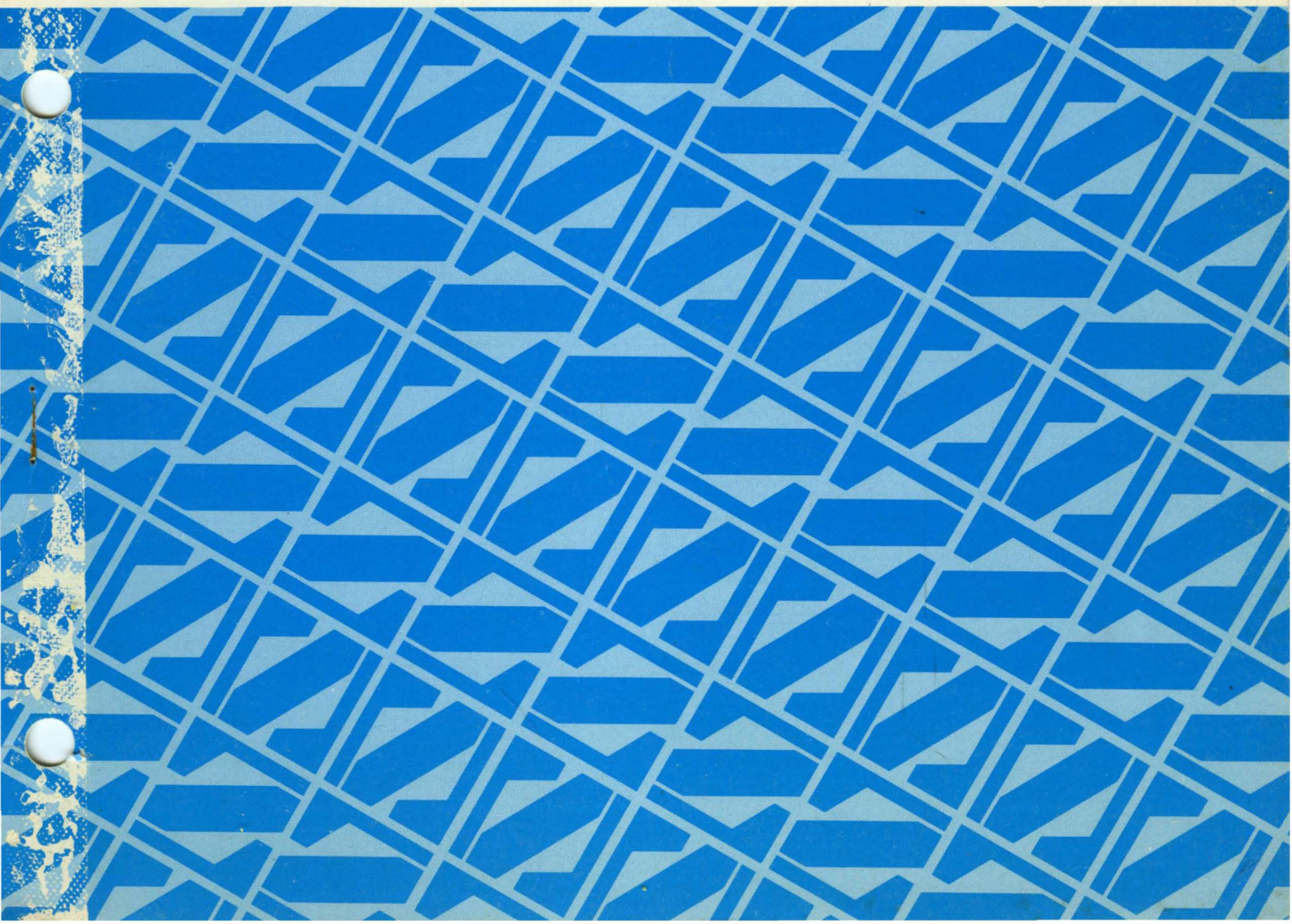
National Semiconductor
Order No. IMP-16L/926X
Pub. No. 4200028X
PRELIMINARY

IMP-16L Users Manual



Order Number IMP-16L/928X
Publication Number 4200028X

IMP-16L MICROCOMPUTER

IMP-16L
USERS MANUAL
(PRELIMINARY)

April 1974

PREFACE

This manual provides information to permit the user to efficiently operate the IMP-16L microcomputer. The manual contains general information including all block diagram functional theory of operation, description, instruction set, control panel operation, firmware description, I/O operation, system data bus operation, and system verification. For purposes of system development and maintenance, the user should have a good working knowledge of software programming, circuit logic, and integrated circuits.

Copies of this publication and other National Semiconductor publications may be obtained from the sales offices listed on the back cover. The following publications are also relevant:

- IMP-16 Utilities Reference Manual (NSC Order No. IMP-16S/025Y)
- IMP-16L Product Description (NSC Order No. IMP-16L/924)
- IMP-16 Programming and Assembler Manual (NSC Order No. IMP-16S/102Y)
- Timeshare Users Manual (NSC Order No. IMP-16S/118Y)
- General Purpose Controllers/Processor MOS/LSI System Kit, Product Description (NSC Order No. IMP-00A/905B)
- CROM Data Sheet (NSC Order No. IMP-16A/521D
- RALU Data Sheet (NSC Order No. IMP-00A/520D

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

TABLES

Figure 1-1. IMP-16L Microcomputer

Chapter 1

GENERAL INFORMATION

1.1 GENERAL DESCRIPTION

The IMP-16L, shown in figure 1-1, is a general purpose 16-bit microcomputer designed expressly for the OEM (original equipment manufacturer) user. A high-speed asynchronous bus affords direct memory access by peripheral devices and, along with the modular configuration of the equipment, gives the user maximum flexibility for configuration of his systems. The basic IMP-16L and its options provide an unusually versatile capability for developing a variety of OEM equipment, software, and full-scale processing systems. Both the memory and the microprocessor include LSI circuits that are advanced products of proven semiconductor technology, thus providing lower cost, higher reliability, smaller size, and lower power consumption.

The system manufacturer may initially procure the IMP-16L as a stand-alone system in order to expedite software development and to permit rapid assembly of models for field trial. Later, it may be more economical to purchase printed circuit cards that are packaged in an enclosure customized for the application. When production quantities warrant, the system manufacturer may later decide to purchase at the semiconductor component level. Thus, the manufacturer may optimize both his development and production phases due to the versatility of the IMP-16L design and options.

A Programmers Control Panel is supplied with the IMP-16L to provide access to the Central Processing Unit (CPU) registers and memory. Using the Programmers Control Panel, the operator may address, load, and examine memory and CPU registers and control the operation of the microcomputer to debug software and I/O hardware.

The functional configuration of the major units comprising the basic IMP-16L is shown in the block diagram of figure 1-2. The CPU, Memory, and Peripheral Devices communicate and exchange data with one another by way of a bidirectional 16-bit asynchronous data bus. Both programmed data transfers and direct memory access (DMA) transfers take place between memory and peripheral devices on this data bus independent of CPU operation. The maximum bus transfer rate is one million 16-bit words per second (approximately).

1.2 OPERATIONAL FEATURES

IMP-16L operational features are listed in table 1-1.

Table 1-1. IMP-16L Operational Features

Item	Parameter
Word Length	16 bits
Instructions	60 general purpose instructions
User Register	4 multipurpose
Stack	16-word last-in/first-out
Memory	4096 16-bit words of semiconductor memory, expandable in increments of 4096 words

Table 1-1. IMP-16L Operational Features (Continued)

Item	Parameter
Addressing Modes	Direct - 256 words Relative to Program Counter - 256 words Relative to Accumulator 2 or 3 - 256 words* Indirect - 65,536 words
Arithmetic	Parallel, binary, fixed point, twos-complement, single- and double-precision
Speed	Register-to-register addition — 4.9 usec Memory-to-register addition — 8.4 usec Register input/output ——— 10.5 usec DMA (direct memory access) Transfer ——— 1.05 usec
Input/Output	Bidirectional asynchronous 16-bit data bus Up to one million words-per-second DMA transfer rate 4 interrupt levels - one level provides device vectored routine entry. Addressing for 256 peripheral devices
Control Panels:	
Operators Control Panel	Contains controls for program loading, RUN, HALT, INITIALIZE, peripheral controls (AUX1 and AUX2), and power and panel key switches
Programmers Control Panel	Display and data entry switches for registers, push-down stack, and memory Single instruction, increment memory address, and load data switches
Input Power	120 VAC $\pm$ 10%, 47-63 Hz or 230 VAC $\pm$ 10%, 47-63 Hz
Temperature:	
Operating Storage	0^{o} to 50^{o}C -20^{o} to 55^{o}C
Humidity	To 90% without condensation
Dimensions (Overall)	10.5 inches high, 17 inches wide, 24 inches long

* Indexing gives maximum range of 65,536 words in page sizes of 256 words each.

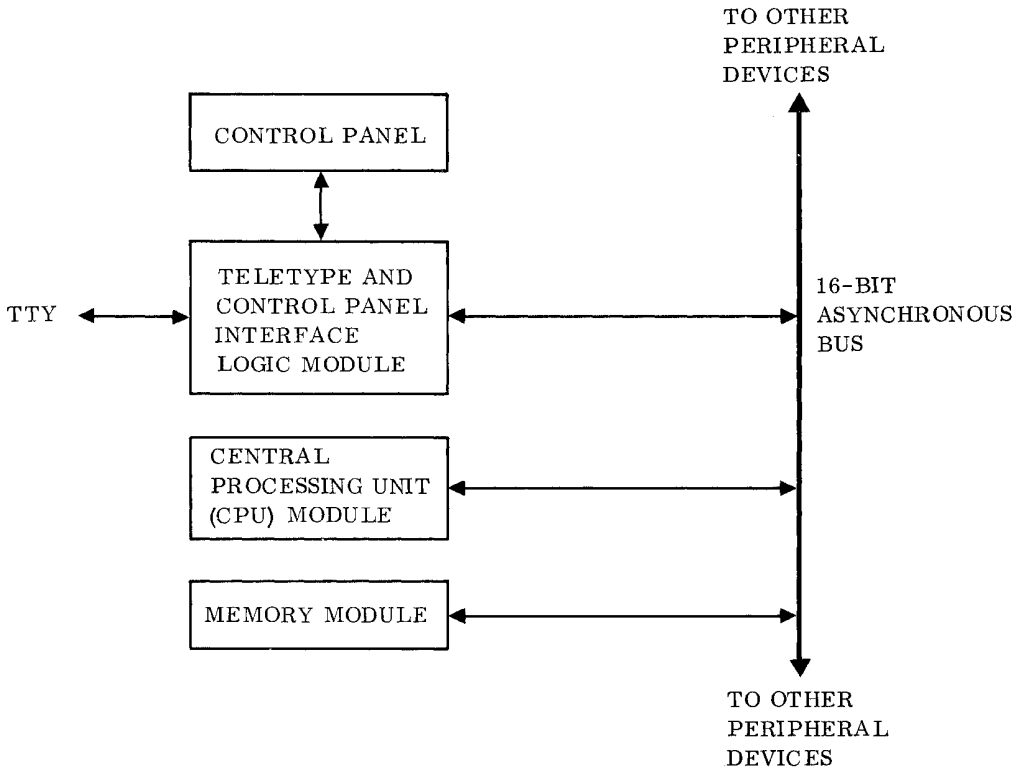


Figure 1-2. IMP-16L Simplified Block Diagram

1.3 PHYSICAL DESCRIPTION

The physical layout of the IMP-16L is shown in figure 1-3 and described in the following paragraphs. Table 1-2 lists the basic and optional items and modules included in the IMP-16L microcomputer.

1.3.1 Card Modules

Printed circuit card modules are 8-1/2 by 11 inches. Each card is associated with a specific function and is available separately. The following card modules are available:

- CPU — This module consists of one card which contains the central processor, bus controller logic, and bus interface logic circuits.
- Memory — This module consists of one card which contains 4096 words (16-bits) of read/write memory (RAM), sockets for 512 words of read-only memory (ROM or PROM), bus interface circuits, and memory control circuits.
- Card Reader Controller (Optional) — This module contains two printed circuit cards which contain the DMA controller circuits for the Card Reader.
- Control Panel and TTY Interface — This module consists of one card which contains the interface hardware to implement all panel functions and to provide the serial TTY interface.

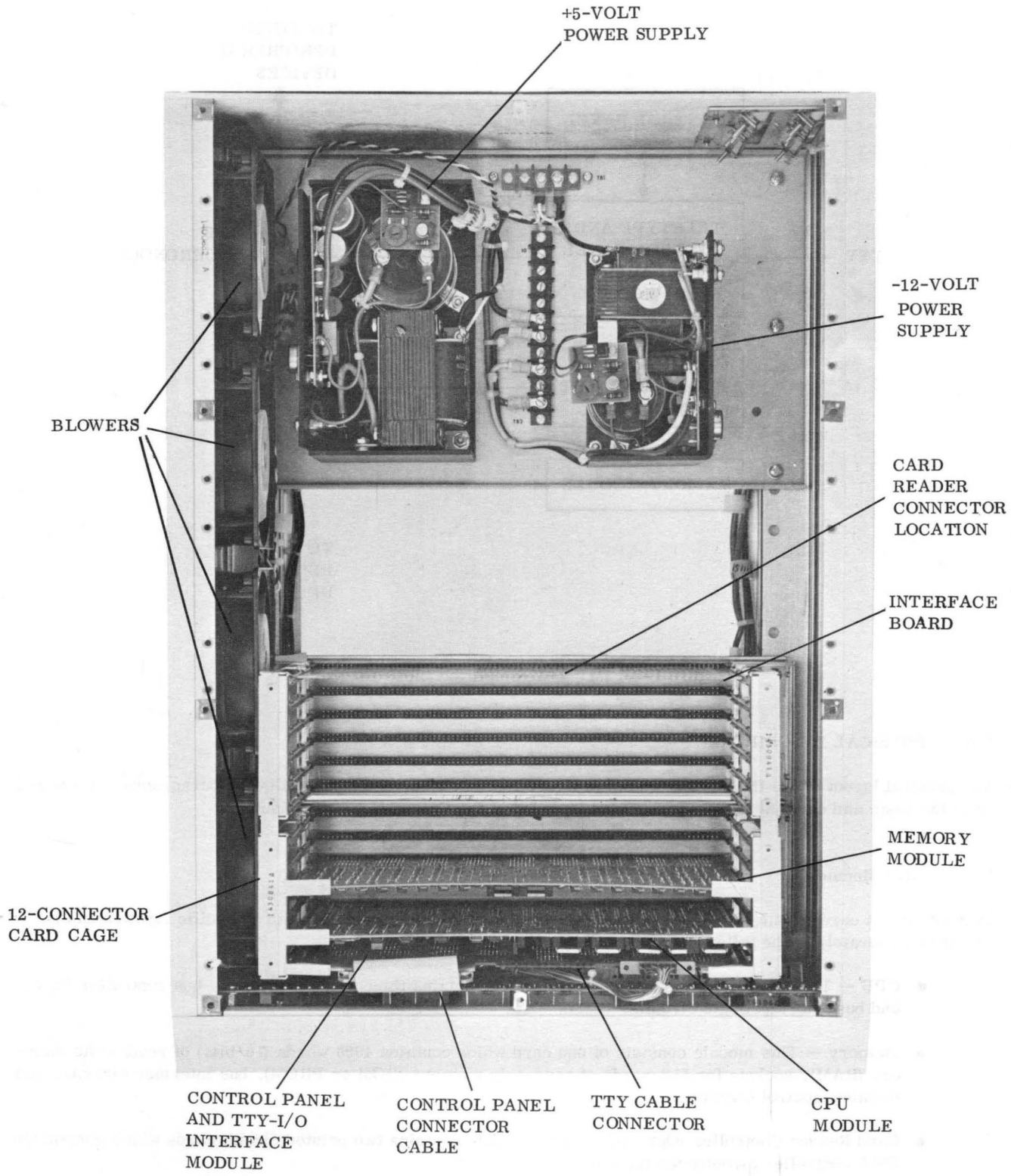


Figure 1-3. Top Internal View of IMP-16L Microcomputer

Table 1-2. Major IMP-16L Units and Options

Item	Unit, Contents, Options	Order Number
1	IMP-16L microcomputer system with DMA bus Hardware: • Chassis (wired for the following: CPU, control panel/TTY logic, 8K memory, card reader controller, and PROM programmer card. • Power supply assembly • Teletype/control panel interface logic card • Teletype I/O cable • 12-connector card cage • Operators Control Panel • Programmers Control Panel • CPU card (includes two CROMs for 60 general-purpose instructions) • 4K memory card (includes two 256-by-8 PROM/ROMs programmed for paper tape loader, control panel service, and TTY control; and two sockets for additional 256 words of PROM/ROM) • Interface board Reference Manuals: • IMP-16L Users Manual • Timeshare Users Manual • IMP-16 Utilities Reference Manual • IMP-16 Programming and Assembler Manual • Engineering Documentation Package Software Paper Tapes and Listings (16LBASSOFT): • IMP-16 Resident Assembler (IMASM and IMPASP) • IMP-16L Source Editor (EDIT 16) • IMP-16L CPU Diagnostic Routine(CPUXDI) • IMP-16L Memory Diagnostic Routine (MEMDIL) • IMP-16L Software Debug Routine (DEBUG - listing not supplied)	IMP-16L/304 or IMP-16L/308 (with second 4K RAM card) NOTE For 230-VAC (50- or 60-Hz) operation, add "E" suffix to above order numbers.

Table 1-2. Major IMP-16L Units and Options (Continued)

Item	Unit, Contents, Options	Order Number
1 (cont'd)	• IMP-16L Relocating Loader Routine (GENLDR) • Teletype Software Package (STTYIO) • IMP-16L Basic Firmware Listing (BFIRL)	
	Options	
2	Card Reader (Documation Model M300L – 300 cards per minute)	IMP-00/825 (For 230-VAC, 50-Hz operation add "E" suffix to above order number.)
3	Card Reader Interface Package Hardware: • Card Reader Controller - two PC boards • One I/O cable assembly Software (16LCRSOFT): • IMP-16 Resident Assembler - requires 8K memory (IMPASM 8K and IMPASP 8K use absolute loader) • IMP-16L CPU Diagnostic Routine (CPUXDI - uses absolute loader) • IMP-16L Memory Diagnostic Routine (MEMDIL - uses absolute loader) • IMP-16L Software Debug Routine (DEBUG - relocatable loader format) • Card Reader Bootstrap Loader Routine (CRBOOT - binary loader format) • Card Reader Absolute Loader (ABSCR - card reader bootstrap format) • IMP-16L Relocating Loader Format (GENLDR) • IMP-16L Teletypewriter Software Package (STTYIO - relocatable loader format)	IMP-16L/825W

Table 1-2. Major IMP-16L Units and Options (Continued)

Item	Unit, Contents, Options	Order Number
4	IMP-16 Assembler (cross assembler written in FORTRAN IV) • Source deck • Listing • IMP-16 Programming and Assembler Manual • Program Description Manual • Installation of IMP-16L Assembler on System 360/370 Operators Manual • Card decks with listings for card punch routines for CRBOOT format and RLM format	IMP-16S/900A
5	Teletypewriter (ASR Model 33 wired with cable for IMP-16L) and paper tape reader control option	IMP-00/810 (For 230-VAC, 50-Hz operation, add suffix "E".)
6	4K RAM card	IMP-16L/004
7	64-socket blank circuit card for OEM prototyping	IMP-00H/891
8	90-socket blank circuit card for OEM prototyping	IMP-00H/892
9	Card extender	IMP-00H/890
10	Paper tape reader relay control kit for ASR Model 33 Teletype	IMP-00/810R
11	Single-card connector - 144 pins	IMP-00H/882
12	3-card connector panel: Card frame with three 144-pin wire-wrap connectors and card guides, spaced to accomodate wire-wrap Prototyping Cards	IMP-00H/881
13	6-card connector panel: Card frame as above but with six 144-pin wire-wrap connectors and card guides, spaced to accommodate PC cards	IMP-00H/880

Table 1-2. Major IMP-16L Units and Options (Continued)

Item	Unit, Contents, Options	Order Number
14	PROM Programmer Package: For programming MM5203 and MM5204 PROMs. Packaged on one printed circuit card which plugs into IMP-16L backplane - operates under program control of microcomputer. Complete with software package (paper tape) and operating manual.	IMP-16L/805

1.3.2 Card Cage

The card cage contains 12 connectors to accomodate the card modules in the IMP-16L. One additional card cage, to accomodate six additional printed circuit cards, is optional. If still more expansion is required, the power supplies may be removed and remotely installed to allow additional room for two more six-connector card cages. With the full complement, up to 30 cards may be installed in a system.

1.3.3 Chassis

The chassis may be a stand-alone unit or may be adapted for rack mounting. Convenient means of mounting a control panel, card cages, and power supplies are provided by the chassis. Figure 1-3 shows a typical IMP-16L microcomputer, with the cover removed to show component and assembly locations.

1.3.4 Control Panels

The standard system is supplied with a Programmers Control Panel and an Operators Control Panel. Both control panels are shown in figure 1-1. The Programmers Control Panel provides access to registers and memory for use in program development. The Operators Control Panel has a minimum number of controls. Use of the control panels is discussed in chapter 4.

1.3.5 Power Supply

The output voltages of the standard power supply are as follows:

- +5 volts at 12 amperes
- -12 volts at 3 amperes

For larger systems, there is a higher current power supply available.

1.4 IMP-16L HARDWARE/SOFTWARE ITEMS

Various items of hardware and software support are listed in table 1-2. Among these is the resident assembler program (supplied with the IMP-16L) which can be loaded into the IMP-16L memory and then used to assemble the user's application programs.

A number of IMP-16L loader programs and an IMP-16L debug program are also available to support development of user's application programs.

In addition to the standard items supplied, a cross-assembler program is available to support the preparation of user application programs. The cross assembler, written in ANSI FORTRAN IV, may be used to assemble programs written in the IMP-16L language. It is for use on a large, general-purpose computer. The cross-assembler has also been installed on the worldwide timesharing computer utilities.

Chapter 2

CPU AND MEMORY FUNCTIONAL DESCRIPTION

2.1 GENERAL

This chapter provides a functional description of the IMP-16L CPU and Memory Modules on a block diagram level. Block diagrams are used throughout the text to describe individual units; an overall functional block diagram, representing the circuits contained in the schematic diagrams (supplied under separate cover with the IMP-16L microcomputer), is included at the end of the chapter. Figure 2-1 is a block diagram showing the major functional groups in the system which may be configured in a variety of ways to suit the user. The CPU Module and Memory Module are basic to any given configuration.

The CPU Module contains circuits which control the functions of the arithmetic section, internal and external bus timing and control, and interfacing with peripheral devices and memory.

The Memory Module contains the 4096-word read/write memory; the 256-word Read-only Memory (ROM) in which the various firmware service routines are stored; and input/output buffers.

2.2 CPU MODULE

The CPU Module is configured around four 4-bit Register and Arithmetic Logic Units (RALUs), a Microprogrammed Control Section, and other logic for control and gating functions. In order to appreciate the operation of the CPU Module, an understanding of the RALUs and Microprogrammed Control Section is helpful. These units are described in the following paragraphs.

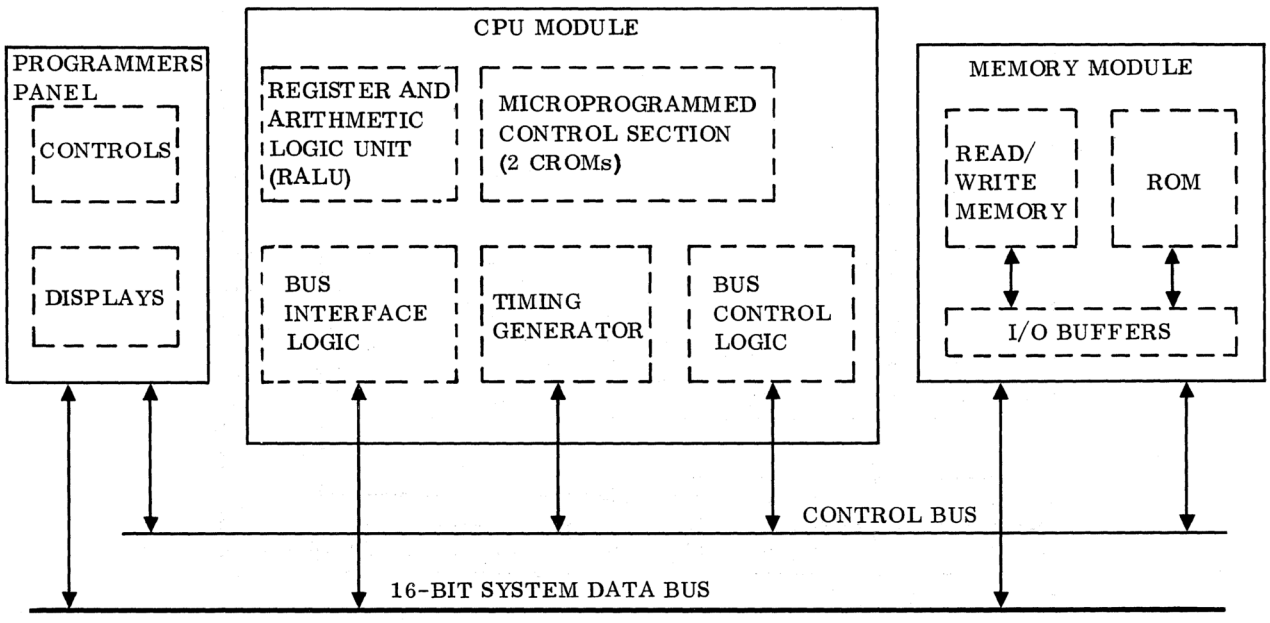


Figure 2-1. IMP-16L Functional Block Diagram

2.2.1 Register and Arithmetic Logic Unit (RALU)

The units comprising the arithmetic section are composed of four 4-bit RALUs which are logically implemented in parallel to form a 16-bit unit. A RALU block diagram is shown in figure 2-2.

The input/output multiplexer transfers data, address, and instruction words into and out of the RALU registers. The multiplexer also interfaces with the CPU data bus.

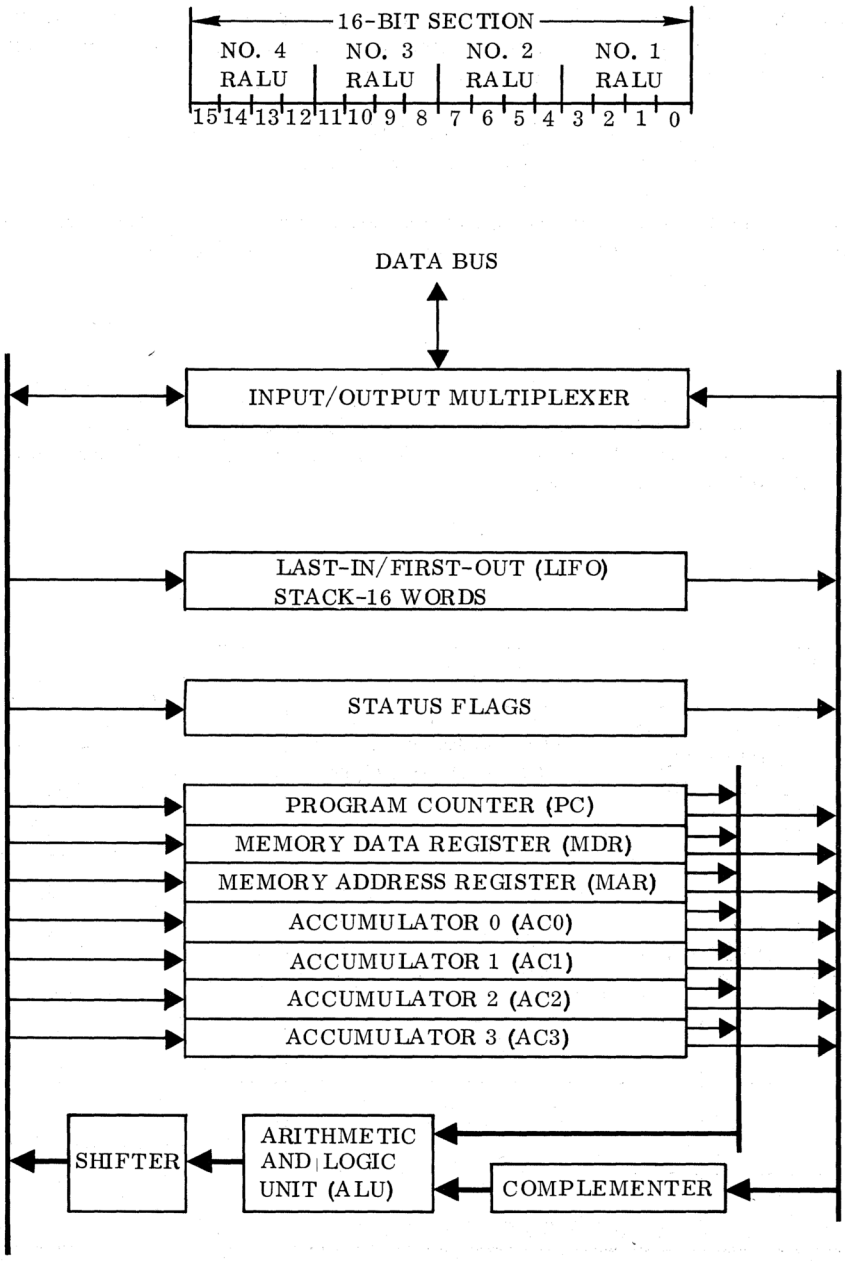


Figure 2-2. RALU Block Diagram

There are sixteen 16-bit words in the Last-in/First-out (LIFO) Stack. The contents of the accumulators, status flags, or Program Counter may be pushed onto or pulled from the top 16-bit location of the stack. As subsequent words are pushed onto the stack, the contents already in the top location are moved downward to the second highest location. The contents of the second highest are moved to the third highest, and so forth. If more than 16 words are pushed onto the stack, the contents of the bottom of the stack are lost.

Words are retrieved from the stack in reverse order of the loading sequence. As the top word is pulled, all words in lower locations are moved upward toward the top location. A 16-bit word consisting of all zeros is loaded into the bottom of the stack each time a word is pulled from the top. A stack-full interrupt is generated whenever the bottom stack word contains nonzero data.

The 16 status flags are implemented as a register. These flags are used for indicating the various levels of interrupts, overflow and carry arithmetic functions, and link operations associated with double-word usage. The16 status flags may be pushed onto or pulled from the LIFO Stack, set or cleared, and individually tested. A complete description of the use of the status flags is given in chapter 3.

The Program Counter (PC) holds the address of the next instruction to be executed. It is incremented by 1 immediately following the fetching of each instruction during execution of the current instruction. In the case of a double-word instruction, the Program Counter is incremented by two. At initialization or power on, the Program Counter is set to X'FFFE.

The Memory Data Register (MDR) holds data transferred from the memory to the CPU or vice versa. When fetching data, the address is placed in the Memory Address Register (MAR). The fetch instruction causes the data word to be transferred from the designated memory location to the Memory Data Register. Conversely, when storing data in the memory, the data word is placed in the Memory Data Register; the address is placed in the Memory Address Register; and the store instruction causes the data word to be transferred to the designated memory location. The Memory Address Register (MAR) holds the address used when data is to be fetched.

There are four 16-bit accumulators: AC0, AC1, AC2, and AC3. In general, the accumulators contain operands for arithmetic and logical operations. Also, the result of arithmetic operations is usually stored temporarily in one of the four accumulators. Data words may be fetched from memory to an accumulator, or stored from an accumulator into memory. The particular accumulator to take part in an operation must be specified by the programmer in the appropriate instruction. AC0 may be used to transfer data to and from peripheral devices.

The Arithmetic and Logic Unit (ALU) performs both arithmetic and logical operations: binary addition, AND, OR, and EXCLUSIVE OR. The Shifter permits left or right shifting of the contents of the Arithmetic and Logic Unit, and the Complementer is used to complement the contents of the A-bus.

2.2.2 Microprogrammed Control Section

Two Control and Read-only Memories (CROMs) comprise the Microprogrammed Control Section of the IMP-16L. Figure 2-3 is a block diagram of a CROM. Operation is as follows. The CROM contains a microprogram that implements the instruction set. This microprogram resides in a 100-by-23-bit Read-only Memory (ROM) within the CROM. During an instruction fetch, the 9 most significant bits of the instruction word are transferred to the CROM; these 9 bits comprise the op code and other pertinent control fields of an instruction word. The instruction bits are decoded, and then the ROM Address Control in the CROM directs the control sequence to an entry point in the microprogram. The sequence continues until execution of the fetched instruction is completed. Then the CROM goes through another fetch cycle to fetch the next instruction from memory. This process is continuously repeated.

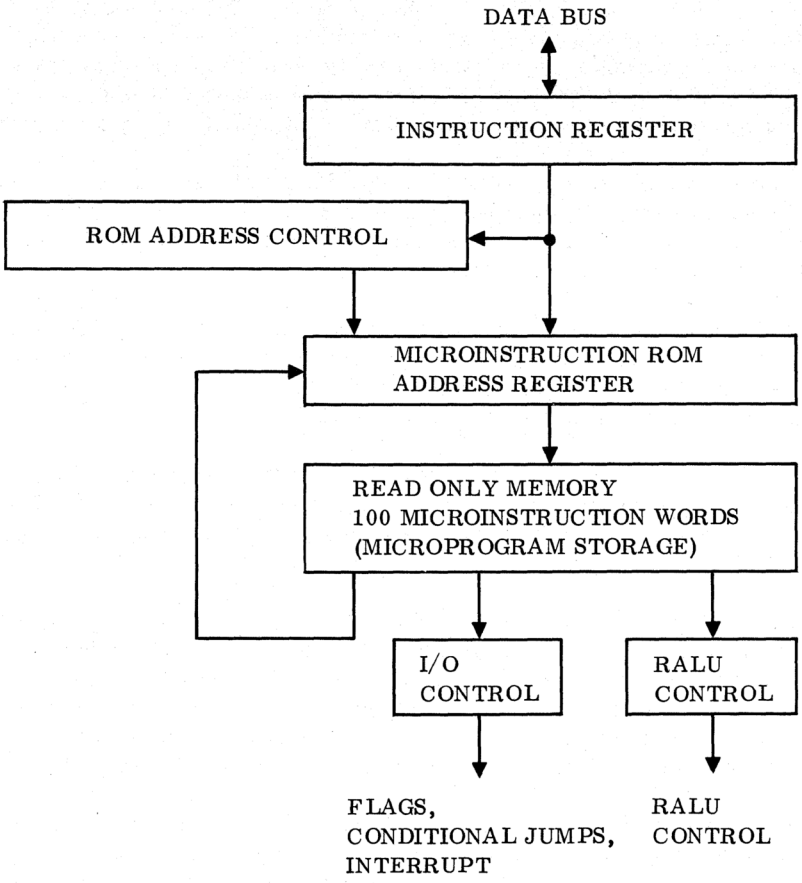


Figure 2-3. Control and Read-Only Memory (CROM) Simplified Block Diagram

2.2.3 Data Flow

Figure 2-4 is a block diagram of the IMP-16L CPU Module. It shows the data flow between the major functional units within the module. The numbers shown in parentheses indicate the number of lines employed to accomplish communication tasks. Operation is as follows. The CPU may establish communication with a peripheral device (or memory) via the System Data Bus after first acquiring priority and access through the System Bus Controller. This controller, described in chapter 7, provides the proper priority and timing for peripheral communication. Incoming data, from peripheral devices (or memory) is routed through the I/O Data Buffers to the Arithmetic Section for manipulation. If the incoming word is an instruction, data bits 7 through 15 of the instruction word are routed to the Microprogrammed Control Section. The control logic within the CROMs directs the arithmetic functions associated with the incoming data. If the incoming word is data, it may be routed to a register under control of the microprogram.

The flag addresses sent out by the CROM are latched in the Jump/Flag Buffer Latch to keep them stable. The latched addresses are then used to select one of 16 jump conditions. The Complemented Flag Enable Signal (NFLEN) enables the selection of a flag. Four user-assigned jump conditions and six user-assigned control flag lines are brought out to pins on the edge connector of the CPU Module. The 8 most significant flags of the 16 control flags may be set using the Set Flag Instruction and cleared or pulsed using the Pulse Flag Instruction; the assignments of these flags are listed with their codes in table 3-15. The 8 least significant flags may be modified by the CROM-resident microprograms.

Control of IMP-16L operations is accomplished by routines that constitute the microprogram stored in the CROMs. The microprogram effects the implementation of the macroinstructions that comprise the IMP-16L instruction set. The actual data processing operations are performed in the RALUs.

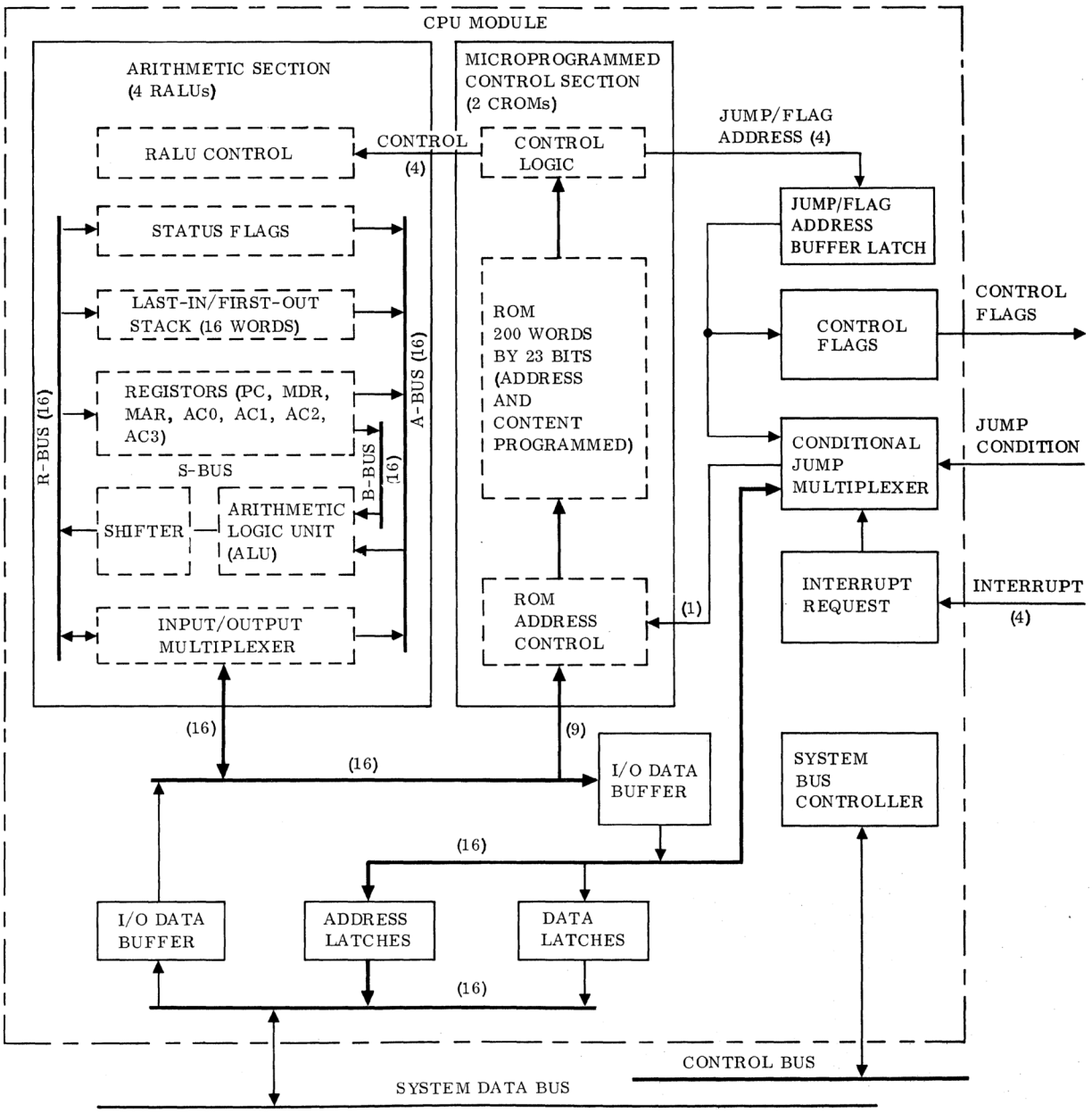


Figure 2-4. IMP-16L CPU Module

2.2.4 System Timing

The basic internal machine cycle of the IMP-16L consists of the execution of a single microprogram step. This cycle time period comprises eight time periods, T1 through T8. As shown in figure 2-5, clock pulses occur on four clock lines at the respective odd-time periods T1, T3, T5, and T7. The Phase Clock Signals (PH1, PH3, PH5, and PH7) provide the timing for the gating of control information from the CROMs to the RALUs via the 4-line time-multiplexed control bus. For a detailed description of the CROM and RALU control signals, refer to the CROM and RALU Data Sheets.

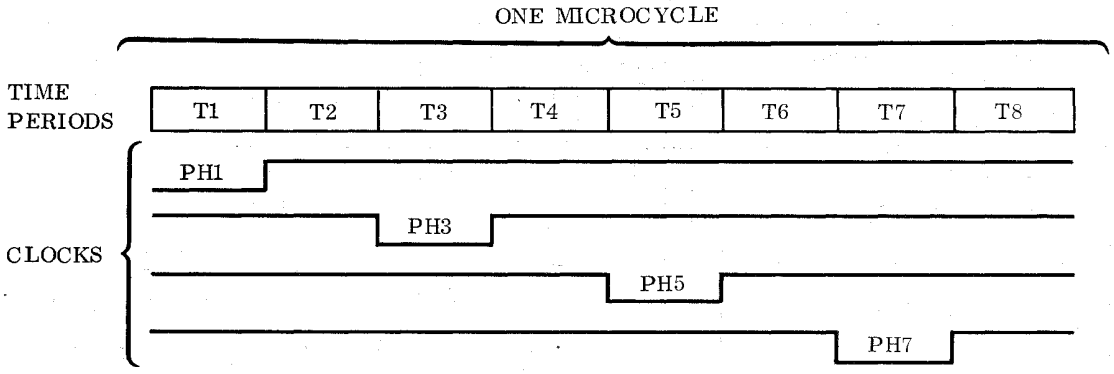


Figure 2-5. IMP-16L Timing and Phase/Clock Relationships

2.2.5 Basic CROM Flowchart

Figure 2-6 is a flowchart of the events that occur during a typical operation of the CROM. General operation is as follows:

- INITIALIZATION

  When power is first applied to the processor or an external System Clear Signal is received, all RALU registers, flags, and the stack are cleared to zero. The microprogram then sets the Program Counter (PC) to $FFFE_{16}$, the start of the initialization sequence.

- INSTRUCTION FETCH

  Following the initialize sequence, control is transferred to the first step in the fetch microroutine. During this first microcycle of the fetch routine, the contents of the Program Counter (PC) are sent out on the CPU data bus and latched in the 16 address latches. The Read Memory Flag is set high. The Read Memory Flag actuates a read operation in the system memory. During T7 of the same microcycle, the 16-bit instruction comes back from memory ready to be decoded.

  In the next microcycle, if there is no interrupt, the Program Counter is incremented by one and points to the next consecutive memory location.

  A decision is made as to whether the instruction just read needs to make reference to memory. If it does not (for register-to-register operations and other nonmemory-reference instructions), then the instruction decode circuit steers control to a microprogram entry point that corresponds to the particular instruction. If the instruction requires a memory reference, then two additional microprogram steps are required to compute an effective address and to bring in the new word: first, the memory address is computed, and, second, the data word is transferred from memory to the register designated by the preceding instruction.

  If an interrupt occurs, a decision is made concerning the Vectored/Level Zero Interrupt status. If there is no such interrupt, the Program Counter value is pushed into the stack and the instruction in memory location one is executed. If there is a Vectored/Level Zero Interrupt, then the peripheral provides the next instruction to be executed. After execution, the Interrupt Enable Flag is cleared.

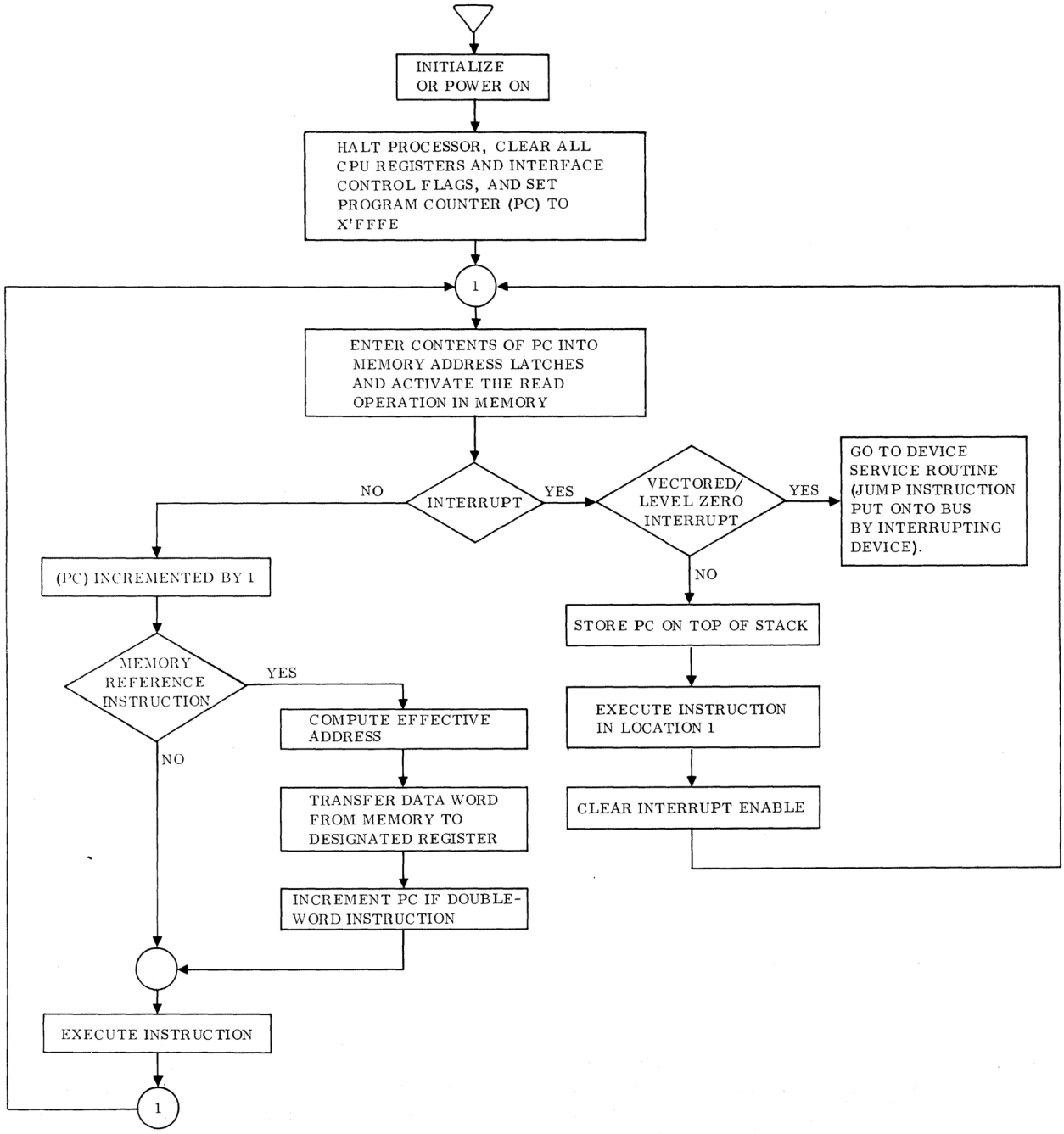


Figure 2-6. IMP-16L Operational Flowchart

In the IMP-16L, the processor itself is always running. If the users program halts, the control panel generates a continuous vectored interrupt which causes continuous execution of the Control Panel Service Routine. When the user runs his program, the control panel ceases the continuous interrupt and interrupts only once every 100 milliseconds to refresh the panel displays.

2.3 MEMORY MODULE

Figure 2-7 is a simplified block diagram of the IMP-16L Memory Module. The diagram shows the various functional groups within the Memory Module and their working relationships with each other. The numbers

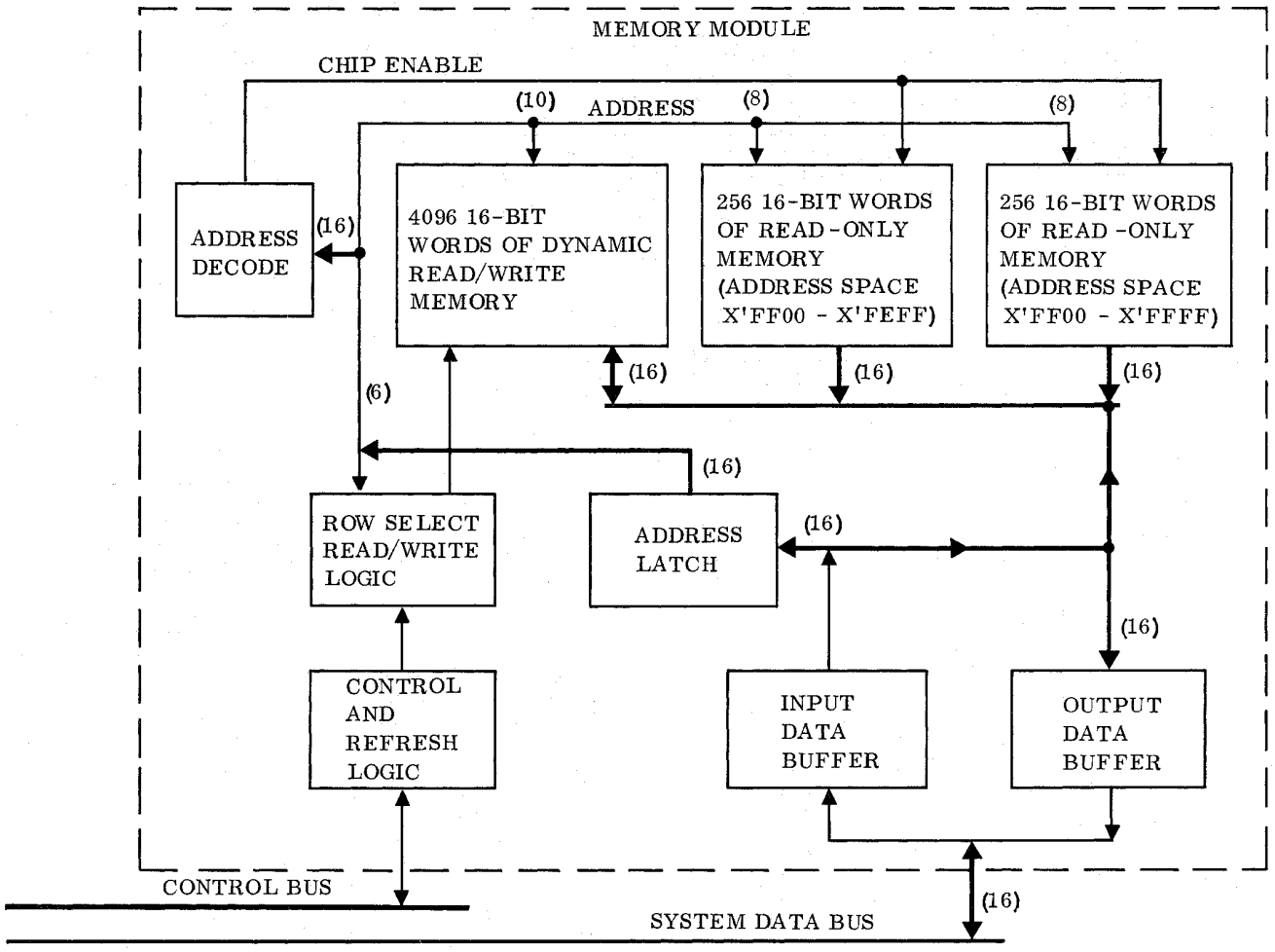


Figure 2-7. IMP-16L Memory Module

shown in parentheses indicate the number of lines employed to accomplish communication tasks. Operation is as follows:

After access and priority are established, address and data are routed from the System Data Bus to the I/O Data Buffers. In the case of a write operation, data from the Input Data Buffer is routed (open arrows) to the 4096-word Dynamic Read/Write Memory. The Address Decode and Row Read/Write Select Logic circuits determine the memory location to be used for data storage. In the case of a read operation, the Address Decode provides a Chip-enable Signal to the ROMs, while the Address Decode and Row Select Read/Write Logic circuits determine the data to be read and outputted (via the Output Data Buffer) to the System Data Bus.

Chapter 3

IMP-16L INSTRUCTION SET

3.1 INTRODUCTION

Twelve functional types of instructions comprise the IMP-16L instruction set:

- Load and Store Instructions
- Single-precision Arithmetic Instructions
- Double-precision Arithmetic Instructions
- Logical Instructions
- Skip Instructions
- Transfer-of-control Instructions
- Interrupt Handling Instructions
- Shift Instructions
- Register Instructions
- Byte Instructions
- Input/Output, Halt, and Control Flag Instructions
- Bit and Status Flag Instructions

The instructions for each functional type are described as a group. For each instruction, the name of the instruction, its mnemonic, its word format, its operation in the form of an equation, and a succinct explanation of its operation are given. A tabulated summary of each type of instruction precedes the detailed descriptions.

Before describing the instructions, brief descriptions of the registers referred to in the instruction descriptions are given.

The IMP-16L instructions and their assembler language op code mnemonics are summarized in appendix A.

3.2 ARITHMETIC AND LOGIC UNITS REFERENCED IN IMP-16L INSTRUCTIONS

The units referenced in the ensuing description of the IMP-16L instructions are listed below and are shown as solid-line blocks in figure 3-1.

- Last-in/First-out (LIFO) Stack
- Register and Arithmetic Logic Unit (RALU) Status Flags
- Program Counter (PC)
- Accumulator 0 (AC0)
- Accumulator 1 (AC1)
- Accumulator 2 (AC2)
- Accumulator 3 (AC3)

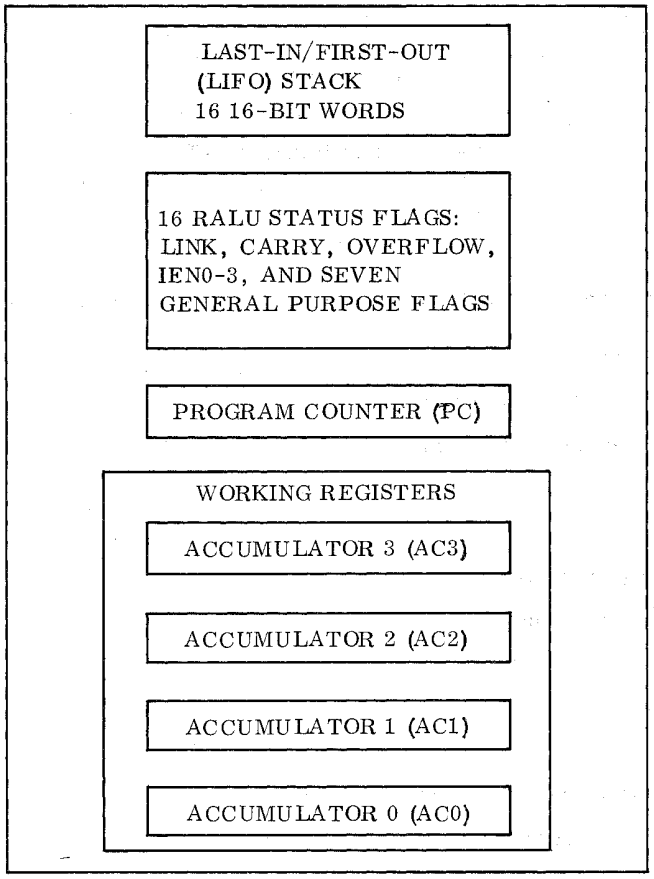


Figure 3-1. Arithmetic and Logic Units Referenced in IMP-16L Instructions

3.2.1 Last-in/First-out Stack

The IMP-16L has a hardware stack that data may be stored in or retrieved from on a last-in/first-out basis. The stack is 16 words deep and is accessible through the top location. As a data word is entered into the stack, the contents of the top location and each other location are pushed downward to the next lower level; if the stack is full, the word in the bottom location is lost. Conversely, the contents of the top location are pulled from the stack during retrieval of a data word; the top location and each lower location are replaced by the contents of the next lower location, and zeros are entered into the bottom location.

The stack is used primarily for saving status during interrupts and for saving subroutine return addresses. It may be used also for temporary storage of data using the PUSH, PULL, XCHRS, PUSHF, AND PULLF instructions (described later in this chapter).

3.2.2 Register and Arithmetic Logic Unit (RALU) Status Flags

There are 16 RALU Status Flags. These may be pushed onto the stack (saved) or loaded from the stack (restored), set, cleared, or tested individually. The Overflow Flag (OV) is set during an arithmetic operation if an arithmetic overflow occurs (that is, if the result is greater than the largest acceptable positive number [32767 or X'EFFF] , the result is less than the smallest acceptable negative number [-32768 or X'8000] , or the possible result of a division is negative). The Carry Flag (CY) is set during an add operation if there is a carry out of the most significant bit.

During such operations, the flags are configured as a 16-bit word; the L (Link), CY (Carry), and OV (Overflow) Flags are the first, second, and third most significant bits, respectively. The remaining 13 flags including

the four Interrupt Enable Flags (IEN0, IEN1, IEN2, and IEN3) comprise the remaining 13 less significant bits.

The L Flag is primarily used in some shifting operations, and the CY and OV Flags are adjuncts for arithmetic operations. The specific uses of the flags are elaborated upon in the appropriate instruction descriptions appearing later in this chapter.

3.2.3 Program Counter

The Program Counter (PC) holds the address of the next memory location in the instruction sequence. When there is a branch to another address in the main memory, the branch address is set into the Program Counter. A skip instruction merely increments the Program Counter by 1, thus causing the one memory location to be skipped.

3.2.4 Accumulators 0, 1, 2, and 3 (AC0, AC1, AC2, and AC3)

The accumulators are used as working registers for data manipulation. Data may be fetched from a location in memory to an accumulator and may be stored from an accumulator to a location in memory. The particular accumulator to take part in an operation is specified by the programmer in the appropriate instruction. Data may be transferred to or from a peripheral via AC0.

3.3 DATA AND INSTRUCTIONS

3.3.1 Data Representation

Data are represented in the IMP-16L in twos-complement integer notation. In this system, the negative of a number is formed by complementing each bit in the data word and adding 1 to the complemented number. The sign is indicated by the most significant bit. In the 16-bit word of the IMP-16L, when bit 15 is a '0', it denotes a positive number; when it is a '1', it denotes a negative number. Maximum number ranges for this system are $7FFF_{16}$ ($+32767_{10}$) and 8000_{16} (-32768_{10}) for single-precision operations. For double-precision operations, the first consecutive word addressed in memory contains the high-order part, and the next consecutive word contains the low-order part of a double-precision number. Bit 15 of the high-order part is the sign bit.

3.3.2 Instructions

There are 12 classes of IMP-16L instructions. Each class of instruction and the associated instructions are summarized in a table preceding the descriptions of the instructions. Also, the applicable instruction word formats are defined.

3.4 MEMORY ADDRESSING

The IMP-16L instruction set provides for direct and indirect memory addressing. For direct addressing, three distinct modes are available: base page (or absolute), Program-Counter relative, and indexed. The mode of addressing is specified by the XR field of the simplified instruction word format shown in figure 3-2. The addressing modes are summarized in table 3-1.

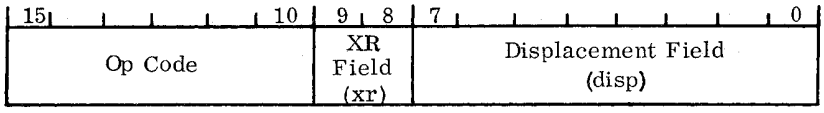


Figure 3-2. Instruction Word Format for Addressing Memory

Table 3-1. Summary of Addressing Modes

XR Field	Addressing Mode	Effective Address	Range
00	Base Page	EA = disp	$0 \leqslant$ disp $\leqslant 255$
01	Relative to Program Counter	EA = disp + (PC)	$-128 \leqslant$ disp $\leqslant 127$
10	Relative to AC2	EA = disp + (AC2)	$-128 \leqslant$ disp $\leqslant 127$
11	Relative to AC3	EA = disp + (AC3)	$-128 \leqslant$ disp $\leqslant 127$

3.4.1 Base Page Addressing

When the XR field is 00, it specifies base page addressing. Base page is directly accessible from any location in the address space of the memory. In this mode, the effective address is formed by setting bits 8 through 15 to zero and using the value of the 8-bit displacement field as an absolute address. Up to 256 words (locations 0 through 255) may be addressed in this way.

3.4.2 Program Counter Relative Addressing

Program-Counter relative addressing is specified when the XR field is 01. The displacement is treated as a signed number such that its sign bit (bit 7) is propagated to bits 8 through 15, and the effective address is formed by adding the contents of the PC to the resulting number. This permits PC-relative addressing -128 and +127 locations from the PC value; however, at the time of formation of the address, the PC has already been incremented in the microprogram and is pointing to the next memory location. Because of this, the actual addressing range is from -127 to +128 from the current instruction.

3.4.3 Indexed Addressing

Indexed addressing is done with reference to Accumulator 2 or 3 (AC2 or AC3). In this mode, the displacement field is again interpreted as a signed 8-bit number from -128 to +127 with the sign (bit 7) extended through bits 8 through 15. The contents of the chosen index register (AC2 when xr = 10_2 and AC3 when xr = 11_2) are added to the number formed from the displacement value to yield an effective address that can reach any location in 65,536 words of memory.

3.4.4 Indirect Addressing

Indirect addressing is accomplished by first calculating the effective address (EA) using the same method used for direct addressing; the memory location at this address contains a number that is then used as the address of the operand. The following instructions use indirect addressing:

- Load Indirect (see table 3-3)
- Store Indirect (see table 3-3)
- Jump Indirect (see table 3-8)
- Jump to Subroutine Indirect (see table 3-8)

A special case of indirect addressing is the Jump to Level 0 Interrupt, Indirect. This instruction adds its displacement to 120_{16} to get the indirect address (to be described later).

3.4.5 Double Word Addressing

Six of the sixty instructions have a double-word instruction format. These instructions use base page, Program-Counter relative, and indexed addressing similar to that described in paragraph 3.3 except for the displacement field length. The six memory-reference instructions are as follows:

- Multiply (MPY)
- Divide (DIV)
- Double Precision Add (DADD)
- Double Precision Subtract (DSUB)
- Load Byte (LDB)
- Store Byte (STB)

The modified instruction format (shown in figure 3-3) is a double-word instruction format with a 16-bit displacement field. When using these instructions, all of memory is directly addressable, although all indexing modes may still be used. The addressing modes are the same as those indicated in table 3-1. It is important to note when considering PC-relative addressing that the PC contains the address of the displacement word of the instruction, rather than the address of the next instruction.

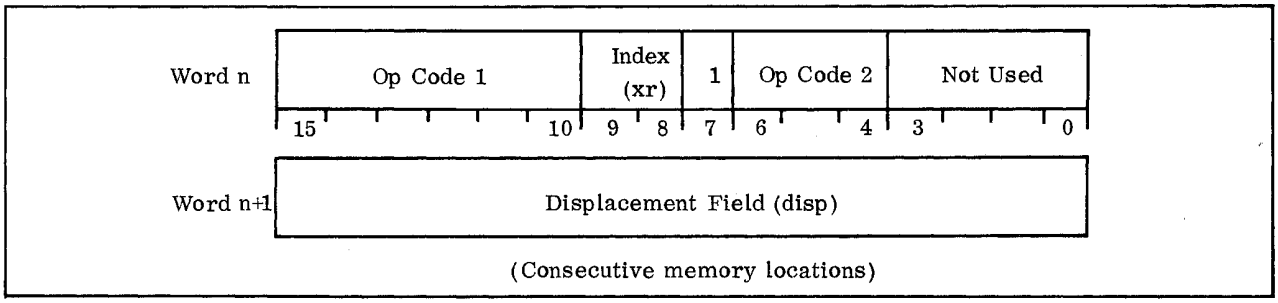


Figure 3-3. Double-word Memory Reference Instruction Format

3.5 NOTATION AND SYMBOLS USED IN IMP-16L INSTRUCTION DESCRIPTIONS

Refer to table 3-2 for definitions of the notation and symbols used in the IMP-16L instruction descriptions. The notations are given first in alphabetical order followed by the symbols. Upper-case mnemonics refer to fields in the instruction word; lower-case mnemonics refer to the numerical value of the corresponding fields. In cases where both lower and upper-case mnemonics are composed of the same letters, only the lower-case mnemonic is given in table 3-2. The use of lower-case notation designates variables.

Table 3-2. Notation Used in Instruction Descriptions

Notation	Meaning
ACr	Denotes a specific working register (AC0, AC1, AC2, or AC3), where r is the number of the accumulator referenced in the instruction.
AR	Denotes the address register used for addressing memory or peripheral devices.
cc	Denotes the 4-bit condition code value for conditional branch instructions.
ctl	Denotes the 7-bit control-field value for flag, input/output, and miscellaneous instructions.

Table 3-2. Notation Used in Instruction Descriptions (Continued)

Notation	Meaning
CY	Indicates that the Carry Flag is set if there is a carry due to the instruction (either an addition or a subtraction).
disp	Stands for displacement value and it represents an operand in a nonmemory reference instruction or an address field in a memory reference instruction. It is a signed twos-complement number except when base page is referenced; in the latter case, it is unassigned.
dr	Denotes the number of a destination working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
EA	Denotes the effective address specified by the instruction directly, indirectly, or by indexing. The contents of the effective address are used during execution of an instruction. See table 3-1.
fc	Denotes the number of the referenced flag (see table 3-15 under paragraph 3.6.11, Input/Output, Halt, and Flag Instructions).
INTEN	Denotes the Interrupt Enable Control Flag.
IOREG	Denotes an input/output register in a peripheral device.
L	Denotes 1-bit Link (L) Flag.
OV	Indicates that the Overflow Flag is set if there is an overflow due to the instruction (either an addition, subtraction, or division).
PC	Denotes the Program Counter. During address formation, it is incremented by 1 to contain an address 1 greater than that of the instruction being executed.
r	Denotes the number of a working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
SEL	Denotes the Select Control Flag. It is used to select the carry or overflow for output on the Carry and Overflow (CYOV) Line of the CPU, and to include the Link Bit (L) in shift operations.
sr	Denotes the number of a source working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
xr	When not zero, this value designates the number of the register to be used in the indexed and relative memory-addressing modes. See table 3-1.
( )	Denotes the contents of the item within the parentheses. (ACr) is read as "the contents of ACr." (EA) is read as "the contents of EA."
[ ]	Denotes "the result of."
~	Indicates the logical complement (ones complement) of the value on the right-hand side of ~.
←	Means "is replaced by."
@	Appearing in the operand field of an instruction, denotes indirect addressing.

Table 3-2. Notation Used in Instruction Descriptions (Continued)

Notation	Meaning
$\wedge$	Denotes an AND operation.
$\vee$	Denotes an OR operation.
$\bar{\vee}$	Denotes an exclusive OR operation.

3.6 INSTRUCTION DESCRIPTIONS

Each class and subclass of instruction is introduced by a table that lists and summarizes the instructions. The word format then is illustrated. Detailed descriptions are given, providing the following information:

- Name of instruction followed by operation code mnemonic in parentheses
- Operation Code in word format diagram
- Operation in equation notation
- Description of operation in detail

3.6.1 Load and Store Instructions

There are four instructions in this group. These are summarized in table 3-3, and then individually described. The word format is shown in figure 3-4.

Table 3-3. Load and Store Instructions

Instruction	Op Code	Operation	Assembler Format	
Load	1000	(ACr) ← (EA)	LD	r, disp(xr)
Load Indirect	1001	(ACr) ← ((EA))	LD	r, @disp(xr)
Store	1010	(EA) ← (ACr)	ST	r, disp(xr)
Store Indirect	1011	((EA)) ← (ACr)	ST	r, @disp(xr)

NOTE

For indirect operations, the symbol @ must precede the memory location designated in the operand field of the assembler instruction.

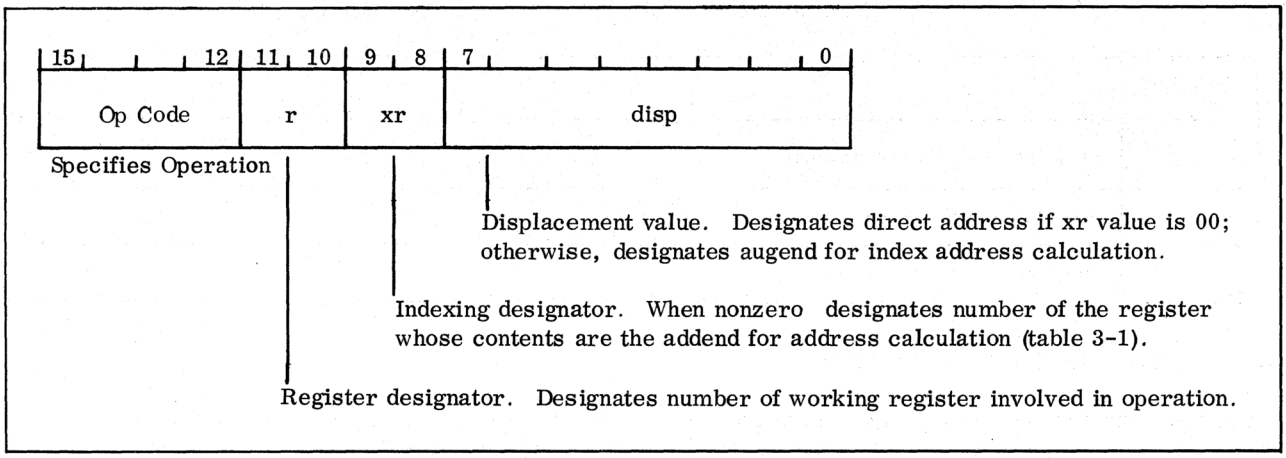
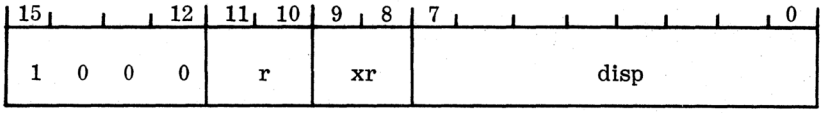


Figure 3-4. Load and Store Instruction Format

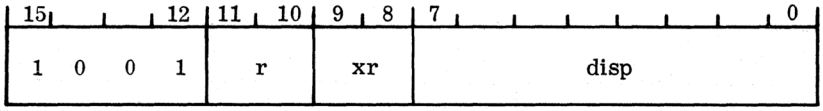
Load (LD)



Operation: (ACr) ← (EA)

Description: The contents of ACr are replaced by the contents of EA. The initial contents of ACr are lost; the contents of EA are unaltered.

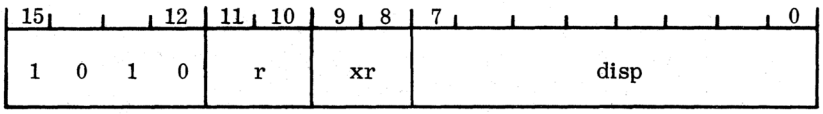
Load (LD) Indirect



Operation: (ACr) ← ((EA))

Description: The contents of ACr are replaced indirectly by the contents of EA. The initial contents of ACr are lost; the contents of EA and the location that designates EA are unaltered.

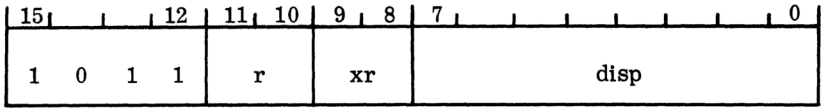
Store (ST)



Operation: (EA) ← (ACr)

Description: The contents of EA are replaced by the contents of ACr. The initial contents of EA are lost; the contents of ACr are unaltered.

Store (ST) Indirect



Operation: ((EA)) ← (ACr)

Description: The contents of EA are replaced indirectly by ACr. The initial contents of EA are lost; the contents of ACr and the location that designates EA are unaltered.

3.6.2 Single-precision Arithmetic Instructions

There are four instructions in this group, summarized in table 3-4 and then described individually. Two of the instructions are single-word instructions (Add and Subtract), and two are double-word instructions (Multiply and Divide). The word format for the single-word instructions is shown in figure 3-5, and for the double-word instructions, in figure 3-6.

Table 3-4. Single-precision Arithmetic Instructions

Instruction	Op Code	Operation	Assembler Format
Add (ADD)	1100	(ACr) ← (ACr) + (EA); OV; CY	ADD r, disp(xr)
Subtract (SUB)	1101	(ACr) ← (ACr) + ~(EA) + 1; OV; CY	SUB r, disp(xr)

Instruction	Op Code		Operation	Assembler Format
	OP1	OP2		
Multiply (MPY) ((EA) ≥ 0)	000001	1000	(AC0), (AC1) ← (AC1)*(EA); SEL ← 0; L altered	MPY disp(xr)
Divide (DIV) ((EA) >0))	000001	1001	(AC0) ← INTEGER PART OF [(AC0), (AC1) ÷(EA)]; (AC1) ← REMAINDER OF [(AC0), (AC1) ÷(EA)]; OV; SEL ←0; L altered	DIV disp(xr)

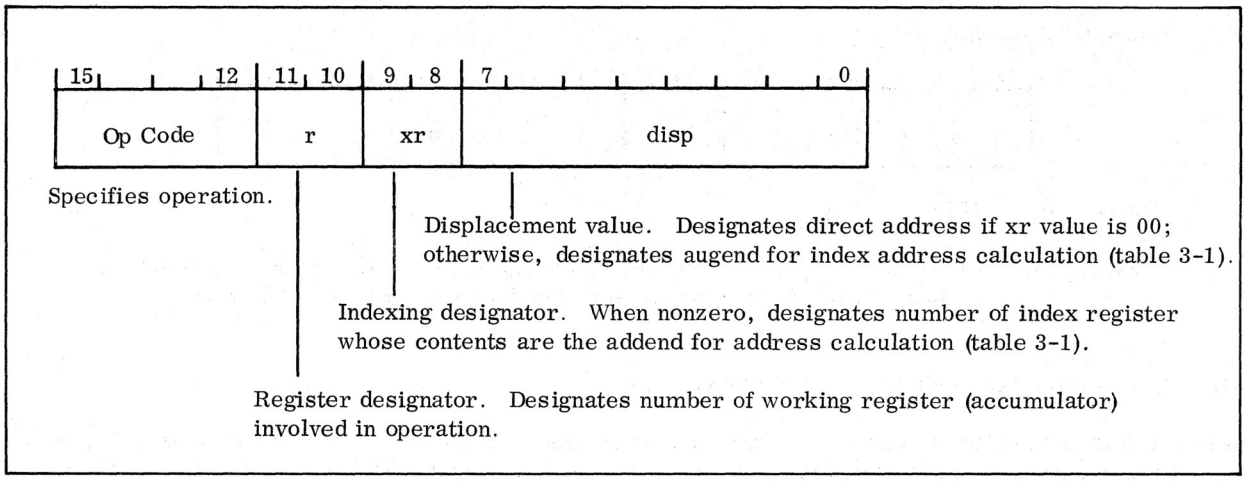
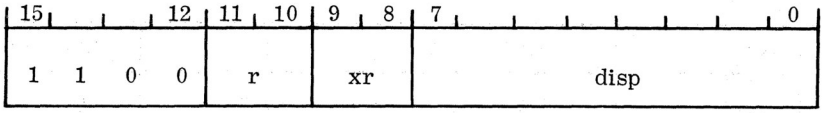


Figure 3-5. Single-precision (Single-word) Arithmetic Instruction Format

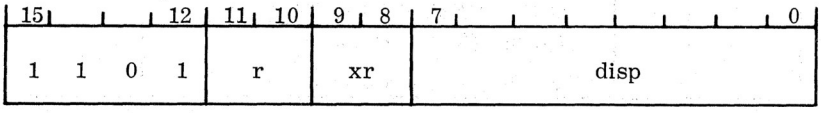
Add (ADD)



Operation: (ACr) ← (ACr) + (EA), OV, CY

Description: The contents of ACr are added algebraically to the contents of the effective memory location EA. The sum is stored in ACr, and the contents of EA are unaltered. The preceding contents of ACr are lost. The Carry and Overflow Flags are set according to the result of the operation.

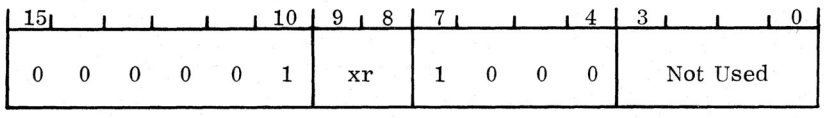
Subtract (SUB)

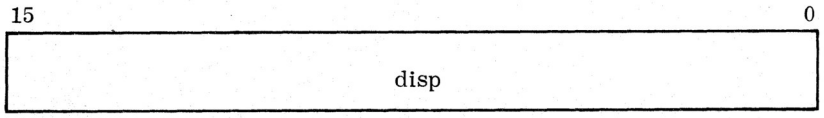


Operation: (ACr) ← (ACr) + ∼(EA) + 1, OV, CY

Description: The contents of ACr are added to the twos-complement of the effective memory location EA. The result is stored in ACr, and the effective memory location is unaltered. The Carry and Overflow Flags are set according to the result of the operation.

Multiply (MPY)

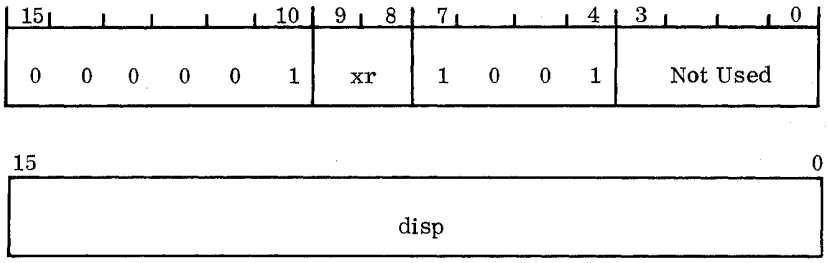




Operation: (AC0), (AC1) ← (AC1) * (EA); SEL ← 0; L altered

Description: The unsigned integer in AC1 is multiplied by the positive integer in the effective memory location EA. The high-order part of the 32-bit result is stored in AC0 and the low-order part is stored in AC1. The previous contents of AC0 and AC1 are lost. The contents of EA are unaffected. The Select Flag is cleared. The Link Flag is left in an arbitrary state.

Divide (DIV)



Operation: (AC0), (AC1) ← (AC0), (AC1) ÷ (EA); OV; SEL ← 0; L altered

Description: The positive 32-bit integer in AC0 (high-order part) and AC1 (low-order part) is divided by the contents (a positive number) of the effective memory location EA. The integer quotient is placed in AC1 and the remainder in AC0. The Overflow Flag(OV) is set if either of the following occurs: (1) the high-order part of the dividend (initial contents of AC0) is greater than or equal to the divisor; or (2) the quotient is negative. The Select Flag is cleared. The Link Flag is left in an arbitrary state. The contents of EA are unchanged. Division by zero, an illegal operation, falls into case one above.

3.6.3 Double-precision Arithmetic Instructions

There are two instructions in this group, both double-word instructions. These are summarized in table 3-5 and then individually described. The word format is shown in figure 3-6.

Table 3-5. Double-precision Arithmetic Instructions

Instruction	Op Code		Operation	Assembler Format
	OP1	OP2		
Double-precision Add	000001	1010	(AC0), (AC1) ← (AC0), (AC1) + (EA), (EA + 1); OV; CY; SEL ← 0	DADD disp(xr)
Double-precision Subtract	000001	1011	(AC0), (AC1) ← (AC0), (AC1) + ~(EA), ~(EA + 1) + 1; OV; CY; SEL ← 0	DSUB disp(xr)

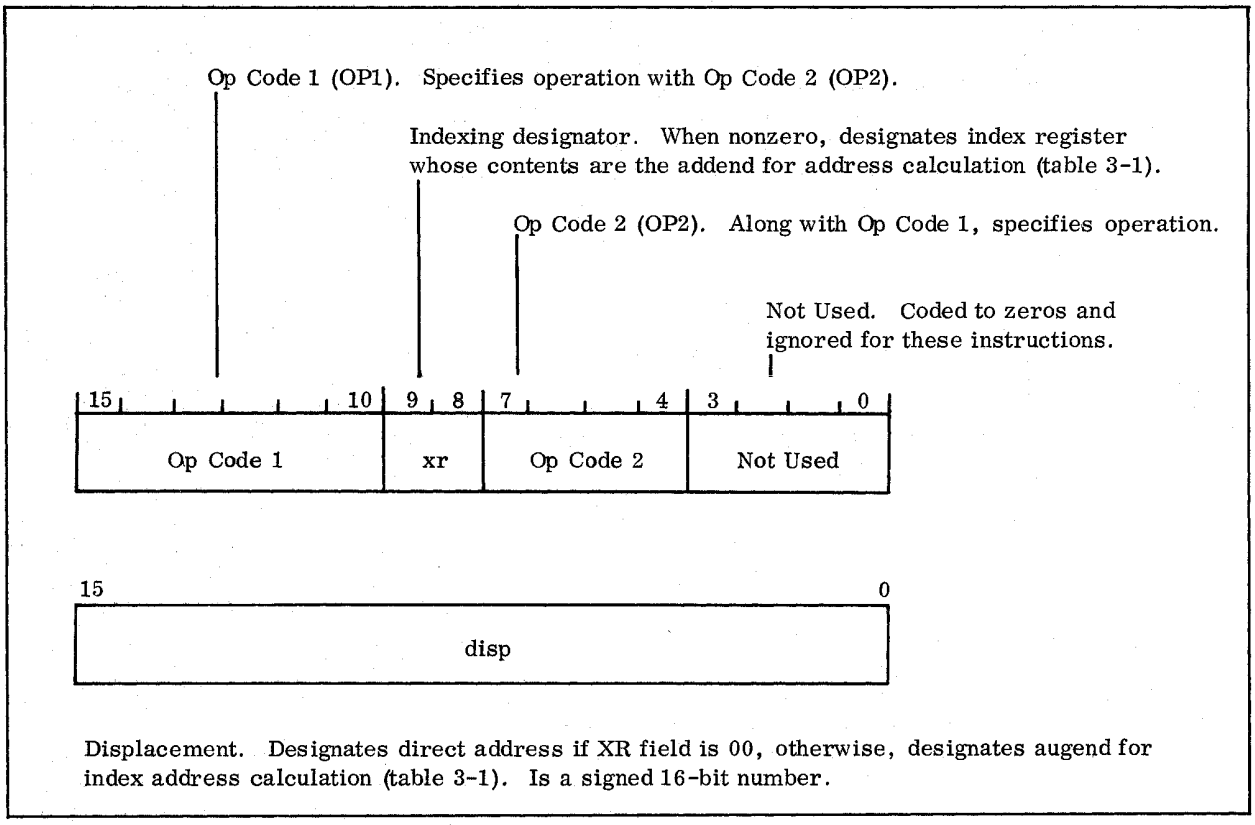
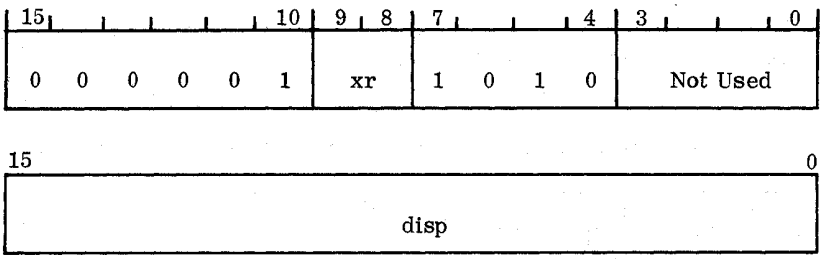


Figure 3-6. Double-precision (Double-word) Arithmetic Instruction Format

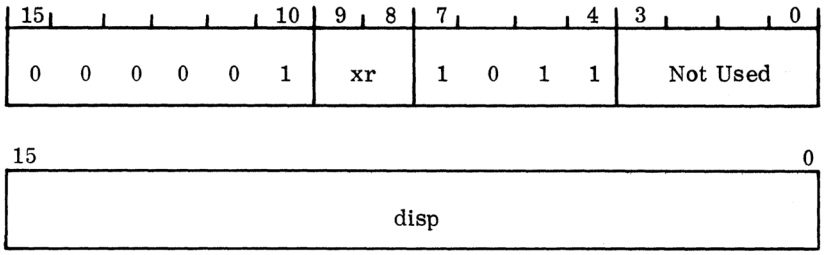
Double-precision Add (DADD)



Operation: (AC0), (AC1) ← (AC0), (AC1) + (EA), (EA+1); OV; CY; SEL ← 0

Description: The double-precision twos-complement value in AC0 (high order) and AC1 (low order) is added to the double-precision twos-complement value in EA (high order) and EA+1 (low order) and the result is stored in AC0 and AC1. The contents of EA and EA+1 are unchanged. The Overflow or Carry Flag is set if an overflow or carry occurs, respectively; otherwise, they are cleared. The Select Flag is cleared.

Double-precision Subtract (DSUB)



Operation: (AC0), (AC1) ← (AC0), (AC1) + ~ [(EA), (EA+1)] + 1; OV; CY; SEL← 0

Description: The double-precision twos-complement value in EA (high order) and EA+1 (low order) is subtracted from the double-precision twos-complement value in AC0 (high order) and AC1 (low order). The contents of EA and EA+1 are unchanged. The Overflow or Carry Flag is set if an overflow or carry occurs, respectively; otherwise, they are cleared. The Select Flag is cleared.

3.6.4 Logical Instructions

There are two instructions in this group, summarized in table 3-6 and then described individually. Either of these instructions may be carried out with only two of the general-purpose Accumulators, AC0 or AC1. The word format is shown in figure 3-7.

Table 3-6. Logical Instructions

Instruction	Op Code	Operation	Assembler Format
AND	01100	(ACr) ← (ACr) ∧ (EA)	AND r, disp(xr)
OR	01101	(ACr) ← (ACr) ∨ (EA)	OR r, disp(xr)

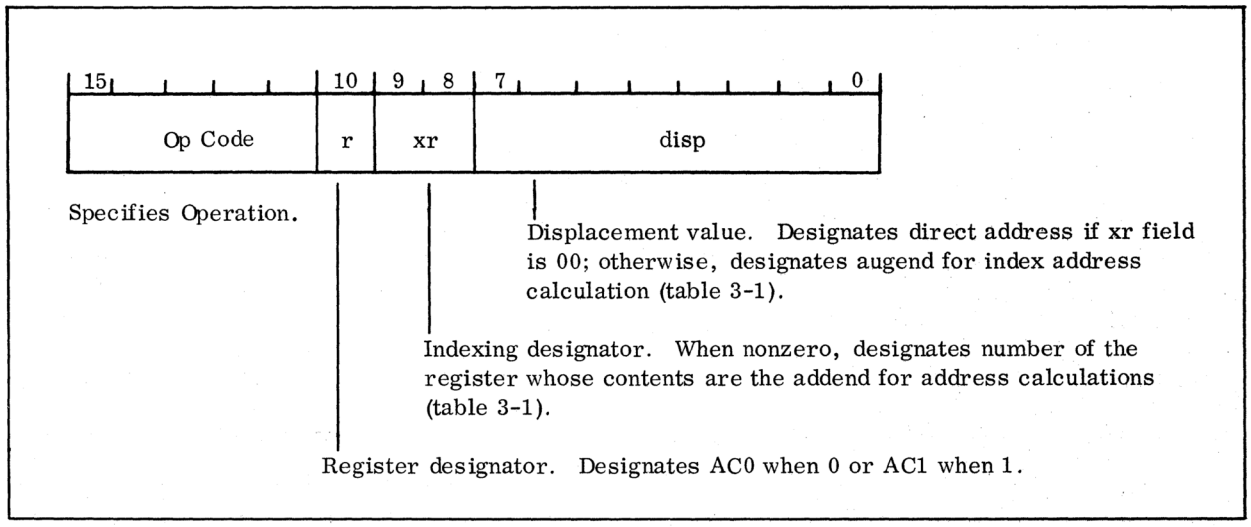
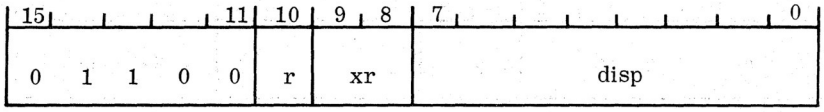


Figure 3-7. Logical Instruction Format

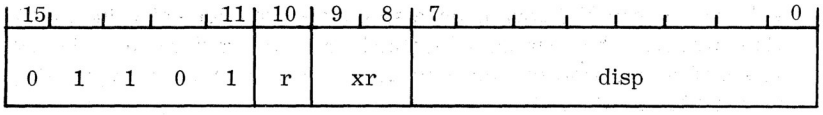
And (AND)



Operation: (ACr) ← (ACr) ∧ (EA)

Description: The contents of ACr (where r is either 0 or 1) and the contents of the effective memory location EA are ANDed, and the result is stored in ACr. The initial contents of ACr are lost, and the contents of EA are unaltered.

Or (OR)



Operation: (ACr) ← (ACr) ∨ (EA)

Description: The contents of ACr (where r is either 0 or 1) and the contents of the effective memory location EA are ORed inclusively, and the result is stored in ACr. The initial contents of ACr are lost, and the contents of EA are unaltered.

3.6.5 Skip Instructions

Seven instructions comprise the skip instructions, summarized in table 3-7. Five word formats are required; the instruction formats are shown in figure 3-8.

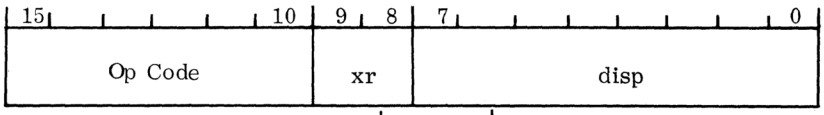
Table 3-7. Skip Instructions

Instruction	Op Code	Operation	Assembler Format
Memory References:			
Increment and Skip if Zero	011110	(EA) ← (EA) + 1; If (EA) = 0, (PC) ← (PC) + 1	ISZ disp(xr)
Decrement and Skip if Zero	011111	(EA) ← (EA) - 1; If (EA) = 0, (PC) ← (PC) + 1	DSZ disp(xr)
Register References:			
Skip if Greater	1110	If (ACr) > (EA), (PC) ← (PC) + 1	SKG r, disp(xr)
Skip if Not Equal	1111	If (ACr) ≠ (EA), (PC) ← (PC) + 1	SKNE r, disp(xr)
Limited Register Reference:			
Skip if AND is Zero	01110	If [(ACr) ∧ (EA)] = 0, (PC) ← (PC) + 1	SKAZ r, disp(xr)

Table 3-7. Skip Instructions (Continued)

Instruction	OP2	Operation	Assembler Format
Skip if Status Flag True	1110100	If Status Flag n = 1, then (PC) ← (PC) + 1; SEL ← 0	SKSTF n
Skip if Bit True	1110101	If $AC0_n$ = 1, then (PC) ← (PC) + 1; SEL ← 0	SKBIT n

Memory Reference Skip Instruction

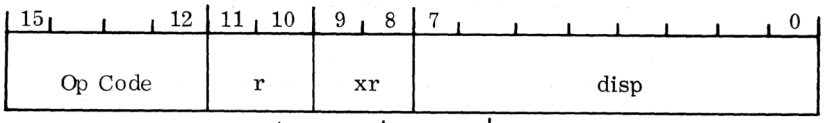


Specifies operation.

Displacement Value. Designates direct address if xr value is 00; otherwise, designates augend for the address calculation (table 3-1).

Indexing Designator. When nonzero, designates number of the register whose contents are the addend for address calculation (table 3-1).

Register Reference Instruction



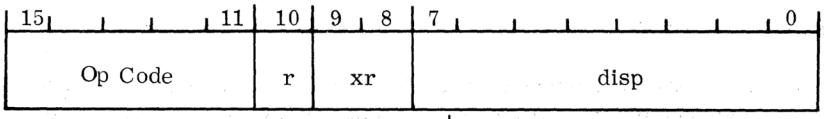
Specifies operation.

Displacement value. Designates direct address if xr value is 00; otherwise, designates augend for the address calculation (table 3-1).

Indexing designator. When nonzero, designates number of index register whose contents are the addend for address calculation (table 3-1).

Register designator. Specifies number of register whose contents are to be compared with contents of effective address in determining whether skip operation occurs. May be AC0, 1, 2, or 3.

Limited Register Reference Instruction



Specifies operation.

Displacement value. Designates direct address if xr value is 00; otherwise, designates augend for the address calculation (table 3-1).

Indexing designator. When nonzero, designates number of the register whose contents are the addend for address calculation (table 3-1).

Register designator. Specifies number of register whose contents are compared with contents of effective address in determining whether skip operation occurs. May be only AC0 or AC1.

Figure 3-8. Skip Instruction Formats (Sheet 1 of 2)

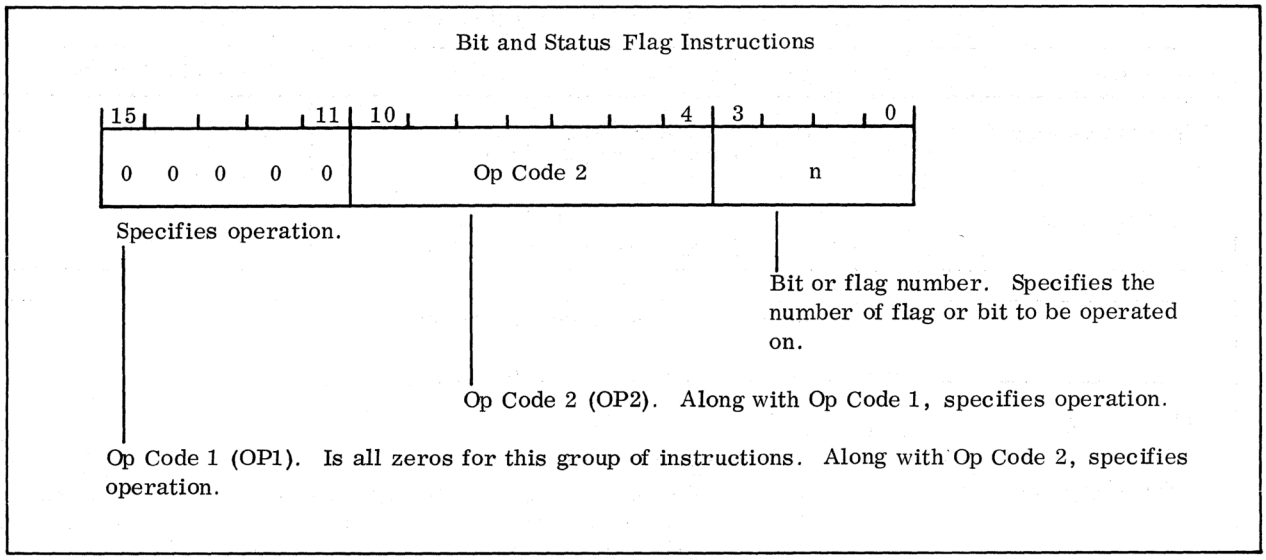
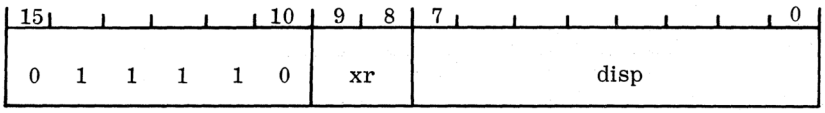


Figure 3-8. Skip Instruction Formats (Sheet 2 of 2)

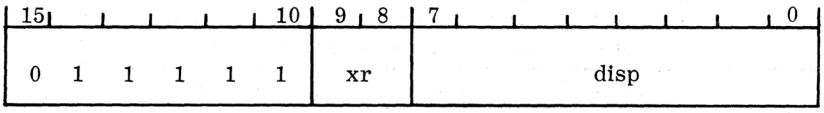
Increment and Skip If Zero (ISZ)



Operation: (EA) ← (EA) + 1; if (EA) = 0, (PC) ←(PC) + 1

Description: The contents of EA are incremented by 1. The new contents of EA are tested to determine whether they equal zero. If the new contents of EA equal zero, the contents of PC are incremented by 1, thus skipping the memory location designated by the initial contents of PC. The contents of EA are unaltered.

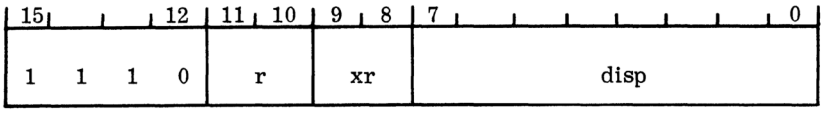
Decrement and Skip If Zero (DSZ)



Operation: (EA) ←(EA) - 1; if (EA) = 0, (PC) ←(PC) + 1

Description: The contents of EA are decremented by 1. The new contents of EA are tested to determine whether they equal zero. If the new contents of EA equal zero, the contents of PC are incremented by 1, thus skipping the memory location designated by the initial contents of PC.

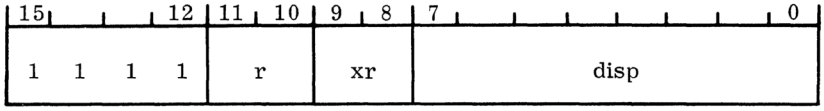
Skip If Greater (SKG)



Operation: If (ACr) > (EA), (PC) ← (PC) + 1

Description: The contents of ACr (when r is AC0, 1, 2, or 3) and the contents of the effective memory location EA are compared on an algebraic basis with due regard to the signs of the two operands. If the contents of ACr are greater than the contents of EA, the contents of PC are incremented by 1, thus skipping the memory location designated by the initial contents of PC. The initial contents of PC are lost. The contents of ACr and EA are unaltered.

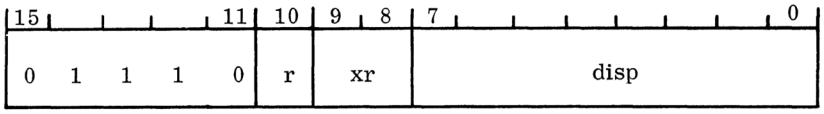
Skip If Not Equal (SKNE)



Operation: If ACr ≠ (EA), (PC) ← (PC) + 1

Description: The contents of ACr (where ACr is AC0, 1, 2, or 3) and the contents of the effective memory location EA are compared. If the contents of ACr and the effective memory location EA are not equal, the contents of PC are incremented, thus skipping the memory location designated by the initial contents of PC. The initial contents of PC are lost. The contents of ACr and EA are unaltered.

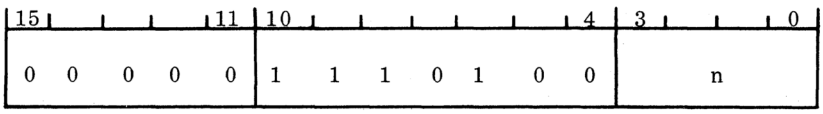
Skip If AND Is Zero (SKAZ)



Operation: If [(ACr) ∧ (EA)] = 0, (PC) ← (PC) + 1

Description: The contents of ACr (where r is either 0 or 1) and the contents of the effective memory location EA are ANDed. If the result equals zero, the contents of PC are incremented by 1, thus skipping the memory location designated by the initial contents of PC. The initial contents of PC are lost. The contents of ACr and EA are unaltered.

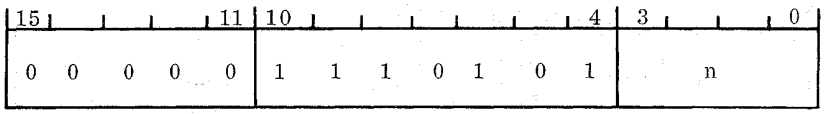
Skip If Status Flag True (SKSTF)



Operation: If n Status Flag is 1, (PC) ← (PC) + 1; SEL ← 0

Description: If Status Flag n is true, the next memory location is skipped. The contents of the Status Flags are unaffected. The Select Flag is cleared. (15 ≥ n ≥ 0)

Skip If Bit True (SKBIT)



Operation: If $AC0_n = 1$, (PC) ← (PC) + 1; SEL ← 0

Description: If bit n of AC0 is true, the next memory location in sequence is skipped. The contents of AC0 are unaffected. The Select Flag is cleared. ($15 \geq n \geq 0$)

NOTE

Caution should be taken when coding a skip instruction to prevent the skip condition from jumping into the displacement field of a double-word instruction.

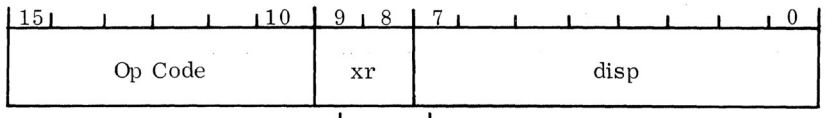
3.6.6 Transfer-of-control Instructions

There are ten instructions in this group, summarized in table 3-8 and then described individually. Five word formats are required and are shown in figure 3-9.

Table 3-8. Transfer-of-control Instructions

Instruction	Op Code	Operation	Assembler Format	
Jumps:				
Jump	001000	(PC) ← EA	JMP	disp(xr)
Jump Indirect	001001	(PC) ← (EA)	JMP	@disp(xr)
Jump to Subroutine	001010	(STK) ← (PC); (PC) ← EA	JSR	disp(xr)
Jump to Subroutine Indirect	001011	(STK) ← (PC); (PC) ← (EA)	JSR	@disp(xr)
Branch:				
Branch-on Condition	0001	If CC IS TRUE (PC) ← (PC) + disp	BOC	cc, disp
Returns:				
Return from Interrupt	000000010	(PC) ← (STK) + ctl; INTEN FLAG SET	RTI	ctl
Return from Subroutine	000000100	(PC) ← (STK) + ctl	RTS	ctl
Special Jump Instructions:				
Jump to Subroutine Implied	000000111	(STK) ← (PC); (PC) ← $FF80_{16}$ + ctl	JSRI	ctl
Jump to Subroutine through Pointer	000000110	(STK) ← (PC); (PC) ← (ctl + 100_{16})	JSRP	ctl
Jump through Pointer	000001010	(PC) ← (100_{16} + disp)	JMPP	disp

Jump Instructions

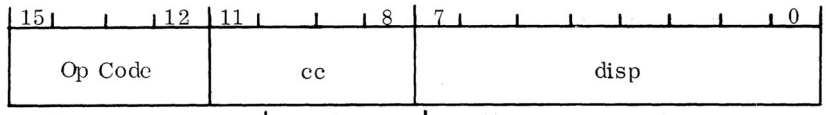


Specifies operation.

Displacement value. Designates direct address if xr value is 00; otherwise, designates augend for the address calculation (table 3-1).

Indexing designator. When nonzero, designates number of the register whose contents are the addend for address calculation (table 3-1).

Branch-on Condition (BOC) Instruction

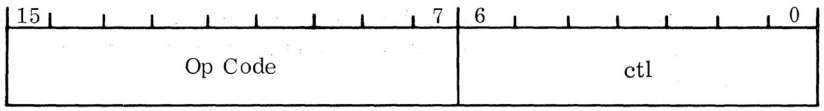


Specifies operation.

Displacement value. Designates immediate value that is added to the contents of the Program Counter (PC) to calculate the address for the next instruction that follows the branch instruction if the branch condition is true.

Condition code. Specifies branch condition code (listed in table 3-8).

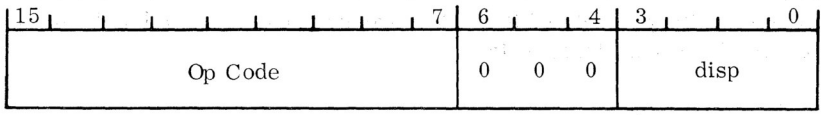
Returns, Jump to Subroutine Implied, and Jump to Subroutine through Pointer Instructions



Specifies operation.

Control field. Designates a value added to the return address when used in return instructions. Designates the control field transferred to peripheral device in input/output instructions. The ctl value is an unsigned 7-bit number.

Jump through Pointer

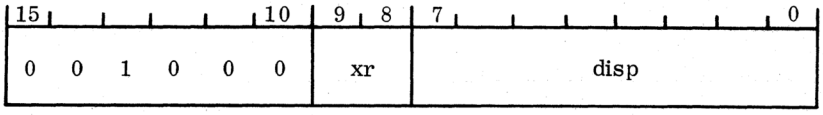


Specifies operation.

Displacement field. Designates a value added to X'100 to calculate pointer address.

Figure 3-9. Transfer-of-control Instruction Formats

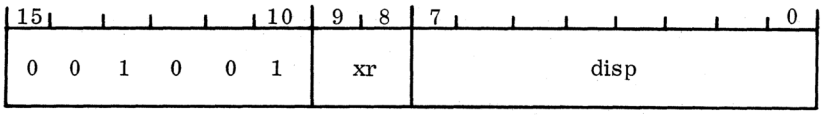
Jump (JMP)



Operation: (PC) ← EA

Description: The effective address EA replaces the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

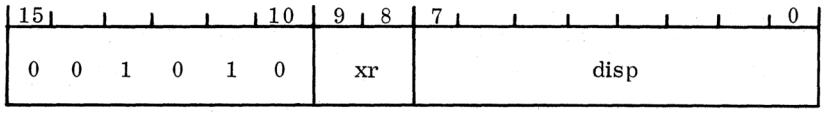
Jump (JMP) Indirect



Operation: (PC) ← (EA)

Description: The contents of the effective address (EA) replaces the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

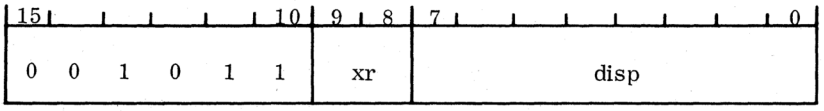
Jump to Subroutine (JSR)



Operation: (STK) ← (PC), (PC) ← (EA)

Description: The contents of PC are pushed onto the top of the stack. The effective address EA replaces the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

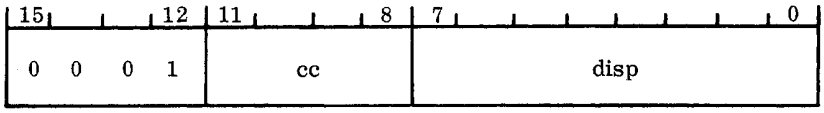
Jump to Subroutine (JSR) Indirect



Operation: (STK) ← (PC), (PC) ← (EA)

Description: The contents of PC are pushed onto the top of the stack. The contents of the effective address (EA) replace the contents of PC. The next instruction is fetched from the location designated by the new contents of PC.

Branch-on Condition (BOC)



Operation: (PC) ← (PC) + disp (sign extended from bit 7 through bit 15)

Description: There are 16 possible condition codes (cc). These are listed in table 3-9. If the condition for branching designated by cc is true, the value of disp (sign extended from bit 7 through bit 15) is added to the contents of PC, and the sum is stored in PC. The initial contents of PC are lost. Program control is transferred to the location specified by the new contents of PC.

NOTE

(1) PC is always incremented by 1 immediately following the fetching of an instruction, so the contents of PC during execution of an instruction is 1 greater than the address of that instruction. This must be considered during execution of the BOC instruction: for example, if the address of the BOC instruction is 100, then 101 is added to the value of disp (sign extended).

(2) The disp field is a signed 8-bit number, whose sign is extended from bit 7 through bit 15 to form a 16-bit number (including sign). Thus, the range of addressing with a BOC instruction is -127 to +128 relative to the address of the current instruction.

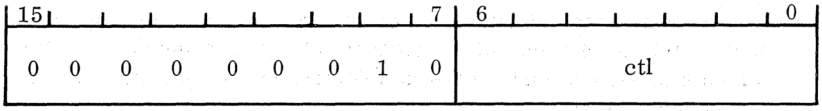
Table 3-9. Branch-on-condition Codes

Condition Code	Condition Tested	Remarks
0000	Interrupt Line = 1	Interrupt need not be tested by macroprogram
0001	(AC0) = 0	
0010	(AC0) ≥ 0	
0011	Bit 0 of AC0 = 1	
0100	Bit 1 of AC0 = 1	
0101	(AC0) ≠ 0	
0110	Control Panel Interrupt Line = 1	Usable only by microprogram
0111	Control Panel Start = 1	
1000	Stack Full Line = 1	
1001	Interrupt Enable = 1	
1010	Carry/Overflow = 1	Carry if SEL = 0; overflow if SEL = 1
1011	(AC0) ≤ 0	
1100	POA	Peripheral Order Accepted
1101	SEL	Select
1110	User	Available for general-purpose use
1111	User	

NOTE

For both the following instructions (RTI and RTS), the ctl value is an unsigned 7-bit number.

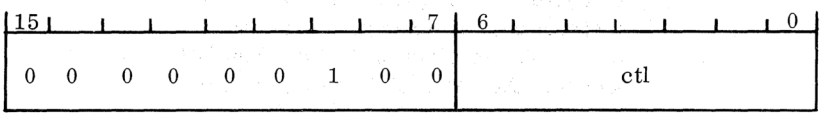
Return from Interrupt (RTI)



Operation: Set INT EN (Interrupt Enable Flag)
(PC) ← (STK) + ctl

Description: The Interrupt Enable Flag (INT EN) is set. The contents of PC are replaced by the sum of ctl and the contents of STK. Program control is transferred to the location specified by the new contents of PC. (RTI is used primarily to exit from an interrupt routine.)

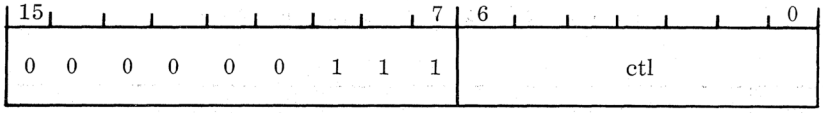
Return from Subroutine (RTS)



Operation: (PC) ← (STK) + ctl

Description: The contents of PC are replaced by the sum of ctl and the contents of STK. Program control is transferred to the location specified by the new contents of PC. (RTS is used primarily to return from subroutines entered by JSR.)

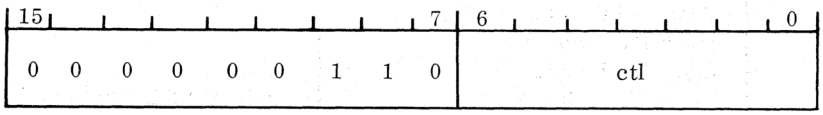
Jump to Subroutine Implied (JSRI)



Operation: (STK) ← (PC), (PC) ← $FF80_{16}$ + ctl

Description: The contents of PC are pushed onto the stack. The contents of PC are replaced by the address implied by the sum of the ctl value and the number $FF80_{16}$. This enables a subroutine jump to memory locations $FF80_{16}$ through $FFFF_{16}$ ($0 \leq ctl \leq 7F_{16}$).

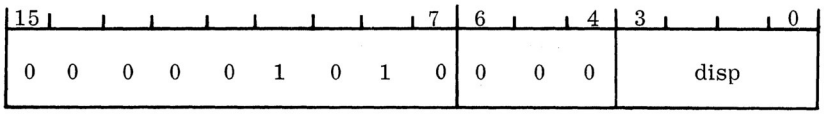
Jump to Subroutine Through Pointer (JSRP)



Operation: (STK) ← (PC), (PC) ← ([(100_{16}) + ctl])

Description: The contents of the Program Counter are pushed onto the top of the stack. The new contents of the PC are set equal to the contents of the memory location addressed by the sum of the contents of location 100_{16} and ctl. The control value ctl is an unsigned 7-bit number for the JSRP instruction.

Jump Through Pointer (JMPP)



Operation: (PC) ← ([(100_{16}) + disp])

Description: The contents of the PC are set equal to the contents of the memory location addressed by the sum of the contents of location 100_{16} and disp. The displacement value disp is an unsigned 4-bit number for the JMPP instruction.

3.6.7 Interrupt Handling Instructions

There are two instructions in this group, listed in table 3-10. Their word format is shown in figure 3-10. The instruction descriptions follow.

Table 3-10. Interrupt Handling Instructions

Instruction	Op Code 2	Operation	Assembler Format

Interrupt Scan	1010001	If (AC1) = 0, SEL ← 0;
		If (AC1) ≠ 0, SEL ← 0;
		AC1 ← [shift AC right 1] until 1 shifted out
		(AC2) ← (AC2) + number of shifts; (PC) ← (PC) + 1
			ISCAN

Jump Indirect to Level 0 Interrupt	1010010	(STK) ← (PC); (PC) ← ([120_{16} + disp]) INTEN ← 0	JINT disp

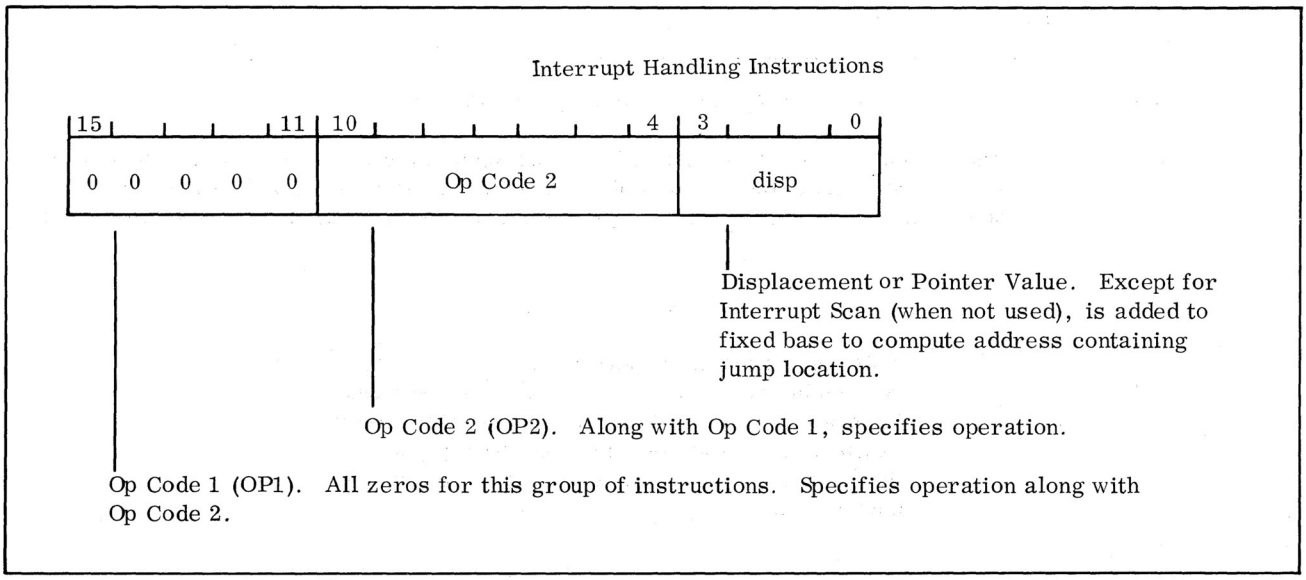
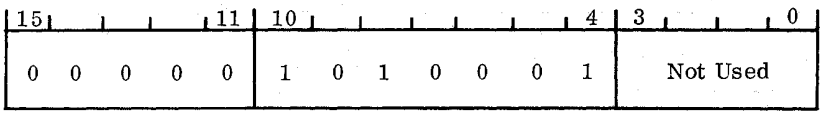


Figure 3-10. Interrupt and Word Jump Formats

Interrupt Scan (ISCAN)



Operation: If AC1 = 0, Sel ← 0;

Else, Sel ← 0;

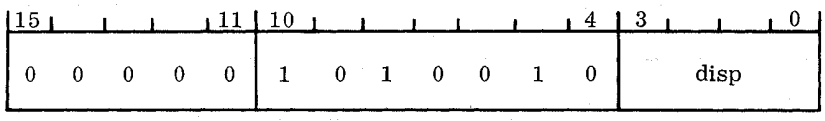
(AC1) ← (shift AC1 right 1) until 1 shifted out
(AC2) ← (AC2) + number of shifts;
(PC) ← (PC) + 1

Description: If AC1 = 0, the Select Flag is cleared and the next sequential instruction is executed. If AC1 ≠ 0, then the Select Flag is cleared and AC1 is shifted right until a 1 is shifted out of bit zero. The number of shifts which occurred is added to the contents of AC2. The next memory location is skipped.

NOTE

This instruction is intended for use in servicing devices which respond to the Interrupt Select Status Order code described in chapter 5. AC1 should be loaded with the Interrupt Select Status word and AC2 should be loaded with the base address of a pointer table for the Interrupt Service Routines. The base address is 1 less than the address of the first pointer.

Jump to Level 0 Interrupt, Indirect (JINT)



Operation: (STK) ← (PC); (PC)←([120_{16} + disp]); INTEN ←0

Description: The contents of the PC are pushed onto the top of the stack. The new contents of the PC are set equal to the contents of the memory location whose address is formed by adding disp to 120_{16}. The Interrupt Enable Flag is cleared. Flag 13 (SCPIE) is pulsed.

NOTE

This instruction is intended for use in servicing devices on interrupt request level 0 as described in paragraph 6.3.3. This instruction is normally executed as a result of a level 0 interrupt, with the instruction being provided by the interrupting device's interface.

3.6.8 Shift Instructions

Four instructions comprise this group. All four instructions may be used with the Link (L) bit by setting the SEL Flag. This is accomplished with a Set Flag (SFLG) Instruction before executing the Shift or Rotate Instruction. Examples of shifting with and without SEL set are given in diagram form for each instruction in the descriptions that follow. Note that the SEL Flag also affects the BOC instructions as indicated in table 3-9.

The shift instructions are summarized in table 3-11, and the word format is shown in figure 3-11.

All shift and rotate operations may be carried out with any of the four general-purpose Accumulators, AC0, 1, 2, or 3.

Table 3-11. Shift Instructions

Instruction	Op Code	Operation		Assembler Format
		SEL = 0	SEL = 1	
Rotate Left (disp = m)	010110	$(ACr_0) \leftarrow (ACr_{15})$, $(ACrn) \leftarrow (ACrn-1)$	$(ACr_0) \leftarrow (L)$, $(L) \leftarrow (ACr15)$, $(ACr_n) \leftarrow (ACr_{n-1})$	ROL r, m
Rotate Right (disp = -m)	010110	$(ACr_{15}) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$	$(ACr_{15}) \leftarrow (L)$, $(L) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$	ROR r, m
Shift Left (disp = m)	010111	$(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$	$(L) \leftarrow (ACr_{15})$, $(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$	SHL r, m
Shift Right (disp = -m)	010111	$(ACr_{15}) \leftarrow 0$, $(ACr_n) \leftarrow (ACr_{n+1})$	$(L) \leftarrow 0$, $(ACr_{15}) \leftarrow (L)$, $(ACr_n) \leftarrow (ACr_{n+1})$	SHR r, m

NOTE: For all shift and rotate instructions, "m" denotes the number of positions to be shifted or rotated, and is equal to the absolute value of disp.

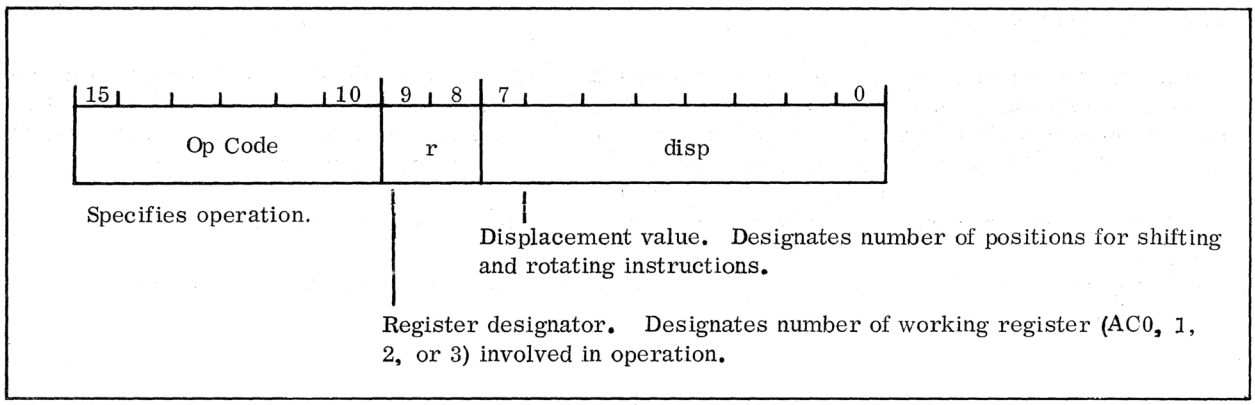
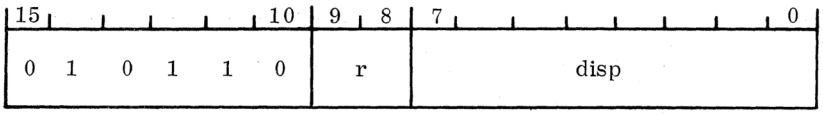


Figure 3-11. Shift Instruction Format

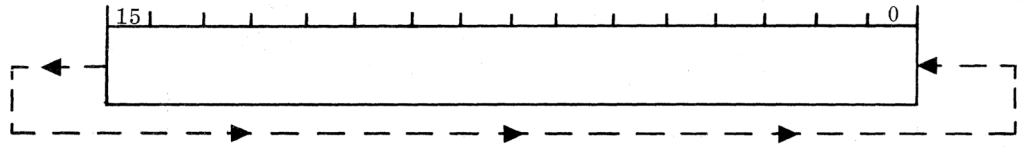
Rotate Left (ROL) (for disp > 0)



SEL = 0

Operation: $(ACr_0) \leftarrow (ACr_{15})$, $(ACr_n) \leftarrow (ACr_{n-1})$

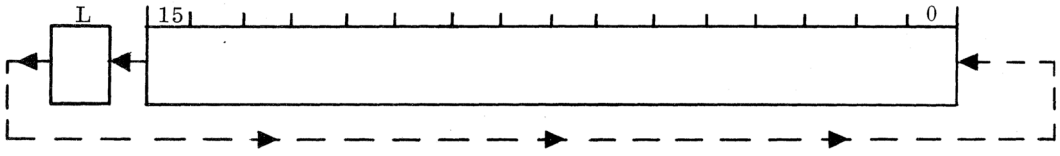
Description: The contents of ACr are shifted around to the left disp times. (ACr_{15}) replaces (ACr_0) for each shift.



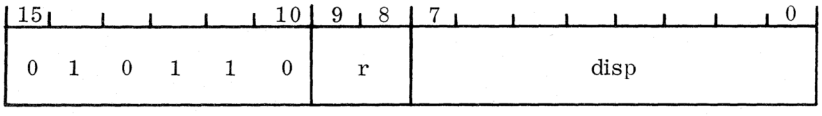
SEL = 1

Operation: $(ACr_0) \leftarrow (L)$, $(L) \leftarrow (ACr_{15})$, $(ACr_n) \leftarrow (ACr_{n-1})$

Description: The contents of ACr are shifted around to the left disp times. (L) replaces (ACr_0), and (ACr_{15}) replaces (L) for each shift.



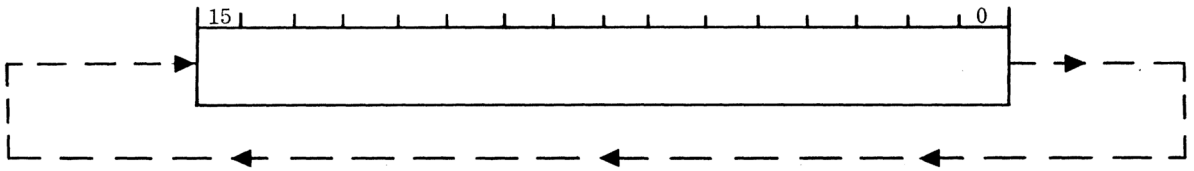
Rotate Right (ROR) (for disp <0)



SEL = 0

Operation: $(ACr_{15}) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$

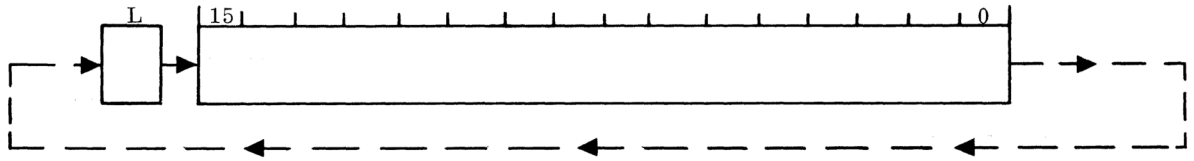
Description: The contents of ACr are shifted around to the right disp times. (ACr_0) replaces (ACr_{15}) for each shift.



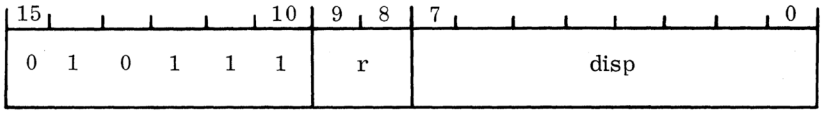
SEL = 1

Operation: $(ACr_{15}) \leftarrow (L)$, $(L) \leftarrow (ACr_0)$, $(ACr_n) \leftarrow (ACr_{n+1})$

Description: The contents of ACr are shifted around to the right disp times. (L) replaces (ACr_{15}), and (ACr_0) replaces (L) for each shift.



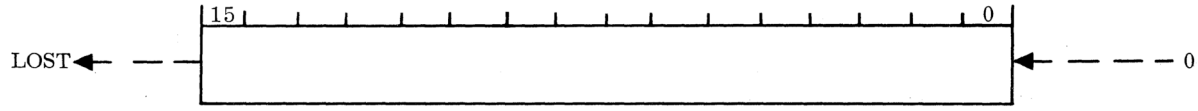
Shift Left (SHL) (for disp > 0)



SEL = 0

Operation: $(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$

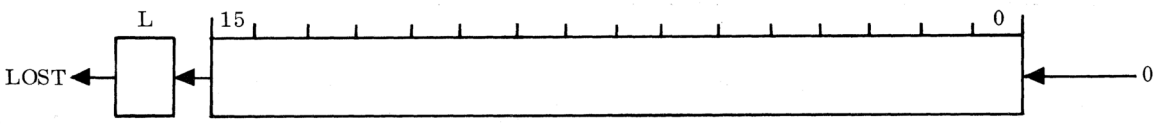
Description: The contents of ACr are shifted to the left disp times. (ACr_{15}) is lost, and zero replaces (ACr_0) for each shift.



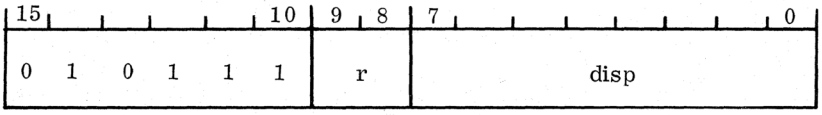
SEL = 1

Operation: $(L) \leftarrow (ACr_{15})$, $(ACr_n) \leftarrow (ACr_{n-1})$, $(ACr_0) \leftarrow 0$

Description: The contents of ACr are shifted to the left disp times. (L) is lost. (ACr_{15}) replaces (L), and zero replaces (ACr_0) for each shift.



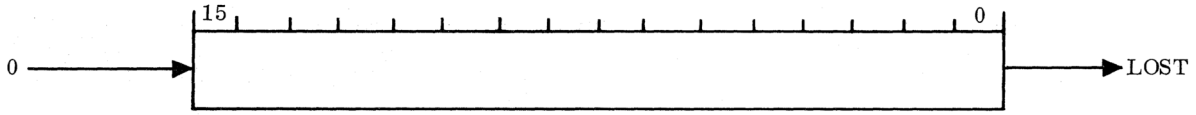
Shift Right (SHR) (for disp <0)



SEL = 0

Operation: $(ACr_{15}) \leftarrow 0$, $(ACr_n) \leftarrow (ACr_{n+1})$

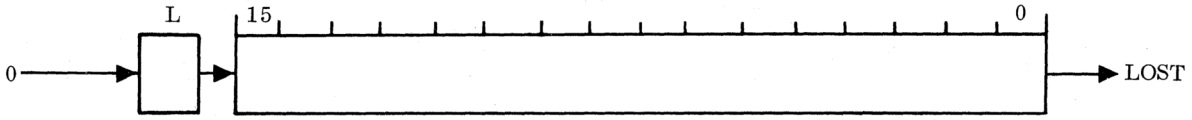
Description: The contents of ACr are shifted to the right disp times. (ACr_0) is lost, and zero replaces (ACr_{15}) for each shift.



SEL = 1

Operation: $(L) \leftarrow 0$, $(ACr_{15}) \leftarrow (L)$, $(ACr_n) \leftarrow (ACr_{n+1})$

Description: The contents of ACr are shifted to the right disp times. (ACr_0) is lost, (L) replaces ACr_{15}, and zero replaces (L) for each shift.



3.6.9 Register Instructions

There are eleven instructions in this group, summarized in table 3-12. Three word formats are required and are shown in figure 3-12.

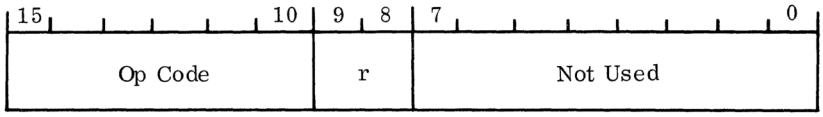
Table 3-12. Register Instructions

Instruction	Op Code	Operation	Assembler Format
Register and Stack:			
Push onto Stack	010000	(STK) ← (ACr)	PUSH r
Pull from Stack	010001	(ACr) ← (STK)	PULL r
Exchange Register and Stack	010101	(STK) ← (ACr), (ACr) ← (STK)	XCHRS r
Register and Immediate:			
Load Immediate	010011	(ACr) ← disp (sign extended)	LI r, disp
Add Immediate, Skip if Zero	010010	(ACr) ← (ACr) + disp (sign extended), OV, CY; if (ACr) = 0, (PC) ← (PC) + 1	AISZ r, disp
Complement and Add Immediate	010100	(ACr) ← ~ (ACr) + disp (sign extended)	CAI r, disp

Table 3-12. Register Instructions (Continued)

Instruction	Op Code			Operation	Assembler Format	
	OP1	OP2	OP3			
Register to Register:						
Register Add	0011	0	00	(ACdr) ← (ACsr) + (ACdr), OV, CY	RADD	sr,dr
Register Exchange	0011	1	00	(ACsr) ← (ACdr), (ACdr) ← (ACsr)	RXCH	sr,dr
Register Copy	0011	1	01	(ACdr) ← (ACsr)	RCPY	sr,dr
Register EXCLUSIVE-OR	0011	1	10	(ACdr) ← (ACsr) ▽ (ACdr)	RXOR	sr,dr
Register AND	0011	1	11	(ACdr) ← (ACsr) ∧ (ACdr)	RAND	sr,dr

Register and Stack Instructions

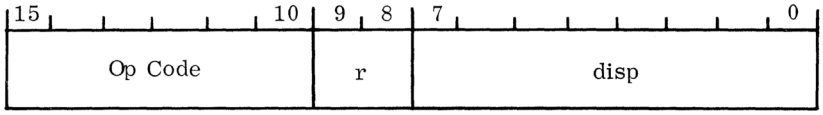


Specifies operation.

Not used. These bits are coded to zeros and are ignored for these instructions.

Register designator. Designates number of working register (AC0, 1, 2, or 3) involved in operation.

Register and Immediate Instructions



Specifies operation.

Displacement value. Designates immediate value, which is an operand in those instructions involving an immediate operand.

Register designator. Designates number of working register (AC0, 1, 2, or 3) involved in operation.

Figure 3-12. Register Instruction Formats (Sheet 1 of 2)

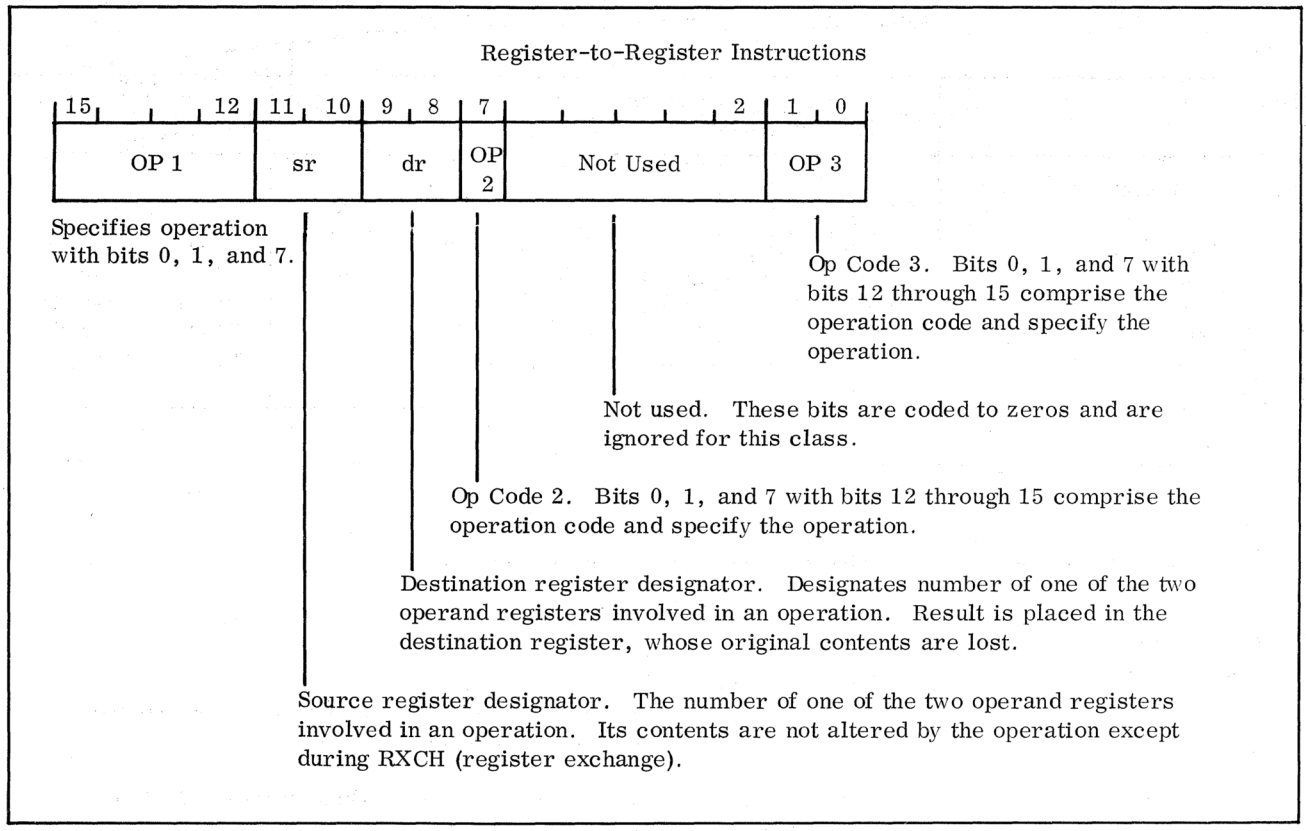
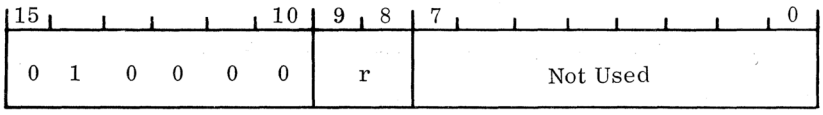


Figure 3-12. Register Instruction Formats (Sheet 2 of 2)

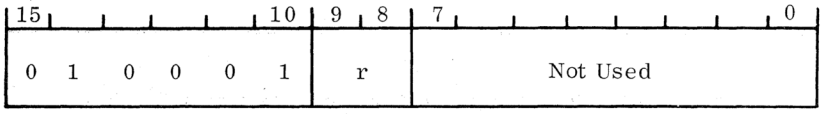
Push onto Stack (PUSH)



Operation: (STK) ← (ACr)

Description: The stack is pushed by the contents of the register (AC0, 1, 2, or 3) designated by r. Thus, the top of the stack then holds the contents of ACr, and the contents of all other levels in the stack are moved down one level. If the stack is full before the push occurs, the contents of the lowest level are lost. The initial contents of ACr are unaltered.

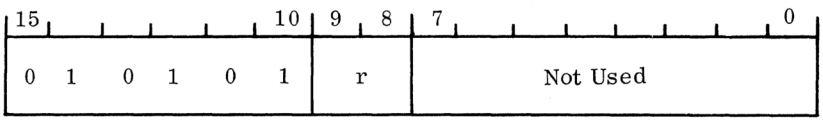
Pull from Stack (PULL)



Operation: (ACr) ← (STK)

Description: The stack is pulled. The contents from the top of the stack replace the contents of register number r (AC0, 1, 2, or 3). The initial contents of ACr are lost. The contents of each level of the stack moves up one level. Zeros enter the bottom of the stack.

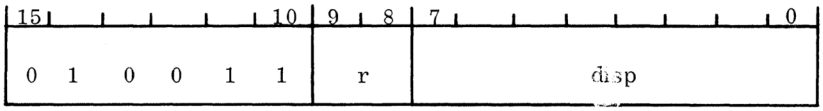
Exchange Register and Stack (XCHRS)



Operation: (STK) ← (ACr), (ACr) ← (STK)

Description: The contents of the top of the stack STK and the register designated by r (AC0, 1, 2, or 3) are exchanged.

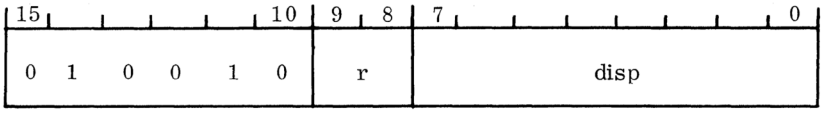
Load Immediate (LI)



Operation: (ACr) ← disp (sign extended)

Description: The value of disp with sign bit 7 extended through bit 15 replaces the contents of ACr (AC0, 1, 2, or 3). The initial contents of ACr are lost. The immediate operand range is -128 to +127.

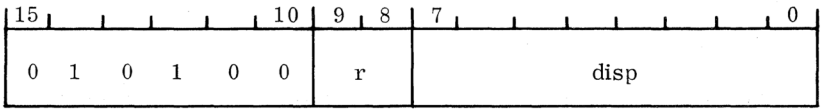
Add Immediate, Skip If Zero (AISZ)



Operation: (ACr) ← (ACr) + disp (sign extended), OV, CY
If new (ACr) = 0, (PC) ← (PC) + 1

Description: The contents of register ACr are replaced by the sum of the contents of ACr and disp (sign bit 7 extended through bit 15). The initial contents of ACr are lost. The Overflow and Carry Flags are set according to the result of the operation. If the new contents of ACr equal zero, the contents of PC are incremented by 1, thus skipping the next memory location. The immediate operand range is -128 to +127.

Complement and Add Immediate (CAI)



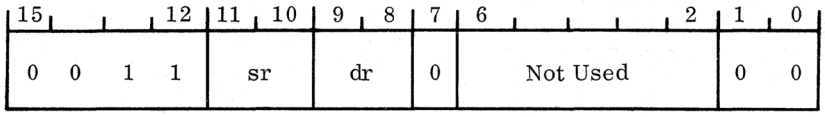
Operation: (ACr)← ~ (ACr) + disp (sign extended)

Description: The contents of register ACr are complemented and then added to disp (sign bit extended through bit 15). The result is then stored in ACr. The initial contents of ACr are lost. The immediate operand range is -128 to +127. Note that the Carry and Overflow Flags are not affected by this instruction.

NOTE

If disp = 0, the result is a ones complement.
If disp = 1, the result is a twos complement.

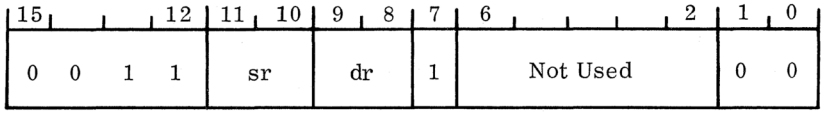
Register Add (RADD)



Operation: (ACdr) ← (ACsr) + (ACdr), OV, CY

Description: The contents of the destination register ACdr (AC0, 1, 2, or 3) are replaced by the sum of the contents of ACdr and the source register ACsr (AC0, 1, 2, or 3). The initial contents of ACdr are lost, and the contents of ACsr are unaltered. The Overflow and Carry Flags are set according to the result of the operation.

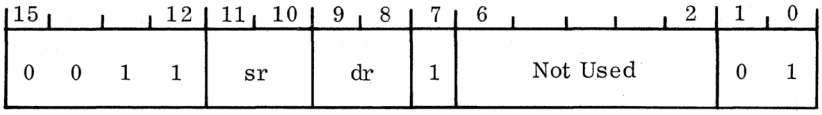
Register Exchange (RXCH)



Operation: (ACsr) ← (ACdr), (ACdr) ← (ACsr)

Description: The contents of ACsr (AC0, 1, 2, or 3) and ACdr (AC0, 1, 2, or 3) are exchanged.

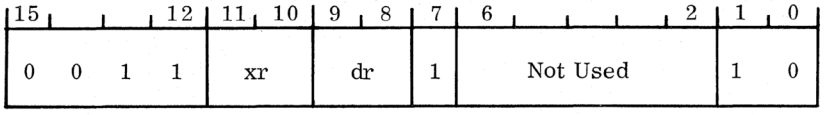
Register Copy (RCPY)



Operation: (ACdr) ← (ACsr)

Description: The contents of the destination register ACdr (AC0, 1, 2, or 3) are replaced by the contents of the source register ACsr (AC0, 1, 2, or 3). The initial contents of ACdr are lost, and the initial contents of ACsr are unaltered.

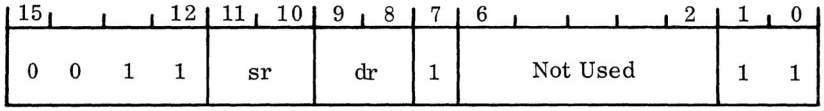
Register EXCLUSIVE OR (RXOR)



Operation: (ACdr) ← (ACdr) $\forall$ (ACsr)

Description: The contents of the destination register ACdr (AC0, 1, 2, or 3) are replaced by the result of exclusively ORing the contents of ACdr with the contents of the source register ACsr (AC0, 1, 2, or 3). The initial contents of ACdr are lost, and the initial contents of ACsr are unaltered.

Register AND (RAND)



Operation: (ACdr) ← (ACdr) ∧ (ACsr)

Description: The contents of the destination register ACdr (AC0, 1, 2, or 3) are replaced by the result of ANDing the contents of ACdr with the contents of the source register ACsr (AC0, 1, 2, or 3). The initial contents of ACdr are lost, and the initial contents of ACsr are unaltered.

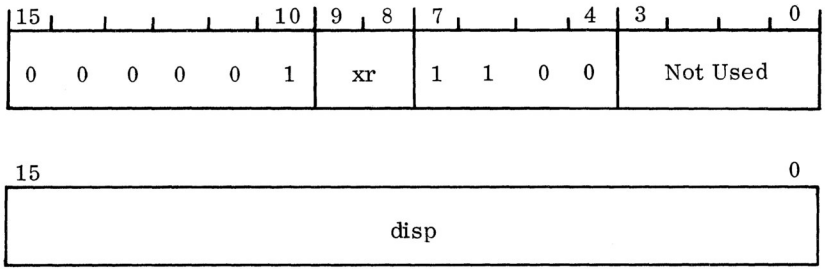
3.6.10 Byte Instructions

There are two instructions in this category; both are double-word, memory-reference instructions. They are summarized in table 3-13 and then individually described. Their word format is shown in figure 3-6 and is the same as the format for double-precision arithmetic instructions.

Table 3-13. Byte Instructions

Instruction	Op Code		Operation	Assembler Format
	OP1	OP2		
Load Byte	000001	1100	(Low-order byte of AC0) ←byte from (EA ÷ 2); SEL ← 0	LDB disp(xr)
Store Byte	000001	1101	Byte of (EA ÷ 2) ←(low-order byte of AC0); SEL ← 0	STB disp(xr)

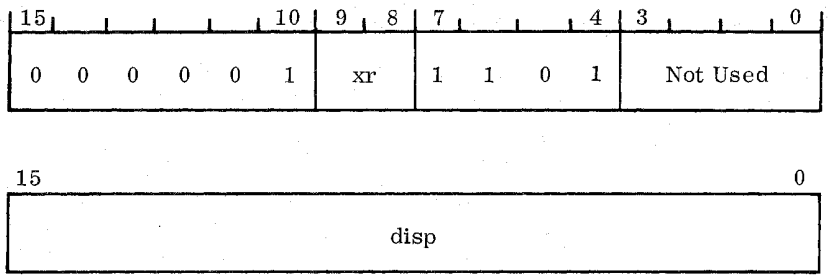
Load Byte (LDB)



Operation: Low-order byte of (AC0) ←byte from (EA ÷ 2); SEL ←0

Description: The low-order byte of AC0 is loaded with a byte from (EA ÷ 2). If the low-order bit of EA is 1, the low-order byte is loaded; otherwise, the high-order byte is loaded. (Note: EA ÷ 2 is the effective address shifted right one position.) The high-order byte of AC0 is set equal to zero. The Select Flag is cleared.

Store Byte (STB)



Operation: Byte of (EA ÷ 2) ← low-order byte from (AC0); SEL← 0

Description: The low-order byte of AC0 is stored into the byte of (EA ÷ 2) specified by the low-order bit of EA. If the low-order bit is 1, the low-order byte is specified; otherwise, the high-order byte is specified. (Note: EA ÷ 2 is the effective address shifted right one position.) The unspecified byte of EA ÷ 2 and the contents of AC0 are unaffected. The Select Flag is cleared.

NOTE

The effective address is formed by adding the contents of the index register to the displacement (EA = xr + disp). Byte addresses are formed by shifting this quantity right one bit position (EA ÷ 2 = [xr + disp] /2 = xr/2 + disp/2). Bit 0 of the EA specifies the left byte if equal to one or the right bit if equal to zero.

3.6.11 Input/Output, Halt, and Control Flag Instructions

Five instructions comprise this group, summarized in table 3-14. Two word formats are required and are shown in figure 3-13.

Table 3-14. Input/Output, Halt, and Flag Instructions

Instruction	Op Code	Operation	Assembler Format
Input/Output:			
Register In	000001000	(AR) ← ctl + (AC3); (AC0) ← (IOREG)	RIN ctl
Register Out	000001100	(AR) ← ctl + (AC3); (IOREG) ← (AC0)	ROUT ctl
Halt:			
Halt	000000000	Processor halts.	HALT

Instruction	Op Code		Operation	Assembler Format
	OP1	OP2		
Control Flags:				
Set Flag	00001	0	fc set; (AR) ← ctl	SFLG fc
Pulse Flag	00001	1	fc pulsed; (AR) ← ctl	PFLG fc

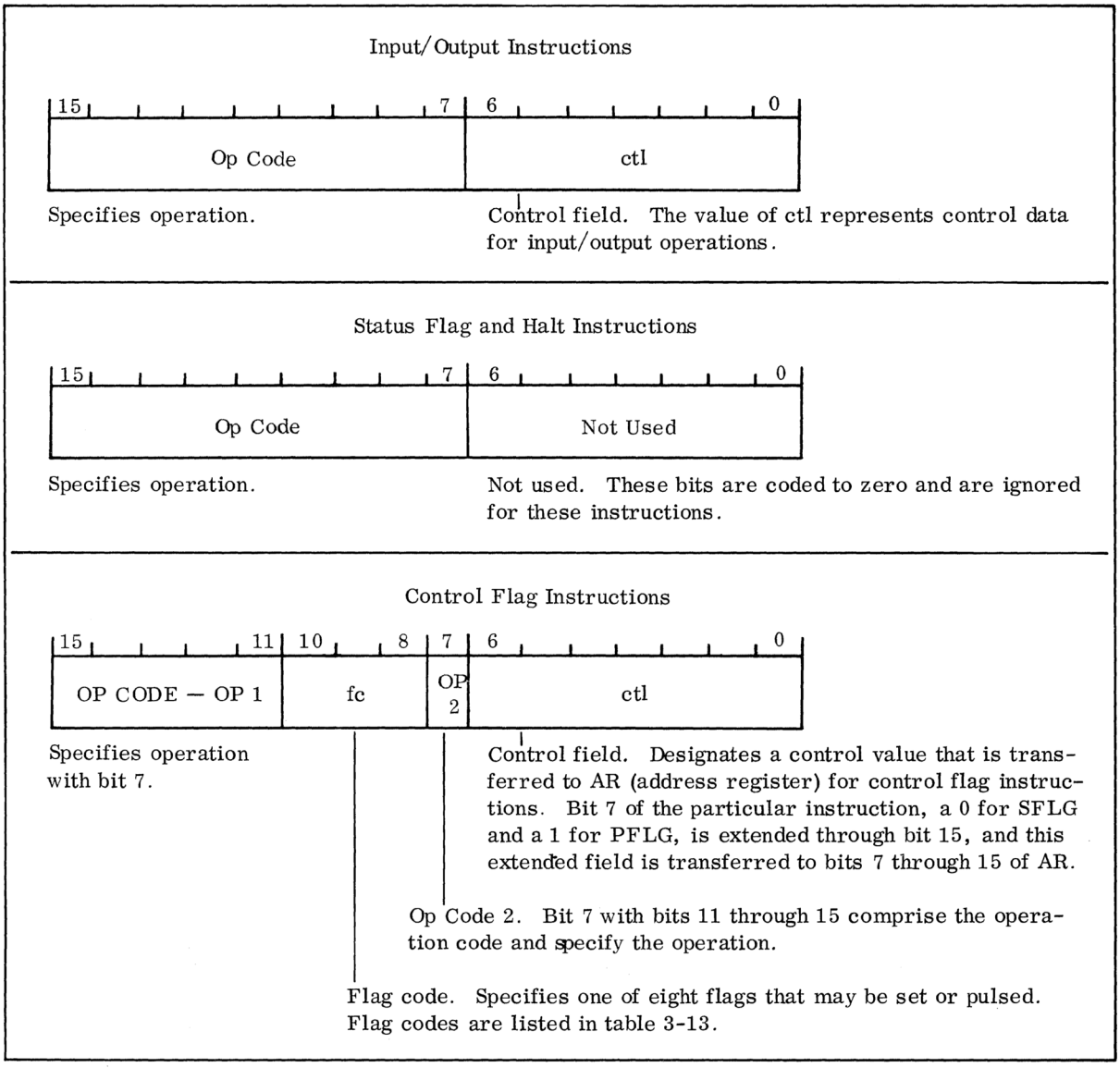
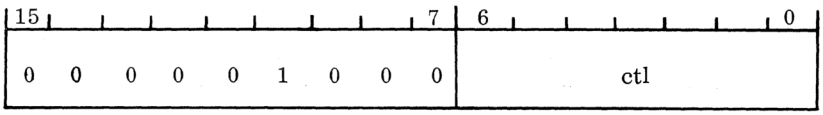


Figure 3-13. Input/Output, Halt, and Control Flag Instruction Formats

Register In (RIN)



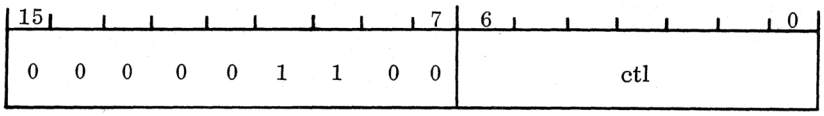
Operation: (AR) ←ctl + (AC3), (AC0) ← (IOREG)

Description: The contents of AR (address register) are replaced by the sum of ctl and the contents of AC3. The new contents of AR constitute the address of a peripheral device and a command, both of which are received by the addressed peripheral device. The peripheral device responds by transferring the contents of its input/output register (IOREG) to the processor AC0.

NOTE

ctl is a 7-bit unsigned integer.

Register Out (ROUT)



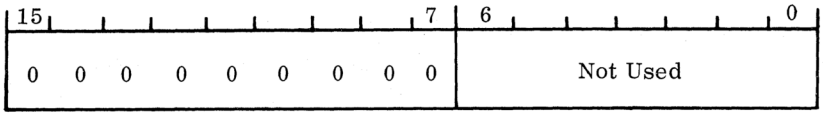
Operation: (AR) ← ctl + (AC3), (IOREG) ← (AC0)

Description: The contents of AR (address register) are replaced by the sum of ctl and the contents of AC3. The new contents of AR constitute the address of a peripheral device and a command, both of which are received by the addressed peripheral device. The processor then transfers the contents of AC0 to IOREG in the peripheral device.

NOTE

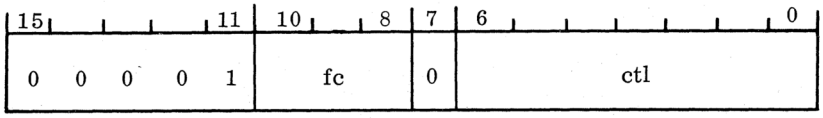
ctl is a 7-bit unsigned integer.

Halt (HALT)



Description: The processor halts and remains halted until the RUN Button is pushed.

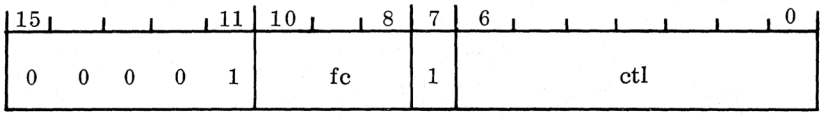
Set Flag (SFLG)



Operation: FC set, (AR) ← ctl (bit 7 extended through bit 15; that is, bits 8 through 15 set to 0)

Description: The control flag designated by the flag code FC is set. The contents of the address register AR are replaced by the value of ctl. Flag codes are defined in table 3-16.

Pulse Flag (PFLG)



Operation: FC pulsed, (AR) ← ctl (bit 7 extended through bit 15; that is, bits 8 through 15 set to 1)

Description: The control flag designated by the flag code FC is pulsed (notes 2 and 4 below). The contents of the address register AR are replaced by the value of ctl. Flag codes are defined in table 3-15.

NOTE

(1) SFLG and PFLG refer to control flags external to the RALUs. These flags should not be confused with the RALU-internal flags, which are referenced by PUSHF, PULLF, SETST, CLRST and SKSTF.

(2) Pulsing a control flag sets the flag at T2 and resets it at T6 during the same microcycle. The flag remains reset until again set or pulsed.

(3) The ctl value plus the extended bit 7 through bit 15 are transferred to the AR (address register). This word in the AR has no specified use and may be used as desired by the system programmer.

(4) If the referenced flag is set initially, a PFLG instruction causes it to be reset.

Table 3-15. Control Flag Codes

FC	Flag Mnemonic	Significance
000	F8	User Specified
001	INT EN	Interrupt Enable
010	SEL	Select
011	F11	User Specified
100	F12	User Specified
101	F13	SCPIE
110	F14	User Specified
111	F15	User Specified

NOTE: (1) The flag designated by the flag code (FC) is set or pulsed. Only control flags (shown in Flag Mnemonic column) with addresses between 8 and 15 may be accessed with these instructions. (The flag address is 8, binary 1000, greater than the corresponding flag code, FC.) This is done because control flags with flag addresses 0 through 7 are used for various input/output operations controlled by the processor, and are not usable by the programmer.

(2) SCPIE denotes Special Control Panel Interrupt Enable.

3.6.12 Bit and Status Flag Instructions

There are seven single-word instructions in this group. They are summarized in table 3-16 and then described individually. Their word formats are shown in figure 3-14.

Table 3-16. Bit and Status Flag Single-word Instructions

Instruction	Op Code 2	Operation	Assembler Format
Set Status Flag	1110000	Status Flag n ← 1; SEL ← 0	SETST n
Clear Status Flag	1110001	Status Flag n ← 0; SEL ← 0	CLRST n
Set Bit	1110010	$AC0_n$ ← 1; SEL ← 0	SETBIT n
Clear Bit	1110011	$AC0_n$ ← 0; SEL ← 0	CLRBIT n
Complement Bit	1110110	$AC0_n$ ← ~ $AC0_n$; SEL ← 0	CMPBIT n

Instruction	Op Code	Operation	Assembler Format
Push Status Flags Onto Stack	000000001	(STK) ← (STATUS FLAGS)	PUSHF
Pull Status Flags From Stack	000000101	(STATUS FLAGS) ← (STK)	PULLF

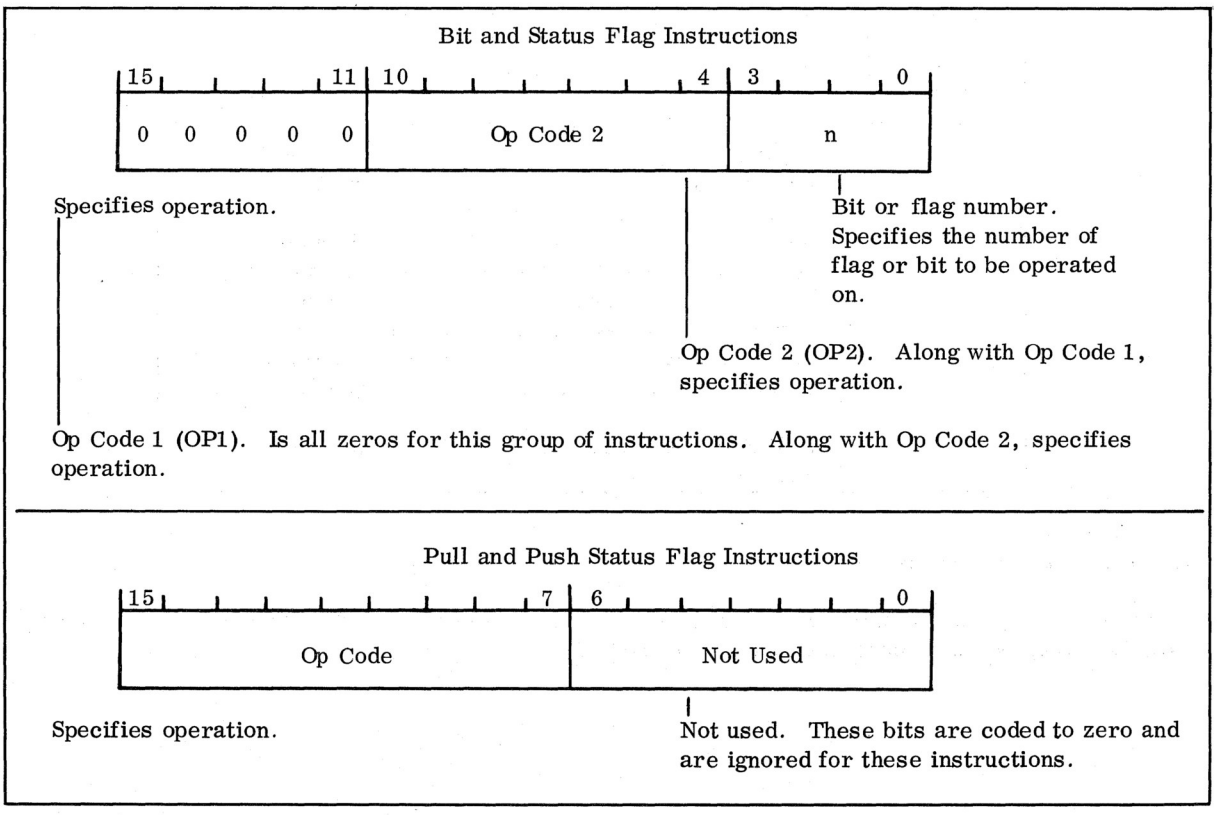
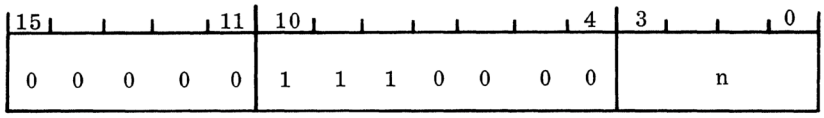


Figure 3-14. Bit and Status Flag Instruction Formats

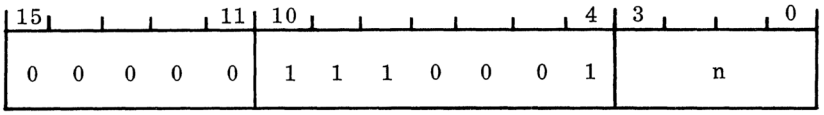
Set Status Flag (SETST)



Operation: Status Flag n ← 1; SEL ← 0

Description: Bit n of the Status Flag Register is set true. All other bits are unaffected. The Select Flag is cleared. ($0 \leq n \leq 15$)

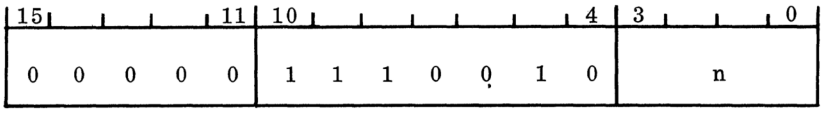
Clear Status Flag (CLRST)



Operation: Status Flag n ← 0; SEL ← 0

Description: Bit n of the Status Flag Register is cleared. All other bits are unaffected. The Select Flag is cleared. ($0 \leq n \leq 15$)

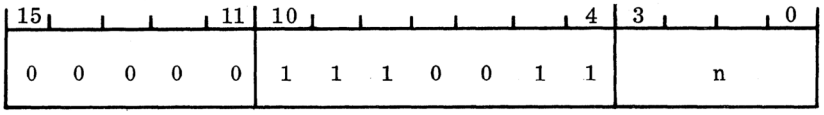
Set Bit (SETBIT)



Operation: $AC0_n \leftarrow 1$, SEL ← 0

Description: Bit n of AC0 is set true. All other bits are unaffected. The Select Flag is cleared. ($0 \leq n \leq 15$)

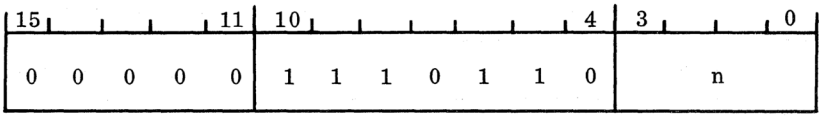
Clear Bit (CLRBIT)



Operation: $AC0_n \leftarrow 0$; SEL ← 0

Description: Bit n of AC0 is cleared. All other bits are unaffected. The Select Flag is cleared. ($0 \leq n \leq 15$)

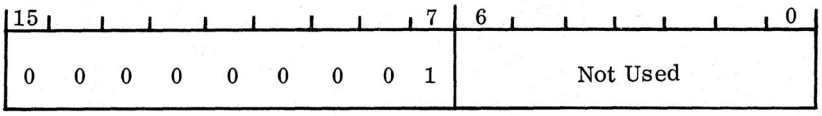
Complement Bit (CMPBIT)



Operation: $AC0_n \leftarrow \sim AC0_n$; SEL ← 0

Description: Bit n of AC0 is complemented. All other bits are unaffected. The Select Flag is cleared. ($0 \leq n \leq 15$)

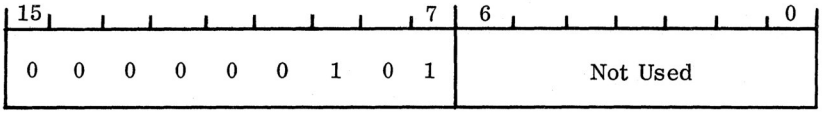
Push Status Flags onto Stack (PUSHF)



Operation: (STK) ← (STATUS FLAGS)

Description: The contents of the top of the stack STK are replaced by the contents of the Status Flags. See figure 3-15 for the configuration of processor flags on the stack and table 3-17 for processor flag definitions. The previous contents of the top of the stack and lower levels are pushed down one level. The contents of the lowest level of the stack are lost.

Pull Status Flags from Stack (PULLF)



Operation: (STATUS FLAGS) ← (STK)

Description: The contents of the Status Flags are replaced by the contents of the top of the stack (STK). See figure 3-15 for the configuration of processor flags on the stack and table 3-17 for processor-flag definitions. The previous contents of lower levels of the stack are pulled up by one level with zeros replacing the contents of the lowest level.

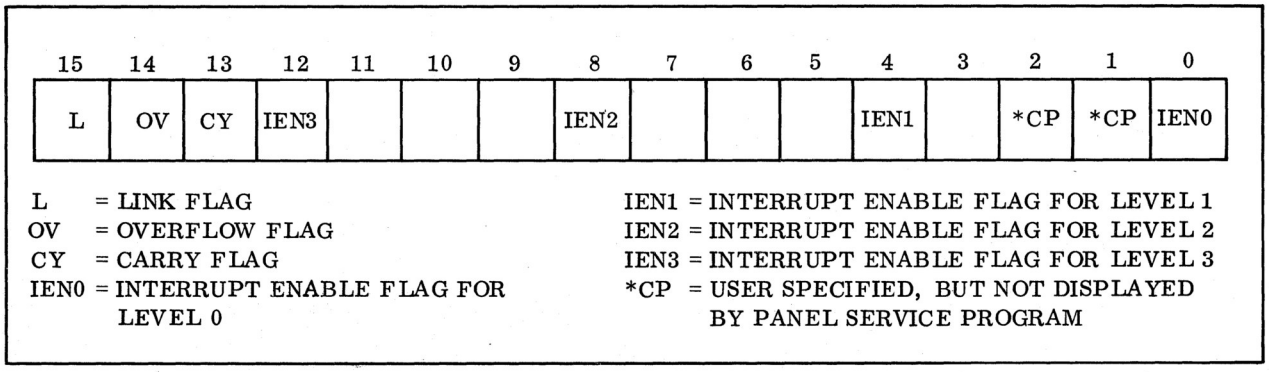


Figure 3-15. Configuration of Status Flags

Table 3-17. Status Flags

Bit Position	Flag Name	Mnemonic	Significance
15	Link	L	Used for double-word shifts
14	Overflow	OV	Set if an arithmetic overflow occurs
13	Carry	CY	Set if a carry occurs (from most significant bit) during an arithmetic operation
12		IEN 3	Interrupt Enable Flag for Level 3

Table 3-17. Status Flags (Continued)

Bit Position	Flag Name	Mnemonic	Significance
11 thru 9	Gen. Purp. Flags	GF	Use Specified by Programmer
8		IEN 2	Interrupt Enable Flag for Level 2
7 thru 5	Gen. Purp. Flags	GF	Use Specified by Programmer
4		IEN 1	Interrupt Enable Flag for Level 1
3	Gen. Purp. Flag	GF	Use Specified by Programmer
2	Gen. Purp. Flag	*CP	User Specified — not displayed
1	Gen. Purp. Flag	*CP	User Specified — not displayed
0		IEN 0	Interrupt Enable Flag for Level 0

Chapter 4

CONTROL PANEL OPERATION

4.1 GENERAL

The control panel consists of two sections: the Operators Control Panel, which is a basic part of the machine, and the Programmers Control Panel, which is removable. The Operators Control Panel contains a minimum number of controls for general operational purposes. The Programmers Control Panel permits direct access to the CPU registers and memory. The control panel functions require the use of the first backplane card slot. The control panel logic card occupies this first slot and has two cables attached: one for the Operators Control Panel and the other for the Programmers Control Panel.

NOTE

X' prefix indicates hexadecimal notation and is used interchangeably with subscript notation (for example, $A1BC_{16}$ = X'A1BC).

4.2 OPERATORS CONTROL PANEL

The Operators Control Panel, shown in figure 4-1, provides the basic controls necessary to load and execute programs. The controls are as follows:

- POWER – A two-position keyswitch used to prevent unauthorized application or removal of system power. In the OFF position, the system power is disconnected. In the ON position, the system power is connected. Turning the keyswitch to the ON position clears all CPU registers and interface control flags. After clearing, the microprocessor halts with the Program Counter (PC) set equal to X'FFFE. The key is removable from either position.

- PANEL – A two-position keyswitch used to restrict "destructive" system access. In the UNLOCK position, all panel functions may be used. In the LOCK position, only nondestructive functions may be used. In the Operators Control Panel, the only nondestructive function is RUN. The disabled functions of the Programmers Control Panel are described in paragraph 4.3. The key is removable from either position.

- AUX1 – A pushbutton switch wired to the first backplane card slot, available for use as a card reader bootstrap switch or other user function.

- AUX2 – A pushbutton switch wired to the first backplane card slot, available for use as a disc bootstrap switch or other user function.

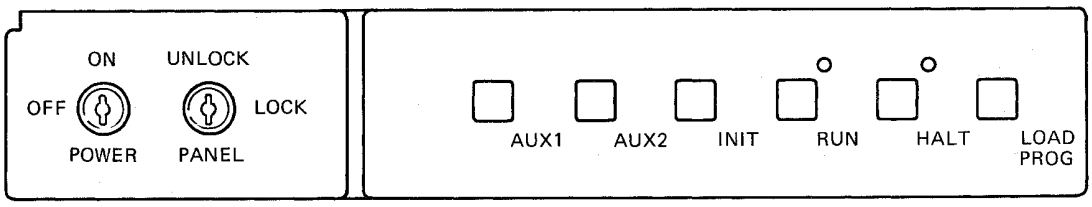


Figure 4-1. Operators Control Panel

- INIT — A pushbutton switch which halts the processor, clears all CPU registers and interface control flags, and sets the Program Counter (PC) equal to X'FFFE. The functions performed are identical to powering ON.

- RUN — A pushbutton switch which causes program execution to begin at the location specified by the contents of the Program Counter (PC). The RUN Pushbutton Lamp remains lit until the HALT Pushbutton is activated or a Halt Instruction is executed.

- HALT — A pushbutton switch which causes program execution to be terminated at the end of the current instruction. It has the same effect as execution of the Halt Instruction. The HALT Pushbutton Lamp is lit whenever a halt occurs.

- LOAD PROG — A pushbutton switch which forces the program to branch to an absolute paper tape loader program stored in ROM. Use of this pushbutton causes the 16-word pushdown stack to be cleared and the accumulators to be altered.

4.3 PROGRAMMERS CONTROL PANEL

The Programmers Control Panel, shown in figure 4-2, provides complete program debug facilities in addition to the normal operational features. It is a self-contained unit which may be used interchangeably among several systems. This control panel interfaces with the system via the control panel interface logic in the same manner as a peripheral device, and communication is programmed by data transfer to and from the CPU. The program for controlling the Programmers Control Panel is stored in read-only memory (ROM), occupying locations X'FF00 through X'FFFF, and requires the use of memory locations X'101 through X'117 in read/write memory (described in chapter 5). Communication is over a 50-wire cable from the control panel logic card, located in the first backplane card slot. A description of the panel controls and indicators follows:

- PROGRAM COUNTER/MEMORY ADDRESS — Sixteen indicator lamps normally indicate the current value of the Program Counter. However, when the Display Selector Switch (described below) is in the MEMory DATA position, these lamps indicate the current memory address being inspected.

- DATA DISPLAY — Sixteen indicator lamps used for displaying the information specified by the position of the Display Selector Switch (the bit assignments of the internal flags are indicated when the FLAGS position is selected).

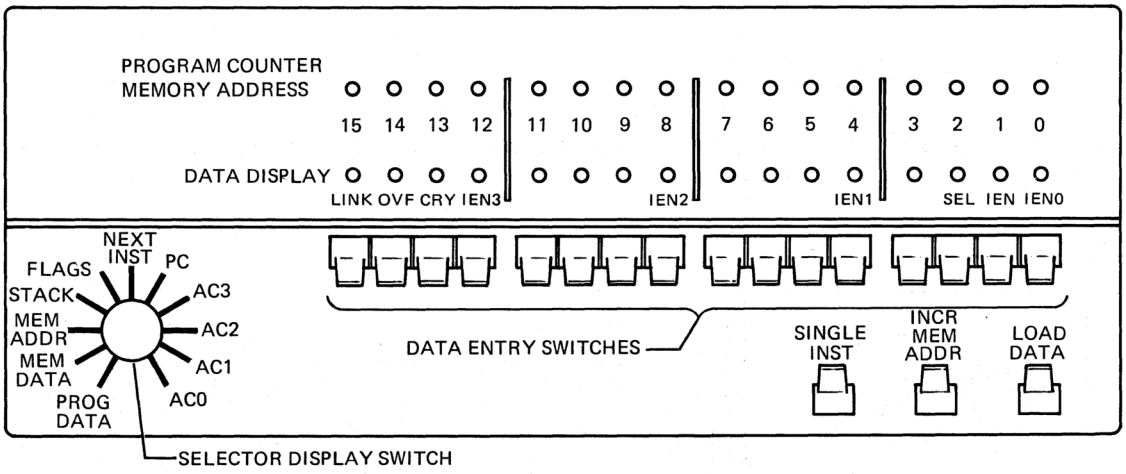


Figure 4-2. Programmers Control Panel

- Data Entry — Sixteen two-position toggle switches used for loading data into the register (or location) specified by the Display Selector Switch when the LOAD DATA Pushbutton Switch is activated. Switch positions are "up" for a logic 1, "down" for a logic zero.

- SINGLE INST — Pressing this switch causes the instruction at the location specified by the contents of the Program Counter to be executed. If the processor is running when this switch is pressed, execution halts after completion of the current instruction (as in the case of the HALT Switch on the Operators Control Panel). The SINGLE INST Switch is disabled by the LOCK Keyswitch.

- INCR MEM ADDR — Pressing this switch instructs the Control Panel Service Routine to add one to the present value of the memory address pointer (described with the Display Selector Switch, below). The PANEL Keyswitch LOCK position has no effect on this switch.

- LOAD DATA — When this switch is pressed, the Control Panel Service Routine takes the value indicated by the Data Entry Switches and stores this value into the location specified by the Display Selector Switch. The LOAD DATA Switch is disabled by the panel LOCK Keyswitch for all Display Selector Switch positions except MEMory ADDRess. Data may be loaded in any mode when the processor is running or halted.

- Display Selector — Eleven-position rotary switch. Used to select the location for display or data entry functions. The selected location is displayed continuously on the 16 DATA DISPLAY Indicators during Halt and Execute Modes. During Execute Mode, the display is refreshed 10 times per second by a hardware interrupt transparent to the user. Individual switch positions permit the following functions:

  - AC0, AC1, AC2, AC3 — These four switch positions permit the contents of the CPU accumulators to be displayed on the panel.

  - PC — This switch position displays the contents of the Program Counter (which points to the next instruction).

  - NEXT INST — This switch position displays the next instruction to be executed (which is the contents of the memory location specified by PC).

  - FLAGS — This switch position permits the CPU internal status flags to be displayed in the bit positions indicated by the 16 DATA DISPLAY Indicators. In addition, the external Interrupt Enable and Select Flags are tested by the Control Panel Service Routine, and their states are displayed in bit positions 2 and 3. (The actual contents of flag bits 2 and 3 are unaffected, but are not displayed; however, they may be examined in memory location X'115.)

  - STACK — This switch position permits the top word of the stack to be displayed. (Other locations in the stack may be examined, or loaded, by examining memory locations X'105 through X'114 which are loaded with stack words 0 through 14 by the Control Panel Service Routine. Word 15 is pushed off the stack and is not saved.)

  - MEM ADDR — This switch position permits the contents of the memory address pointer in the Control Panel Service Routine to be displayed. (This pointer is not altered by program execution and is not the same as the Memory Address Register [MAR] in the CPU.)

  - MEM DATA — This switch position permits the contents of the memory word specified by the memory address pointer to be displayed. (In this position, the upper 16 indicators contain the memory address instead of the contents of the Program Counter.)

  - PROG DATA — This switch position disables the periodic interrupt for the Control Panel Service Routine when the processor is running. When halted, and in the Program Data Mode, the firmware tests the switches for any depressions but does not alter the display. (This leaves the display under control of the program currently being executed, which can write into either display register as a peripheral device. With the switch in this position, the control panel refresh interrupt, which occurs in all other positions, is inhibited. Therefore, the fastest program execution occurs.)

4.4 CONTROL PANEL INTERFACE HARDWARE

The IMP-16L Programmers Control Panel is treated as a peripheral device which is serviced via the System Data Bus. The Control Panel Service Routine resides in read-only memory and interprets and executes control panel commands in a manner which is transparent to the user's program.

As a peripheral device, the Programmers Control Panel is accessible to the user and may be used as an input/output device in appropriate applications. Figure 4-3 is a simplified block diagram of the Programmers Control Panel showing the control registers as peripheral devices. The PC Display Register may be used for program output display by using the ROUT Instruction with a peripheral address X'EC. The contents of a Selected Data Display Register may be written by using the ROUT Instruction with a peripheral device address of X'ED. (When the user writes to either display row, it is essential that the Display Selector Switch be set to the PROG DATA position or the data will be overwritten with the contents of the selected display by the firmware program.) The Data Entry Switch settings may be read by using the RIN Instruction with a peripheral device address of X'EE. The Display Selector Switch and functional switches command word may be read using the RIN Instruction with a peripheral address of X'EF. (See chapter 5 for bit positions of the command word.)

The Operators Control Panel is under hardware control except for the LOAD PROG Pushbutton which is sampled by the Control Panel Service Routine.

4.5 OPERATING PROCEDURES

The basic control panel operating procedures are as follows:

- Power Up — To power up the system, turn the POWER Keyswitch ON. The system comes up in the Halt Mode with all registers, control flags, and status flags cleared. A system clear pulse is issued to all peripherals. The Program Counter is set to location X'FFFE.
- Power Down — To remove power from the system, turn the POWER Keyswitch to OFF.
- Program Loading — Programs may be loaded into memory using the absolute paper tape loader program contained in ROM by pressing the LOAD PROG Pushbutton, on the Operators Control Panel. (Use of the Control Panel Service Routine is described in chapter 5.)
- Program Execution — If the Program Counter is not set to the starting address, use the LOAD DATA Pushbutton to set the value of the Program Counter. If it is desired to suppress control panel refresh, set the Display Selector Switch to PROG DATA position; then press the RUN Pushbutton.
- Debug — The Programmers Control Panel may be used to examine and modify CPU registers and memory contents in both Execute and Halt Modes as explained in paragraph 4.3. This provides convenience for program debugging and may be used to a limited extent for debugging hardware.

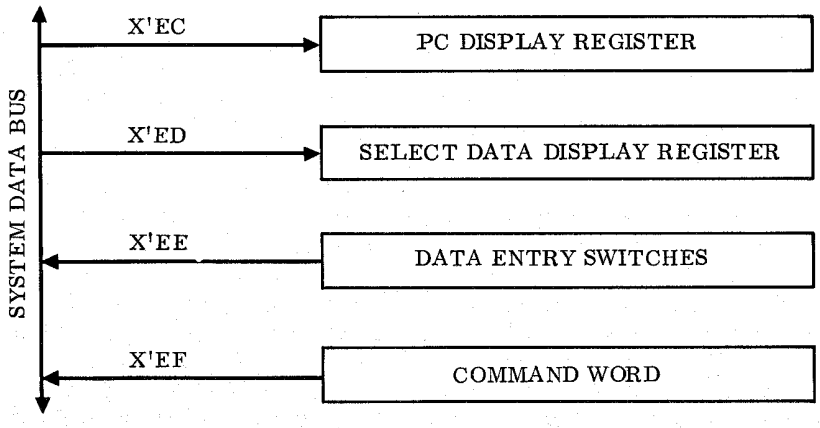


Figure 4-3. Simplified Block Diagram of Programmers Control Panel

Chapter 5

FIRMWARE

5.1 INTRODUCTION

This chapter describes the programs contained in the IMP-16L Basic Firmware Package referred to as BFIRL in the software documentation.

A fundamental group of firmware programs is provided in Read-only Memory (ROM) of the Memory Module to enhance communication with the IMP-16L. These programs include a Teletype Absolute Paper Tape Loader Program, a Control Panel Service Routine, a Teletype Character Receive Routine, and a Teletype Bit Delay Routine.

These programs reside in 256 of the 512 words of read-only memory space provided by the Memory Module and occupy the address space between X'FF00 and X'FFFF inclusive. These programs are described in the following paragraphs.

5.2 TELETYPE ABSOLUTE PAPER TAPE LOADER PROGRAM

A Teletype Absolute Paper Tape Loader Program is resident in read-only memory. This loader is a stand-alone program that reads and loads into main memory (RAM) for execution of an absolute Relocatable Load Module (RLM) produced by the IMP-16 assembler. The program loads any RLM that has the absolute memory addresses assigned at assembly time.

The Absolute Paper Tape Loader Program is the ABSPT program described in the IMP-16 Utilities Reference Manual (NSC Order Number IMP-16S/025Y). Paper tapes read by this program contain eight channels of binary data in standard RLM format. This format is defined in appendix A of the IMP-16 Programming Assembler Manual (NSC Order No. IMP-16S/102Y).

The RLM paper tape is composed of successive RLM records, each preceded by a Start-of-transmission (STX) character. Since each record contains its own length, no extra characters may appear within records; but any character may appear between records.

5.2.1 Paper Tape Program Loading Procedure

The procedure for loading paper tape is as follows:

1. Place the RLM paper tape into the Paper Tape Reader on the Teletype.
2. Press the LOAD PROG Button on the IMP-16L control panel.
3. Turn on the Paper Tape Reader.

The RLM is loaded, the entry point address is saved in Accumulator AC2, and the IMP-16L is halted. At this point, the user may do one of the following:

1. Press RUN Pushbutton to cause execution of the program just loaded (executes a Jump Instruction indexed through AC2).
2. Alter the entry point address contained in AC2 and press RUN to cause execution to commence at the modified entry point.
3. Load another RLM in the same manner as described above.

If the loader detects a checksum error, it halts the machine. The user, noting that the paper tape has not completed loading, may restart the loading procedure by positioning the paper tape at the beginning of the bad checksum record, pressing RUN Pushbutton, and turning on the reader.

NOTE

This procedure applies only to Paper Tape Readers equipped with a reader relay option. In systems not having this option, only the computer halts and the user must determine the incorrect record by observation.

5.2.2 Program Loader Listing

Table 5-1 is a complete listing of the Teletype Absolute Loader Routine.

Table 5-1. Teletype Absolute Loader Routine

```
489 FF94                        .PAGE   ' TTY ABSOLUTE LOADER  6-7-73'
490 FF94          .LOCAL
491 FF94
492 FF94          ;       LOADS RLM FROM 8 CHANNEL PAPER TAPE
493 FF94          ;
494 FF94          ;       EACH RECORD MUST BE PRECEDED BY A STX CHARACTER
495 FF94          ;
496 FF94 612A A TORS:   AND     0,H3FFF         ; TITLE OR SYMBOL RECORD
497 FF95 3181 A         RCPY    0,1             ; PROCESSING
498 FF96 2930 A         JSR     RDWD            ; SLOUGH RECORD BODY
499 FF97 49FF A         AISZ    1,-1            ; AND THE CKSUM WORD
500 FF98 21FD A         JMP     .-2
501 FF99 292D A         JSR     RDWD
502 FF9A                                        ;
503 FF9A          RLOADER:
504 FF9A 4F10 A         LI      3,16            ; CLEAR THE STACK SO IT
505 FF9B 4400 A         PULL    0               ; DOESN'T OVERFLOW WHEN
506 FF9C 4BFF A         AISZ    3,-1            ; THE OPERATOR PUSHES
507 FF9D 21FD A         JMP     .-2             ; LOAD PROG MORE THAN
508 FF9E                                        ; ONCE
509 FF9E 2935 A $0:     JSR     RTTYSR
510 FF9F 48FE A         AISZ 0,-2               ; LOOK FOR STX
511 FFA0 21FD A         JMP     $0
512 FFA1 2925 A TTY1:   JSR     RDWD            ; PROCESS RECORD CONTROL
513 FFA2                                        ; INFORMATION
514 FFA2 12F1 A         BOC     2,TORS          ; BRANCH IF R0(15) IS 0
515 FFA3                                        ; GO TO TITLE OR SYMBOL
516 FFA3                                        ; RECORD PROCESSING
517 FFA3 5C01 A         SHL     0,1             ; BIT 15 IS A 1.
518 FFA4 1201 A         BOC     2,.+2           ; BRANCH IF R0(14) IS 0 TO
519 FFA5                                        ; DATA RECORD
520 FFA5 211A A         JMP     ENDREC          ; OTHERWISE GO TO END
521 FFA6                                        ; RECORD PROCESSING
```

SOURCE STATEMENTS

RELOCATION CHARACTER: A - absolute, X - external, B - base, T - top, and I - indirect

INSTRUCTION OR DATA

MEMORY LOCATION

SOURCE LINE NUMBER

Table 5-1. Teletype Absolute Loader Routine (Continued)

```
522 FFA6                                                    ;
523 FFA6                                                    ; DATA RECORD PROCESSING
524 FFA6 5CFF A          SHR     0,1                        ;
525 FFA7 3381 A          RCPY    0,3                        ; R3 HAS THE RECORD BODY
526 FFA8                                                    ; LENGTH
527 FFA8 48FC A          AISZ    3,-4
528 FFA9 291D A          JSR     RDWD
529 FFAA 3181 A          RCPY    0,1                        ; R1 HAS CKSUM
530 FFAB 4000 A          PUSH    0                          ; SAVE CHECKSUM WORD
531 FFAC 5101 A          CAI     1,1                        ; R1 HAS -CKSUM MODE WORD
532 FFAD 29CE A          JSR     RDWDCK                     ; SLOUGH ADDRESS MODE
533 FFAE 290D A          JSR     RDWDCK                     ; GET THE INITIAL LOAD
534 FFAF                                                    ; ADDRESS AND COPY IT
535 FFAF 3281 A          RCPY    0,2                        ; INTO R2.
536 FFB0 2908 A          JSR     RDWDCK                     ; SLOUGH THE RELOCATION
537 FFB1 290A A          JSR     RDWDCK                     ; MODE WORDS
538 FFB2 2909 A TTY2:     JSR     RDWDCK                     ; GET WORD
539 FFB3 A200 A          ST      0,(2)
540 FFB4 4401 A          AISZ    2,1
541 FFB5 4BFF A          AISZ    3,-1
542 FFB6 21FB A          JMP     TTY2
543 FFB7 4400 A TCKSUM:   PULL    0                          ; GET CHECKSUM WORD
544 FFB8 11E5 A          BOC     REQ0,$0                    ; IF THE CHECKSUM WORD
545 FFB9                                                    ; IS 0 THEN DON'T TEST
546 FFB9                                                    ; THE ERROR DETECTION
547 FFB9                                                    ; WORD IN R1.
548 FFB9                                                    ;
549 FFB9 4900 A          AISZ    1,0                        ; IF THE ERROR DETECTION
550 FFBA 0000 A          HALT                               ; WORD IS 0 THEN NO ERROR
551 FFBB 21E2 A          JMP     $0                         ; IS DETECTED. IF IT IS
552 FFBC                                                    ; NOT 0, A CHECKSUM ERROR
553 FFBC                                                    ; IS DETECTED AND THE
554 FFBC                                                    ; MACHINE HALTS.
555 FFBC 290A A RDWDCK:   JSR     RDWD                       ; READ WORD AND ADD IT
556 FFBD 3100 A          RADD    0,1                        ; TO THE CHECKSUM.
557 FFBE 0200 A          RTS
558 FFBF 0001 A REQ0      =       1
559 FFBF 3FFF A H3FFF:    .WORD   X'3FFF
560 FFC0                                                    ;
561 FFC0 2906 A ENDREC:   JSR     RDWD                       ; SLOUGH CHECKSUM
562 FFC1 2905 A          JSR     RDWD                       ; SLOUGH ENTRY ADDRESS
563 FFC2                                                    ; MODE
564 FFC2 2904 A          JSR     RDWD                       ; GET ENTRY ADDRESS
565 FFC3 3281 A          RCPY    0,2                        ; PUT EA INTO R2
566 FFC4 4F00 A          LI      3,0                        ; LOAD DEVICE IS TTY
567 FFC5 0000 A          HALT
568 FFC6 2200 A          JMP     (2)
569 FFC7          ;
570 FFC7          ;
571 FFC7          ;     READ WORD SUBROUTINE. READS TWO CHARACTERS FROM THE
572 FFC7          ;       TELETYPE AND PACKS THEM INTO R0. OTHER REGISTERS
573 FFC7          ;       ARE LEFT UNDISTURBED.
574 FFC7 4300 A RDWD:     PUSH    3
575 FFC8 4200 A          PUSH    2
576 FFC9 4100 A          PUSH    1
577 FFCA 2909 A          JSR     RTTYSR
578 FFCB 5C08 A          SHL     0,8
579 FFCC 4000 A          PUSH    0
580 FFCD 2906 A          JSR     RTTYSR
581 FFCE 4500 A          PULL    1
582 FFCF 3482 A          RXOR    1,0
583 FFD0 4500 A          PULL    1
584 FFD1 4600 A          PULL    2
585 FFD2 4700 A          PULL    3
586 FFD3 0200 A          RTS
```

5.3 CONTROL PANEL SERVICE ROUTINE

The Control Panel Service Routine resides in ROM and controls all operations of the control panels except SYSTEM CLEAR, EXECUTE, HALT, SINGle INSTruction, and keyswitch functions. The control panel logic causes execution of this routine by generating an interrupt through an interrupt channel which is dedicated to the control panel and has priority over all other CPU operations. The Control Panel Service Routine performs the following operations in the sequence shown:

- Save registers, flags, and stack.
- Read command word from control panel.
- Interpret and execute command.
- Refresh PC display.
- Refresh Selected Display.
- Restore registers, flags, and stack.
- Return to interrupted program or repeat.

In the Halt Mode, the control panel displays are continuously updated by the Control Panel Service Routine. In the Execute Mode, the control panel displays are updated about 10 times per second and, therefore, do not continuously display the current contents of the selected display. The updating requires about 1 percent of CPU time and may cause interrupt service delays of up to 1500 microseconds. If this is undesirable for a particular application, the refresh may be inhibited by setting the Display Selector Switch to PROG DATA position, or by not grounding the Control Panel Inhibit (CPINH*) Line on the CPU Module. The Control Panel Service Routine entry address is through memory location CONSOLE = X'FFFD. Whenever the special interrupt channel causes transfers of control to the Control Panel Service Routine, the first tasks performed include saving the registers, stack, and status flags in the reserved memory locations X'101 through X'119. (Table 5-2 is a listing of the storage assignments in this reserved area.) After the processor's state is preserved in memory, the Control Panel Service Routine fetches the panel command word and the contents of the data switches from the control panel hardware.

The panel command word indicates the position of the Display Selector Switch and the state of the control buttons on the Programmers Control Panel and Operators Control Panel. This word has bits assigned for the LOAD DATA Button. The INCRement MEMory ADDRess Button, the LOAD PROGram Button, a bit to indicate the buttons are serviced once, and for each position of the Display Selector Switch. Table 5-3 is a listing of the bit assignments in the panel command word. The panel command word is received inverted from the panel hardware; that is, a logical zero indicates the position or button that is selected. The following paragraphs provide a sequential description of the Control Panel Service Routine operations.

If it is determined from the panel command word that the INCRement MEMory ADDRess Button has been pressed, the Memory Address Pointer (maintained in memory location X'117) is incremented. If the LOAD DATA Button is pushed, the data selected by the Display Selector Switch is replaced by the data switches.

If the LOAD PROGram Button is pushed, the start address of the Paper Tape Loader is stored into the PC location in memory and the stack is cleared. The loader is then executed after exiting the Control Panel Service Routine.

Next, the Control Panel Service Routine outputs the data specified by the Display Selector Switch to the Program Counter/Memory Address and selected display registers in the programmers hardware; these data are subsequently displayed to the operator. (If the Display Selector Switch is in the PROG DATA position or an intermediate position, there is no output display.)

Table 5-2. Programmers Control Panel CPU Status Storage Assignments

Memory Location (Hexadecimal)	Contents
101	AC0
102	AC1
103	AC2
104	AC3
105	Program Counter (Current Top of Stack) - Displayed PC.
106	Previous Top of Stack. Stack Display Word.
107 Through 114	Stack
115	Displayed Flags
116	RALU Flags
117	Memory Address Pointer
118	STK 13
119	STK 14
11A	Entry at Panel Stack Overflow
11B	Scratch Pad
11C	Scratch Pad
11D	Scratch Pad
11E	Scratch Pad
11F	Scratch Pad

Table 5-3. Panel Command Word Bit Assignments

Bit	Indication
15	A zero indicates that the console has been serviced at least once since the last button was pressed.
14	LOAD DATA Pushbutton
13	INCRement MEMory ADDRess Pushbutton
12	LOAD PROGram Pushbutton
11	(Not Used)
10	AC0 Selected
9	AC1 Selected
8	AC2 Selected
7	AC3 Selected
6	Program Counter Selected
5	Next Instruction Selected
4	Flags Selected
3	Stack Selected
2	Memory Address Pointer Selected
1	Memory Data Selected
0	Programmed Data Selected

Note: Data in bits 0 through 14 is returned inverted; that is, a logic zero indicates an active switch position.

At this point, the Interrupt Enable and Select Flags are restored. Then the stack is restored. If the stack is not full, the status flags and registers are restored, and control is returned to the user program.

Because the control panel uses the highest priority interrupt level, it is possible that data may be pushed from the bottom of the stack and lost if more than 14 stack locations are used. The user is warned of a possible loss of data from the bottom of the stack unless precautions are taken in the program against this eventuality.

When the Control Panel Service Routine determines that it may have pushed the bottom word from the stack, the service routine saves the current bottom two stack words in memory locations X'118 and X'119. Location X'118 has the next to the bottom word and location X'119 has the bottom word on the stack. Once these words are saved, the bottom two stack-words are cleared; the Interrupt Enable Flag is cleared; and control is returned to location X'11A with the registers and status flags restored and the preceding contents of the Program Counter entered at the top of the stack.

To guarantee the detection of a stack-full condition caused by the control panel, the user should not push words of all zeros onto the stack. To guarantee that data are not lost, the user should put two nonzero "dummy" words on the stack at initialization time. Furthermore, the user may have the Interrupt Enable Flag (INTEN) turned on in order to detect a stack-full condition via the normal interrupt sequence. Whether or not the user detects a Stack-full Interrupt via a Hardware Interrupt, he should take care to set location X'11A (stack-full track location) to either a Halt Instruction (X'0000) or a Jump Instruction to his Stack-full Software Routine.

Table 5-4 is a complete listing of the Control Panel Service Routine.

Table 5-4. Control Panel Service Routine

```
47  000                     .PAGE   'IMP-16L CONTROL CONSOLE SERVICE'
48  000 FF00 A              .=      ROMORG
49  FF00            ;
50  FF00            ;
51  FF00            ; CONTROL CONSOLE COMMANDS :
52  FF00            ;
53  FF00            ;   1. TO EXECUTE CONTROL PANEL COMMANDS, R3 MUST
54  FF00            ;      CONTAIN THE CONTROL PANEL ADDRESS (CPAD).
55  FF00            ;
56  FF00            ;      LD       3,CPAD
57  FF00            ;
58  FF00            ;   2. TO LOAD THE PROGRAM COUNTER DISPLAY REGISTER
59  FF00            ;      WITH THE CONTENTS OF R0-
60  FF00            ;
61  FF00            ;      ROUT     LPCDR
62  FF00            ;
63  FF00            ;   3. TO LOAD THE DATA REGISTER WITH THE CONTENTS
64  FF00            ;      OF R0-
65  FF00            ;
66  FF00            ;      ROUT     LDR
67  FF00            ;
68  FF00            ;
69  FF00            ;   4. TO GET THE DATA SWITCHES AND LOAD THEM INTO R0-
```

SOURCE STATEMENTS

RELOCATION CHARACTER: A - absolute, X - external, B - base, T - top, and I - indirect

INSTRUCTION OR DATA

MEMORY LOCATION

SOURCE LINE NUMBER

Table 5-4. Control Panel Service Routine (Continued)

```
                        ;
                        ;       RIN     GDS
                        ;
                        ;  5.  TO GET THE PANEL COMMAND SWITCH AND LOAD IT
                        ;       INTO R0-
                        ;
                        ;       RIN     GPCS
                        ;
FF00 FF04 A             ;       CPAD  =   H700          ; (H700 IS THE ADDRESS OF
                        ;
FF00 0060 A             ;       LPCDR =   X'60          ; A WORD THAT CONTAINS X'700)
                        ;
FF00 0068 A             ;       LDR   =   X'68
FF00
FF00 0070 A             ;       GDS   =   X'70
FF00
FF00 0078 A             ;       GPCS  =   X'78


FF00                            .PAGE
FF00                    ;
FF00                    ;
FF00                    ;  COMMAND WORD BIT ASSIGNMENTS
FF00                    ;
FF00                    ;
FF00                    ;       BIT     INDICATION
FF00                    ;       ---     ----------
FF00                    ;
FF00                    ;        15     A ZERO INDICATES THAT THE CONSOLE HAS
FF00                    ;               BEEN SERVICED AT LEAST ONCE SINCE THE
FF00                    ;               LAST BUTTON WAS DEPRESSED.
FF00                    ;
FF00                    ;        14     LOAD DATA PUSHBUTTON
FF00                    ;
FF00                    ;        13     INCREMENT MEMORY ADDRESS PUSHBUTTON
FF00                    ;
FF00                    ;        12     LOAD PROGRAM PUSHBUTTON.
FF00                    ;
FF00                    ;        11     (NOT USED)
FF00                    ;
FF00                    ;        10     AC0 SELECTED
FF00                    ;
FF00                    ;         9     AC1 SELECTED
FF00                    ;
FF00                    ;         8     AC2 SELECTED
FF00                    ;
FF00                    ;         7     AC3 SELECTED
FF00                    ;
FF00                    ;         6     PROGRAM COUNTER SELECTED
FF00                    ;
FF00                    ;         5     NEXT INSTRUCTION SELECTED
FF00                    ;
FF00                    ;         4     FLAGS SELECTED
FF00                    ;
FF00                    ;         3     STACK SELECTED
FF00                    ;
FF00                    ;         2     MEMORY ADDRESS POINTER SELECTED
FF00                    ;
FF00                    ;         1     MEMORY DATA SELECTED
FF00                    ;
FF00                    ;         0     PROGRAMMED DATA SELECTED.
```

Table 5-4. Control Panel Service Routine (Continued)

```
129 FF00                .PAGE
130 FF00        ;
131 FF00        ; PROGRAMMER'S CONSOLE CPU STATUS STORAGE ASSIGNMENTS
132 FF00        ;
133 FF00        ;  MEMORY LOCATION     CONTENTS
134 FF00        ;   (HEXIDECIMAL)
135 FF00        ;       ---            ------------------------------------
136 FF00        ;
137 FF00        ;       101            AC0
138 FF00        ;
139 FF00        ;       102            AC1
140 FF00        ;
141 FF00        ;       103            AC2
142 FF00        ;
143 FF00        ;       104            AC3
144 FF00        ;
145 FF00        ;       105            PROGRAM COUNTER (CURRENT TOP
146 FF00        ;                       OF STACK). DISPLAYED PC.
147 FF00        ;
148 FF00        ;       106            PREVIOUS TOP OF STACK. STACK
149 FF00        ;                       DISPLAY WORD.
150 FF00        ;
151 FF00        ;       107 THRU 114   STACK
152 FF00        ;
153 FF00        ;       115            DISPLAYED FLAGS
154 FF00        ;
155 FF00        ;       116            RALU FLAGS
156 FF00        ;
157 FF00        ;       117            MEMORY ADDRESS POINTER
158 FF00        ;
159 FF00        ;       118            STK13
160 FF00        ;
161 FF00        ;       119            STK14
162 FF00        ;
163 FF00        ;       11A            ENTRY AT PANEL STACK OVERFLOW
164 FF00        ;
165 FF00        ;       11B            SCRATCH PAD
166 FF00        ;       11C            SCRATCH PAD
167 FF00        ;       11D            SCRATCH PAD
168 FF00        ;       11E            SCRATCH PAD
169 FF00        ;       11F            SCRATCH PAD


170 FF00        .PAGE
171 FF00        ;
172 FF00        ;
173 FF00        ;
174 FF00        ;  CONTROL PANEL SERVICE
175 FF00        ;  DISPLACEMENT TABLE
176 FF00        ;                          AC2
177 FF00        ;LOCA-  USAGE              DISPLACEMENT
178 FF00        ;TION
179 FF00        ;                          (IN DECIMAL)
180 FF00        ;                          ENTRY  CONSOLE
181 FF00        ;                          EXIT   SERVICE
182 FF00        ;---    -----------        ---    ---
183 FF00        ;
184 FF00        ;101    AC0                 -2     -18
185 FF00        ;102    AC1                 -1     -17
186 FF00        ;103    AC2                  0     -16
187 FF00        ;104    AC3                  1     -15
188 FF00        ;105    PC                   2     -14
189 FF00        ;106    STACK                3     -13
```

Table 5-4. Control Panel Service Routine (Continued)

```
190 FF00          ;107    STK1              4      -12
191 FF00          ;108    STK2              5      -11
192 FF00          ;109    STK3              6      -10
193 FF00          ;10A    STK4              7       -9
194 FF00          ;10B    STK5              8       -8
195 FF00          ;10C    STK6              9       -7
196 FF00          ;10D    STK7             10       -6
197 FF00          ;10E    STK8             11       -5
198 FF00          ;10F    STK9             12       -4
199 FF00          ;110    STK10            13       -3
200 FF00          ;111    STK11            14       -2
201 FF00          ;112    STK12            15       -1
202 FF00          ;113    STK13            16        0
203 FF00          ;114    STK14            17        1
204 FF00          ;115    DISPLAYD FLAGS   18        2
205 FF00          ;116    RALU FLAGS       19        3
206 FF00          ;117    MEM ADDRESS PT   20        4
207 FF00          ;118    SAVED STK13      21        5
208 FF00          ;119    SAVED STK14      22        6
209 FF00          ;11A    STK OVFLW ENTY   23        7
210 FF00          ;11B                     24        8
211 FF00          ;11C                     25        9
212 FF00          ;11D                     26       10
213 FF00          ;11E                     27       11
214 FF00          ;11F                     28       12


215 FF00                  .PAGE
216 FF00                  .FORM   W,16
217 FF00 0002 A D2:       W       2
218 FF01 0004 A D4:       W       4
219 FF02 0006 A D6:       W       6
220 FF03 0101 A RAM       =       X'101
221 FF03 0103 A AC2P:     .WORD   RAM + 2
222 FF04 0700 A H700:     .WORD   0700
223 FF05 FFF9 A HFFF9:    .WORD   X'FFF9
224 FF06 FF00 A IENS      =       D2
225 FF06 FF01 A SELS      =       D4
226 FF06 0008 A STKFULL   =       8
227 FF06 0009 A IEN       =       9
228 FF06 000B A RLEO      =       11
229 FF06 000D A SEL       =       13
230 FF06 0002 A SELF      =       2
231 FF06 0001 A IENF      =       1
232 FF06 0002 A NDATA15   =       2


233 FF06                  .PAGE
234 FF06                  .LOCAL
235 FF06          RCONSOLE:
236 FF06                                                ; SAVE ACCUMULATORS
237 FF06                                                ;
238 FF06 B9FC A           ST      2,@AC2P               ;
239 FF07 89FB A           LD      2,AC2P                ;
240 FF08 A2FE A           ST      0,-2(2)               ;
241 FF09 A6FF A           ST      1,-1(2)               ;
242 FF0A AE01 A           ST      3,1(2)                ; R2 POINTS TO THE AC2
243 FF0B                                                ; SAVE LOCATION.
244 FF0B                                                ;
245 FF0B                                                ; SAVE THE RALU FLAGS
246 FF0B                                                ; IN R0.
247 FF0B                                                ;
```

Table 5-4. Control Panel Service Routine (Continued)

```
248 FF0B 4400 A          PULL    0                ;
249 FF0C 0080 A          PUSHF                    ;
250 FF0D AC00 A          XCHRS   0                ;
251 FF0E                                          ;
252 FF0E                                          ;
253 FF0E                                          ;
254 FF0E                                          ;
255 FF0E                                          ; SAVE THE STACK
256 FF0E                                          ;  4 PASS STACK SAVE
257 FF0E                                          ; SEQUENCE REQUIRES 6
258 FF0E                                          ; MORE WORDS THAN A 16
259 FF0E                                          ; PASS SEQUENCE
260 FF0E 4D04 A          LI      1,4              ; EXECUTION TIME FOR
261 FF0F 4700 A $0:      PULL    3                ; THIS SEQUENCE IS 255
262 FF10 AE02 A          ST      3,2(2)           ; MICROCYCLES VS.
263 FF11 4700 A          PULL    3                ; 440 MICROCYCLES FOR
264 FF12 AE03 A          ST      3,3(2)           ; THE 16 PASS CASE.
265 FF13 4700 A          PULL    3
266 FF14 AE04 A          ST      3,4(2)
267 FF15 4700 A          PULL    3
268 FF16 AE05 A          ST      3,5(2)
269 FF17 4A04 A          AISZ    2,4              ; INCREMENT AC2 BY 4
270 FF18 49FF A          AISZ    1,-1             ; DECREMENT PASS COUNTER
271 FF19 21F5 A          JMP     $0               ; JUMP BACK TO $0 IF NOT
272 FF1A                                          ; DONE.
273 FF1A          ;
274 FF1A          ; ON EXIT AC2 CONTAINS X'113 OR (AC2P) + 16.
275 FF1A          ;
276 FF1A          ; PUT THE SELECT AND INTERRUPT ENABLE FLAGS INTO
277 FF1A          ;    BIT POSITIONS 2 AND 1 OF THE FLAG WORD TO BE
278 FF1A          ;    DISPLAYED.
279 FF1A          ;
280 FF1A          ;     FIRST SET BITS 1 AND 2
281 FF1A          ;
282 FF1A A203 A            ST      0,3(2)         ; SAVE THE RALJ FLAGS
283 FF1B                                          ; IN MEMORY LOCATION
284 FF1B                                          ; X'116
285 FF1B 69E6 A            OR      0,D6           ; SET BITS 1 AND 2 TO
286 FF1C                                          ; ONE
287 FF1C 1901 A            BOC     IEN,.+2        ; BRANCH IF THE INTERRUPT
288 FF1D                                          ; ENABLE IS SET
289 FF1D 48FE A            AISZ    0,-2           ; IT IS NOT SET, CLEAR
290 FF1E                                          ; BIT 1.
291 FF1E                                          ;
292 FF1E 1D02 A            BOC     SEL,.+3        ; BRANCH IF THE SELECT
293 FF1F                                          ; FLAG IS SET
294 FF1F 48FC A            AISZ    0,-4           ; IT IS NOT SET, CLEAR
295 FF20                                          ; BIT 2. SKIP MAY OCCUR
296 FF20 2100 A            JMP     .+1            ; SO DO A HIGH SPEED NOP
297 FF21                                          ;
298 FF21 A202 A            ST      0,2(2)         ; STORE THE DISPLAYED
299 FF22                                          ; FLAGS IN LOCATION X'115
300 FF22          ;
301 FF22          ; THE LAST STACK WORD IS STORED IN LOC AC2 + 1 =  X'114.
302 FF22          ;
303 FF22          ;
304 FF22          ; INPUT THE COMMAND AND DATA WORDS FROM THE CONSOLE
305 FF22          ;
306 FF22 8DE1 A            LD      3,CPAD         ; PUT THE CONTROL PANEL
307 FF23 0478 A            RIN     GPCS           ; ADDRESS IN AC3 AND
308 FF24                                          ; FETCH THE PANEL CONTROL
309 FF24 3181 A            RCPY    0,1            ; WORD. SAVE IT IN AC1.
310 FF25 0470 A            RIN     GDS            ; FETCH THE DATA SWITCHES
```

Table 5-4. Control Panel Service Routine (Continued)

```
311 FF26 3180 A          RXCH    0,1
312 FF27 4F01 A          LI      3,1             ; AC3 WILL INDICATE WHETHER
313 FF28                                         ; OR NOT THE LOAD DATA
314 FF28                                         ; SWITCH HAS BEEN DEPRESSED
315 FF28 0A80 A          PFLG    SELF
316 FF29 5803 A          ROL     0,3
317 FF2A 71D6 A          SKAZ    0,D4            ; TEST BIT 2.IF IT'S A
318 FF2B 2101 A          JMP     .+2             ; 1 THEN TEST THE OTHER
319 FF2C 2109 A          JMP     $3              ; SWITCHES. IF IT IS A  0
320 FF2D                                         ; DO NOT TEST THEM.
321 FF2D 1B03 A          BOC     RLEQ,$1         ; BRANCH IF BIT 15 IS
322 FF2E 8D13 A          LD      3,LOADER        ; NOT 0. THE COMMAND
323 FF2F                                         ; WORD WILL NEVER BE 0).
324 FF2F                                         ; IF BIT 15 IS 0 THEN
325 FF2F AEF2 A          ST      3,-14(2)        ; PUT THE LOADER ADDRESS
326 FF30 4F01 A          LI      3,1             ; IN THE RETURN PC.
327 FF31 1302 A $1:      BOC     3,$2
328 FF32 7A04 A          ISZ     4(2)            ; INCREMENT THE MEMORY
329 FF33                                         ; ADDRESS POINTER THEN
330 FF33 2100 A          JMP     .+1             ; DO A HIGH SPEED NO OP
331 FF34 1401 A $2:      BOC     4,$3
332 FF35 4F00 A          LI      3,0             ; THE LOAD DATA SWITCH
333 FF36                                         ; HAS BEEN DEPRESSED.
334 FF36                                         ; SET AC3 TO 0 TO
335 FF36                                         ; INDICATE THIS.
336 FF36 CD0D A $3:      ADD     3,ADD1
337 FF37 5000 A          CAI     0,0
338 FF38 5CFE A          SHR     0,2
339 FF39 6109 A          AND     0,HFFC          ; MASK OFF THE UNWANTED
340 FF3A                                         ; BITS.
341 FF3A 5CFE A $6:      SHR     0,2
342 FF3B 1304 A          BOC     3,$7
343 FF3C 1404 A          BOC     4,$8
344 FF3D 4B06 A          AISZ    3,6
345 FF3E 112A A          BOC     REQ0,RESTORE    ; SWITCH IS IN AN
346 FF3F                                         ; INTERMEDIATE POSITION
347 FF3F 21FA A          JMP     $6
348 FF40 2300 A $7:      JMP     (3)
349 FF41 2303 A $8:      JMP     3(3)
350 FF42 FF9A A LOADER:  .WORD   RLOADER
351 FF43 0FFC A HFFC:    .WORD   X'0FFC
352 FF44 FF47 A ADD1:    .WORD   DMD
353 FF45 8204 A DMDA:    LD      0,4(2)          ; LOAD AC0 WITH THE
354 FF46                                         ; MEMORY ADDRESS POINTER
355 FF46                                         ; (LOCATION X'117).
356 FF46 211E A          JMP     D01
357 FF47 B604 A DMD:     ST      1,@4(2)         ; STORE THE DATA SWITCHES
358 FF48                                         ; IN THE LOCATION SPECIFIED
359 FF48                                         ; BY MEMORY LOCATION X'117.
360 FF48 9604 A          LD      1,@4(2)         ; LOAD AC1 WITH THE
361 FF49                                         ; CONTENTS OF THE LOCATION
362 FF49                                         ; SPECIFIED IN LOC X'117.
363 FF49 21FB A          JMP     DMDA
364 FF4A A604 A DMAR:    ST      1,4(2)          ; STORE THE DATA SWITCHES
365 FF4B                                         ; IN THE MEMORY ADDRESS
366 FF4B                                         ; POINTER (LOCATION X'117.)
367 FF4B 8604 A          LD      1,4(2)          ; LOAD AC1 WITH THE
368 FF4C                                         ; CONTENTS OF THE MEMORY
369 FF4C                                         ; ADDRESS POINTER (LOC X'117)
370 FF4C 2117 A          JMP     PCDEX           ; JUMP TO DISPLAY PC,AC1.
371 FF4D A6F3 A DSTACK:  ST      1,-13(2)        ; DISPLAY THE TOP OF THE STACK
372 FF4E 86F3 A          LD      1,-13(2)
373 FF4F 2114 A          JMP     PCDEX
```

Table 5-4. Control Panel Service Routine (Continued)

```
374 FF50 A602 A DFLAGS: ST      1,2(2)          ; DISPLAY THE FLAGS
375 FF51 8602 A         LD      1,2(2)
376 FF52 2111 A         JMP     PCDEX
377 FF53 B6F2 A DNI:    ST      1,@-14(2)       ; DISPLAY THE NEXT
378 FF54 96F2 A         LD      1,@-14(2)       ; INSTRUCTION
379 FF55 210E A         JMP     PCDEX
380 FF56 A6F2 A DPC:    ST      1,-14(2)        ; DISPLAY THE
381 FF57 86F2 A         LD      1,-14(2)        ; PROGRAM COUNTER
382 FF58 2108 A         JMP     PCDEX
383 FF59 A6F1 A DAC3:   ST      1,-15(2)        ; DISPLAY AC3
384 FF5A 86F1 A         LD      1,-15(2)
385 FF5B 2108 A         JMP     PCDEX
386 FF5C A6F0 A DAC2:   ST      1,-16(2)        ; DISPLAY AC2
387 FF5D 86F0 A         LD      1,-16(2)
388 FF5E 2105 A         JMP     PCDEX
389 FF5F A6EF A DAC1:   ST      1,-17(2)        ; DISPLAY AC1
390 FF60 86EF A         LD      1,-17(2)
391 FF61 2102 A         JMP     PCDEX
392 FF62 A6EE A DAC0:   ST      1,-18(2)        ; DISPLAY AC0
393 FF63 86EE A         LD      1,-18(2)
394 FF64 82F2 A PCDEX:  LD      0,-14(2)        ; PUT THE DISPLAYED PC
395 FF65                                        ; IN AC0
396 FF65 8D9E A D01:    LD      3,CPAD          ; DISPLAY THE CONTENTS OF
397 FF66                                        ; AC0 IN THE PC/MEM ADDRESS
398 FF66                                        ; LIGHTS AND DISPLAY AC1
399 FF66                                        ; IN THE SELECTED DISPLAY
400 FF66                                        ; LIGHTS.
401 FF66 0660 A         ROUT    LPCDR
402 FF67 3481 A         RCPY    1,0
403 FF68 0668 A         ROUT    LDR
404 FF69                                        ;
405 FF69        RESTORE:
406 FF69                                        ; RESTORE THE INTERRUPT
407 FF69                                        ; ENABLE AND SELECT
408 FF69                                        ; FLAGS.
409 FF69 8202 A         LD      0,2(2)
410 FF6A 7196 A         SKAZ    0,SELS          ; THE SELECT FLAG WAS
411 FF6B 0A00 A         SFLG    SELF            ; CLEARED IN THE PANEL
412 FF6C 0980 A         PFLG    IENF            ; SERVICE SEQUENCE.
413 FF6D 7192 A         SKAZ    0,IENS
414 FF6E 0900 A         SFLG    IENF
415 FF6F 6195 A         AND     0,HFFF9         ; MAKE UP THE NEW RALU
416 FF70 8603 A         LD      1,3(2)          ; FLAGS
417 FF71 6590 A         AND     1,D6
418 FF72 3100 A         RADD    0,1             ; THE NEW FLAGS ARE SAVED
419 FF73                                        ; IN AC1 WHILE THE STACK
420 FF73                                        ; IS RESTORED
421 FF73                                        ;
422 FF73                                        ; RESTORE THE STACK
423 FF73                                        ;
424 FF73                .LOCAL
425 FF73 4F04 A         LI      3,4             ; 4 PASS SEQUENCE REQUIRES
426 FF74 4AFC A $4:     AISZ    2,-4            ; 6 MORE WORDS THAN A 16
427 FF75 8205 A         LD      0,5(2)          ; PASS SEQUENCE BUT
428 FF76 4000 A         PUSH    0               ; EXECUTES IN 247
429 FF77 8204 A         LD      0,4(2)          ; MICROCYCLES AS
430 FF78 4000 A         PUSH    0               ; OPPOSED TO 432
431 FF79 8203 A         LD      0,3(2)          ; MICROCYCLES
432 FF7A 4000 A         PUSH    0
433 FF7B 8202 A         LD      0,2(2)
434 FF7C 4000 A         PUSH    0
435 FF7D 4BFF A         AISZ    3,-1
436 FF7E 21F5 A         JMP     $4
```

Table 5-4. Control Panel Service Routine (Continued)

```
437 FF7F          ;
438 FF7F          ; STACK OVERFLOW HANDLER
439 FF7F          ;
440 FF7F          ; IF THE STACK IS FULL THEN IT IS POSSIBLE THAT THE
441 FF7F          ; CONSOLE SERVICE ROUTINE HAS PUSHED A VALUE OFF THE
442 FF7F          ; BOTTOM OF THE STACK. IN THIS EVENTUALITY, THE
443 FF7F          ; BOTTOM TWO STACK VALUES ARE SAVED IN LOCATIONS X'118
444 FF7F          ; AND X'119 THEN THE BOTTOM TWO STACK POSITIONS ARE
445 FF7F          ; CLEARED. THE INTERRUPT ENABLE FLAG IS CLEARED AND
446 FF7F          ; CONTROL IS RETURNED TO LOC X'11A WITH THE OLD PC
447 FF7F          ; ON TOP OF THE STACK. THE TWO STACK POSITIONS MUST BE
448 FF7F          ; CLEARED TO GUARANTEE PROGRAM STABILITY IN THE OVER-
449 FF7F          ; FLOW CONDITION WHILE IN THE SINGLE INSTRUCTION MODE.
450 FF7F          ;
451 FF7F          ;
452 FF7F 1801 A          BOC     STKFULL,.+2     ; IS THE STACK FULL?
453 FF80 2109 A          JMP     $6              ; NO, JUMP AROUND THIS
454 FF81 8210 A          LD      0,16(2)         ; YES, SAVE STK13 IN
455 FF82 A215 A          ST      0,21(2)         ; LOC X'118 AND SAVE
456 FF83 8211 A          LD      0,17(2)         ; STK14 IN LOC X'119
457 FF84 A216 A          ST      0,22(2)         ; THEN PUT THE PANEL
458 FF85 810D A          LD      0,PRTN          ; OVERFLOW RETURN LOCA-
459 FF86 4000 A          PUSH    0               ; TION ON THE STACK
460 FF87 4000 A          PUSH    0               ; PUSHING OFF STK14 THEN
461 FF88 4400 A          PULL    0               ; PUSH OFF STK13 AND
462 FF89                                         ; PULL ONCE TO CLEAR THE
463 FF89                                         ; STACK BOTTOM.  THE NEXT
464 FF89                                         ; TO STACK BOTTOM WILL BE
465 FF89                                         ; CLEARED WITH THE RTS
466 FF89 0980 A          PFLG    IENF            ; RETURN. FINALLY, CLEAR
467 FF8A          $6:                            ; THE INTERRUPT ENABLE.
468 FF8A                                         ;
469 FF8A                                         ; RESTORE THE RALU FLAGS
470 FF8A                                         ;
471 FF8A AD00 A          XCHRS   1
472 FF8B 0280 A          PULLF
473 FF8C 4100 A          PUSH    1
474 FF8D                                         ;
475 FF8D                                         ; RESTORE THE REGISTERS
476 FF8D                                         ;
477 FF8D 82FE A          LD      0,-2(2)
478 FF8E 86FF A          LD      1,-1(2)
479 FF8F 8E01 A          LD      3,1(2)
480 FF90 8A00 A          LD      2,(2)
481 FF91                                         ;
482 FF91                                         ; RETURN
483 FF91                                         ;
484 FF91 0D80 A          PFLG    5               ; RESET THE CONTROL
485 FF92                                         ; PANEL INTERRUPT
486 FF92 0200 A          RTS                     ; ENABLE AND RETURN.
487 FF93 011A A PRTN:    .WORD   RAM + 2 + 23    ; PANEL SERVICE STACK
488 FF94                                         ; OVERFLOW RETURN.
```

5.4 TELETYPE CHARACTER RECEIVE ROUTINE AND BIT DELAY ROUTINE

The Teletype Character Receive Routine is used by the program loader to obtain information from the Teletype. This routine along with the associated Bit Delay Routine are also directly accessible by the user for Teletype operations.

The Character Receive Routine fetches a character from the keyboard or reader and returns it in the right-hand byte of AC0. The left-hand byte of AC0 is 0. In addition, the other accumulators contain the following at the exit: AC1 is the same as AC0; AC2 is 0; and AC3 contains the Teletype peripheral address, X'38.

The user may call the Character Receive Subroutine by coding:

```
JSRI   TTYSR
```

where

```
TTYSR = X'FFFB.
```

The Teletype Bit Delay Routine is used to provide a stable interval of 6320 microcycles* (8848 μ sec) or selected intervals in 350 microcycle increments.

*To obtain time in seconds, multiply by 1.4.

The user may call the 6320-microcycle interval by coding:

```
JSRI   DELAY
```

where

```
DELAY = X'FFF5.
```

He may call a selected interval of 15-1/2 + 350*N microcycles by placing N in AC0 and coding:

```
JSRI   DELAY1
```

where

```
DELAY1 = X'FFF6.
```

In each case, AC0 is 0 upon return. The other accumulators are returned unchanged. The Select (SEL) Flag is cleared.

The complete package of Teletype Service Routines, including routines to type (or type and punch) a character, type a character as it is received from the keyboard, or type a message, is included with the IMP-16L.

Table 5-5 is a complete listing of the Teletype Character Receive Routine.

Table 5-5. Teletype Character Receive Routine

```
587 FFD4                RTTYSR: .PAGE   'TELETYPE CHARACTER RECEIVE ROUTINE'
588 FFD4 0038 A TTYAD   =       7 * 8               ; IMP16L TELETYPE ADDRESS
589 FFD4                                            ; IS 7.
590 FFD4                                            ;
591 FFD4                                            ;
592 FFD4 0011 A E       =       17
593 FFD4 000C A D       =       12
594 FFD4 0009 A C       =       9
595 FFD4                        .LOCAL
596 FFD4 4F38 A $00:    LI      3,TTYAD             ; LOAD TTY ADDRESS INTO
597 FFD5 3181 A         RCPY    0,1                 ; AC3.
598 FFD6 0605 A         ROUT    5                   ; RESET TTY CONTROLLER
599 FFD7 4E08 A $30:    LI      2,8                 ; SET BIT COUNT TO 8.
600 FFD8 0402 A $40:    RIN     2                   ; LOAD TTY DATA INTO
601 FFD9                                            ; AC0 BIT 15.
602 FFD9 0604 A         ROUT    4                   ; ENABLE TTY READER
603 FFDA 1201 A $50:    BOC     2,$60               ; TEST FOR START BIT
604 FFDB 21FC A         JMP     $40                 ;
605 FFDC 4C09 A $60:    LI      0,C                 ; DELAY 1/2 BIT TIME
606 FFDD 2918 A         JSR     DELAY+1
607 FFDE 58F4 A         ROR     0,0                 ; COMPENSATION
608 FFDF 0402 A $70:    RIN     2                   ; TEST IF THE START BIT
609 FFE0                                            ; IS STILL THERE.
610 FFE0 1201 A $80:    BOC     2,$90               ; BRANCH IF GOOD START
611 FFE1 21F3 A         JMP     $00 + 1             ; FALSE START RETURN FOR
612 FFE2                                            ; A RESTART.
613 FFE2 2912 A $90:    JSR     DELAY
614 FFE3 5811 A         ROL     0,E                 ; COMPENSATION
615 FFE4 0402 A $100:   RIN     2                   ; LOAD DATA FROM TTY
616 FFE5 6108 A $110:   AND     0,$M                ; MASK UNWANTED BITS
617 FFE6 5DFF A         SHR     1,1                 ; SHIFT DATA
618 FFE7 3182 A         RXOR    0,1                 ; ADD NEW BIT TO DATA
619 FFE8 4AFF A         AISZ    2,-1                ; TEST IF DONE
620 FFE9 21F8 A $141:   JMP     $90                 ; NOT DONE, GET NEXT BIT
621 FFEA 290A A         JSR     DELAY               ; WAIT INTO THE FIRST
622 FFEB                                            ; STOP BIT.
623 FFEB 5DF8 A         SHR     1,8
624 FFEC 3481 A         RCPY    1,0
625 FFED 0200 A $160:   RTS                         ; DONE RETURN
626 FFEE 8000 A $M:     .WORD   X'8000
627 FFEF                .END
```

SOURCE STATEMENTS

RELOCATION CHARACTER: A - absolute, X - external, B - base, T - top, and I - indirect

INSTRUCTION OR DATA

MEMORY LOCATION

SOURCE LINE NUMBER

5.5 PROCESSOR INITIALIZATION

At initialization, the Program Counter is set at X'FFFE by the microprogram. The IMP-16L firmware has a jump to location-zero instruction in memory location X'FFFE that directs the processor to the user program in the Random Access Memory (RAM). The control panel hardware is initialized in the HALT state so that when power is applied or the INITialize Button is pressed, the displayed program counter is X'FFFE and the HALT lamp is lit. The contents of X'FFFE are listed in table 5-6.

5.6 CONTROL PANEL, LOADER, AND ROM ROUTINE ENTRY POINTS

As described earlier, certain memory locations are reserved for hardware-to-firmware communication. The reserved locations occupy the address space between X'FFF0 and X'FFFF. Table 5-6 is a listing of these locations. As may be seen in the listing, these locations contain Jump and Jump Indirect Instructions with the associated indirect addresses. These instructions direct requests for firmware service to the appropriate routines.

The Jump to Subroutine Implied (JSRI) Instruction is used by the hardware and may be used by the software to provide immediate linkage to these entry points.

The control panel hardware uses the JSRI Instruction as follows: when a control panel interrupt is recognized by the microcomputer, it produces a Read Control Panel (RDCP) Flag pulse followed by a read peripheral bus cycle with an address of zero. The control panel hardware responds with X'03FD (or X'03FC if there is no Programmers Control Panel) during the write data strobe interval. This data is a JSRI Instruction to location X'FFFD (X'FFFC) which is accepted by the microcomputer as the next instruction. As may be seen in the listing in table 5-6, location X'FFFD (X'FFFC) contains a jump indirect to the Control Panel Service Routine discussed earlier.

The locations X'FFFB, X'FFF5, and X'FFF6 are reserved as fixed entry points to the utility subroutines TTYSR, DELAY, and DELAY1, respectively, discussed in paragraph 5.4.

Finally, a provision is made on the IMP-16L Memory Module for ROMs in the fixed address space X'FE00 through X'FEFF. To provide direct linkage from software or possibly peripherals with level-zero interrupt capability, locations X'FFEF, X'FFF1, and X'FFF3 provide linkage to locations X'FE02, X'FE01, and X'FE00, respectively.

Table 5-6. Initialization Pointer, Control Panel Pointers, and Delay Subroutines

```
 1 0000                     .TITLE  BFIRL,'000157A   07/30/73'
 2 0000          ;          BFIRL IS COMPRISED OF THE BASIC FIRMWARE TO BE DELIVERED
 3 0000          ;          WITH THE IMP-16L COMPUTER. BFIRL PROVIDES THE FOLLOWING
 4 0000          ;          FUNCTIONS FOR THE IMP-16L USER:
 5 0000          ;
 6 0000          ;               1.  CONTROL CONSOLE SERVICE ROUTINE
 7 0000          ;               2.  STACK OVERFLOW HANDLER
 8 0000          ;               3.  ABSOLUTE PAPER TAPE LOADER
 9 0000          ;               4.  TTY CHARACTER-RECEIVE ROUTINE
10 0000          ;
11 0000          ;          BFIRL IS RESIDENT IN ROM AND IS CONFIGURED IN THE FOLLOWING
12 0000          ;          MANNER:
13 0000          ;
14 0000          ;               ROM DESIG     ADDRESS     HALF-WORD     COORD
15 0000          ;               ---------     -------     ---------     -----
16 0000          ;                  BDN         X'FF00       LEFT         9B
17 0000          ;                  BDO         X'FF00      RIGHT        10B
18 0000          ;
19 0000 FF00 A             ROMORG = X'FF00
20 0000                    .ASECT
21 0000                                                  ; CONSOLE AND INITIALIZATION
22 0000                                                  ; POINTERS
23 0000 FFEF A             . = ROMORG + 0EF
24 FFEF 2500 A POINT2:     JMP     @.+1                  ; THE LOCATIONS POINT0,
25 FFF0 FE02 A             .WORD   X'FE02                ; POINT1, AND POINT2
26 FFF1 2500 A POINT1:     JMP     @.+1                  ; PROVIDE DIRECT LINKAGE
27 FFF2 FE01 A             .WORD   X'FE01                ; TO LOCATIONS FE00,
28 FFF3 2500 A POINT0:     JMP     @.+1                  ; FE01, & FE02, RESP.
29 FFF4 FE00 A             .WORD   X'FE00
30 FFF5                                                  ;
31 FFF5                                                  ; TELETYPE BIT DELAY
32 FFF5                                                  ; SUBROUTINE
33 FFF5                                                  ;
34 FFF5 0070 A K           =       112                   ; THIS NUMBER MUST BE AN
35 FFF5 0012 A N           =       18                    ; INTEGER MULTIPLE OF 16.
36 FFF5 4C12 A DELAY:      LI      0,N
37 FFF6 0A80 A DELAY1:     PFLG    SELF
38 FFF7 5870 A             ROL     0,K
39 FFF8 48FF A             AISZ    0,-1
40 FFF9 21FD A             JMP     .-2
41 FFFA 0200 A             RTS
42 FFFB 21D8 A TTYSR:      JMP     RTTYSR
43 FFFC 2502 A OPHALT:     JMP     @CPOINT
44 FFFD 2501 A CONSOLE:    JMP     @CPOINT               ; CONTROL CONSOLE ENTRY
45 FFFE 2000 A INIT:       JMP     0                     ; INITIALIZATION ENTRY POINT
46 FFFF FF06 A CPOINT:     .WORD   RCONSOLE
```

SOURCE STATEMENTS

RELOCATION CHARACTER: A - absolute, X - external, B - base, T - top, and I - indirect

INSTRUCTION OR DATA

MEMORY LOCATION

SOURCE LINE NUMBER

Chapter 6

INPUT/OUTPUT OPERATION

6.1 GENERAL

Input/output operations may be initiated by the program or by the IMP-16L interrupt facility. Both of these operations are described in this chapter.

Two instructions, RIN (Register IN) and ROUT (Register OUT), effect communications between the processor and peripheral devices. Transfers into or out of the processor are made through CPU Accumulator AC0. RIN transfers data into the processor, and ROUT transfers data out of the processor.

Exchanges of data and control information for input/output operations take place over the 16-line System Data Bus. These exchanges occur between the external peripheral controller and the internal CPU Accumulator AC0.

The control signals associated with processor input/output are indicated in the block diagram of figure 6-1. The actual transfer of data is handled by the System Data Bus Control Logic which is described in chapter 7. The Control Flag Logic is used to initiate bus transactions, control portions of the interrupt system, and provide a number of other control functions. The control flags, which are accessible to the programmer, are listed in table 3-15. The Status Flag Logic is used to enable individual interrupt levels and is explained in paragraph 6.3, together with the Interrupt Logic. The Jump Condition Logic provides 16 different conditions for the BOC Instruction, including the state of the Interrupt Enable Flag. The jump condition assignments are listed in table 3-9. It should be noted that while condition 0000 is a branch-on interrupt, it is not necessary for the program to test this condition. This condition is tested by the hardware to cause an automatic branch to an interrupt routine at memory location 1. The hardware branch occurs at the completion of the current instruction when the interrupt line goes true, provided that the user has set the Interrupt Enable Flags.

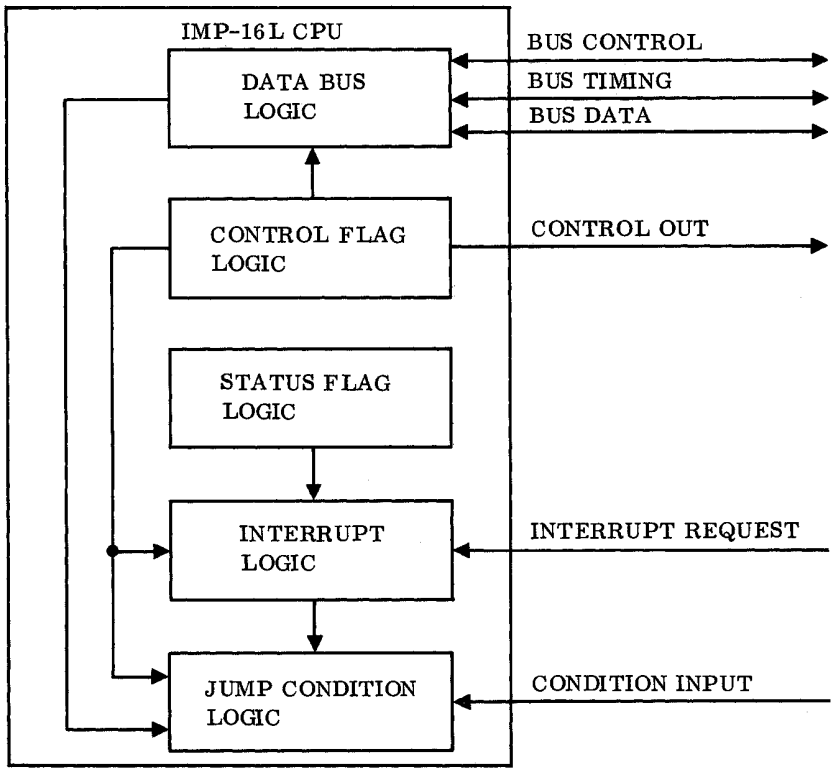
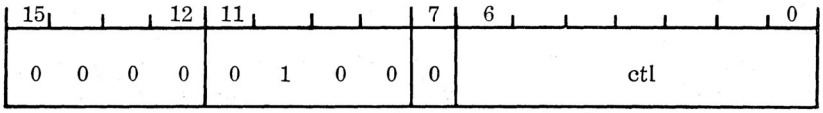


Figure 6-1. Processor I/O Control Signals

6.2 I/O INSTRUCTIONS

Both RIN and ROUT Instructions have the same format, as shown in figure 6-2. The ROUT Instruction sends two words out: a command word and a data word. The RIN Instruction sends one command word out and receives one data word. The command word is formed by the CPU from the sum of the contents of AC3 and the RIN or ROUT control field (ctl). The format of the command word is shown in figure 6-3.

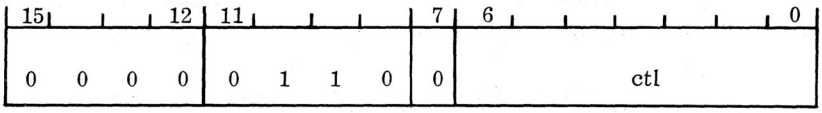
REGISTER IN (RIN)



Operation: (AR) ← ctl + (AC3), (AC0) ← (IOREG)

Description: The contents of AR (address register) are replaced by the sum of ctl and the contents of AC3. The new contents of AR constitute the address of a peripheral device and a command, both of which are received by the addressed peripheral device. The peripheral device responds by transferring the contents of its input/output register (IOREG) to the processor Accumulator AC0.

REGISTER OUT (ROUT)



Operation: (AR) ← ctl + (AC3), (IOREG) ← (AC0)

Description: The contents of AR (address register) are replaced by the sum of ctl and the contents of AC3. The new contents of AR constitute the address of a peripheral device and a command, both of which are received by the addressed peripheral device. The processor then tranofers the contents of AC0 to IOREG in the peripheral device.

Figure 6-2. Input/Output Instruction Formats

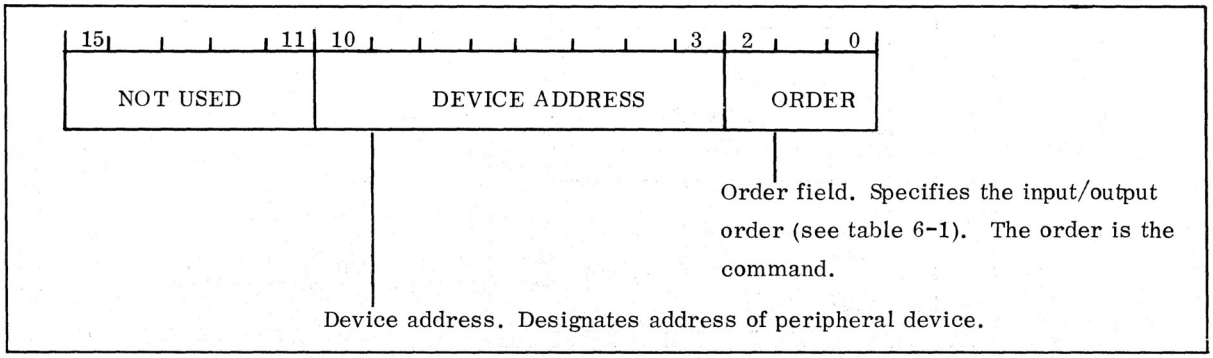


Figure 6-3. Command Word Format of RIN/ROUT Instructions

A maximum of eight RIN and eight ROUT orders is possible using the 3-bit order field of the RIN/ROUT command word. A maximum of 256 devices may be addressed by the 8-bit device address field.

Some orders have a different effect on different peripheral devices, and all orders do not apply to all devices. Thus, a comprehensive description that explains the orders for each device is not given here but is given in the description for the device. A general description of the typical usage of the orders is given in table 6-1, and typical bit assignments for the status word are listed in table 6-2.

The peripheral device acknowledges an order by setting the Peripheral Order Accepted (POA) Control Flag in the CPU. The program may then test this flag by use of the BOC Instruction. Details of the timing for this and other input/output signals are given in chapter 7.

Table 6-1. Typical RIN/ROUT Order Codes

Command Word	Order Code	Type of Transfer	Typical Usage
RIN	000	Unassigned	
	001	Peripheral Device Status	Returns status of controller flags to CPU (see table 6-2)
	010	Data Word (Read)	Reads data word from peripheral to CPU
	011	Unassigned	
	100	Unassigned	
	101	Unassigned	
	110	Interrupt Select Status 1	Place interrupt request status on assigned Data Bus Line
	111	Interrupt Select Status 2	
ROUT	000	Unassigned	
	001	Control to Peripheral Device	Transfers peripheral device control signals to controller; for example, set interrupt enable, rewind tape, and so forth.
	010	Read Order	Commands peripheral device to begin sending data to memory (using DMA)
	011	Read Check Order	Commands peripheral device to check validity of data; for example, cyclic redundancy check
	100	Set Address	Transfers the starting memory address for DMA operations
	101	Reset Peripheral Controller	Resets peripheral device controller flags
	110	Unassigned	
	111	Write Order	Commands peripheral device to receive data from the CPU or memory

Table 6-2. Typical RIN Status Bit Assignments

Bit Position	Significance
0	Input Ready
1	Output Ready
2	On Line
3	Device Busy
4	Interrupt Enable
5	Not Assigned
6	Error 1 (Data Overrun)
7	Error 2 (Transmission Error)
8	Not Assigned
9	Not Assigned
10	Not Assigned
11	Not Assigned
12	Not Assigned
13	Not Assigned
14	Not Assigned
15	Not Assigned
NOTE: Error status is typically reset with a ROUT Reset Peripheral Device Order or with an order to initiate another operation.	

Typically, once a peripheral device starts executing a particular operation, the only orders that are accepted by the peripheral device are Read Peripheral Device Status and Reset Peripheral Controller until the operation under execution is completed. All other orders are rejected. The peripheral device rejects an order simply by failing to signal the processor that the order has been accepted. Hence, the POA Flag is not set when an order is not accepted.

6.2.1 Use of RIN and ROUT Instructions

The use and operation of the RIN and ROUT Instructions are summarized below.

RIN Orders perform the following functions:

1. Read status of peripheral device.
2. Transmit data from peripheral device to processor.
3. Transmit Interrupt Select Status to processor.

RIN Instructions are executed by the processor in the following steps:

1. Form effective input command word by adding CTL field to AC3.
2. Load effective input command word onto System Data Bus.

3. Peripheral Controller transmits 16 bits to System Data Bus, and processor loads AC0 from bus.
4. POA Flag possibly is pulsed (if order was accepted).

ROUT Orders perform the following functions:

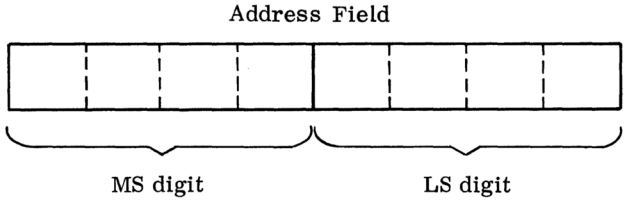
1. Initiate peripheral device operations.
2. Establish control status.
3. Transmit DMA-related addresses from processor to peripheral device.
4. Transfer data to peripheral device from processor.

ROUT Instructions are executed by the processor in the following steps:

1. Form effective command word by adding CTL field to AC3.
2. Load effective command word onto System Data Bus.
3. Processor loads bus with contents of AC0.
4. Peripheral Controller reads data from System Data Bus.
5. POA Flag possibly is pulsed (if order was accepted).

6.2.2 Peripheral Device Address Assignment

An address of a peripheral device comprises a 2-digit hexadecimal number, requiring an 8-bit address field as shown below.



The following addresses are fixed for any system configuration:

Address	Device
X'00	Reserved for control panel functions and interrupt level 0
X'07	Serial Teletype interface
X'EC	Control Panel Program Counter Display Register
X'ED	Control Panel Data Display Register
X'EE	Control Panel Data Switch Register
X'EF	Control Panel Command Switch Register

The assignment of other addresses is system configuration dependent. An example of address assignments in a possible system is given below:

Address	Peripheral Unit
X'01	Teletype (parallel interface)
X'02	Card Reader
X'03	Disc
X'04	Communications

Address	Peripheral Unit
X'05	Interval Timer
X'09	Line Printer

6.3 INTERRUPT OPERATION

There are four processor interrupt request levels plus a stack-full interrupt request. All the peripheral devices for one level are wired to the single input line for that level using an open collector gate. (See figure 6-4.) The processor responds to low inputs on the Interrupt Request Lines which are labeled IRREQ0* through IRREQ3*. If any device generates an interrupt request, the line goes low and interrupts the processor (if that level of interrupt is enabled). There is a separate Interrupt Enable Flag for each of the four interrupt levels (labeled IEN0 through IEN3), the master Interrupt Enable (INTEN) for all four levels and the stack-full interrupt. The Interrupt Enable Flags for each individual level are part of the CPU status flags. They are modified by use of the PUSHF, PULL, PUSH, and PULLF Instructions or by the Set Status Flag or Clear Status Flag Instructions. The INTEN Flag is one of the CPU control flags and is modified by use of the SFLG and PFLG Instructions.

Several different interrupt levels provide a convenient means of controlling interrupts for devices of different priority in a system with a number of peripherals. In a system with only one Interrupt Request Line, the interrupt service program must issue an order to each device of lower priority than the one being serviced to reset the lower priority Interrupt Enable Flags on the peripheral controllers. In a system with several interrupt levels, all devices of like priority are tied to the same Interrupt Request Line. All devices on an individual level have, at the processor, a common Interrupt Enable Flag which can disable all devices on that line simultaneously. When an interrupt occurs, lower priority levels may be disabled in a minimum amount of time.

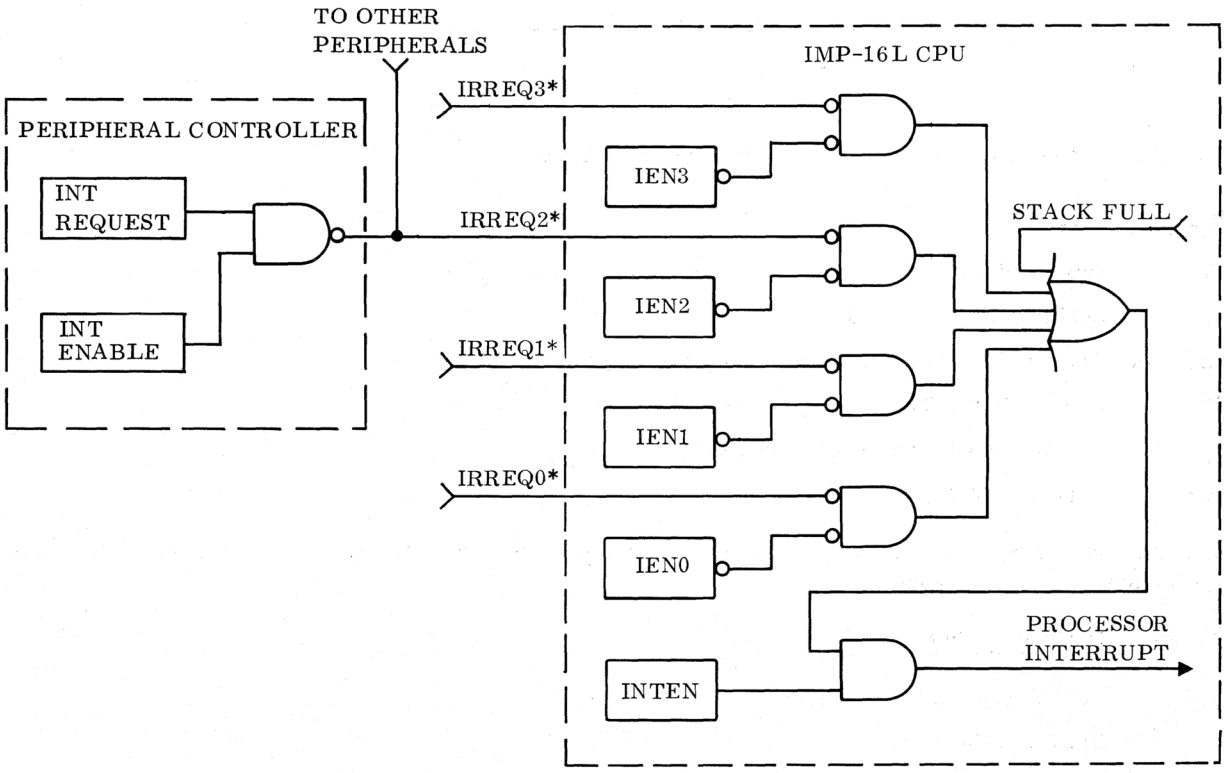


Figure 6-4. Processor Interrupt System

6.3.1 Interrupt Response

Response to a processor interrupt occurs at the end of the instruction being executed. The interrupt causes the processor to save the current state of the Program Counter (PC) on the top of the stack and to set the new contents of the PC equal to 1. The Interrupt Enable (INTEN) Flag is then turned off and the processor executes the instruction in memory location 1, which is the start of some Interrupt Service Routine. Interrupts on level 0 provide an automatic branch to the device service routine as explained in paragraph 6.3.3. The state of interrupts on levels 1, 2, and 3 may be determined by the use of the Interrupt Select Status Order as explained in paragraph 6.3.2. Interrupt service software determines the presence of a stack-full interrupt by use of the BOC Instruction.

6.3.2 Level 1, 2, and 3 Interrupts

A peripheral controller requiring interrupt servicing by the processor sets its open-collector Interrupt Request (IRREQ*) Flag, thus causing a low-level logic signal on the Interrupt Request Line, provided the peripheral Interrupt Enable Flag is set true. The Interrupt Request Line is common to all peripheral controllers on a given level. If both the processor INTEN and the Interrupt Enable for the given level are enabled, the processor recognizes an Interrupt Request Routine. When the processor responds to an interrupt request, it performs the following steps in order to transfer program control to the Interrupt Service Routine:

- Transfers the contents of the Program Counter (PC) to the top of the stack.
- Disables (clears) the processor INTEN Flag to prevent further interrupt.
- Fetches the next instruction from memory location 1, thus initiating the Interrupt Service Routine.

When an interrupt occurs, an Interrupt Service Routine should direct the processor to determine the devices requiring service and to select one peripheral device. This may be accomplished as follows:

- An Interrupt Select Status Order (RIN 6 or RIN 7) is sent out to all peripheral controllers. The order field of the command word is the only field recognized by the peripheral controllers, and the address field is ignored. Upon receipt of the Interrupt Select Status Order, the INT REQ Flags at the peripheral devices are cleared in most cases. An exception to this is the serial Teletype interface, which is described in paragraph 6.4.
- Each peripheral device is assigned one of the 16 System Data Bus Lines to report its interrupt status. Each peripheral device responds simultaneously with other peripheral devices, indicating whether or not it requires interrupt servicing. A binary 1 indicates a service request. (Typical interrupt assignments are shown in table 6-3.) Unused interrupt status bits should be masked out by the interrupt Service Routine since their value is undefined.
- The Interrupt Service Routine resolves interrupt priority and selects the peripheral device for interrupt servicing.

NOTE

Although there are only 16 System Data Bus Lines that are used for reporting interrupt status, by use of two Interrupt Select Status Orders, the status of 32 peripheral devices may be determined, one group of 16 peripheral devices responding to Interrupt Select Status 1 and another group to Interrupt Select Status Order 2. This concept can be extended to more than two levels of Interrupt Select Status Orders if necessary.

After a peripheral device obtains interrupt access, the applicable RIN or ROUT command then effects the transfer of data between the processor and the peripheral controller. Upon completion of the interrupt operation, program control may be transferred back to the main program by use of the RTI Instruction. The RTI Instruction causes the Program Counter to be loaded by adding the top word of the stack to the CTL field of the instruction. The INTEN flip-flop is then enabled.

Table 6-3. Interrupt Select Status 1 Bit Assignments

Bit	Assigned Peripheral
0	Unassigned
1	Parallel Teletype
2	Card Reader
3	Disc
4	Communications
5	Interval Timer
6	Unassigned
7	Serial Teletype
8	Modem
9	Line Printer
10	Unassigned
11	Unassigned
12	Unassigned
13	Unassigned
14	Unassigned
15	Unassigned

6.3.3 Level Zero Interrupts

Interrupt requests on level zero require that the interrupting device return an instruction to the processor that causes the processor to branch directly to the Interrupt Service Routine for that particular device. The primary advantage of this is that the interrupt is serviced in a minimal amount of time. The primary disadvantage is that the peripheral device controller is more complex and therefore more expensive to manufacture.

All devices connected to level zero must have a priority determination network so that when an interrupt request is acknowledged, only one device places its interrupt instruction on the System Data Bus.

Level zero interrupt servicing consists of the following steps:

- Peripheral Controller Interrupt Line goes true.
- If the device has priority, then it causes the IRREQ0* Line to go low.
- The device then causes all other devices to be locked out.
- The CPU responds to the interrupt (if INTEN is enabled) by initiating a read peripheral bus transaction with a device address of zero.

- The peripheral device responds by putting the Jump-to-level-zero Interrupt Indirect (JINT) Instruction on the bus.
- The low-order 4 bits of the instruction determine the address of the Interrupt Service Routine. (See chapter 3 for description of the JINT Instruction.)
- The INTEN Flag is turned off.
- The contents of the Program Counter are pushed on the stack and then the Program Counter is loaded with the address of the device service routine.

6.3.4 Stack-full Interrupt

In the event that the LIFO Stack in the CPU becomes full, a stack-full interrupt is set. If the INTEN Flag is set, the processor is interrupted. The Interrupt Service Routine may then test jump condition 8 to determine whether or not the interrupt is caused by a stack-full condition.

In most applications, the software can be written in a manner that guarantees that a stack-full condition does not occur. However, even in these cases, it is possible that in the system programming development a stack-full condition accidentally will occur. Because the interrupt process itself uses the stack, some words of the stack may be lost in the stack-full interrupting sequence.

The programmer must take precautions to guarantee that a stack-overflow error does not go undetected and that data on the stack is not lost. The programmer must not push words of all zeros onto the stack as data; two non-zero protection words should be kept at the bottom of the stack. These are relatively minor restrictions since, in most applications, use of the stack is reserved for subroutine return addresses, interrupt return addresses, and for saving the status flags while servicing interrupts.

The control console firmware provides further protection from an undetected stack-full condition. These extra precautions taken by the firmware are described in chapter 5.

6.4 SERIAL TELETYPE INTERFACE HARDWARE

A program-controlled serial Teletype interface is provided on the Control Panel Logic Module. A Teletype Character Receive Routine and a Bit Delay Routine are provided in ROM for convenience in using the Teletypewriter interface. These routines are described in chapter 5.

6.4.1 Interface Description

The Teletype interface is a two-way program-controlled bit-serial communication path which allows the CPU to send or receive serial bit streams to and from the Teletype. The formatting and timing of the bit stream is controlled by the CPU.

A block diagram of the Teletype interface is shown in figure 6-5. All communication between the CPU and the Teletype interface is over the System Data Bus and the interrupt facility. The RIN and ROUT Instructions are used for information and order code transfers.

The Address Decode Section enables the interface logic when the 8-bit address field specifies the Teletype address of X'07. In this event, the 3-bit order field is decoded (in the Order Decode Section) to determine which order should be executed by the interface logic. The Teletype receive circuit sets the Interrupt Flag in the Interrupt Request Logic whenever a start bit is received from the Teletype. If the Interrupt Enable Flag is set in the processor, as well as in the interface card, an interrupt occurs.

The interrupt may be used to initiate a program-controlled transfer of a Teletype character. The data received by the Teletype is read over the System Data Bus in bit position 15 by use of the RIN Instruction. The data format is shown in figure 6-6. Bits 0 through 14 are undefined and should be masked by the program. The bits comprising the character are transmitted serially, least significant bit first.

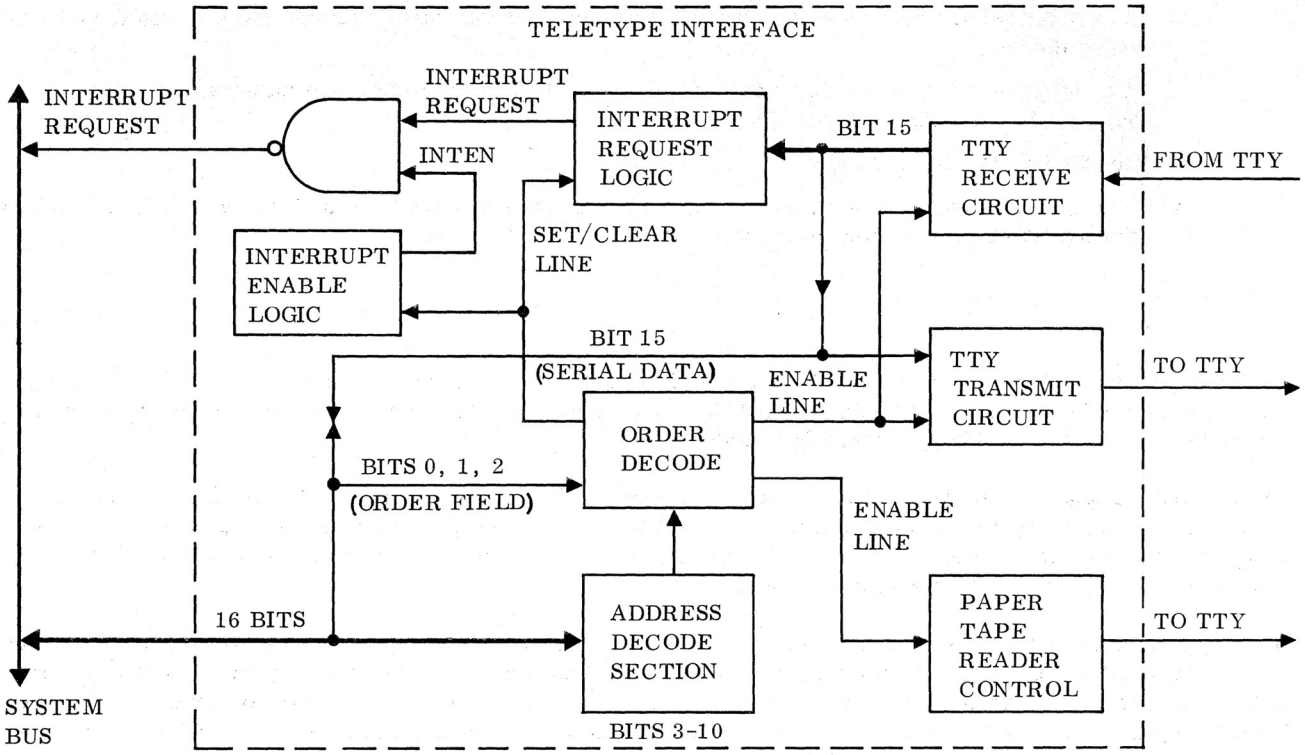


Figure 6-5. Serial Teletype Interface

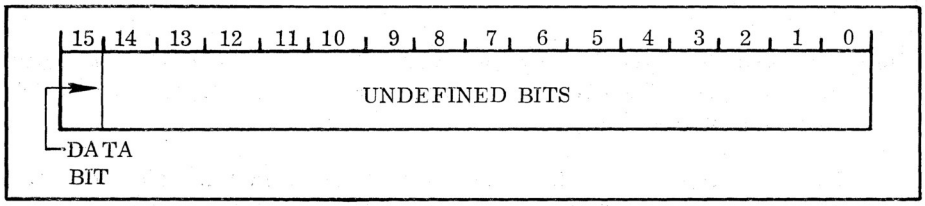


Figure 6-6. Teletype Data Word Format

The Teletype Transmit interface may be instructed to transmit a 1 or a 0 by use of the ROUT Instruction. Once set to a value, the interface continues to transmit that value until instructed to transmit another value or it is cleared with a RESET order. The transmit value is set from bit 15 of the System Data Bus. Bits 0 through 14 are not used and may be any value. The bits comprising the character should be transmitted serially, least significant bit first.

The Paper Tape Reader control may be used for Teletypes that have a Paper Tape Reader Relay Control Circuit. This optional feature allows program control of the Paper Tape Reader; this feature is especially desirable for applications where the data are processed as they are read, since the Paper Tape Reader may be stopped while processing occurs. This option is required when paper tape is used as the source input to the resident assembler

6.4.2 Order Codes

The orders acknowledged by the Teletype interface are listed in table 6-4. The effect of each order is explained below.

- RIN Code 2 - Bit 15 of the data bus is set equal to the output of the Teletype. The processor, in turn, transfers the contents of the data bus to AC0. Bits 0 through 14 are undefined.
- RIN Code 4 - The Paper Tape Reader is turned on.
- RIN Code 5 - The Interrupt Request and Interrupt Enable Flags are turned off. The Teletype output is set to the idle (marking) state. The Paper Tape Reader Enable is turned off.
- RIN Code 6 - The Teletype responds to the Interrupt Select Status 1 Order by setting data bit 7 equal to the state of the Interrupt Request Flag.
- ROUT Code 1 - The Teletype Interrupt Enable Flag is turned on.
- ROUT Code 3 - The Teletype transmit circuit is set to the value of data bit 15.
- ROUT Code 4 - The Paper Tape Reader is turned on.
- ROUT Code 5 - The Interrupt Request and Interrupt Enable Flags are turned off. The Teletype output is set to idle (marking) state. The Paper Tape Reader Enable is turned off.

Table 6-4. Serial Teletype Order Codes

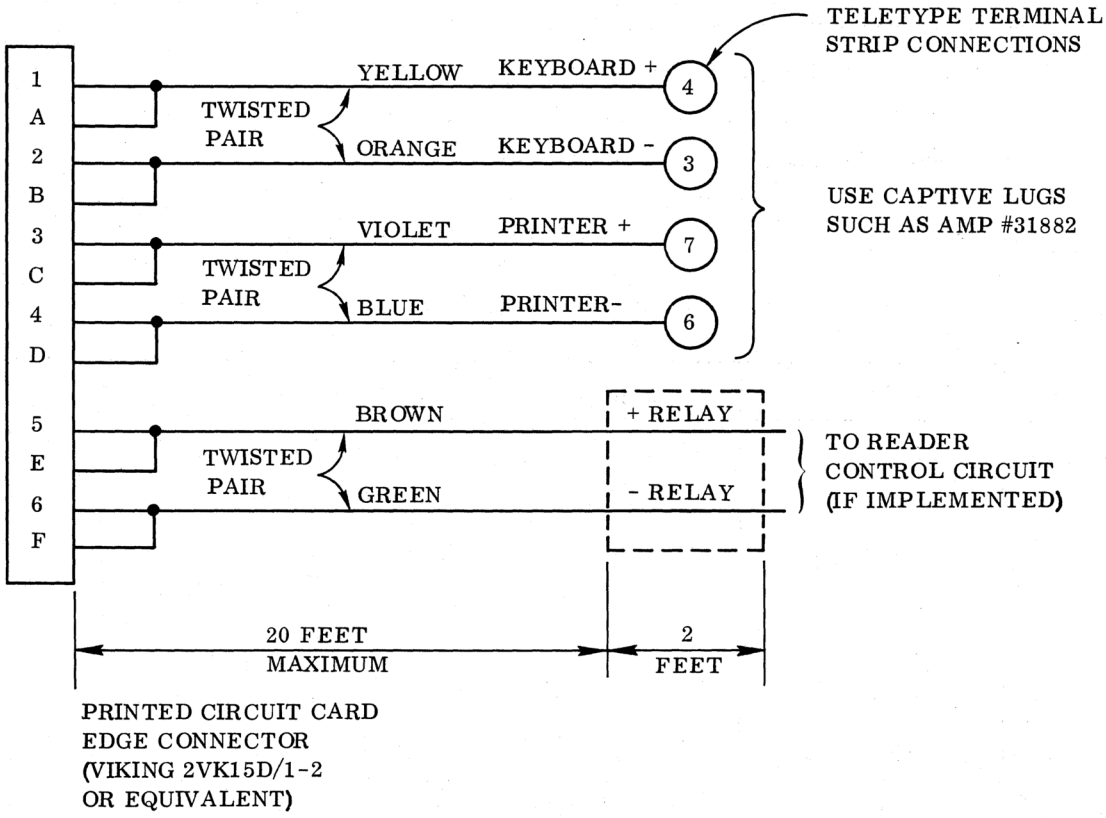
Instruction	Order Code	Action
RIN	000	No Action
RIN	001	Not Allowed
RIN	010	Read Bit From TTY
RIN	011	Not Allowed
RIN	100	Start Tape Reader
RIN	101	Reset
RIN	110	INT Status to Bit 7
RIN	111	No Action
ROUT	000	No Action
ROUT	001	Set Inten
ROUT	010	Not Allowed

Table 6-4. Serial Teletype Order Codes (Continued)

Instruction	Order Code	Action
ROUT	011	Write Bit to TTY
ROUT	100	Start Tape Reader
ROUT	101	Reset
ROUT	110	Not Allowed
ROUT	111	No Action

6.4.3 Teletype Interface Connections

Figure 6-7 shows the interfacing cable required to connect the Operators Control Panel Module to a Teletype. The Teletype equipment must be strapped to the 20-milliampere option.



NOTE: USE #22 AWG STRANDED WIRE.

Figure 6-7. Teletype Interface Cable

Chapter 7

SYSTEM DATA BUS

7.1 GENERAL

All communication in the IMP-16L processor occurs over the System Data Bus. The bus is independent of the processor and provides a communication link between any two devices connected to the bus. In order to transfer information over the bus, the device must first request access through the Bus Priority Network. If no higher priority request is present, control of the bus is granted, and the requesting device then becomes bus master for one bus cycle. During this cycle, the master may address any other device on the bus and command a transfer of one 16-bit data word to or from the device.

The capacity of the bus is approximately one million 16-bit words per second. While the bus provides the data-handling capabilities required for high-speed peripherals, such as a disc, slower devices that require byte-by-byte service from the processor are not hindered by this design.

7.2 BUS STRUCTURE

The System Data Bus, figure 7-1, is composed of three signal buses: a 16-bit Bidirectional Data Bus, a Timing Signal Bus, and a Control Signal Bus. Functional operation is as follows.

- The 16-bit Bidirectional Data Bus uses TRI-STATE ®positive logic, and is time-multiplexed to transfer addresses as well as data.
- The Timing Signal Bus provides the basic system clocks as well as address and data strobes which indicate when data is valid on the bus.
- The Control Signal Bus provides a priority system for bus access; signals to specify whether the current bus transaction is a read-memory, write-memory, read-peripheral, or write-peripheral operation; an extended bus cycle signal; a response line to indicate that a peripheral has accepted an order sent over the System Data Bus; and interface signals to the Bus Priority Network used to determine the next bus master.

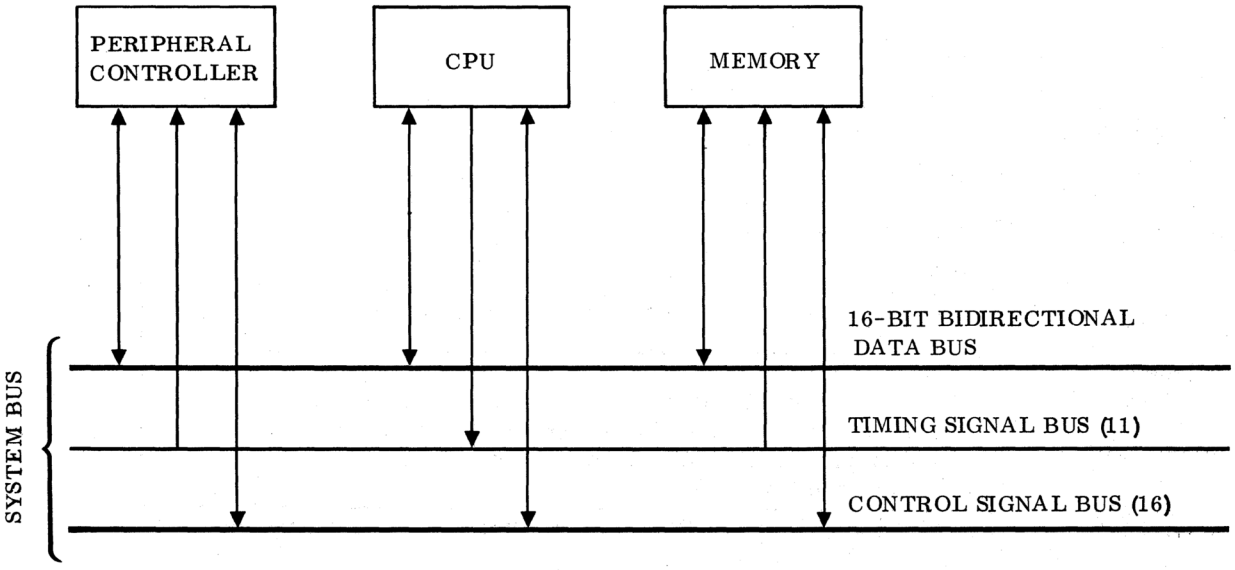


Figure 7-1. System Data Bus Structure

A list of the System Data Bus signals is given in table 7-1. Most of the signals originate in the CPU card. Four inverters on the control panel logic card are available to invert active high signals into active low signals for use in bus requests or as required.

Table 7-1. System Data Bus Signals

Name	Pin Number	Description
		Timing Bus Signals
ICLKA	69	Interface Bus Clock A
ICLKA*	70	Interface Bus Clock A
ICLKB	73	Interface Bus Clock B
ICLKB*	74	Interface Bus Clock B
IADS	75	Interface Address Data Strobe
IADS*	76	Interface Address Data Strobe
IWDS	77	Interface Write Data Strobe
IWDS*	78	Interface Write Data Strobe
IRDS	79	Interface Read Data Strobe
IRDS*	80	Interface Read Data Strobe
BICLKB	68	Buffered Bus Clock B
		Control Bus Signals
IRMC*	67	Interface Read Memory Cycle
IWMC*	65	Interface Write Memory Cycle
IRPC*	129	Interface Read Peripheral Cycle
IWPC*	127	Interface Write Peripheral Cycle
IPOA*	131	Interface Peripheral Order Accepted
IBHLD*	82	Interface Bus Hold
IBR0*	45	Interface Bus Request 0
IBR1*	46	Interface Bus Request 1
IBR2*	47	Interface Bus Request 2
IBR3*	48	Interface Bus Request 3
IPS0*	49	Interface Priority Select 0
IPS1*	50	Interface Priority Select 1

NOTE: The asterisk (*) indicates that the associated signal is an active low signal (that is, ICLKB and ICLKB* are complement signals; ICLKB is the active high, and ICLKB* is the active low).

Table 7-1. System Data Bus Signals (Continued)

Name	Pin Number	Description
Control Bus Signals (Continued)		
IPS2*	51	Interface Priority Select 2
IPS3*	52	Interface Priority Select 3
APVO*	95	Interface Approve Out[1]
APVI*	96	Interface Priority Approve In[1]
IAI	10	Interface Inverter A Input[2]
IBI	14	Interface Inverter B Input[2]
ICI	16	Interface Inverter C Input[2]
IDI	22	Interface Inverter D Input[2]
IAO	12	Interface Inverter A Output[2]
IBO	18	Interface Inverter B Output[2]
ICO	20	Interface Inverter C Output[2]
IDO	26	Interface Inverter D Output[2]
Data Bus Signals		
IDB00	9	Interface Data Bit 0
IDB01	11	Interface Data Bit 1
IDB02	13	Interface Data Bit 2
IDB03	15	Interface Data Bit 3
IDB04	17	Interface Data Bit 4
IDB05	19	Interface Data Bit 5
IDB06	21	Interface Data Bit 6
IDB07	23	Interface Data Bit 7
IDB08	27	Interface Data Bit 8
IDB09	29	Interface Data Bit 9
IDB10	33	Interface Data Bit 10
IDB11	35	Interface Data Bit 11
IDB12	37	Interface Data Bit 12
IDB13	39	Interface Data Bit 13
IDB14	41	Interface Data Bit 14
IDB15	43	Interface Data Bit 15

[1]Provided for expansion of DMA bus to more than four DMA devices.

[2]Spare inverters.

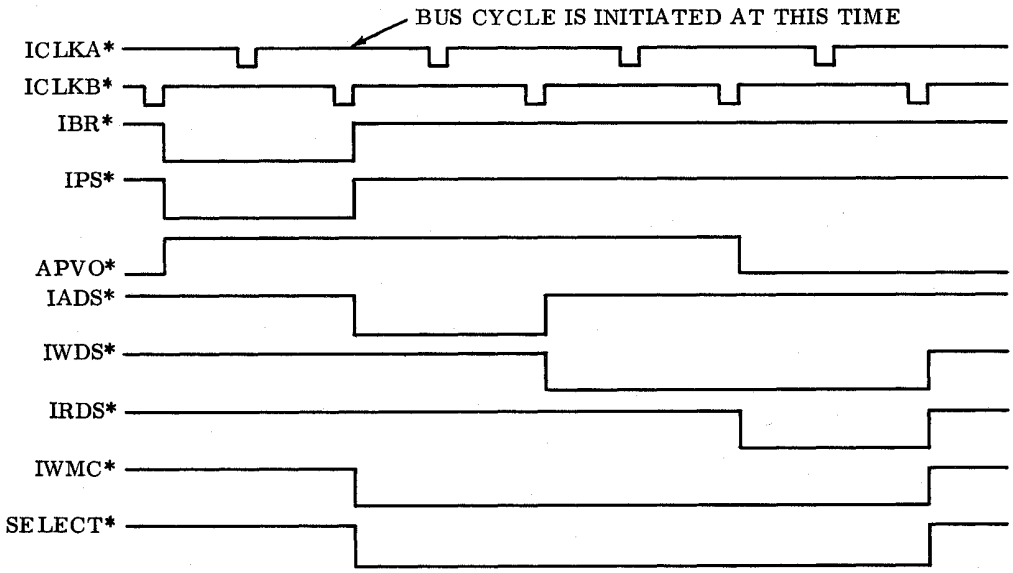
7.3 BUS TIMING

Figure 7-2 is a timing diagram which shows the time relationships existing during a typical bus write operation. The clock and control signals perform the synchronization of the data transfer over the System Data Bus. The basic system clock generates two clocks called ICLKA (Interface Bus Clock A) and ICLKB (Interface Bus Clock B). The trailing edge of ICLKB initiates and terminates various bus signals such as IADS (Interface Address Data Strobe), IWDS (Interface Write Data Strobe), and IRDS (Interface Read Data Strobe). The trailing edge of ICLKA may be used to clock data received over the bus.

When a peripheral is selected for access to the System Data Bus, the address and data transfers within the read or write sequences are fixed. Within the clock period following priority selection, the selected peripheral must gate a 16-bit address word onto the System Data Bus. This address word must remain stable for one clock period. The Control Bus Logic provides an Interface Address Data Strobe (IADS*) delimiter which is energized during this period of time. The Control Bus Logic also provides an Interface Write Data Strobe (IWDS) and an Interface Read Data Strobe (IRDS) to synchronize the data transfer. IWDS is energized for two or more clock periods following IADS, allowing time for the data to be written into the memory register soon enough for a write memory operation. IRDS is energized during the third or final clock period of the data transfer cycle.

The bus write timing diagram, figure 7-2, illustrates what takes place when the CPU generates a write memory request. Operation is as follows:

- When IBR* (Interface Bus Request) goes low, a bus request is generated (if no higher priority is requesting, and the bus is free).



NOTE: THE ASTERISK (*) INDICATES THAT THE ASSOCIATED SIGNAL IS AN ACTIVE LOW SIGNAL.

ICLKA*	=	INTERFACE CLOCK A
ICLKB*	=	INTERFACE CLOCK B
IBR*	=	INTERFACE BUS REQUEST
IPS*	=	INTERFACE PRIORITY SELECT
APVO*	=	APPROVE OUTPUT
IADS*	=	INTERFACE ADDRESS DATA STROBE
IWDS*	=	INTERFACE WRITE DATA STROBE
IRDS*	=	INTERFACE READ DATA STROBE
IWMC*	=	INTERFACE WRITE MEMORY CYCLE

Figure 7-2. Bus Write Timing Diagram

- The bus request is granted in the form of the IPS* (Interface Priority Select) Signal at the trailing edge of the next ICLKB*, initiating a bus cycle.

- At the time IADS* and IWMC* are activated, and (in the case of the memory write operation) a memory write cycle is initiated.

NOTE

In the event that the bus is not free, or a higher priority device is requesting, IPS* will not be low at the trailing edge of ICLKB* and a bus cycle therefore cannot be initiated. For this reason, the state of these two lines (IPS* and ICLKB*) is monitored by the requesting bus controller.

During the entire bus cycle, the bus master (in this case the CPU) drives one of the four following signal lines low to specify the type of bus transaction:

IWMC* = Interface Write Memory Cycle

IRMC* = Interface Read Memory Cycle

IWPC* = Interface Write Peripheral Cycle

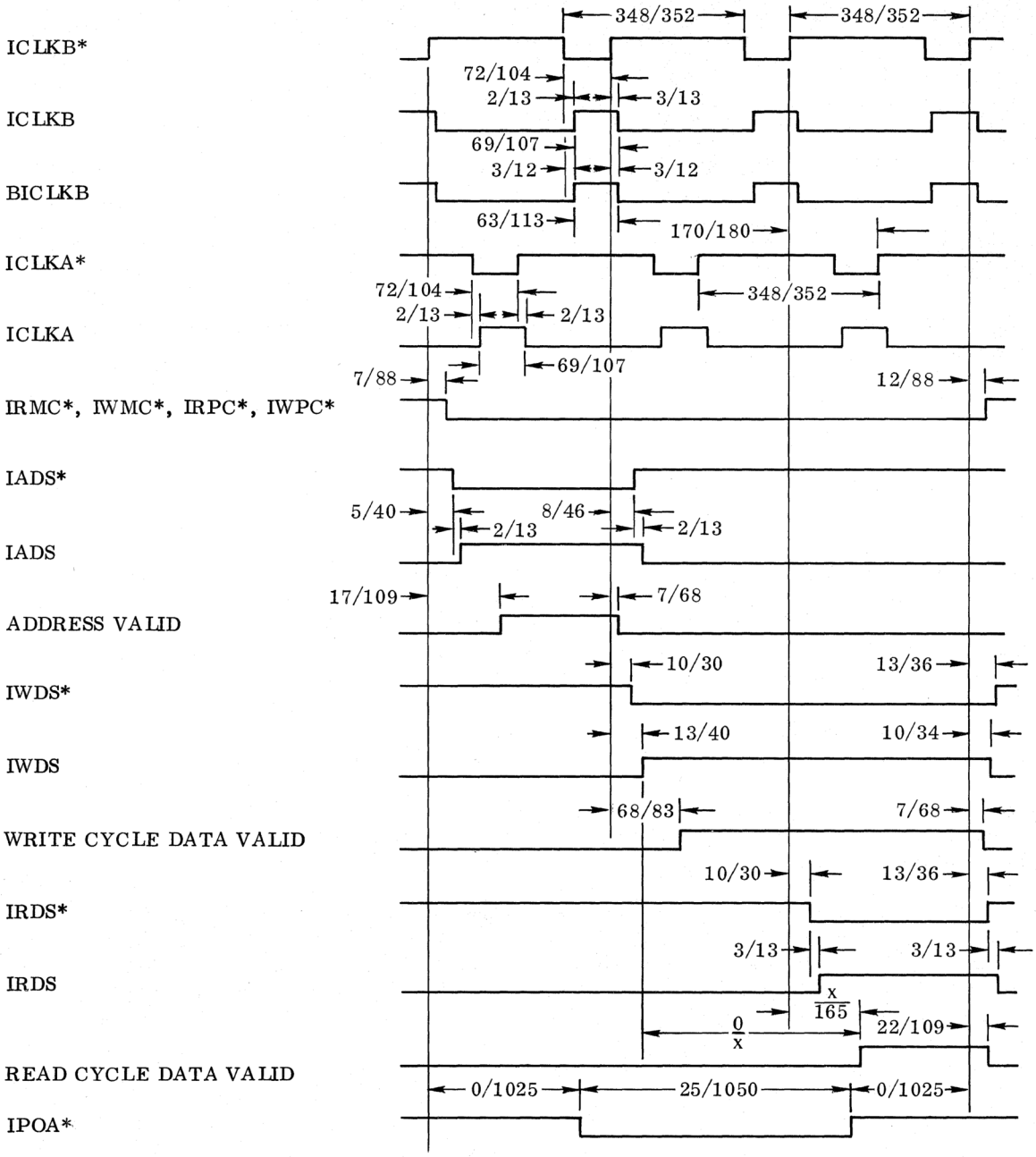
IRPC* = Interface Read Peripheral Cycle

Once the bus request is granted, the bus control circuit in the CPU provides three data strobe signals to control the data transfer: IADS, IWDS, and IRDS. IADS is used by the master as a delimiter gate for putting the address onto the System Data Bus. The address must be valid after the IADS* goes low and remain valid as shown in figure 7-3. This ensures that data will be valid before the trailing edge of ICLKA. If the transfer is a Write Memory or Write Peripheral operation, IWDS* is used as a delimiter to gate the data onto the bus. The data must be valid as shown in figure 7-3. IRDS* is used by the receiving device during an operation as the delimiting gate for strobing data from the bus. Data is valid as shown in figure 7-3. For all three of the above strobe signals (IADS*, IWDS*, and IRDS*) it is desirable to remove the strobed data from the bus as soon as possible after the strobe goes away to prevent excessive current when the next data word is strobed onto the bus. At the latest, the data must be removed in time to let the next data word stabilize before the next leading edge of ICLKA. This allows the receiving latch time to set up before being clocked. Interfacing devices should use the trailing edge of ICLKB* to clock in address and data information as this is overlapped by IADS, IWDS, and IRDS.

When a Read or Write Peripheral Order code is sent over the bus, the peripheral device may respond with an Interface Peripheral Order Accepted (IPOA*) Signal. This signal goes to the direct set input of a latch and, if IPOA* goes low at any time during the cycle, the latch is set. The state of this latch may be tested by the BOC Instruction to verify that the order was received. This line should be driven only during bus transactions in which the processor is bus master and in accordance with timing restrictions of figure 7-3.

In order to allow for memories whose access time exceeds the time available in a normal bus cycle, an extended cycle capability is provided. Any bus cycle may be extended by a low signal on the Interface Bus Hold (IBHLD*) Line. This feature should be used with caution, as it may upset the timing of other system functions, such as memory refresh, or programmed input/output operations. The timing requirements for IBHLD* are shown in figure 7-4. IBHLD* must be low ahead of the ICLKB* which would normally initiate IRDS. It may remain low for any length of time less than 4 microseconds.

IRDS* is initiated on the ICLKB* following the positive going transition of IBHLD*. IBHLD* must not change state while ICLKB* is low. To read memory, the CPU or peripheral controller must activate the Interface Read Memory Control (IRMC*) Signal during the clock period following priority selection. IRMC* must remain active until the data transfer is complete. Similarly, to write memory, the Interface Write Memory Control (IWMC*) Signal must be active.



- MAXIMUM DATA LINE CAPACITANCE = 320 pf
- MAXIMUM CLOCK OR CONTROL SIGNAL CAP. = 210 pf
- DELAYS SPECIFIED AT 1.5 V
- ALL DELAYS EXPRESSED IN NANOSECONDS AS: $\frac{\text{MINIMUM DELAY}}{\text{MAXIMUM DELAY}}$

Figure 7-3. Timing Specifications

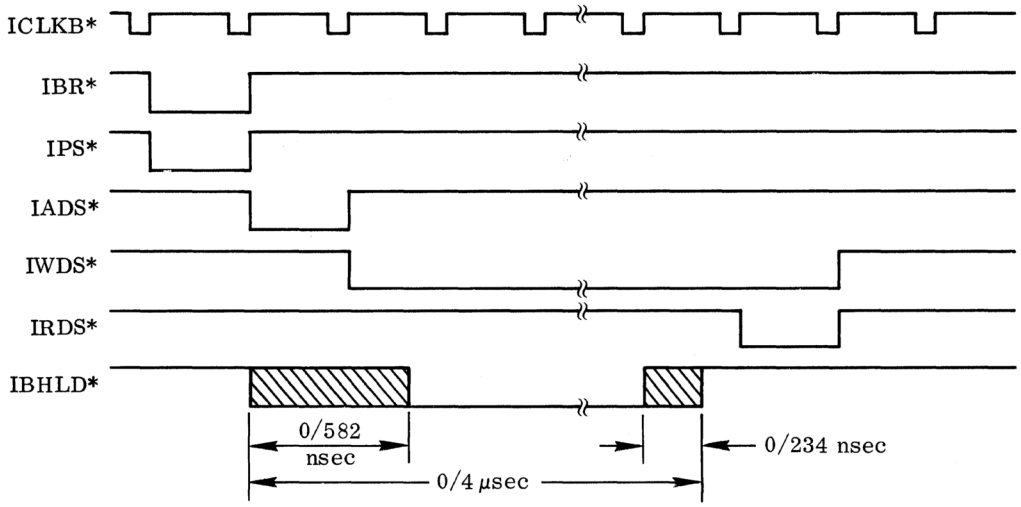


Figure 7-4. Bus Hold Timing Requirements

To select a peripheral device for input/output operation, the CPU must obtain access to the System Data Bus. When Interface Priority Select (IPS*) is activated, the CPU generates either an IRPC* (Interface Read Peripheral Control) or an IWPC* (Interface Write Peripheral Control) Line. Once the Bus Control Logic has started its sequence, the CPU must gate the peripheral address/command word onto the System Data Bus during Interface Address Data Strobe (IADS) time.

Each peripheral device decodes bits 3 through 10 of the peripheral address/command word to determine if it is being addressed. If a peripheral detects its address in conjunction with IRPC* or IWPC*, it proceeds as follows:

1. If the peripheral device is busy, it ignores the command word and continues its input/output function. However, if the command word is a Read Status Order, then the peripheral device gates its status onto the System Data Bus to be read by the processor.

2. If the peripheral device is not busy, it accepts the order information and activates the IPOA* (Peripheral Order Accepted*) Line. (Note that not all devices incorporate the IPOA function.)

The functions performed as a result of a peripheral command vary from one type of peripheral device to another. For example, a Read Peripheral Order to a Card Reader initiates a block transfer of data on a complete 80-column card from the Card Reader into main memory. In contrast, for a Teletype, a Read Peripheral Order executes the transfer of one bit from the Teletype Controller to the processor.

7.4 BUS PRIORITY

Priority on the System Data Bus is determined by the Priority Logic, which controls the Priority Network (see figure 7-5). Each direct-memory-access peripheral device or subsystem is assigned a priority level. When access to the bus is required, the requesting device drives its bus request line to the priority network. If it is the highest priority request, it receives an Interface Priority Select (IPS) Signal back from the priority network.

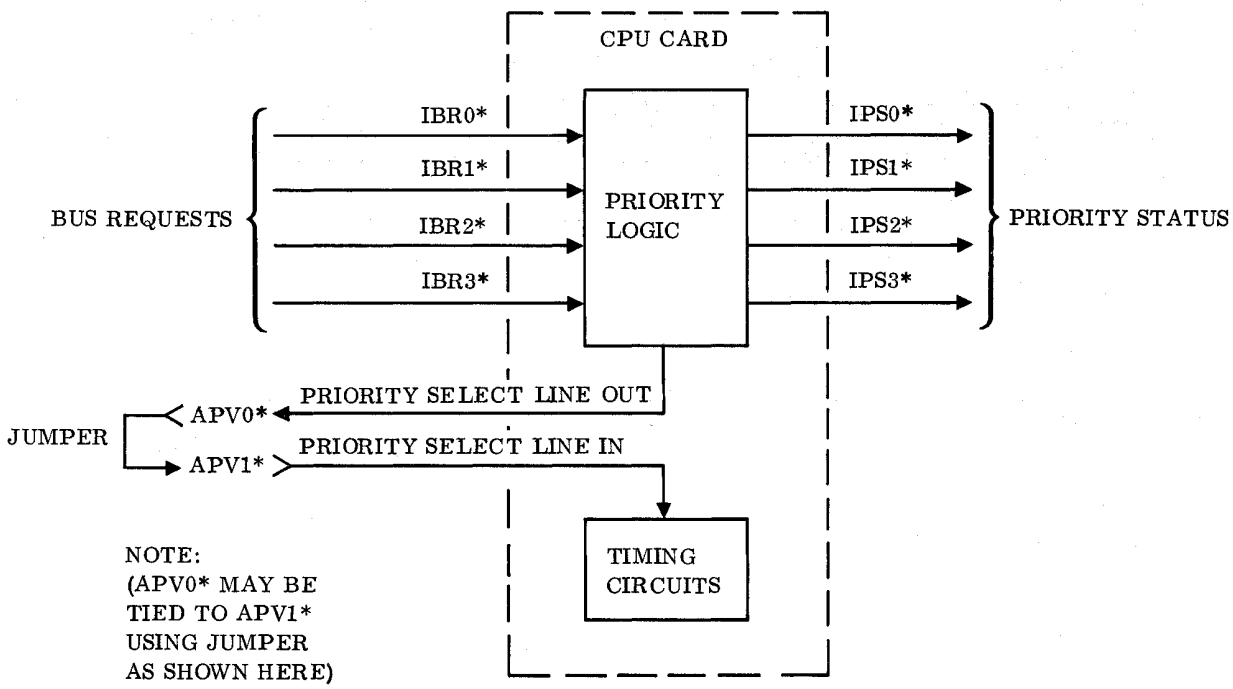


Figure 7-5. Bus Priority Signals

Access to the System Data Bus must be obtained through the bus Priority Logic. Signals associated with this logic are shown in figure 7-6. Four levels of bus priority are provided on the CPU card with level 0 having the highest priority. The APVO* (Priority Approve Output*) of the Priority Network may be used to increase the number of levels available. If no external levels are added, it should be tied directly to APVI* (Priority Approve IN*) at the CPU connector. Figure 7-6 shows the general structure of the Priority Logic for extending the Priority Logic to an expanded system.

An Interface Bus Request (IBR*) may be initiated on the trailing edge of any ICLKB regardless of the present bus activity. The corresponding Interface Priority Select (IPS) Signal should be sampled on the trailing edge of all following ICLKB* pulses until access to the bus is granted (as indicated by IPS* going low). The bus cycle immediately follows granting access to the bus and the bus request should be terminated before the start of the IRDS (Interface Read Data Strobe). The bus grant is removed at the start of the cycle.

The Priority Approve Out (APVO*) is disabled during a bus transaction until the last cycle when it is enabled again. This allows back-to-back bus transactions to take place. Using this scheme, no overhead is lost while deciding the next priority.

Figure 7-7 illustrates the timing of two levels of priority. Signals that initiate these two levels are identified in the timing diagram of figure 7-7 as IBR2* and IBR1*. Operation is as follows: The level 2 device makes a bus request by setting IBR2* low. Normally, this request is granted at the next clock ICLKB*. However, before the request can be granted, the device on level 1 (IBR1*) makes a request by setting IBR1* low and the level 1 request is granted. Level 2 then retains its request by keeping IBR2* low, and the level 2 request is granted later at the conclusion of the level 1 transaction.

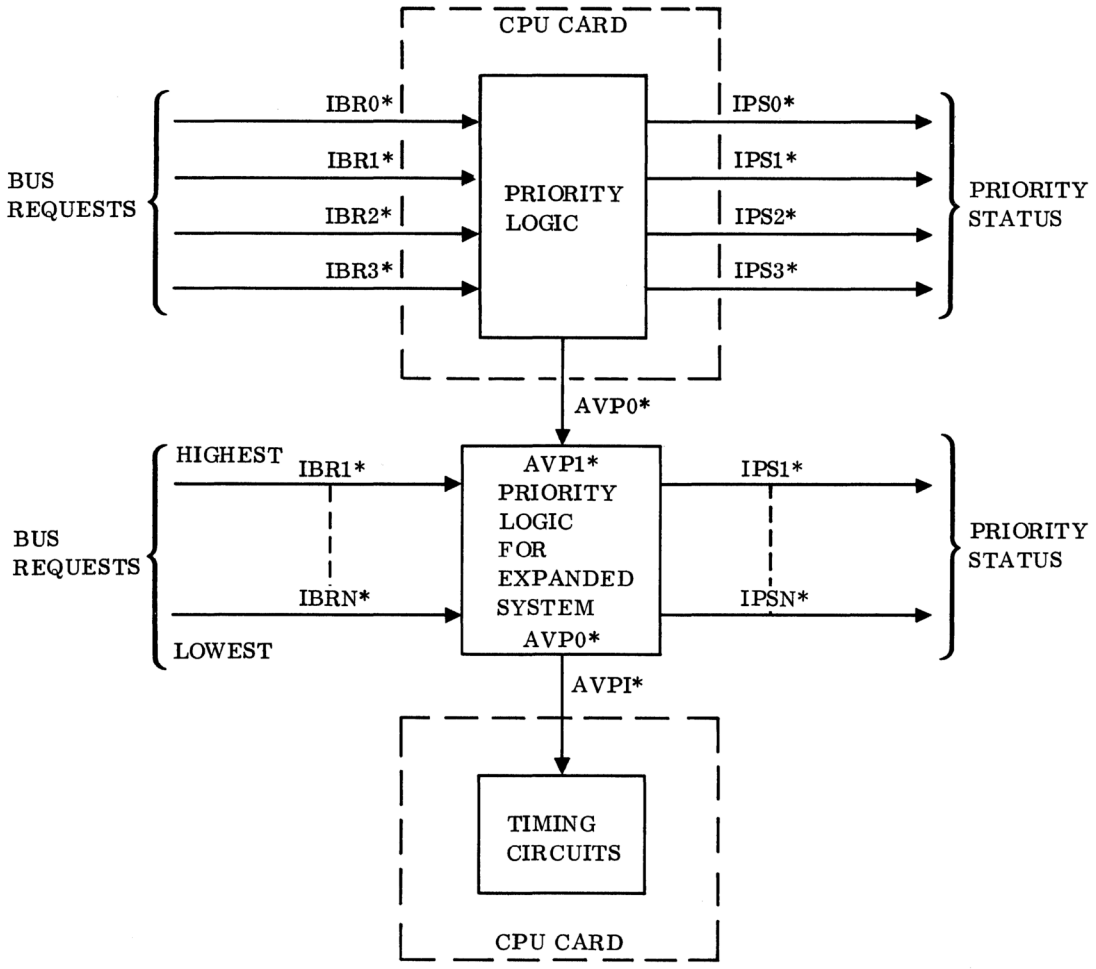


Figure 7-6. General Structure for Extended Priority Logic

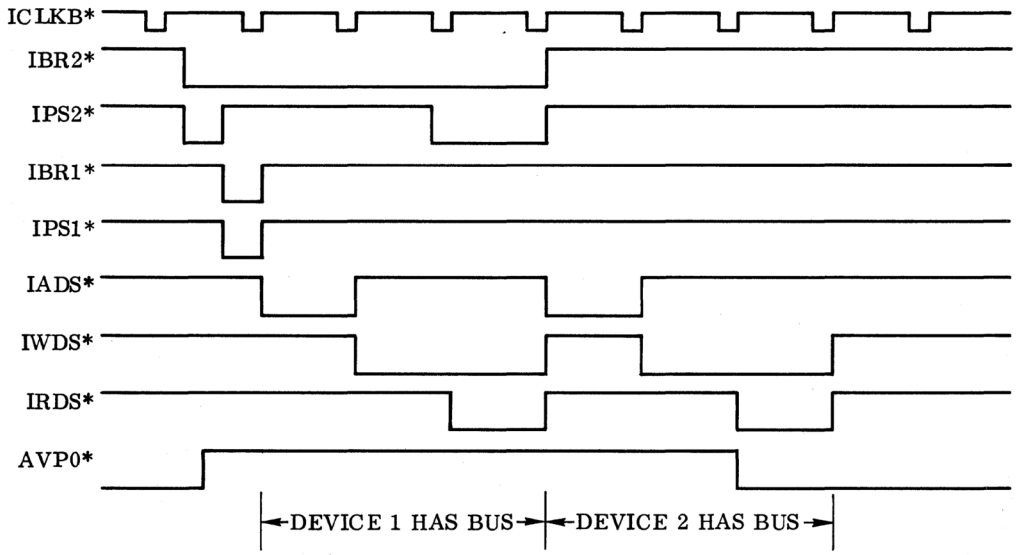


Figure 7-7. Two-level Priority Timing

Chapter 8

IMP-16L MODULE CHARACTERISTICS

8.1 INTRODUCTION

This chapter contains functional descriptions of the following modules:

- CPU Module
- Memory Module
- Operators Control Panel and TTY/IO Interface Module
- Standard Programmers Control Panel Module

Figures and illustrations are provided to show the physical location of components on the PC cards, and tables and text are provided to describe the operational features.

8.2 IMP-16L CPU MODULE

Figure 8-1 shows the physical layout of components on the IMP-16L CPU PC card. Characteristics are as follows:

- 16-bit word length
- 4 16-bit accumulators
- 1 million word-per-second bus capacity
- 16-word 16-bit last-in/first-out stack
- Double-precision arithmetic
- DMA bus structure
- 2 user jump conditions
- 60 instructions
- 4 levels of interrupt
- 4 priority bus ports, expandable
- 5 methods of addressing
- Multiply/divide instructions
- 5 user control flags
- Capacity:

  Approximately one million 16-bit words per second on the System Data Bus
- Modes of Bus Operation:

  Read Memory

  Write Memory

  Read Peripheral

  Write Peripheral

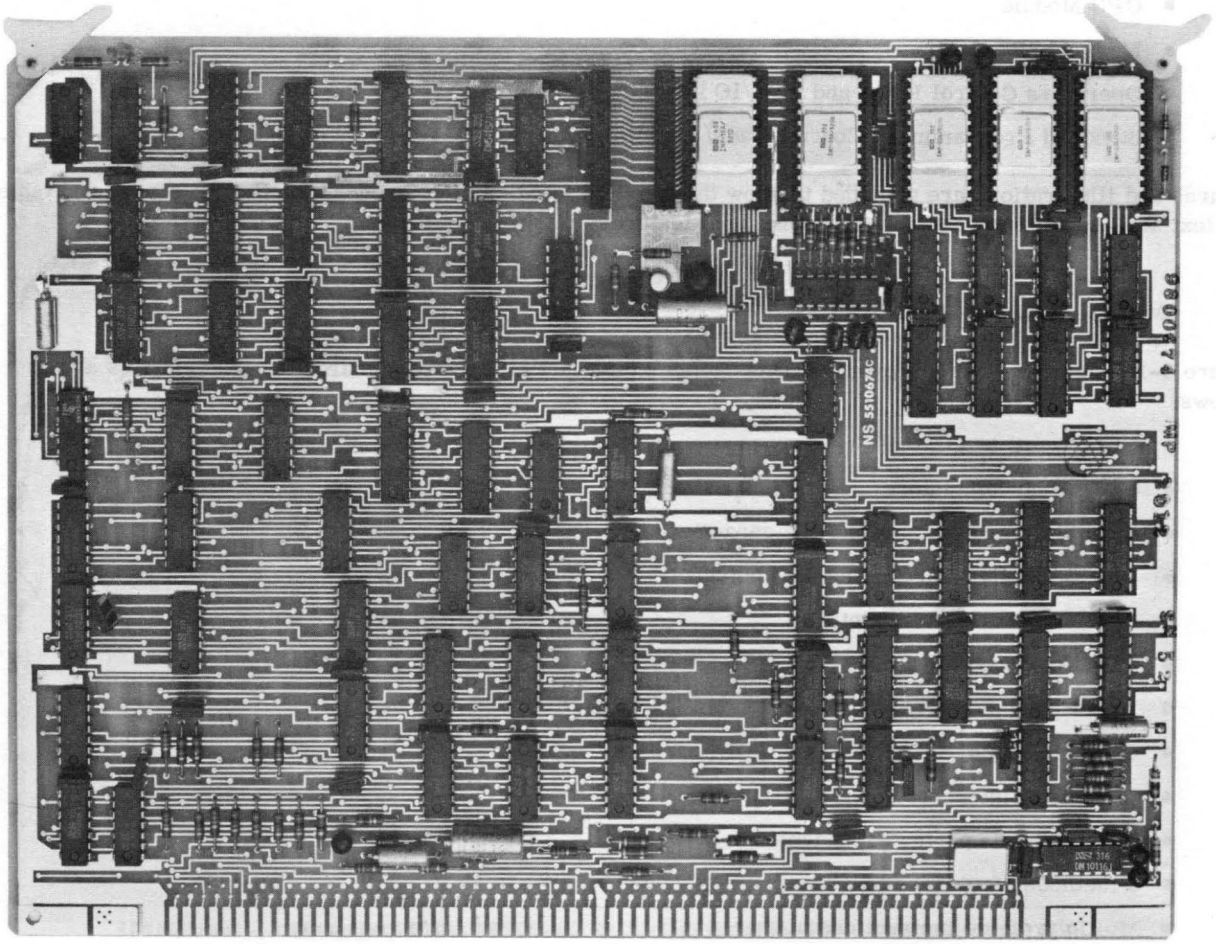


Figure 8-1. IMP-16L CPU Module

- Interface Characteristics:

  16 TRI-STATE® address/data lines time-multiplexed

  TTL compatible control signals

  DMA transfer

  Programmed I/O data transfer

- Environment:

Temperature:	0^{o}C to 60^{o}C operating (ambient)
	-20^{o}C to 100^{o}C nonoperating
Relative Humidity:	0 to 90% with no condensation

- DC Power Requirements at Room Temperature:

+5V ±5%	2.6 amperes maximum
-12V±5%	0.2 ampere maximum

- Description

  The IMP-16L CPU card is designed around four 4-bit MOS/LSI Register and Arithmetic Logic Units (RALUs) using two Control Read-only Memories (CROMs) for executing instructions. The bus structure is a DMA-type with approximately a one million word-per-second capacity. There are four ports and Priority Logic on the CPU card with provisions for expansion. All timing and control signals for bus priority are contained on the CPU card. The data bus is bidirectional TRI-STATE® with address and data time-multiplexed.

  Four levels of external interrupts, each of which may be individually enabled, are provided plus a control panel interrupt and a stack-full interrupt. A master interrupt enable can enable or disable all interrupts except the control panel interrupt.

  Two external Branch-on-condition (BOC) Signals are available to the user. Five external control flags, which may be either set or pulsed, are available for the user. System timing is normally from a crystal-controlled oscillator. Provisions are available via backplane wiring to use an external clock. Figure 8-2 is a functional block diagram of the IMP-16L CPU Module.

8.3 MEMORY MODULE

Figure 8-3 shows the physical layout of components on the IMP-16L Memory PC card. Characteristics are as follows:

- 4096 words by 16 bits
- Modular expandable
- Modular interchangeability
- Up to 512 16-bit words of ROM
- Two power supply voltages
- Refresh Logic on card
- Directly addressable to 32K words
- Description

  The IMP-16L memory system is designed to meet the needs of the IMP-16L microcomputer. This memory provides high reliability and performance at a low cost through the use of MOS/LSI technology. This memory system has a capacity of 4096 16-bit words on each PC card. Each PC card

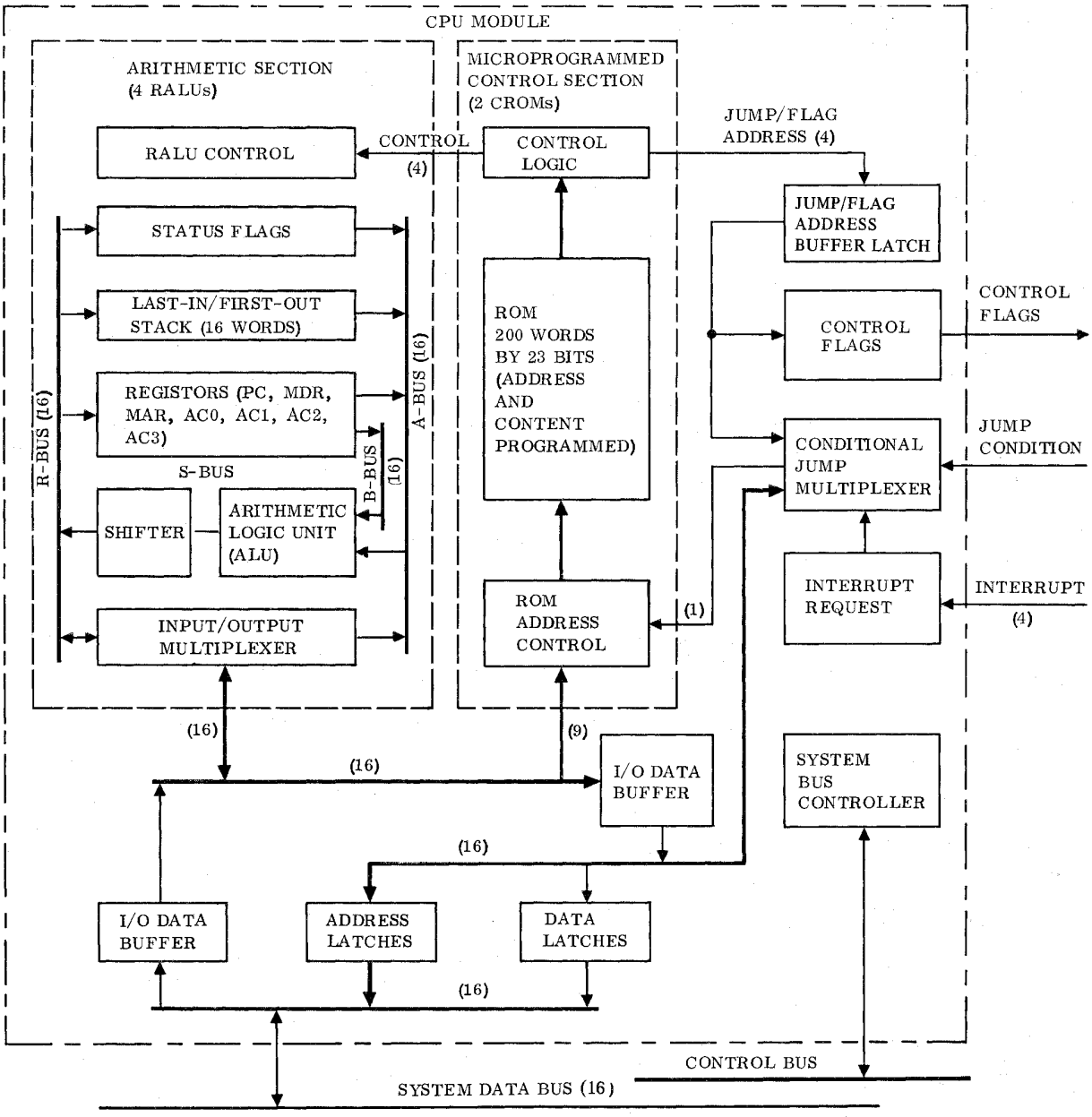


Figure 8-2. Functional Block Diagram of IMP-16L CPU Module

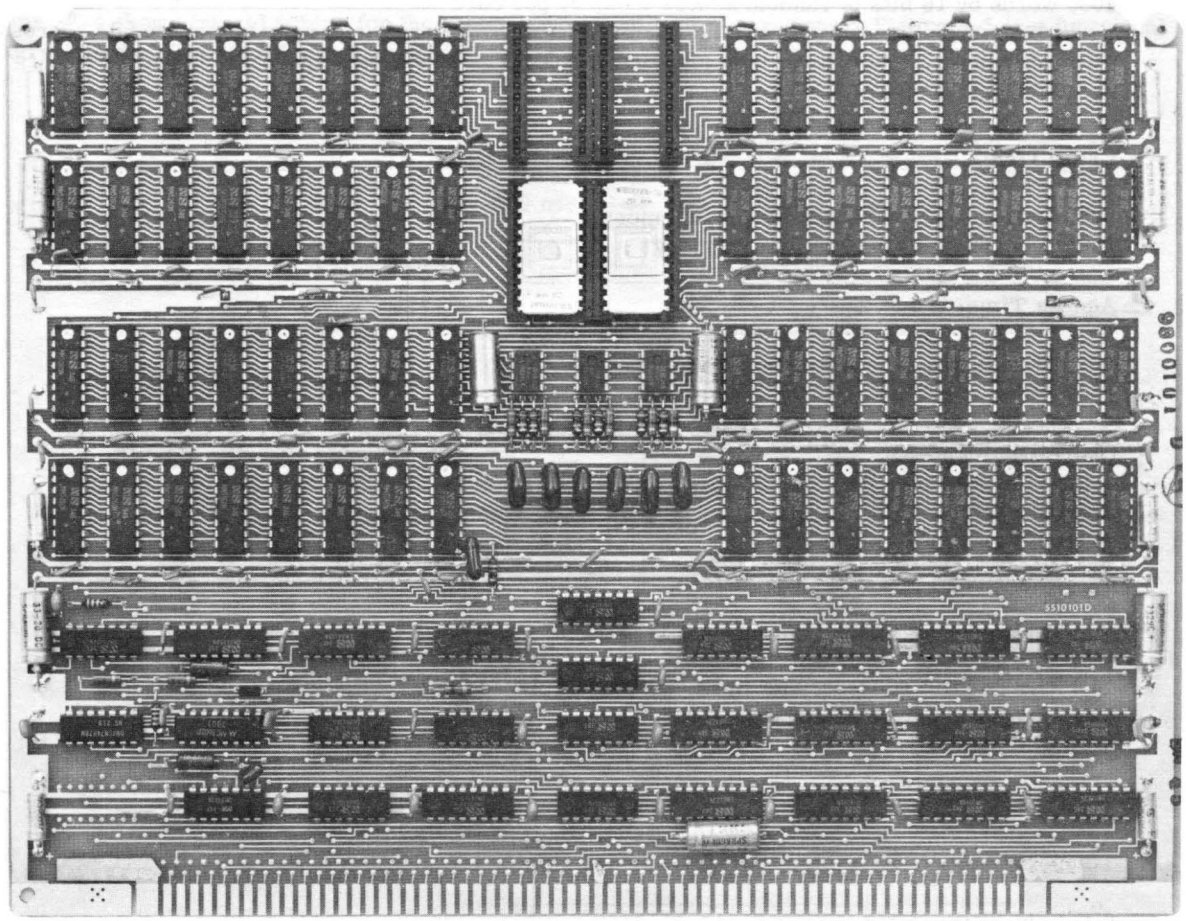


Figure 8-3. IMP-16L Memory Module

is a complete memory system with TRI-STATE® TTL interface and Refresh Logic.

- Dimensions:

Memory board (4K x 16):	11.0 inches long 8.5 inches wide 0.50 inch high

- Capacity:

4096 words by 16 bits of random access memory per card
Up to 32K words directly addressable memory using backplane jumpers to select module address
Two 256-word by 16-bit Read-only Memories (ROMs) using address space $FE00_{16}$ to $FEFF_{16}$ and $FF00_{16}$ to $FFFF_{16}$

- Cycle Time:

900 nanoseconds for random access memory
1500 nanoseconds for read-only memory

- Access Time:

600 nanoseconds for random access memory
1000 nanoseconds for read-only memory

- Operational Modes:

Read (nondestructive)
Write
Refresh

- Interface Characteristics:

16 TRI-STATE® address/data lines, time-multiplexed
TTL compatible control signals
TTL Refresh Logic

- Environment:

Temperature:

$0^{o}C$ to $60^{o}C$ operating (ambient)
$-20^{o}C$ to $100^{o}C$ nonoperating

Relative humidity:

0 to 90% with no condensation

- DC Power Requirements at Room Temperature:

+ 5V ± 5%	2.3 amperes maximum
-12V ± 5%	1.1 amperes maximum
+ 6.85V (1.85V ± 0.25V more positive than + 5V supply)	5 amperes maximum

- Options

ROMENI Signal:

To use the read-only memories, the ROMENI Signal must be connected to the ROMEN Signal. This connection is made on the backplane.

KSEL Signal:

To select the address space for the read/write memory (in increments of 4K), connect the corresponding KSEL X*' Signal (where X denotes which 4K) to the KSEL* input. This connection is made on the backplane.

Figure 8-4 shows typical timing for the memory read and write operations. Figure 8-5 is a functional block diagram of the IMP-16L Memory Module.

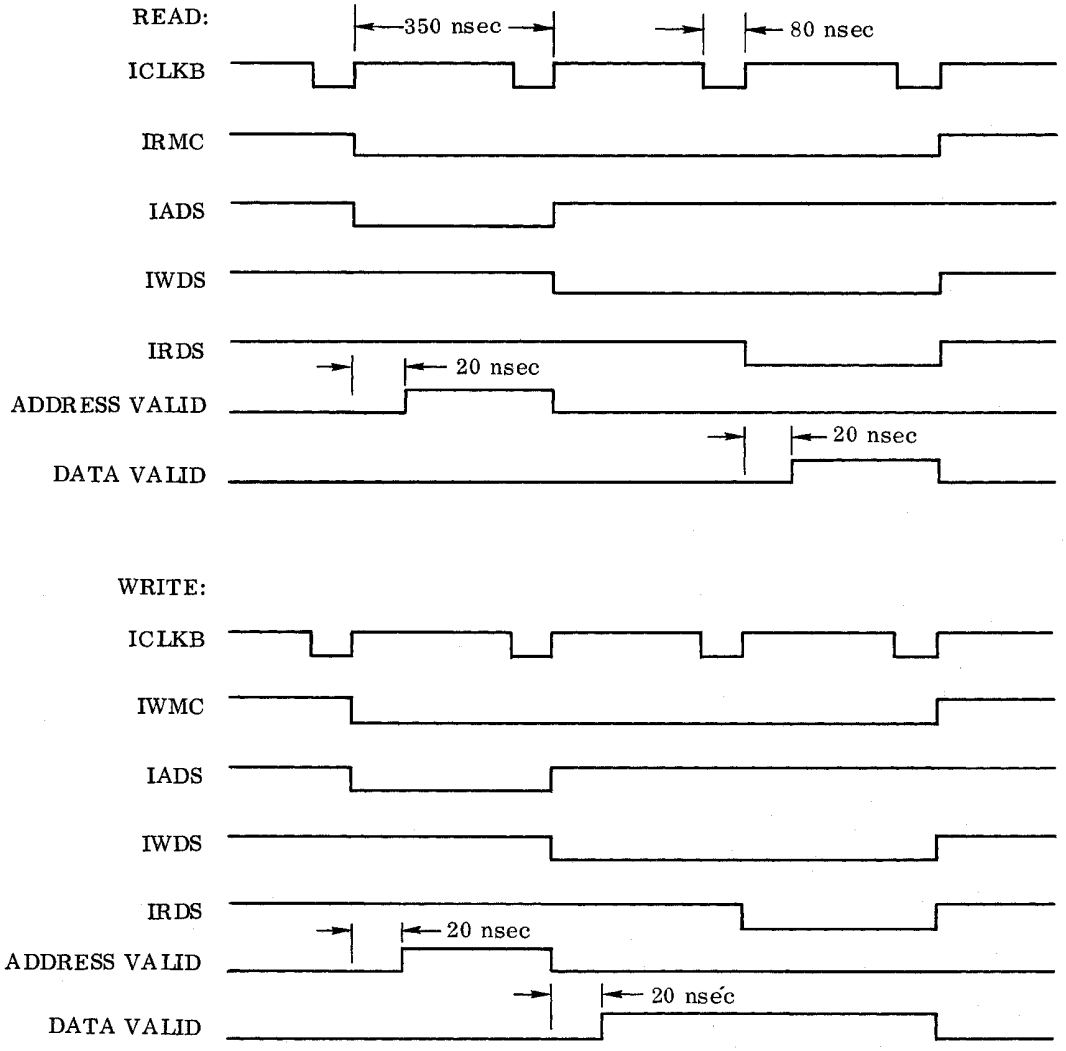


Figure 8-4. Memory Read and Write Typical Timing Diagrams

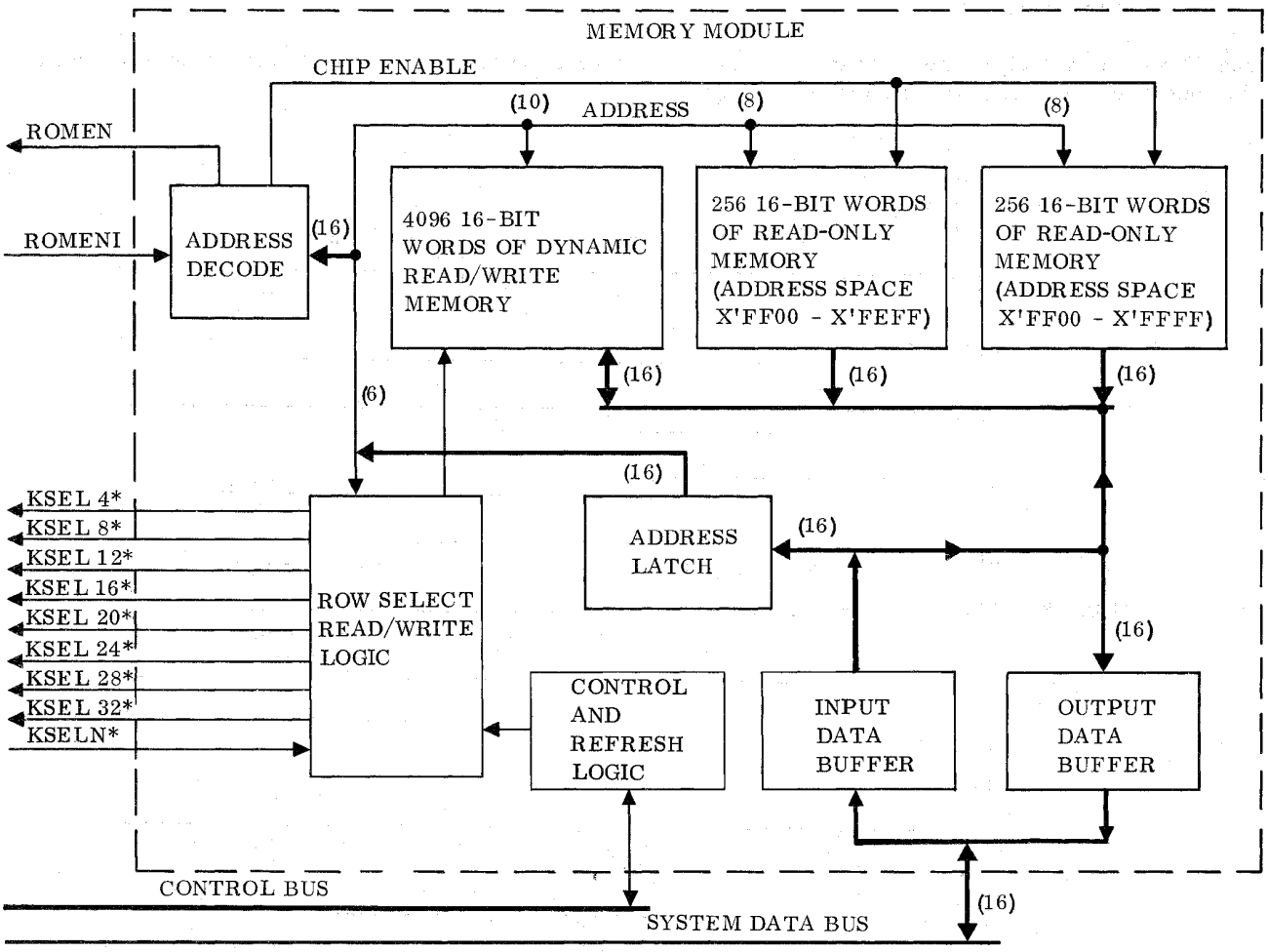


Figure 8-5. Functional Block Diagram of IMP-16L Memory Module

8.4 IMP-16L OPERATORS CONTROL PANEL AND TTY/IO INTERFACE MODULE

Figure 8-6 shows the physical layout of components on the IMP-16L Operators Control Panel and TTY/IO Interface PC card. Characteristics are as follows:

- 16-bit bidirectional Programmers Control Panel interface bus
- Single instruction step logic
- Full duplex, serial Teletype (programmed control) interface
- Optional highspeed Teletype current loop input
- Teletype interrupt
- Four extra inverters available to the user on the backplane
- Programmers Control Panel display refresh interrupt
- Level zero interrupt (highest priority)
- Operators Control Panel Description

  The IMP-16L Operators Control Panel is designed to interface the CPU to the Programmers Control Panel and to provide basic operational features such as: Run, Halt, Initialize, Lock, and Load Program functions. Logic is included to debounce the RUN, LOCK, INIT and LOAD PROGRAM Push-buttons and to insure that they are serviced exactly once each depression.

  To make the panel functions transparent to the user, the control panel is designed as the highest priority level zero interrupt source. The control panel interrupt is not disabled when INEN is set low. However, the control panel interrupt may be disabled by setting a logic '1' on the Control Inhibit (CPINH) Line of the CPU.

  During the Halt Mode, the control panel issues a continuous interrupt to the CPU. When recognized by the CPU, the panel hardware gates an X'03FD (or X'03FC if there is no Programmers Control Panel) onto the bus to be executed as the next instruction (see paragraph 6.3.3). This is a Jump-to-subroutine-implied Instruction (JSRI) to location X'FFFD (X'FFFC), - the start of the Control Panel Service Routine. CPU hardware ignores all interrupts until panel service is completed (see paragraph 5.3).

  In the Run Mode, the Control Panel Module has a monostable retriggerable one-shot multivibrator that interrupts approximately 10 times per second to refresh the programmers' display. Switch depressions such as Single-instruction, Load Data, Load Program, and Increment Memory override the one-shot interrupt and cause an immediate interrupt to respond to the switch depression.

- Operational Functions

RUN	Panel LOCK
HALT	SINGLE INSTruction
INITialize	LOAD PROGram

- TTY/IO Interface

  The TTY/IO interface is a general-purpose, program-controlled, full-duplex serial interface designed for a 20-milliampere current loop transmission line. Logic is provided for read, write, interrupt enable and clear, interrupt status response, and Paper Tape Reader relay control (to be used if the Teletype is equipped with a Paper Tape Reader relay). One backplane option is provided for an auxiliary jump. Grounding the Auxiliary Jump Pin (AUXJ*) on the backplane causes the interrupt instruction to change from X'03FD to X'03F1 (from JSRI X'FFFD to JSRI X'FFF1). A Jump Instruction to X'FE01 is located at X'FFF1 in the Control Panel Service Routine. This instruction option enables the user to develop special purpose firmware located in the 256-word ROM space X'FF00 through X'FEFF.

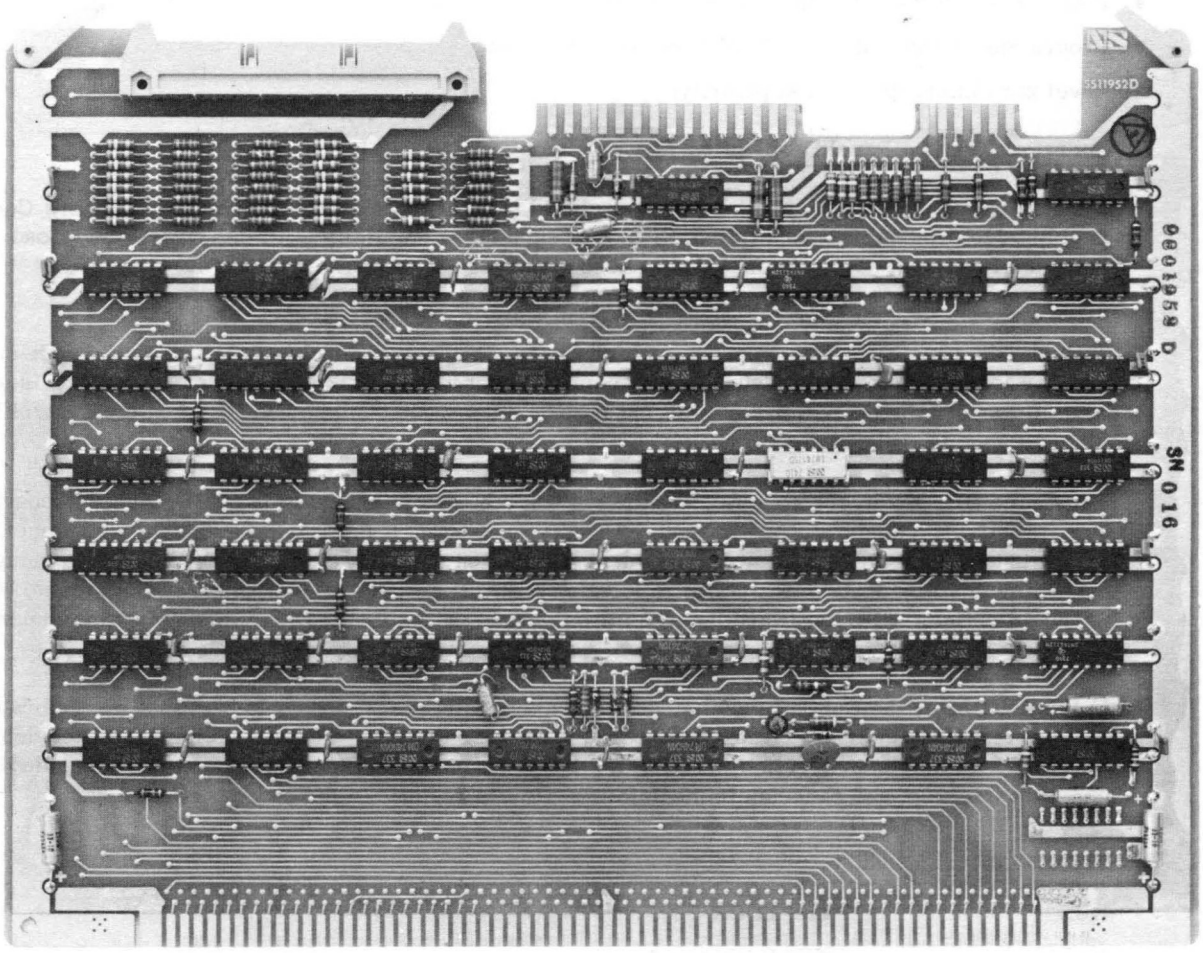


Figure 8-6. IMP-16L Operators Control Panel and TTY/IO Interface Module

(This option works only if the programmers panel is installed. If it is absent and the auxiliary jump line is low, the Interrupt Instruction is X'03F0. Therefore, executing the instruction at X'FFF0 is illegal. The SCPIE Flag on the CPU (flag 13) must be pulsed to enable the control panel interrupt.

- Interface Characteristics

  16-bit bidirectional open collector Programmers Control Panel interface bus capable of driving up to 10 feet over a 3M-Scotchflex 28 AWG flat cable.

  Open collector control signal drivers to Programmers Control Panel. Programmed TTY serial data transfer to CPU over System Data Bus.

- Environment

  Temperature:

  $0^{o}C$ to $60^{o}C$ operating ambient
  $-20^{o}C$ to $100^{o}C$ nonoperating

  Relative Humidity:

  0 to 90% with no condensation

- DC Power Requirements at Room Temperature

+ 5 volts ± 5%	2.0 amperes maximum
-12 volts ± 5%	0.25 ampere maximum

8.5 STANDARD PROGRAMMERS CONTROL PANEL MODULE

The standard Programmers Control Panel Module is shown in figure 8-7. Characteristics are as follows:

- 16-bit bidirectional interface bus
- Eleven-position Display Selector Switch
- Single Instruction, Increment Memory Address, and Load Data functions
- LED Display Lamps
- Description

  The standard Programmers Control Panel is designed to enable the user access to the CPU registers, memory, stack, flags, and to permit program debug operations such as Single Instruction and Increment Memory Address. Logic is minimized to enhance universal applications of the panel.

  Data is transmitted between the Programmers Control Panel Module and the Operators Control Panel PC card under program control.

- Environment

  Temperature:

  $0^{o}C$ to $50^{o}C$ operating ambient
  $-20^{o}C$ to $55^{o}C$ nonoperating

Relative Humidity:

0 to 90% with no condensation

- DC Power Requirements at Room Temperature

(Power supplied over separate power cable)

+5 volts ±5% 2 amperes maximum

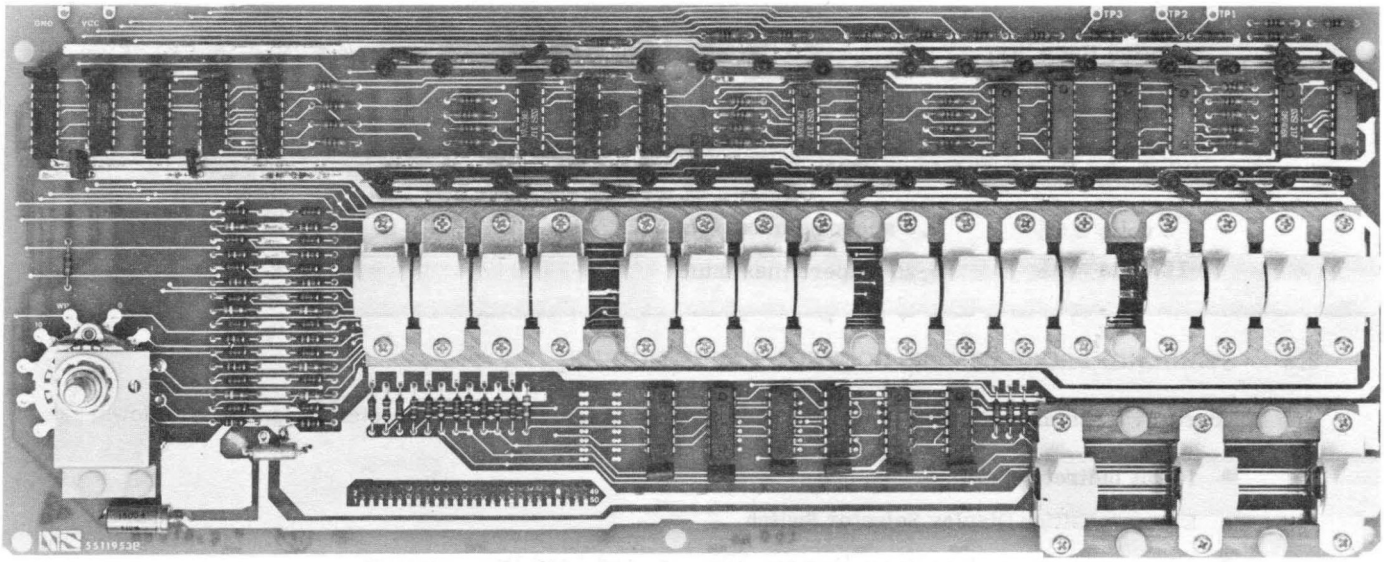


Figure 8-7. Standard Programmers Control Panel Module

Chapter 9

IMP-16L OPTIONS

9.1 INTRODUCTION

This chapter of the users manual describes current options of the IMP-16L.

9.2 CARD READER

9.2.1 General Description

The Card Reader Subsystem, figure 9-1, is designed to interface to the IMP-16L DMA system. The Card Reader inputs a card at a time directly into memory. Data may be entered in either of two formats as selected by the processor, or input in one format under subsystem control when used in the Bootstrap Mode.

9.2.2 Card Reader Characteristics

The subsystem is designed to accept data from the Documation M300L Card Reader (or equivalent). This reader inputs 300 cards per minute over a TTL interface. (See paragraph 9.2.11 for Card Reader timing and control information.)

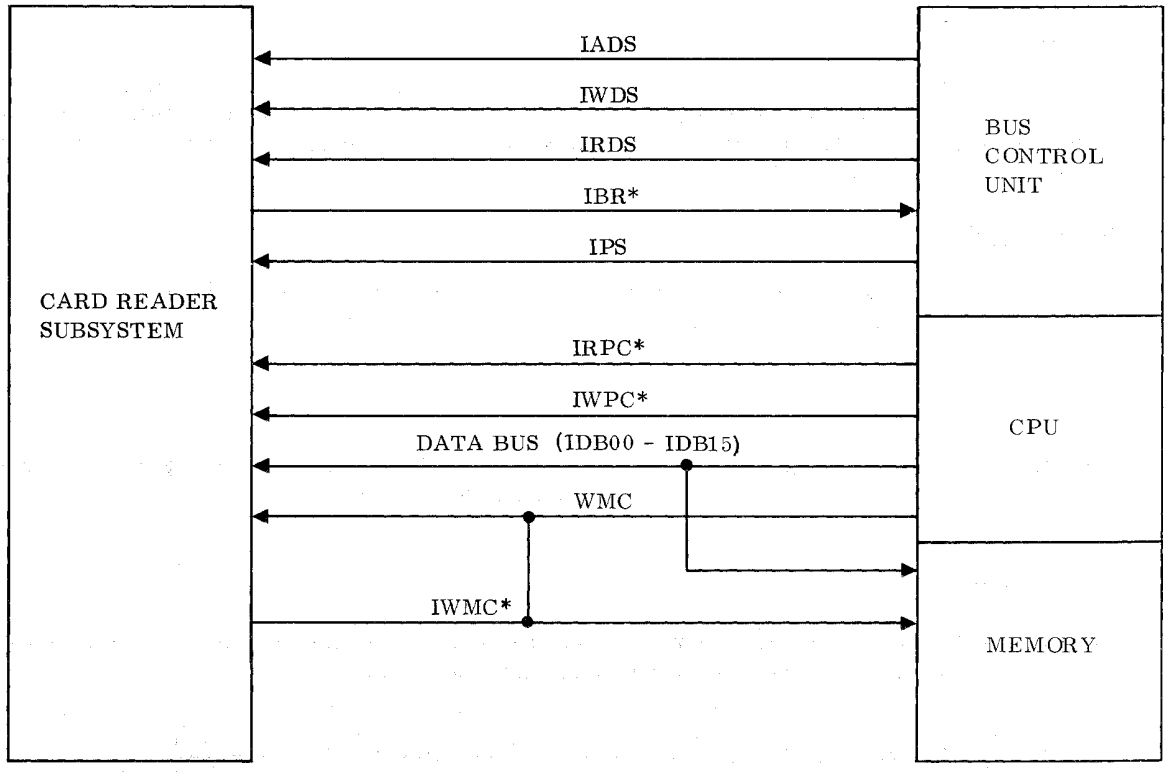


Figure 9-1. Subsystem/Processor System Interface

Card Reader specifications are as follows:

• Read Rate	300 cards per minute
• Card Type	Standard 80-column EIA card
• Hopper/Stacker	1000-card capacity
• Light Source	Infrared light emitting diodes
• Read Station	Photo transistor, 12 bits parallel
• Electronics	5-volt TTL interface
• Logic Convention	Positive logic convention
• Voltage	115 VAC ±10%, single phase, 60 Hz or 230 VAC ±10%, single phase, 50 Hz
	NOTE For 115 VAC, 60 Hz, Order Number IMP-00/825; and for 230 VAC, 50 Hz, Order Number IMP-00/825E.
• Power	1650 VA maximum starting load for 3 seconds 600 VA maximum running load

9.2.3 Modes of Operation

There are two basic modes of operation.

- Bootstrap Mode

  When the control panel momentary switch (AUX 1) is pressed, one card is read from the Card Reader and the information deposited in Packed format into the first 40 locations in memory. The processor has no control over the subsystem while in this mode. The subsystem reverts to the Normal Mode when the momentary switch is released.

- Normal Mode

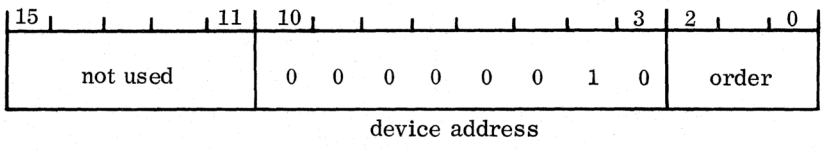
  The subsystem reads directly to memory under processor control when in the Normal Mode. This specification refers to the Normal Mode unless otherwise stated.

9.2.4 Data Format

The data read from a card is presented to the memory in one of two formats as specified by the programmer. Figure 9-2 illustrates the Packed format while figure 9-3 shows the Standard format.

9.2.5 Command Format

As with all RINs and ROUTs, the command word is presented to the subsystem from the processor with ADS during an RPC or WPC operation. The command word format is as follows.



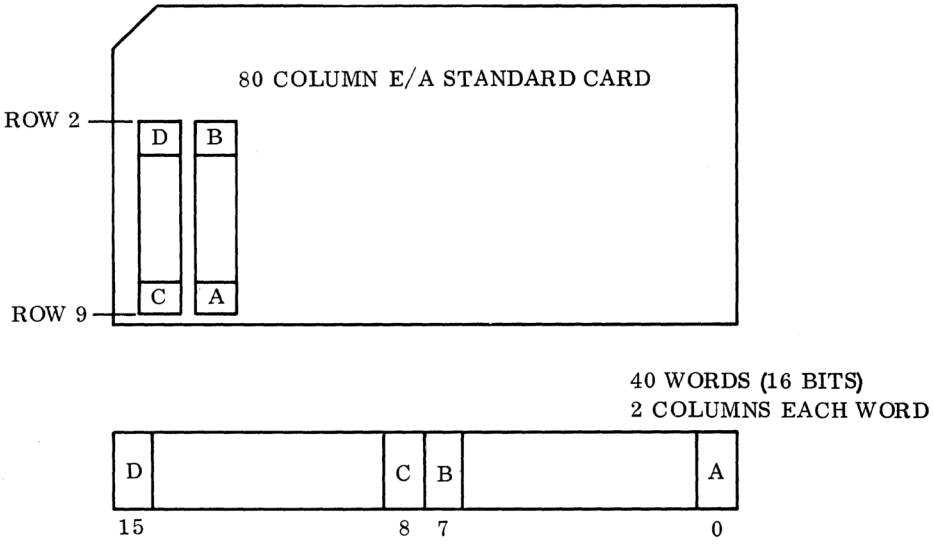


Figure 9-2. Packed Format

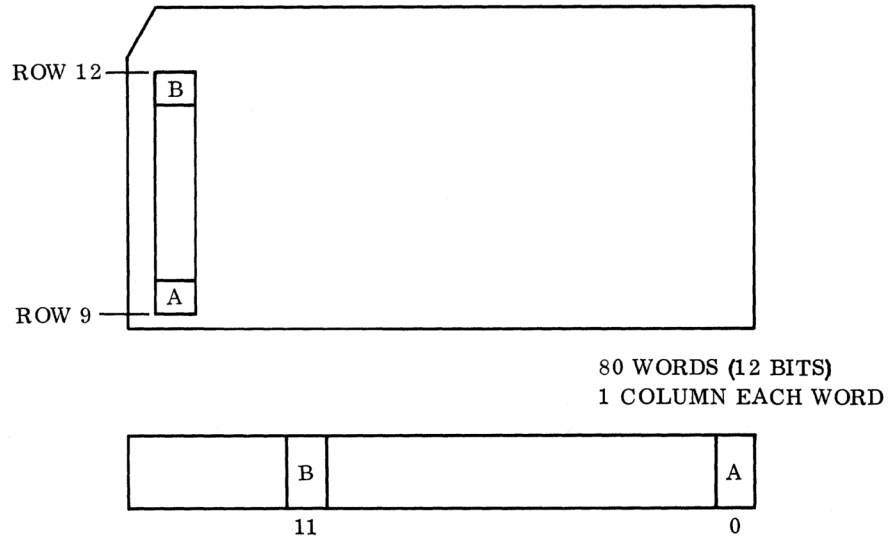


Figure 9-3. Standard Format

The Card Reader Subsystem is assigned address X'02.

Orders recognized by the Card Reader Subsystem fall into two classes. The input class refers to those orders which require the subsystem to transfer data to the processor. Orders falling into the output class require transfers from the processor to the subsystem. The two classes are given below.

Input Class:

Read Status	(RIN 1)
Interrupt Select	(RIN 6)

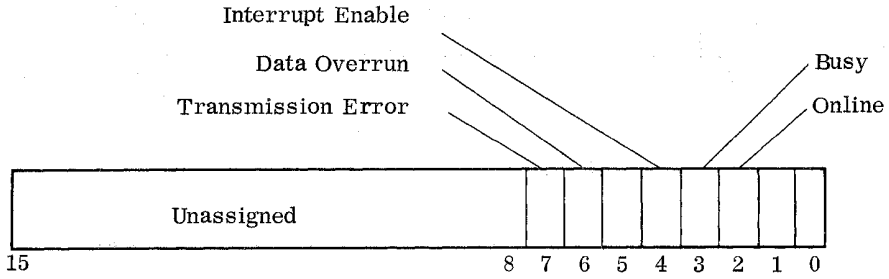
Output Class:

Read Standard	(ROUT 2)
Read Packed	(ROUT 3)
Set Control	(ROUT 1)
Reset	(ROUT 5)

9.2.6 Command Data Word Formats

A data word exchange between the processor and the Card Reader Subsystem occurs each time an order is issued to the subsystem. The format of the data word corresponding to each order follows.

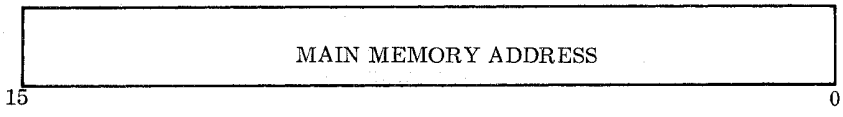
The Read Status Order data word returns the subsystem status word to AC0.



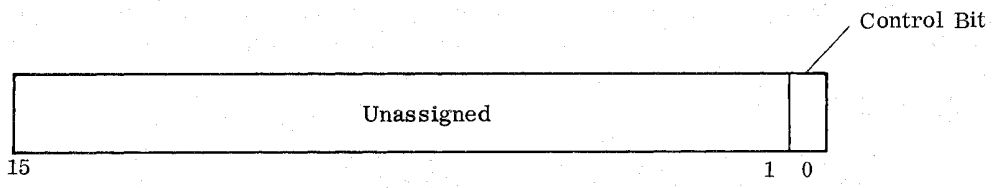
The transmission error bit is set if the Card Reader senses a light/dark read check. A data-overrun condition is indicated by bit 6 being set. Bit 4 is a logic '1' if the Card Reader is enabled for interrupts and logic '0' if interrupts are disabled. Busy (bit 3) is a logical '1' only while the subsystem is actually in the process of reading a card. Online is indicated whenever the Card Reader power is applied; cards are in the input hopper; and there are no reader error conditions.

Execution of the Interrupt Select 1 Order causes the Card Reader Subsystem to drive bit line 2 (only) of the System Data Bus. This bit line is driven to a '1' if requesting an interrupt or to a '0' if the subsystem is not in an interrupt state.

The Read Standard and Read Packed data words contain a 16-bit main memory address. This address is transferred to the Card Reader from the processor AC0 where it is deposited in the subsystem Memory Address Counter.



The set control order data word in AC0 contains only one bit which concerns the Card Reader Subsystem. If bit 0 is a logical '1' the Interrupt Enable Control is set to the enabled state. If bit 0 is a logical '0', the control is set to the disabled state.



The Reset Order has no data word associated with it.

9.2.7 Operation

Figure 9-4 is a block diagram of the Card Reader/Subsystem Interface. Operation of the Card Reader Subsystem (except for the Bootstrap Mode) is controlled by the processor. This subsystem responds to the six orders issued by the processor in the following way.

Execution of the Read Status Order causes the subsystem to gate its status information into AC0 in the processor as if it were a word of data. The temporary status bits (error condition bits 6 and 7) are reset with the next Read Standard/Packed or Reset operation.

The Interrupt Select 1 Order causes the Card Reader Subsystem to drive one (only) Data Bus Line indicating the interrupt status of the subsystem. This order allows the processor to test the interrupt status of 16 devices simultaneously in order to determine servicing priority. Completion of this order causes the Interrupt Enable Flag to be cleared.

The Read Standard operation gates the 16-bit starting memory address to the Card Reader Subsystem. The subsystem then reads a card and deposits the data in memory in the Standard format beginning at the starting address and ending 80 memory locations later. The subsystem then signals the processor that it has completed the operation by issuing an interrupt.

The Read Packed operation is identical to the Read Standard operation except for the data format. The 80 card columns are packed into 40 words of memory.

Execution of the Set Control Order causes the Card Reader Subsystem to sense the appropriate Data Bus Line and either set or reset the Interrupt Enable Control Flip-flop. This allows interrupts to be enabled or disabled under program control.

The Reset Order allows the processor to clear the Card Reader Subsystem under program control.

9.2.8 Error Condition Operation

Card Reader error conditions are indicated to the processor by the status bits. A Read check sets the transmission error status bit. Motion or Hopper checks are indicated with a Reset Online status bit.

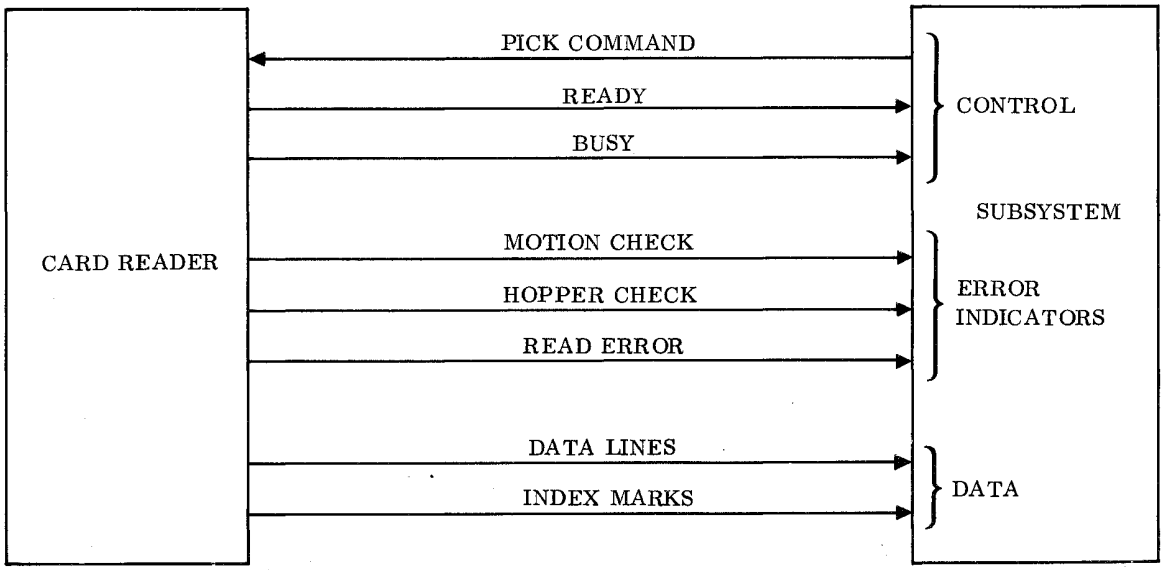


Figure 9-4. Card Reader/Subsystem Interface

9.2.9 Interrupt

The Card Reader Subsystem interrupts the processor under the following conditions.

- Motion or Hopper check
- Upon completion of a card read (whether or not a Read check occurred)

9.2.10 Interface Signal Glossary

- Control Signals

Pick Command	Command issued to Card Reader which initiates the card pick cycle
Ready	Indicates reader is ready to honor a Pick Command
Busy	Indicates card is in process of being read

- Error Alarm Signals

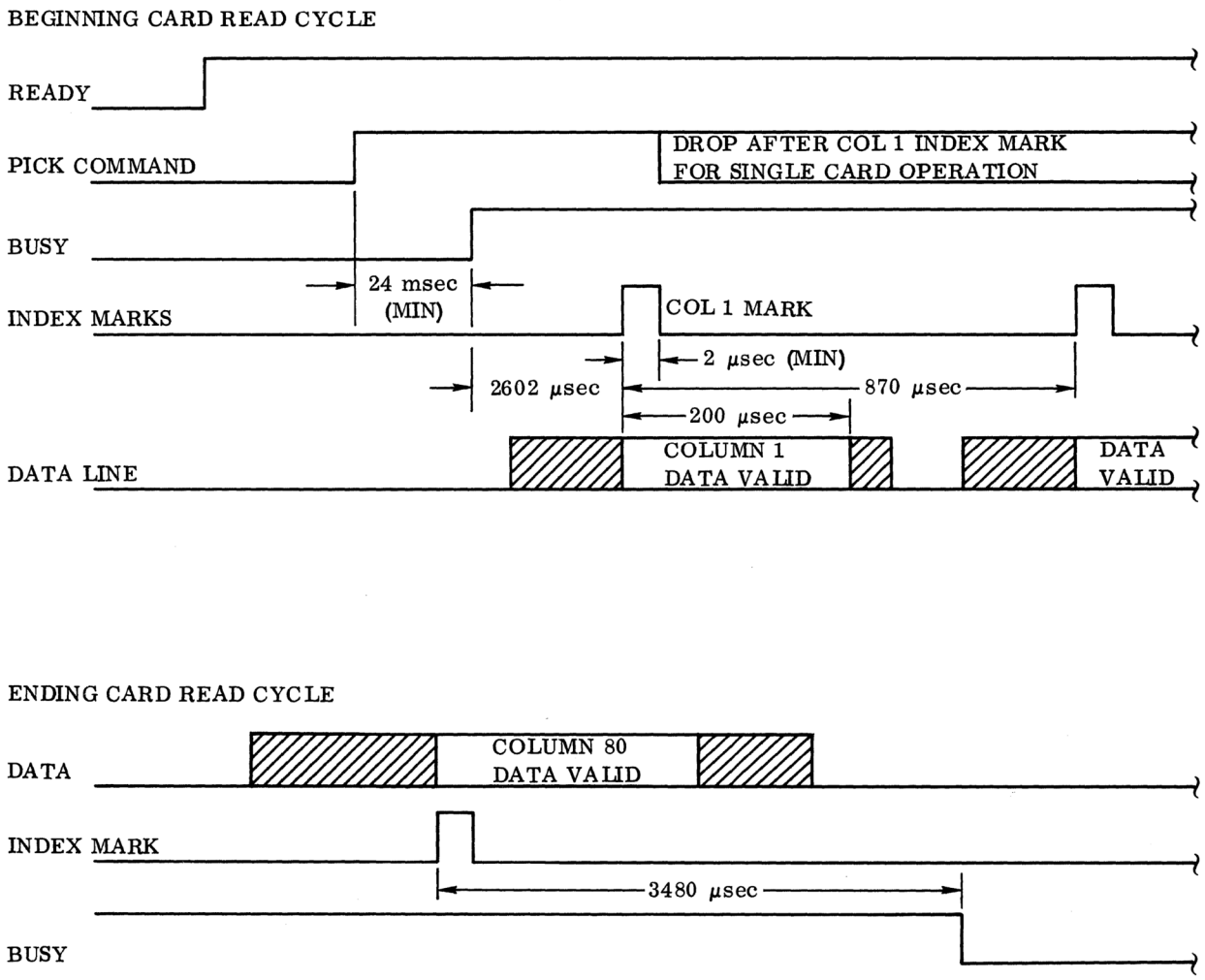
Read Error	Indicates card being read failed the light/dark check
Motion Check	Indicates that a card has hung up in the track or cannot be picked
Hopper Check	Indicates either input hopper empty or output stacker full

- Data Signals

Index Mark	Pulse generated with each column read signaling the presence of data
Data Lines	12 parallel lines corresponding to the holes in the card

9.2.11 Read Card Timing

Figure 9-5 illustrates the timing of the beginning and ending of a card read cycle.



NOTE: IF PICK COMMAND HAS REMAINED TRUE THROUGH CYCLE THEN A NEW PICK SEQUENCE BEGINS 1 μsec AFTER THE FALL OF BUSY

Figure 9-5. Beginning Card Read Cycle and Ending Card Read Cycle Timing

Chapter 10

SYSTEM VERIFICATION

10.1 INTRODUCTION

This chapter contains procedures that can be used to verify the proper basic operation of an IMP-16L. If, during verification, any difference from the indicated result is encountered, refer to chapter 2 for a functional description of the CPU and memory cards. Refer to chapter 4 for a description of the controls and indicators on the Programmers and Operators Control Panels.

10.2 INITIAL SYSTEM VERIFICATION

10.2.1 Control Panel Verification

To verify Control Panel Operation, proceed as follows:

1. On the Operators Control Panel, set POWER Keyswitch to OFF and PANEL Keyswitch to LOCK.
2. Connect IMP-16L to appropriate input power (115 ± 10 vac, 60 Hz standard; 220 vac and 50 Hz optional).
3. Set POWER Keyswitch to ON; observe that blower operates and HALT Indicator is lighted.
4. Set Display Selector Rotary Switch to AC0; observe that DATA DISPLAY Indicators display X'0000.
5. Set Display Selector Rotary Switch to AC1; observe that DATA DISPLAY Indicators display X'0000.
6. Set Display Selector Rotary Switch to AC2; observe that DATA DISPLAY Indicators display X'0000.
7. Set Display Selector Rotary Switch to AC3; observe that DATA DISPLAY Indicators display X'0000.
8. Set Display Selector Rotary Switch to PC; observe that DATA DISPLAY Indicators display X'FFFE.
9. Set Display Selector Rotary Switch to NEXT INST; observe that DATA DISPLAY Indicators display X'2000.
10. Set Display Selector Rotary Switch to STACK; observe that DATA DISPLAY Indicators display X'0000.
11. Set PANEL Keyswitch to UNLOCK.
12. Set Display Selector Rotary Switch to MEM ADDR.
13. Set Data Switches to X'FFFF.
14. Press and release LOAD DATA Switch; observe that DATA DISPLAY Indicators display X'FFFF.
15. Set Data Switches to X'0000.
16. Press and release LOAD DATA Switch; observe that DATA DISPLAY Indicators display X'0000.
17. Set Data Switches to X'AAAA.

18. Press and release LOAD DATA Switch; observe that DATA DISPLAY Indicators display X'AAAA.

19. Set Data Switches to X'5555.

20. Press and release LOAD DATA Switch; observe that DATA DISPLAY Indicators display X'5555.

21. Set Display Selector Rotary Switch to AC0. Set Data Switches to X'0001. Press and release LOAD DATA Switch.

22. Set Display Selector Rotary Switch to AC1. Set Data Switches to X'0002. Press and release LOAD DATA Switch.

23. Set Display Selector Rotary Switch to AC2. Set Data Switches to X'0004. Press and release LOAD DATA Switch.

24. Set Display Selector Rotary Switch to AC3. Set Data Switches to X'0008. Press and release LOAD DATA Switch.

25. Set Display Selector Rotary Switch to PC. Set Data Switches to X'0010. Press and release LOAD DATA Switch.

26. Set Display Selector Rotary Switch to FLAGS. Set Data Switches to X'0020. Press and release LOAD DATA switch.

27. Set Display Selector Rotary Switch to STACK. Set Data Switches to X'0040. Press and release LOAD DATA Switch.

28. Set Display Selector Rotary Switch to AC0; observe that DISPLAY DATA Indicators display X'0001.

29. Set Display Selector Rotary Switch to AC1; observe that DISPLAY DATA Indicators display X'0002.

30. Set Display Selector Rotary Switch to AC2; observe that DISPLAY DATA Indicators display X'0004.

31. Set Display Selector Rotary Switch to AC3; observe that DISPLAY DATA Indicators display X'0008.

32. Set Display Selector Rotary Switch to PC; observe that DISPLAY DATA Indicators display X'0010.

33. Set Display Selector Rotary Switch to FLAGS; observe that DISPLAY DATA Indicators display X'0020.

34. Set Display Selector Rotary Switch to STACK: observe that DISPLAY DATA Indicators display X'0040.

10.2.2 Operational Verification

To verify the system operation in the Run Mode, proceed as follows:

1. Initial control panel settings:

   a. POWER Keyswitch – ON

   b. PANEL Keyswitch – UNLOCK

2. Press and release INIT Pushbutton; observe that HALT Indicator is lighted.

3. Set Display Selector Rotary Switch to MEM ADDR.

4. Set Data Switches to X'0050.

5. Press and release LOAD DATA Switch.

6. Set Display Selector Rotary Switch to MEM DATA; observe that PROGRAM COUNTER-MEMORY ADDRESS Indicators display X'0050.

7. Set Data Switches to X'3081.

8. Press and release LOAD DATA Switch.

9. Press and release INCR MEM ADDR Switch.

10. Load data into memory locations X'0051 through X'0062 by repeating steps 8 and 9, but substitute the data values in the following list for the data value in step 7.

NOTE

The PROGRAM COUNTER-MEMORY ADDRESS Indicators display the current (incremented) memory locations.

Memory Location	Data Value	Operation
X'0051	X'3081	NOP
X'0052	X'3081	NOP
X'0053	X'3081	NOP
X'0054	X'3081	NOP
X'0055	X'3081	NOP
X'0056	X'3081	NOP
X'0057	X'3081	NOP
X'0058	X'3081	NOP
X'0059	X'3081	NOP
X'005A	X'3081	NOP
X'005B	X'3081	NOP
X'005C	X'3081	NOP
X'005D	X'3081	NOP
X'005E	X'3081	NOP
X'005F	X'3081	NOP
X'0060	X'3081	NOP
X'0061	X'2050	JMP X'50

11. Set Display Selector Rotary Switch to PC.

12. Set Data Switches to X'0050.

13. Press and release LOAD DATA Switch; observe that PROGRAM COUNTER-MEMORY ADDRESS Indicators display X'0050.

14. Press and release RUN Pushbutton; observe that RUN Indicator is lighted.

15. Press and release HALT Switch; observe that the HALT Indicator is lighted.

16. Observe data value (memory location) displayed on PROGRAM COUNTER-MEMORY ADDRESS Indicators; verify that data value is between X'0050 and X'0061.

17. Press and release SINGLE INST Switch; observe that data value displayed on PROGRAM COUNTER-MEMORY ADDRESS Indicators is incremented.

18. Set Display Selector Rotary Switch to NEXT INST; observe that data value displayed on DISPLAY DATA Indicators corresponds to data value specified for that memory location in the NOP program.

19. Repeat steps 17 and 18 until all memory locations in the NOP program are examined.

20. Press and release RUN Pushbutton; observe that RUN Indicator lights and HALT Indicator is extinguished.

21. Press and release HALT Pushbutton; observe that HALT Indicator is lighted and RUN Indicator is extinguished.

22. Repeat steps 20 and 21.

23. Press and release RUN Pushbutton; observe that RUN Indicator is lighted.

24. Set PANEL Keyswitch to LOCK.

25. Set Data Switches to a data value of X'0000.

26. Set Display Selector Rotary Switch to MEM ADDR. Press and release LOAD DATA. Observe that PROGRAM COUNTER—MEMORY ADDRESS displays X'0000.

27. Press and release the following switches; observe that RUN Indicator remains lighted.

    a. HALT

    b. LOAD PROG

    c. INIT

    d. RUN

    e. LOAD DATA

    f. SINGLE INST

    g. INCR MEM ADDR (Observe that PROGRAM COUNTER-MEMORY ADDRESS is incremented.)

28. Set PANEL Keyswitch to UNLOCK.

29. Set Display Selector Rotary Switch to PC. Press and release HALT Pushbutton; observe that HALT Indicator is lighted and PROGRAM COUNTER-MEMORY ADDRESS Indicators display a data value between X'0050 and X'0061, inclusive.

10.3 VERIFICATION OF TTY AND CARD READER INTERFACES

10.3.1 Card Reader Interface

The following procedure verifies correct Card Reader operation via the interface:

1. Connect Card Reader to Interface Card (see figure 1-3 in chapter 1).

2. On IMP-16L, set POWER Keyswitch to ON and PANEL Keyswitch to UNLOCK.

3. On Card Reader, press and release POWER Switch so POWER legend is illuminated.

4. Load the two CRBOOT program cards and MEMDIL card deck into Card Reader hopper. Make sure that CRBOOT cards are in front of MEMDIL card deck.

5. Press and release INIT Switch.

6. On Card Reader, press and release RESET Switch. Wait until RESET Switch illuminates green.

7. Press and release AUX1 Switch. (One card should be read.)

8. After card is read, press and release RUN Switch.

9. When Card Reader stops reading card deck (all cards should be read), set IMP-16L Display Selector Rotary Switch to MEM ADDR.

10. Set Data Switches to value (hexadecimal) of memory address containing known data that was loaded via Card Reader.

11. Press and release LOAD DATA Switch.

12. Set Display Selector Rotary Switch to MEM DATA. Verify that DATA DISPLAY Indicators display correct data value for associated memory address.

Card Reader interface verification is complete.

10.3.2 TTY Interface

The following procedure verifies correct TTY operation via the interface:

1. Connect TTY to Control Panel Interface Card connector (see figure 1-3 in chapter 1).

2. On IMP-16L, set POWER Keyswitch to ON and PANEL Keyswitch to UNLOCK.

3. On Teletype, turn Power Switch to LINE.

4. Press and release INIT Switch.

5. Insert leading edge of paper tape MEMDIL into TTY Tape Reader.

6. Press and release LOAD PROG Switch.

7. Set TTY Tape Reader to START. Verify that tape reads but does not echo to TTY printer.

8. Verify that HALT Switch is illuminated and machine halts when tape has been read into machine.

9. When tape has been read, set Display Selector Rotary Switch to MEM ADDR.

10. Set Data Switches to value (hexadecimal) of memory address containing known data that was loaded via paper tape.

11. Press and release LOAD DATA Switch.

12. Set Display Selector Rotary Switch to MEM DATA. Verify that DATA DISPLAY Indicators display correct data value for associated memory address.

TTY Interface verification is complete.

This concludes the initial system verification. Further system verification can be accomplished by using the diagnostic programs.

10.4 DIAGNOSTIC PROGRAMS

The diagnostic programs provided with the system include a CPU diagnostic (CPUXDI) and a memory diagnostic (MEMDIL). The following paragraphs provide a brief description of the diagnostics and present the loading procedures, via both Paper Tape and Card Reader, for the programs. In addition, the normal operating sequence for the programs is presented after the loading procedures. For detailed program information, refer to the program listings supplied with the system.

10.4.1 CPU Diagnostic

The CPUXDI exercises the IMP-16L to verify the reliable performance of all CPU hardware functions. CPUXDI assumes that a limited amount of CPU hardware, including the extended instruction set, is functional and, then, proceeds to exercise the CPU.

The CPUXDI program permits program control parameters to be entered into CPU Accumulators 0 through 3. The type of program control effected is determined by the accumulator entered and the parameter value loaded into that accumulator. The program control parameters are entered into the IMP-16L after the CPUXDI is loaded. The available program control parameters, together with loading procedures for each, are as follows:

Parameter:

Start Test Number

Location:

AC0

Range:

Start Test and End Test maximum range limits:

X'0001 through X'0057 – Basic Instruction Set
X'0058 through X'0077 – Extended Instruction Set

Loading Instructions:

1. Set Display Selector Rotary Switch to AC0.
2. Set Data Switches to number of desired starting test.
3. Press and release LOAD DATA Switch.

Parameter:

End Test Number

Location:

AC1

NOTE

The End Test Number must be greater than the Start Test Number previously loaded into AC0.

Loading Instructions:

1. Set Display Selector Rotary Switch to AC1.

2. Set Data Switches to number of desired ending test.

3. Press and release LOAD DATA Switch.

Parameter:

Looping Mode

Location:

AC2

Range:

X'0000 no looping
X'0001 enables unconditional looping
X'0002 enables unconditional looping on test that detects error
X'0003 enables looping on error condition

NOTE

If none of the aforementioned parameters are loaded into AC2, no looping on an individual test can occur.

Loading Instructions:

1. Set Display Selector Rotary Switch to AC2.

2. Set Data Switches to parameter value of desired looping mode as listed under "Range".

3. Press and release LOAD DATA Switch.

Parameter:

Continuous Testing Mode

Location:

AC3

Range:

X'FFFF (-1) enables continuous execution of the selected tests (one-by-one) until program execution is completed or an error is detected.

Loading Instructions:

1. Set Display Selector Rotary Switch to AC3.

2. Set Data Switches to X'FFFF.

3. Press and release LOAD DATA Switch.

NOTE

If, after loading CPUXDI into the IMP-16L, none of the aforementioned parameters are stored in Accumulators 0 through 3, the default parameters entered by loading CPUXDI are as follows:

AC0 – X'0001
AC1 – X'0077
AC2 – X'0000
AC3 – X'0000

10.4.2 CPUXDI Loading Via Card Reader

The following procedure outlines the steps required to load CPUXDI via a Card Reader:

1. Set IMP-16L POWER Keyswitch to ON and PANEL Keyswitch to UNLOCK.
2. On Card Reader, press and release POWER Switch so POWER legend is illuminated.
3. Place two CRBOOT program cards and CPUXDI card deck into Card Reader hopper. Make sure that CRBOOT cards are in front of CPUXDI card deck.
4. Press and release INIT Switch.
5. On Card Reader, press and release RESET Switch. Wait until RESET Switch illuminates green.
6. Press and release AUX 1 Switch. Observe that first CRBOOT card is read by Card Reader.
7. Press and release RUN Switch.
8. When Card Reader halts, set IMP-16L Display Selector Rotary Switch to PC; observe that DATA DISPLAY Indicator displays X'0135.
9. Perform CPUXDI normal operating sequence of paragraph 10.4.4. Start at step 4 of the procedure.

10.4.3 CPUXDI Loading Via Paper Tape Reader

The following procedure outlines the steps required to load CPUXDI via a Paper Tape Reader:

CAUTION

Since the Paper Tape Reader and TTY keyboard are simultaneously enabled, care should be exercised not to input data via the keyboard when a paper tape is being read. If both keyboard and reader simultaneously enter data to the IMP-16L, the results are unpredictable.

1. Set IMP-16L POWER Keyswitch to ON and PANEL Keyswitch to UNLOCK.
2. Press and release INIT Switch.
3. Insert leading edge of paper tape containing CPUXDI into TTY Paper Tape Reader.

4. Press and release LOAD PROG Switch.

5. Set TTY Paper Tape Reader to START.

NOTE

The loader loads the first diagnostic RLM. Repeat step number 4 four times to load remainder of diagnostic.

6. Observe that fourth tape is loaded and HALT lamp is illuminated.

7. Set Display Selector Rotary Switch to PC.

8. Set Data Switches to X'0120.

9. Press and release LOAD DATA Switch.

10. Perform CPUXDI normal operating sequence of paragraph 10.4.4. Start at step 3 of this procedure.

10.4.4 CPUXDI Normal Operating Sequence

The normal operating sequence for CPUXDI is as follows:

1. Load CPUXDI into main memory in accordance with appropriate loading instructions previously given.

2. Program should halt when loading is complete.

3. Set Display Selector Rotary Switch to PC. DATA DISPLAY Indicators should display X'0135.

4. Load Start Test Number into AC0 in accordance with loading instructions previously given.

5. Load End Test Number into AC1 in accordance with loading instructions previously given.

6. Load Looping Mode parameter into AC2 in accordance with loading instructions previously given.

7. Load Continuous Testing Mode parameter into AC3 in accordance with loading instructions previously given.

NOTE

If AC3 is loaded with X'0000, program halts after each complete test. To resume program, press and release RUN Switch.

10. Press and release RUN Switch to begin program execution.

11. When program halts, set Display Selector Rotary Switch to PC. If DATA DISPLAY Indicators display X'0135, all selected tests were executed without detecting any errors. To repeat execution of previously selected tests, press and release RUN Switch. To select different tests, repeat steps 4 through 11. If contents of PC (as displayed by DATA DISPLAY Indicators) is X'016B, a test has detected an error. Proceed to step 12.

12. When an error is detected, AC3 contains the test number. AC0, AC1, and AC2 contain the test results. The contents of AC0, AC1, AC2, and AC3 can be displayed for interpretation on the DATA DISPLAY Indicators by appropriately positioning the Display Selector Rotary Switch. To repeat execution of previously selected tests, press and release RUN Switch. To select different tests, repeat steps 4 through 11.

13. When no errors exist, the program execution does not halt whether a Looping Mode or the Continuous Testing Mode is selected. To interrupt any continuous testing, perform the following:

    a. Press and release HALT Switch.

    b. Press and release INIT Switch.

    c. Press and release RUN Switch. The computer should halt at the initial halt location (PC = X'0135).

    d. Repeat steps 4 through 13.

NOTE

When a halt-on-error occurs and program execution is continued from the error halt (PC = X'016B), the program executes a long loop on the current test that detected the error if a Looping Mode was previously selected.

This concludes the normal operating sequence for CPUXDI.

10.4.5 Memory Diagnostic

The MEMDIL exercises the IMP-16L Memory Storage to verify reliable performance of the memory. MEMDIL is loaded, via Card Reader or Paper Tape Reader into the IMP-16L main memory. After loading MEMDIL, program control parameters can be selected to test memory addresses, words, and/or bits. In addition, Looping Modes that are implemented when an error is detected can be selected. The type of program control effected is determined by the accumulator entered and the parameter value loaded into that accumulator. The available program control parameters , together with loading procedures for each, are as follows:

Parameter:

Functions Performed:

Function	Bit
Address Test	0
Word Test	1
Bit Test	2
Halt On Error	3
Loop On Selected Tests	4
Redefine Pattern	5
Loop On Single Test	6
Loop On Error	7
Reset Range	8
Relocate Program (New Program address is loaded in AC3)	15

NOTE

Whenever program is relocated, all patterns and ranges must be redefined.

Location:

AC0

Loading Instructions:

1. Set Display Selector Rotary Switch to AC0.
2. Select functions to be performed and set associated Data Switches.
3. Press and release LOAD DATA Switch.

Parameter:

Test Start Address

Location:

AC1

Loading Instructions:

1. Set Display Selector Rotary Switch to AC1.
2. Set Data Switches to desired start address (hexadecimal) of test.
3. Press and release LOAD DATA Switch.

Parameter:

Test End Address

Location:

AC2

Loading Instructions:

1. Set Display Selector Rotary Switch to AC2.
2. Set Data Switches to desired end address (hexadecimal) of test.
3. Press and release LOAD DATA Switch.

10.4.6 MEMDIL Loading Via Card Reader

The following procedure outlines the steps required to load MEMDIL via a Card Reader:

1. Set IMP-16L POWER Keyswitch to ON and PANEL Keyswitch to UNLOCK.
2. On Card Reader, press and release POWER Switch so POWER legend is illuminated.

3. Place two CRBOOT program cards and MEMDIL card deck into Card Reader hopper. Make sure that CRBOOT cards are in front of MEMDIL card deck.

4. Press and release INIT Switch.

5. On Card Reader, press and release RESET Switch. Wait until RESET Switch illuminates green.

6. Press and release AUX 1 Switch. Observe that first CRBOOT card is read by Card Reader.

7. Press and release RUN Switch.

8. When Card Reader halts, set IMP-16L Display Selector Rotary Switch to PC; observe that DATA DISPLAY Indicators display X'0124.

9. Perform MEMDIL normal operating sequence of paragraph 10.4.8. Start at step 4 of this procedure.

10.4.7 MEMDIL Loading Via Paper Tape Reader

The following procedure outlines the steps required to load MEMDIL via a Paper Tape Reader:

CAUTION

Since the Paper Tape Reader and TTY Keyboard are simultaneously enabled, care should be exercised not to input data via the keyboard when a paper tape is being read. If both keyboard and reader simultaneously enter data to the IMP-16L, the results are unpredictable.

1. Set IMP-16L POWER Keyswitch to ON and PANEL Keyswitch to UNLOCK.

2. Press and release INIT Switch.

3. Insert leading edge of paper tape containing MEMDIL into TTY Paper Tape Reader.

4. Press and release LOAD PROG Switch.

5. Set TTY Paper Tape Reader to START.

6. Upon completion of load, program halts. To begin execution, press and release RUN Switch.

7. Perform MEMDIL normal operating sequence of paragraph 10.4.8. Start at step 3 of this procedure

10.4.8 MEMDIL Normal Operating Sequence

The normal operating sequence for MEMDIL is as follows:

1. Load MEMDIL into main memory in accordance with appropriate loading instructions previously given.

2. Program should halt when loading is complete.

3. Set Display Selector Rotary Switch to PC. DATA DISPLAY Indicators should indicate X'0124.

4. Select desired program control parameters of paragraph 10.4.5.

5. Load program functions to be performed into AC0 in accordance with loading instructions previously given.

6. Load Test Start Address into AC1 in accordance with instructions previously given.

7. Load Test End Address into AC2 in accordance with instructions previously given.

8. If program relocation is desired, perform the following:

NOTE

When program address relocation is implemented, steps 4 through 7 must be repeated after loading AC3 with new program address.

a. Set Display Selector Rotary Switch to AC3.

b. Set Data Switches to program relocation address (hexadecimal).

c. Press and release LOAD DATA Switch.

d. Press and release RUN Pushbutton twice.

e. Repeat steps 4 through 7.

9. Press and release RUN Switch to begin program execution.

10. When program halts, set Display Selector Rotary Switch to PC. If DATA DISPLAY Indicators display X'0124, all selected tests were executed without detecting any errors. To repeat the selected testing, press and release the RUN Switch. To select new program control parameters, repeat steps 4 through 11. If contents of PC (as displayed by DATA DISPLAY Indicators) is X'0182, an addressing error was detected. Proceed to step 11. If contents of PC is X'020C, a pattern mismatch was detected. Proceed to step 12.

11. When an addressing error is detected, AC1 contains X'0000 (indicating address error); AC2 contains the word read from memory; and AC3 contains the address referenced. The contents of AC1, AC2, and AC3 can be displayed for interpretation on the DATA DISPLAY Indicators by appropriately positioning the Display Selector Rotary Switch.

12. When a pattern mismatch is detected, AC0 contains the bits that failed; AC1 contains the test that failed; AC2 contains the failure address; and AC3 contains the correct pattern. The contents of AC0, AC1, AC2, and AC3 can be displayed for interpretation on the DATA DISPLAY Indicators by appropriately positioning the Display Selector Rotary Switch.

NOTE

When pattern mismatch occurs and RUN Switch is pressed and released, program halts with PC containing X'020E and AC1 containing the bits under test.

10.5 PROGRAM DEBUG CONSIDERATIONS

10.5.1 Introduction to DEBUG

DEBUG is a relocatable object program that supervises the operation of a user's program during checkout. DEBUG provides computer program test facilities as follows:

- Printing selected areas of memory in hexadecimal or ANSI format.
- Modifying the contents of selected areas in memory.
- Modifying computer registers, including the stack.
- Instruction breakpoint halts.
- Snapshots during execution of user's program.
- Initiating execution at any point in program.
- Memory searching.

10.5.2 Use of DEBUG

DEBUG can be used for IMP-16L programs as well as user-generated programs. The IMP-16L control panels are not normally used in association with DEBUG as the TTY provides complete printout facilities for interpretation of programs supervised by DEBUG. If a control panel interrupt is initiated during DEBUG execution, DEBUG continues from the interrupted point when control panel action is terminated. One advantage of DEBUG over the Programmers Control Panel, as applied to the IMP-16L, is that all 16 levels of the stack can be printed out for examination.

The IMP-16L DEBUG program is loaded into top sector memory via the General Loader (GENLDR) program. Some locations in the memory Base Page are used to store instructions and linkages whereby the user can enter DEBUG. Two entry points (DEBUG and DEBUG1) exist for the IMP-16L DEBUG program. When program entry is made via DEBUG1 an Inherent Initialization Routine provides the DEBUG user with a capability for recovering to DEBUG by performing the following steps:

1. Press and release INIT Switch.
2. Press and release RUN Switch.

After the DEBUG program is loaded by GENLDR, instruct the IMP-16L via TTY or Card Reader to printout a symbols table on the TTY. The symbols table provides the Global entry points for DEBUG and DEBUG1. For more detailed information concerning loading procedures and use of DEBUG and GENLDR, refer to the IMP-16 Utilities Reference Manual, National Semiconductor Order Number IMP-16S/025Y.

Appendix A

SUMMARY OF INSTRUCTIONS

Table A-1. IMP-16L Basic Instruction Set (Executed by CROM I)

Instruction	Mnemonic	Execution Cycles	Memory Read Cycles	Memory Write Cycles
Memory Reference Instructions:				
Load	LD	5	2	–
Load Indirect[1]	LD	5	3	–
Store	ST	6	1	1
Store Indirect[1]	ST	8	2	1
Add	ADD	5	2	–
Subtract	SUB	5	2	–
Jump	JMP	3	1	–
Jump Indirect[1]	JMP	5	2	–
Jump to Subroutine	JSR	4	1	–
Jump to Subroutine Indirect[1]	JSR	6	2	–
Increment and Skip if Zero	ISZ	7,8 if SKIP	2	1
Decrement and Skip if Zero	DSZ	8,9 if SKIP	2	1
Skip if AND is Zero	SKAZ	6,7 if SKIP	2	–
Skip if Greater	SKG	Like Signs: 8,9 if SKIP Unlike Signs: 9,10 if SKIP	2	–
Skip if Not Equal	SKNE	6	2	–
And	AND	5	2	–
Or	OR	5	2	–
Register Reference Instructions:				
Push on to Stack Register	PUSH	3	1	–
Pull from Stack	PULL	3	1	–
Add Immediate, Skip if Zero	AISZ	4,5 if SKIP	1	–
Load Immediate	LI	3	1	–
Complement and Add Immediate	CAI	3	1	–
Register Copy	RCPY	6	1	–
Exchange Register and Top of Stack	XCHRS	5	1	–
Exchange Registers	RXCH	8	1	–
Register And	RAND	6	1	–
Register Exclusive Or	RXOR	6	1	–
Register Add	RADD	3	1	–
Shift Left	SHL	4 + 3K	1	–
Shift Right	SHR	4 + 3K	1	–
Rotate Left	ROL	4 + 3K	1	–
Rotate Right	ROR	4 + 3K	1	–

1 – The symbol @ must precede the designation of the memory location whose contents become the effective address by indirection.

Table A-1. IMP-16L Basic Instruction Set (Executed by CROM I) (Continued)

Instruction	Mnemonic	Execution Cycles	Memory Read Cycles	Memory Write Cycles
Input/Output, Flag, and Halt Instructions:				
Set Flag	SFLG	4	1	–
Pulse Flag	PFLG	4	1	–
Push Flags on Stack	PUSHF	4	1	–
Pull Flags from Stack	PULLF	5	1	–
Register In	RIN	7	1	–
Register Out	ROUT	7	1	–
Halt	HALT	–	–	–
Transfer of Control Instructions:				
Branch-on Condition	BOC	4,5 if branch	1	–
Return from Subroutine	RTS	4	1	–
Return from Interrupt	RTI	5	1	–
Jump to Subroutine Implied	JSRI	4	1	–

Typical Execution Time = $(E + 0.5R_1 + R_2)T$

where

E = number of execution cycles
R_1 = number of read cycles from RAM
R_2 = number of read cycles from ROM
T = time for one microcycle (1.40 μsec)

The 0.5 factor for the read cycle from RAM is included because the main clocks are stopped for one-half microcycle during read operations from RAM. ROM accesses typically cause the clocks to be held for 1.0 ± 0.25 microcycles. For the shift and rotate instructions, K refers to the number of positions shifted or rotated. Note that these time calculations do not take into consideration time increases due to bus access time delays for memory refresh or other DMA transfers.

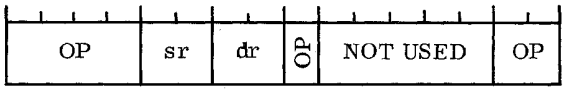
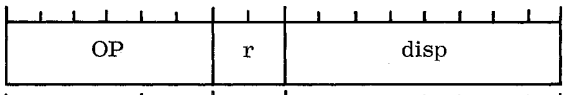
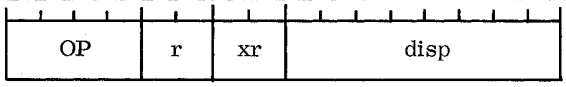
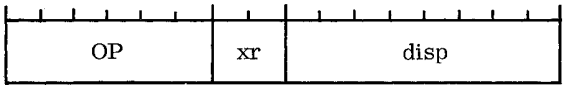
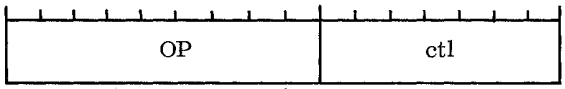
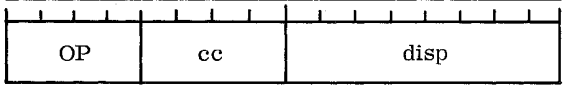
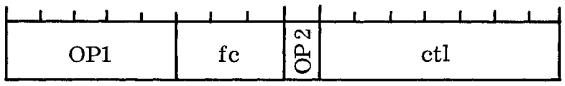
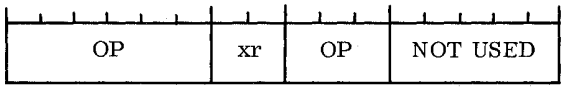
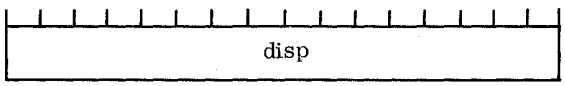
Table A-2. IMP-16L Extended Instruction Set (Executed by CROM II)

Instruction	Mnemonic	Execution Cycles	Memory Read Cycles	Memory Write Cycles
Multiply	MPY	106 to 122	3	–
Divide	DIV	125 to 159	3	–
Double Precision Add	DADD	12	4	–
Double Precision Subtract	DSUB	12	4	–
Load Byte	LDB	20 (left) 12 (right)	4	–
Store Byte	STB	24 (left) 17 (right)	4	1
Set Status Flag	SETST	17 to 36	1	–
Clear Status Flag	CLRST	17 to 36	1	–
Skip If Status Flag True	SKSTF	19 to 39	1	–
Set Bit	SETBIT	15 to 34	1	–
Clear Bit	CLRBIT	15 to 34	1	–
Complement Bit	CMPBIT	15 to 34	1	–
Skip If Bit True	SKBIT	19 to 39	1	–
Interrupt Scan	ISCAN	9 to 80	1	–
Jump Indirect to Level Zero Interrupt	JINT	7	2	–
Jump Through Pointer	JMPP	7	3	–
Jump to Subroutine Through Pointer	JSRP	8	3	–

Appendix B

FORMAT OF INSTRUCTIONS

A summary of the instruction types and their assembler language formats is given below for reference. A more-detailed breakdown of the instruction codes is shown in the next table; it is suitable for hand-coding small programs.

Instruction Type	Machine Format	Assembler Language Format		Remarks
Register to Register		Op	sr, dr	
Register to Memory		Op	r, disp	
Memory Reference (Class 1)		Op Op	r, disp (xr) r, @disp (xr)	Direct Indirect
Memory Reference (Class 2)		Op Op	disp (xr) @disp (xr)	Direct Indirect
I/O and Miscellaneous		Op	ctl	
Branch		Op	cc, disp	
Control Flags		Op	fc, ctl	
Memory Reference (Double Word)		Op	(xr)	
			disp	

Explanation of Symbols

Op — Instruction Mnemonic

OP — Operation Code

sr — Source Register Value

dr — Destination Register Value

xr — Index Register Value (2 or 3)

disp — Displacement Value

cc — Condition Code Value

r — Register Value

ctl — Control Bits Value

Table B-1. Basic Instruction Set with Bit Patterns

Mnemonic	Base
LD	8000
LD Indirect	9000
ST	A000
ST Indirect	B000
ADD	C000
SUB	D000
SKG	E000
SKNE	F000

Word Format

I = BASE ∨ r ∨ xr ∨ disp

r	REGISTER
0000	0
0400	1
0800	2
0C00	3

xr	ADDRESSING TECHNIQUE
0000	BASE PAGE
0100	PC RELATIVE
0200	INDEXED — AC2
0300	INDEXED — AC3

AND	6000
OR	6800
SKAZ	7000

r	REGISTER
0000	0
0400	1

ISZ	7800	JMP Indirect 2400
DSZ	7C00	JSR 2800
JMP	2000	JSR Indirect 2C00

Word Format

BOC 1000

I = BASE ∨ cc ∨ disp

Branch on	INTRPT=1	AC0=0	AC0 ≥ 0	AC0 ODD	AC0 Bit 1=1	AC0 = 0	CPINT =1	START =1
CC	0000	0100	0200	0300	0400	0500	0600	0700

Branch on	STFL=1	INTEN=1	CYOV=1	AC0≤0	POA	SEL	USER	USER
CC	0800	0900	0A00	0B00	0C00	0D00	0E00	0F00

Word Format

I = BASE ∨ r ∨ disp

AISZ	4800
LI	4C00
CAI	5000
PUSH	4000
PULL	4400
XCHRS	5400
ROR/ROL	5800
SHR/SHL	5C00

LEFT DISP POSITIVE
RIGHT DISP NEGATIVE

r	REGISTER
0000	0
0100	1
0200	2
0300	3

Word Format

I = BASE ∨ sr ∨ dr

RADD	3000
RXCH	3080
RCPY	3081
RXOR	3082
RAND	3083

sr	dr	REGISTER
0000	0000	0
0400	0100	1
0800	0200	2
0C00	0300	3

Table B-1. Basic Instruction Set with Bit Patterns (Continued)

Mnemonic	Base	fc	FLAG	Word Format
SFLG	0800			I = BASE ∨ fc ∨ ctl
PFLG	0880	0000	8	
		0100	9	
		0200	10	
		0300	11	
		0400	12	
		0500	13	
		0600	14	
		0700	15	

						Word Format
						I = BASE ∨ ctl
HALT	0000	RTI	0100	RIN	0400	
PUSHF	0080	RTS	0200	ROUT	0600	
PULLF	0280	JSRI	0380	(Address range = FF80 to FFFF)		

The instruction is formed by the inclusive OR of each field. For example, the instruction RADD 2,3 is coded as X'3B00

For instructions that use the CTL field, only the first 7 bits (bits 0 through 6) are considered.

Examples of coding follow:

Example 1	Comments
RADD 2,3	Add AC2 to AC3.

BASE = 3000
sr = 0800
dr = 0300
INSTRUCTION = 3B00

Example 2	Comments
JMP-1 (3)	Jump to the location specified by the index register AC3 modified by the displacement -1.

BASE = 2000
xr = 0300
disp = 00FF
INSTRUCTION = 23FF

Example 3	Comments
SHR 0,1	Shift the contents of AC0 one place to the right.

BASE = 5C00
r = 0000
disp = 00FF
INSTRUCTION = 5CFF

Table B-2. Extended Instruction Set with Bit Patterns

Mnemonic	Base	Word Format
MPY	0480	I (Word 1) = BASE ∨ xr
DIV	0490	I (Word 2) = disp
DADD	04A0	
DSUB	04B0	

xr	ADDRESSING TECHNIQUE
0000	Direct
0100	PC Relative
0200	Indexed - AC2
0300	Indexed - AC3

Mnemonic	Base	Word Format
LDB	04C0	I (Word 1) = BASE ∨ xr
STB	04D0	I (Word 2) = 2. disp ∨ Byte
		Byte = 0 for right, 1 for left.
ISCAN	0510	

Mnemonic	Base	Word Format
SETST	0700	I = BASE ∨ Bit
CLRST	0710	
SETBIT	0720	
CLRBIT	0730	
SKSTF	0740	
SKBIT	0750	
CMPBIT	0760	

Mnemonic	Base	BASADR	MAXADR	Word Format
JSRP	0300	(0100)	007F	I = BASE ∨ Incr
JMPP	0500	(0100)	000F	Incr = ADDR - BASADR
JINT	0520	0120	000F	BASADR ≤ ADDR ≤ BASADR + MAXADR

Example 1	Comments
LDB X'A0A	Load AC0 with the right byte of location X'0A0A; direct addressing.

BASE = 04C0
xr = 0000
INSTRUCTION = 04C0 (Word 1)

disp = X'0A0A
Byte = 0 (Right)

INSTRUCTION = 1413 (Word 2)

Table B-2. Extended Instruction Set with Bit Patterns (Continued)

Example 2	Comments
JMPP 2	Jump through pointer located at location X'0102.
BASE = 0500 BASADR = (0100) Implied INCR = 0002	
INSTRUCTION = 0502	

National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
(408) 732-5000
TWX: 910-339-9240

National Semiconductor Electronics SDNBHD
Batu Berendam
Free Trade Zone
Malacca, Malaysia
Telephone: 5171
Telex: NSELECT 519 MALACCA (c/o Kuala Lumpur)

National Semiconductor GmbH
D 808 Fuerstenfeldbruck
Industriestrasse 10
West Germany
Telephone: (08141) 1371
Telex: 27649

National Semiconductor (UK) Ltd.
Larkfield Industrial Estates
Greenock, Scotland
Telephone: (0475) 33251
Telex: 778 632

NS Electronics (PTE) Ltd.
No. 1100 Lower Delta Rd.
Singapore 3
Telephone: 630011
Telex: 21402

REGIONAL AND DISTRICT SALES OFFICES

ALABAMA
DIXIE DISTRICT OFFICE
3322 Memorial Parkway, S.W. #67
Huntsville, Alabama 35802
(205) 881-0622
TWX: 810-726-2207

ARIZONA
*ROCKY MOUNTAIN REGIONAL OFFICE
7349 Sixth Avenue
Scottsdale, Arizona 85251
(602) 945-8473
TWX: 910-950-1195

CALIFORNIA
*NORTH-WEST REGIONAL OFFICE
2680 Bayshore Frontage Road, Suite 112
Mountain View, California 94043
(415) 961-4740
TWX: 910-379-6432

NATIONAL SEMICONDUCTOR
*DISTRICT SALES OFFICE
Valley Freeway Center Building
15300 Ventura Boulevard, Suite 305
Sherman Oaks, California 91403
(213) 783-8272
TWX: 910-495-1773

NATIONAL SEMICONDUCTOR
SOUTH-WEST REGIONAL OFFICE
17452 Irvine Boulevard, Suite M
Tustin, California 92680
(714) 832-8113
TWX: 910-595-1523

CONNECTICUT
AREA OFFICE
Commerce Park
Danbury, Connecticut 06810
(203) 744-2350

*DISTRICT SALES OFFICE
25 Sylvan Road South
Westport, Connecticut 06880
(203) 226-6833

FLORIDA
*AREA SALES OFFICE
2721 South Bayshore Drive, Suite 121
Miami, Florida 33133
(305) 446-8309
TWX: 810-848-9725

CARIBBEAN REGIONAL SALES OFFICE
P.O. Box 6335
Clearwater, Florida 33518
(813) 441-3504

ILLINOIS
NATIONAL SEMICONDUCTOR
WEST-CENTRAL REGIONAL OFFICE
800 E. Northwest Highway, Suite 203
Mt. Prospect, Illinois 60056
(312) 394-8040
TWX: 910-689-3346

INDIANA
NATIONAL SEMICONDUCTOR
NORTH-CENTRAL REGIONAL OFFICE
P.O. Box 40073
Indianapolis, Indiana 46240
(317) 255-5822

KANSAS
DISTRICT SALES OFFICE
13201 West 82nd Street
Lenexa, Kansas 66215
(816) 358-8102

MARYLAND
CAPITAL REGIONAL SALES OFFICE
300 Hospital Drive, No. 232
Glen Burnie, Maryland 21061
(301) 760-5220
TWX: 710-861-0519

MASSACHUSETTS
*NORTH-EAST REGIONAL OFFICE
No. 3 New England, Exec. Office Park
Burlington, Massachusetts 01803
(617) 273-1350
TWX: 710-332-0166

MICHIGAN
*DISTRICT SALES OFFICE
23629 Liberty Street
Farmington, Michigan 48024
(313) 477-0400

MINNESOTA
DISTRICT SALES OFFICE
8053 Bloomington Freeway, Suite 101
Minneapolis, Minnesota 55420
(612) 888-3060
Telex: 290766

NEW JERSEY/NEW YORK CITY
MID-ATLANTIC REGIONAL OFFICE
301 Sylvan Avenue
Englewood Cliffs, New Jersey 07632
(201) 871-4410
TWX: 710-991-9734

NEW YORK (UPSTATE)
CAN-AM REGIONAL SALES OFFICE
104 Pickard Drive
Syracuse, New York 13211
(315) 455-5858

OHIO/PENNSYLVANIA/
W. VIRGINIA/KENTUCKY
EAST-CENTRAL REGIONAL OFFICE
Financial South Building
5335 Far Hills, Suite 214
Dayton, Ohio 45429
(513) 434-0097

TEXAS
*SOUTH-CENTRAL REGIONAL OFFICE
5925 Forest Lane, Suite 205
Dallas, Texas 75230
(214) 233-6801
TWX: 910-860-5091

WASHINGTON
DISTRICT OFFICE
300 120th Avenue N.E.
Building 2, Suite 205
Bellevue, Washington 98005
(206) 454-4600

INTERNATIONAL SALES OFFICES

AUSTRALIA
NS ELECTRONICS PTY, LTD.
Cnr. Stud Road & Mountain Highway
Bayswater, Victoria 3153
Australia
Telephone: 729-6333
Telex: 32096

CANADA
*NATIONAL SEMICONDUCTOR CORP.
1111 Finch Avenue West
Downsview, Ontario, Canada
(416) 635-9880
TWX: 610-492-1334

DENMARK
NATIONAL SEMICONDUCTOR
SCANDINAVIA
Vordingborggade 22
2100 Copenhagen
Denmark
Telephone: (01) 92-OBRO-5610
Telex: DK 6827 MAGNA

ENGLAND
NATIONAL SEMICONDUCTOR (UK) LTD.
The Precinct
Broxbourne, Hertfordshire
England
Telephone: Hoddesdon 69571
Telex: 267-204

FRANCE
NATIONAL SEMICONDUCTOR
FRANCE S.A.R.L.
28, Rue de la Redoute
92260-Fontenay-Aux-Roses
Telephone: 660-81-40
TWX: NSF 25956F

HONG KONG
*NATIONAL SEMICONDUCTOR
HONG KONG LTD.
9 Lai Yip Street
Kwun Tung, Kowloon
Hong Kong
Telephone: 3-458888
Telex: HX3866

JAPAN
*NATIONAL SEMICONDUCTOR JAPAN
Nakazawa Building
1-19 Yotsuya, Shinjuku-Ku
Tokyo, Japan 160
Telephone: 03-359-4571
Telex: J 28592

SWEDEN
NATIONAL SEMICONDUCTOR SWEDEN
Sikvagen 17
13500 Tyreso
Stockholm
Sweden
Telephone: (08) 712-04-80

WEST GERMANY
*NATIONAL SEMICONDUCTOR GMBH
8000 Munchen 81
Cosimstrasse 4
Telephone: (0811) 915-027

***Microprocessor System Specialist Available**

T2M54