c.	75 cps Paper Tape Punch	(serial bus)
d.	Removable Cartridge DISK File	(parallel bus)
e.	Real Time Clock	(serial bus)

Programming examples illustrating the use of these peripherals and interfaces are provided in section five, Peripheral Descriptions. Also provided in that section is a sample interrupt handling routine.

4.1 GENERAL PERIPHERAL INTERFACE OPERATION

A peripheral will be attached to one of the I/O busses by an interface. This interface will contain a status register and buffer registers for input or output. The processor addresses these registers and transfers data into or from them to control a peripheral. Data transfers may be either byte or full word depending on the register.

4.1.1 STATUS REGISTERS

This register usually contains bits indicating the status of the peripheral. A bit is usually provided to enable an interrupt request on the basis of certain status bits going from low to high as well as a DONE bit. The DONE bit is usually used to indicate completion of a peripheral operation initiated by loading an output buffer or by setting a function bit in the status register. Errors in peripheral operation are usually indicated by ERROR bits in the status register. If an interface controls multiple devices the status register may contain unit selection bits.

4.1.2 PERIPHERAL INTERRUPTS

The interface usually contains an interrupt request bit which is set by the low to high transition of various status bits; e.g., the DONE bit. Once set, it generates a request at a predetermined level and permits the interface to respond to polls of that level. The interface's poll position and request level and hence its transfer vector address are usually set by backpanel wiring. The request bit is reset when the status register is read by the processor.

SECTION 5
PERIPHERAL DESCRIPTIONS

5.0 INTRODUCTION

Standard peripherals are provided with the MIP to provide standardized input-output capabilities. These units are interfaced to the MIP using the techniques outlined in Section 4, Programming of Peripherals.

In the paragraphs which follow these devices and their interfaces are described. Programming examples are provided for each device to illustrate their operation. Device addresses and transfer vectors are given in Section 5.5.

5.1 ASR-33 TELETYPE

This device contains two separate subunits - a printer/punch and a keyboard/reader- capable of handling 10 characters per second to and from the computer simultaneously. The teletype unit is attached to the serial I/O bus through a teletype interface.

5.1.1 TELETYPE DEVICE CHARACTERISTICS

5.1.1.1 Codes

The 8 bit code generated by the keyboard and printed by the printer corresponds to the seven bit American Standard Code for Information Interchange (ASCII) modified. To convert ASCII to teletype code set bit 8 to a one. The teletype code is shown in table 5-1 below. The eighth bit corresponds to a parity bit. Most teletypes keyboards set this to a one for all data; however, some teletypes set this to an odd parity bit (i.e., a one if odd number of data bits). This bit is ignored by the printer.

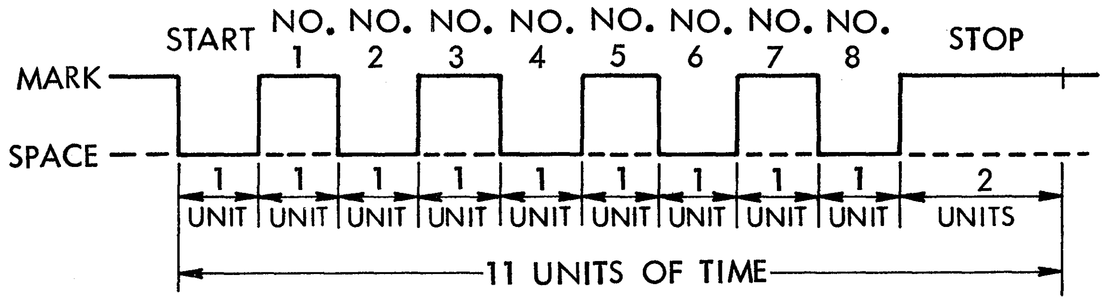
The reader/punch will transmit/receive characters exactly as they appear on the tape or as they are sent from the computer. The keyboard can generate all characters except $E0_{16}$ to FC_{16} and FE_{16} (lower case letters).

5.1.1.2 Teletype Serial Code

Data is transmitted to and from the interface to the teletype using serial, 11 unit code, at 10 characters per second. This code is illustrated in figure 5-1. Twenty milliampere,

TABLE 5-1 TELETYPE CODE

Teletype Character	Teletype Code	Teletype Character	Teletype Code	Teletype Character	Teletype Code
0	B0	blank	A0	BEL	87
1	B1	!	A1	FE	88
2	B2	”	A2	HTAB	89
3	B3	#	A3	LINE FEED	8A
4	B4	$	A4	V TAB	8B
5	B5	%	A5	FORM	8C
6	B6	&	A6	RETURN	8D
7	B7		A7	SO	8E
8	B8	(	A8	SI	8F
9	B9	)	A9	DCO	90
A	C1	*	AA	X-ON	91
B	C2	+	AB	TAPE ON	92
C	C3	,	AC	X-OFF	93
D	C4	-	AD	~~TAPE~~ OFF	94
E	C5	.	AE	ERROR	95
F	C6	/	AF	SYNC	96
G	C7	:	BA	LEM	97
H	C8	;	BB	S0	98
I	C9	<	BC	S1	99
J	CA	=	BD	S2	9A
K	CB	>	BE	S3	9B
L	CC	?	BF	S4	9C
M	CD	@	CO	S5	9D
N	CE		DB	S6	9E
O	CF		DC	S7	9F
P	D0		DD		
Q	D1		DE		
R	D2		DF		
S	D3	RUBOUT	FF		
T	D4	NUL	80		
U	D5	SOM	81		
V	D6	EOA	82		
W	D7	EOM	83		
X	D8	EOT	84		
Y	D9	WRU	85		
Z	DA	RU	86		



CURRENT WAVE FORM FOR LETTER "U"

MARK IS A PUNCH OR A 1 IN THE BINARY CODE
SPACE IS A NO PUNCH OR A 0 IN THE BINARY CODE

Figure 5-1 Teletype Serial Code.

full duplex current loops are used. A cable of up to ten feet in length is attached to an accessory plug at the rear of the enclosure.

5.1.1.3 Operating Power

Approximately 500 watts of power are required. This is supplied from the accessory plug at the rear of the computer enclosure so that the main computer power switch will control the teletype power. Depending on the model either 115V ± 10%, 60 Hz ± 0.45 Hz or 230V ± 10%, 50 ± 0.75 Hz power may be used.

5.1.1.4 Installation

The unit is mounted on a stand and is 33 inches in height. It requires floor space 22-1/4" wide × 18-1/2" deep. The unit weight is 56 pounds with stand.

5.1.1.5 Environment

Commercial Lab Environment:
0 to 40°C continuous temperature operating
0 to 90% relatively humidity without condensation

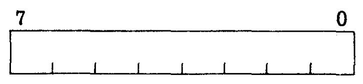
5.1.2 TELETYPE INTERFACE CHARACTERISTICS

The interface between the computer and the teletype is located on a standard 8 × 11 inch card along with the punch and reader interfaces. The interface is attached to the serial bus and uses interrupt poll positions, 2 and 1 for the punch and reader respectively. This corresponds to transfer vector addresses of 244-247 (Printer) and 248-251 (Keyboard). The poll position may be altered by backpanel wiring but should be below the reader and punch.

5.1.2.1 Interface Registers

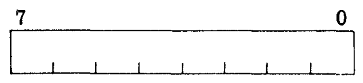
The interface contains teletype keyboard/reader input buffer (TKB), a printer/punch output buffer TPB, a keyboard/reader status register (TKS), and a printer/punch status register (TPS). These have the following format.

Input Buffer (TKB)



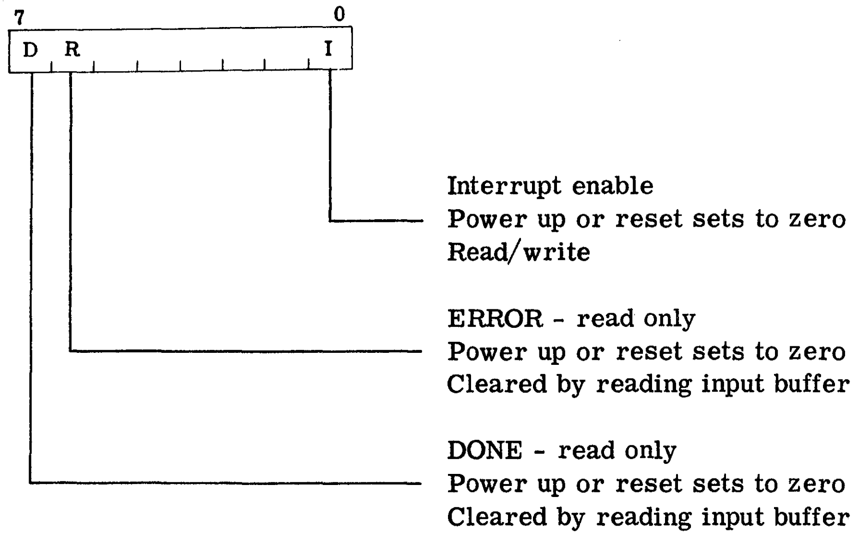
Address FFF2

Output Buffer (TPB)

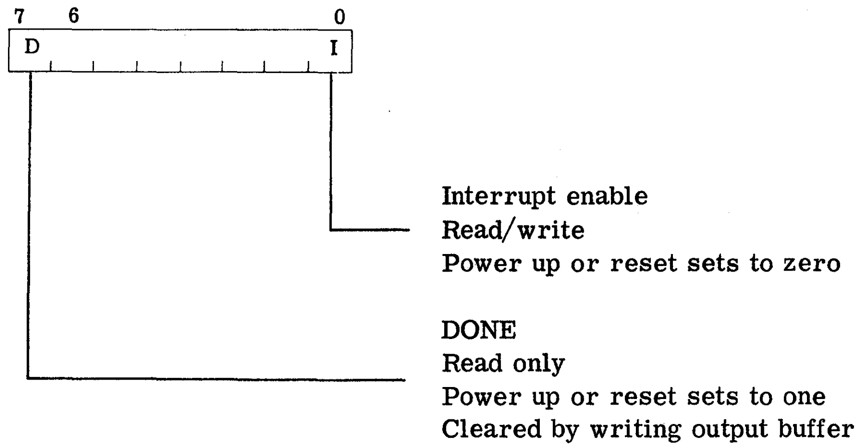


Address FFF6

Teletype Keyboard/Reader Status Register (TKS) Address FFF0


7
0
D
R
I
Interrupt enable
Power up or reset sets to zero
Read/write
ERROR - read only
Power up or reset sets to zero
Cleared by reading input buffer
DONE - read only
Power up or reset sets to zero
Cleared by reading input buffer

Teletype Printer/Punch Status (TPS) Address FFF4


7
6
0
D
I
Interrupt enable
Read/write
Power up or reset sets to zero
DONE
Read only
Power up or reset sets to one
Cleared by writing output buffer

5.1.2.2 Reader/Keyboard Interface Operation

Keying in a character or receiving a character from the reader causes the teletype to transmit a character into the interface input buffer. Receipt of the complete character causes DONE to be set in the TKS. Reading the input buffer will cause DONE and ERROR to be reset. If the teletype transmits a character before DONE has been reset the ERROR bit will be set. The input buffer must be read within 100 milliseconds of DONE to insure no loss of data.

DONE or ERROR going to a one will set an internal interrupt request bit, if interrupt is enabled. This will cause an interrupt through the serial I/O level. This bit may be reset by reading the status register.

5.1.3 PRINTER/PUNCH INTERFACE OPERATION

The computer initiates printing or punching of a character by sending it to the output buffer (TPB). This resets the DONE bit in TPS. The bit is set when the character has been transmitted and the buffer reloaded. If the interrupt is enabled the 0 to 1 transition of DONE will cause the interrupt request bit to be set. This bit is reset by reading the status register (TPS).

5.1.4 PROGRAMMING EXAMPLES

5.1.4.1 Interrupt Disabled

The following routine echoes what is entered on the keyboard onto the printer.

```
Label     Opcode     Operand          Comments

TPS       EQU        'FFF4
TKS       EQU        'FFF0
TPB       EQU        'FFF6
TKB       EQU        'FFF2
ECHO      LB         R0, TKS(2)       : Get Status
          BNM        ECHO             : WAIT until character entered
          LB         R1, TPB(2)
WAIT      LB         R0, TPS(2)
          BNM        WAIT             : WAIT until punch done
          SB         R1, TTB(2)
          BR         ECHO
```

5.1.4.2 Interrupt Enabled

The following routine transfers a block of characters to the teleprinter using interrupt. The printer routine is initiated by establishing unit control block containing the count and the address of the data to be transmitted. The setup routine starts I/O and then returns to the program. As each character is transmitted it interrupts the computer. The handler reloads the buffer until the count is exhausted.

```
Label     Opcode     Operand          Comments

          ORG        244              : TTY Printer T.V.
          DATA       TTYP             : PC
          DATA       0                : PSR

TTYP      STM        R0, 2            : Begin Handler
          LB         R0, TPS(2)       : Turn off Request

TPS       EQU        'FFF4
          L          R0, UCB(1)       : Load Count
          ADDL       R0, -1           : Decrement
          BGT        MORE
```

```
Label     Opcode    Operand          Comments

DONE      LDM       R1, 2
          RTI

MORE      S         R0, UCB(1)       : Update Count
          L         R0, UCB+2(1)     : Get Address
          LIB       R1, R0           : Get Character
          S         R0, UCB+2(1)     : Update Address
          SB        R0, TPB(2)       : Print Char and turn off DONE

TPB       EQU       'FFF6
          BR        DONE

UCB       DATA      0, 0

PRNT      L         R0, BLK          : Get Count Address
          L         R1, R0           : Get Count
          S         R1, UCB(2)       : Store Count

CHK       LB        R1, TPS(2)       : Check if Printer DONE
          BNM       CHK              : WAIT if not
          LIB       R1, R0           : Get Character
          SB        R1, TPB(2)       : Print and turn off DONE
          S         R0, UCB+2(2)     : Store Text Address
          BR        PROG             : Do rest of program

BLK       DATA      7
          TEXT      /EXAMPLE/
```

5.2 HIGH SPEED PAPER TAPE READER

5.2.1 READER CHARACTERISTICS

5.2.1.1 General Characteristics

This unit senses perforated paper tape photoelectrically at 300 characters per second using sprocket feet. Tapes may be 5 to 8 channel, oiled, unoiled, or mylar tape with less than 60% transmissivity. Fan fold boxes are provided on the 7 inch front panel with a capacity of 110 feet. The unit is attached to the serial bus through a paper tape reader interface.

5.2.1.2 Codes

Any standard codes may be used as the unit simply transfers the data punched on tape to the interface buffer.

5.2.1.3 Operating Power

Approximately 250 watts of power are required. This may be supplied from either 115V, 47-400 Hz or 230V, 50 Hz single phase AC. The power switch is located on the front panel.

5.2.1.4 Environment

Commercial Lab Environment:

Operating 0 to 55°C ambient, 90% relative humidity, 0-10K feet altitude

Non-Operating -20°C to +85°C ambient, 100% relative humidity without condensation, 0-10K feet altitude

5.2.1.5 Installation

Unit is designed for a 19 inch rack mounting. The environment in the rack must not exceed the limits of 5.2.1.4. Panel height is 7 inches and mounting depth is 8-1/2 inches. The unit weight is 19 pounds.

5.2.1.6 Interconnection

Unit signal and power connections are provided through an accessory plug in the rear of the computer enclosure. The reader is attached by a cable of up to 10 feet in length.

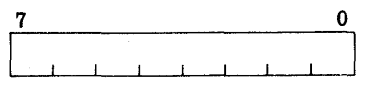
5.2.2 INTERFACE CHARACTERISTICS

The interface between the reader and the computer is mounted on a standard 8 × 11 inch card along with the punch and teletype interfaces. The interface is attached to the serial bus and uses interrupt poll position 3 corresponding to a transfer vector address of 232-235. The poll position and register addresses may be altered by backpanel wiring but should be above the punch and teletype.

5.2.2.1 Interface Registers

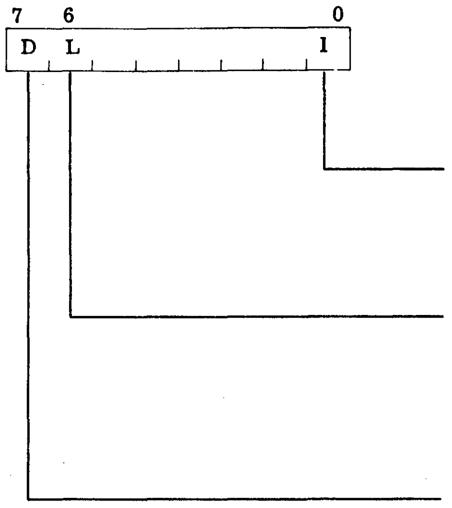
The interface contains a reader buffer (PRB) and status register (PRS) with the following formats.

Paper Tape Reader Buffer (PRB)



Address FFEA

Paper Tape Reader Status Register (PRS) - Address FFE8



Interrupt enable
Read/write
Set to zero by power up or reset

LOAD switch
Read only
Set equal to one when load switch is raised

DONE
Read only
Set to one by completion of character input
Cleared by power up, reset or reading input buffer

5.2.3 INTERFACE OPERATION

A read is initiated under program control by reading the input buffer. This causes the DONE bit to be reset and the reader to advance one character. When the character has been read the DONE bit is set. The load switch on the front panel, if high, disables the read advance and sets the load bit in the interface. Lowering the load switch enables the reader and resets the load bit in the interface. DONE may not be set after load goes high and is reset by load going high. Lowering the load switch causes the input buffer to be filled and DONE set.

If interrupt is enabled, the low to high transition of DONE or LOAD will cause an internal interrupt request bit to be set. This bit is cleared by reading the status register.

Reading the last character of a record will cause the tape to advance and a new character to be read. If no interrupt is desired, the interrupt enable should be turned off before the last read.

5.2.4 PROGRAMMING EXAMPLE

To read data until a rubout (FF) is encountered and store it in memory without interrupt.

```
Label    Opcode     Operand

MEM       EQU      1000

PRS       EQU      'FFE8

PRB       EQU      'FFEA
```

```
Label    Opcode    Operand          Comments

STRT     L         R0, MEM        : Load Address

READ     LB        R1, PRS(2)     : Get Status
         BNM       READ           : Wait until done set
         LB        R1, PRB(2)     : Get Data, turn off DONE and get NEXT
         CMPB      R1, 'FF        : Check if RUBOUT

DONE     BZ        *
         SI        R1, R0         : Store Data
         BR        READ           : Go to get new character
```

5.3 HIGH SPEED PAPER TAPE PUNCH

5.3.1 DEVICE CHARACTERISTICS

5.3.1.1 General Characteristics

This unit is capable of perforating standard fanfold tapes at a rate of 75 characters per second. The unit contains its own power supplies and is designed for rack mounting. The punch is attached to the serial I/O bus by an interface located within the computer enclosure.

5.3.1.2 Tape Characteristics

Five or eight channel, fanfold, oiled, unoiled, or mylar tapes may be used. They should have a thickness between 3 and 4.3 mils.

5.3.1.3 Power

Requires 175 VA of power from either 115V, 47-440 Hz or 230V, 50 Hz single phase AC.

5.3.1.4 Installation

The unit is designed for rack mounting in a standard 19 inch rack. Panel height is 10-1/2 inches and depth behind panel is 12 inches. The unit weight is approximately 38 pounds. The unit is mounted on chassis slides for access to load paper tape.

5.3.1.5 Environment

Commercial Lab Environment
Temperature: -5°C to +55°C operating ambient
Humidity: 10% to 90% relative humidity without condensation

5.3.1.6 Operator Controls

a. Power On/Off Control

b. Tape Feed Control - momentary switch to cause punching of loader at 75 cps

c. Run/Load Control - RUN position enables punch operation LOAD position enables tape to be loaded and causes Tape Error

d. Tape Low Indication - Indicates tape supply nearly exhausted

5.3.1.7 Interconnection

Unit is attached to interface by a ten foot signal and power cable. This mates with an accessory plug at the rear of the computer enclosure.

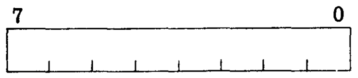
5.3.2 INTERFACE CHARACTERISTICS

The interface between the punch and the computer is mounted on a standard 8 × 11 inch card along with the reader and teletype interfaces. The interface is attached to the bus and uses interrupt poll position 2 corresponding to a transfer vector address of 240-243. The poll position and register addresses may be altered by backpanel wiring but the poll position should be below the reader and above the teletype.

5.3.2.1 Interface Registers

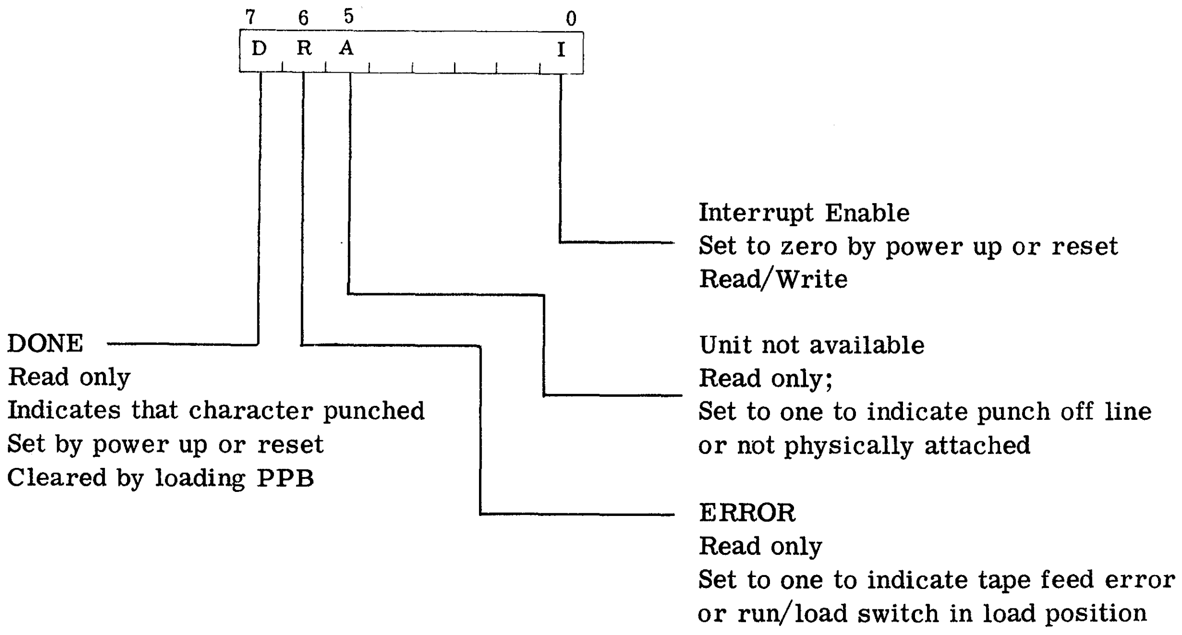
The interface contains a punch buffer (PPB) and status register (PPS) with the following formats:

Paper Tape Punch Buffer (PPB)



Address FFEE

Paper Tape Punch Status Register (PPS) - Address FFEC



7
6
5
0
D
R
A
I
Interrupt Enable
Set to zero by power up or reset
Read/Write
DONE
Read only
Indicates that character punched
Set by power up or reset
Cleared by loading PPB
Unit not available
Read only;
Set to one to indicate punch off line
or not physically attached
ERROR
Read only
Set to one to indicate tape feed error
or run/load switch in load position

5.3.3 INTERFACE OPERATION

The computer initiates a punch cycle by loading a character into the buffer. This resets the DONE bit which is set again at the end of the punch cycle. The status register should be checked before the write cycle to insure that the punch is available, has no errors and has finished its last cycle.

If the unit is not available or has an error DONE and the punch cycle are inhibited. Data loaded into PPB will be ignored in this condition.

If the interrupt is enabled, an interrupt request bit will be set by either DONE, ERROR, or Unit Not Available going to a one. This request bit will be reset when the status register is read.

5.3.4 PROGRAMMING EXAMPLE

To transfer a word from memory to the punch without interrupt:

```
Label     Opcode     Operand              Comments
MEM       TEXT       /EXAMPLE/

PPS       EQU        'FFEC

PPB       EQU        'FFEE

STRT      LM         R0, MEM           : Get address of data
          MOVL       R1, 7

PNCH      LMB        R2, PPS(2)        : Get status
          BNM        PNCH              : Wait until ready
          LI         R2, R0            : Get data
          SMB        R2, PPB(2)        : Store data and turn off DONE
          BCT        R1, PNCH          : Test and branch if more data

DONE      BR         *                 : Finished
```

5.4 REAL TIME CLOCK (RTC)

5.4.1 DEVICE CHARACTERISTICS

This unit provides a way of generating interrupts as a rate set by program control. Four rates may be selected as follows:

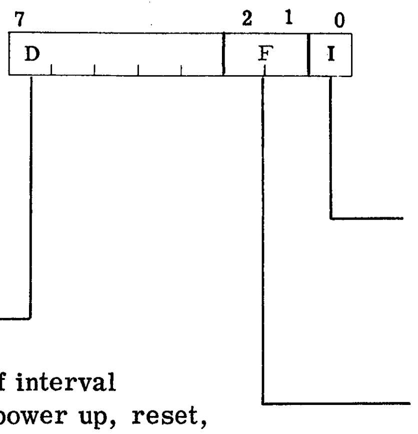
a. 1000 Hz
b. 200 Hz
c. 100 Hz
d. 20 Hz

The unit is attached to the serial bus through a real time clock interface.

5.4.2.1 Interface Register

The interface contains a status register with the following format:

Real Time Clock Status Register (RTS) Address FFE6



Interrupt enable
Read/write
Set to zero by power up or reset

DONE
Read Only
Set by end of interval
Cleared by power up, reset, or reading status register

Frequency
Read/write
Set to zero by power up or reset

00	20 Hz
01	100 Hz
10	200 Hz
11	1000 Hz

5.4.3 INTERFACE OPERATION

The DONE bit will be set at a rate determined by the selected frequency. If interrupt is enabled this will generate an interrupt request. This will be cleared and done reset by reading the status register. DONE must be cleared before the end of the next interval after the one which set it in order not to lose a clock.

5.4.4 PROGRAMMING EXAMPLE

This routine generates a 65 second timer in storage with a 1 ms resolution.

```
Label   Opcode   Operand        Comments

TIME    DATA     0

RTS     EQU      'FFE6

STRT    MOVL     R0,'06         : Set to 1 KHz, no interrupt
        SMB      R0, RTS(2)

WAIT    LMB      R0, RTS(2)     : Turn off DONE if set
        BNM      WAIT           : Was DONE set
        LM       R0, TIME(1)
        ADDL     R0, 1
        SM       R0, TIME(1)
        BNV      WAIT           : Overflow after 65 seconds
        BR       EXIT
```

5.5 SERIAL BUS PERIPHERAL ADDRESSES

The following are standard address assignments and poll positions for the devices on the serial bus. Latency times are the time within which the done bit must be reset after being set to ensure maximum speed operation.

Device	Register	Address	T.V.	Poll Position	Latency
Teletype Keyboard/Reader	TKB	FFF2	248	1	100 ms
	TKS	FFF0			
Teletype Printer/ Punch	TPB	FFF6	244	2	100 ms
	TPS	FFF4			
Paper Tape Punch	PPB	FFEE	240	3	13.3 ms
	PPS	FFEC			
Real Time Clock	RTS	FFE6	236	5	1 ms
Paper Tape Reader	PRB	FFEA	232	4	3 ms
	PRS	FFE8			
Control Panel	DSR	FFFA	252	0	Indefinite
	DDR	FFFD			
	DAR	FFFC			

SECTION 6

OPERATOR'S CONTROL PANEL

6.0 INTRODUCTION

The MIP-116 Operator's Control Panel (OCP) permits the processor to be controlled from distances of up to fifty feet. Using the panel, data in memory or processor registers can be read or altered. The panel is shown in figure 6-1.

Using the OCP the operator may perform any of the following functions:

a. Read and modify any of the 16 general purpose registers
b. Read and modify PC
c. Read and modify PSR
d. Read and modify any I/O address
e. Inhibit power up or down interrupts
f. Control execution of instructions
g. Control execution of microinstructions
h. Reset system
i. Power ON/OFF control
j. Initiate interrupt

For debugging, these provide the capability of halting program execution, reading and modifying registers or memory, perform single instruction execution, and then resume normal execution. Program start functions include initializing any register or memory location, resetting system, loading PC and PSR and then initiating execution.

The OCP may be used as an I/O device under program control. The programmer may load the data display and read the address register or switch register. The operator may initiate interrupts using the control panel interrupt.

6.1 OCP DISPLAYS

6.1.1 DATA DISPLAY

This consists of 16 binary indicators located above the data switches. An indicator is lit to indicate a binary one.

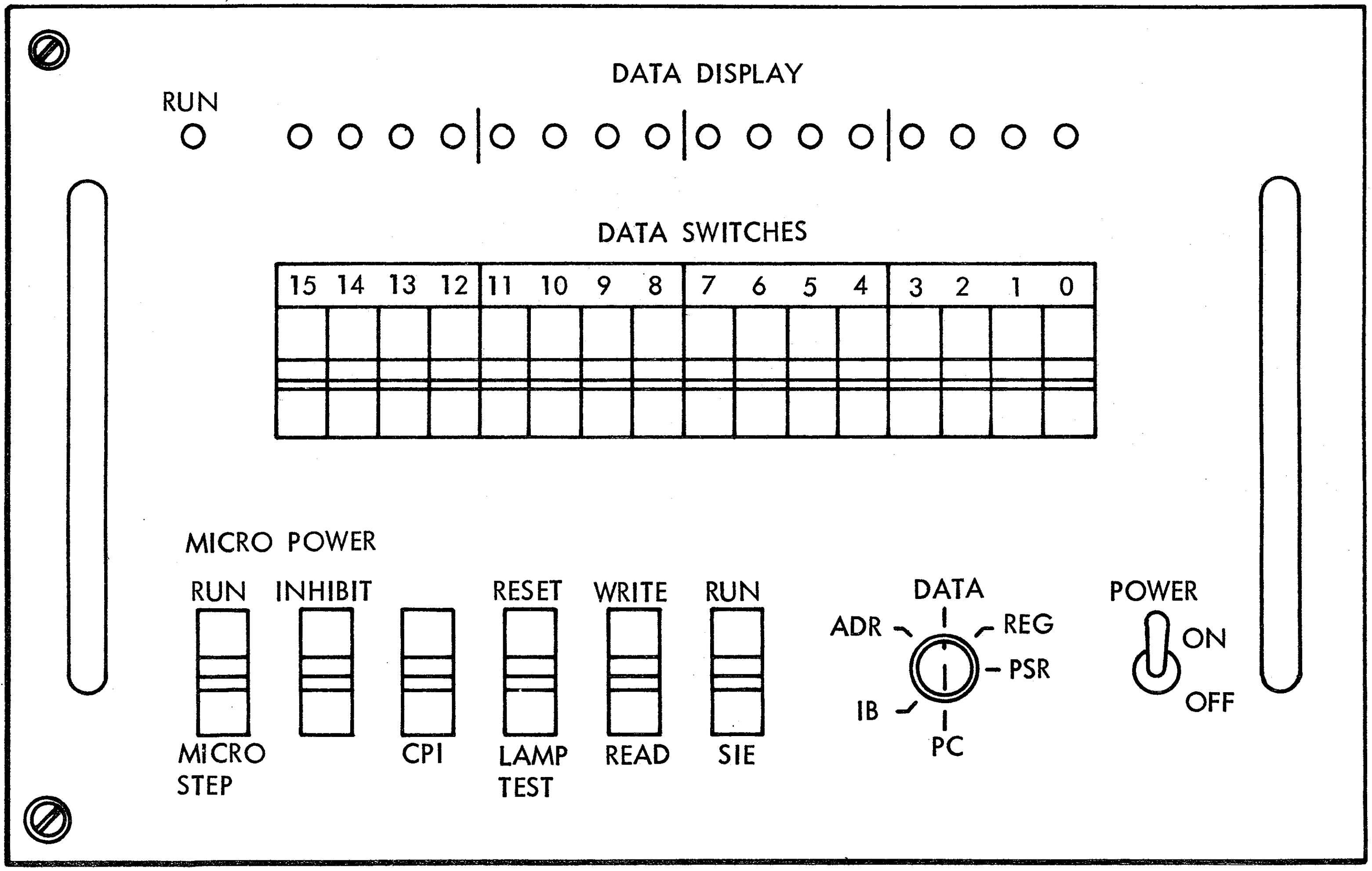


Figure 6-I Operators Control Panel.

6.1.2 INDICATORS

6.1.2.1 RUN/HALT

This indicator is lit when the computer is executing instructions.

6.2 OCP CONTROLS

The OCP has six control switches and a display selector switch.

6.2.1 RUN/HALT/SIE SWITCH

This three position controls of the execution of instructions. Pressing the switch up to the RUN position will latch it and cause execution to begin at the address in the program counter. In RUN position the Read/Write and Reset controls are locked out.

Moving the switch to the middle position will cause execution to halt at the end of the instruction being executed. The address of the next instruction will be displayed on the data display.

Depressing the switch into the momentary down position will cause a single instruction to be executed at the address specified by PC. The address of the next instruction will be displayed at the end of the cycle.

6.2.2 SELECTOR SWITCH

This operates in conjunction with the read/write switch to examine/alter registers or memory. It is a six position rotary switch with the following positions:

a. Address
b. Data
c. Register
d. PSR
e. PC
f. IB

The display will be driven by the OCP address register when the switch is in the address position and by the data register in any other position.

6.2.3 WRITE/READ SWITCH

This momentary up/down switch is used in conjunction with the selector switch to examine/alter registers or memory.

6.2.3.1 Address Selector Switch

The OCP address register is displayed by the selector switch. A Read has no effect. A Write transfers the data on the switch register into the control panel address register.

6.2.3.2 Data Selector Switch

A Read will cause the memory location addressed by the OCP address register to be read and loaded into the data register where it will be displayed. The second read with the selector switch in this position after address register load will cause the address register to be incremented by 2 and then the data to be fetched and displayed.

A Write will cause the memory location addressed by the OCP address register to be loaded from the data switches. A second write with the selector switch in this position after a read or loading the address register will cause the address register to be incremented by 2 and the data switches to be transferred to memory.

With the selector switch in this mode the contents of memory may be examined, altered or loaded. Read sequences are used to examine successive locations. Read-Write sequencies are used to read and then alter that word in successive locations. Write sequencies are used to load memory locations.

Note that memory and parallel and serial I/O device registers may be read or written with this mode. Operations are always performed in word mode.

6.2.3.3 Register Selector Switch

This mode is used to examine, alter or load any of the sixteen general registers. The OCP address register is loaded with addresses 0 through 30 to read or write registers 0 through 16 respectively. Operation from that point is the same as for the Data Selector Switch mode described above.

6.2.3.4 PSR Selector Switch

In this mode, a Read transfers the PSR onto the display and a write loads the PSR from the data switches. The address register is not used and will not count.

6.2.3.5 PC Selector Switch

A Read transfers the PC onto the display and a write loads the PC from the data switches. The address register is not used and will not count.

6.2.3.6 Selector Switch

A Read transfers the instruction addressed by PC onto the display and Write loads the memory location addressed by PC. The OCP address register is not used and will not count.

6.2.4 RESET/LAMP TEST SWITCH

Raising this switch to the momentary up position causes the system to be reset. Depressing the switch to the momentary down position forces all the lights in the display to be lit. Reset is not active if in RUN mode.

6.2.5 CONTROL PANEL INTERRUPT (CPI) SWITCH

Depressing this momentary down switch causes an interrupt to be initiated which may be serviced under program control. This switch is only active in RUN mode.

6.2.6 POWER UP/DOWN INHIBIT

Raising this switch to the latched up position inhibits the normal power up/down interrupt operation. Removing the inhibit after a power up will not cause a power up sequence.

6.2.7 MICROINSTRUCTION RUN/HALT/STEP

This switch should be normally latched up and is used only for diagnostic purposes. Raising this switch to the latched up position permits microinstructions to be executed by the processor. In the middle position no microinstructions are executed. Depressing the switch to the momentary down position causes the processor to execute one microinstruction. Normal operation of the control panel may only be obtained if this switch is in the latched up position.

6.3 OCP PROGRAMMING

The OCP may be used as an I/O device by the programmer. It is attached to the serial I/O bus at the lowest interrupt poll position. This capability is used by the microroutines to provide the function capability described above.

6.3.1 OCP I/O REGISTERS

6.3.1.1 Address Register

This is a 16 bit counter which may be loaded and incremented by the front panel controls. It is read only and is addressed by address FFFC in word mode.

6.3.1.2 Data Switch Register

This is a 16 bit register loaded from the 16 panel data switches. It is read only and is addressed by FFFA in word mode.

6.3.1.3 Data Display Register

This is a 16 bit register used to drive the display when the selector switch is not in address position. It is write only and is addressed by FFFD in word mode.

6.3.1.4 Function Switch Register

This 8 bit register is used to indicate control panel functions. It is active only when the panel is not in RUN mode and is used only by the microprogram to handle control panel function requests such as read or write. It is read only and is addressed by FFFF in byte mode.

6.3.2 OCP I/O OPERATION

The programmer may load data into the display register or read data from the address or switch register at any time. Control Panel Interrupt sets a serial interrupt request bit which is reset by performing any of the three I/O operations.

In use, the operator will set up data on the data switches and then indicate its availability by hitting CPI. Or the computer may load a control word into the display to indicate its progress in a program. CPI could be used here to indicate operator response.

The transfer vector for CPI is located in positions 252-255.

6.4 OCP INTERCONNECTION

When attached as part of computer enclosure the OCP is wired directly to the backpanel. When operated remotely the OCP is connected to the serial I/O bus (5 pairs) but requires 3 additional twisted pair connections as well. These include:

a. Function Request
b. Microinstruction Control
c. Power Up/Down Inhibit

A total of only 16 wires are required to connect the OCP to the remote processor.

SECTION 7
INTERFACING

7.0 INTRODUCTION

The CPU communicates with all peripheral devices and memories via two busses.

The Parallel Asynchronous I/O Bus is used for instruction fetch and high speed data transfer. This bus utilizes parallel address and data busses to achieve transfer rates of 5 MHz.

The Serial I/O Bus transmits addresses and data serially, two bits in parallel, over bi-directional lines to achieve 250 KHz transfer rates.

Both busses operate in a polling mode to affect priority assignment for interrupts.

One interrupt level is associated with the Serial I/O bus and sixteen devices may be assigned to that level. The Serial I/O interrupt has lowest priority.

The next four higher levels of interrupt are assigned to the Parallel I/O bus. Each of these levels may have eight devices assigned.

Operation of the data transfer and interrupt polling for the Parallel and Serial I/O busses is described in Sections 7.1 and 7.2 respectively.

7.1 PARALLEL ASYNCHRONOUS I/O BUS

7.1.1 GENERAL BUS DESCRIPTION

The Parallel Asynchronous I/O Bus is used for instruction fetch, data fetch, data store, and priority interrupts. Devices and memories of different types, sizes and speeds may be attached to the bus.

Maximum word transfer rate over the bus is limited by the speed of the device and memory, but can be as high as 8 MHz. Words as well as bytes may be transferred. The total addressing capability is 64 K bytes or 32 K words.

External devices may share the bus to effect Direct Memory Access (DMA). This takes place through "cycle stealing" while the processor is operating.

Devices requiring priority interrupts also utilize the Bus for interrupt requests and interrupt device poll.

A block diagram of the Parallel Asynchronous I/O Bus is shown in Figure 7-1.

Devices and memories are connected to the bus on a command-response basis. Each device connected to the bus contains an I/O mode decoder and an address decoder for function and device selection.

Devices which do not require bus control for DMA connect to the I/O Address, I/O Data, I/O Mode, I/O Initiate, and I/O Response lines. Devices which require bus control for DMA also connect to the Bus Request and Bus Grant lines. Devices requiring priority interrupt connect to one of four interrupt lines, the I/O Initiate, I/O Mode and I/O Data lines.

Each of the 43 I/O bus signals is transmitted over a twisted pair, giving a total of 86 connections for the bus including signals and return. External connections are made via connectors at the rear of the enclosure. When external devices are not connected to the bus, termination plugs must be in place at the bus connectors.

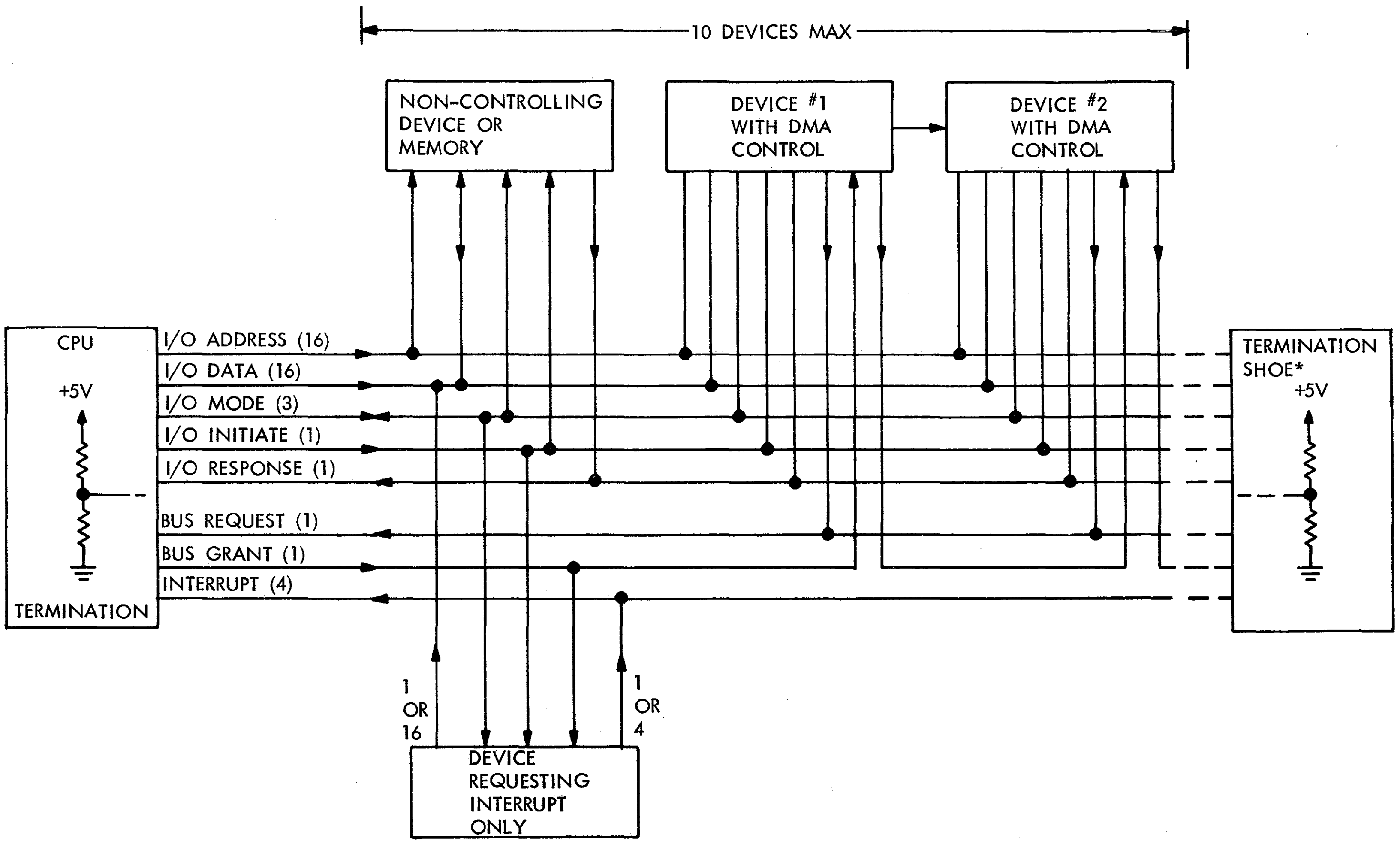
The maximum bus length must not exceed five feet total, but to minimize line propagation times it is desirable to place high speed devices as close to the CPU as possible. The Parallel Asynchronous Bus is not designed to be connected between units in a high noise environment.

7.1.2 BUS SIGNAL DESCRIPTIONS

Table 7-1 shows the signals which comprise the Parallel Asynchronous I/O Bus. All signals are active-low logic with the exception of the MODE lines which are a three-bit encoded function.

Table 7-2 describes the code used for the I/O MODE functions.

All signals are transmitted by open-collector gates over twisted pairs having a characteristic impedance of approximately 120 ohms. Signal returns are run through the connectors on each interface card and should be grounded at the ground pin of the circuit which drives or receives the associated signal. Drive capability of each signal and the number of loads presented to the bus by the processor is shown in Table 7-1



*TERMINATION SHOE MAY BE REPLACED WITH TERMINATING RESISTORS ON LAST DEVICE INTERFACE CARD ON BUS.

Figure 7-1 Parallel Asynchronous I/O Bus.

TABLE 7-1
ASYNCHRONOUS PARALLEL I/O BUS SIGNALS

					D-C CHARACTERISTICS *			
					HIGH LEVEL (MA)		LOW LEVEL (MA)	
SIGNAL	MNEMONIC	NUMBER OF PAIRS	ACTIVE LOGIC LEVEL	DRIVE STRUCTURE	FANOUT	FAN IN	FANOUT	FAN IN
I/O DATA	IOD	16	LOW	OPEN COLLECTOR	0.9	100 μA	16.4	3.6
I/O ADDRESS	IOA	16	LOW	TOTEM POLE	1.0	-	20.0	
I/O MODE	MODE	3	ENCODED	OPEN COLLECTOR	1.0	-	20.0	
I/O INITIATE	INIT	1	LOW	TOTEM POLE	1.0	-	20.0	
I/O Response	RSPNS	1	LOW	OPEN COLLECTOR	-	50 μA	-	2.0
DMA Request	RQST	1	LOW	OPEN COLLECTOR	-	40 μA	-	1.6
DMA Grant	GRNT	1	LOW	TOTEM POLE	1.0	-	20.0	
INTERRUPT REQUEST	PIOI	4	LOW	OPEN COLLECTOR	-	40 μA	-	1.6

*Loading includes effect of external terminations shoes.

TABLE 7-2
MODE FUNCTIONS

MODE 2	MODE 1	MODE Ø	
L	L	L	I/O Store Byte
L	L	H	I/O Fetch
L	H	L	I/O Store Word
L	H	H	I/O Reset
H	L	L	Poll Level 3
H	L	H	Poll Level 2
H	H	L	Poll Level 1
H	H	H	Poll Level Ø

L = Low Voltage Level
H = High Voltage Level

7.1.3 DESCRIPTION OF BUS OPERATIONS

Figures 7-2 through 7-6 are typical timing diagrams for the various bus operations. Table 7-3 lists the minimum, typical and maximum intervals shown in the timing diagrams. Note that all timing diagrams show active-high logic signals, whereas the actual logic sense is as shown in Table 7-1 .

7.1.3.1 Data or Instruction Fetch (Figure 7-2)

Data or instruction fetches are begun at the processor cycle-execute time. The processor gates the IOA and MODE lines and then initiates the Fetch by activating the INIT line.

The device may immediately access data since the address is valid. During the access time of the device or memory the higher order address bits are decoded, and after the device data is valid the data will be gated to the IOD lines. The device also activates the RSPNS line, indicating that valid data is on the IOD lines.

The processor receives the response, accepts the data, and releases the INIT line, indicating that the data has been accepted. The device immediately removes data from the IOD lines, resets, and waits for the next INIT command.

Alternatively the device address decoding may be done prior to accessing the device, but the device address decoder will add directly to the effective access time of the device.

For word data fetches the 16 data bits from the IOD bus are loaded into the appropriate register. For byte data fetches the data from the appropriate byte address is loaded into the least significant byte of the designated register.

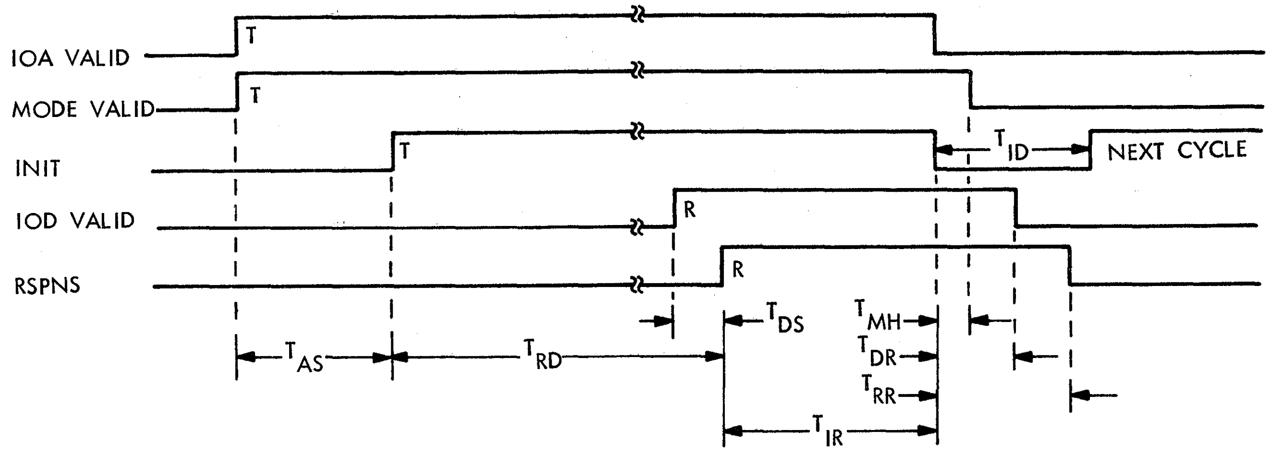
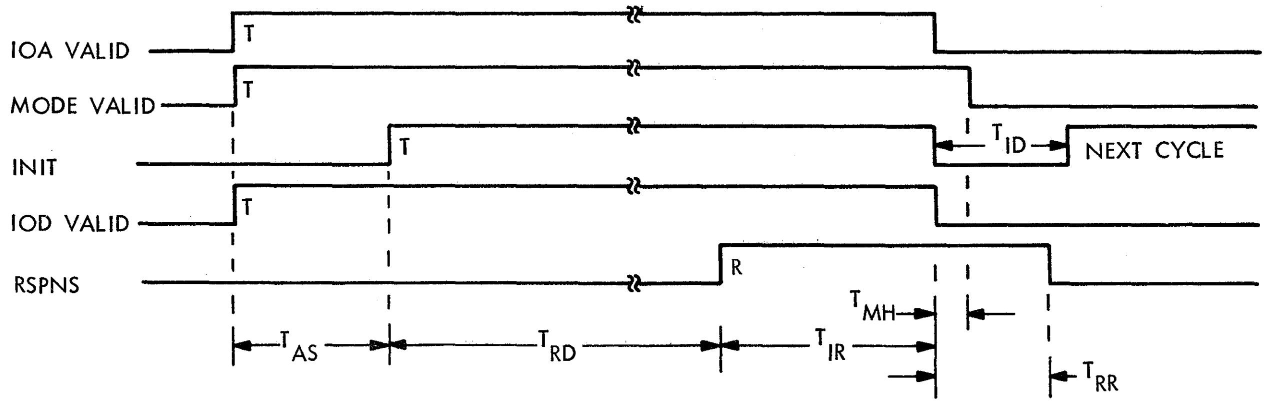


Figure 7-2 Data or Instruction Fetch.



T DENOTES SIGNAL TRANSMITTED BY PROCESSOR

R DENOTES SIGNAL RECEIVED BY PROCESSOR

Figure 7-3 Data Store Timing.

7.1.3.2 Data Store (Figure 7-3)

Data stores may be either byte or word operations depending upon the state of the MODE lines. For word stores, the entire 16-bit word on the IOD bus is stored at the appropriate word address. For byte stores, the least significant eight bits of the IOD bus are stored at the appropriate byte address. The upper eight bits of the IOD bus are ignored.

As with fetches, data stores are begun at processor execute time. The processor gates the IOA, MODE, and IOD lines and then initiates the store operation by activating the INIT line.

The device decodes the most significant bits of the IOA bus, determining whether it is being addressed. The addressed device then loads the data from the IOD bus into a buffer, and activates the RSPNS line, indicating to the processor that data has been accepted. The processor releases the INIT line and continues other operations while the device completes data storage.

Alternatively the device may capture the processor for a maximum of 10 μsec in applications where a data buffer in the device is not desirable. During this period, the processor will wait for the RSPNS line to be activated and will hold the I/O bus stable. At the end of the 10 μsec period the processor will be interrupted through the internal I/O ERROR interrupt.

7.1.3.3 DMA Trap (Figure 7-4)

Devices requiring bus control for DMA are connected in hard-wired priority configuration shown in Figure 7-9 below.

Devices requesting the bus will activate the RQST line, causing the processor to relinquish bus control at the end of its present cycle.

The processor responds by activating the GRNT line and the INIT line which propagates to the requesting device with the highest priority. The MODE lines are in the "Poll level zero" state and the DMA address to be supplied by the requesting device is gated onto the IOD bus. The RSPNS line is activated by the last device in the priority chain, indicating that the DMA address is present on the IOD bus. The INIT signal is removed by the CPU and the DMA address is gated onto the IOA bus by the CPU. The requesting DMA device activates the MODE and IOD lines. The CPU activates the INIT signal causing DMA transfer. The RSPNS signal causes GRNT to be deactivated, ending DMA.

7.1.3.4 Parallel I/O Interrupt (Figure 7-5)

Any device connected to the Parallel I/O bus may cause an interrupt at any of the four parallel I/O interrupt levels. A maximum of eight devices per level are allowed, for a total of 32 parallel I/O interrupts.

The requesting devices are assigned a level of interrupt (0 to 3, with level 3 having highest priority) and a bit on the IOD bus (IOD0 to IOD7 with bit 7 having highest priority).

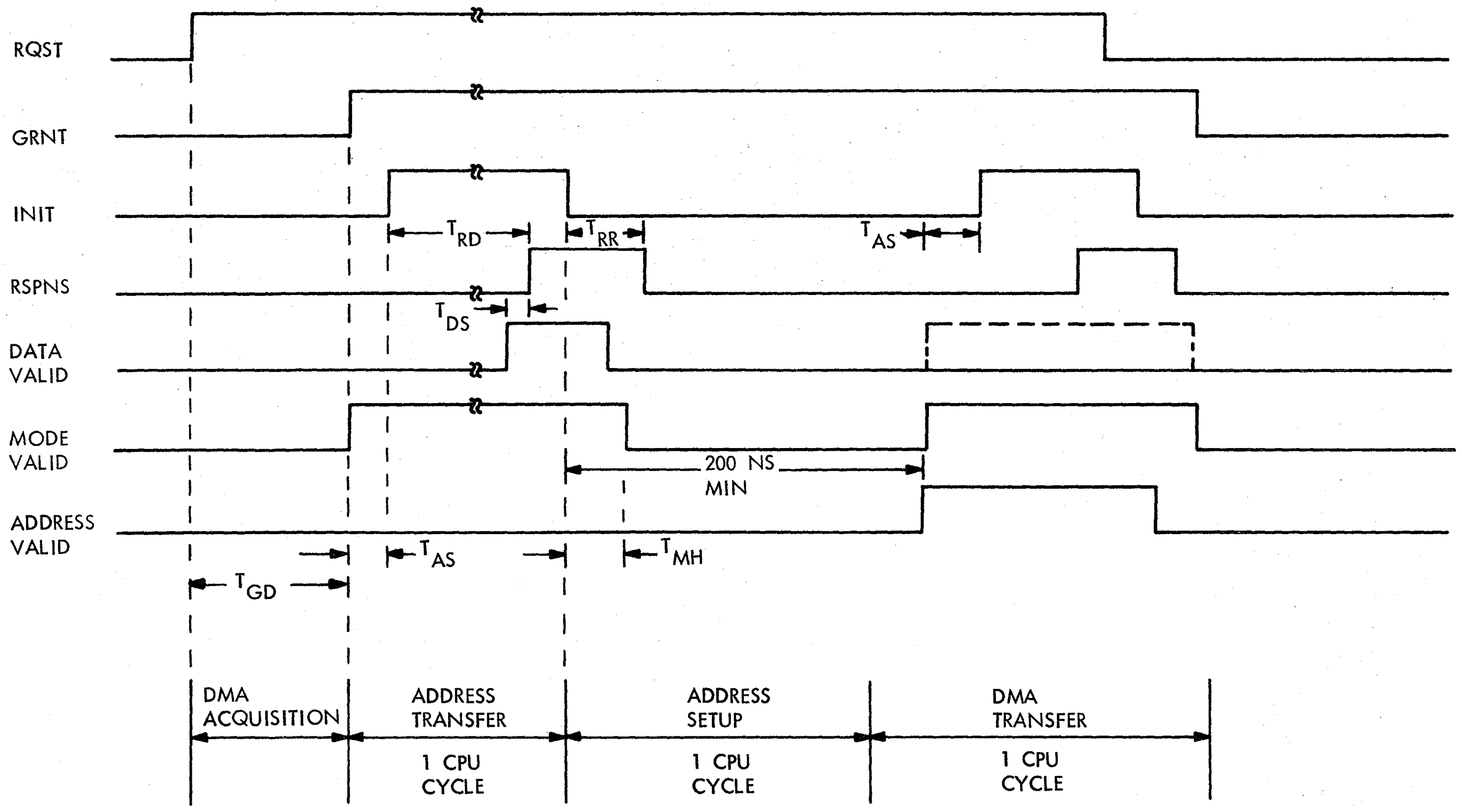


Figure 7-4 DMA Transfer Timing.

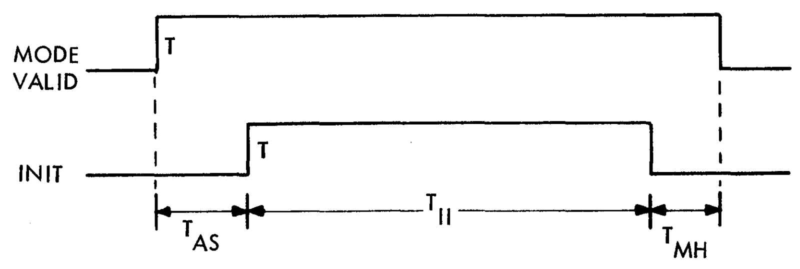


Figure 7-5 Reset Timing.

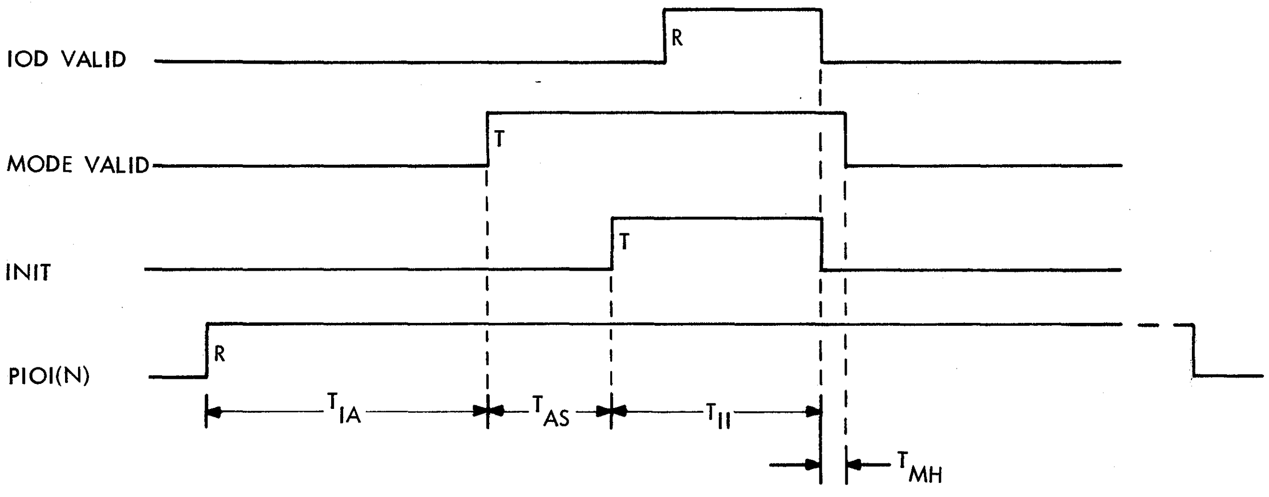


Figure 7-6 Interrupt Poll Timing.

To cause an interrupt, the requesting device activates the appropriate PIOI line, causing the processor to sense a pending interrupt at the end of the present instruction execution.

The processor will toll the highest requesting level by issuing a POLL LEVEL N (N = 0, 1, 2, 3) command on the MODE lines.

Each requesting device assigned to level N will activate one of the eight least significant bits of the IOD bus, indicating that an interrupt has been requested. Priority selection is made and control is transferred to the interrupting routine.

TABLE 7- 3
MIP-116 BUS TIMING

Interval		Time (nanoseconds)		
		Min	Typ	Max
T_{AS}	Address, Mode, Data Out Setup Time	85	100	115
T_{RD}	Device Response Delay	-	*	10 μsec
T_{DS}	Processor Data Setup Time	0	10	-
T_{IR}	Initiate Release Time	15	30	45
T_{MH}	Mode Hold Time	10		
T_{DR}	Data Release Time	0	*	30
T_{RR}	Response Release Time	0	*	45
T_{ID}	Initiate Dead Time	-	100	-
T_{GD}	Grant Delay		200	**
T_{II}	Initiate Width		100	
T_{IA}	Interrupt Acknowledge Time		**	

*Length of interval depends upon device characteristics
**Depends upon instruction being executed

7.1.4 TYPICAL INTERFACE CONNECTIONS

Figures 7-7 through 7-13 show typical circuits which may be used to interface to the Parallel Asynchronous I/O Bus.

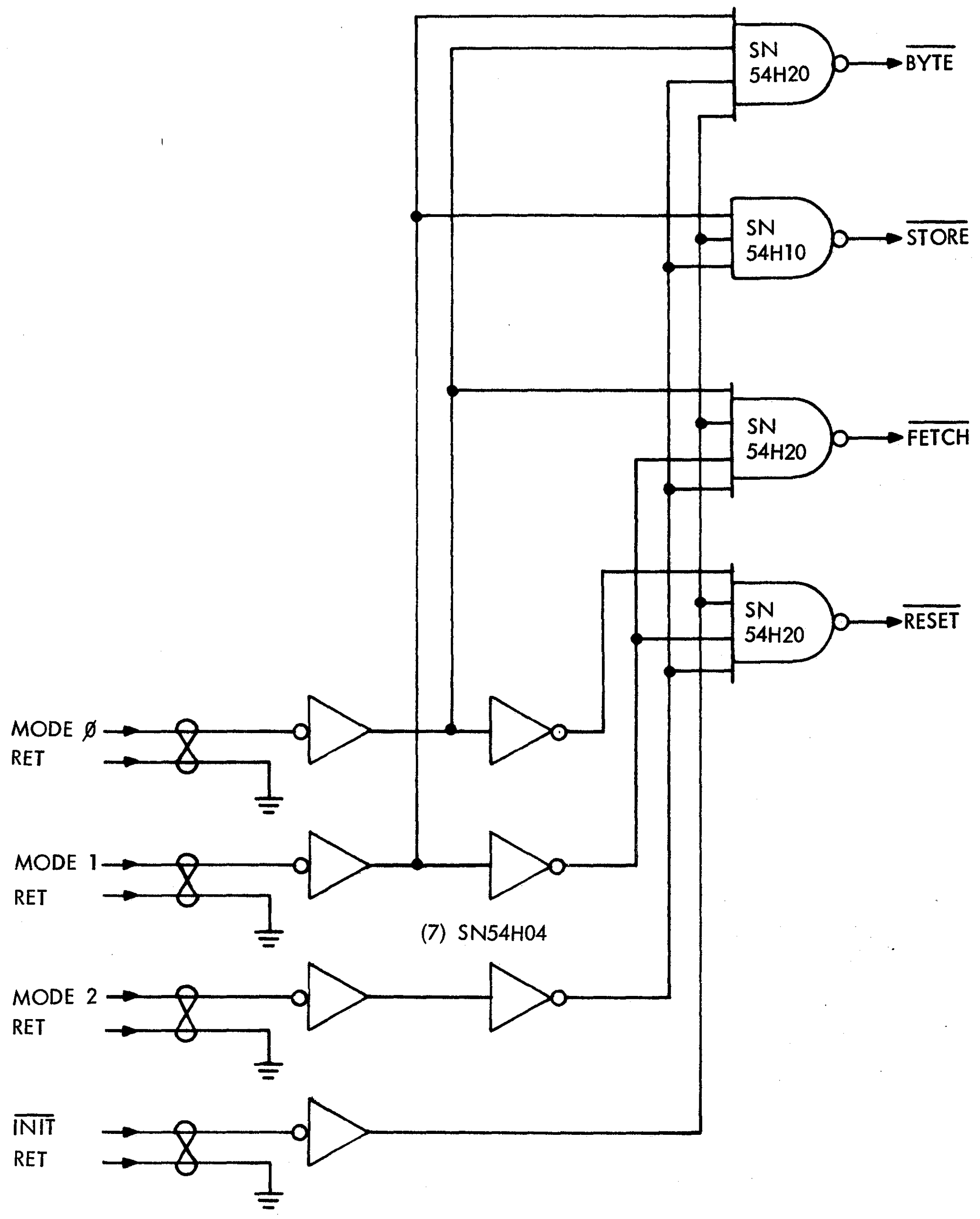


Figure 7-7 Typical Control Decoder

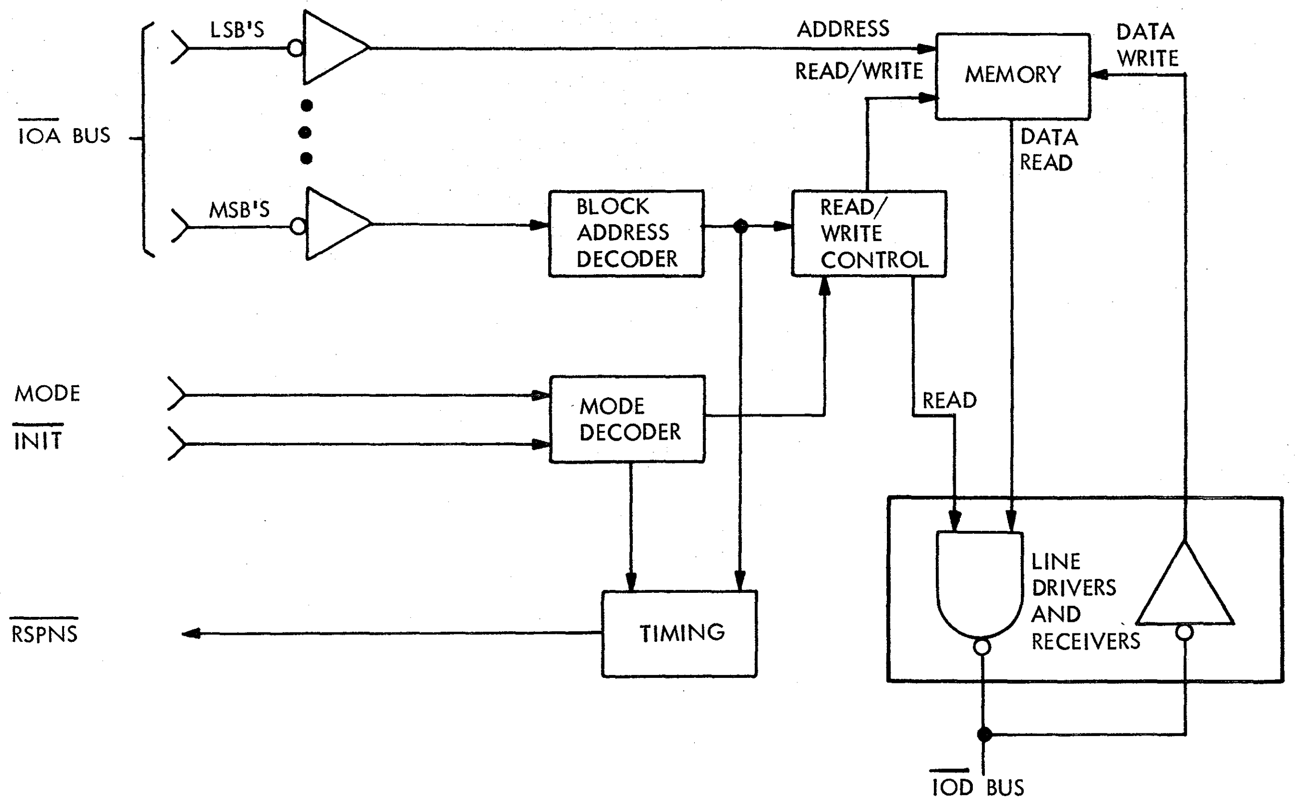


Figure 7-8 Typical Memory Interface

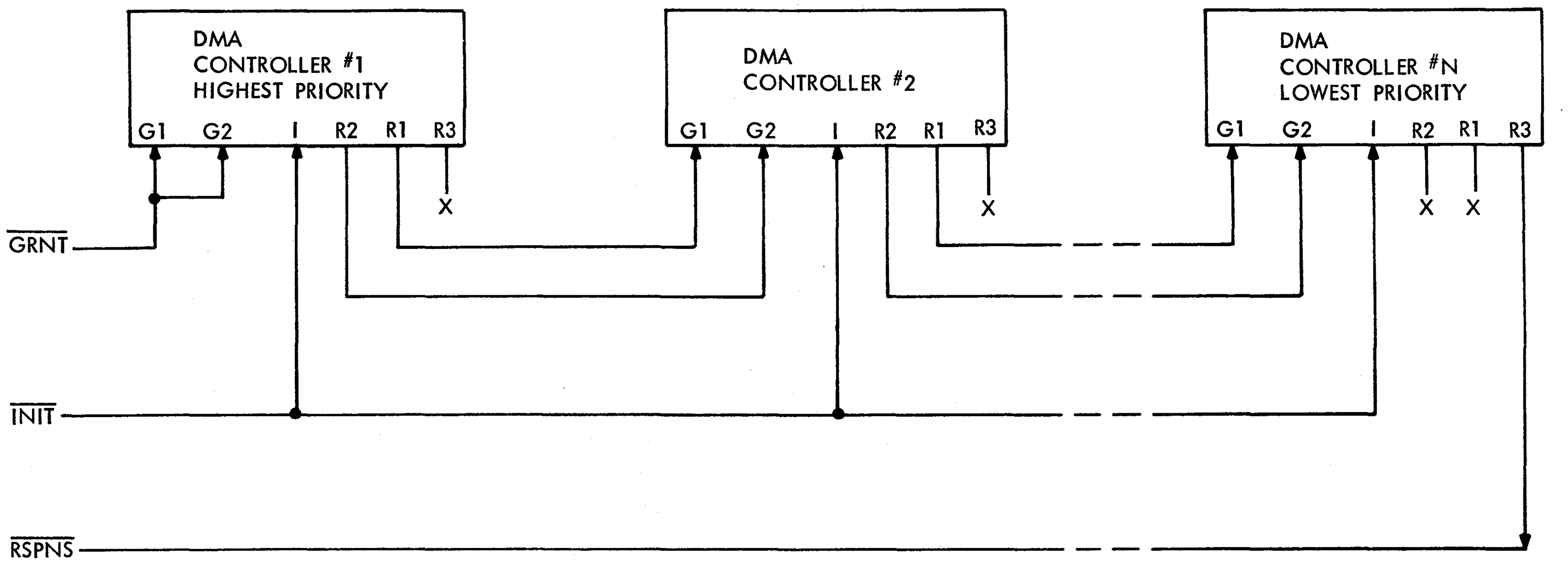

DMA CONTROLLER #1 HIGHEST PRIORITY
DMA CONTROLLER #2
DMA CONTROLLER #N LOWEST PRIORITY
G1
G2
I
R2
R1
R3
X
GRNT
INIT
RSPNS

Figure 7-9 Typical DMA Connections

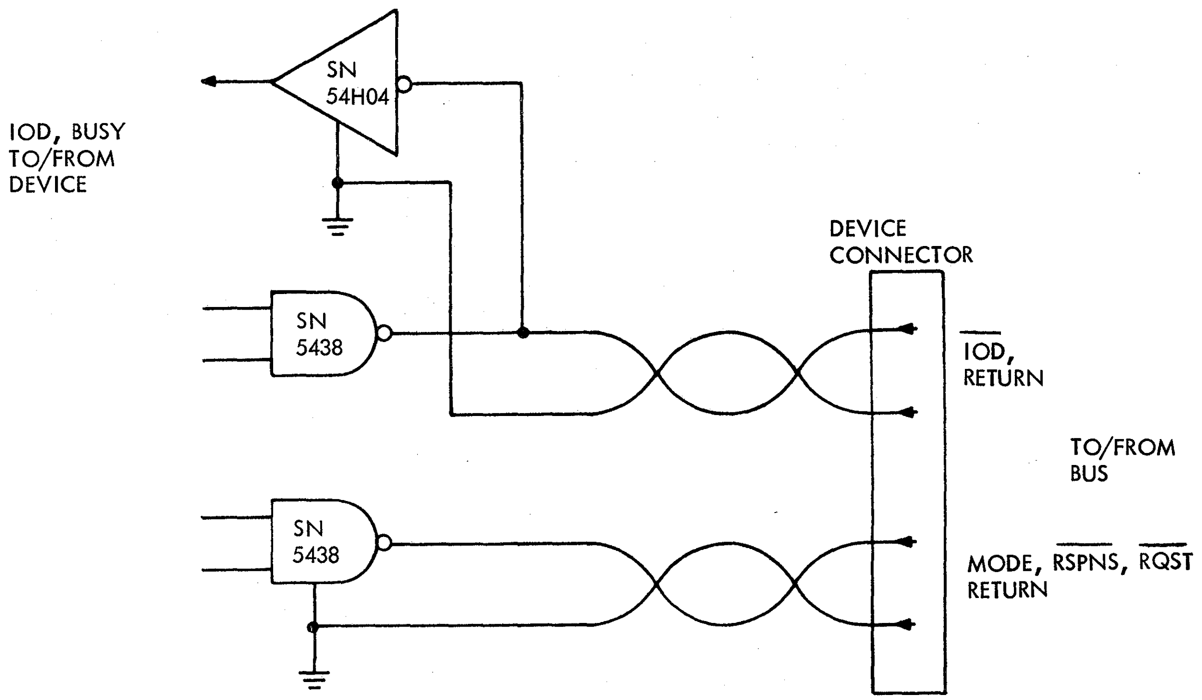


Figure 7-10 Standard Line Drivers

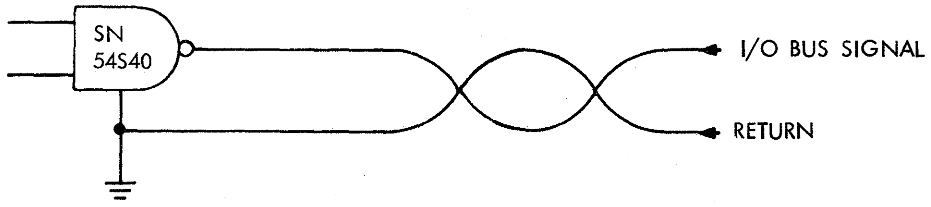
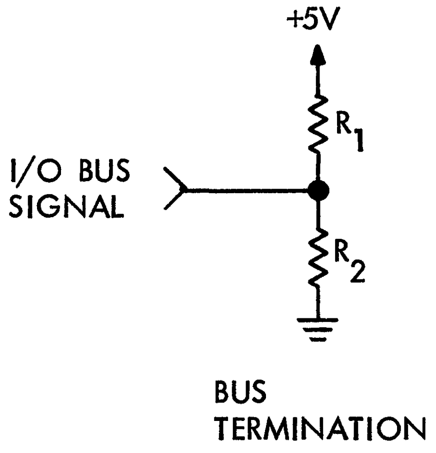
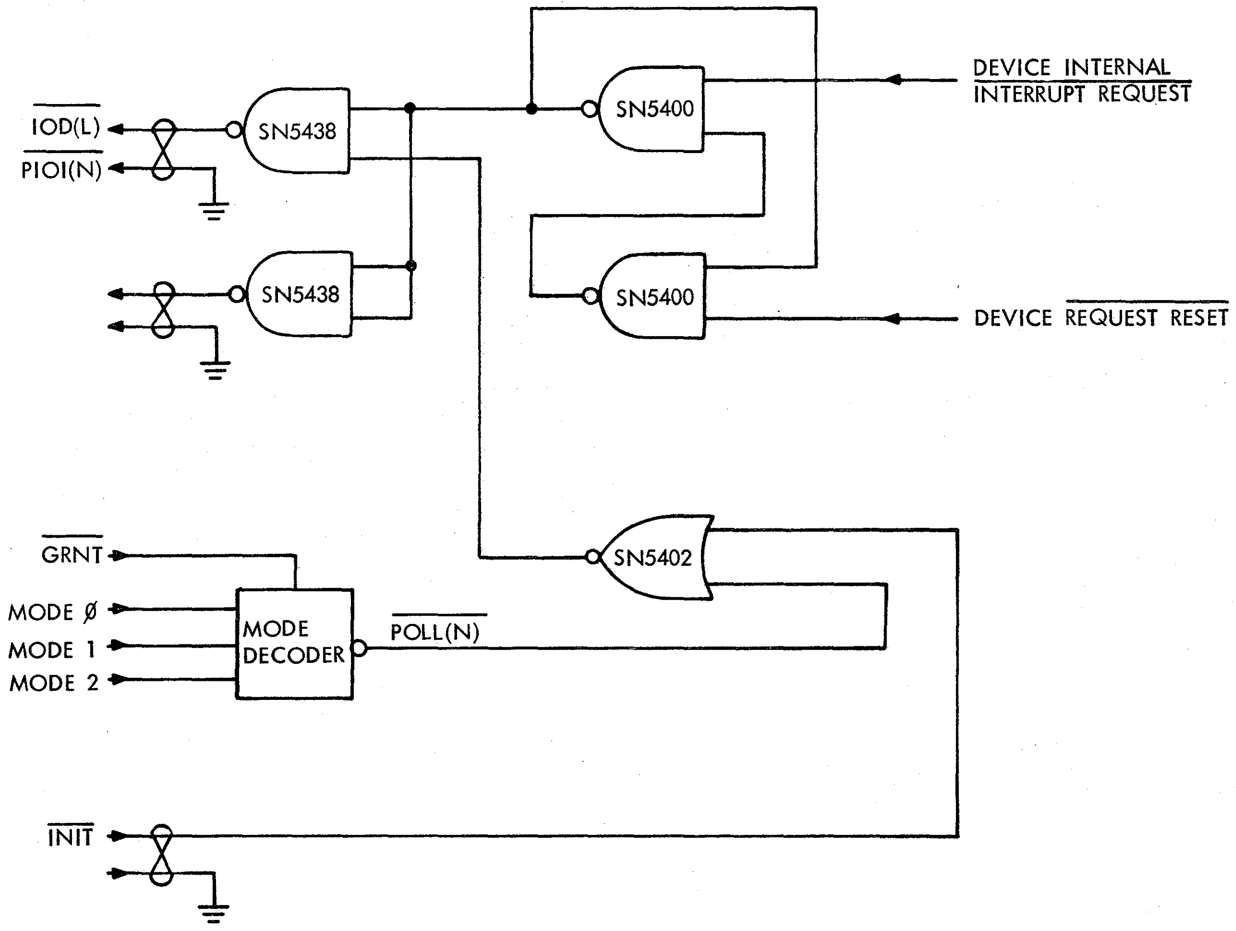


Figure 7-11 High Speed Line Drivers



	R1	R2
STANDARD INTERFACE (ONE SN5438 DRIVER)	220	270
HIGH SPEED INTERFACE (ONE SN54S40 DRIVER)	220	270

Figure 7-12 I/O Bus Terminations



IOD(N) = I/O DATA LINE ASSIGNED TO DEVICE PRIORITY "N"
PIOI(L) = INTERRUPT REQUEST FOR PRIORITY LEVEL "L"

Figure 7-13 Typical Interrupt Connections

7.2 SERIAL I/O BUS

7.2.1 GENERAL SERIAL BUS DESCRIPTION

The Serial I/OBus is designed to permit the processor to communicate with remotely located peripheral devices. Distances of up to 50 feet in extremely high noise environments are possible. High transfer rates are achieved while permitting the use of low power logic devices in the serial interfaces.

A block diagram of the transmission system is shown in figure 7-14. The Serial I/O Bus is composed of 5 signal lines which may be up to 50 feet in length. There are two bi-directional serial data lines designated SIODA and SIODB in the following discussion. Other bus signals are: a uni-directional serial clock line, a uni-directional reset line, and a uni-directional interrupt line. As many as 25 interfaces may be connected to the Serial I/O Bus

The Serial I/O Bus connections are available at the signal connector of the enclosure. All bus signals must be terminated at the extreme ends of the line, and if the serial bus does not leave the enclosure, a termination plug must be in place at the connector.

All Serial I/O Bus signals are differential balanced signals transmitted over twisted/pair lines. Because the bus is fully synchronous, reliable high-speed operation requires a tradeoff of devices attached to the bus to ensure that the timing constraints of Figures 7-16 and 7-16 and Table 7-5 are satisfied.

7.2.2 BUS SIGNAL DESCRIPTIONS

A list of Serial I/O Bus signals is shown in Table 7-4 indicating whether the CPU drives or receives the signal.

Table 7-5 lists the order of bit transfers over the SIODA and SIODB lines while Figure 7-15 identifies the position of these bits in a CPU register.

TABLE 7-4 SERIAL I/O BUS SIGNALS

		CPU	
Signal	Mnemonic	Drives	Receives
Serial Data Line A	SIODA	X	X
Serial Data Line B	SIODB	X	X
Serial Clock	SIOCLK	X	
Serial I/O Reset	SIO RST	X	
Serial I/O Interrupt	SIOI		X

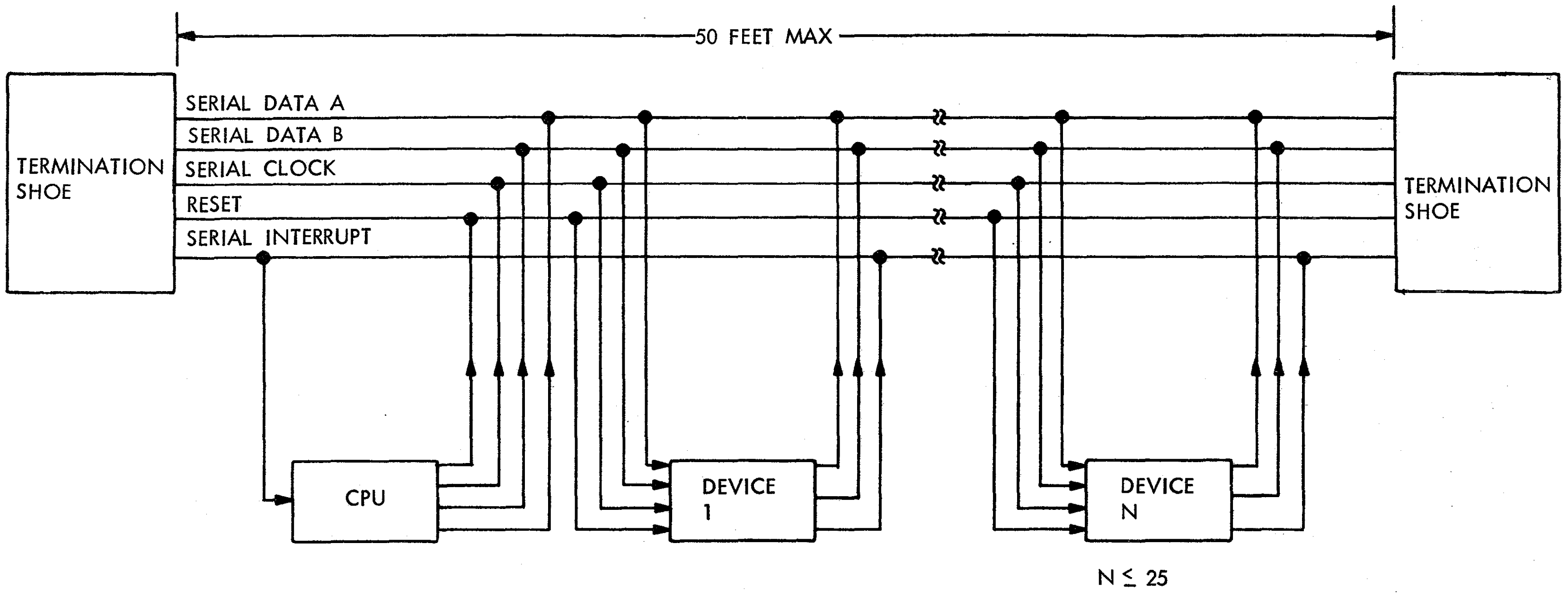


Figure 7-14 Typical Serial I/O Connections.

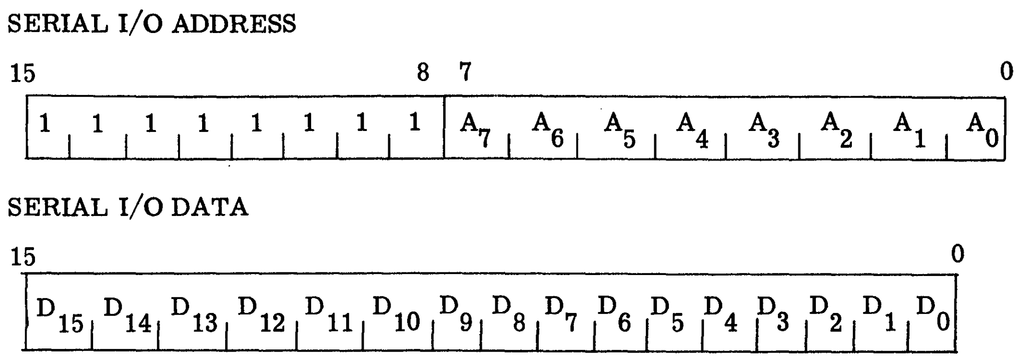

SERIAL I/O ADDRESS
15
8
7
0
1 1 1 1 1 1 1 1
A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0
SERIAL I/O DATA
15
0
D_{15} D_{14} D_{13} D_{12} D_{11} D_{10} D_9 D_8 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

Figure 7-15 Serial I/O Data and Address Bits

TABLE 7-5 BIT TRANSFER FORMATS

	DATA OR ADDRESS BIT ON SIODA/SIODB									
Order of Bit Pair Transfer	STORE BYTE		STORE WORD		LOAD BYTE		LOAD WORD		POLL	
	A	B	A	B	A	B	A	B	A	B
1	0	1	0	0	1	1	1	0	1	0
2	A7	A3	A7	A3	A7	A3	A7	A3	1	A3
3	A6	A2	A6	A2	A6	A2	A6	A2	1	A2
4	A5	A1	A5	A1	A5	A1	A5	A1	1	A1
5	A4	A0	A4	A0	A4	A0	A4	A0	1	A0
6	D7	D3	D7	D15	D7	D6	D15	D14	POLL 15	POLL 14
7	D6	D2	D6	D14	D5	D4	D13	D12	POLL 13	POLL 12
8	D5	D1	D5	D13	D3	D2	D11	D10	POLL 11	POLL 10
9	D4	D0	D4	D12	D1	D0	D9	D8	POLL 9	POLL 8
10	-	-	D3	D11	-	-	D7	D6	POLL 7	POLL 6
11	-	-	D2	D10	-	-	D5	D4	POLL 5	PTR
12	-	-	D1	D9	-	-	D3	D2	PTP	TTYR
13	-	-	D0	D8	-	-	D1	D0	TTYP	OCP

7.2.3 DESCRIPTION OF BUS OPERATIONS

The two basic types of data transfers which may occur on the serial I/O bus are STORE and LOAD. These operations may be either full word or byte. The first bit-pair transmitted by the CPU on the two serial data lines (SIODA and SIODB) designates the operation:

A	B	Operation	Transfer Time MIP-116
0	0	STORE WORD	2.6 μ s
0	1	STORE BYTE	1.8 μ s
1	0	LOAD WORD	4.4 μ s
1	1	LOAD BYTE	2.8 μ s

Figures 7-16 and 7-17 show timing waveforms for load and store operations on the serial I/O bus. Table 7-6 lists the timing specifications for these operations.

All timing specifications are listed as measured at the CPU bus connections.

TABLE 7-6 MIP-116 SERIAL I/O BUS TIMING

Interval		Time (Nanoseconds)		
		MIN	TYP	MAX
T_{AS}	Data - Clock Skew	-15	0	15
T_{AH}	Data Out Hold Time	10	25	-
T_{CW}	Clock Pulse Width	-	100	-
T_{AP}	Data Out Clock Period	-	200	-
T_{DS}	Data Input Setup Time	50	-	-
T_{DH}	Data Input Hold Time	25	-	-
T_{DP}	Data Input Clock Period	-	-	-

7.2.3.1 Data Stores

Figure 7-16 shows typical timing for a data store operation. The first bit pair which appears on the SIODA and SIODB lines specifies the mode of operation of the bus, as shown in Table 7-5 . The next four bit pairs comprise an 8-bit serial I/O device address. All devices receive these first five bit pairs, clocking the bits into an address buffer using the trailing edge of the SIOCLK signal. An unselected device will then count the data transfer clock, return to an idle state and wait for the next address bits to be transmitted. The four bit-pairs following the address (eight bit pairs in the case of full word transfers) are loaded into the selected device data buffer using the trailing edge of the SIOCLK signal.

7.2.3.2 Data Fetches

Figure 7-17 shows typical timing for a data fetch operation. The mode code and address bits are transmitted exactly as for data store operations. Unlike data stores, the CPU transmits no data on the clock pulse following address transmission, but prepares to receive data. Upon receiving this sixth clock pulse, the selected device puts data on the bus and on the trailing edge of the seventh transmitted clock pulse, the CPU accepts the data present on the bus. The selected device also changes the data on the bus on the trailing edge of the seventh pulse. Three additional clock pulses are transmitted by the processor (seven additional bits for full word transfers), the last bit resetting the selected device.

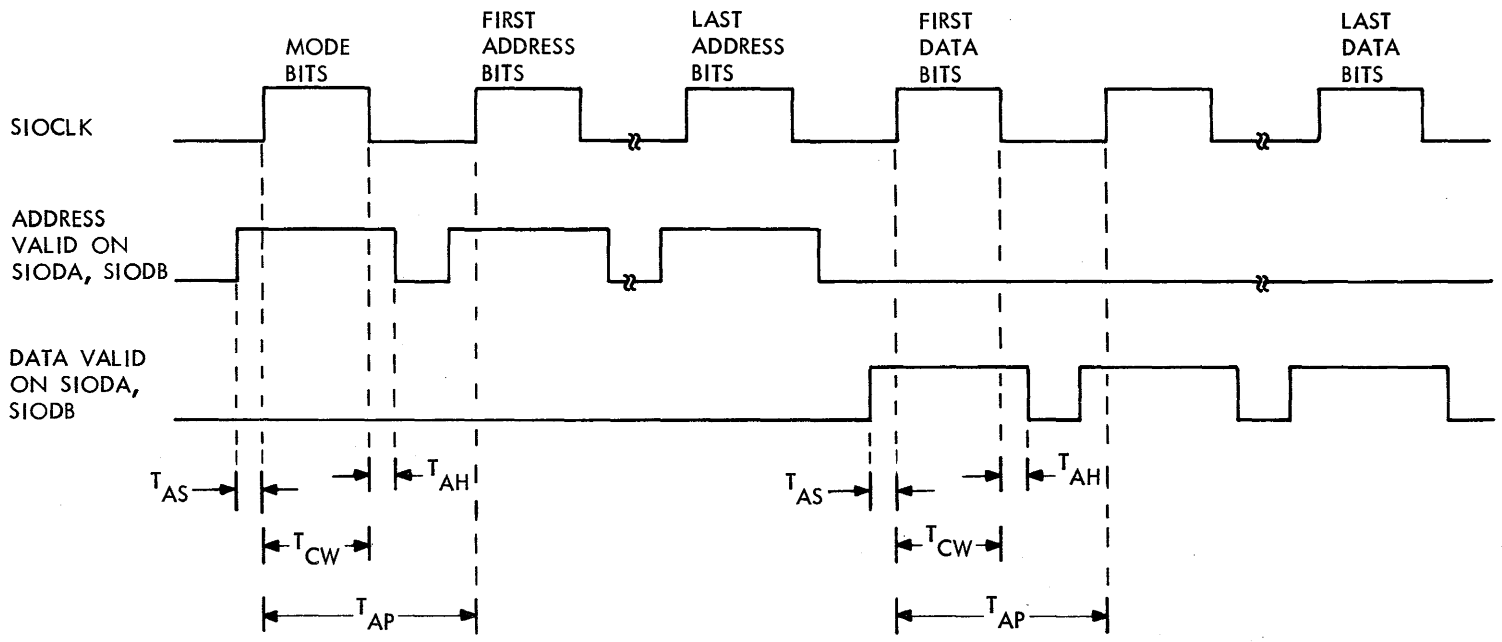


Figure 7-16 Serial I/O Store Timing.

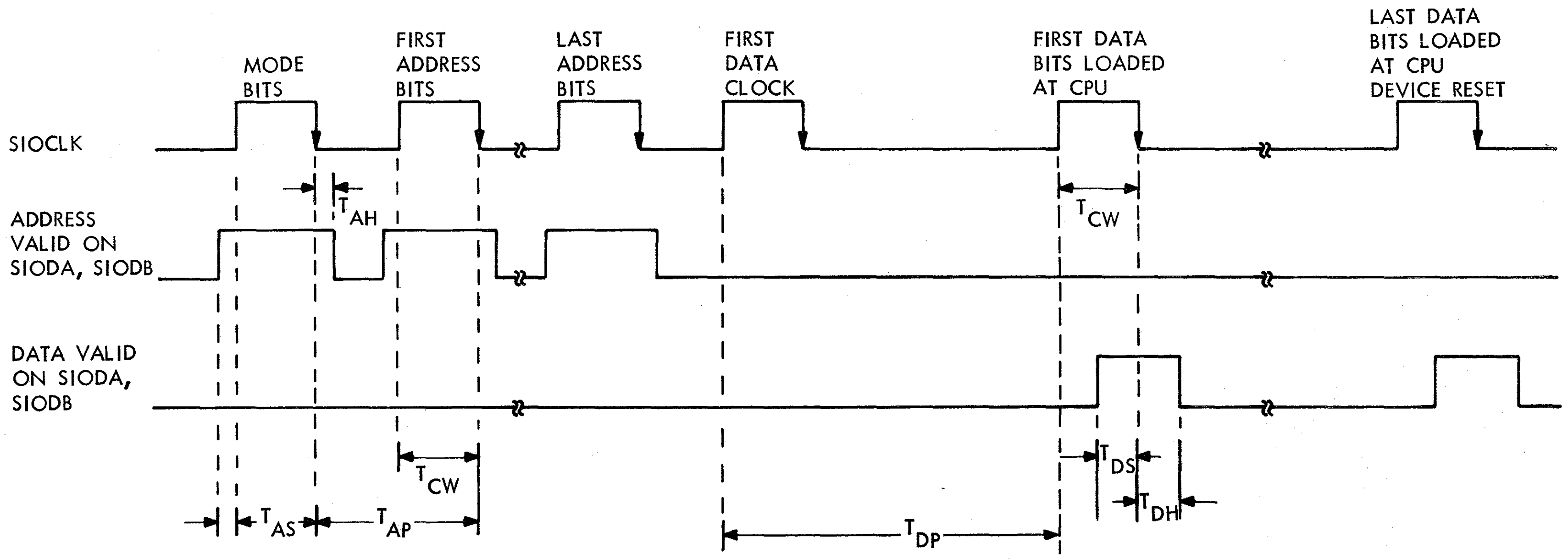


Figure 7-17 Serial I/O Load Timing.

7.2.3.3 Serial I/O Poll

The interrupt poll operation is similar to a LoadWord operation. A special address signifies that all devices with interrupts pending will respond. Each device requesting an interrupt will transmit a data bit corresponding to one bit of the 16-bit data word. Upon receipt of the first (highest priority data bit) the CPU will transfer control to the appropriate interrupting subroutine.

7.2.3.4 Reset

All devices on the Serial I/O Bus are reset by activation of the SIORST line. This signal is activated by the power up routine, by the RESET instruction or by resetting from the OCP.

7.2.4 TYPICAL INTERFACE CONNECTIONS

Figures 7-18 and 7-19 show typical interfacing circuits for use with the serial I/O bus.

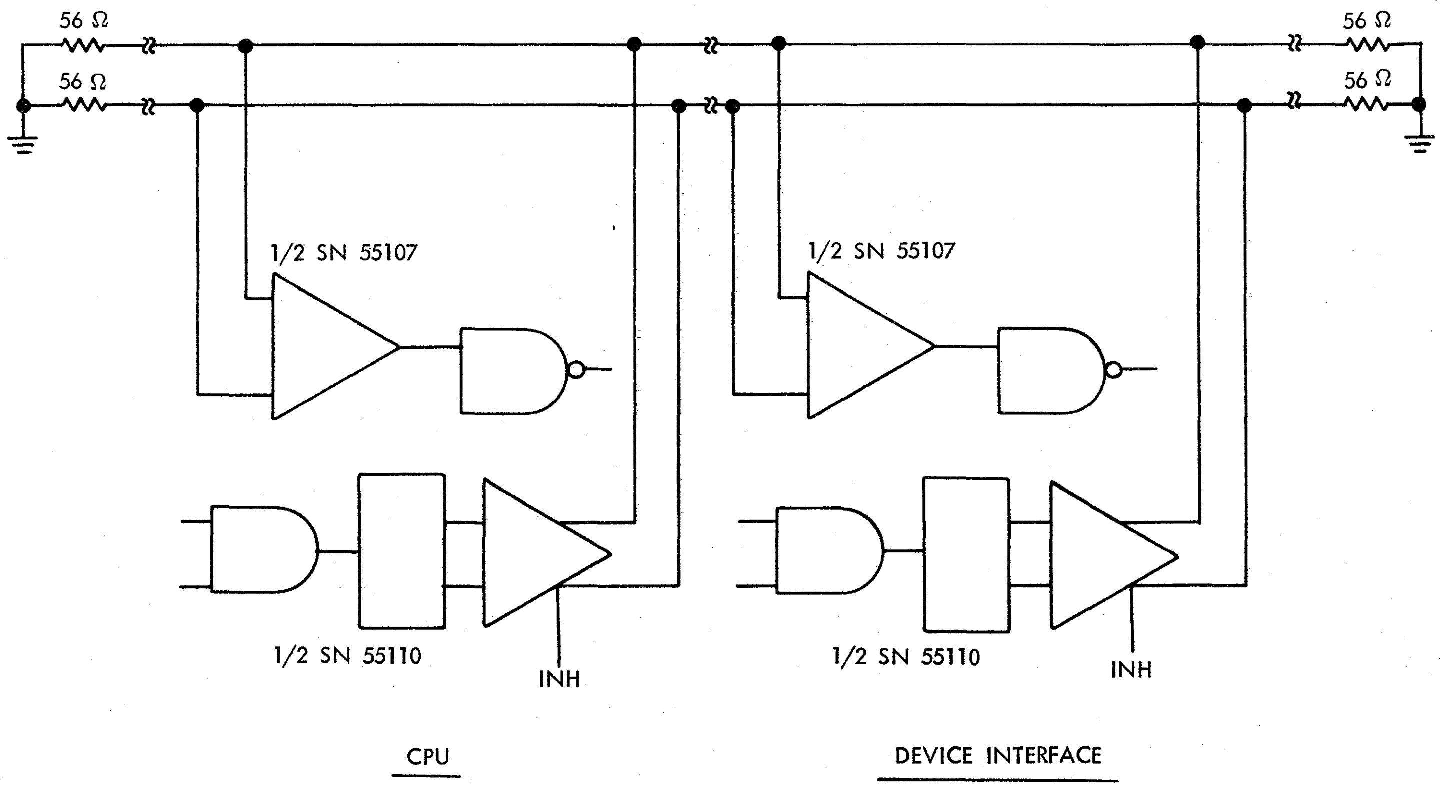


Figure 7-18 Typical Serial I/O Bus Drivers/Receivers/Terminations.

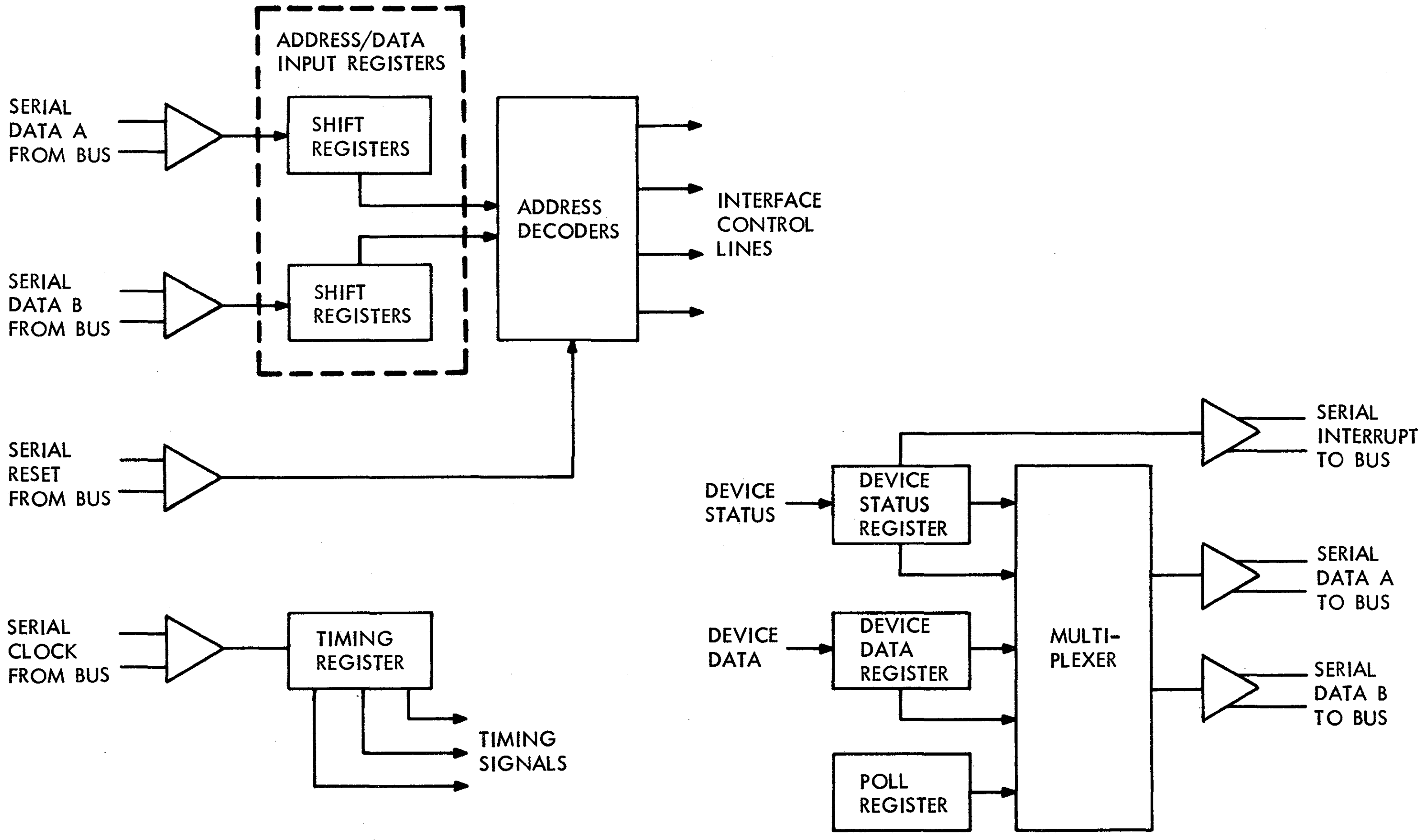


Figure 7-19 Typical Serial I/O Interface.

SECTION 8
SOFTWARE SYSTEMS

8.0 INTRODUCTION

Support software is provided for users of MIP-16 systems on two levels. The Basic Software System (BSS) is for those users primarily interested in development of dedicated application programs. The Basic Operating System (BOS) is provided for those users who require a more general capability. BSS requires a very minimal system while BOS requires a disk and 8K words of storage. These systems are described in detail in their reference manuals.

8.1 BASIC SOFTWARE SYSTEM

The Basic Software System consists of four elements:

a. Time shared Assembly Program (TSAP)
b. Diagnostic Exerciser
c. Bootstrap Loader
d. Debug Program

TSAP operates on a PDP-10 and permits the user to generate absolute programs in time sharing mode using the full facilities of the PDP-10. For use, the programs are loaded via the bootstrap loader.

The exerciser program tests each operational instruction of the processor, halting at a specified location if an error is encountered.

The bootstrap is used to enter programs or data into the processor's memory. It will accept binary objective output from TSAP on either the teletype or high speed reader.

The debug program will store into or type out an address or general purpose register. It will also execute between two specified addresses of the user's program. Control is always returned to the debug program after each function is executed.

8.2 BASIC OPERATING SYSTEM

The Basic Operating System provides facilities for the user to create programs using a

MIP-16 system alone. The following is included in the BOS:

a. Supervisor
b. Assembler
c. Editor
d. Linking Loader
e. Debugging Editor
f. Subroutine Library

The Supervisor provides facilities for job, task, and data management. The user may initiate execution of programs stored on disk or he can generate programs using the editor, assembler, and loader and store them on disk. Data management provides access methods and peripheral handlers for access of data through peripherals. The subroutine library provides reentrant and relocatable common floating and fixed point routines useful for writing user programs. Included are:

a. Input/Output Format Conversion
b. SIN/COS
c. ATAN
d. LOG
e. EXP
f. SQRT