CHAPTER 1
SYSTEM INTRODUCTION

GENERAL

This chapter is divided into two sections: Section 1 describes the PDP-8/E basic processor and section 2 describes the PDP-8/M basic processor.

SECTION 1 THE PDP-8/E BASIC SYSTEM

The development of the PDP-8/E is the successful culmination of many years of computer design research—a process that has enabled Digital Equipment Corporation to provide better computers at the lowest possible price.

The PDP-8/E is specially designed as a general purpose computer. It is fast, compact, inexpensive, and easy to interface. The PDP-8/E is designed to meet the needs of the average user and is capable of modular expansion to accomodate most individual requirements for a user's specific applications.

The PDP-8/E basic processor is a single-address, fixed word length, parallel-transfer computer using 12-bit, 2's complement arithmetic. The cycle time of the 4096-word random address magnetic core memory is 1.2 microseconds for fetch and defer cycles without autoindex; and 1.4 microseconds for all other cycles. Standard features include indiret addressing and facilities for instruction skip and program interrupt as a function of the input/output device condition.

Five 12-bit registers are used to control computer operations, address memory, operate on data and store data. A Programmer's console provides switches to allow addressing and loading memory and indicators to observe the results. The PDP-8/E may also be programmed using the console Teletype with a reader/punch facility. Thus, programs can be loaded into memory using the switches on the Programmer's console, the Teletype keyboard, or the paper tape reader. Processor operation includes addressing memory, storing data, retrieving data, receiving and transmitting data and mathematical computations.

The 1.2/1.4 microsecond cycle time of the machine provides a computation rate of 385,000 additions per second. Each addition requires 2.6 microseconds (with one number in the accumulator) and subtraction requires 5.0 microseconds (with the subtrahend in the accumulator). Multiplication is performed in 256.5 microseconds or less by a subroutine that operates on two signed 12-bit numbers to produce a 24-bit product, leaving the 12 most significant bits in the accumulator. Division of two signed 12-bit numbers is performed in 342.4 microseconds or less by a subroutine that produces a 12-bit quotient in the accumulator and a 12-bit remainder in core memory. Similar signed multiplication and division operations are performed in approximately 40 microseconds, utilizing the optional Extended Arithmetic Element.

The flexible, high-capacity input/ouput capabilities of the computer allow it to operate a large variety of peripheral machines. Besides the standard keyboard and paper-tape punch and reader equipment, these

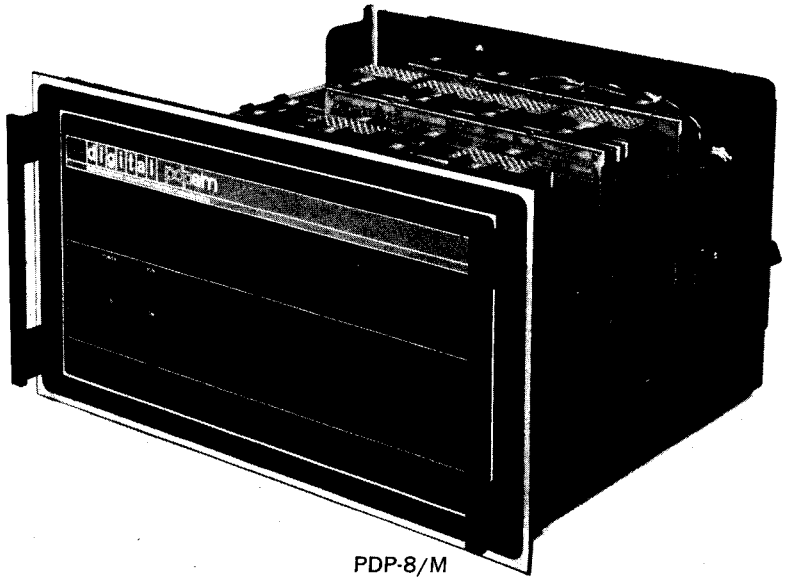
computers are capable of operating in conjunction with a number of optional devices (such as high-speed perforated-tape punch and reader equipment, card reader equipment, line printers, analog-to-digital converters, cathode ray tube (CRT) displays, magnetic tape equipment, a 32,764-word random-access disk file, a 262,112-word random-access disk file, etc.).

The PDP-8/E system is completely self-contained, and requires no special power sources or environmental conditions. A single source of 115V or 230V at 47 to 63 Hz, single-phase power is required. Internal power supplies produce the necessary operating voltages for the system.

The user has a choice of two basic configurations. Table Top or Rack Mountable. Standard DEC cabinets are available to accommodate those users desiring many peripherals. The Table Top version is a convenient approach for those desiring to use the processor in a small area, such as an office.

SECTION 2 THE PDP-8/M OEM PROCESSOR

DEC has recently introduced the OEM version of the PDP-8/E called the PDP-8/M. Because the OEM version is functionally the same processor as the PDP-8/E, all chapters contained in this handbook apply to the PDP-8/M as well as the PDP-8/E with the following exceptions:



PDP-8/M

General Description

The PDP-8/M is a 12-bit parallel computer with identical performance as the PDP-8/E. The PDP-8/M contains one OMNIBUS which defines the maximum basic system configuration. Expansion of the system to three (3) OMNIBUSes is possible.

The Basic PDP-8/M consists of the following components:

- KK8-E Central Processor—same as used in the PDP-8/E.
- MM8-E 4K Memory—same as the PDP-8/E.
- OMNIBUS—The PDP-8/M contains only one OMNIBUS (20 slots).
- The new H740 power supply is sufficient to drive one fully expanded OMNIBUS:

	+5V	−15V	+15V	# of Slots
BASIC 8/M-MC Options Permitted	6.6A	3.3A	0.6A	8
	10.4A	1.7A	0.4A	12
Total Available	17.0A	5.0A	1.0A	20

- The power switch can turn on the PDP-8/M directly, or can operate a remote power control, or both.
- Front Panel—The PDP-8/M offers two front panels corresponding to the two models offered.

The PDP-8/M-MC includes the KC8-M Operators Panel and 4K memory. This panel contains a Power On switch, a Power On indicator, a SW switch (for MI8-E Bootstrap Loader) and a RUN indicator. The indicators are solid state light emitting diodes.

The PDP-8/M-DC includes the KC8-ML Programmers Panel and 4K Memory. This panel is similar to, and has all the features (lights and switches) of the KC8-EA panel standard on the PDP-8/E. All lights, however, are solid state LED's. This panel is also offered as an option.

NOTE

There is no way to "initialize" the PDP-8/M with the KC8/M Operators Panel. An optional KL8-ML Programmers Panel, or a MI8-E Bootstrap Loader, or KP8-EA Power Fail and Auto Restart option, or customer defined loader is required.

All PDP-8/E options are applicable.

SECTION 3 PDP-8/E COMPUTER ORGANIZATION

The PDP-8/E system consists of a central processor, core memory, and input/output equipment facilities; all of which interrelate by means of a common bus called "OMNIBUS."

All arithmetic, logic, and system control operations are performed by the central processor. Information storage and retrieval operations are performed by the core memory. The memory is continuously cycling, automatically performing a read and write operation during each computer cycle. Input and output address and data buffering of the core memory are performed by registers in the central processor, and the operation of the core memory is under control of timing signals produced by the central processor. Because of the close relationship of operations performed by the central processor and the core memory, both are described in this chapter.

Central processor interface circuits provide bussed connections to a variety of peripheral input/output equipment. Each input/output device is responsible for detecting its own select code and for providing all required input or output gating. Individually programmed data transfers between the central processor and peripheral equipment take place through the central processor accumulator. Data break transfers can be initiated by peripheral equipment, rather than under program control, through the data break facilities. Standard features of the computer allow peripheral equipment to perform certain control functions, i.e., instruction skipping and a transfer of program control initiated by a program interrupt.

Standard equipment provided with each system includes a console teleprinter control, which drives and controls a Teletype Model 33 Automatic Send-Receive (ASR). The ASR set is a standard machine operating from serial 11-unit code characters at a rate of 10 characters per second. The ASR provides a means of supplying data to the computer from keyboard or perforated tape; and supplies data as output from the computer in the form of typed copy, or typed copy and perforated tape. The Teletype control serves as a serial-to-parallel converter for teletype inputs to the computer and serves as a parallel-to-serial converter for computer output signals to the Teletype unit. The Teletype and other input/output equipment options are discussed in Chapter 7 of this handbook.

The basic system is illustrated in Figure 1-1. It contains ten PDP-8/E FLIP-CHIP modules called: Major register (M8300); Register Control (M8310); Bus-Loads (M8320); Timing Generator (M8330); Panel type KE8-EA; Teletype control (M8560); XY Driver and Current Source (G227); Memory Stack (H220); Sense/Inhibit (G104). (The last three modules are memory system modules.) and RFI Shield (M849).

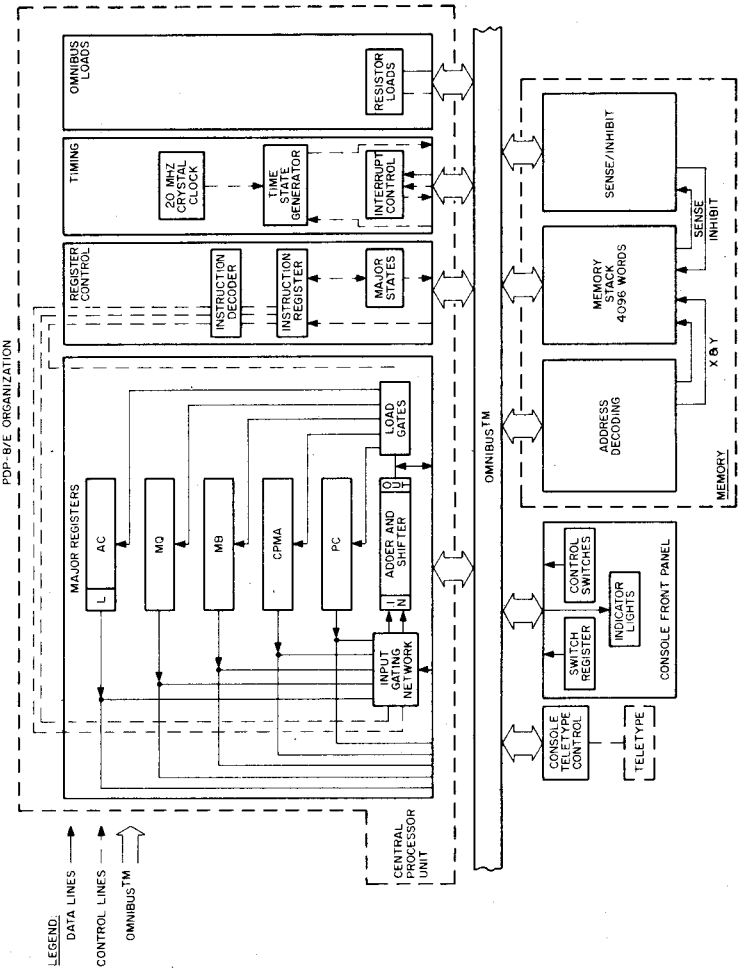


Figure 1-1 PDP-8/E Basic System Block Diagram

MAJOR REGISTERS (M8300)

In order to store, retrieve, control, and modify information and to perform the required logical, arithmetic, and data processing operations, the core memory and the central processor employ the logic complement and major registers shown in Figure 1-1 and described in the following paragraphs.

Accumulator (AC)

The AC is a 12-bit register in which arithmetic and logic operations are performed. Under program control the AC can be cleared or complemented, or its contents can be rotated right or left. The contents of the memory buffer register can be added to the contents of the AC (via the adder circuit), and the result stored in the AC. The contents of both of these registers can be combined by the logical AND operation with the result remaining in the AC. The inclusive OR may be performed between the AC and the switch register (on the programmer's console), and the result left in the AC. The AC also serves as an input/output register; all programmed information transfers between the core memory and an I/O device are passed through the AC to data lines located on the OMNIBUS.

Multiplier Quotient (MQ) Register

The MQ is a 12-bit bidirectional shift register that acts as an extension of the AC during EAE operations. The MQ contains the multiplier at the beginning of a multiplication and the least significant half of the product at the conclusion. The MQ contains the least significant half of the dividend at the start of a division and the quotient at the end. The MQ contains the least significant part of a number during a shift or a normalize operation. The MQ is also available as a temporary storage register adding additional capability and flexibility for the programmer.

Program Counter (PC)

The PC is a 12-bit register that is used to control the program sequence; that is, the order in which instructions are performed is determined by the PC. The PC contains the address of the core memory location from which the next instruction is taken. Information enters the PC from the core memory via the Memory Buffer and from the Memory Address Register. Information in the PC is transferred into the Memory Address register to determine the core memory address from which each instruction is taken. Incrementing the contents of the PC establishes the successive program core memory locations and provides skipping of an Instruction based upon a programmed test of information or conditions.

Central Processor Memory Address (CPMA) Register

The CPMA register is a 12-bit register that contains the address in core memory that is currently selected for reading or writing. Therefore, all 4096 words of core memory can be addressed directly by the CPMA. Data can be transferred into the CPMA from the Memory Buffer, from the Program Counter and from the switch register on the operator's console. Memory Addressing is also accomplished by each data break interface (refer to Chapter 6). This register is never cleared. New information is always jam transferred in and the original content is lost.

Memory Buffer (MB) Register

The MB register is a 12-bit register that is used for all information

transfers between the central processor registers and the core memory. Information can be transferred and temporarily held in the MB from the AC or PC. Also, the MB can simultaneously be loaded and incremented by one before being read back into memory. Information can be loaded into the MB from an I/O device during a data break or from core memory. Information is read from a memory location in 0.6 microseconds and rewritten in the same location in another 0.6 microsecond of a single 1.2-microsecond duration memory cycle. Many machine cycles require modification of memory data. In such cycles, an extra 0.2 microsecond is inserted between read and write.

Data Gates and Adders

The Major Registers module also contains the gating necessary to move data from one register to another. At the heart of the data gating is a 12-bit parallel adder. Information from a register is gated to the adder inputs. The output of the adder is applied to a set of shift gates. The output of the shift gates serves as data input to all of the major registers.

REGISTER CONTROLS (M8310)

The Register Control module contains the Link, the Major Register Control circuits, the Major States register, the Instruction register, and the necessary control circuits for the Major Sates register and the Instruction register.

Link (L)

The Link is a 1-bit register that is used to extend the arithmetic facilities of the AC. It is used as the carry register for 2s complement arithmetic. Overflow into the L from the AC can be checked by the program to greatly simplify and speed up single and multiple-precision arithmetic routine. Under program control, the L may be either cleared, complemented, or rotated as part of the AC.

Major Register Control Circuits

The Major Register Control Circuits enable the adder input and shift gates of the Major Register module. They also gate time pulses to cause loading of the appropriate major register.

Major State Register

The Major State register is the control for the three major states of the PDP-8/E. Conditional inputs, consisting of processor instructions combined with the output of the Major State register, determine which of the three major states (FETCH, DEFER, or EXECUTE) the processor is about to enter. Each of the major state signals, when asserted, is used to enable the corresponding register control circuitry.

Instruction Register (IR)

The IR is a 3-bit register that contains the operation code of the instruction currently being performed by the machine. The three most significant bits of the current instruction are loaded into the IR from the memory during a fetch cycle. The contents of the IR are decoded to produce the eight basic instructions and affect the cycles and states entered during each step of the program.

BUS LOADS (M8320)

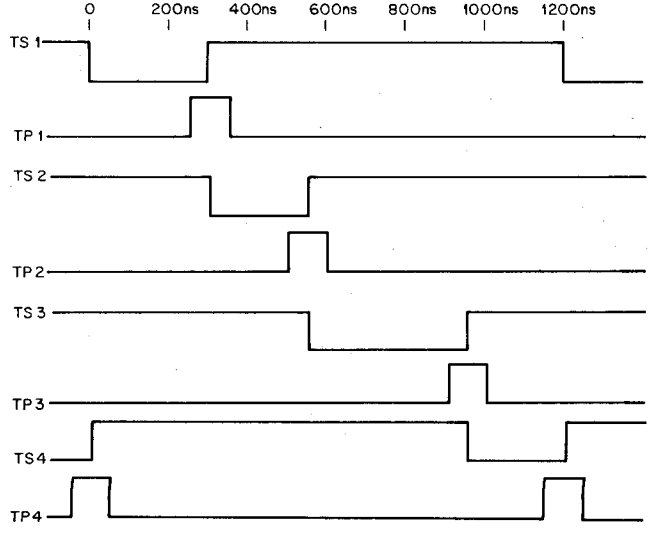
The Bus Loads module contains all of the necessary load resistors required to maintain a high inactive level for each of the busses in the system. There are basically seven groups of signals that the Bus Loads module services. These are: 1. Memory Address (MA); 2. Memory Data (MD); 3. Data; 4. I/O Control; 5. Break Control; 6. Timing; and 7. Miscellaneous Signals. Most lines are considered by the system to be inactive (voltage level high) until the line level is pulled to ground by some component connecting to the corresponding signal line.

TIMING GENERATOR (M8330)

The Timing Generator module contains the time pulse generator, Interrupt Control circuits, the Processor IOT Decoder, and miscellaneous control circuits.

The time pulse generator provides the timing pulses that determine the computer cycle time and are used to initiate sequential time-synchronized gating operations. Pulses that reset registers and control circuits during power turn-on and turn-off operations are produced by the power clear pulse generator. Several of these pulses are available for peripheral device control to be utilized with devices using programmed or data break information transfers.

Four time states, TS1 through TS4, are provided by the time pulse generator. In addition, four time pulses are generated for use as gating pulses throughout the system. Each time pulse overlaps the end of one time state into the beginning of the next time state (refer to Figure 1-2). Memory timing is also provided by the time pulse generator.



MEMORY TIMING CYCLE (FAST CYCLE SHOWN)

The Interrupt Control circuits comprise the major portion of the Interrupt System. The circuitry responds whenever an INTERRUPT REQUEST signal is received from an interface controller module.

The Processor IOT Decoder decodes the last 9 memory data bits and determines the type of IOT instruction that is to be performed.

PROGRAMMER'S CONSOLE (KE8-EA)
The Programmer's Console contains a convenient array of controls and indicators that are used specifically for operation and maintenance. The Console has been configured to achieve convenient control of the system. Through switches and keys on the Console, the operator can STOP, START, EXAMINE, MODIFY, or CONTINUE a program. The indicators, when properly selected, display the machine status and contents of major registers.

A lighted indicator denotes the presence of a Binary 1 in a specific register bit position or control flip-flop.

The Programmer's Console contains a 12-bit switch register, ten control switches, and an indicator selector switch capable of selecting seven individual major states and status registers to be displayed on a 28-lamp indicator panel.

TELETYPE CONTROL (M8350)
The Teletype Control Module contains the necessary receive and transmit circuitry along with the control circuitry to interface the ASR 33 Teletype terminal with the processor.

PDP-8/E MEMORY SYSTEM (MM8-E)
The basic PDP-8/E memory system (MM8-E) is a 4096 word, 12-bit random access core memory that performs all normal functions of data storage and retrieval. The same basic 4K memory, consisting of 3 quad modules, can be used as an extended memory to increase the memory capacity up to the addressing capability (32K) of the PDP-8/E.

Memory location 0 is used to store the contents of the PC following a program interrupt, and location 1 is used to store the first instruction to be executed following a program interrupt. When a program interrupt occurs, the contents of the PC are stored in location 0 and program control is transferred to location 1 automatically. Core memory locations 10 (octal) through 17 (octal) are used for auto-indexing. All other locations can be used for the storage of either instructions or data.

The memory system contains circuits such as read/write switches, address decoders, inhibit drivers, and sense amplifiers. These circuits perform the electrical conversions necessary to transfer information to or from the core array. They perform no arithmetic or logic operations upon the data.

The XY Driver & Current Source (G227)
This PDP-8/E module contains the circuitry required to decode the address lines and drive the XY wires of a 4096 word core memory (i.e., address decoding, selection switches, XY current sources, stack discharge switch, and power on/off write protection). The XY currents are

controlled remotely by a control on the Sense/Inhibit board. The XY Driver and Current Source module requires no adjustments. The same module is used also in the memory parity option.

Memory Stack (H220)
The memory cores are mounted on a G619 Planar Stack Board. The whole module assembly is the H220 Stack. It contains 4096 words of 12-bit core memory and the X-axis and Y axis diode selection matrix. It also includes a resistor/thermistor combination that supplies temperature information to the XY current control. The core memory is a 3D/3-wire memory with center tapped sense/inhibit wire. This module has no connections from and to the OMNIBUS. The same stack is also used in the memory parity option.

Sense/Inhibit Module (G104)
The Sense/Inhibit module (G104) is a PDP-8/E module containing the sense amplifiers, memory register, and the inhibit drivers for a word length of 12 bits. It also includes the slice control and the —6V supply for the sense amplifiers, the current control for the XY current source, control logic for the strobe and clear, and the field select, which is used in the Sense/Inhibit as well as in the XY Driver. Three jumper connections determine the field. Slice level, strobe delay, and XY current can be selected within four discrete steps by appropriate jumper connections (two per axis). With a given stack the proper combination is known and the jumper connection can be selected. Adjustments in a system are, therefore, eliminated.

The memory parity option, consisting of another 3 modules, adds all the circuitry necessary to read, write, and store the parity for 32K of memory. Additional memory options are a 256-word Read Only Memory, a 1024-word Read Only Memory, a 256-word Read/Write Memory, and a Bootstrap Loader Read Only Memory. Refer to Chapter 7 for the description of each memory option.

MAJOR PROCESSOR STATES
The PDP-8/E utilizes three processor states to execute programmed instructions. To accomplish this, a major state generator is used to establish one state for each computer timing cycle. The major processor states are: FETCH, DEFER and EXECUTE. FETCH, DEFER and EXECUTE states determine and execute instructions.

Fetch (F) State
The computer enters the FETCH state to obtain a 12-bit instruction word. At the start of FETCH cycle, the contents of the PC are loaded into the CPMA, giving the first memory address and starting the memory cycle.

At the start of the FETCH cycle, the computer obtains the contents of an addressed memory location and places the 12 bits on the Memory Data lines of the OMNIBUS. The contents of these lines are decoded to determine the kind of instruction the processor must next perform. Once the processor decides the kind of instruction it must do, it then begins performing the instruction, entering a DEFER, or an EXECUTE state as required. When the instruction has been completely performed, the computer again enters the FETCH state to obtain the next instruction.

Assuming the computer is a fully automatic, running condition; that is, the computer is continuously functioning to FETCH and/or EXECUTE instructions, the PC register's contents are transferred to the MA register at TP4 time, initiating the FETCH state. When this is accomplished, the MA is simultaneously incremented by one and the result is transferred to the PC register. Sometime during TS2, the memory is strobed. The contents of the memory appear at the output of the sense amplifiers and are transferred to the MD lines. The contents of the first three bits of the memory data are transferred to the IR. This completes the activity during time TS2 of the FETCH state.

If the instruction is a multicycle (2 or 3) instruction, the memory address is computed, but no further action takes place until the Defer or Execute cycle. If, however, the instruction is a single cycle instruction (such as IOT, OPR, or JMP and bit 3 = 0) the instruction is carried out immediately. The Major State register gating causes the computer to enter the appropriate major state at the end of the cycle.

Defer (D) State
At the end of a FETCH cycle, the current instruction is directed to DEFER whenever indirect addressing is required.

The DEFER state is entered if a binary 1 is present in bit 3 of a memory reference instruction. This state causes the central processor to obtain the full 12-bit address of the operand from an address in either the current page or page zero, as specified by bits 4 through 11 of the instruction. This process of address deferring is called indirect addressing, because access to the operand is addressed indirectly, or deferred, to another memory location.

Execute (E) State
The EXECUTE state is entered for all memory reference instructions except JMP. During an AND, 2s complement add, or increment and skip if zero instruction, the contents of the core memory location specified by the address portion of the instruction are read into the MB and the operation specified by the Instruction Register is performed. During a deposit and clear accumulator instruction, the contents of the AC are transferred into the MB and stored in core memory at the address specified in the instruction. During a jump to a subroutine instruction, the EXECUTE state occurs to write the contents of the PC into the core memory location as designated by the instruction and to transfer this address +1 into the PC to bring about a change in program control.

In addition to the three major states described, the processor responds to other functions such as Data Break. During this time period, FETCH, DEFER and EXECUTE states are held inactive.

Direct Memory Access (DMA) State
A fourth state exists when any one of the other three major states is not enabled. This state, called DMA, is used to independently address memory and to store or read out information without the aid of processor instructions. DMA is used at the Programmer's Console when information is added to or taken from memory. Data Break devices also use the DMA state to perform block transfers.

INTERFACING

The PDP-8/E offers two approaches to interfacing with peripheral equipment. The OMNIBUS is an internal Input/Output Bus on which all I/O data and control signals are transferred. A variety of peripherals can interconnect through a peripheral control module from this bus. The OMNIBUS has eliminated wires by providing an etched circuit board on which connectors are mounted. For the convenience of the user, each pin assignment on one connector is identical to the next connector, allowing a module to be placed anywhere on the bus. The PDP-8/E options provide a complete line of peripherals used in most computer operations. However, for those requirements that are not covered by options, Chapter 9 provides the information needed so that a user can build his own interface. Special interface modules can also be constructed by DEC.

An External Bus is the second approach to interfacing to peripherals. The External Bus connects to the OMNIBUS and provides an extension to the bus system for users who already possess or desire to use PDP-8/I or PDP-8/L compatible peripherals. The interfacing details are provided in Chapter 10 of this handbook.

Three types of data transfer systems are available to the user. One system is the straight Input/Output transfer, which is the simplest and most direct type of transfer. The Program Interrupt system is a type of data transfer system useful for installations having more than one peripheral. For installations using extremely fast transfer rates, the data break system (sometimes called Direct Memory Access, or DMA) is available (refer to Chapter 6 for details on the Data Break system and Chapter 5 for details on programmed transfers).

DIFFERENCES BETWEEN PDP-8/E AND ITS PREDECESSORS

As a new computer is developed, differences between it and its predecessors inevitably result. In many cases these differences are either benign or beneficial. All differences are listed below in order to give users of earlier machines a concise summary.

Instruction Differences

NEW INSTRUCTION	OCTAL	PREVIOUS FUNCTIONS
Byte Swap (BSW)	7002	Rotate 2, no direction (no operation)
Swap MQ and AC (SWP)	7521	MQL MQA (worked as a SWP in KE8-I, but not documented)
Reserved for future expansion	7014	RAR RAL*
Reserved for future expansion	7016	RTR RTL*
MQ instructions	74X1	Only available with EAE on previous machines; otherwise produced a NOP.
Skip if interrupt on (SKON)	6000	No operation
Skip on interrupt request (SRQ)	6003	(ION)

NEW INSTRUCTION	OCTAL	PREVIOUS FUNCTIONS
Get flags (GTF)	6004	No operation, or ADC in PDP-8 with 189 A/D Converter
Restore flags (RTF)	6005	(ION) (ORed with ADC)
Skip if Greater Than (SGT)	6006	(IOF) (ORed with ADC)
Clear all flags (CAF)	6007	(ION) (ORed with ADC)

* These instructions produced predictable but undocumented results in PDP-8/I and PDP-8/L. They may have been used by some programmers to load the constants 3776 and 5775 into the AC. In the PDP-8/E, these codes are specifically reserved for future expansion, and should not be executed.

TTY Differences

The console TTY uses the same IOT's as in earlier machines, but also has added IOTs to enable or disable TTY flags onto the interrupt bus. Also included are additional IOTs to clear keyboard flags without advancing the reader and an IOT to set the printer flag. The skip IOT's can no longer be microprogrammed with the other IOTs of the same device code. This should impose no constraint on the user, since skips are generally not combined with other IOT's. Reader Run is no longer set by INITIALIZE. Hence any routines using the reader must begin with a KCC instruction.

External I/O Bus

In general, the signals and functions at the External I/O Bus Interface are the same as for previous machines. Users who constructed peripherals using previous editions of the Small Computer Handbook as a guide may expect their peripherals to work on the PDP-8/E. (Please note, however, that PDP-8/E is equipped only with a positive bus; and a DW08A bus converter is necessary to interface to older, negative bus equipment.)

The BAC lines at the External I/O Bus interface are merely the buffered DATA bits of the OMNIBUS. Since the DATA lines are used for bi-directional transfer, any input of data at the External I/O Bus interface will cause an immediate change at the BAC outputs. Simultaneous input and output transfers in the same IOP should be checked. In such situations, the register in the peripheral must be edge-triggered.

Also, at the conclusion of the IOP dialogue, the DATA bus is used for updating the PC, and then for determining break or API priority. Users can no longer rely on the BAC lines being available until the end of the major state. However, the IOP width and separation may be adjusted if desired, to accommodate slow I/O devices. External IOTs are faster in all cases except for IOTs ending in 7.

EAE
The Extended Arithmetic Element has been redesigned, and several powerful features have been added. Previous EAE users may use the EAE without modifying their programs, but it is a wise move to recode the EAE programs in order to make use of the new SAM, DCM, DAD, DST, DPIC and DPSZ instructions.

Data Break
The time required to access the Data Break system has been greatly reduced. Maximum benefit can be obtained on machines without EAE, and with only internal options (or options which are not activated while Data Break is in use). An added feature, ADM, allows the user to add an input word to the contents of a memory location. An internal multiplexing scheme allows the use of several (12 max.) external and/or internal break devices.

Control Panel
The control panel of PDP-8/E differs from panels of its predecessors as follows:

1. Only the MA (and EMA) and the RUN status are permanently displayed. All other registers are selected by a rotary switch.
2. Machine stops occur after TP4. Thus the MA lights indicate the next address to be accessed. Similarly, the Major State indicators show the next major state to be executed.
3. Extended field information is loaded via SR6-11 and the EXTD ADDR LOAD switch.
4. Operation of the ADDR LOAD switch places the CP in the FETCH state.
5. Programs are started by operating and releasing the CLEAR key, then operating and releasing the CONT key.
6. Turning the Power Switch to the PANEL LOCK position extinguishes all indicators except the RUN light.

ADDRESSING NONEXISTING CORE
An attempt to deposit data in a non-existant memory field will not stop the machine. An attempt to read data from a non-existant memory yields a zero operand. A jump to a non-existant memory will, of course, "hang up" the program, since there is then no way to jump back to existant memory.

CHAPTER 2

STANDARD SYSTEM OPERATION

GENERAL

The PDP-8/E Computer allows the operator to manually program the machine using the switch register located on the Programmer's Console or use the more automatic process with the ASR 33 Teletype Console which contains the Tape Reader/Punch combination as well as the standard Teletype keyboard. This chapter defines the operation of communicating with the processor in both the manual and program control modes.

The user should be thoroughly acquainted with the content of chapters 3 and 4 before operating the system.

CONTROLS AND INDICATORS

The controls and indicators on the Programmer's Console provide manual control and indicate the program conditions of the PDP-8/E. Controls on the Programmer's Console provide the operator with the hardware to start, stop, examine, modify, or continue a program. The indicators on the console provide a visual indication of the machine status and current program, the contents of the major registers, and the condition of the control flip-flops. A lighted indicator denotes the presence of a binary 1 in a specific register bit position or control flip-flop. Table 2-1 lists the functions of controls and indicators. The controls are divided into two groups; switches and keys. Keys are momentary, or spring-return, switches. Figure 2-1 illustrates the console. Controls and indicators of the standard Model 33 ASR Teletype unit are shown in Figure 2-2 and their functions are described in Table 2-2.

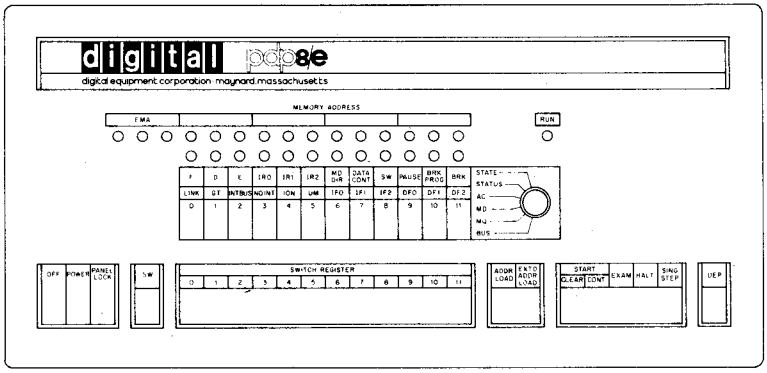


Figure 2-1 Programmer's Console

Table 2-1

Programmer's Console Control and Indicator Functions

CONTROL OR INDICATOR	FUNCTION
Off/Power/Panel Lock	This is a key operated switch. In the counter-clockwise, or OFF, position, the switch disconnects all primary power to the machine. In the POWER, or straight up position, it enables all manual controls and applies primary computer power. In the PANEL LOCK or clockwise position, it disables all keys and switches with the exception of the switch register and mode switch. In this position, a running program is protected from inadvertent switch operation and all panel indicators except the RUN light are turned off.
SW	When this switch is up, the line on the OMNIBUS line called SW is high; when the lever is down, the line is low. This switch is used by special peripheral controls, such as the Bootstrap Loader.
Switch Register Switches (SR)	These 12 switches provide a means of communication between operator and machine. They allow a 12-bit word to be input. When the switch is up it designates a binary 1 to the machine; switch down is a 0 (zero). These switches are used during manual functions or under program control.
Load Address Key (ADDR LOAD)	This key loads the contents of the Switch Register into the CPMA and forces Fetch to be set (no Major States while the Load Address Key is depressed).
Extended Address Load Key (EXTD ADDR LOAD)	This switch loads the contents of SR6-11 into the Data Field and Instruction Field registers of the Memory Extension Control. SR9-11 goes to Data Field 0-2. SR6-8 goes to Instruction Field 0-2.
Clear Key (CLEAR)	This key issues an Initialize Pulse, clearing the AC, LINK, Interrupt system, and I/O Flags.

Table 2-1 (Cont.)

CONTROL OR INDICATOR	FUNCTION
Continue Key (CONT)	This key resumes the computer program by issuing a Memory Start and setting the Run Flip-Flop. The word stored at the address currently held by the CPMA is taken as the first instruction.
Examine Key (EXAM)	Puts the contents of core memory at the address specified by the contents of the CPMA into the MB. Then the contents of the PC and CPMA are incremented by one to allow examination of the contents of sequential core memory addresses by repeating the operation of the examine switch.
Halt Switch (HALT)	This switch clears the Run flip-flop and causes the machine to stop at TS1 of the next Fetch cycle. This switch is also used for single instruction stepping.
Single Step Switch (SING STEP)	This switch clears the Run flip-flop and causes the machine to stop at TS1 of the next cycle. Thereafter, repeated depressing of the continue key steps the program one cycle at a time, so that the contents of registers can be observed in each state.
Deposit Key (DEP)	Loads the contents of the SR into the MB and core memory at the address given by the current contents of the CPMA. Then the contents of the PC and CPMA are incremented by one. This allows storing of information in sequential memory address by repeated operation of the deposit switch.
Indicator Selector Switch	This is a six-position rotary switch, used to select a register for display. The six positions are as follows: 1. STATE — Indicates an individual function for each bit; *Bit* 0—*Fetch* 1—Defer 2—Execute 3—IR 0 4—IR 1 5—IR 2

Table 2-1 (Cont.)

CONTROL OR INDICATOR	FUNCTION
	6—MD DIR 7—Data Control 8—SW 9—Pause 10—Break in Prog 11—Break
	2. STATUS — Indicates an individual function for each bit; Bit 0—Link 1—Greater Than Flag 2—Interrupt Bus 3—No Interrupt Allowed 4—Interrupt On 5—User Mode 6—Instruction Field 0 7—Instruction Field 1 8—Instruction Field 2 9—Data Field 0 10—Data Field 1 11—Data Field 2
Indicator Selector Switch	3. AC — Indicates bits 0-11 of the accumulator at TS1.
	4. MD—Indicates Information just written or rewritten into memory.
	5. MQ—Indicates contents of MQ register during TS1.
	6. BUS—Indicates bits 0-11 of the DATA Lines.
Memory Address	Indicates the contents of the memory address which will be accessed next.
EMA	Indicates which Extended Memory field is being accessed.
Run Light	When lit means machine's timing is enabled and capable of executing instructions.

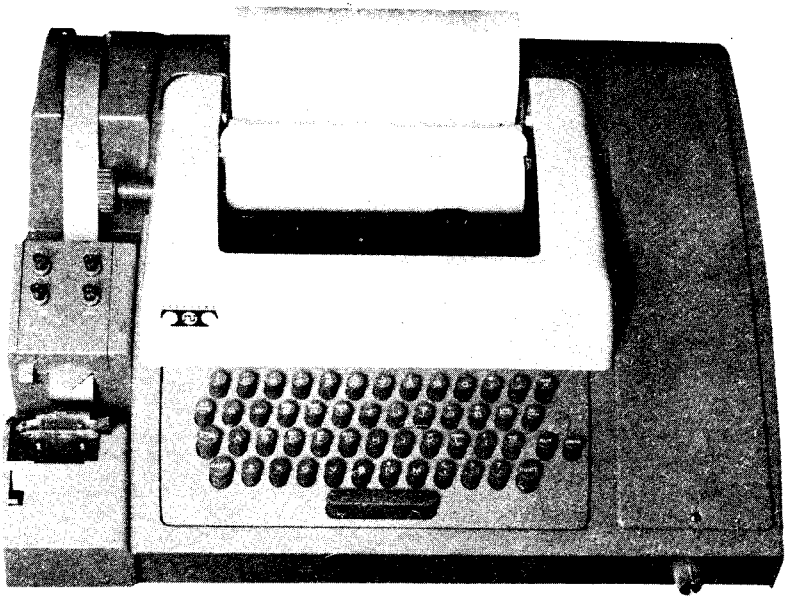


Figure 2-2 Teletype Model ASR33 Console

Table 2-2

ASR33 Teletype Controls and Indicators

CONTROL OR INDICATOR	FUNCTION
REL Pushbutton	Disengages the tape in the punch to allow tape removal or tape loading.
B SP Pushbutton	Backspaces the tape in the punch by one space, allowing manual correction or rubout of the character just punched.
OFF/ON Pushbuttons	Controls use of the tape punch with operation of the Teletype keyboard/printer.
START/STOP/FREE Switch	Controls use of the tape reader with operation of the Teletype. In the FREE position the reader is disengaged, permitting the paper tape to be manually moved within the reader without necessarily reloading or unloading it. In the STOP position the reader mechanism is

Table 2-2 (Cont.)

CONTROL OR INDICATOR	FUNCTION
	engaged but de-energized. In the START position the reader is engaged and operated under program control. The tape may be loaded or unloaded in either the FREE or STOP positions.
Keyboard	Provides a means of printing on paper when used as a typewriter and punching tape when the punch ON pushbutton is pressed; also provides a means of supplying input data to the computer when the LINE/OFF/LOCAL switch is in the LINE position.
LINE/OFF/LOCAL Switch	Controls application of primary power to the Teletype and data connection to the processor. In the LINE position, the Teletype is energized and connected as a computer I/O device. In the OFF position, the Teletype is de-energized. In the LOCAL position, the Teletype is energized for off-line operation, and signal connections to the processor are disconnected. Both LINE and LOCAL use of the Teletype require that the computer be energized through the POWER switch.

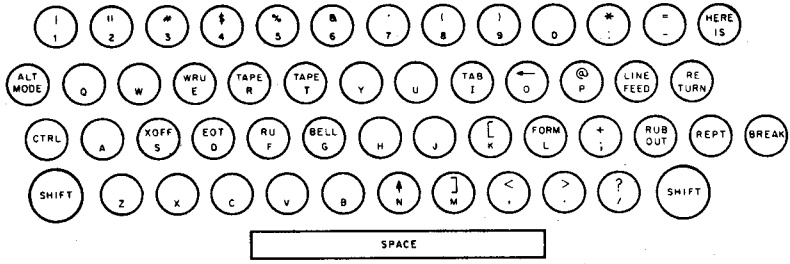


Figure 2-3 Teletype Keyboard

KEYBOARD OPERATION

The Teletype keyboard shown in Figure 2-3 is similar to a typewriter keyboard, except that some nonprinting characters are included as upper case elements. For typing characters or symbols, such as $, %, #, which appear on the upper portion of numeric keys and certain

alphabetic keys, the SHIFT key is held depressed while the desired key is operated.

Designations for certain nonprinting functions are shown on the upper part of some alphabetic keys. By holding the CTRL (control) key depressed and then depressing the desired key, these functions are activated. Table 2-3 lists several commonly used keys that have special functions in the symbolic language of PDP-8/E computers.

Table 2-3

Special Keyboard Functions

KEY	FUNCTION	USE
SPACE	space	used to combine and delimit symbols or numbers in a symbolic program
RETURN	carriage return	used to terminate line of symbolic program
HERE IS	blank tape	used for leader/trailer (effective only in LOCAL)
RUBOUT	rubout	used for deleting characters, punches all channels on paper tape
CTRL/REPT/P	code 200	used for leader/trailer of binary program paper tapes (keys must be released in reverse order: P, REPT, CTRL)
LINE FEED	line feed	follows carriage return to advance printer one line

PRINTER OPERATION

The printer provides a typed copy of input and output at ten characters per second maximum rate. When the Teletype unit is on line (LINE), the copy is generated by the computer; when the Teletype unit is off line (LOCAL), the copy is automatically generated whenever a key is struck.

PAPER TAPE READER OPERATION

The paper tape reader is used to input into memory data punched on eight-channel perforated paper tape at a maximum rate of ten characters per second. The reader control positions are shown in Figure 2-2 and are described below.

START — Activates the reader; reader sprocket wheel is engaged and operative.

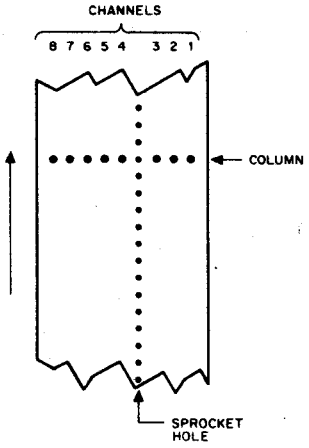
STOP — Deactivates the reader; reader sprocket wheel is engaged but not operative.

FREE — Deactivates the reader; reader sprocket wheel is disengaged.

PAPER TAPE PUNCH OPERATION

The paper tape punch is used to perforate eight-channel rolled oiled paper tape at a maximum rate of ten characters per second. The punch controls are shown in Figure 2-2 and described below.

REL.	Disengages the tape to allow tape removal or loading.
B. SP.	Backspaces the tape one space for each firm depression of the B. SP. button.
ON	Activates the paper tape punch.
OFF	Deactivates the paper tape punch.



Data is recorded (punched) on paper tape by groups of holes arranged in a definite format along the length of the tape. The tape is divided into channels, which run the length of the tape, and into columns, which extend across the width of the tape, as shown in the adjacent diagram. The paper tape readers and punches used with PDP-8 family computers accept eight-channel paper tape.

Generating a Symbolic Tape

The previously described components may be used to generate a symbolic program paper tape through the following procedure.

When switched to LOCAL, the Teletype unit is independent of the computer and functions like an electric typewriter. Any character struck on the keyboard is printed, and also punched on paper tape if the tape punch is ON. Each character struck on the keyboard is represented in code by one row of holes and spaces according to the ASCII code described in the following section and given in Appendix C.

A section of leader-trailer code several inches long is punched at the beginning of the symbolic tape, by pressing the HERE IS key on the

Teletype keyboard. The symbolic program is then carefully typed, following the conventions used in PDP-8 symbolic programs.

A typing error can be corrected using the B. SP. button of the paper tape punch and the RUBOUT key on the Teletype keyboard. The B. SP. button backspaces the paper tape one column for each depression of the button, and the RUBOUT key perforates all eight channels of a column (this perforation is ignored by the computer). Therefore, errors are removed by backspacing the tape to the error and typing rubouts over the error and all following characters. After typing rubouts, the correct information must be typed beginning where the error occurred.

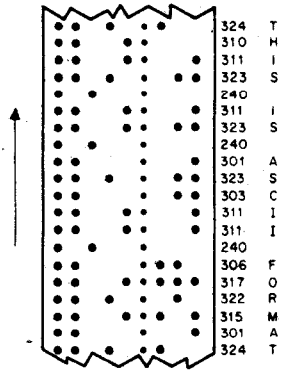
Once the symbolic tape is punched, a form feed is punched, and then more leader-trailer tape is generated by striking the HERE IS key. The tape is removed from the punch unit by tearing against the plastic cover of the punch. The symbolic program thus generated is the input to the assembler described in Chapter 4.

The program may be listed (typed out) by placing the paper tape in the paper tape reader. This is done by releasing the plastic cover of the reader unit and placing the eight-channel tape over the reader head with the smaller sprocket holes over the sprocket wheel, and replacing the cover. If the Teletype control is switched to LOCAL and the reader is switched to START, the tape will advance over the reader head and a printed copy of the program will be typed on the Teletype printer. If the tape punch is also ON, a duplicate of the tape will be generated at the same time.

Paper Tape Formats

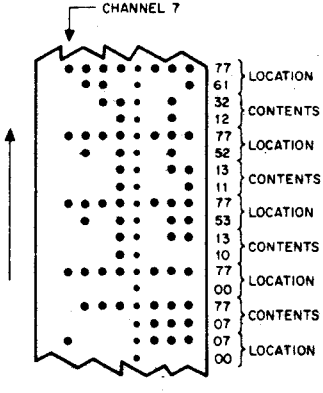
Manual use of the toggle switches on the operator console is a tedious and inefficient means of loading a program. This procedure is necessary in some instances, however, because the PDP-8/E computer must be programmed before any form of input to the memory unit is possible. For example, before any paper tape can be used to input information into the computer, the memory unit must have a stored program which will interpret the paper tape format for the computer. This loader program must be stored in memory with the console switches. A loader program consists of input instructions to accept information from the Teletype paper tape reader and instructions to store the incoming data in the proper memory locations.

Before the loader program can be written to accept information, the format in which the data is represented on the paper tape must be established. There are three basic paper tape formats commonly used in conjunction with PDP-8/E computer. The following paragraphs describe and illustrate these formats.



ASCII FORMAT

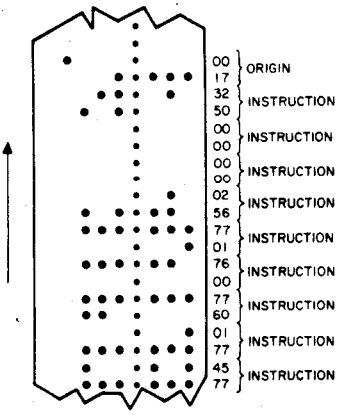
The USA Standard Code for Information Interchange (ASCII) format uses all eight channels* of the paper tape to represent a single character (letter, number, or symbol) as shown in the diagram at left. The complete code is given in Appendix C.



RIM (READ IN MODE) FORMAT

RIM format tape uses adjacent columns to represent 12-bit binary information directly. Channels 1 through 6 are used to represent either address or information to be stored. A channel 7 punch indicates that the adjacent column and the following column are to be interpreted as an address specifying the location in which the information of the following two columns is to be stored. The tape leader and trailer for RIM format tape must be punched in channel 8 only (octal 200).

* Channel 8 is normally designated for parity check. The Teletype units used with the PDP-8/E computer do not generate parity, and Channel 8 is always punched.



BIN (BINARY FORMAT)
BIN format tape is similar to RIM format except that only the first address of consecutive locations is specified. An address is designated by a channel 7 punch and information following an address is stored in sequential locations after the designated address, until another location is specified as an origin. The tape leader/trailer for BIN format tape must be punched in channel 8 (octal 200) only.

Paper Tape Loader Programs

The three previously described paper tape formats are each used for a separate purpose in conjunction with PDP-8/E computer. The ASCII format is used to represent symbolic programs on paper tape, which are then used as input to the assembler. The assembler translates the mnemonic instructions and symbolic addresses into binary instructions and absolute addresses. Once this translation has been performed by the assembler, a binary format tape is generated. ASCII tapes can be quickly recognized by noting that all channels of the tape are used.

The binary format tape is the common means of loading an assembled program into the core memory of a PDP-8/E computer. The BIN (Binary) loader is the program used to load these binary format paper tapes. Program instructions are stored in successive locations beginning with an origin which is signaled by a channel 7 punch on the paper tape. The BIN loader is a lengthy program requiring 83 memory locations. (As an alternative to manually entering the contents of all 83 locations, the RIM (Read In Mode) format is used.) Tapes in binary (BIN) format can be quickly recognized because channel 8 is not used in the middle of the tape, and channel 7 is punched infrequently.

The RIM loader is simpler than the BIN loader because the memory unit is supplied with a location for each incoming instruction. It consists of 17 instructions which must be toggled into memory. The BIN loader is punched in RIM format, and is loaded by the RIM loader; but it is used to load tapes punched in the BIN format, which is the output of the assembly program. Tapes in RIM format are similar in appearance to BIN tapes, but channel 7 is punched every fourth character.

OPERATING PROCEDURES

Many means are available for loading and unloading PDP-8/E information. The means used are dependent upon the form of the information, time limitations, and the peripheral equipment connected to the system. The following procedures are basic to any use of these systems,

and, although they may be used infrequently as the programming and use of the computer become more sophisticated, they are valuable in preparing the initial programs and learning the function of machine input and output transfers.

Manual Data Storage and Modification

Programs and data can be stored or modified manually by means of the facilities on the Programmer's Console. Chief use of manual data storage is made to load the read-in mode (RIM) loader program into the computer core memory. The RIM loader is a program used to automatically load programs into the computer from perforated tape in RIM format. This program and the RIM tape format are described below. Use the following procedure to store data manually in the computer core memory.

Power For Manual Operation

a. Turn the OFF/POWER/PANEL Lock switch clockwise to the POWER position.

Memory Addressing for Manual Operation

a. Set the SWITCH REGISTER switches to correspond to the address bits of the word to be stored.
b. Press the LOAD ADDRESS key.
c. Observe that the address in the switch register is held in the computer as designated by lighted MEMORY ADDRESS indicators corresponding to switches in the 1 (up) position and unlighted indicators corresponding to switches in the 0 (down) position.

Manual Data Input to Addressed Memory Location

a. Set the SWITCH REGISTER switches to correspond to the data or instruction word to be stored at the address just set into the CPMA.
b. Rotate the INDICATOR SELECTOR SWITCH to MD.
c. Lift and release the DEP key.
d. Observe that the data in the SWITCH REGISTER is the same as the data shown on the MD indicators. Data is now stored in the addressed location.
e. Check to see that the MA has been incremented by one so that additional data can be stored at sequential addresses by repeated SWITCH REGISTER settling and deposit key operation.

Checking the Contents of Any Address in Core Memory

a. Perform the Memory Addressing Procedure.
b. Depress the EXAMINE switch.
c. Rotate the INDICATOR SELECTOR switch to the MD position.
d. Observe the data shown on the MD indicators.
e. To observe the next location in core, the contents of the PC and the CPMA are automatically incremented by one. The operator simply depresses the EXAMINE switch and observes the content of the new location.

LOADING DATA UNDER PROGRAM CONTROL

Information can be stored or modified automatically in the computer only by using programs previously stored in core memory. For example,

having the RIM loader stored in core memory allows RIM format tape to be loaded as follows:

INITIALIZING THE SYSTEM

a. Rotate the OFF/POWER/PANEL LOCK switch clockwise to the POWER position.
b. Set the Teletype LINE/OFF/LOCAL switch to the LINE position.
c. Load the tape in the Teletype reader by settling the START/STOP/FREE switch to the FREE position, releasing the cover guard by means of the latch at the right, loading the tape so that the sprocket wheel teeth engage the feed holes in the tape, closing the cover guard, moving the tape either forward or backward until the punched leader section is over the read station, and setting the switch to the STOP position. Tape is loaded in the back of the reader so that it moves toward the front as it is read. Proper positioning of the tape in reader results in three bit positions being sensed to the left of the sprocket wheel and five bit positions being sensed to the right of the sprocket wheel. The directional arrow printed on the tape should point toward the operator.
d. Load the starting address of the RIM loader program (not the address of the program to be loaded) into the PC by means of the Switch Register and the LOAD ADDRESS switches.
e. In sequence, press the computer keys CLEAR and CONTINUE and set the 3-position Teletype reader switch to the START position. The tape is then read automatically.
f. Stop the computer program by means of the Halt switch when the reader reaches the trailer section of tape.

PROGRAM LOADING OPERATION

Automatic storing of the binary loader (BIN) program is performed by means of the RIM loader program as described below. With the BIN loader stored in core memory, program tapes assembled in the program assembly language (PAL III) binary format can be stored as described in the previous procedure, except that the starting address of the BIN loader (usually 7777) is used in step d. When the BIN program is loaded the computer stops; at this point the AC should contain all zeros if the program is stored properly. If the computer stops with a number other than zero in the AC, a checksum error has been detected. When the program has been stored, it can be initiated by loading the program starting address (usually designated on the leader of the tape) into the PC by means of the Switch Register and LOAD ADDRESS switches, then pressing and releasing the CLEAR key and then pressing the CONTinue key.

The steps involved in the process of loading programs and bringing the system up to the point where the user can communicate with the processor is illustrated in Figure 2-4. The loading flow diagram for each type of program to be loaded is referenced, and each flow diagram is accompanied with a corresponding procedure.

This loading procedure is greatly simplified when the user employs a mass storage device such as the Disk Monitor System. Using the Monitor, stored programs are called in from Disk files. This is illustrated in Figure 2-5.

Loaders

When a PDP-8/E computer is first received, it should be assumed that no useful information is in memory. The machine is not capable of performing any arithmetic operations or receiving data.

All tapes in the Program Library are written in binary format. The user, therefore, must load the machine so that it is capable of accepting binary tapes. Initially, sixteen RIM instructions are manually toggled into memory. The binary loader tape is then used to eliminate the necessity of toggling in 86 additional instructions.

RIM allows the binary loader tape to be read into memory and the binary loader tape allows the use of any tape in the Program Library such as symbolic editor.

READ-IN-MODE (RIM) LOADER

The RIM Loader is the very first program loaded into the computer, and it is loaded by the programmer using the console switches. The RIM Loader instructs the computer to receive and store, in core, data punched on paper tape in RIM coded format. (RIM Loader is used to load the BIN Loader described below.)

There are two RIM loader programs: one is used when the input is to be from the low-speed paper tape reader, and the other is used when input is to be from the high-speed paper tape reader. The locations and corresponding instructions for both loaders are listed in Table 2-4.

The procedure for loading (toggling) the RIM Loader into core is illustrated in Figure 2-6.

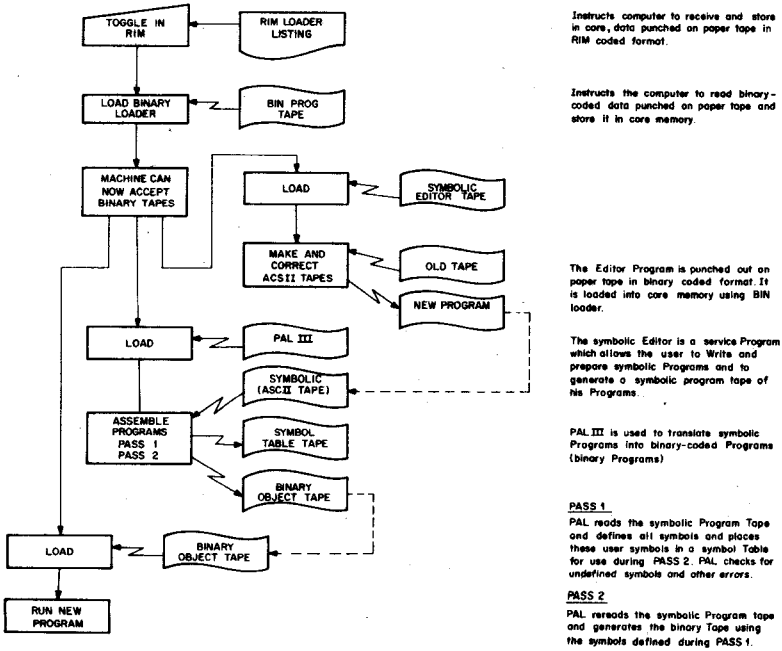


Figure 2-4 Loading Data Under Program Control

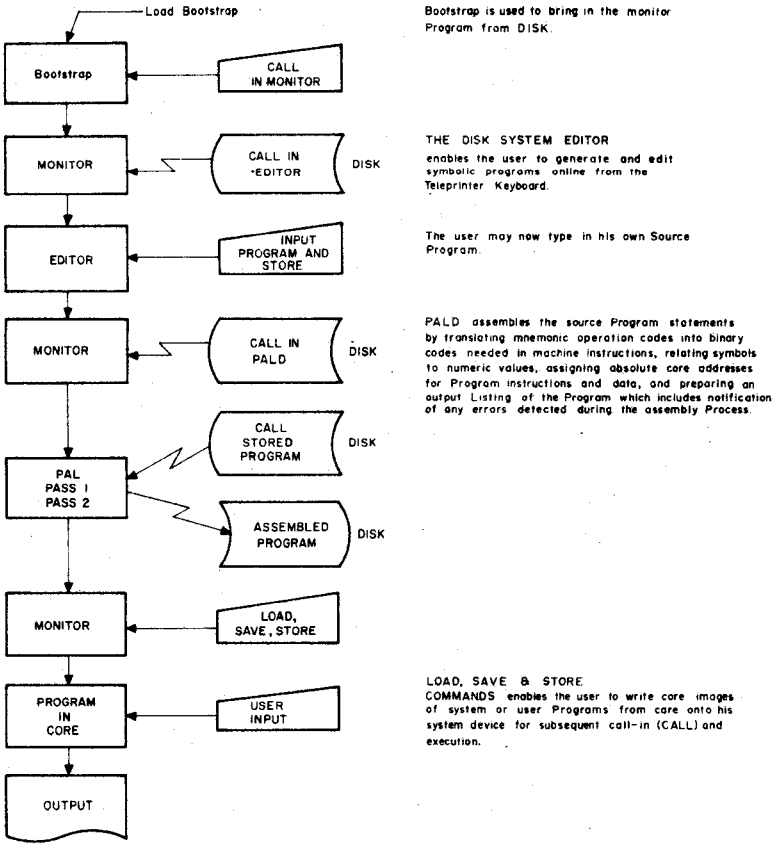


Figure 2-5 Loading Programs with Mass Storage Devices

Table 2-4
RIM Loader Programs

LOCATION	INSTRUCTION	
	Low-Speed Reader	High-Speed Reader
7756	6032	6014
7757	6031	6011
7760	5357	5357
7761	6036	6016
7762	7106	7106
7763	7006	7006
7764	7510	7510
7765	5357	5374
7766	7006	7006

LOCATION	INSTRUCTION	
	Low-Speed Reader	High-Speed Reader
7767	6031	6011
7770	5367	5367
7771	6034	6016
7772	7420	7420
7773	3776	3776
7774	3376	3376
7775	5356	5357
7776	0000	0000

After RIM has been loaded, it is good programming practice to verify that all instructions were stored properly. This can be done by performing the steps illustrated in Figure 2-7, which also shows how to correct an incorrectly stored instruction.
When loaded, the RIM Loader occupies absolute locations 7765 through 7776.

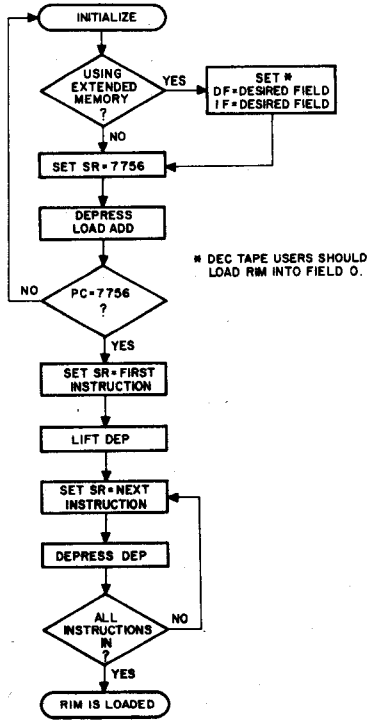


Figure 2-6 Loading the RIM Loader

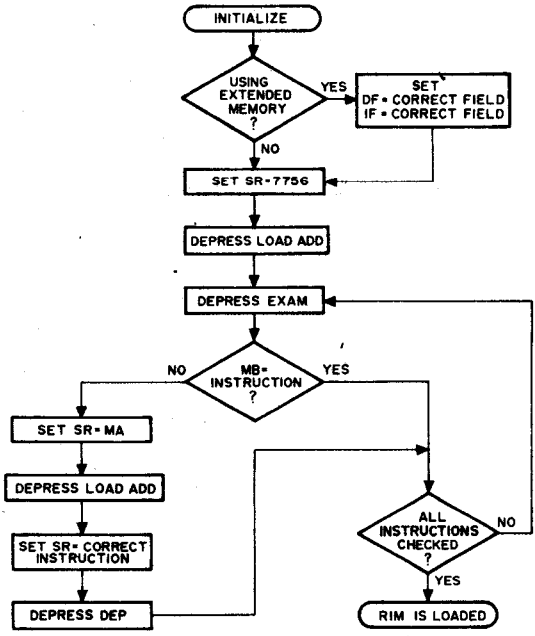


Figure 2-7 Checking the RIM Loader

BINARY (BIN) LOADER

The BIN Loader is a short utility program which, when in core, instructs the computer to read binary-coded data punched on paper tape and store it in core memory. BIN is used primarily to load the programs furnished in the software package (excluding the loaders and certain subroutines) and the programmer's binary tapes.

BIN is furnished to the programmer on punched paper tape in RIM-coded format. Therefore, RIM must be in core before BIN can be loaded. Figure 2-8 illustrates the steps necessary to properly load BIN. When loading, the input device (low- or high-speed reader) must correspond to the version of RIM loaded in the machine.

When stored in core. BIN resides on the last page of core, occupying absolute locations 7625 through 7752 and 7777.

BIN was purposely placed on the last page of core so that it would always be available for use—the programs in DEC's software package do not use the last page of core (excluding the Disk Monitor). The programmer must be aware that if he writes a program which uses the last page of core, BIN will be wiped out when that program runs on the computer. When this happens, the programmer must load RIM and then BIN before he can load another binary tape.

Figure 2-9 illustrates the procedure for loading binary tapes into core.

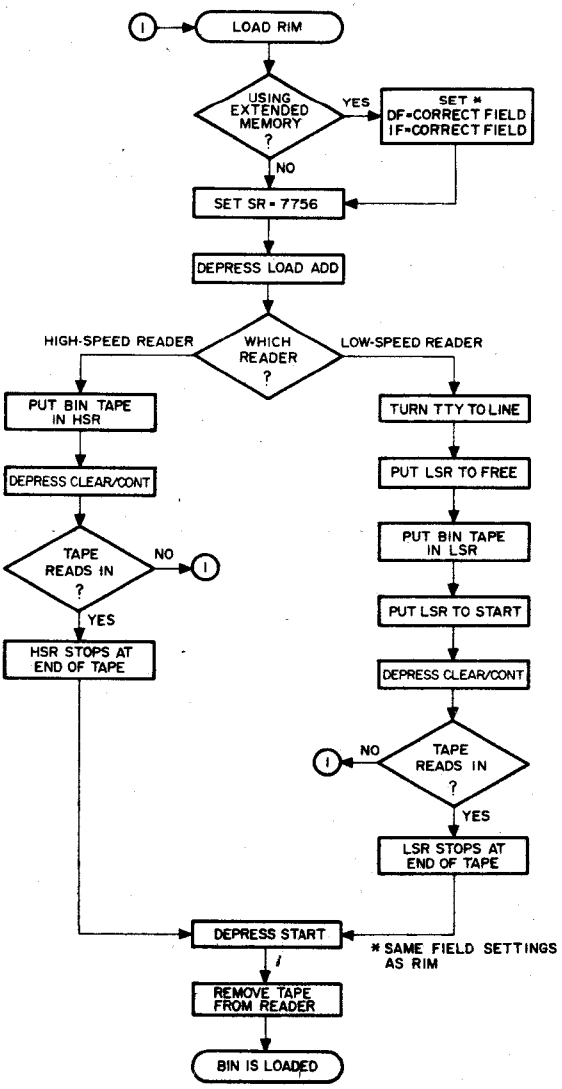


Figure 2-8 Loading the BIN Loader

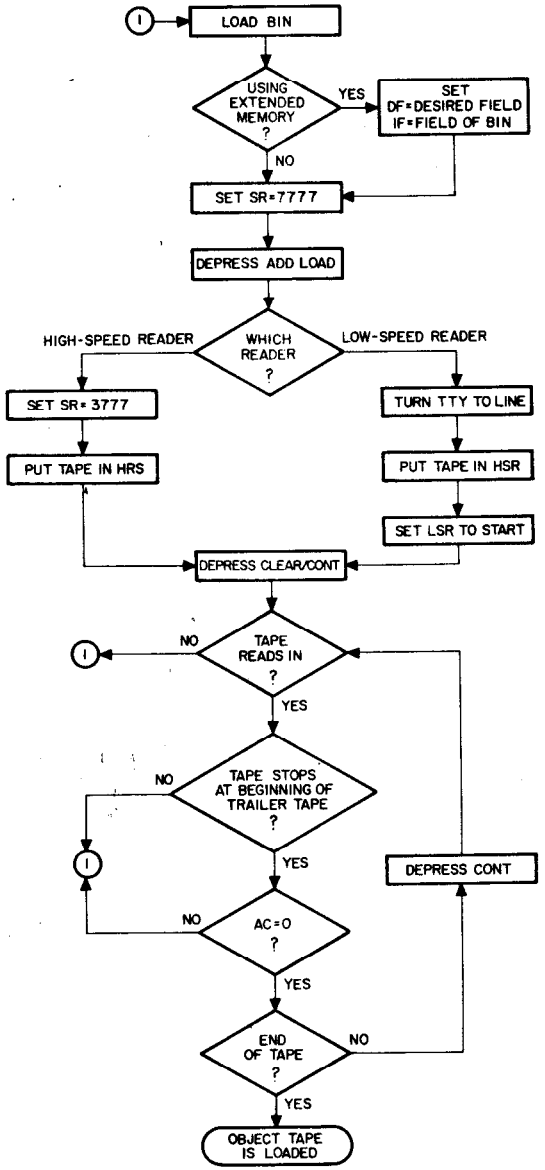


Figure 2-9 Loading A Binary Tape Using BIN

Symbolic Editor

The Symbolic Editor is a service program which allows the programmer to write and prepare symbolic programs and to generate a symbolic program tape of his programs. Editor is very flexible in that the programmer can type his symbolic program online from the Teletype keyboard, thus storing it directly into core memory. Then, using certain Editor commands, the programmer can have his program listed (printed) on the teleprinter for visual inspection.

Editor also allows the programmer to add, correct, or delete any portion of his symbolic program. When the programmer is satisfied that his program is correct and ready to be assembled or compiled, Editor can be commanded to generate a symbolic program tape of the stored program.

The Symbolic Editor program is issued on punched paper tape in binary-coded format. Therefore, it is loaded into core memory using the BIN Loader. When in core, Editor is activated for use by setting the switch register (SR) to 0200 (the starting address) and depressing the LOAD ADD (load address) and then START switches. Editor responds with a carriage return/line feed sequence on the Teletype.

Initially, Editor is in command mode, that is, it is ready to accept commands from the programmer; anything typed by the programmer is interpreted as a command to Editor. Editor accepts only legal commands, and if the programmer types something else, Editor ignores the command and types a question mark (?).

When not in command mode, Editor is in text mode, that is, all characters typed from the keyboard or tapes read in on the tape reader are interpreted as text to be put into the text buffer in the manner specified by a preceding Editor command. Figure 2-10 illustrates how the programmer can transfer Editor from one mode to another.

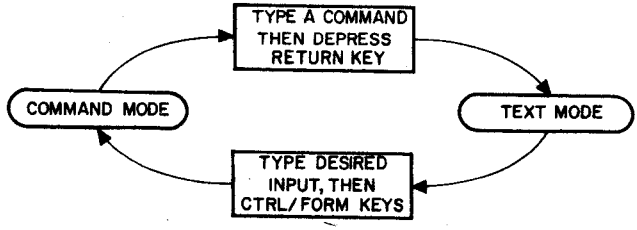


Figure 2-10 Transition Between Editor Modes

Seven of Editor's basic commands are briefly described below.

COMMAND	MEANING
A	Append incoming text from the keyboard into the text buffer immediately following the text currently stored in the buffer.
R	Read incoming text from the tape reader and append it to the text currently stored in the buffer.
L	List entire text buffer; the programmer can specify one line or a group of lines.
C	Change a line; the programmer precedes the command with the decimal line number or line numbers of the lines to be changed.
I	Insert into text buffer; the programmer specifies the decimal line number in his program where the inserted text is to begin.
D	Delete from text buffer; the programmer specifies the line or group of lines to be deleted.
P	Punch text buffer; the programmer can specify one line, a group of lines, or the entire text buffer.

All commands are executed when the RETURN key is depressed except the P command. To execute the P command, press the RETURN key on the Teletype, turn on the punch, and press the CONT (continue) switch on the computer console.

The above commands are only the seven basic commands. A summary of all commands is provided in Table 2-7 at the end of this section.

WRITING A PROGRAM

Now that you have some idea of what you can do with Editor and what Editor can do for you, we will write and edit a short program, explaining each step in the comments to the right of the printout.

The example program finds the larger of two numbers and halts with the number displayed in the accumulator (AC). The program is written in PAL III, to be assembled using the PAL III Assembler described later in this chapter.

The programmer loads Editor using the BIN loader (see Figure 2-8). Editor is then activated by loading the starting address (0200(octal)) and depressing the LOAD ADD, CLEAR and CONT switches. After Editor responds with a carriage return-line feed, the programmer types A and *RETURN key, Editor is now in text mode, that is, subsequent characters* typed are appended to the text buffer. The programmer now types the symbolic program. (Block indenting is facilitated using the CTRL/TAB key, which Editor has programmed to indent in ten-character increments.)

```
A
                *200
                CLA             /CLEAR AC
                TAD NUMB        /GET B
                CMA
                CMA             /1'S COMP B
                IAC             /-B
                TAD NUMA        /ADD -B + A
                SMA             /IF -B LARGER
                JMP .+4         /JUMP 4 LOCATIONS
                CLA             /CLEAR AC
                TAD NUMB        /GET B
                HLT             /B IS LARGER
                CLA             /CLEAR AC
                TAD NUMA        /GET A
                HLT             /A IS LARGER
NUMB,           0000
NUMA,           0000
                $
```

Visual inspection reveals that we have errors in lines 4, 16, and 17. (Editor maintains a line number count in decimal, with the first line typed being 1 and our last line being 18.) Line 4 can be removed using the D (Delete) command, and lines 16 and 17 can be corrected using the C (Change) command. However, Editor is presently in text mode, and in order to issue another command Editor must be transferred to command mode. This is done when the programmer types CTRL/FORM (depress and hold down the CTRL key while typing the FORM key).

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM; Editor responds with CR/LF and rings the teleprinter bell, indicating that it is in command mode.
4D	The programmer types 4D and the RETURN key; Editor responds with a CR/LF and the line is deleted.
15, 16C	The programmer types 15, 16C and the RETURN key, informing Editor that lines 15 and 16 (formerly 16 and 17) are to be changed. Editor responds with a CR/LF, transfers *to text mode, and waits for the programmer to change the lines.*
NUMA, 1111 NUMB, 0011	The programmer types NUMA, 1111 and NUMB, 0011.

The symbolic program should now be correct. However, it is good programming practice to check the program after editing; this can be done using the L (List) command, but since only original lines 4, 16, and 17 were changed it is not necessary to have the whole program listed. The programmer can command Editor to list lines 4 through 17.

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM to return Editor to command mode; Editor responds with CR/LF and rings the bell, and waits for the next command.
4, 17L	The programmer types 4, 17L and the RETURN key; Editor types lines 4 through 17.

```
        CMA         /1'S COMP B
        IAC         /-B
        TAD NUMA    /ADD -B + A
        SMA         /IF -B LARGER
        JMP .+4     /JUMP 4 LOCATIONS
        CLA         /CLEAR AC
        TAD NUMB    /GET B
        HLT         /B IS LARGER
        CLA         /CLEAR AC
        TAD NUMA    /GET A
        HLT         /A IS LARGER
NUMA,   1111
NUMB,   0011
        $
```

The changes were accepted properly. The symbolic program is correct and ready to be punched on paper tape.

GENERATING A PROGRAM TAPE

Before issuing the P (Punch) command, Editor must be in command mode. Figure 2-11 illustrates the procedures required to generate a symbolic program tape using Editor.

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM; Editor responds with a question mark, indicating that Editor was already in command mode.
?	
P	The programmer commands Editor to punch the entire text buffer by typing P and the RETURN key.

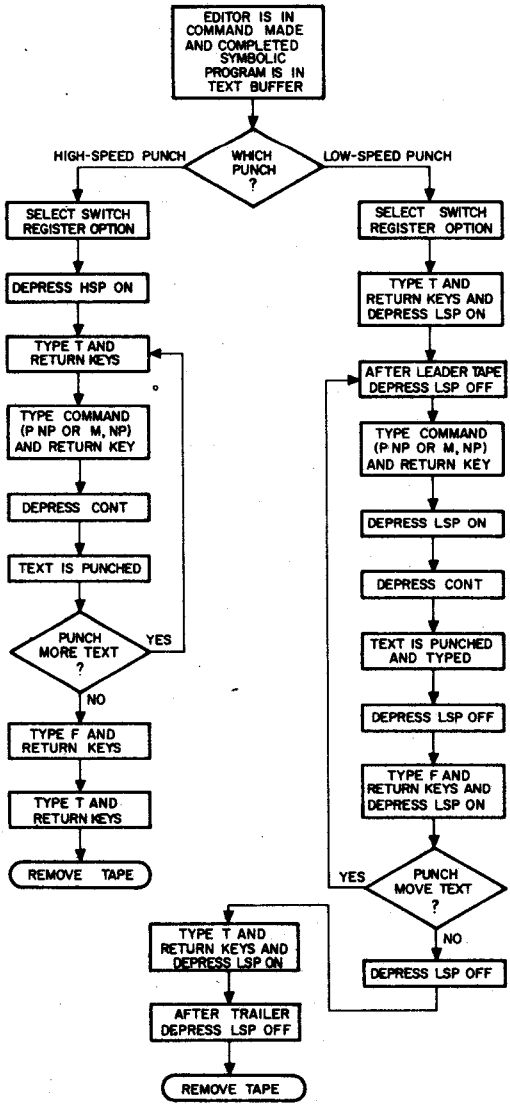


Figure 2-11 Generating a Symbolic Tape Using Editor

When Editor recognizes a P command it waits for the programmer to specify the low- or high-speed punch. If the programmer wants the pro-

gram punched and typed, he sets SR bit 10 to 0 and the program will be punched on the low-speed punch and simultaneously typed on the teleprinter. If the programmer want only a program tape and if he has a high-speed punch available, he sets SR bit 10 to 1 and the program will be punched on the high-speed punch. For the purposes of this discussion, a printed program listing is desired, so the low-speed punch is specified. The programmer turns on the low-speed punch and depresses the CONT switch on the computer console, and Editor begins punching and typing the contents of the entire text buffer.

An image of the stored symbolic program has been punched and typed by Editor

If the programmer stops the computer, e.g., purposely or accidentally turning the computer off, he may restart Editor at location 0200 or 0177 without disturbing the text in the buffer. Editor can also be restarted at location 0176; however, all text currently in the buffer is wiped out. Therefore, the programmer can restart at location 0176 to re-initialize for a new program.

SEARCH FEATURE

A very convenient feature available with Editor is the search feature which allows the programmer to search a line of text for a specified character. When the programmer types a line number followed by S, Editor waits for the user to type in the character for which it is to search. The search character is not echoed (printed on the teleprinter). When Editor locates and types the search character typing stops, and Editor waits for the programmer to either type new text and terminate the line with a RETURN key or to use one of the following special keys.

1.	←	to delete the entire line to the left,
2.	RETURN	to delete the entire line to the right,
3.	RUBOUT	to delete from right to left one character for each RUBOUT typed (a / is echoed for each RUBOUT typed).
4.	LINE FEED	to insert a carriage return/line feed (CR/LF) thus dividing the line into two,
5.	CTRL/FORM	to search for the next occurrence of the search character, and/or
6.	CTRL/BELL	to change the search character to the next character typed by the programmer.

INPUT/OUTPUT CONTROL

Switch register options are used with input and output commands to control the reading and punching of paper tape. The options available to the programmer are shown in Table 2-5. These options are used in conjunction with the "Select Switch Register Option" operation in Figure 2-11.

Table 2-5

Input/Output Control

SR BIT	POSITION	FUNCTION
0	0	Input text as is
	1	Convert all occurrences of 2 or more spaces to a tab
1	0	Output each tab as 8 spaces
	1	Tab is punched as tab/rubout
2	0	Output as specified
	1	Suppress output*
10	0	Low-speed punch and Teleprinter
	1	High-speed punch
11	0	Low-speed reader
	1	High-speed reader

* Bit 2 allows the user to interrupt any output command and return immediately to command mode; when desired, merely set bit 2 to 1.

ERROR DETECTION

Editor checks all commands for nonexistent information and incorrect formatting. When an error is detected, Editor types a question mark (?), and ignores the command. However, if an argument is provided for a command that doesn't require one, the argument is ignored and the command is executed properly.

Editor does not recognize extraneous and illegal control characters; therefore, a tape containing these characters can be cleared up or corrected by merely reading the tape into Editor and punching out a new tape.

SUMMARY OF SPECIAL KEYS AND COMMANDS

Using special keyboard keys and commands, the programmer controls Editor's operation. Certain keys have special meaning to Editor, of which some can be used in either command or text mode. The mode of operation determines the function of each key. The special keys and their functions are shown in Table 2-6.

Table 2-6

Special Keys

KEY	COMMAND MODE	TEXT MODE
RETURN	Execute preceding command	Enter line in text buffer
←	Cancel preceding command (Editor responds with a ? followed by a carriage return and line feed)	Cancel line to the left margin

Table 2-6 (Cont.)

Special Keys

KEY	COMMAND MODE	TEXT MODE
RUBOUT	same as ←	Delete to the left one character for each depression; a \ (backslash) is echoed (not used in Read (R) command)
CTRL/FORM	Respond with question mark and remain in command mode	Return to command mode and ring teleprinter bell
. (period)	Value equal to decimal value of current line (may be used alone or with + or − and a number, e.g., .+8)	Legal text character
/	Value equal to number of last line in buffer; used as an argument	Legal text character
LINE FEED	List next line	Used in Search (S) command to insert CR/LF into line
ALTMODE	List next line	
>	List next line	
<	List previous line	
=	Used with . or / to obtain their value	
	Same as = (gives value of legitimate argument)	
CTRL/TAB		Produces a tab which on output is interpreted as 10 spaces or a tab/rubout, depending on SR option

Editor commands are given when in command mode. There are three basic types of commands: Input, Editing, and Output. Table 2-7 contains a summary of Editor commands and their function.

Table 2-7

Summary of Commands

TYPE	COMMAND	FUNCTION
Input	A	Append incoming text from keyboard into text buffer
	R	Append incoming text from tape reader into text buffer
Editing	L	List entire text buffer
	nL	List line n
	m,nL	List lines m through n inclusively
	nC	Change line n
	m,nC	Change lines m through n inclusively
	I	Insert before first line
	nI	Insert before line n
	K	Delete entire text buffer
	nD	Delete line n
	m,nD	Delete lines m through n inclusively
	m,n$jM	Move lines m through n to before line j
	G	Print next tagged line (if none, Editor types ?)
	nG	Print next tagged line after line n (if none, ?)
	S	Search buffer for character specified after RETURN key and allow modification (search character is not echoed on printer)
	nS	Search line n, as above
	m,nS	Search lines m through n inclusively, as above
Output	P	Punch entire text buffer
	nP	Punch line n
	m,nP	Punch lines m through n inclusively
	T	Punch about 6 inches of leader/trailer tape
	F	Punch a FORM FEED onto tape
	N	Do P, F, K, and R commands

m and n are decimal numbers, and m is smaller than n; j is a decimal number.

The P and N commands halt the Editor to allow the programmer to select I/O control; press CONT to execute these commands.

Commands are executed when the RETURN key is depressed, excluding the P and N commands.

PAL III SYMBOLIC ASSEMBLER

The PAL III Symbolic Assembler (PAL stands for Program Assembly Language) is an indispensable service program used to translate symbolic programs, which are written in the PAL III language, into binary-coded programs (binary programs).

PAL III is a two-pass assembler with an optional third pass, i.e., the symbolic program tape must be passed through the assembler two times to produce the binary-coded tape (binary tape), and the optional third pass produces a complete octal/symbolic program listing which can be

typed and/or punched if desired. A brief explanation of the three passes is given below.

Pass 1. The assembler reads the symbolic program tape and defines all symbols used and places the user symbols in a symbol table for use during Pass 2. The assembler checks for undefined symbols and certain other errors and types an error message on the teleprinter when an error is detected.

Pass 2. The assembler rereads the symbolic program tape and generates the binary tape using the symbols defined during Pass 1. When the low-speed punch is used, meaningless characters will be typed on the teleprinter, and these should be ignored by the programmer. The assembler checks illegal referencing during this pass and types an error message on the teleprinter when any is detected.

Pass 3. The assembler reads the symbolic program tape and types and/or punches the octal/symbolic program assembly listing. This listing thoroughly documents the assembled program and is useful when debugging and modifying the program.

The meaningless characters, error messages, and octal/symbolic program listing will be shown later in this section.

PAL III accepts symbolic program tapes from either the low-speed or high-speed reader and produces the binary tapes on either the low-speed or high-speed punch.

During assembly, the programmer communicates with PAL III via the switches on the computer console. Switch options are used to specify which pass the assembler is to perform and which reader and punch the assembler should accept input from and punch out on.

ASSEMBLING A SYMBOLIC PROGRAM

Earlier in this chapter, the programmer wrote a PAL III symbolic program and generated the symbolic program tape using Editor. That symbolic program can now be assembled to produce a binary program using PAL III. A listing of the symbolic program follows.

```
        *200
        CLA             /CLEAR AC
        TAD NUMB        /GET B
        CMA             /1'S COMP B
        IAC             /-B
        TAD NUMA        /ADD -B + A
        SMA             /IF -B LARGER
        JMP .+4         /JUMP 4 LOCATIONS
        CLA             /CLEAR AC
        TAD NUMB        /GET B
        HLT             /B IS LARGER
        CLA             /CLEAR AC
        TAD NUMA        /GET A
        HLT             /A IS LARGER
NUMA,   1111
NUMB,   0011
        S
```

First, PAL III must be loaded into core memory, and since PAL III is on punched paper tape in binary-coded format, it is loaded into core memory using the BIN Loader (see Figure 2-8 for loading procedures).

With PAL III in core, we are ready to assemble the symbolic program. Figures 2-12 and 2-13 illustrate the procedures for assembling with PAL III using the low-speed reader/punch and high-speed reader/punch, respectively. In these flowcharts, the switch register options are set for the appropriate reader/punch.

The low-speed reader and punch (LSR and LSP) are used in the following assembly (see Figure 2-13).

Initializing and Starting	Load PAL III into core memory using BIN. Set SR = 0200 and depress LOAD ADD. Turn TTY to LINE and put symbolic program tape in LSR .
Entering Pass 1	Set SR = 2200 and set LSR to START. Turn LSP ON and depress START.
NUMA 0215 NUMB 0216	Error messages would be typed now. Symbol table concludes Pass 1. Examine Symbol table. Make sure no "UA" diagnostics appear.
Entering Pass 2 BB: 8 8 = * <: <)	Put symbolic program tape in LSR. Set SR = 4200 and set LSR to START. Depress LSP to ON and depress CONT. Disregard meaningless characters while object tape is being punched. Error message would be typed now.
Entering Pass 3	Put symbolic program tape in LSR. Set SR = 6200 and set LSR to START. Depress LSP to ON and depress CONT. The octal/symbolic program listing is being typed and punched.

```
                        *200
0200    7200            CLA         /CLEAR AC
0201    1216            TAD NUMB    /GET B
0202    7040            CMA         /1'S COMP B
0203    7001            IAC         /-B
0204    1215            TAD NUMA    /ADD -B + A
0205    7500            SMA         /IF -B LARGER
0206    5212            JMP .+4     /JUMP 4 LOCATIONS
0207    7200            CLA         /CLEAR AC
0210    1216            TAD NUMB    /GET B
0211    7402            HLT         /B IS LARGER
0212    7200            CLA         /CLEAR AC
0213    1215            TAD NUMA    /GET A
0214    7402            HLT         /A IS LARGER
0215    1111    NUMA,   1111
0216    0011    NUMB,   0011
NUMA    0215
NUMB    0216
```

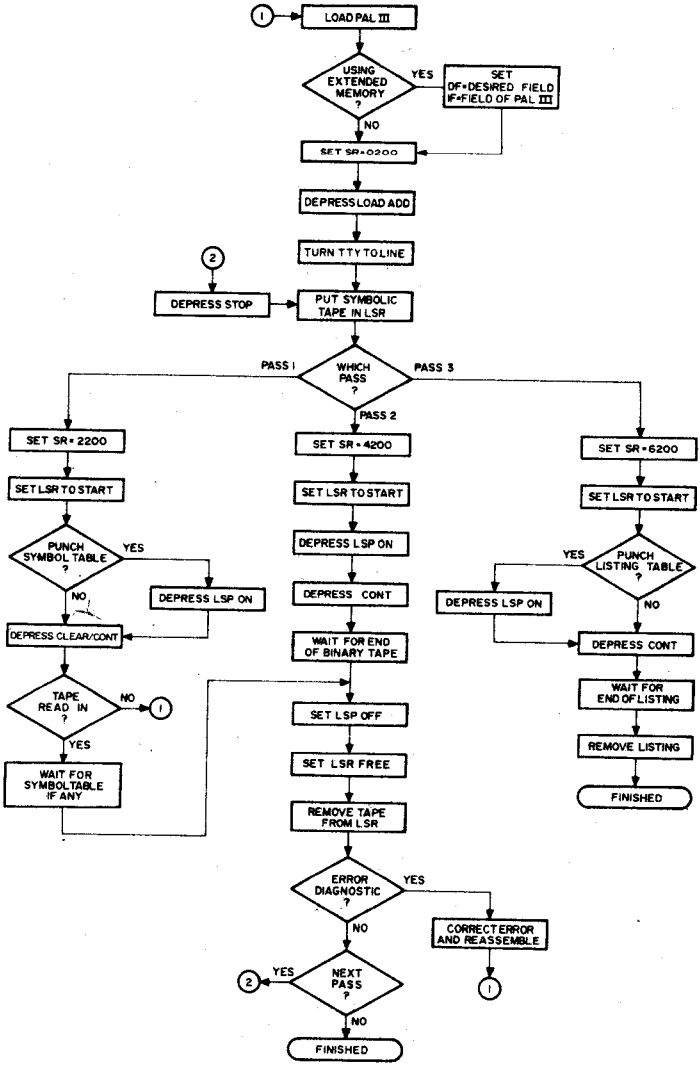


Figure 2-12 Assembling with PAL III Using Low-Speed Reader/Punch

The tape produced during Pass 2 is the binary tape, which is loaded into core memory using the BIN Loader. The symbol table tape produced during Pass 1, the binary tape produced during Pass 2, and the octal/symbolic program listing produced during Pass 3 are used when debugging the program.

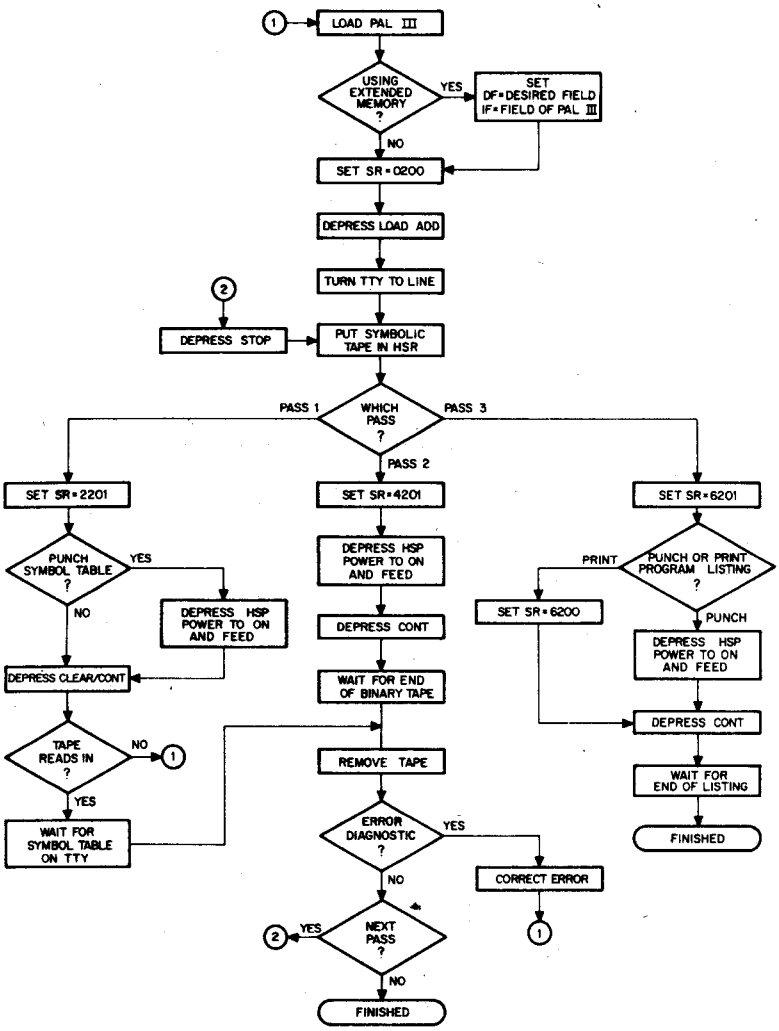


Figure 2-13 Assembling with PAL III Using High-Speed Reader/Punch

Off-Line Teletype Operation

The Teletype can be used separately from the computer for typing, punching tape, or duplicating tapes. To use the Teletye in this manner:

a. Ensure that the computer PANEL LOCK switch is positioned to the PANEL LOCK position.

b. Set the Teletype LINE/OFF/LOCAL switch to the LOCAL position.
c. If the punch is to be used, load it by raising the cover, manually feeding the tape from the top of the roll into the guide at the back of the punch, advancing the tape through the punch by manually turning the friction wheel, and then closing the cover. Energize the punch by pressing the ON pushbutton, and produce about two feet of leader. The leader-trailer can be code 200 or 377. To produce the code 200 leader, simultaneously press and hold the CTRL and SHIFT keys with the left hand, press and hold the REPT key, and press the @ (P) key. When the required amount of leader has been punched, release the @ key, and then all keys. To produce the 377 code, simultaneously press and hold both the REPT and RUB OUT keys until a sufficient amount of leader has been punched.

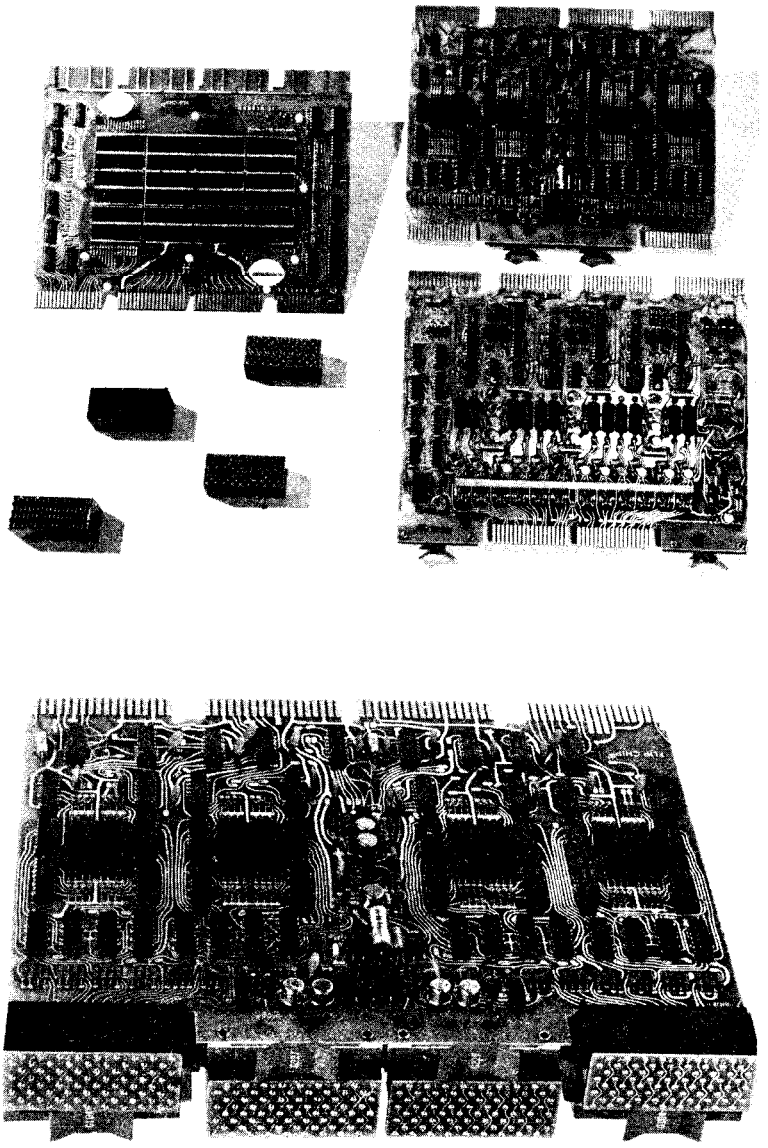
If an incorrect key is struck while punching a tape, the tape can be corrected as follows: If the error is noted after typing and punching any number (n) of characters, press the punch B.SP. (backspace) pushbutton n + 1 times and strike the keyboard RUB OUT key n + 1 times. Then continue typing and punching with the character which was in error.

To duplicate and obtain a listing of an existing tape: Perform the procedure under steps a through c above. Then load the tape to be duplicated as described in step b of the procedure listed under Loading Data Under Program Control. Initiate tape duplication by setting the reader START/STOP/FREE switch to the START position. The punch and teleprinter stops when the tape being duplicated is completely read. Corrections to insert or delete information on a perforated tape can be made by duplicating the correct portion of the tape and manually punching additional information, or inhibiting punching of information to be deleted. This is accomplished by duplicating the tape and carefully observing the information being typed as the tape is read. In this manner, the reader START/STOP/FREE switch can be set to the STOP position just before the point of the correction is reached. Information to be inserted can then be punched manually by means of the keyboard. Information can be deleted by pressing the punch OFF pushbutton and operating the reader until the portion of the tape to be deleted has been typed. It may be necessary to backspace and rub out one or two characters on the new tape if the reader is not stopped precisely on time. The number of characters to be rubbed out can be determined exactly by the typed copy. Be sure to count spaces when counting typed characters. Continue duplicating the tape in the normal manner after making the corrections.

New, duplicated, or corrected perforated tapes should be verified by reading them off-line and carefully proofreading the typed copy.

Program Control

When the program is stopped at the end of an instruction with the single step key, then the load address, examine, and deposit keys may be used without changing the AC. The program may then be resumed by resetting the PC to the address wanted, and then operating the CONTinue key.



4K Memory

CHAPTER 3

MEMORY AND PROCESSOR INSTRUCTIONS

GENERAL

An instruction is a coded program step that tells the computer what to do for a single operation in a program. In the PDP-8 family of computers, there are two major types of instruction words: memory reference and augmented. Memory reference instructions store or retrieve data from core memory, while augmented instructions do not. A third type—a housekeeping instruction is defined later. All instructions are determined by bits 0 through 2 to specify the operation (op) code. Operation codes of 0(octal) through 4(octal) specify memory reference instructions, and code of 6(octal) and 7(octal) specify augmented instructions. Memory reference instruction times are 1.2 μs for instruction fetches and non-auto-indexed defer cycles, and 1.4 μs for all other cycles. IOT (Input/Output Transfer) instructions are 1.2 μs for options which communicate directly with the OMNIBUS; and 2.6, 3.6, or 4.6 μs for options which communicate via the KA8-E Positive I/O Bus Interface. The latter times are + or −5%, all other times are + or −1%.

The instruction repertoire is shown in Figure 3-1. This illustrates the eight basic instructions and the division into the three categories—the Memory Reference Instructions, Augmented Instructions, and the Housekeeping Instruction.

The Memory Reference Instructions are concerned with at least two memory addresses, the address of the initial memory location and the address of the data. Figure 3-2 illustrates an example of the address flow of a Memory Reference Instruction. In order to illustrate the basic flow, a simplified diagram is shown leaving out some of the more complex branching such as JUMP and AUTO INDEX instructions. Two cycles are required (at a minimum) to complete an instruction. The indirect addressing capability requires an additional cycle (DEFER) which must go back to memory for another address. On the next cycle (EXECUTE), the data is brought into the Central Processor and the instruction is completed.

The Augmented Instruction simplified flow program is illustrated in Figure 3-3. Notice that the instruction is completely performed in the FETCH cycle. Once memory is addressed and the instruction brought from memory to the central processor, memory is not referenced (i.e., no additional instruction is brought from memory). Instead, the instruction is completely carried out once the initial decoding has been accomplished. The last 9 bits, instead of being used to specify a memory address, are used as an extension of the basic instruction. The Housekeeping Instruction, JMP (sometimes called Unconditional Branch) is used only to force any desired address into the Program Counter. (See Program Control in chapter 4)

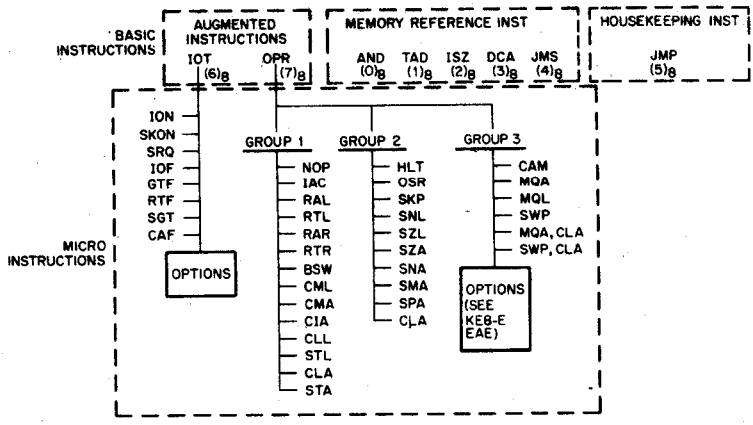


Figure 3-1 PDP-8/E Instruction Repertoire

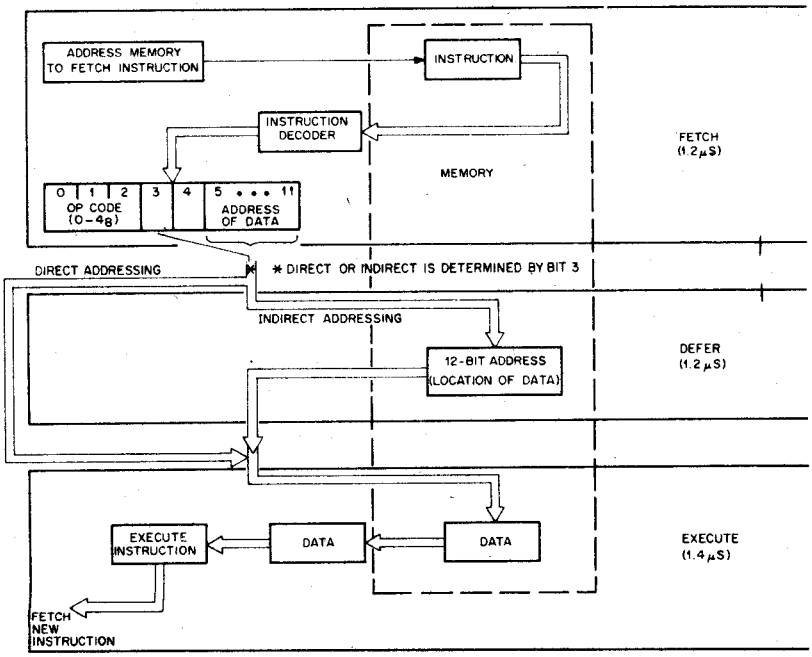


Figure 3-2 Memory Reference Instruction Simplified Flow

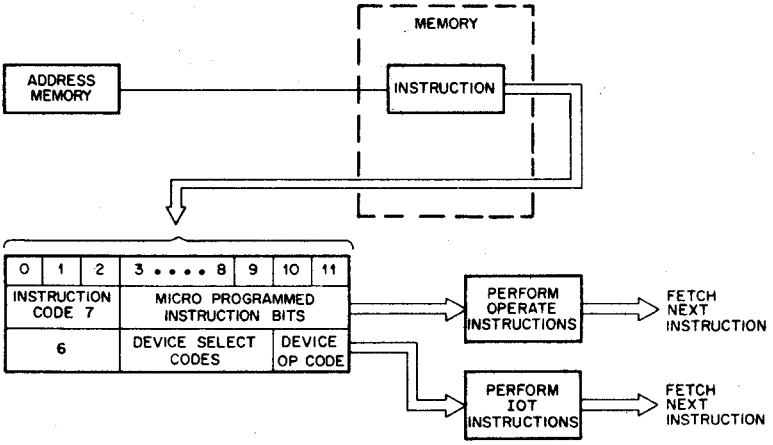


Figure 3-3 Augmented Instruction Simplified Flow Diagram

MEMORY REFERENCE INSTRUCTIONS

Logical AND (AND Y)

Octal Code: 0
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The AND operation is performed between the contents of memory location Y and the contents of AC. The result is left in the AC, the original contents of the AC are lost, and the contents of Y are restored. Corresponding bits of the AC and Y are operated upon independently. This instruction, often called extract or mask, can be considered as a bit-by-bit multiplication.

Example:

Original ACJ	YJ	Final ACJ
0	0	0
0	1	0
1	0	0
1	1	1

Two's Complement Add (TAD Y)

Octal Code: 1
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.

Operation: The contents of memory location Y are added to the contents of the AC in two's complement arithmetic. The result of this addition is held in the AC, the original contents of the AC are lost and the contents of Y are restored. If there is a carry from AC0, the link is complemented. This feature is useful in multiple precision arithmetic.

Increment and Skip if Zero (ISZ Y)

Octal Code: 2
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The contents of memory location Y are incremented by one in two's complement arithmetic. If the resultant contents of Y equal zero, the contents of the PC are incremented by one and the next instruction is skipped. If the resultant contents of Y do not equal zero, the program proceeds to the next instruction. The incremented contents of Y are restored to memory. The contents of the AC are not affected by this instruction.

Deposit and Clear AC (DCA Y)

Octal Code: 3
Major States: F, (D), E
Execution time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The contents of the AC are deposited in core memory at address Y and the AC is cleared. The previous contents of memory location Y are lost.

Jump to Subroutine (JMS Y)

Octal Code: 4
Major States: F, (D), E
Execution Time: 2.6 μs with direct addressing, 3.8 μs with indirect addressing, 4.0 μs with auto-indexed indirect addressing.
Operation: The contents of the PC are deposited in core memory location Y and the next instruction is taken from core memory location Y + 1. The contents of the AC are not affected by this instruction.

HOUSEKEEPING INSTRUCTION
Jump to Y (JMP Y)

Octal Code: 5
Major States: F, (D)
Execution Time: 1.2 μs with direct addressing, 2.4 μs with indirect addressing, 2.6 μs with auto-indexed indirect addressing.

Operation: Address Y is loaded into the PC so that the next instruction is taken from core memory address Y. The original contents of the PC are lost. The contents of the AC are not affected by this instruction.

AUGMENTED INSTRUCTIONS

Augmented instructions are one-cycle (FETCH) instructions that initiate various operations as a function of bit microprogramming. Augmented instructions are divided into two categories, neither of which are memory reference instructions. These are the INPUT/OUTPUT TRANSFER (IOT), which has an operation code of six; and the OPERATE, which has an operation code of seven. Bits 3 through 11 within each instruction function as an extension of the operations to be performed.

OPERATE INSTRUCTION

The operate instruction consists of three groups of microinstructions. Group 1 (OPR 1) is principally for clear, complement, rotate, and increment operations and is designated by the presence of a 0 in bit 3. Group 2 (OPR 2) is used principally in checking the contents of the accumulator and link and continuing to, or skipping, the next instruction based on the check. A 1 in bit 3 and a 0 in bit 11 designates an OPR 2 microinstruction. Group 3 is used to manipulate data between the MQ and AC Registers. A 1 in bits 3 and 11 designate an OPR 3 microinstruction. Each instruction is completed during a FETCH cycle. All operate instructions take place in 1.2 microseconds.

Group 1

The Group 1 microinstructions manipulate the contents of the accumulator and link. These instructions are microprogrammable; that is, they can be combined to perform specialized operations with other Group 1 instructions.

The Group 1 operate microinstruction format is shown in Figure 3-4, and the microinstructions are explained in the succeeding paragraphs. Bits within this group can be combined into one microinstruction. For example, it is possible to assign 1's to bits 5, 6, and 11, thereby creating a single instruction which clears the link, complements the accumulator and increments the accumulator. (The most frequently used combinations are listed in Appendix B.)

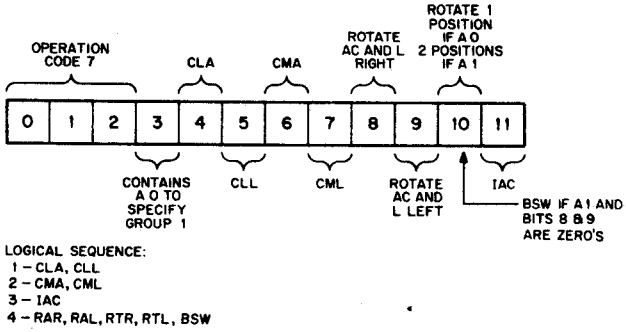


Figure 3-4 Group 1 Operate Instruction Bit Assignments

No Operation (NOP)

Octal Code: 7000
Sequence: None
Operation: This command causes a 1-cycle delay in the program before the next sequential instruction is initiated. This command is used to add execution time to a program, such as to synchronize subroutine or loop timing. The NOP also provides the programmer with a convenient means of removing an instruction.

Increment Accumulator (IAC)

Octal Code: 7001
Sequence: 3
Operation: The contents of the AC are incremented by one in two's complement arithmetic.

Rotate Accumulator Left (RAL)

Octal Code: 7004
Sequence: 4
Operation: The contents of the AC and link are rotated one binary position to the left. The contents of bits AC1-11 are shifted to the next greater significant bit, the content of AC0 is shifted into the L, and the content of the L is shifted into AC11.

Rotate Two Left (RTL)

Octal Code: 7006
Sequence: 4
Operation: The contents of the AC and link are rotated two binary positions to the left. This instruction is logically equal to two successive RAL operations.

Rotate Accumulator Right (RAR)

Octal Code: 7010
Sequence: 4
Operation: The contents of the AC and link are rotated one binary position to the right. The contents of bits AC0-10 are shifted to the next less significant bit, the content of AC11 is shifted into the L, and the content of the L is shifted into AC0.

Rotate Two Right (RTR)

Octal Code: 7012
Sequence: 4
Operation: The contents of the AC and link are rotated two binary positions to the right. This instrucion is logically equal to two successive RAR operations.

Byte Swap (BSW)

Octal Code: 7002
Sequence: 4
Operation: The right six bits of the accumulator are exchanged with the left six bits. AC0 is exchanged with AC6; AC1 with AC7, etc. The contents of the link are not affected.

Complement Link (CML)

Octal Code: 7020
Sequence: 2
Operation: The content of the L is complemented.

Complement Accumulator (CMA)

Octal Code: 7040
Sequence: 2
Operation: The contents of the AC are changed to the one's complement of the current contents of the AC. The content of each bit of the AC is complemented individually.

Complement and Increment Accumulator (CIA)

Octal Code: 7041
Sequence: 2, 3
Operation: The contents of the AC are converted from a binary value to their equivalent two's complement number. This conversion is accomplished by combining the CMA and IAC commands, thus the contents of the AC are complemented during sequence 2 and are incremented by one during sequence 3.

Clear Link (CLL)

Octal Code: 7100
Sequence: 1
Operation: The L is cleared (made equal to 0).

Set Link (STL)

Octal Code: 7120
Sequence: 1, 2
Operation: The L is set. This instruction is logically equal to combining the CLL and CML commands.

Clear Accumulator (CLA)

Octal Code: 7200
Sequence: 1
Operation: The content of each bit of the AC is cleared (made equal to 0).

Set Accumulator (STA)

Octal Code: 7240
Sequence: 1, 2
Operation: Each bit of the AC is set. This operation is logically equal to combining the CLA and CMA commands.

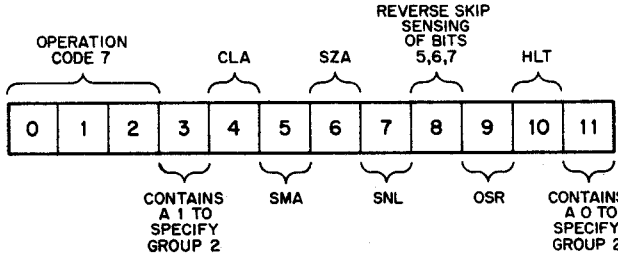
NOTE: The following codes are illegal and specifically reserved for future expansion: RAL RAR (octal code 7014), RTL RTR (octal code 7016) and any microprogramming of these bit combinations.

GROUP 2

Group 2 Operate microinstructions are often referred to as the "skip microinstructions" because they enable the programmer to perform tests on the accumulator and link and to skip the next instruction depending upon the results of the test. A skip instruction causes the computer to check for a specific condition, and if it is present, to skip the next instruction. If the condition is not present, the next instruction is executed.

The Group 2 operate microinstruction format is shown in Figure 3-5 and the primary microinstructions are explained in the following paragraphs. Any logical combination of bits within this group can be combined into one instruction. (The instructions constructed by most logical bit combinations are listed in Appendix B.)

If skips are combined in a single instruction the inclusive OR of the conditions determines the skip when bit 8 is a 0; and the AND of the inverse of the conditions determines the skip when bit 8 is a 1. For example, if 1s are designed in bits 6 and 7 (SZA and SNL), the next instruction is skipped if either the contents of the AC = 0, or the content of L= 1. If 1s are contained in bits 5, 7 and 8, the next instruction is skipped if the AC contains a positive number and the L contains a 0.



LOGICAL SEQUENCE:

1 (BIT 8 IS A 0) – EITHER SMA OR SZA OR SNL
(BIT 8 IS A 1) – BOTH SPA AND SNA AND SZL
2 – CLA
3 – OSR
4 – HLT

Figure 3-5 Group 2 Operate Instruction Bit Assignments

Halt (HLT)

Octal Code: 7402
Sequence: 4
Operation: Clears the RUN flip-flop at Sequence 4, so that the program stops at the conclusion of the current machine

cycle. This command can be combined with others in the OPR 2 group. All other OPR Group 2 Instructions are performed before the program stops.

OR with Switch Register (OSR)

Octal Code: 7404
Sequence: 3
Operation: The inclusive OR operation is performed between the contents of the AC and the contents of the SR. The result is left in the AC, the original contents of the AC are lost. The contents of the SR are unaffected by this command. When combined with the CLA command, the OSR performs a transfer of the contents of the SR into the AC.

Skip, Unconditional (SKP)

Octal Code: 7410
Sequence: 1
Operation: The contents of the PC are incremented by one so that the next sequential instruction is skipped.

Skip on Non-Zero Link (SNL)

Octal Code: 7420
Sequence: 1
Operation: The content of the L is sampled, and if it contains a 1, the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the L contains a 0, no operation occurs and the next sequential instruction is initiated.

Skip on Zero Link (SZL)

Octal Code: 7430
Sequence: 1
Operation: The content of the L is sampled, and if it contains a 0 the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the L contains a 1, no operation occurs and the next sequential instruction is initiated.

Skip on Zero Accumulator (SZA)

Octal Code: 7440
Sequence: 1
Operation: The content of each bit of the AC is sampled, and if all bits contain a 0 the contents of the PC are incremented by one so that the next sequential instruction is skipped. If any bit of the AC contains a 1, no operation occurs and the next sequential instruction is initiated.

Skip on Non-Zero Accumulator (SNA)

Octal Code: 7450
Sequence: 1
Operation: The content of each bit of the AC is sampled, and if any bit contains a 1 the contents of the PC are incremented by one so that the next sequential instruction is skipped. If all bits of the AC contain a 0, no operation occurs and the next sequential instruction is initiated.

Skip on Minus Accumulator (SMA)

Octal Code: 7500
Sequence: 1
Operation: The content of the most significant bit of the AC is sampled, and if it contains a 1, indicating that the AC contains a negative two's complement number, the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the AC contains a positive number no operation occurs and the next sequential instruction is initiated.

Skip on Positive Accumulator (SPA)

Octal Code: 7510
Sequence: 1
Operation: The content of the most significant bit of the AC is sampled, and if it contains a 0, indicating a positive (or zero) two's complement number, the contents of the PC are incremented by one so that the next sequential instruction is skipped. If the AC contains a negative number, no operation occurs and the next sequential instruction is initiated.

Clear Accumulator (CLA)

Octal Code: 7600
Sequence: 2
Operation: Each bit of the AC is cleared to contain a binary 0.

Group 3

The Group 3 Operate microinstructions are concerned with the manipulation of data between the AC and the MQ registers. The MQ register is an auxiliary register for the temporary storage of data. It is sometimes convenient to temporarily store data in an auxiliary register rather than in a memory location. Group 3 instructions enable the loading of AC contents into the MQ; the loading of the MQ contents into the AC, the swapping of AC and MQ contents; the loading of the inclusive OR of the contents of the AC and MQ into the AC; and the clearing of both the AC and MQ. Although the register and instructions are primarily intended to be used with the Extended Arithmetic KE8-E option, they are available for use as a standard feature.

The format of the Group 3 instructions is shown in Figure 3-6. Having an operation code of 7, this instruction class is identified as group 3 only when both bits 3 and 11 contain a 1.

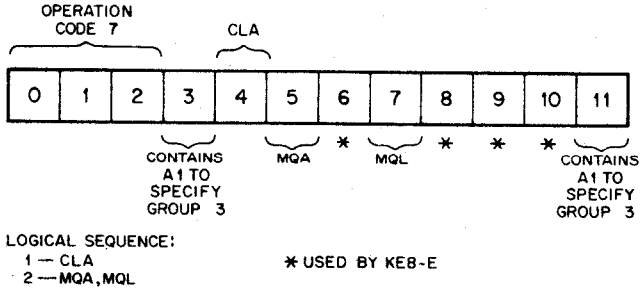


Figure 3-6 Bit Assignments for Group 3 Operate Instructions

Bits 6, 8, 9 and 10 should be zero when the KE8-E option is not employed. The description of the four Group 3 instructions is given in the following:

Instructions

Clear Accumulator and Multiplier Quotient (CAM)

Octal Code: 7621
Execution Time: 1.2 μs
Operation: Clears the AC during logical sequence 1, as in CLA; during logical sequence 2, the MQ is cleared.

Multiplier Quotient Load into Accumulator (MQA)

Octal Code: 7501
Execution Time: 1.2 μs
Operation: The contents of the MQ are inclusively ORed with the AC, and the result loaded into the AC. The previous contents of the AC are lost, but the contents of the MQ are not affected. This instruction provides the programmer with a direct inclusive OR instruction. This instruction may also be combined with a CLA instruction to effect a direct transfer of information from MQ to AC.

Load Multiplier Quotient (MQL)

Octal Code: 7421
Execution Time: 1.2 μs
Operation: Loads the content of the AC into the MQ, and then clears the AC.

Swap MQ and AC (SWP)

Octal Code: 7521
Execution Time: 1.2 μs
Operation: The contents of AC and MQ are exchanged. This instruction can be combined with the CLA bit

to move the contents of MQ to AC and then clear the MQ.

INPUT/OUTPUT TRANSFER (IOT)

Octal Code: 6
Major State: F
Execution Time: If the selected device is internal, the IOT takes place in 1.2 microseconds.
If the selected device is external, the computer enters an expanded cycle of 2.6 microseconds (if the IOP ends in 1, 2 or 4); 3.6 microseconds (if the IOP ends in 3, 5, or 6); or 4.6 microseconds (if the IOP ends in 7). An IOT ending in 0 always takes place in 1.2 microseconds.

Operation: Input/output transfer (IOT) instructions initiate the operation of peripheral equipment and effect information transfers between the processor and an I/O device. Upon recognition of the operation code 6 as an IOT instruction, the computer determines whether the selected device is internal (plugged directly into the OMNIBUS) or external (connected via the KA8-E Positive I/O Bus Interface module). The nature of the OMNIBUS is such that IOT's such as 6000 can be used for control codes for devices directly connected to the OMNIBUS, since the last 3 bits are decoded to determine the device operation.

The last 3 bits of the instruction cause the generation of "IOP PULSES" at the External Bus Interface as follows:

Instruction Bit	IOP	Sequence Time
11	IOP 1	1
10	IOP 2	2
9	IOP 4	3

IOP pulses enact a data transfer or initiate a control operation. Selection of an equipment is accomplished by bits 3 through 8 of the IOT instruction. These bits form a 6-bit code that enables the device selector in a given device.

The format of the IOT instruction is shown in Figure 3-7. Operations performed by IOT microinstructions are explained in Chapters 5, 9, and 10.

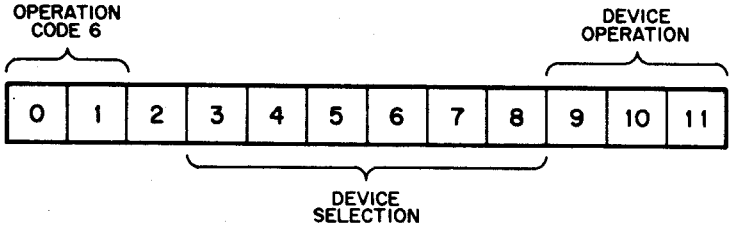


Figure 3-7 IOT Instruction Bit Assignment

PROGRAM INTERRUPT

The program interrupt features allow certain external conditions to interrupt the computer program. Program interrupts are used to either speed the information processing of input/output devices or allow certain alarms to halt the program in progress and initiate another routine. When a program interrupt request is made, the computer completes execution of the instruction in progress before acknowledging the request and entering the interrupt mode. A program interrupt is similar to a JMS to location 0; that is, the contents of the program counter are stored in location 0, and the program resumes operation in location 1 with the interrupt disabled. The interrupt program commencing in location 1 is responsible for identifying the signal causing the interruption, for removing the interrupt condition, and for returning to the original program with the interrupt re-enabled. Exit from the interrupt program, back to the original program, can be accomplished by a JMP I 0 instruction.

When an interrupt request is acknowledged, the interrupt is automatically disabled by the program interrupt synchronization circuits (not by instructions). The next instruction is taken from core memory location 1. Usually, the instruction stored in locations 1 is a JMP, which transfers program control to a subroutine which services the interrupt. At some time during this subroutine, an ION instruction must be given. The ION can be given at the end of the subroutine, just before control is transferred back to the original program. In this application, the ION instruction immediately precedes the last instruction in the routine. A delay of one instruction (regardless of the execution time of the following instruction), is inherent in the ION instruction to allow transfer of program control back to the original program before enabling the interrupt. Exit from the subroutine usually is accomplished by a JMP I 0 instruction.

The ION command can also be given during the subroutine as soon as the I/O device causing the interrupt has been identified. This latter method allows the subroutine which is handling a low priority interrupt to be interrupted, possible by a high priority device. Programming of an interrupt subroutine, which checks for priority and allows itself to be interrupted, must make provisions to relocate the contents of the program counter stored in location 0; so that the return address to the original program is not lost if another interrupt occurs.

Instructions

Interrupt Turn ON (ION)

Octal Code: 6001

Operation: This command enables the computer to respond to a program interrupt request. If the interrupt is disabled when this instruction is given, the computer executes the next instruction, then enables the interrupt. The additional instruction allows exit from the interrupt subroutine before allowing another interrupt to occur. This instruction has no effect upon the condition of the interrupt circuits if it is given when the interrupt is enabled.

Skip If Interrupt ON (SKON)

Octal Code: 6000

Operation: The state of the interrupt enable flip-flop is tested. If this flip-flop is set, the next sequential instruction is skipped. Simultaneously with this test (and before a device flag can cause an interrupt) the interrupt system is turned off as described under the following IOF instruction.

Interrupt Turn Off (IOF)

Octal Code: 6002

Operation: This command causes the program interrupt feature to be disabled.

Skip If Interrupt Request (SRQ)

Octal Code: 6003

Operation: The state of the internal interrupt request bus is tested. If it is low, indicating one or more devices are requesting an interrupt, the next instruction is skipped.

Flag Processing IOTs

Get Flags (GTF)

Operation: 6004

Octal Code: The following machine states are read into the indicated bits of the accumulator:

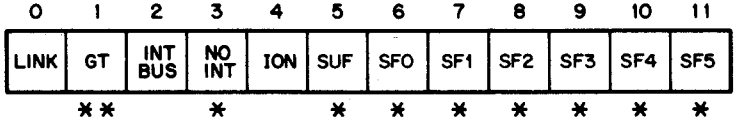


Figure 3-8 Flag Processing States

* Only if Extended Memory Control, type KM8-E, installed.

** Only if Extended Arithmetic Element (EAE), type KE8-E, installed.

Restore Flags (RTF)

Octal Code 6005

Operation This instruction is the converse of GTF. The bits in the AC (see figure 3-8) are used to set the corresponding flip-flops in the processor, the KM8-E Extended Memory Control or the KE8-E Extended Arithmetic Element. RTF enables the interrupt in the same manner as an ION instruction. Refer to the KM8-E Memory Extension and Time-Share Option in Chapter 7 for more details on the RTF instruction.

Skip if Greater Than (SGT)

Octal Code: 6006
Operation: If the GT flag is set, the next instruction is skipped. This instruction is implemented only if the KE8-E is installed.

Clear all Flags (CAF)

Octal Code: 6007
Operation: This instruction is logically equivalent to operating the CLEAR key on the panel. It generates INITIALIZE on the OMNIBUS and at the external I/O interface. The LINK and AC are cleared. The action of INITIALIZE is a function of the design of each peripheral, but generally INITIALIZE clears flags and motion control flip-flops, and sets the interrupt enable flip-flop of all peripherals.

NOTE

A CAF instruction should not be given when a device is active. For example: A CAF instruction should not be given until at least 100ms after a TLS instruction.

INSTRUCTION SUMMARY

A summary of the common usage of the eight basic instructions is provided in Table 3-1. As the reader continues on with the remaining chapters in this handbook, he should keep in mind the capability of all eight instructions since the processor's operation is determined by the use of each instruction. A program, for instance, is simply a collection of instructions strung out in a particular order for a particular purpose such as solving a problem. Each instruction performs a series of steps to satisfy the requirements of the program.

Table 3-1
Basic Instruction Usage Summary

BASIC INSTRUCTION	MINIMUM NO. OF CYCLES	COMMON USAGE	EXAMPLE OF USAGE
AND (0)(octal)	2	Data Manipulation	Strips unwanted bits from the AC.
TAD (1)(octal)	2	Data Manipulation	Provides arithmetic addition. Also serves to load the AC with the contents of some memory location.
ISZ (2)(octal)	2	Program Loops	Used for counting.
DCA (3)(octal)	2	Data Manipulation	Used when placing data into some memory location.
JMS (4)(octal)	2	Subroutine Entry	Provides entry to subroutines.
JMP (5)(octal)	1	Manipulation of Program Counter	Allows the programmer to go to a different portion of his program. Also provides subroutine exit.
IOT (6)(octal)	1	External Communication	Allows the program to converse with peripherals.
OPR (7)(octal)	1	Testing of AC and Link	Operates on and/or tests the contents of the AC and Link. Operates on the contents of the MQ Register.