DEC-9A-MAB0-D

MONITORS

ADVANCED
Software System

PDP-9

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

MONITORS
Programmer's Reference Manual
ADVANCED Software System

Order No. DEC-9A-MAB0-D from Program Library, Maynard, Mass. Price $2.00
Direct comments concerning this manual to Software Quality Control, Maynard, Mass.

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Original Printing March 1967
Revised February 1968

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

ILLUSTRATIONS

PREFACE

This Monitor programmer's reference manual contains information needed to prepare programs for operation in the PDP-9 ADVANCED Software System. Both the I/O Monitor (paper tape systems) and the Keyboard Monitor (bulk storage systems) are described.

Chapter 2 contains information essential to programming both Monitors. The following topics should be studied carefully by all PDP-9 programmers:

a. Data flow in and out of the user's program, including available data modes (ASCII, binary, etc.) and device assignment tables, which provide the linkage between the user program and the supplied device handlers.

b. System I/O macros to perform all I/O functions. Format and use of each instruction is explained.

c. Line buffers, through which all data passes between the user's program and the various I/O devices. Buffer headers and standard sizes are discussed.

d. Examples of buffers and some typical input/output operations using system macros.

For the convenience of PDP-9 ADVANCED Software System programmers, a new operating guide for users of paper tape systems, prepared especially for use at the computer console, is now available. This manual is called "I/O Monitor Guide" (Paper Tape Systems) Order Number DEC-9T-NGAA-D. The I/O Monitor Guide contains all of the essential information needed to operate the system, including detailed loading procedures for system programs, and summaries of console commands.

CHAPTER 1
PDP-9 ADVANCED SOFTWARE SYSTEM

1.1 INTRODUCTION

PDP-9 ADVANCED software provides a complete system for program preparation, compilation, assembly, debugging, and operation. It features total relocatability and can expand to take full advantage of any hardware configuration. Powerful system programs include FORTRAN IV, a sophisticated macro assembler, an on-line debugging system, an on-line editor, and a peripheral interchange program. A versatile and flexible input/output programming system frees the user from the need to create device-handling subroutines and the concerns of device timing.

Two monitor systems are available with PDP-9 ADVANCED software. The Input/Output Monitor operates on a basic PDP-9 with 8,192 (or more) words of memory, high-speed paper tape reader and punch, and a console teleprinter. The I/O Monitor operates in a paper tape (or card) environment and provides for the calling and handling of all input and output functions.

A more sophisticated Keyboard Monitor is available for systems with bulk auxiliary-storage units. It allows for device-independent programming and automatic creation, calling, and loading of programs. Since the Input/Output Monitor is a subset of the Keyboard Monitor, all programs prepared for use with it can be run on the more sophisticated system.

1.2 HARDWARE REQUIREMENTS

To operate the PDP-9 ADVANCED software system under control of the Input/Output Monitor, a basic PDP-9 is required with:

a. 8,192 words of core memory

b. 300 character-per-second paper tape reader

c. 50 character-per-second paper tape punch

d. Console teleprinter (Teletype Model KSR 33 or KSR 35)

Extra memory, the Type KE09A Extended Arithmetic Element and the Type KF09A Automatic Priority Interrupt, if present, can also be utilized by this system. A card reader, Type CR02B, can be used for input in addition to the paper tape reader, and a Type 647 Line Printer can be used for listings.

To utilize the Keyboard Monitor, some form of bulk storage must be added to the basic PDP-9:

a. Type TC02 DECtape Control and two Type TU55 DECtape Transports.

b. Type TC59 Magnetic Tape Control and two 7-channel or 9-channel Magnetic Tape Transports (Type TU20, Type TU20A, or equivalent).

c. Type RM09 Block Transfer Drum System

d. Type RC09 Fixed-Head Disk System

Input/output routines will be provided for these devices.

This system, too, can take full advantage of extra memory, central processor options, and additional input/output options.

1.3 MONITOR SYSTEMS

PDP-9 ADVANCED software monitor systems exist to simplify the handling of input/output functions and to facilitate the creation, debugging, and use of PDP-9 programs. They allow overlapped input/output and computation, simultaneous operation of a number of asynchronous peripheral devices, and (in the case of the Keyboard Monitor) device-independent programming -- all while freeing the user from the need to create device handling subroutines. The Monitors, operating in conjunction with the Input/Output Programming System (IOPS), provide a complete interface between the user's programs and the peripheral hardware.

1.3.1 Input/Output Monitor

The Input/Output Monitor accepts I/O commands from the system or user programs and supervises their execution. By calling upon the device manipulation routines of IOPS, it can allow simultaneous I/O and computation.

The I/O Monitor contains:

a. Routines for its own initialization and control.

b. Tables to allow communication between the Monitor, system programs, user programs, and the Input/Output Programming System.

c. The CAL Handler, which is used to dispatch to the appropriate Monitor subroutines.

d. Device handlers for the Teletype and clock.

The I/O Monitor resides in lower core and occupies about 775_{10} locations.

1.3.2 Keyboard Monitor

The Keyboard Monitor is designed to operate on a PDP-9 system with some form of auxiliary bulk storage (see Hardware Requirements, Section 1.2). It includes all of the facilities of the I/O Monitor plus routines to accept and act upon keyboard commands, the ability to dynamically modify I/O device assignments for a program, and the facilities for automatically storing, calling, loading, and executing system and user programs.

With the ability to alter I/O assignments, the Keyboard Monitor brings to the user true device independence. Programs may be swiftly and simply modified to operate on any configuration, and additions to (or deletions from) an existing system need not result in program reassembly or recompilation.

The Keyboard Monitor also frees the user from the problems of tape or card handling. At the keyboard console, programs can be created, stored, retrieved, loaded, debugged, and operated. Both system and user programs can be called from the bulk storage device with a few simple keyboard commands.

1.3.3 Input/Output Programming System (IOPS)

The Input/Output Programming System (IOPS) consists of an I/O control routine (basically a CAL handler) and individual hardware device handling subroutines which process file and data level commands to the devices. These handlers exist for all standard PDP-9 peripherals (see Section 4.6).

The I/O control routine accepts user program commands and transfers control to the appropriate device handlers. These device handlers are responsible for transferring data between the program and I/O devices, for initiating the reading or writing of files, for the opening and closing of files, and for the performance of all other functions peculiar to a given hardware device. They are also responsible for ignoring functions of which they are incapable (e.g., trying to rewind a card reader or skipping files on a non-file-oriented device). All device handlers will operate with or without the automatic priority interrupt (API) option.

1.4 SYSTEM PROGRAMS

PDP-9 ADVANCED software systems include either the Input/Output Monitor (basic systems) or the Keyboard Monitor (bulk storage systems), the Input/Output Programming System, and the following system programs:

a. FORTRAN IV Compiler, Operating System, and Library.
b. Macro Assembler (MACRO-9)
c. Debugging System (DDT-9)
d. Text Editor (EDIT-9)
e. Peripheral Interchange Program (PIP-9)
f. Linking Loader (LINK-9)
g. PDP-7 Assembly Language to MACRO-9 Assembly Language Converter (CONV-9)
h. System Generator (SGEN-9)
i. DUMP (DUMP-9)
j. Library Update (UPDATE-9)

} With Keyboard Monitor system only

A brief introduction to each is given below.

1.4.1 FORTRAN IV

The PDP-9 FORTRAN IV compiler is a two-pass system which accepts statements written in the FORTRAN IV language and produces a relocatable object code capable of being loaded by the Linking Loader. It is completely compatible with USA FORTRAN IV, as defined in USA Standard X3.9-1966, with the exception of the following features which were modified to allow the compiler to operate in 8,192 words of core storage:

a. Complex arithmetic will not be available.

b. Adjustable array dimensions will not be allowed.

c. Blank Common will be treated as named Common.

d. The implied DO feature will be deleted from the DATA statement.

e. Specification statements must be strictly positioned and ordered.

This FORTRAN IV compiler operates with the PDP-9 program interrupt facility enabled. It generates programs that operate with the program interrupt enabled and can work in conjunction with assembly language programs that recognize and service real-time devices. Subroutines written in either FORTRAN IV or MACRO-9 assembly language can be loaded with and called by FORTRAN IV main programs. Comprehensive source-language diagnostics are produced during compilation, and a symbol table is generated for use in on-line debugging with DDT-9.

PDP-9 FORTRAN IV is described fully in the PDP-9 ADVANCED Software FORTRAN IV Manual (DEC-9A-AF40-D).

1.4.2 Macro Assembler (MACRO-9)

With the Macro Assembler, PDP-9 users are able to utilize highly sophisticated macro generating and calling facilities within the context of a symbolic assembler. Among the more prominent features of MACRO-9 are:

a. The ability to

1. define macros
2. define macros within macros (nesting)
3. re-define macros (in or out of macro definitions)
4. call macros within macro definitions
5. have macros call themselves (recursion)

b. Conditional assembly based on the computational results of symbols or expressions.

c. Repeat functions.

d. Boolean manipulation.

e. Optional octal and symbolic listings.

f. Two forms of radix control (octal, decimal) and two text modes (ASCII and 6-bit trimmed ASCII).

g. Global symbols for easy linking of separately assembled programs.

h. Choice of output format: relocatable, absolute binary (check summed); or full binary capable of being loaded via the hardware READIN switch.

j. Ability to call input/output system macros which expand into IOPS calling sequences.

MACRO-9 is described in detail in the MACRO-9 Manual (DEC-9A-AM9B-D).

1.4.3 Debugging System (DDT-9)

DDT-9 provides on-line debugging facilities within the PDP-9 ADVANCED software system. With it, the user may load and operate his program in a real-time environment while maintaining strict control over the running of each section. DDT-9 allows the operator to insert and delete breakpoints, examine and change registers, patch programs, and search for specific constants or word formats.

The DDT-9 breakpoint feature allows for the insertion and simultaneous use of up to four breakpoints, and any one or all of which may be removed with a single keyboard command. The search facility allows the operator to specify a search through any part of all of an object program with printout of the locations of all registers that are equal (or unequal) to a specified constant. This search feature also works for portions of words as modified by a mask. With DDT-9, registers may be examined and modified in either instruction format or octal code, and addresses may be specified in symbolic relative, octal relative, or octal absolute. Patches may be inserted in either source language or octal.

DDT-9 is described more fully in the PDP-9 Utility Programs Manual (DEC-9A-GUAB-D).

1.4.4 Text Editor (EDIT-9)

The Text Editor of the PDP-9 ADVANCED software system provides the ability to read alphanumeric text from any input device (paper tape reader, card reader, disk, drum, DECtape, magnetic tape, etc.), to examine and correct it, and to write it on any output device. It can also be used to create new symbolic programs.

The Editor operates on lines of symbolic text delimited by carriage return (CR) characters. These lines can be read into a buffer, selectively examined, deleted or modified, and written out. New text may be substituted, inserted, or appended.

For further details of EDIT-9 see PDP-9 Utility Programs Manual (DEC-9A-GUAB-D).

1.4.5 Peripheral Interchange Program (PIP-9)

The primary function of PIP-9 is to facilitate the manipulation and transfer of data files from any input device to any output device. It can be used to update file descriptions, delete, insert, or combine files, perform code conversions, and rewind tapes.

Directions for the use of PIP-9 can be found in the PDP-9 Utility Programs Manual (DEC-9A-GUAB-D).

1.4.6 Linking Loader (LINK-9)

The Linking Loader loads any PDP-9 FORTRAN IV or MACRO-9 object program which exists in relocatable or absolute format. Among its tasks are loading and relocation of programs, loading of

called subroutines, retrieval and loading of implied subroutines and IOPS routines, and building and relocation of the necessary symbol tables. Its operation is discussed in the PDP-9 Utility Program Manual (DEC-9A-GUAB-D).

1.4.7 PDP-7 Assembly Language to MACRO-9 Assembly Language Converter (CONV-9)

This system program converts source programs written in PDP-7 or basic PDP-9 assembly language to a format acceptable to the MACRO-9 assembler.

CONV-9 is described more fully in the PDP-9 Utility Programs Manual (DEC-9A-GUAB-D).

1.4.8 System Generator (SGEN-9)

The System Generator is a standard system program used to create new system tapes. With it, the user can tailor his system to his installation's needs and specify standard input and output devices, memory size, and special I/O and central processor options present. A more complete description of SGEN-9, and details of its use, are given in Section 3.4.4.7.

1.4.9 DUMP (DUMP-9)

This system program gives the user the ability to output on any listing device specified core locations that had been preserved on a bulk storage file via the ↑Q (control key/Q) monitor dump command (see Section 3.2.3.5).

1.4.10 Library Update (UPDATE-9)

This system program gives the user the capability to examine and update the library files on file-oriented devices. A more complete description of UPDATE-9, and details of its use, are given in Section 3.4.4.9.

CHAPTER 2
INPUT/OUTPUT MONITOR
(Paper Tape Systems)

2.1 INPUT/OUTPUT MONITOR FUNCTIONS

The I/O Monitor of the PDP-9 ADVANCED software system simplifies the programming of input and output functions and allows programs to operate, without modifications, in both the Keyboard Monitor environment and the basic paper-tape environment. It serves as an interface between the system and user programs and the external world of device hardware, drawing upon the routines and capabilities of the Input/Output Programming System (IOPS) to relieve the programmer of writing his own device and data handling subroutines. The I/O Monitor allows simultaneous operation of many I/O peripherals and overlapped computation.

The Input/Output Monitor is designed to take advantage of the automatic priority interrupt (API) if it is present on the system. Both the I/O skip chain for the program interrupt control (PIC) and the API channels are set up to handle all devices which have been requested by the user. All unused channels are tied to an error routine to detect spurious interrupts.

2.1.1 Data Transmission Paths

Each system or user program must internally set up line buffers for data transmission to or from the external environment (see Section 4.3 for maximum sizes of line buffers for I/O devices). Each line buffer of n words consists of a two-word header and n-2 words of data. The structure of a line buffer is shown in Figure 2-1 and the header words are detailed in Figure 2-2.

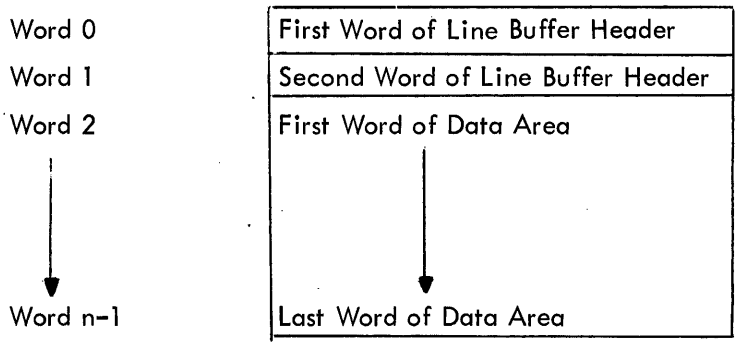


Figure 2-1 Line Buffer Structure

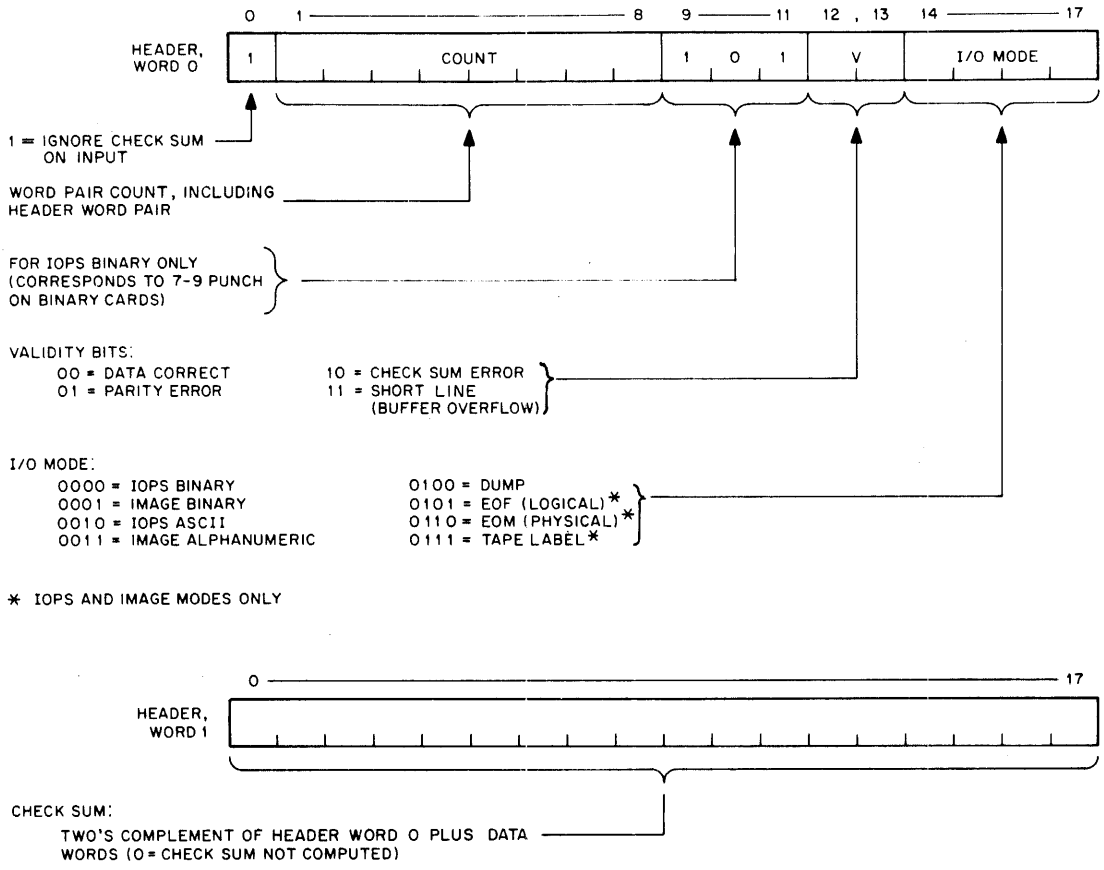


Figure 2-2 Format of Header Word Pair

For output, the program must set the appropriate word pair count in bits 1-8 of word 0 of the line buffer header. This count overrides the one passed on to IOPS by the .WRITE command to the Monitor. In IOPS binary mode, bits 9-11 should be set to 101 to indicate binary mode if the output device is a card punch. The checksum word (header word 1) may be left blank, since checksums are computed by IOPS.

For input, the user should check the validity bits (bits 12 and 13) of word 0 of the line buffer header to determine if the data was read without error. (IOPS sets the appropriate bits if it senses an error. If multiple errors are detected, priority will be given to parity error then to checksum error. It will ignore checksum errors on binary input if bit 0 of word 0 of the line buffer header, contained on the input medium, is set to 1.) IOPS will set the I/O mode bits (bits 14-17 of word 0 of the line buffer header) to 6 (0110_2) if it senses a physical end-of-medium (such as end-of-tape in the paper-tape reader).

The Monitor accepts commands from the system or user program to initiate input to the line buffers and to write out the contents of the line buffers. A full list of these commands is given in Section 2.2.

2.1.2 Data Modes

The Input and Output Programming System (IOPS) will allow data transmission to or from a PDP-9 ADVANCED software system program in one of five general forms:

		Code
a.	IOPS Binary Mode	0
b.	Image Binary Mode	1
c.	IOPS ASCII Mode	2
d.	Image Alphanumeric Mode	3
e.	Dump Mode	4

2.1.2.1 IOPS Modes - 7-bit ASCII is used throughout the IOPS system to accommodate the entire 128-character revised ASCII set. In IOPS ASCII, all alphanumeric data, whatever its original form on input (ASCII, Hollerith, etc.) or final form on output, is converted and stored as "5/7 ASCII."

5/7 ASCII refers to the internal packing and storage scheme for ASCII data. Five 7-bit ASCII characters are packed into two consecutive 18-bit binary words, as shown in Figure 2-3. They are stored as binary data on any bulk storage device.

It is recommended that all input requests in IOPS ASCII mode be made with an even word count to accommodate the paired input.

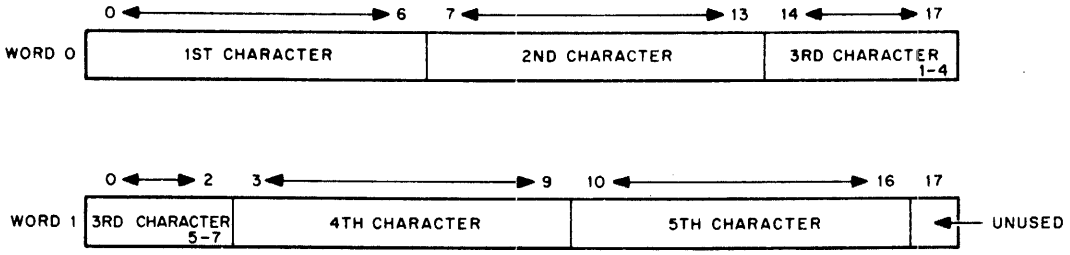


Figure 2-3 5/7 ASCII Packing Scheme

ASCII data is input to or output from IOPS, ordinarily, via the Teletype or paper tape, although it may exist in 5/7 ASCII form on any mass storage device. IOPS ASCII is normally defined as a 7-bit ASCII character with even parity in the eighth (high order) bit, in keeping with USA standards. IOPS performs a parity check on input of IOPS ASCII data prior to the 5/7 packing. On output, IOPS generates the correct parity. (See Appendices I, II.)

Non-IOPS ASCII occurs in data originating at a Model 33, 35, or 37 Teletype, without the parity option. This data always appears with the eighth (high order) bit set to 1. Apart from parity checking, the IOPS routines handle IOPS ASCII and non-IOPS ASCII data identically.

An alphanumeric line consists of an initial form control character, the body of the line, and a carriage return (CR) or ALT mode. CR (or ALT mode) is a required line terminator in IOPS ASCII mode. Control character scanning is performed for editing or control purposes (see Section 4.6 for affects of control characters on specific devices).

IOPS Binary data is blocked in an even number of words, with each block preceded by a two-word header. On paper tape, IOPS binary utilizes six bits per frame, with the eighth channel always set to 1, and the seventh channel containing the parity bit (odd parity) for the other seven channels. The parity feature supplements the checksumming as a data validity provision in paper tape IOPS binary.

Mixed data (both 5/7 ASCII and IOPS Binary) can be accepted as input (generated as output) by adjacent IOPS ASCII and IOPS binary ·READ's (·WRITE's.)

2.1.2.2 Dump Mode - Dump Mode data is always binary and is the mode used to output from or load directly into any core memory area, bypassing use of line buffers. Each dump mode statement has arguments defining the core memory area to be dumped. Although dump mode is normally used with bulk storage devices, it is also possible to use it with paper tape output and input.

2.1.2.3 Image Modes - Image Mode data is read, written, and stored in the binary or alphanumeric form of the source or terminal device, one character per word. No conversion, checking, or packing is permitted, and character scanning is generally omitted.

2.1.3 Device Assignment Tables

Both FORTRAN and MACRO-9 coded user programs, as well as the system programs, specify their input and output operations in terms of I/O commands to logical I/O units. One of the Monitor's functions is relating logical units to actual hardware devices. To do so, the I/O Monitor contains a device assignment table (.DAT) whose slot numbers correspond to logical I/O units.

Each .DAT slot referenced contains the device unit number (if applicable) and a pointer to the device handler for this device (see Section 2.3 and Figure 2.4).

All communications between programs, their line buffers and the actual device handlers are accomplished via .DAT slot numbers.

2.2 USER PROGRAM COMMANDS

Requests to the I/O Monitor for input or output functions are initiated by the system or user programs. In FORTRAN IV, these requests are in the form of READ (u); WRITE (u); READ (u,f); and WRITE (u,f) statements which are translated by the compiler into the proper calling sequences for the FORTRAN Object-Time System. The FORTRAN Object-Time System, in turn, executes the required Monitor calls. In MACRO-9, these requests are in the form of I/O macros which are written as part of the program and which expand into the proper Monitor calling sequences.

User commands available in MACRO-9 programs include:

Name	Purpose
.INIT	Initializes the device and device handler.
.READ	Transfers data from the device to the line buffer.
.WRITE	Transfers data from the line buffer to the device.
.WAIT	Detects availability of the user's line buffer.
.CLOSE	Terminates use of a file.
.TIMER	Calls and utilizes real-time clock.
.EXIT	Returns control to the Monitor.
.SEEK	Finds file on file-oriented device and begins data input.
.ENTER	Primes file-oriented device for file output.
.CLEAR	Initializes file structure on file-oriented device.
.MTAPE	Provides special commands for IBM-compatible magnetic tape.
.TRAN	Reads or records user-specified blocks on bulk storage devices, providing the user with the capability to determine the structure of the files on the device.
.DLETE	Deletes file from file-oriented device.
.RENAM	Renames file on file-oriented device.
.FSTAT	Checks presence of file on file-oriented device.

A complete discussion of these macros, and their expansions, is given in Sections 2.2.1 and 2.2.2.

2.2.1 General Commands

The following commands can be used on all devices, whether file-oriented or not.

2.2.1.1 .INIT (Initialize)

FORM: .INIT␣a, F, R

VARIABLES: a = Device Assignment Table (.DAT) slot number (in octal radix)

F = File Type: { 0 = Input File; 1 = Output File }

R = User Restart Address*

EXPANSION:

LOC	CAL + F_{7-8} + a_{9-17}	
LOC + 1	.1	/The CAL handler will place the unit /number (if applicable) associated with /.DAT slot a into bits 0-2 of this word.**
LOC + 2	R	
LOC + 3	n	/Maximum size of line buffer associated /with .DAT slot a; i.e., 225_{10} for DEC- /tape (see Section 4.2)***

DESCRIPTION: The macro .INIT causes the device and device handler associated with .DAT slot a to be initialized. .INIT must be given prior to any I/O commands referencing .DAT slot a; a separate .INIT command must be given for each .DAT slot referenced by the program.**** Since a .DAT slot may refer to only one type of file (input or output), only one file type specification (0 or 1) may be made in an .INIT statement. If a .DAT slot first references an input file, then an output file (or vice versa), a second .INIT command must be executed to change the transfer direction prior to the actual data transfer command.

---

*Has meaning only for .INIT commands referencing slots used by Teletype (the last .INIT command encountered for any slot referencing the keyboard or teleprinter takes precedence). When the user types ↑P, control is transferred to R. For example, the Linking Loader takes advantage of this feature to restart the system when a new medium has been placed in the input device.

**Has no direct effect upon the user's program, but should be noted so that no attempt will be made to use LOC + 1 as a constant.

***Size is returned by the handler so that the program, in a device-independent environment, can use it to properly set up line buffers.

****Each initialized .DAT slot constitutes an open file to the device handler and must be .CLOSEd (Section 2.2.1.5).

2.2.1.2 .READ

FORM: .READ␣a, M, L, W

VARIABLES: a = .DAT slot number (octal radix)

M = Data mode
- 0 = IOPS Binary
- 1 = Image Binary
- 2 = IOPS ASCII
- 3 = Image Alphanumeric
- 4 = Dump Mode

L = Line buffer address

W = Line buffer word count (decimal radix), including the two-word header

EXPANSION:

LOC	CAL + M_{6-8} + a_{9-17}	
LOC + 1	10	/CAL Handler will place unit number /(if applicable) into bits 0-2.
LOC + 2	L	
	.DEC	/Decimal radix
LOC + 3	-W	

DESCRIPTION: The .READ command is used to transfer the next line of data from the device assigned to .DAT slot a to the line buffer in the user's program. In this operation, M defines the mode of the data to be transferred (see Section 2.1.2 for a discussion of data modes); L is the address of the line buffer; and W is the count of the number of words in the line buffer (including the two-word header).

Since I/O operations and internal data transfers may proceed asynchronously with computation, a .WAIT command (see Section 2.2.1.4) must be used after a .READ command before the user attempts to use the data in the line buffer or to read another line into it.

When a .READ (non-dump mode) has been completed, the program should interrogate bits 12-13 of the first word of the line buffer header to ascertain that the line was read without error. Bits 14-17 should be checked for end-of-medium and end-of-file conditions.

2.2.1.3 .WRITE

FORM: .WRITE␣a, M, L, W

VARIABLES: a = .DAT slot number (octal radix)

M = Data Mode
- 0 = IOPS Binary
- 1 = Image Binary
- 2 = IOPS ASCII
- 3 = Image Alphanumeric
- 4 = Dump Mode

L = Line buffer address

W = Line buffer word count (decimal radix), including the two-word header

EXPANSION:

LOC	CAL + M_{6-8} + a_{9-17}	
LOC + 1	11	/The CAL Handler will place the unit /number (if applicable) associated with /.DAT slot a into bits 0-2
LOC + 2	L	
	.DEC	/Decimal radix
LOC + 3	-W	

DESCRIPTION: .WRITE is used to transfer a line of data from the user's line buffer to the device associated with .DAT slot a.

.WAIT must be used after a .WRITE command, before the line buffer is used again, to insure that the tranfer to the device has been completed.

Only in the IOPS binary mode are headers output with the data (bits 9 and 11 of header word 0 should be set to 1). In image modes the header space cannot be used for data, even though the headers are not written out. The word pair count in the header takes precedence in all modes and must be inserted by the user.

For both .READ and .WRITE macros, dump mode causes the transfer of the specified core area to or from one record on magnetic or paper tape. One or more blocks on DECtape, disc or drum may be occupied by a single dump command. A subsequent .WRITE in DUMP mode will utilize the unfilled portion of the last block.

2.2.1.4 .WAIT

FORM: .WAIT␣a

VARIABLES: a = .DAT slot number (octal radix)

EXPANSION:

LOC	CAL + a_{9-17}	
LOC + 1	12	/The CAL Handler will place the unit /number (if applicable) associated with /.DAT slot a into bits 0-2.

DESCRIPTION: .WAIT is used to detect the availability of the user's line buffer (being filled by .READ or emptied by .WRITE). If the line buffer is available, control is returned to the user immediately after the .WAIT macro expansion (LOC + 2). If the transfer has not been completed, control is returned to the .WAIT macro. .WAIT must also be used after the .TRAN command.

2.2.1.5 .CLOSE

FORM: .CLOSE␣a

VARIABLES: a = .DAT slot number (octal radix)

EXPANSION:

```
LOC       CAL + a9-17
LOC + 1   6          /The CAL Handler will place the unit
                     /(if applicable) associated with .DAT
                     /slot a into bits 0-2
```

DESCRIPTION: Whenever action has been initiated (.INIT or .SEEK or .ENTER) on a file (whether the device is file-oriented or not) this action must be terminated by a .CLOSE command.

On input, it is assumed that the user is finished with the file when the .CLOSE macro is used, so the file is closed. On output, all associated output is allowed to finish, and then an EOF line is output before the file is finally closed. If a refers to a file-oriented device, earlier files of the same name and extension, as currently referenced, will be deleted from its directory providing automatic storage retrieval.

2.2.1.6 .TIMER

FORM: .TIMER n, C

n = number of clock increments (decimal radix)

C = address of subroutine to handle interrupt at end of interval.

EXPANSION:

```
LOC       CAL
LOC + 1   14
LOC + 2   C
          .DEC       /Decimal radix
LOC + #   -n
```

DESCRIPTION: .TIMER is used to set the real-time clock to n and to start it. Each clock increment represents 1/60 sec for 60 cycle systems and 1/50 sec for 50 cycle systems.

C + 1 is the location to which control will be returned when the Monitor has finished servicing the clock interrupt. The coding at C should be in subroutine form; i.e.,

```
C         0              /C + 1 IS REACHED VIA JMS
          DAC SAVEAC
          :
          LAC C          /DO NOT DBR TO RESTORE LINK.
          RAL
          LAC SAVEAC
XIT       JMP*   C
```

so that control will return to the originally-interrupted sequence when the interval-handling routine has been completed. The Monitor automatically re-enables the interrupt system before transferring control to C + 1.

2.2.1.7 .EXIT

FORM:	.EXIT	
EXPANSION:	LOC	CAL
	LOC + 1	15

DESCRIPTION: .EXIT provides the standard method for returning to the Monitor after completion of a system or user program. In the I/O Monitor environment, it causes a program halt; in the Keyboard Monitor environment, it causes the Keyboard Monitor to be reloaded. When the reloading process has been completed, the Monitor types

```
MONITOR
$
```

on the teleprinter, indicating that it is ready to accept the next command.

2.2.2 Mass Storage Device Commands

The macros

```
.SEEK
.ENTER
.CLEAR
.TRAN
.DLETE
.RENAM
.FSTAT
```

normally apply to the file-oriented devices DECtape (DT), disc (DK), drum (DR), and magnetic tape (MT); they are ignored by non-file-oriented devices.

Another macro, .MTAPE, handles the none-file-oriented functions of magnetic tape (REWIND, BACKSPACE, etc). If these non-file-oriented commands are given to file-oriented devices, they are generally ignored by the device-handling routines.

Two of the .MTAPE commands (REWIND TO LOAD POINT, BACKSPACE RECORD), however, are bulk storage device independent if encountered prior to a .SEEK or .ENTER command for the referenced device.

Non-File Oriented DECtape

The terms file oriented and non-file oriented seem to evoke considerable confusion. A DECtape is said to be non-file oriented when it is treated as magnetic tape by issuing the MTAPE commands: REWIND, BACKSPACE, followed by READ or WRITE. No directory or identifying information of any kind is recorded on the tape. A block of data (255_{10} word maximum), exactly as presented by the user program, is transferred into the handler buffer and recorded at each WRITE command, where the final (256th) word is the "data link" to the next DECtape block of data. A CLOSE terminates recording with a simulated end-of-file consisting of two words: 1005, 776773. The data link of this EOF DECtape block is 777777. Note that the simulated end-of-file is identical whether executing a CLOSE in a file oriented or non-file oriented environment. (See Figure 2-2).

Because braking on DECtape allows for tape roll, staggered recording of blocks is employed in the PDP-9 ADVANCED Software System to avoid the constant turn-around or time consuming back and forth motion of sequential block recording. When recorded as a non-file oriented DECtape, block 0 is the first recorded in the forward direction. Thereafter, every fourth block is recorded until the end of the tape is reached, at which time recording, also staggered, begins in the reverse direction. Four passes over the tape are required to record 576_{10} blocks ($0\text{-}1077_8$).

Just as a REWIND or BACKSPACE command declare a DECtape to be non-file oriented, a SEEK or ENTER implies that a DECtape is to be considered file oriented. The term file oriented means simply that a directory exists on the DECtape to identify as to name and location the files which are recorded on this DECtape. A directory listing of any DECtape so recorded is available via the (L)ist command in PIP-9 or the (D)irect command in KM-9. A fresh directory may be recorded via the (N)ew-dir in KM-9 or Z switch in PIP.

The directory occupies the first 200_8 locations of DECtape block 100_8. It is divided into two sections: (1) a 40_8 word Directory Bit Map; and (2) a 140_8 word Directory Entry Section.

The Directory Bit Map defines block availability. One bit is allocated for each DECtape block (576_{10} bits = 32_{10} words). When set to 1, the bit indicates that the DECtape block is occupied and may not be used to record new information.

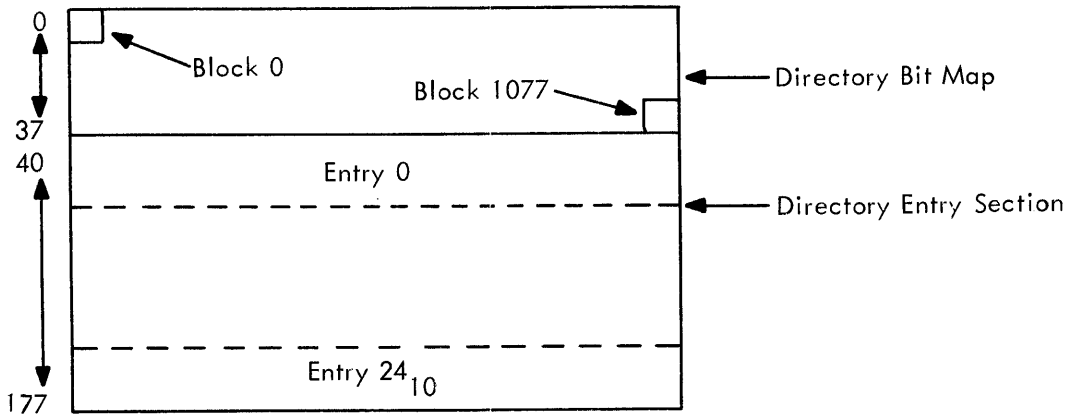
The Directory Entry Section provides for a maximum of 24_{10} files on a DECtape. A four word entry exists for each file on DECtape, where each entry includes the 6-bit trimmed ASCII file name (6 characters maximum), and file name extension (3 characters maximum), a pointer to the first DECtape block of the file, and a file active or present bit.

The second 200_8 words of DECtape block 100_8 contain basic directory information (blocks occupied by system programs), used by KM-9, PIP-9 and SGEN-9.

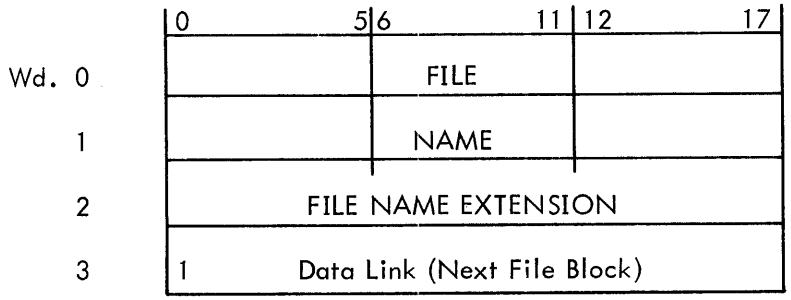
---

*Early versions of the PDP-9 ADVANCED Software System stagger recording on every fifth block.

DECTAPE DIRECTORY



A DIRECTORY ENTRY

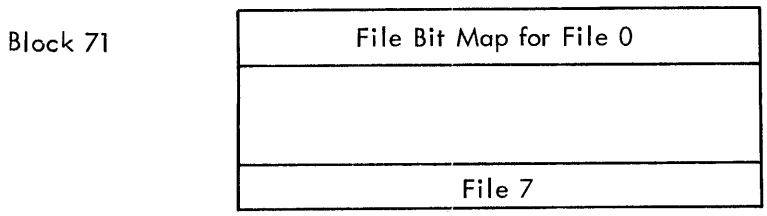


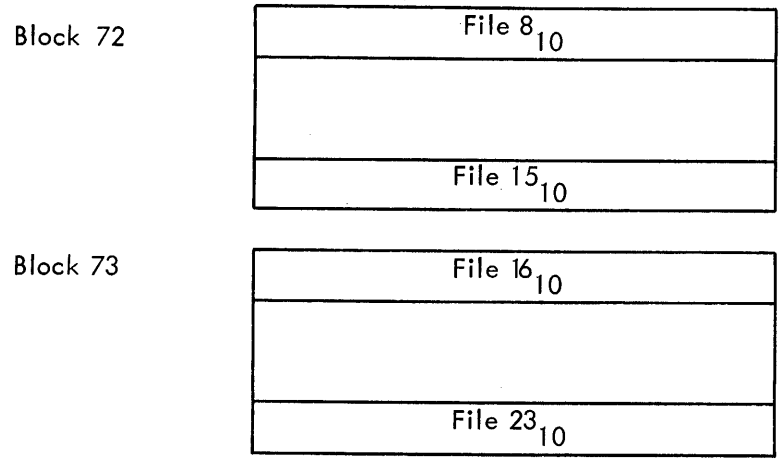
Sign Bit: 1 = File Active

Note: Nulls (0) fill in short file names. A file name extension is not required.

Additional file information is stored in blocks 71, 72 and 73 of every file oriented DECtape. These are the File Bit Map Blocks. For each file in the Directory a 32_{10} word File Bit Map is reserved in block 71, 72 or 73 as a function of file name position in the Directory Entry Section of block 100. Each block (71, 72, 73) is divided into eight File Bit Map Blocks. A File Bit Map specifies the blocks occupied by that particular file and provides a rapid, convenient method to perform DECtape storage retrieval for deleted or replaced files. Note that a file is never deleted until the new one of the same name is completely recorded, i.e., on the CLOSE of the new file.

FILE BIT MAP BLOCKS





When a fresh Directory is written on DECtape, blocks 100, 71, 72 and 73 are always indicated as occupied in the Directory Bit Map.

Staggered recording (at least every fourth block) is used on file oriented DECtapes, where the first block to be recorded is determined by examination of the Directory Bit Map for a free block. The first block is always recorded in the forward direction. Thereafter, free blocks are chosen which are at least four beyond the last one recorded. When turnaround is necessary, recording proceeds in the same manner in the opposite direction. When reading, turnaround is determined by examining the data link. If reading has thus far been in the forward direction, and the data link is smaller than the last block read, turnaround is required. If reverse, a block number greater than the last block read implies turnaround.

A simulated end-of-file terminates every file and consists of a two word header (1005, 776773) as the last line recorded. The data link of this final block is 777777.

Sections 2.1.1 and 2.1.2 of this manual discuss IOPS data modes. Data organization for each I/O medium is a function of these data modes. On file oriented DECtape there are two forms in which data is recorded: (1) packed lines - IOPS ASCII, IOPS binary, Image ASCII and Image binary; (2) dump mode data - Dump Mode.

In IOPS or image modes, each line (including header) is packed into the DECtape buffer. A 2s complement checksum is computed and stored for each line of information. When a line is encountered which will exceed the remaining buffer capacity, the buffer is output, after which the new line is placed in the empty buffer. No line may exceed 254_{10} words, including header, because of the data link and even word requirement of the header word pair count. An end-of-file is recorded on a CLOSE. It is packed in the same manner as any other line, i.e., if the buffer will contain it, fine. Otherwise, it goes into the next free block chosen.

In dump mode, the word count is always taken from the I/O macro. If a word count is specified, which is greater than 255_{10} (note that space for the data link must be allowed for again), the DECtape handler will transfer 255_{10} word increments into the DECtape buffer and from there to DECtape.

If some number of words less than 255_{10} remain as the final element of the dump mode WRITE, they will be stored in the DECtape buffer, which will then be filled on the next WRITE, or with an EOF if the next command is CLOSE. DECtape storage use is thus optimized in dump mode, since data is stored back to back without headers.

2.2.2.1 .SEEK

FORM: .SEEK␣a, D

VARIABLES: a = .DAT slot number (octal radix)

D = Address of user directory entry block

EXPANSION:

LOC	CAL + a_{9-17}	
LOC + 1	3	/The CAL Handler will place unit /number (if applicable) into bits 0-2
LOC + 2	D	

DESCRIPTION: .SEEK is used to search the directory of file-oriented device a for a desired file and to begin input for subsequent .READ commands. D is a pointer to (i.e., the address of) a three-word entry in the user's program containing the file name and extension information. The device's file directory block is searched for a matching entry and if found, input of the file into the handler's internal buffer begins. If no matching entry is found, control is transferred to an error-handling routine in the Monitor, an error message is printed on the teleprinter and the Monitor resumes control. Execution of the .FSTAT command (2.2.2.8) allows the user to check the directory for a named file and to retain control if it is not found.

The entry format in the user's file directory entry block (in core) is as follows:

	0 5	6 11	12 17
D	N	A	M
D+1	E	0	0
D+2	E	X	T

File Name: up to six 6-bit trimmed ASCII characters, padded, if necessary, with zero bits.

File Name Extension: Up to three 6-bit trimmed ASCII characters, padded with zero bits.

The file name is essentially nine characters (six of file name and three of file name extension); the file-searching of the .SEEK command takes into account all nine characters.

System programs use predetermined filename extensions in their operation. For example, if FORTRAN IV or MACRO-9 wishes to .SEEK program ABCDEF as source input, it searches for ABCDEF SRC (ABCDEF, Source). The binary output produced would be named ABCDEF BIN (ABCDEF, Relocatable Binary), while the listings produced would be named ABCDEF LST (ABCDEF, Listing). The Linking Loader, if told to load ABCDEF, would .SEEK ABCDEF BIN.

2.2.2.2 .ENTER

FORM: .ENTER␣a, D

VARIABLES: a = .DAT slot number (octal radix)

D = Address of user directory entry block

EXPANSION:

LOC	CAL + a_{9-17}	
LOC + 1	4	/The CAL Handler will place the unit /number (if applicable) associated with /.DAT slot a into bits 0-2.
LOC + 2	D	

DESCRIPTION: .ENTER is used to examine the directory of the device referenced by .DAT slot a, to find a free four-word directory entry block in which to place the three-word block at D and one word of retrieval information when .CLOSE is later issued. Deletion of earlier files with the same name and extension is performed by the .CLOSE macro. Control is transferred to the error handling routine in the Monitor to output an appropriate error message if there is no available space in the file directory at the time when .ENTER is executed.

2.2.2.3 .CLEAR

FORM: .CLEAR␣a

VARIABLES: a = .DAT slot number (octal radix)

EXPANSION:

LOC	CAL + a_{9-17}	
LOC + 1	5	/The CAL Handler will place the unit /number (if applicable) associated with /.DAT slot a into bits 0-2.

DESCRIPTION: .CLEAR is used to initiate the IOPS file structuring of the device referenced by .DAT slot a or to re-initiate its existing directory. The directory area and file bit map blocks on the file-structured device are set to 0.

In order to avoid clearing a directory when its files are still in use, the directory is checked for open files. If there are no open files, the directory is cleared; if not, control is transferred to the monitor error handling routine to output an appropriate error message.

2.2.2.4 .MTAPE

FORM: .MTAPE␣ a, XX

VARIABLES: a = .DAT slot number (octal radix)

XX = Number of a magnetic tape function

Rewind to load point = 00

Backspace record = 02

Backspace file = 03

Write end-of-file = 04

Skip record = 05

Skip file = 06

Skip to logical end-of-tape = 07

Describe tape configuration = 10 ⟶ 16

7-channel, even parity, 200 BPI = 10
7-channel, even parity, 556 BPI = 11
7-channel, even parity, 800 BPI = 12
9-channel, even parity, 800 BPI = 13
7-channel, odd parity, 200 BPI = 14
7-channel, odd parity, 556 BPI = 15
7-channel, odd parity, 800 BPI = 16
9-channel, odd parity, 800 BPI = 17

EXPANSION:

```
LOC       CAL + XX5-8 + a9-17
LOC + 1                        /The CAL Handler will place the unit
                               /number (if applicable) associated with
                               /.DAT slot a into bits 0-2.
```

DESCRIPTION: .MTAPE is used to perform functions unique to non-file-oriented bulk storage devices. In general, these functions are device dependent for non-file-oriented magnetic tape. However, two of the functions, REWIND TO LOAD POINT and BACKSPACE RECORD may be used with any bulk storage device that is to be employed in a non-file-oriented manner. For example, the DECtape Handler is directed to work in a file-oriented mode for a particular .DAT slot if it encounters a .SEEK or .ENTER as the next command after the .INIT command for that .DAT slot. If it encounters .MTAPE REWIND or BACKSPACE as the first command after .INIT, it sets up to work in non-file-oriented modes and interprets subsequent .READ and .WRITE commands appropriately. After the mode is established, commands in the other mode must not be executed.

2.2.2.5 .TRAN

FORM: .TRAN␣a, D, B, L, W

VARIABLES:

a = .DAT slot number (octal radix)

D = Transfer direction

Input Forward = 0

Output Forward = 1

*Input Reverse = 2

*Output Reverse = 3

B = Device address e.g., block number (octal radix) for DECtape

L = Core starting address

W = Word count (decimal radix)

*DECtape only

EXPANSION:

LOC	CAL + D_{7-8} + a_{9-17}	
LOC + 1	13	/The CAL Handler will place the unit /number (if applicable) associated with /.DAT slot a into bits 0-2.
LOC + 2	B	
LOC + 3	L	
	.DEC	/Decimal radix
LOC + 4	—W	

DESCRIPTION: .TRAN is employed when the user desires total freedom in data structuring of random access bulk storage devices. It provides the facility to read or record user specified areas on the device. .TRAN should be followed by a .WAIT macro to ensure that the transfer has been completed.

2.2.2.6 .DLETE

FORM: .DLETE␣a, D

VARIABLES: a = .DAT slot number (octal radix)
D = Starting address of three-word block of storage in user area containing the file name and extension of the file to be deleted from the device associated with .DAT slot a.

EXPANSION:

LOC	CAL + 1000 + a_{9-17}	
LOC + 1	2	/The CAL Handler will place the unit /number associated with .DAT slot a
LOC + 2	D	/into bits 0-2 of LOC + 1.

DESCRIPTION: .DLETE deletes the file specified by the file entry block at D from the device associated with .DAT slot a and retrieves the storage blocks released by that file. The contents of the AC will be 0 on return if the specified file could not be found.

2.2.2.7 .RENAM

FORM: .RENAM␣a, D

VARIABLES: a = .DAT slot number (octal radix)
D = Starting address of two 3-word blocks of storage in user area containing the file names and extensions of the file to be renamed and the new name, respectively.

EXPANSION:

```
LOC       CAL + 2000 + a9-17
LOC + 1   2                    /The CAL handler will place the unit number
                               /associated with .DAT slot a into bits 0-2
LOC + 2   D                    /of LOC + 1
```

DESCRIPTION: .RENAM renames the file specified by the file entry block at D with the name in the file entry block at D + 3 on the device associated with .DAT slot a. The contents of the AC will be zero on return, if the file specified at D could not be found.

2.2.2.8 .FSTAT

FORM: .FSTAT␣ a, D

VARIABLES: a = .DAT slot number (octal radix)

D = Starting address of three-word block of storage in user area containing the file name and extension of the file whose presence on the device associated with .DAT slot a is to be examined.

EXPANSION:

```
LOC       CAL + 3000 + a9-17
LOC + 1   2                    /The CAL handler will place the unit number
                               /associated with .DAT slot a into bits 0-2
LOC + 2   D*                   /of LOC + 1
```

DESCRIPTION: .FSAT checks the status of the file specified by the file entry block at D on the device associated with .DAT slot a. On return, the AC will contain the first block number of the file if found. The contents of the AC will be zero on return, if the specified file is not on the device. It is recommended that .FSTAT be used prior to .SEEK, if the user prefers to retain program control when a file is not found in the Directory. Otherwise, control is automatically returned to the Monitor.

2.3 DEVICE ASSIGNMENTS

The device assignment table used by the I/O Monitor is fixed both in length and in the assignments it contains. It is composed of two sections: the upper section is for use by system programs; the lower section is referenced by all user programs.

The upper portion of the .DAT contains 13 slots, referenced as -1 through -15_8. The lower section has 8 slots numbered 1 through 10_8. The fixed assignments for the device assignment table are shown in Figure 2-4.

---

*Bits 0-2 of LOC + 2 must be set to zero prior to the execution of the CAL at LOC. On return, bits 0-2 of LOC + 2 will contain a code indicating the type of device associated with .DAT slot a.

0 = Non-file-oriented devices

1 = DECtape

(2-7 To be specified)

These assignments will be changed by Digital as additional peripherals are added to a PDP-9 system. For example, .DAT slot 3 might be associated with a card reader, while .DAT slot 4 could be assigned to a line printer.

2.4 PROGRAMMING WITH USER PROGRAM COMMANDS

In the FORTRAN IV system, I/O commands, READ (u), WRITE (u), READ (u, f), WRITE (u, f), automatically generate calls to the FORTRAN IV Object-Time System, which performs data level I/O via .DAT slot u. In MACRO-9 programs, the user program commands (discussed above) must be used explicitly.

	.DAT SLOT	DEVICE	HANDLER*	USE
.DATBG	-15	Paper Tape Punch	(PPA.)	Editor and Converter Output
	-14	Paper Tape Reader	(PRA.)	Editor and Converter Input
	-13	Paper Tape Punch	(PPB.)	MACRO-9, FORTRAN IV Output
	-12	TTY Printer	(TTA.)	MACRO-9, FORTRAN IV and Converter Listing
	-11	Paper Tape Reader	(PRB.)	MACRO-9, FORTRAN IV Input
	-10	Paper Tape Reader	(PRA.)	DDT-9 Input and Editor, MACRO-9 Secondary Input
	-7	0		Not Used
	-6	Paper Tape Punch	(PPA.)	DDT-9 Output
	-5	0		Not Used
	-4	Paper Tape Reader	(PRA.)	System Input (Linking Loader)
	-3	TTY Printer	(TTA.)	Teleprinter Output
	-2	TTY Keyboard	(TTA.)	Keyboard Input
	-1	Paper Tape Reader	(PRA.)	System Device (Linking Loader)
	.DAT	.DAT		
	1	TTY Printer	(TTA.)	Teleprinter Output
	2	TTY Keyboard	(TTA.)	Keyboard Input
	3	Paper Tape Reader	(PRA.)	Input
	4	TTY Printer	(TTA.)	Listing
	5	Paper Tape Punch	(PPA.)	Output
	6	Paper Tape Reader	(PRA.)	Scratch

Figure 2-4 Device Assignment Table for I/O Monitor

*See Section 4.6 for a description of the handlers.

.DAT SLOT	DEVICE	HANDLER*	USE
7	Paper Tape Punch	(PPA.)	Scratch
10	Paper Tape Reader	(PRA.)	Scratch

.DATND = .

Figure 2-4 Device Assignment Table for I/O Monitor (cont)

2.4.1 Defining Line Buffers

Each device (.DAT slot) to be utilized in a user program should have at least one line buffer associated with it. This buffer(s) is used to set up output lines, before their transmittal to the output device, or to receive input lines from the associated input device. Line buffers are internal to the user's program and must be defined and tagged within it. The .BLOCK pseudo-op can be used to reserve space for a line buffer; the tag is necessary to allow referencing by .READ or .WRITE macros. Section 4.3 shows the maximum sizes for line buffers.

EXAMPLES (all examples assume decimal radix):

```
LINEIN      .BLOCK 52      /creates 52-word line buffer named LINEIN
LINOUT      .BLOCK 52      /creates 52-word line buffer named LINOUT
```

2.4.2 Specifying Devices Used to the Linking Loader

Whenever a MACRO-9 program wishes to utilize the user program commands discussed above, a pseudo-op (.IODEV) must appear somewhere in the program to specify to the Linking Loader which .DAT slots will be used. The appearance of this pseudo-op causes a code to be generated; the Linking Loader recognizes this code and employs it in loading the appropriate device handling routines. (FORTRAN IV programs cause the compiler to generate this code based on the units specified in the READ and WRITE statements.

EXAMPLE:

```
.IODEV 3, 5, 6
```

The macro assembly language program containing this statement can utilize .DAT slots 3, 5, and 6. The Linking Loader then calls for the device handlers specified by these slots; if a slot called by a user program is unassigned (has not been set up at system generation or ASSIGN time), an error message will result.

*See Section 4.6 for a description of the handlers.

2.4.3 Using Program Commands with MACRO-9

In addition to setting up line buffers within its own area and informing the Loader of the .DAT slots it will use, a MACRO-9 program must also initialize its devices and device handlers via .INIT commands and call for input or output via .READ or .WRITE commands.

Consider the following example, where input will be read from the device specified by .DAT slot 3, and output will be to the device specified by .DAT slot 5. It is assumed that the devices are not file-oriented, and only the I/O and related commands are shown. Note that bits 1-8 of LINOUT must be set to the word pair count (i.e., 26_{10}) before the .WRITE at OUT is executed. Bits 9 and 11 of LINOUT should be set to 1 to indicate binary mode.

EXAMPLE:

```
                    .
                    .
                    .
                .IODEV 3,
                SOURCE = 3
                OBJECT = 5
LINEIN          .BLOCK 52               /SET UP INPUT LINE BUFFER
LINOUT          .BLOCK 52               /SET UP OUTPUT LINE BUFFER
                .INIT SOURCE, 0, R      /INITIALIZE INPUT (R=RESTART ADDRESS)
                .INIT OBJECT, 1, R      /INITIALIZE OUTPUT (R=RESTART ADDRESS)
                    .
                    .
                    .
IN              .READ SOURCE, 2, LINEIN, 52/READ STATEMENT IN IOPS ASCII
                    .
                    .
                    .
                .WAIT SOURCE
                    .
                    .
                    .
OUT             .WRITE OBJECT, 0, LINOUT,52/WRITE STATEMENT IN IOPS BINARY
                    .
                    .
                    .
                .WAIT OBJECT
                    .
                .CLOSE OBJECT           /TERMINATE OUTPUT
                .CLOSE SOURCE           /TERMINATE INPUT
                    .
                .EXIT
```

Notice that the input mode is specified as IOPS ASCII (2), while output is IOPS Binary (0). The user restart address, R, specified in the .INIT commands, is ignored unless the device is the Teletype keyboard or printer.

It is important, also, to note that double or triple buffering is left to the discretion of the user. The following example shows a means of double buffering an input device.

EXAMPLE:

```
/THIS PROGRAM READS IN IOPS BINARY MODE
/THE INPUT DEVICE ASSOCIATED WITH .DAT
/SLOT 3 UNTIL THE END OF MEDIUM
/IS SENSED.
                .IODEV 3
START           .INIT 3, 0, R           /INITIALIZE INPUT DEVICE.
                LAC  LNBUF              /SET UP BUFFER POINTER
                DAC  LNBUFP             /TO POINT TO BUFFER #1.
                LAC  LNBUF2             /SET UP SECOND .READ
                DAC  READ 2+2           /TO READ IN TO BUFFER #2.
                .READ 3,0,L1BUFF,52     /READ INTO BUFFER #1.
WAIT            .WAIT 3                 /.WAIT ON READ
                LAC* LNBUFP             /CHECK FOR END OF
                AND  (17                /MEDIUM
                SAD  (6
                JMP  EOM                /END OF MEDIUM
                LAC* LNBUFP             /CHECK FOR CHECKSUM,
                AND  (60                /PARITY, SHORT BUFFER
                SZA                     /ERRORS.
                JMP  ERROR              /VALIDITY ERROR.
READ 2          .READ 3,0,X,  52        /READ INTO BUFFER X.
                /THIS SPACE
                /RESERVED TO
                /PROCESS LINE BUFFER
                /ALREADY IN.
                LAC  READ2+2            /SET UP TO PROCESS
                DAC  LNBUFP             /BUFFER X.
                SAD  LNBUF              /SWITCH .READ AT
                JMP  DBLRD1             /READ 2 TO INPUT
                LAC  LNBUF              /INTO OTHER BUFFER
DBLRD2          DAC  READ2+2            /(NOT X).
                JMP  WAIT               /GO TO .WAIT
DBLRD1          LAC  LNBUF2
                JMP  DBLRD2
EOM             .CLOSE 3                /CLEAN UP ON
                .EXIT                   /END OF MEDIUM.

LNBUF           L1BUFF                  /ADDRESS OF LINE BUFFER #1.
L1BUFF          .BLOCK  52
LNBUF2          L2BUFF                  /ADDRESS OF LINE BUFFER #2.
L2BUFF          .BLOCK  52
LNBUFP          0                       /BUFFER PROCESSING POINTER.
```

If the input and output devices had been file-oriented, the .SEEK, and .ENTER commands would have been used.

EXAMPLE:

```
                .
                .
                .
                .IODEV 3, 5, 6
LINEIN          .BLOCK 52                   /INPUT LINE BUFFER
LINOUT          .BLOCK 52                   /OUTPUT LINE BUFFER
                .INIT 3, 0, R               /INITIALIZE INPUT
                .INIT 5, 1, R               /INITIALIZE OUTPUT
                .
                .
                .SEEK 3, FILEA              /LOCATE FILEA
                .
                .
IN              .READ 3, 2, LINEIN, 52      /READ FILEA INTO LINEIN.
                .
                .
                .WAIT 3
                .
                .
                .ENTER 5, FILEB             /LOCATE FILE
OUT             .WRITE 5, 0, LINOUT, 52     /WRITE
                .
                .
                .WAIT 5
                .
                .
                .CLOSE 3                    /CLOSE INPUT FILE
                .CLOSE 5                    /CLOSE OUTPUT FILE
                .EXIT
```

If the user had wished to dump a section of core onto (non-file-oriented) device 5,

```
                .
                .
                .
                .IODEV 5
                 COUNT1 =  1000
                .INIT 5, 3, R               /INIT FOR DUMP
                .
                .
                .WRITE 5, 4, DUMP, COUNT1 /DUMP COMMAND
                .
                .
                .WAIT 5
                .
                .
                .CLOSE 5
                .EXIT
```

In this example, the .WRITE command specified dump mode (4) and the start (DUMP) and size (COUNT1) of the area to be output. The result of this section of coding is the dumping on device 5 of 1000 words from DUMP to DUMP + 999.

2.5 LOADING PROGRAMS WITH THE I/O MONITOR

In the paper tape system, each system program, accompanied by the necessary I/O device handlers and an appropriate version of the I/O Monitor, resides on a separate paper tape in absolute format. The eight system tapes supplied are:

FORTRAN IV
MACRO-9
PIP-9
Editor (EDIT-9)
Linking Loader (LINK-9)
DDT-9 (without Patch File Capabilities)
DDT-9 (with Patch File Capabilities)
7-TO-9 CONVERTER (CONV-9)

See Figure 3-4 for Memory Maps of I/O Monitor System.

At the beginning of each tape is a Bootstrap Loader in hardware READIN mode. By setting the starting address of the Loader* on the console address switches, depressing I/O RESET, and then depressing the READIN switch, these system tapes may be loaded.

Since the tape also contain appropriate versions of the I/O Monitor and the necessary I/O device handlers, the system programs (FORTRAN IV, MACRO-9, Editor, CONV-9 and PIP-9) can be loaded, ready for operation, in a single step.

User programs, however, normally exist in relocatable form, as output from FORTRAN IV or MACRO-9; these tapes do not contain copies of the I/O Monitor. To load these programs, a copy of the Linking Loader or DDT-9, should be loaded first. The version of the I/O Monitor (including the device handlers) contained on the Linking Loader or DDT-9 tape may be used with user programs, and the Linking Loader or DDT-9 can be used to load the necessary device handlers as well as the object programs.

Once the system program (FORTRAN IV, MACRO-9, Editor, CONV-9, PIP-9 or Linking Loader/DDT-9) has been loaded and takes control, the individual operating procedures (Section 2.5.1 through 2.5.7) come into use.

2.5.1 FORTRAN IV Compiler

When FORTRAN IV is loaded and ready for operation, it prints the following statement on the teleprinter:

```
FORTRAN 4
>
```

*17720 for 8,192-word systems, 37720 for 16,384 words, 57720 for 24,576 words, and 77720 for 32,768 words.

This indicates that FORTRAN IV is ready to accept a command string (on the same line as the >) telling it what action to take. At this point, the user should set up the input device for pass 1 of the source tape (if the input device is the paper tape reader, the tape-feed control should be depressed momentarily after that tape is in the reader to clear the reader-out-of-tape flag).

The format expected by the FORTRAN IV command string processor is as follows:

Options	File Name	Terminator
O, S, L, B	FILEX,	or ALT mode

where

O = object listing
S = symbol map
L = source listing
B = binary

← . must be a legal FORTRAN name

↵ return control to monitor after compiling current program.

↵ indicates batch compilation; after compiling current program, types

FORTRAN 4
>

and waits for next command string.

Default Assumptions

NO object listing
NO symbol map
NO source listing
NO binary

The options desired may appear in any order, separated by commas and terminated by ←. If default conditions are wanted, ← is sufficient, with the sole output being compiler diagnostics on the teleprinter. The comma after the file name is absolutely essential. Rubouts may be used to delete unwanted characters prior to typing the command string terminator.

At the end of pass 1 (when the END statement is encountered for the first time), FORTRAN IV indicates:

END PASS 1

This allows the user to prepare his input device for pass 2 (if the input device is the paper tape reader, depressing the tape-feed control after loading the source tape will again clear the flag). The second pass is then initiated by typing:

↑P

If the input device is a form of bulk storage (DECtape, magnetic tape, drum, or disc) FORTRAN IV automatically starts pass 2 without operator intervention.

NOTE: At the end of pass 1 when FORTRAN IV or MACRO-9 is waiting for a ↑P to begin pass 2, two (2) ↑Ps will force control to pass 1. Something must be in the reader during the interval between the two ↑Ps.

2.5.2 Macro Assembler (MACRO-9)

When MACRO-9 is loaded and reay to accept the command string from the keyboard, it indicates readiness by typing on the teleprinter

```
MACRO
>
```

At this point the user should set up the input device for pass 1 (if the paper tape reader, momentarily depresses the tape feed control to clear reader-out-of-tape flag).

The format expected by the MACRO-9 command string processor is as follows:

Options	File Name	Terminator
S, L, B, P	← FILEX	↵ or ALT mode

↵ or ALT mode
- return control to monitor after assembling current program.

return to MACRO-9 after assembling current program, types
```
MACRO
>
```
and waits for next assembly command string.

where

B = binary
L = listing
S = symbol table (on listing device)
P = secondary input is to be used

Default Assumptions

NO binary
NO symbol table
No assembly listing - if no assembly listing requested, all errors will be listed on the teleprinter.
NO secondary input

The options may appear in any order, separated by commas and terminated by ←. If no options are wanted, ← is sufficient and the sole output will be assembly error messages on the teleprinter.

Rubouts may be used to delete unwanted characters, prior to typing the command string terminator.

If this is a multi-medium assembly (where the first n mediums are terminated with .EOT and the last is terminated with .END), MACRO-9 indicates the end of each segment by typing .EOT on the teleprinter. This allows the user to prepare for the next segment of his input (if paper tape reader, depress the tape-feed control) and then type ↑P.

At the end of pass 1 (.END encountered for the first time), MACRO-9 outputs PASS 1 COMPLETED on the teleprinter. This tells the user to prepare the input device for the beginning of pass 2 (if paper tape reader depress the tape-feed control) and then type ↑P. If the input device is bulk storage (DECtape, magnetic tape, disk, drum) and .EOT was not encountered during pass 1, MACRO-9 automatically starts processing pass 2 without receiving a ↑P from the Teletype keyboard.

A secondary input is allowed at assembly time. This must contain only direct assignment statements, and is specified by writing "P" (for parameter tape) in the command string.

Format of Secondary Input. Secondary input must consist of direct assignments only, and must terminate with →|.EOT). If the secondary input is assigned to the Teletype, the user must type an additional) (for double buffering purposes) or may terminate with ↑D. The parameter tape will be inputted in pass 1 only, and not in pass 2.

Secondary input is useful, for example, to enter octal codes of IOTs deleted from the Permanent Symbol Table, such as:

```
PSF=700201
RSB=700144
```

2.5.3 PIP-9 (Peripheral Interchange Program)

PIP-9 is a utility program in the PDP-9 ADVANCED Software System used to transfer data files from one standard peripheral storage device to another. During transfer, PIP-9 can update file descriptions, rename, delete, insert and combine files. Since automatic storage retrieval is a standard function of the Monitor I/O Handlers, no provision for it is necessary in PIP-9.

PIP-9 operates under control of the Monitor, using the Monitor I/O Handler routines. The user directs PIP-9 functions by typing Teletype commands.

When PIP-9 is loaded and ready for operation , it outputs the following on the teleprinter.

```
PIP
>
```

This indicates that PIP is ready to accept a keyboard command typed on the same line as the right angle bracket (>). At this point, the user should prepare the I/O devices required in the operation and then type the PIP command.

Successful completion and readiness for the next command is normally acknowledged by

```
>
```

unless there has been intermediate output ot the teleprinter by PIP. In the latter case,

```
PIP
>
```

is output once again for ease of late printout examination.

The general format of a PIP command string is as follows.

F DDU:FILE1;EX1, FILE2;EX2(S)←SDU:FILE3;EX3

terminated by a carriage return or ALT mode. Colon (:) and semicolon (;) are not required to delimit the device and file names, respectively. A space may be used instead of a ":" or ";". Spaces are meaningless elsewhere in the command string and may be used freely except within the body of the device or file names.

F is a function character

T = transfer file

L = list directory

D = delete file

C = copy

N = rename file

DDU is the destination device (and unit if applicable)

PR = paper tape reader

PP = paper tape punch

TT = teletype

LP = line printer

D Tu = DECtape

MTu = Magnetic tape

CD = card reader

DRu = drum

DKu = disk

FILEN;EXN are the file names and extensions involved in the operations (they are omitted if device is not file oriented)

S indicates the switch options

A = IOPS ASCII

B = IOPS binary

I = Image alphanumeric

H = Image binary

D = Dump

W = strip .EOT from IOPS ASCII input

or W = strip EOF from IOPS binary input

C = convert multiple spaces to tabs

S = create new system directory

Z = zero out directory

E = directs PIP to convert horizontal tabs to spaces for off-line listing on Model 33 Teletypes. It may be used only with IOPS ASCII (A) as the data mode.

Example: T PP←DT1 FILEA SRC (AE)↵
will punch FILEA on paper tape, with spaces instead of horizontal tabs.

G = directs PIP examine input parity, allowing the user to modify input lines in error from the teletype. It is legal with IOPS ASCII only. On the occurrence of an input parity error, PIP always displays the following message on the teletype.

INPUT PARITY ERROR

If the G switch has been omitted from the input string, control returns immediately to the Monitor at this point. However, with the G switch on, the line in error is also typed out and the user may take one of four possible actions:

a. The line may be deleted by typing

D↵

b. The line may be accepted as is by typing

↵

c. The line may be replaced by typing a new line to replace the one in error, followed by a carriage return.

d. PIP may be restarted by typing ↑P

↑P

Example: T DT1 FILEA SRC (AG)←PR↵

SDU is the source device (and unit if applicable) (same options as DDU)

Use of ALT mode to terminate a command string, forces PIP-9 to exit to the Monitor upon successful completion of the command. Use of carriage return as the terminator causes PIP-9 to wait for another command upon completion of the current one.

Correction Procedures = Procedures available for command string corrections are

a. The rubout (RO) key deletes the input character immediately preceding. A \ is echoed for each RO input.

b. ↑U deletes the entire input line. The user begins the line from the first input character.

When PIP-9 detects a command string error, the questionable command string is output up to but not including the offending character or element followed by "?", requiring correct completion by the user. If the user prefers to retype the command, a carriage return will in this instance signal PIP-9 to accept a new command from the beginning. The character RO and ↑U may not be used since the Teletype handler (which no longer has access to the erroneous command string) and not PIP-9 interprets and acts upon RO and ↑U.

A task may be aborted and PIP-9 restarted by typing in the ↑P character at any time. ↑P has a secondary use in PIP-9 which is to indicate loading of the next in a series of paper tapes. For example,

at the end of each of several paper tapes to be combined into one output file, PIP-9 will output to the teleprinter ↑P, informing the user that he may load the next paper tape and type ↑P for PIP-9 continuation.

Refer to the PDP-9 Utility Systems Manual (DEC-9A-GUAB-D) for detailed descriptions of all the functions available accompanied by pertinent examples.

2.5.4 Text Editor (EDIT-9)

When the Text Editor is loaded and ready to accept an input command string, it prints

```
EDITOR
>
```

on the Teletype. If the input device is the paper tape reader, the user should load the tape be be edited, then momentarily depress the tape-feed control button to clear the reader-out-of-tape flag. At this point the user may type

OPEN FILNME EXT ↵

or

↵

The first format (OPEN FILNME EXT ↵) is used when a current file, called FILNME with an extension EXT (SRC is assumed if the extension is omitted) is on the device associated with .DAT slot -14 and is to be modified. The editing process results in a new file, with a temporary name, being created on the device associated with .DAT slot -15.

At the end of the editing process, the temporary file in the device associated with .DAT slot -15 replaces FILNME EXT (and is renamed FILNME EXT) on the device associated with .DAT slot -14. Of course, this replacement only occurs when both devices (.DAT slot -14 and -15) are bulk storage. Otherwise the final output is on .DAT slot -15.

The second format (CR) is used when a new file is to be created via Teletype keyboard input on the device associated with .DAT slot -15. If the devices associated with .DAT slots -14 and -15 are both bulk storage, at the end of the editing process, the temporary file on .DAT slot -15 will be written onto .DAT slot -14 with the file name and extension specified in the command string of the CLOSE command to the Editor.

Whenever the Editor is waiting for a command from the keyboard, it indicates this by outputting

```
>
```

on the teleprinter.

The CLOSE command to the Editor closes the current file and causes the Editor to cycle back to its initial printout

```
EDITOR
>
```

The EXIT command to the Editor causes a .EXIT return to the MONITOR.

2.5.5 Linking Loader (LINK-9)

When the Linking Loader is loaded into core and is ready to accept an input command string from the keyboard, it will type:

```
LOADER
>
```

on the Teletype. At this point, the input device should be set up. If it is the paper tape reader, the tape-feed control button should be depressed momentarily, after the paper tape is loaded, to clear the reader-out-of-tape flag.

The input command string to the Linking Loader should consist of the list of file names of all programs to be unconditionally loaded from the system input device (.DAT slot -4). The file names specified to the Linking Loader should agree with the names originally used in the FORTRAN IV compilation or MACRO-9 assembly of the programs; they should be typed on the Teletype keyboard in the following format

```
LOADER
>NAME 1, NAME 2, NAME 3 ↵
>NAME 4, NAME 5 (ALT MODE)
```

The main program must be requested first, followed by any desired subprograms. The file names should be one to six characters in length, with any characters over six being ignored. Only the file names should be specified; the Linking Loader will automatically assume that the file name extension is BIN (relocatable binary) and will search on both file name and extension.

A file name, in the input command string, is terminated by a comma (,), a carriage return (↵), or an ALT MODE. Until one of these three characters is encountered, n rubouts may be used to delete the previous n characters of the file name. ALT MODE terminates the input command string, and carriage return (↵) causes the LOADER to echo a line feed (↓), close angle bracket (>), so that the following format may also be used

```
>NAME 1↵ ↓
>NAME 2↵ ↓
   ⋮
>NAME (ALT MODE)
```

When the input device is not file-oriented, n commas, followed by the ALT mode character, prepares the Linking Loader to load n+1 programs from the system input device.

After loading the programs requested in the keyboard input command string, the Loader scans the system library (.DAT slot -1) and attempts to resolve all unsatisfied subroutine requests. The Loader prints a memory map on the Teletype during loading.

If the Loader detects an end-of-medium condition on either the system input device (.DAT slot -4) or the system library device (.DAT slot -1)*, it will type

↑P

on the Teletype. Additional input should be placed in the device (and the tape feed control button depressed momentarily to clear the reader-out-of-tape flag, if the device is the paper tape reader). To continue, the user should also type

↑P

on the keyboard.

After printing the memory map (Section 2.5.5.1), if all requested programs have been loaded and all library requests satisfied, the Loader will print

↑S

on the Teletype and sit in a program loop (JMP.). At this point, any changes in input and output devices should be made (including depressing the tape-feed control button on the paper tape reader, if it is to be used). When all I/O devices are properly set up, the user should type

↑S

on the keyboard. This transfers control to the starting address of the user's main program.

When the user's program has completed its operation and has terminated by means of an .EXIT command, the computer halts (I/O Monitor).

2.5.5.1 Linking Loader Memory Map - The memory map printed on the Teletype during loading consists of a list of names or programs loaded, a list of subroutines loaded or required, but not found, and their relocation factors (or starting load addresses). The format of the memory map is as follows (note that programs are loaded starting at the top of core).

```
NAME 1   16572
NAME 2   14301
NAME 3   10765
NAME 4   06427
NAME 5   06313
LIBR1    05304
LIBR2    04112
```

2.5.5.2 Error Messages - If an error occurs during loading, the Linking Loader types

.LOAD n

on the Teletype and halts. The type of error in indicated by n, as shown in the following table.

*The system library can consist of n paper tapes as long as only the last one is terminated by an end-of-file unit.

Error Code	Meaning
1	Memory overflow - the Loader's symbol table and the user's program have overlapped. At this point the Loader memory map will show the addresses of all programs loaded successfully before the overflow. Increased use of COMMON storage may allow the program to be loaded as COMMON can overlay the Loader and its symbol table, since it is not loaded into until run time.
2	Input data error - parity error, checksum error, illegal data code, or buffer overflow (input line bigger than Loader's buffer).
3	Unresolved Globals - Any programs or subroutines required but not found, whether called explicitly or implicitly, are indicated in the memory map with an address of 00000. If any of the entries in the memory map have a 00000 address, loading was not successful; the cause of trouble should be remedied and the procedure repeated.
4	Illegal .DAT slot request - The .DAT slot requested was: a. Out of range of legal .DAT slot numbers, b. Zero, or c. Unassigned, i.e., was not set up at System Generation Time or (in the case of the Keyboard Monitor) was not set up by an ASSIGN command.

2.5.5.3 File Structure of the System Library - The System Library (on .DAT slot -1) consists of one file named .LIBR. It is composed of program units, so ordered on the medium that only one pass through the library loads all requested subroutines, the subroutines that they might require, and all necessary I/O device handlers. The library file may consist of more than one paper tape with only the last tape terminated by the end-of-file.

A program unit consists of the following elements: a program size descriptor, internal GLOBALS, the loading address descriptor, the actual program, virtual GLOBALS, and an end code. It appears on the storage medium as one or more IOPS binary buffers.

An IOPS binary buffer consists of 50_{10} words. The first two words are a 2-word header; the other 48_{10} words are twelve (12) 4-word groups. Each 4-word group contains one (1) descriptor word and three (3) data words, so each 50-word IOPS binary buffer contains 36 data words.

The full structural break-down of the library file is shown in Figure 2-5.

2.5.6 DDT-9

DDT-9 is identical to the Linking Loader (Section 2.5.5) except that, on completion of loading and building of the DDT symbol table (exclusive of the symbol for the library routine and DDT), control is automatically transferred to the starting address of DDT-9.

DDT-9 types

```
DDT
>
```

to inform the user that it is waiting for a command.

The user can force control back to the starting address of DDT-9 at any time by typing:

```
↑T
```

on the Teletype keyboard. All previous DDT conditions are left intact; i.e. breakpoints, register modifications.

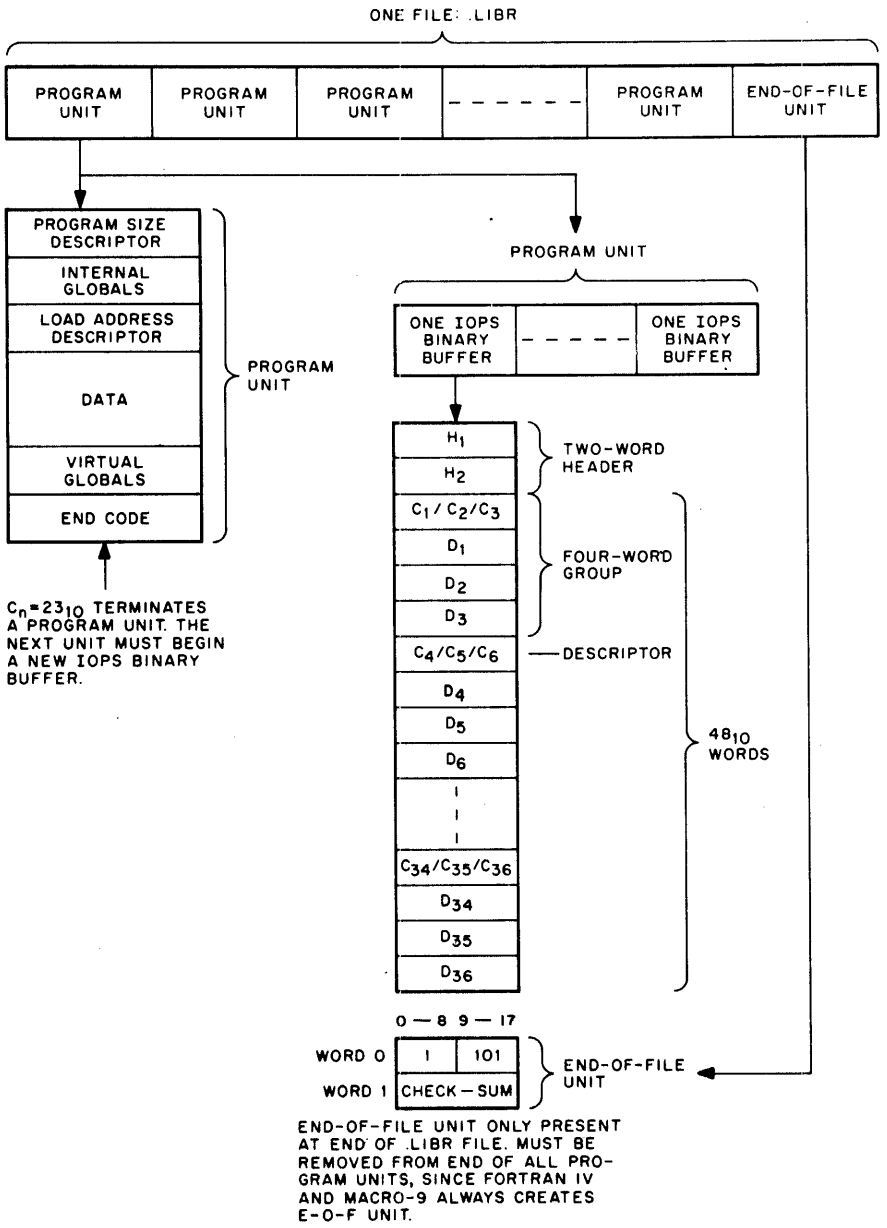


Figure 2-5 Library File Structure

2.5.7 7-to-9 Converter (CONV-9)

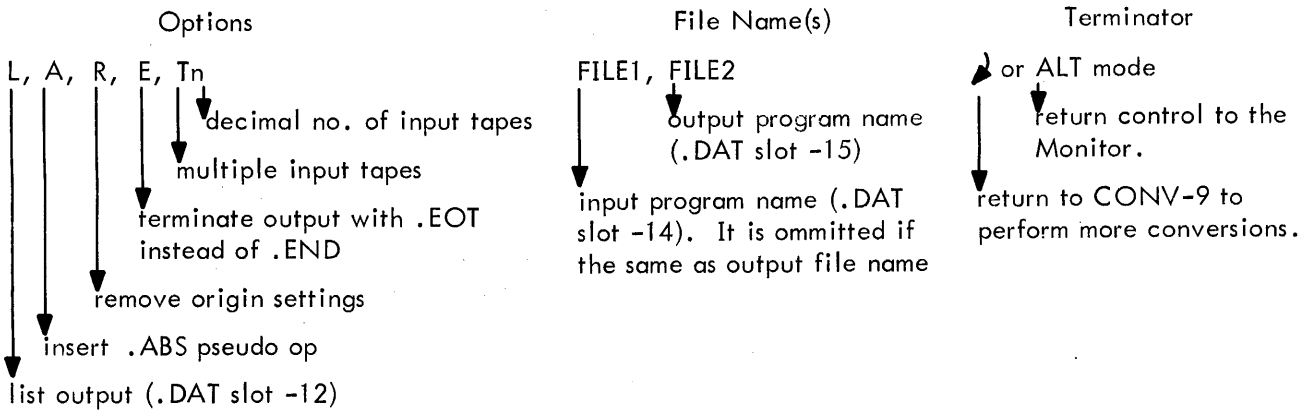
The 7-to-9 Converter converts source programs written for the PDP-7 or basic PDP-9 assemblers to a format acceptable to the ADVANCED Software PDP-9 MACRO-9 Assembler.

When CONV-9 is loaded and ready for operation, it prints the following on the teleprinter.

```
7-TO-9 CONVERTER
>
```

This indicates that CONV-9 is ready to accept a command string (on the same line as the >) telling it what action to take.

The command string format is:



Default Assumptions

NO listing
NO inserting of .ABS pseudo-op
NO removal of original settings
.END instead of .EOT
ONE input tape

The options desired may appear in any order, separated by commas and terminated by ◄—— . If no options are desired, ◄—— is sufficient. Rubouts may be used to delete unwanted characters prior to typing the command string terminator. If an error in the command string is detected, CONV-9 types

```
COMMAND STRING ERROR
>
```

and waits for a new command string.

Refer to PDP-9 Utility Programs Manual (DEC-9A-GUAB-D) for lists of items that cannot be correctly converted by CONV-9 and must be modified by the programmer himself.

2.6 ERROR DETECTION AND HANDLING

When either .IOPS or a device handler detect an illegal situation, .IOPS will output an appropriate error message on the teleprinter in one of the two following formats:

a. .IOPS NN XXXXXX

where NN is the appropriate error code and XXXXXX is additional related information (as shown below):

Error Code (octal radix)	Error	XXXXXX
00	Illegal function CAL	Address of CAL
01	Illegal CAL indirect	Address of CAL
02	Illegal .DAT slot request or no device assigned to .DAT slot	Address of CAL
03	Illegal I/O interrupt	IORS status word
05	Illegal .SETUP CAL (associated skip IOT not in the skip chain)	Address of CAL
06	Illegal function for device handler	Address of CAL
07	Illegal data mode for device handler	Address of CAL
10	Previous file still active when .CLEAR .SEEK .ENTER executed to same .DAT slot	Address of CAL
11	.SEEK (.ENTER) not executed prior to .READ (.WRITE) to file oriented device	Address of CAL
12	Irrecoverable data error (mark track error or EOT during .READ or .WRITE)	DECtape status register B
13	File not found (on .SEEK)	Address of CAL
14	Directory full (on .ENTER)	Address of CAL
15	DECtape full (on .ENTER or .WRITE)	Address of CAL
16	Output buffer overflow (excessive word pair count)	Address of CAL
17	Excessive number of files simultaneously referenced	Address of CAL
30	API software level error (channel registers 40-43 not setup prior to interrupt request at associated software level)	API status register

After outputting the message, the processor will halt.

b. IOPS 4 - indicates that I/O was requested on a "not ready" device. The user should make the device ready and type

↑R

on the Teletype keyboard to continue.

CHAPTER 3
KEYBOARD MONITOR SYSTEM

3.1 KEYBOARD MONITOR FUNCTIONS

The Keyboard Monitor is designed to operate on PDP-9 systems with some form of auxiliary bulk storage (see Hardware Requirements, Section 1.2). It includes all of the elements of the Input/Output Monitor plus routines to accept keyboard commands, change device assignments, and automatically load and initiate system and user programs. (The user should be completely familiar with Section 2: Input/Output Monitor System, before utilizing the Keyboard Monitor.)

3.1.1 System Device and Bootstrap

Each installation employing the Keyboard Monitor version of PDP-9 ADVANCED software must reserve tape unit 0 (DECtape or magnetic tape) or a lower section of the disc or drum as the system device. This unit contains the System Tape including the Monitor, the entire input/output programming system and all system and library programs needed by the user.

The System Bootstrap is supplied as a paper tape in hardware READIN format. By setting the starting load address of the bootstrap (17637 of the highest memory bank available) on the console address switches, depressing I/O RESET and then the READIN switch, the bootstrap is loaded into upper core. It clears the flags, turns on EXTEND MODE, disables the program interrupt (and the automatic priority interrupt, if available), loads the Keyboard Monitor into lower core, and transfers control to it.

3.1.2 Keyboard Monitor Structure

The Keyboard Monitor has five major sections: keyboard listener, monitor command decoder, the input/output programming system, a monitor error diagnostic program, and the necessary tables for communication between the programs and the I/O handlers. The first two of these five sections are non-resident, in the sense that they are overlaid by user and system programs, while the last three smaller sections are always resident in core.

An .EXIT from a user of system program transfers control to the System Bootstrap which then brings the non-resident sections of the Keyboard Monitor back into core.

3.1.2.1 Non-Resident Sections of the Keyboard Monitor - The keyboard listener (.KLIST) accepts user commands from the Teletype keyboard to process system information request commands, to change the device assignment table and to load and run system and user programs. The keyboard listener also handles Monitor initialization and some system bookkeeping.

The monitor command decoder (.MCD) acts upon the commands accepted by .KLIST; it responds to requests for system programs by loading the System Loader and by initiating loading of the desired package. It is also responsible for making dynamic changes to the device assignment table.

3.1.2.2 Resident Sections of the Keyboard Monitor - Includes the I/O control routine (.IOPS) and the device handlers and interrupt service routines for the Teletype keyboard and printer, and the internal clock.

The monitor error diagnostic (.MED) program utilizes the PDP-9 system teleprinter to output error messages to the user.

The device assignment table (.DAT) is created by the user with the System Generator at system generation time (see Sections 3.4.2 and 3.4.4.7). .DAT contains all of the standard I/O device assignments that the user normally employs for system, FORTRAN IV, and MACRO-9 programs. Key-board commands by the user may be used to vary .DAT slot assignments prior to the loading of system or user programs. This facility provides the user with true dynamic device independence.

Two other tables are contained in the resident portion of the Monitor and are used to control the I/O functions:

a. The system communication table (.SCOM) provides a list of registers which may be referenced by the Monitor, IOPS, and system programs. Included in .SCOM are the starting location addresses of system programs (.SCOM+5) and user programs (.SCOM+6) when loaded into core. Table 4 of the appendix contains a complete list of .SCOM entries.

b. The monitor/IOPS command table (.COMTB) contains pointers to the routines utilized by user programs (.READ, .WRITE, .SEEK, etc.). The structure of .COMTB is shown in Table 5 of the appendix.

3.2 KEYBOARD COMMANDS

The Keyboard Monitor provides three advantages over the Input/Output Monitor:

a. The ability to request system information and directions for system operation.

b. I/O device independence, through the ability to dynamically change I/O device assignments before loading a program.

c. The ability to call, load, and execute system and user programs via simple keyboard commands.

When the Keyboard Monitor initially gets control it outputs

```
MONITOR
$
```

to the teleprinter to indicate readiness to accept a keyboard command. Subsequently, it outputs "$" to indicate readiness. In both cases, the keyboard command should be typed on the same line as the

dollar sign ($).

Three different groups of commands are available to the user:

> NOTE: All numbers typed in Monitor Keyboard commands are interpreted in octal radix.

a. Device examination, assignment, and information commands

LOG	REQUEST
SCOM	ASSIGN
INSTRUCT	DIRECT
NEWDIR	

b. Commands to load system programs

LOAD	DDT
GLOAD	DDTNS (DDT without symbol table)
EDIT	PIP
MACRO	F4
UPDATE	SGEN
CONV	DUMP

c. Commands affecting program operation

↑S	↑Q
↑C	GET
↑T	↑P
↑R	SDUMP
	HALT

Most commands to the Keyboard Monitor are of a symbolic nature and are terminated by a carriage return (CR) or ALT mode (ESC). However, some special 1-character commands, such as ↑S (control S) are used when only the resident portion of the Monitor is available.

3.2.1 Device Examination, Assignment, and Information Commands

The first letter of the command may be used instead of the entire command.

3.2.1.1 LOG - The LOG command is used to make hard copy records of the user's comments on the Teletype. Upon encountering the LOG command, the Monitor enters LOG mode and ignores all typing up to and including the next ALT mode (ESC).

EXAMPLE:

```
$LOG THIS IS AN EXAMPLE OF A LOG COMMAND
     TO KM9
```

3.2.1.2 SCOM - The SCOM command prints out certain system information.

EXAMPLE:

```
$SCOM
SYSTEM COMMUNICATIONS
17645       1ST FREE LOCATION BELOW BOOTSTRAP
1447        1ST FREE LOCATION ABOVE RESIDENT MONITOR
CONFIGURATION HAS API
CONFIGURATION HAS EAE
DEVICE HANDLERS AVAILABLE:
    TTA     TELETYPE:  INPUT/OUTPUT, ASCII MODES, ALL FUNCTIONS
    PRA     PAPER TAPE READER:  INPUT, ALL MODES, ALL FUNCTIONS
    PRB     PAPER TAPE READER:  INPUT, IOPS ASCII MODE, ALL FUNCTIONS
    PPA     PAPER TAPE PUNCH:  OUTPUT, ALL MODES, ALL FUNCTIONS
    PPB     PAPER TAPE PUNCH:  OUTPUT, ALL MODES LESS IOPS ASCII, ALL FUNCTIONS
    PPC     PAPER TAPE PUNCH:  OUTPUT, IOPS BINARY MODE, ALL FUNCTIONS
    LPA     LINE PRINTER:  OUTPUT, IOPS ASCII MODE, ALL FUNCTIONS
    CDE     CARD READER:  CR01E
    CDB     CARD READER:  CR02B
    DTA     DEC TAPE:  3 FILES, INPUT/OUTPUT, ALL MODES, ALL FUNCTIONS
    DTB     DEC TAPE:  2 FILES, INPUT/OUTPUT, IOPS MODES, LIMITED FUNCTIONS
    DTC     DEC TAPE:  1 FILE, INPUT, IOPS MODES, LIMITED FUNCTIONS
    DTD     DEC TAPE:  1 FILE, INPUT/OUTPUT, ALL MODES, ALL FUNCTIONS
```

3.2.1.3 INSTRUCT - The INSTRUCT command prints out the basic operational instructions for the various system programs. The command takes the following form:

INSTRUCT XXXXXX

where XXXXXX is the system program name, null for Monitor operational instructions, or ERRORS for explanation of .IOPS error printouts.

EXAMPLES:

```
$INSTRUCT
MONITOR: INFORMATION AND MODIFICATION COMMANDS
         LOG:  USER COMMENTS TERMINATED BY ALTMODE
         SCOM:  SYSTEMS INFORMATION
         INSTRUCT OR INSTRUCT PRGNAM:  OPERATING INSTRUCTIONS
         REQUEST, REQUEST USER, OR REQUEST PRGNAM:  .DAT SLOT USAGE
         ASSIGN DEVN A, B, .../ETC.:  .DAT SLOT MODIFICATIONS
         DIRECT OR DIRECT N:  DIRECTORY LISTING OF UNIT OF SYSTEM DEVICE
         SDUMP:  SET TO SAVE CORE ON .IOPS ERROR
         HALT:  SET TO HALT ON .IOPS ERROR
         ↑Q:  SAVE CORE
         GET OR GET XXXXX:  RESTORE SAVED CORE
         ↑C:  RESTORE MONITOR
MONITOR: PROGRAM LOADING COMMANDS AND ALSO PRGNAM IN ABOVE
         LOAD:  LINK LOADER AND STOP
         GLOAD:  LINK LOADER AND GO
         DDT:  LINK LOADER AND GIVE CONTROL TO DDT
```

MACRO: MACRO9 ASSEMBLER
F4: FORTRAN IV COMPILER
EDIT: SYMBOLIC EDITOR
PIP: PERIPHERAL INTERCHANGE PROGRAM
SGEN: SYSTEM GENERATOR
DUMP: BULK STORAGE DEVICE DUMP
UPDATE: LIBRARY FILE UPDATE
CONV: 7-TO-9 CONVERTER

```
$INSTRUCT DUMP
DUMP:     LISTING DUMP OF BULK STORAGE DEVICE
          TERMINATE COMMAND STRING BY ALTMODE FOR EXIT
          ALL: DUMP OF ALL OF SAVED CORE
          XXX, YYY: DUMP OF SAVED CORE FROM XXX TO YYY
          N#: DUMP OF BLOCK N
```

3.2.1.4 REQUEST - The REQUEST command allows examination of the .DAT slots associated with various system programs.* The command takes the following form:

REQUEST XXXXXX

where XXXXXX is the system program name (i.e., the system program load command) or USER for all positive .DAT slots or blank for an entire .DAT table printout.

EXAMPLES:

$REQUEST

.DAT	DEVICE	UNIT	USE
-15	DTA	2	OUTPUT - EDITOR, UPDATE, SYSGEN, CONV
-14	DTA	1	INPUT - EDITOR, UPDATE, SYSGEN, DUMP, CONV
-13	PPC	0	OUTPUT - MACRO, FORTRAN
-12	TTA	0	LISTING - MACRO, FORTRAN, UPDATE, DUMP, CONV
-11	DTC	1	INPUT - MACRO, FORTRAN
-10	PRA	0	INPUT - DDT, EDITOR, UPDATE
-7	DTC	0	SYSTEM DEVICE - SYSTEM LOADER
-6	PPC	0	OUTPUT - DDT
-5	NONE	0	EXTERNAL LIBRARY - LINKING LOADER
-4	PRA	0	INPUT - LINKING LOADER
-3	TTA	0	TELEPRINTER OUTPUT - ALL SYS.PROGS.
-2	TTA	0	KEYBOARD INPUT - ALL SYS.PROGS.
-1	DTC	0	SYSTEM LIBRARY - LINKING LOADER.
1	NONE	0	USER
2	NONE	0	USER
3	NONE	0	USER
4	NONE	0	USER
5	NONE	0	USER
6	NONE	0	USER
7	NONE	0	USER
10	NONE	0	USER

*See Section 4.6.10 for .DAT slots used by system programs, their uses, and I/O handlers that are acceptable.

$REQUEST MACRO

.DAT	DEVICE	UNIT	USE
-13	PPC	0	OUTPUT
-12	TTA	0	LISTING
-11	DTC	1	INPUT
-10	TTA	0	SECONDARY INPUT
-3	TTA	0	CONTROL AND ERROR MESSAGES
-2	TTA	0	COMMAND STRING

3.2.1.5 ASSIGN - The ASSIGN command allows reassignment of .DAT slots to devices other than those set at system generation time. The change of assignment is only effective for the current job, since the permanent assignments are restored whenever control is returned to the Monitor. The command takes the following form:

ASSIGN DEVn a, b, etc/DEVm x, y, etc

where DEV is the device handler name (the list of legal handlers for a particular system may be requested via the SCOM command*).

n, m are unit numbers,

a, b, x, y, etc are .DAT slot numbers.

EXAMPLES:

$ASSIGN DTA4 -10, -6/PRA -5

$ASSIGN PPB -6/DTB2 3/DTB3 5

$ASSIGN DTA1 6, 7, 10

DEVn can be replaced by NONE to clear .DAT slots.

$ASSIGN NONE 4, 5, 10

.DAT slots -2 and -3 are permanent and may not be modified.

.DAT slot -7 may be modified only at system generation time.

3.2.1.6 DIRECT - The DIRECT command allows printout of the directory associated with any unit on the system device control (i.e., eight units on DECtape control).

The command takes the following form:

DIRECT N

where N is the unit number (unit 0 is the default assumption).

*See Section 4.6.10 for .DAT slots used by system programs, their uses, and I/O handlers that are acceptable. Many of the devices, DECtape for example, have more than one I/O handler associated with them. It is imperative that only one version of a device handler be present during a particular run as confusion occurs because of lack of communication between the two interrupt handlers.

EXAMPLE:

$DIRECT

FILE NAME	EXTENSION	START BLOCK
.LOAD	BIN	31
.LIBR	BIN	27
CREATE	BIN	32
DDT9	BIN	47
PPTEST	BIN	167
PPTEST	SRC	166
PPTEST	LST	171
.SGEN	BIN	234

556 FREE BLOCKS

3.2.1.7 N (NEWDIR) = The "N (NEWDIR) unit number" will clear the directory on the specified unit of the system device control (unit 0 illegal).

3.2.2 Loading System Programs

Loading commands instruct the Keyboard Monitor to bring in the System Loader which is used to load all system programs from the system device. The commands available to the user for loading systems programs via the Keyboard Monitor are:

Command	Program Called
LOAD	Linking Loader
GLOAD	Linking Loader and Go
EDIT	Symbolic Editor
MACRO	MACRO-9 Assembler
DDT	Debugging Program (DDT-9)
DDTNS	Debugging Program (DDT-9) without user symbol table
PIP	Peripheral Interchange Program (PIP-9)
F4	FORTRAN IV Compiler
SGEN	System Generator
DUMP	Dump Program for saved file
UPDATE	Update the library file of a file oriented device
CONV	7-to-9 Converter

All commands should be terminated by a carriage return (CR) or ALT mode (ESC). When the requested program has been loaded and is waiting for Keyboard input, an indication is given on the teleprinter with an appropriate message; such as

```
        LOADER
        >
or      FORTRAN 4
        >
or      EDITOR
        >
        etc.
```

3.2.3 Other Keyboard Commands

The following keyboard commands are also available to the operator:

3.2.3.1 ↑S (Control S)

FORM: ↑S (echoed by the Monitor)

DESCRIPTION: ↑S is used to start a program after the Linking Loader has brought it into core via a LOAD command.

3.2.3.2 ↑C (Control C)

FORM: ↑C (echoed by the Monitor)

DESCRIPTION: ↑C forces control back to the Keyboard Monitor which types

```
MONITOR
$
```

to indicate that it is waiting for a keyboard command. ↑C will be honored whenever typed. If the non-resident section of the Monitor was in core, the Monitor is not reinitialized; thus, previous conditions, such as .DAT slot assignments, are as they were prior to the ↑C. If the non-resident section of the Monitor was not in core, it is brought in and all conditions revert to the standard.

3.2.3.3 ↑T (Control T)

FORM: ↑T (echoed by the Monitor if DDT present; otherwise ignored)

DESCRIPTION: ↑T forces control back to DDT which types

```
DDT
>
```

to indicate its readiness for another DDT command.* ↑T will be honored whenever typed with DDT in memory. Otherwise it will be ignored.

---

*All previous DDT conditions are left intact; i.e., breakpoints, register modifications.

3.2.3.4 ↑P (Control P)

FORM: ↑P (echoed by the Monitor)

DESCRIPTION: ↑P may be used at any time to force control to the user's restart address (the address specified in the user's last .INIT to reference the Teletype). Action is unspecified if the user did not .INIT the Teletype.

3.2.3.5 ↑Q (Control Q)

FORM: ↑Q (echoed by the Monitor) N**

VARIABLE: N = number (0-7) of unit on system device control where file is to be saved.

DESCRIPTION: ↑Q dumps the current job, in core image, onto prespecified blocks of unit N on the system device control (the WRITE switch on this unit must be enabled). For example, when the system device is DECtape unit 0, ↑Q requests may be made only to DECtape. These core images may be retrieved and reloaded by GET commands (see Section 3.2.3.6) or examined by a DUMP command (see Section 3.4.4.8). ↑Q will be honored whenever typed.

3.2.3.6 GET (Retrieve File)

FORM: GET N or GET N XXXXX or GET N HALT

VARIABLES: XXXXX = program starting address

N = number (0-7) of unit on system device control that contains file to be retrieved.

DESCRIPTION: GET retrieves the core image (including the Monitor) stored on specified blocks of unit N on the system device control by ↑Q commands, and restores it to memory. Control is transferred to XXXXX, if specified; control halts if HALT was specified. To start, in this case, the starting address should be placed in the ADDRESS switches and the START button depressed (PIC and API are enabled). If neither XXXXXX of HALT are specified, the job will be restarted where it was terminated.

3.2.3.7 SDUMP (Set Dump)

FORM: SDUMP

DESCRIPTION: SDUMP is used to force automatic execution of the ↑Q command (on unit 0) on non-recoverable error calls to the monitor error diagnostic program (.MED). It must be issued prior to the LOAD, GLOAD, DDTNS or DDT command used to load the user program.** Note that the WRITE switch on the system device should be enabled in case of error; otherwise an .IOPS 4 (not ready) error will follow the initial error.

---

*Wait for echoing of ↑Q before typing the unit number N. No error checking is done on the unit number.
** SDUMP or HALT issued prior to a GET has no effect as the Monitor at ↑Q time overlays the Monitor primed by SDUMP or HALT.

3.2.3.8 ↑R (Continue After Readying I/O Device)

FORM: ↑R (echoed by the Monitor)

DESCRIPTION: When .IOPS detects a not ready condition on a called I/O device, it prints:

.IOPS 4

on the teleprinter. The user may ready the device and then continue by typing ↑R.

3.2.3.9 HALT (Halt after Error Message Printout)

FORM: HALT

DESCRIPTION: When the monitor detects an error condition and prints out the associated error message, it will halt. Depressing the CONTINUE button will reload the Keyboard Monitor. HALT must be issued prior to the LOAD, GLOAD, DDTNS or DDT command.*

3.3 PROGRAMMING FOR DEVICE INDEPENDENCE

When writing programs with the PDP-9 ADVANCED Software System, recognition should be made of its device-independent programming capabilities, even when there are no immediate plans to use them.

Several simple steps can be taken, when the programs are being coded, to insure that they will operate in a device-independent environment.

a. In .READ and .WRITE instructions, the line buffer size specified should be large enough to satisfy the requirements of any device that may be associated with a particular .DAT slot.

b. File manipulation commands should be included even when the originally contemplated use of the program is with hard copy devices. These commands will be ignored if not needed, and the program will not have to be rewritten for use with file-structured mass storage devices.

c. Only IOPS ASCII and IOPS binary data modes should be used, since image modes are device dependent.

3.4 OPERATING THE KEYBOARD MONITOR SYSTEM

PDP-9 ADVANCED software is a complete system for program preparation and use. It can generate its own system tape, change device assignments, call system or user programs, control their input/output functions, and report system errors to the user.

---

*SDUMP or HALT issued prior to a GET has no effect as the Monitor at ↑Q time overlays the Monitor primed by SDUMP or HALT.

3.4.1 Loading the Monitor

The System Bootstrap is provided as a self-starting paper tape in hardware READIN format and is loaded into upper memory.* It puts the system into EXTEND mode, turns off the program interrupt and API, loads the Keyboard Monitor from the system device and transfers control to it. The Monitor will type

```
MONITOR
$
```

when it is ready to accept commands from the user.

The System Bootstrap may be restarted without reloading the paper tape, if it has not been destroyed. Setting the ADDRESS switches to 17646 of the highest memory bank, depressing I/O RESET and then START will restart it.

3.4.2 System Generation

PDP-9 installations without DECtape or magnetic tape receive their PDP-9 ADVANCED software as a set of paper tapes which can be used to create a system tape.** Those installations having DECtape of magnetic tape systems will receive a system tape on DECtape or magnetic tape.

The System Generator (.SGEN) is a standard system program used to create new system tapes. Upon first receipt of a PDP-9 bulk-storage system, the user should immediately create a standard system tape for his installation. This is done by loading the System Bootstrap, which calls the system into core, and using the Keyboard Monitor to call the System Generator. .SGEN will output the new system tape on the device associated with .DAT slot -15; so, the ASSIGN command should be used prior to calling .SGEN to assign a bulk storage device to slot -15 and the old system device to slots -10 and -14, i.e,

```
$ASSIGN          DTA 2  -15          /DTA 0 -14, -10
$SGEN
```

Once loaded, .SGEN communicates with the user in a conversational mode via the Teletype to obtain the information needed to create a system tape. Among the items of information it needs to know are:

a. On which device the system tape will operate, so that
   1. The system device slots in the device assignment table (.DAT) can be set.
   2. A new System Bootstrap can be punched out for this device.
   3. The PIC skip chain and API channels can be set up for the system device.

b. All devices present in the PIC skip chain and their order. Non-basic devices can be added to the skip chain at this time, by supplying the device mnemonic and the skip IOT(s).

---

*Set the console address switches to 17637 of the highest memory bank available, depress I/O RESET and then the READIN switch.

**In this context, "tape" means any applicable bulk storage device, DECtape, disk, or drum.

c. Total core capacity (8, 16, 24, or 32K) of the installation.

d. Special options present at the installation, API, EAE, etc.

e. The structure of .DAT. All system slots (-1 to -15) and slots 1 to 10 should be assigned.

When .SGEN has received all of the information necessary, it creates a new system tape, then returns control to the Monitor.

New system tapes can be created whenever a significant change in the installation configuration occurs.

3.4.3 Assigning Devices

Before calling a system or user program, the operator should make all device assignments necessary to the program(s) to be run.

The ASSIGN command (see Section 3.2.1.5) is used to attach hardware devices to the slots of the device assignment table. Figure 3-1 shows the normal setup of .DAT. Only system slots -2, -3, and -7 cannot be modified by the ASSIGN command, since these must be used by the Monitor.

System programs use the negative .DAT slots while user programs should use the positive .DAT slots. PIP-9 (Peripheral Interchange Program) is an exception to this rule in that it uses all the positive .DAT slots (1 to 10).

	.DAT Slot	Device Handler*	Unit	Use
.DATBG	-15	DTA.	2	Output (EDITOR, UPDAT, CONVERTER, SYSGEN)
	-14	DTA.	1	Input (EDITOR, UPDAT, CONVERTER, SYSGEN, DUMP)
	-13	PPC.		Output (MACRO-9, FORTRAN IV)
	-12	TTA.		Listing (MACRO-9, FORTRAN IV, UPDATE, DUMP, CONVERTER)
	-11	DTC.	1	Input (MACRO-9, FORTRAN IV)
	-10	PRA.		Input (DDT) Secondary Input (EDITOR UPDATE, MACRO-9, SYSGEN)
	-7	DTC.	0	System Device (System Loader)
	-6	PPC.		Output (DDT)
	-5	NONE		External Library (Linking Loader)
	-4	PRA.		Input (Linking Loader) — All system programs
	-3	TTA.		Teleprinter Output — All system programs
	-2	TTA.		Keyboard Input — All system programs
	-1	DTC.	0	System Library (Linking Loader)

Figure 3-1 Functions of Slots in .DAT for Keyboard Monitor

*See Section 4.6 for a description of the handlers.

	.DAT Slot	Device Handler*	Unit	Use
.DAT		.DAT		
	1	NONE		
	2	NONE		
	3	NONE		
	4	NONE		User and PIP-9 .DAT slots
	5	NONE		
	6	NONE		
	7	NONE		
	10	NONE		
.DATND=.				

Figure 3-1 Function of Slots in .DAT for Keyboard Monitor (cont)

3.4.4 Loading Via Console Commands

The FORTRAN IV Compiler, Macro Assembler, Symbolic Editor, Peripheral Interchange Program, DUMP Program, Library Update Program, 7-to-9 Converter and System Generator are called by unique keyboard commands (F4, MACRO, etc.). All user programs are called by loading the Linking Loaded (via LOAD, GLOAD, DDTNS, or DDT commands) and asking it to load the desired program. (See Figure 3-3 for memory maps of programs loaded in the Keyboard Monitor System.)

Whenever a keyboard command requests a new program, the Keyboard Monitor loads the System Loader and the system device handler into core. The System Loader is basically the Linking Loader in absolute form and always requires the same device handler to acquire input from the system device. The Linking Loader, on the other hand, is relocatable and device independent.

The System Loader is used to bring in all system programs, including the Linking Loader, and their associated device handlers. Once loaded, the Linking Loader is used to bring in user programs, their subroutines, and their device handlers.

The System Loader can print the same error messages as the Linking Loader (see Section 2.5.5.2), except that it precedes the error code with the symbol .SYSLD. It returns control to the System Bootstrap to re-initialize the Keyboard Monitor if an error occurs.

Once a system program is loaded by the System Loader, the loaded program assumes control. At this stage, it is ready to accept an input command string from the keyboard telling it how to proceed. The specified procedures for each system program are shown in Sections 3.4.4.1 - 3.4.4.10; in most cases they are identical to procedures followed by the same program in the I/O Monitor environment. Note that all system programs assume that a file oriented device is file structured.

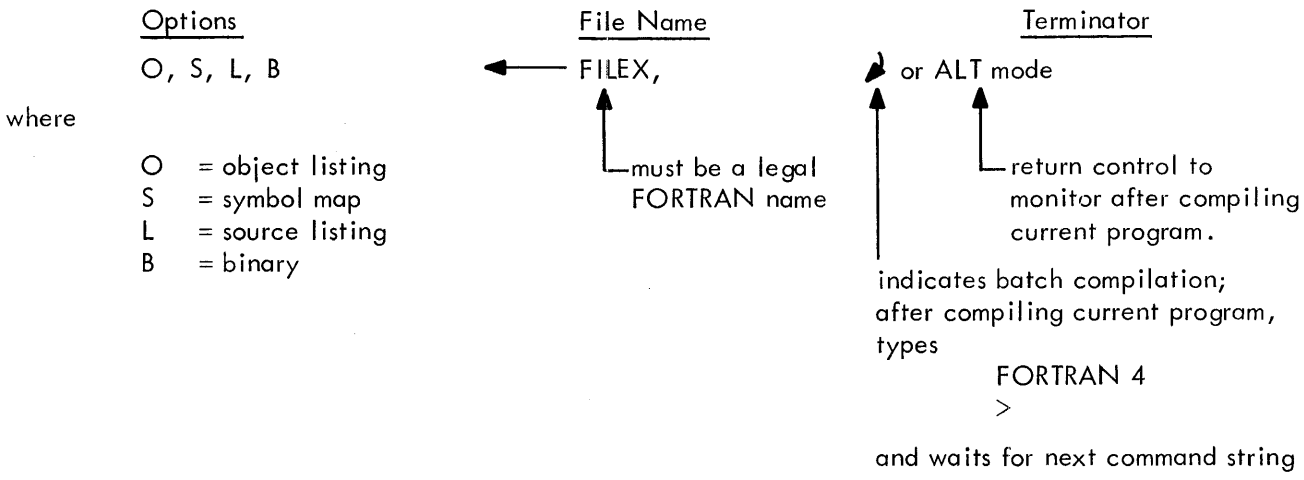
*See Section 4.6 for a description of the handlers.

3.4.4.1 FORTRAN IV Compiler - When FORTRAN IV is loaded and ready for operation, it prints the following on the teleprinter:

```
FORTRAN 4
>
```

This indicates that FORTRAN IV is ready to accept a command string telling it what action to take. At this point, the user should set up the input device for pass 1 of the sourec tape (if the input device is the paper tape reader, the tape-feed control should be depressed momentarily to clear the reader-out-of tape flag).

The format expected by the FORTRAN IV command string processor is as follows:



where

O = object listing
S = symbol map
L = source listing
B = binary

Default Assumptions

NO object listing
NO symbol map
NO source listing
NO binary

The options desired may appear in any order, separated by commas and terminated by ←. If default conditons are wanted, ← is sufficient. Rubouts may be used to delete unwanted characters prior to typing the command string terminator.

At the end of pass 1 (when the END statement is encountered for the first time), FORTRAN IV indicates

```
END PASS 1
```

This allows the user to prepare his input device for pass 2 (if the input device is the paper tape reader, depressing the tape-feed control after loading the source tape will again clear the flag). The second pass is then initiated by typing

```
↑P
```

If the input device is a form of bulk storage (DECtape, magnetic tape, drum, or disc) FORTRAN IV automatically starts pass 2 without operator intervention.

NOTE: At the end of pass 1 when FORTRAN IV or MACRO-9 is waiting for a ↑P to begin pass 2, two (2) ↑Ps will force control to pass 1. Something must be in the reader during the interval between the two ↑Ps.

3.4.4.2 Macro Assembler (MACRO-9) - When MACRO-9 is loaded and ready to accept the command string from the keyboard, it indicates readiness by typing on the teleprinter

```
MACRO
>
```

At this point the user should set up the input device for pass 1 (if the paper tape reader, momentarily depress the tape-feed control to clear reader-out-of-tape flag).

The format expected by the MACRO-9 command string processor is as follows:

Options	File Name	Terminator
S, L, B, P	← FILEX	or ALT mode → return control to monitor after assembling current program.
		return to MACRO-9 after assembling current program, types MACRO > and waits for next assembly command string.

where

B = binary
L = listing
S = symbol table (on listing device)
P = secondary input is to be used.

Default Assumptions

NO binary
NO symbol table
NO assembly listing - if no assembly listing requested, all errors will be listed on the teleprinter.
NO secondary input

The options may appear in any order, separated by commas and termined by ←. If no options are wanted, ← is sufficient.

Rubouts may be used to delete unwanted characters, prior to typing the command string terminator.

If this is a multi-medium assembly (where the first n media are terminated with .EOT and the last is terminated with .END), MACRO-9 indicates the end of each segment by typing .EOT on the teleprinter. This allows the user to prepare for the next segment of his input (if paper tape reader, depress the tape-feed control after loading the tape) and then type ↑P.

At the end of pass 1 (.END encountered for the first time), MACRO-9 outputs PASS 1 COMPLETED on the teleprinter. This informs the user to prepare the input device for the beginning of pass 2 (if paper tape reader depress the tape-feed control) and then type ↑P. If the input device is

bulk storage (DECtape, magnetic tape, disk, drum) and .EOT was not encountered during pass 1, MACRO-9 automatically starts processing pass 2.

A secondary input is allowed at assembly time. This must contain only direct assignment statements, and is specified by writing "P" (for parameter tape) in the command string.

Secondary input must consist of direct assignments only, and must terminate with →| .EOT↵ . If the secondary input is assigned to the Teletype, the user must type an additional ↵ (for double buffering purposes) or may terminate with ↑D. The parameter tape will be inputted in pass 1 only, and not in pass 2.

Secondary input is useful, for example, to enter octal codes of IOTs deleted from the Permanent Symbol Table, such as:

```
PSF=700201
RSB=700144
```

3.4.4.3 PIP-9 (Peripheral Interchange Program)

PIP-9 is a utility program in the PDP-9 Advanced Software System used to transfer data files from one standard peripheral storage device to another. During transfer, PIP-9 can update file descriptions, rename, delete, insert and combine files. Since automatic storage retrieval is a standard function of the Monitor I/O Handlers, no provision for it is necessary in PIP-9.

PIP-9 operates under control of the Monitor, using the Monitor I/O Handler routines attached to .DAT slots 1 to 10 (unwanted handler requests should be removed from .DAT slots 1 to 10 via the ASSIGN NONE feature). The user directs PIP-9 functions by typing Teletype commands.

When PIP-9 is loaded and ready for operation, it outputs the following on the teleprinter

```
PIP
>
```

This indicates that PIP is ready to accept a keyboard command typed on the same line as the right angle bracket (>). At this point, the user should prepare the input/output devices required in the operation and then type the PIP command.

Successful completion and readiness for the next command is normally acknowledged by

```
>
```

unless there has been intermediate output to the teleprinter by PIP. In the latter case,

```
PIP
>
```

is output once again for ease of later printout examination.

The general format of a PIP command string is as follows:

F DDU: FILE1; EX1, FILE2; EX2 (S) ← SDU: FILE3; EX3

terminated by a carriage return or ALT mode.

Colon (:) and semicolon (;) are not required to delimit the device and file names, respectively. A space may be used instead of a ":" or ";". Spaces are meaningless elsewhere in the command string and may be used freely except within the body of the device or file names.

where F is a function character

T = transfer file
L = list directory
D = delete file
C = copy
N = rename file

DDU is the destination device (and unit, if applicable)

PR = paper tape reader
PP = paper tape punch
TT = Teletype
LP = line printer
DTu = DECtape
MTu = Magnetic tape
CD = card reader
DRu = drum
DKu = disc

FILEN; EXN are the file names and extensions involved in the operation. (They are omitted if device not file oriented.)

S indicates the switch options

A = IOPS ASCII
B = IOPS Binary
I = Image Alphanumeric
H = Image Binary
D = Dump
W = strip .EOT from IOPS ASCII input
or W = strip EOF from IOPS Binary
C = convert multiple spaces to tabs
S = create new system directory
Z = zero out directory

E = directs PIP to convert horizontal tabs to spaces for off-line listing on Model 33 Teletypes. It may be used only with IOPS ASCII (A) as the data mode.

Example: T PP←DT1 FILEA SRC (AE)↵
will punch FILEA on paper tape, with spaces instead of horizontal tabs.

G = directs PIP to examine input parity, allowing the user to modify input lines in error from the teletype. It is legal with IOPS ASCII only. On the occurrence of an input parity error, PIP always displays the following message on the teletype.

INPUT PARITY ERROR

If the G switch has been omitted from the input string, control returns immediately to the Monitor at this point.

However, with the G switch on, the line in error is also typed out and the user may take one of four possible actions:

a. The line may be deleted by typing

D↵

b. The line may be accepted as is by typing

↵

c. The line may be replaced by typing a new line to replace the one in error, followed by a carriage return.

d. PIP may be restarted by typing

↑P

Example: T DT1 FILEA SRC (AG) ←—PR↵

SDU is the source device (and unit if applicable) (Same options as DDU.)

Use of ALT mode to terminate a command string, forces PIP-9 to exit to the Monitor upon successful completion of the command. Use of carriage return as the terminator, causes PIP-9 to wait for another command upon completion of the current one.

Correction Procedures

Procedures available for command string correction are:

a. The rubout (RO) key deletes the input character immediately preceding. A\ is echoed for each RO input.

b. ↑U deletes the entire input line. The user begins the line from the first input characater.

When PIP-9 detects a command string error, the questionable command string is output up to but not including the offending character followed by "?", requiring correct completion by the user. If the user prefers to retype the command, a carriage return will in this instance signal PIP-9 to accept a new command from the beginning. The character RO and ↑U may not be used in this context since the Teletype handler (which no longer has access to the erroneous command string) and not PIP-9 interprets and acts upon RO and ↑U.

A task may be aborted and PIP-9 restarted by typing in the ↑P character at any time. ↑P has a secondary use in PIP-9 which is to indicate loading of the next in a series of paper tapes. For example, at the end of each of several paper tapes to be combined into one output file, PIP-9 will output to the teleprinter ↑P, informing the user that he may load the next paper tape and type ↑P for PIP-9 continuation.

Refer to the PDP-9 Utility Systems Manual (DEC-9A-GUAB-D) for detailed descriptions of all the functions available accompanied by pertinent examples.

3.4.4.4 Text Editor (EDIT-9) - When the Text Editor is loaded and ready to accept an input command string, it prints:

```
EDITOR
>
```

on the Teletype. If the input device is the paper tape reader, the user should load the tape to be edited, then momentarily depress the tape-feed control button to clear the reader-out-of-tape flag. At this point the user may type:

```
OPEN FILNME EXT ↵
```

or

```
↵
```

The first format (OPEN FILNME EXT ↵) is used when a current file, called FILNME with an extension EXT (SRC is assumed if the extension is omitted) is on the device associated with .DAT slot -14 and is to be modified. The editing process results in a new file, with a temporary name, being created on the device associated with .DAT slot -15. At the end of the editing process, the temporary file on the device associated with .DAT slot -15 replaces FILNME EXT (and is renamed FILNME EXT) on the device associated with .DAT slot -14. Of course, this replacement only occurs when both devices (.DAT slot -14 and -15) are bulk storage. Otherwise the final output is on .DAT slot -15.

The second format (CR) is used when a new file is to be created, via Teletype keyboard input on the device associated with .DAT slot -15. If the devices associated with .DAT slots -14 and -15 are both bulk storage, at the end of the editing process, the temporary file on .DAT slot -15 will be written on .DAT slot -14 with the file name and extension specified in the command string on the CLOSE command to the EDITOR.

Whenever the EDITOR is waiting for a command from the keyboard, it indicates this by outputting

```
>
```

on the teleprinter.

The CLOSE command to the EDITOR closes the current file and causes the EDITOR to cycle back to its initial printout

```
EDITOR
>
```

The EXIT command to the EDITOR causes an .EXIT return to the Monitor.

3.4.4.5 Linking Loader (LINK-9) - When the Linking Loader is loaded into core and is ready to accept an input command string from the keyboard, it will type:

```
LOADER
>
```

on the Teletype. At this point, the input device should be set up; if it is the paper tape reader, the tape-feed control button should be depressed momentarily, after the paper tape is loaded, to clear the reader-out-of-tape flag.

The input command string to the Linking Loader should consist of the list of file names of all programs to be unconditionally loaded from the sytem input device (.DAT slot -4). The file names specified to the Linking Loader should agree with the names originally used in the FORTRAN IV compilation or MACRO-9 assembly of the programs; they should be typed on the Teletype keyboard in the following format:

```
LOADER
>NAME 1, NAME 2, NAME 3↵
>NAME 4, NAME 5, (ALT MODE)
```

The main program must be requested first, followed by any desired subprograms. The file names should be one to six characters in length, with any characters over six being ignored. Only the file names should be specified; the Linking Loader will automatically assume that the file name extension is BIN (relocatable binary) and will search on both file name and extension.

A file name, in the input command string, is terminated by a comma (,), a carriage return (↵), or an ALT MODE. Until one of these three characters is encountered, n rubouts may be used to delete the previous n characters of the file name. ALT MODE terminates the input command string, and carriage return (↵) causes the loader to echo a line feed (↓), close angle bracket (>), so that the following format may also be used.

```
>NAME 1↵ ↓
>NAME 2↵ ↓
  .
  .
  .
>NAMEM (ALT MODE)
```

When the input device is not file-oriented, n commas, followed by the ALT mode character, prepares the Linking Loader to load n + 1 programs from the system input device.

After loading all programs requested in the keyboard input command string, the Loader will attempt to resolve all unsatisfied subroutine requests by scanning the external library (.DAT slot -5) and system library (.DAT slot -1), in that order.

If an external (user) library file exists and should be used for the programs being loaded, inform the Linking Loader of this by using the ASSIGN command to assign the I/O device on which the file is contained to .DAT slot -5 (before using the LOAD, DDT, DDTNS, or GLOAD command).

EXAMPLE:

```
ASSIGN  DTA4  -5
$LOAD
```

The external library file must be named .LIBR, with a file name extension of BIN; it must be in the format shown in Figure 3-2.

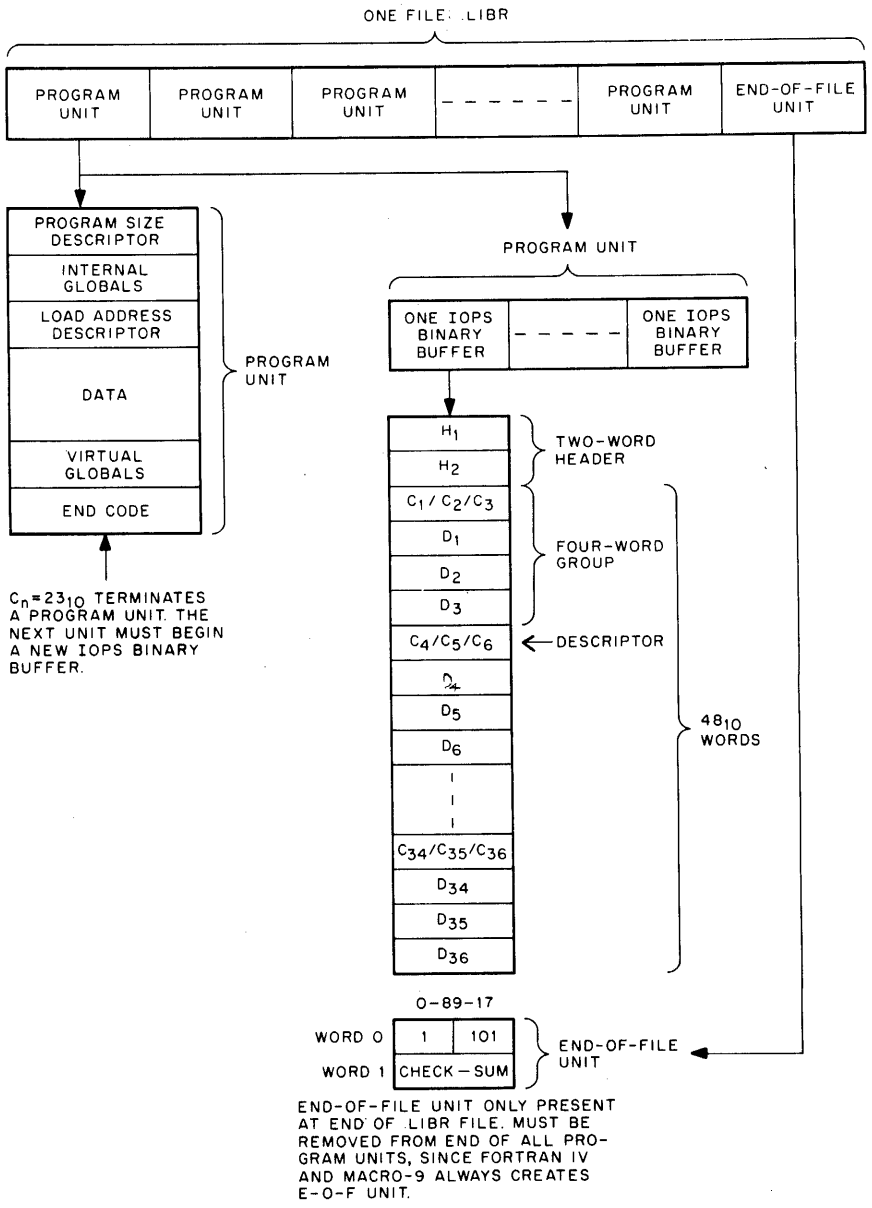


Figure 3-2 Library File Structure

If a loading error occurs, an appropriate error message (see Section 2.5.5.2) is printed on the Teletype, and control transfers to the System Bootstrap to re-initialize the Keyboard Monitor.

When all requested programs are loaded and all library requests (explicit and implicit) are satisfied, the Linking Loader will either:

a. (If the Linking Loader was called via LOAD) output ↑S and wait for the user to type ↑S on the Teletype, then transfer control to the starting address of the user's main program, or

b. (If called by GLOAD) immediately transfer control to the starting address of the user's main program.

When the user program finishes and returns control to the Monitor via an .EXIT command, the System Bootstrap is called to re-initialize the Keyboard Monitor. It then awaits its next command.

A memory map of the program(s) is printed during loading. It will have the same format as discussed in Section 2.5.5.1.

3.4.4.6 Debugging System (DDT-9) - The procedures for loading a program with DDT-9 are identical to those used with the Linking Loader (see Section 3.4.4.5) with one exception: after loading the program and the DDT symbol table*, control is automatically transferred to the starting address of DDT-9.

When ready, DDT-9 will type

```
DDT
>
```

informing the user that it is ready to accept a command.

The user can force control back to the starting address of DDT-9 at any time by typing

```
↑T
```

on the Teletype keyboard.

DDT-9 uses .DAT slots -6 and -10 for hard copy patch output and input, respectively. If the user is not going to take advantage of the patch capabilities of DDT-9, he should assign NONE to those .DAT slots so that unnecessary device handlers are not loaded into core, i.e.;

```
ASSIGN  NONE  -6, -10
$DDT
```

3.4.4.7 System Generator (SGEN-9) - Prior to requesting the System Generator via the SGEN Keyboard command, request information about the current operating environment by entering the following device examination and information keyboard commands:

*The DDT symbol table will not be loaded if DDTNS, instead of DDT, keyboard command was used to initiate loading.

SCOM	Causes output of certain system information, including a list of the device handlers available and a brief description of each of their features.
REQUEST SGEN	Causes output of .DAT slot assignments used by the System Generator and the use made of each .DAT slot. If any assignment does not agree with the user's need, it can be changed via the ASSIGN keyboard command (being aware of the handlers available as listed by the SCOM command and the handler requirements of SGEN (Section 4.6.10).
INSTRUCT SGEN	Causes printout of the basic operational instructions for the System Generator.

NOTE: It is imperative that your old system device be assigned to .DAT slots -10 and -14 and the unit that will contain your new system device be assigned to .DAT -15.

When the System Generator is loaded (via the SGEN command) and ready to begin questioning the user on items pertinent to the building of a new system tape, it outputs the following introduction and then proceeds with the questioning*.

```
SYSTEM GENERATOR

THIS PROGRAM WILL GENERATE A NEW SYSTEM TAPE
ON THE DEVICE SPECIFIED IN .DAT SLOT -15.  IT WILL
DETERMINE THE CHARACTERISTICS OF THIS SYSTEM
TAPE BY ASKING YOU A SERIES OF QUESTIONS.
IF IT CANNOT UNDERSTAND THE ANSWER YOU GIVE,
IT WILL REPEAT THE QUESTION.  HERE GOES!

HOW MUCH CORE IS AVAILABLE?  TYPE 8, 16, 24, OR 32.
>16
IS AN API AVAILABLE?  TYPE Y OR N.
>N
IS AN EAE AVAILABLE?  TYPE Y OR N.
>Y
INDICATE THE PRESENCE OR ABSENCE OF THE FOLLOWING DEVICE
HANDLERS BY TYPING Y OR N:
```

*The answers to questions must be terminated by a carriage return.

```
PRA?    >Y
PRB?    >Y
PPA?    >Y
PPB?    >N
PPC?    >Y
LPA?    >N
CDE?    >N
CDB?    >N
DTA?    >Y
DTB?    >Y
DTC?    >Y
DTD?    >Y
ARE ANY OTHER DEVICE HANDLERS PRESENT?  TYPE Y OR N.
>Y
HOW MANY?  TYPE OCTAL NUMBER.
>2
TYPE THREE CHARACTER HANDLER NAME FOR NO. 01.
>AAA
HOW MANY SKIP IOTS SHOULD BE IN SKIP CHAIN FOR THIS
DEVICE HANDLER?  TYPE OCTAL NUMBER.
>1
TYPE UP TO FIVE CHARACTER MNEMONIC FOR SKIP IOT NO. 01,
A COMMA, AND OCTAL SKIP IOT.
>AASF, 701111
TYPE THREE CHARACTER HANDLER NAME FOR NO. 02.
>ZZZ
HOW MANY SKIP IOTS SHOULD BE IN SKIP CHAIN FOR THIS
DEVICE HANDLER?  TYPE OCTAL NUMBER.
>3
 TYPE UP TO FIVE CHARACTER MNEMONIC FOR SKIP IOT NO. 01,
A COMMA, AND OCTAL SKIP IOT.
>ZSF, 702221
TYPE UP TO FIVE CHARACTER MNEMONIC FOR SKIP IOT NO. 02,
A COMMA, AND OCTAL SKIP IOT.
>ZZSF, 703331
TYPE UP TO FIVE CHARACTER MNEMONIC FOR SKIP IOT NO. 03,
A COMMA, AND OCTAL SKIP IOT.
>ZZZSF, 704441
```

NOTE: Refer to Section 4.6 for detailed description of all I/O Device Handlers.

```
THE FOLLOWING SKIP IOTS ARE TO BE INCLUDED IN THE
SYSTEM SKIP CHAIN:

CLSF
KSF
TSF
RSF
PSF
DTDF
DTEF
AASF
ZSF
ZZSF
ZZZSF

TYPE THEM IN SKIP CHAIN ORDER, ONE PER NUMBER.

NO. 01?     >ZSF
NO. 02?     >ZZSF
NO. 03?     >ZZZSF
NO. 04?     >AASF
NO. 05?     >CLSF
NO. 06?     >DTDF
NO. 07?     >RSF
NO. 10?     >PSF
NO. 11?     >KSF
NO. 12?     >TSF
NO. 13?     >DTEF

WHICH DEVICE HANDLER IS TO BE USED FOR THE SYSTEM DEVICE?
(NOTE: THIS MUST BE SMALLEST INPUT ONLY HANDLER)
>DTC

TYPE THE DEVICE HANDLER NAME (NON FOR NONE) AND THE UNIT NO. FOR THE
FOLLOWING .DAT SLOTS:

-15?    >DTA4
-14?    >DTA3
-13?    >DTB1
-12?    >TTA
-11?    >DTB3
-10?    >PRA
 -6?    >PPC
 -5?    >DTC2
 -4?    >DTC1
 -1?    >DTC0
  1?    >TTA
  2?    >TTA
  3?    >DTA1
  4?    >TTA
  5?    >DTA1
  6?    >PRA
  7?    >AAA
 10?    >ZZZ
```

NOTE: Structure the skip chain so that those devices requiring faster service have their skip IOT(s) at the beginning of the chain.

NOTE: Refer to Section 4.6.10 for system program .DAT slot requirements in setting up the negative .DAT slots. The positive .DAT slots belong to the user, and the standard setup should be a function of the needs of the users at your installation.

Once the new system tape has been constructed, it can then be used as the installation standard tape by mounting it on the system device unit (i.e., unit 0 for DECtape), reading in the paper tape system bootstrap, etc.

3.4.4.8 Dump Routine (DUMP-9) - This system program gives the user the ability to output on any listing device specified core locations that had been preserved on a bulk storage file via the ↑Q monitor dump command (see Section 3.2.3.5). When DUMP-9 is ready to accept the user's dump command string from the keyboard, it prints:

```
DUMP
>
```

on the Teletype.

The formats expected by the DUMP-9 command string processor are as follows:

COMMAND	MEANING
ALL	The entire ↑Q area (from location 10 to the address in .SCOM) on the device associated with .DAT slot -14 (at ↑Q time, this device was the specified output device) is listed on the device associated with .DAT slot -12.
XXXXX-YYYYY	The ↑Q area between absolute addresses XXXXX and YYYYY on the device associated with .DAT slot -14 is listed on the device associated with .DAT slot -12. At ↑Q time, this device (.DAT slot -14) was the specified output device and XXXXX and YYYYY were the absolute (octal) bounds of the core area to be dumped.
ZZZ#	The content of block #ZZZ on the device associated with .DAT slot -14 is listed on the device associated with .DAT slot -12. The block number is in octal radix.

If a command is terminated by a carriage return (↵), control returns to the command string processor after completion of the request.

```
DUMP
>
```

will be printed on the teleprinter indicating readiness for another command.

If a command string is terminated by the ALT mode character, control returns to the Monitor upon completion of the request.

Any unrecognizable commands cause ? to be typed and control to return to the command string processor will type > to indicate its readiness for a command.

EXAMPLE:

```
MONITOR
$ASSIGN DTD0-14
$ DUMP
```

```
DUMP
>16730-16750
 16730  000032  003740  013777  000000  000000  413420  013422  463356
 16740  127400  463356  127400  000612  003766  003773  000000  020202
 16750  000000
DUMP
  >
```

3.4.4.9 Library Update Program (UPDATE-9) - This system program gives the user the capability of building, examining and updating library files.

When UPDATE-9 is ready to accept the user's file specifying command string from the keyboard, it outputs

```
UPDATE
>
```

on the teleprinter.

The user should then type, on the same line as the right angle bracket (>), a file specifying command string in the following format:

Options	File Name	Terminator
L, U, N	← FILEX	↵ or ALT mode

L - Library file listing on .DAT -12

U - Update from .DAT -14 to .DAT -15 with secondary input on .DAT -10

N - Create from .DAT -10 onto .DAT-15

Neither U or N - Input on .DAT -14 (no output on .DAT -15 or secondary input on secondary input on .DAT -10) used primarily with CLOSE command to get clean library file listing on .DAT -12.

The default library file name is .LIBR (the file name used in library .SEEK's by the Linking Loader). The file name extension is always assumed to be BIN.

If the file specifying the command string is terminated by

a. ALT mode, control will be returned to the Monitor when updating of the current file is completed.

b. carriage return (↵), control will remain with UPDATE when work on the current file is complete and it will output

```
UPDATE
>
```

to the teleprinter to indicate readiness for the next file specifying command.

When UPDATE is ready for a libary file manipulation command, it outputs >

to the teleprinter. The user should now type a file manipulation command on the same line as the right angle bracket (>) terminated by carriage return (↵) and in the following format.

DELETE [D]* NAME	Delete the named routine from file, copying all previous routines (valid command only if U option)
REPLACE [R]* NAME 1, NAME 2	Replace NAME 1 with NAME 2 (default is NAME 1), copying all previous routines (valid command only if U option). Replacement comes from .DAT -10.
INSERT [I]* NAME3, NAME 4	Insert NAME 3 after NAME 4 (default is last routine processed or beginning of file) (copying all previous routines if U option). Specifying a second argument (NAME 4) is only valid in U mode. Insert comes from .DAT -10.
END [E]*	Position at end of file (copying all routines if U option).
KILL [K]*	Abort operations on current file, destroying bad output file. Issued when user detects trouble with updating process.
CLOSE [C]* FILENM	Performs END if not done; clears up at end of update satisfying all options and giving the output file the name FILENM (default is .LIBR even if a NAME was given in file specifying command string, i.e.; U ←NAME ↵). EXIT to Monitor or remain in UPDATE for next file as a function of the file specifying command string terminator (ALT mode or ↵ .

The library file listing on .DAT slot -12 will be in the following format.

LIBRARY FILE LISTING FOR FILE NM PAGE 1

PROGRAM NAME	PROGRAM SIZE	ACTION
	0	DELETE NAME
NAME 2	477_8	REPLACE NAME 1, NAME 2
NAME 4	352_8	
NAME 3	517	INSERT NAME 3, NAME 4
⋮	⋮	⋮

*Entire name or first character of name will suffice as file manipulation commands.

Error messages - recoverable

If command completely unintelligable

```
?
>
```

If Delete, Replace, Insert (with 2 arguments) used with other than U option

```
VALID ONLY IN U MODE - COMMAND IGNORED
>
```

If Delete, Replace, Insert (with 1 argument) used without U or N option

```
VALID ONLY IN U OR N MODE - COMMAND IGNORED
>
```

If no name given after Insert, Delete, Replace

```
ILLEGAL COMMAND STRUCTURE - COMMAND IGNORED
>
```

If program requested in any command not found in forward direction (tape at end)

```
EOF REACHED BY SEARCH - COMMAND IGNORED
>
```

This file is still open and may be accessed via INSERT, CLOSE and KILL commands.

If wrong program used as input on .DAT slot -10 for Replace or Insert command

```
WRONG PROGRAM AS INPUT - CORRECT INPUT AND ↑P
```

Set up input device with the correct program and then type ↑P on the keyboard.

Error messages - terminal (new file specifying command required)

If end code found before program name on binary input

```
PROGRAM NAME MISSING - DYNAMIC KILL
UPDATE
>
```

If not enough room in core for program

```
BUFFER OVERFLOW - DYNAMIC KILL
UPDATE
>
```

If read error on input buffer

```
UNRECOVERABLE READ ERROR ON .DAT N - DYNAMIC KILL
UPDATE
>
```

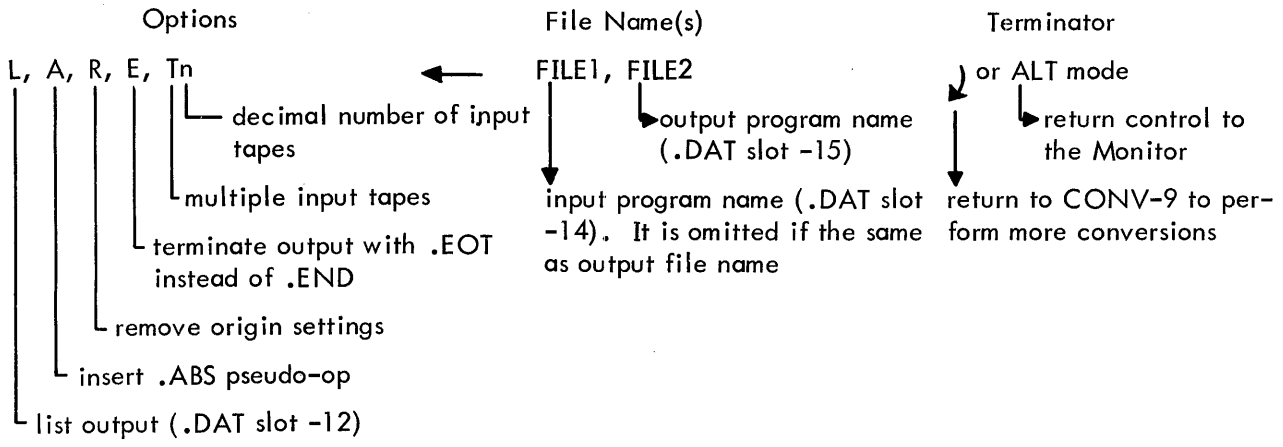
3.4.4.10 7-to-9 Converter, (CONV-9) - The 7-to-9 Converter converts source programs written for the PDP-7 or basic PDP-9 assemblers to a format acceptable to the ADVANCED Software PDP-9 MACRO-9 Assembler.

When CONV-9 is loaded and ready for operation, it prints the following on the teleprinter.

```
7-TO-9 CONVERTER
>
```

This indicates that CONV-9 is ready to accept a command string (on the same line as the >) telling it what action to take.

The command string format is:



Default Assumptions

NO listing

NO inserting of .ABS pseudo-op

NO removal of origin settings

.END instead of .EOT

ONE input tape

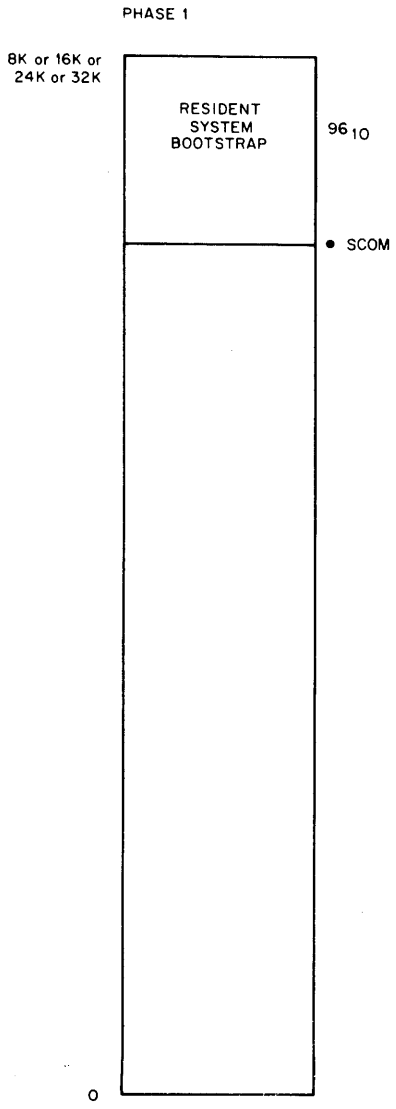
The options desired may appear in any order, separated by commas and terminated by ←. If no options are wanted, ← is sufficient. Rubouts may be used to delete unwanted characters prior to typing the command string terminator. If an error in the command string is detected, CONV-9 types

```
COMMAND STRING ERROR
>
```

and waits for a new command string. Refer to PDP-9 Utility Programs Manual (DEC-9A-GUAB-D) for lists of items that cannot be correctly converted by CONV-9 and must be modified by the programmer himself.

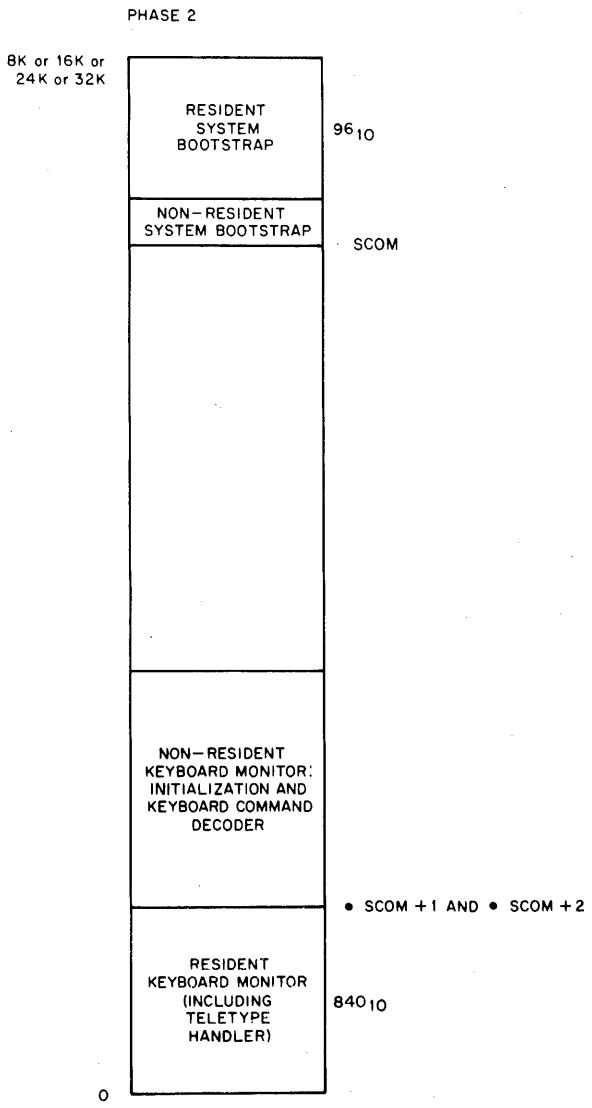
3.4.5 Error Detection and Handling

Error detection in the Keyboard Monitor system is handled in the same manner as in the I/O Monitor environment (see Section 2.6) except that after error messages are output, control automatically returns to the System Bootstrap for re-initialization of the Keyboard Monitor. If SDUMP has been issued prior to execution of this program, an automatic ↑Q (dump the current job, in core image, onto pre-specified blocks of the system device) takes place before control is returned to the Bootstrap. This dumped file can than be selectively interrogated (listed) by the system program DUMP.



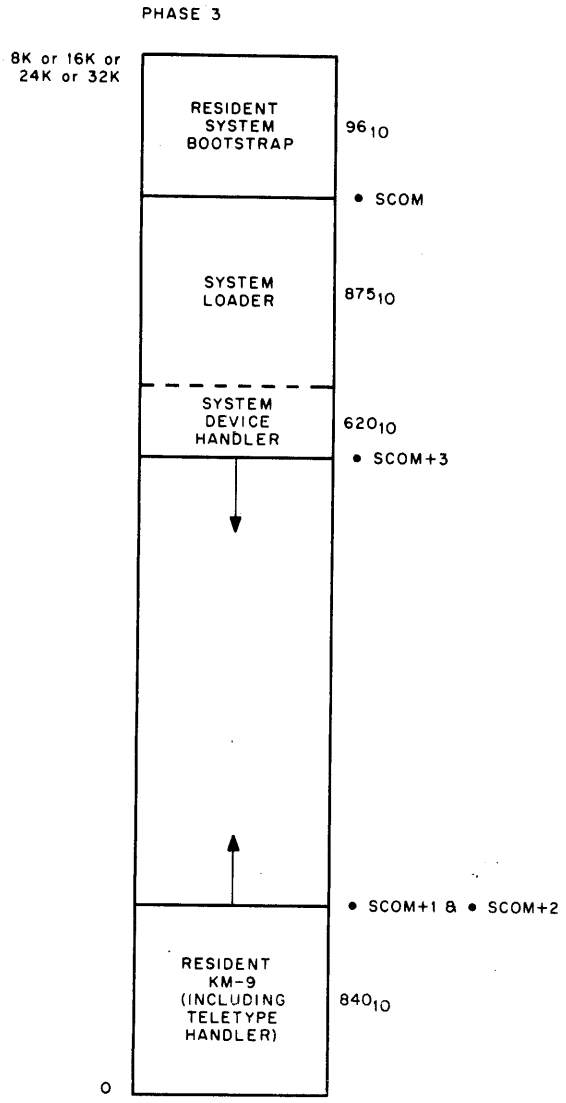
The System Bootstrap is loaded via the paper tape reader in HRM mode.

Figure 3-3 Memory Maps: Bulk Storage Systems



The System Bootstrap loads (DUMP mode) the Keyboard Monitor (resident and non-resident) from the system device.

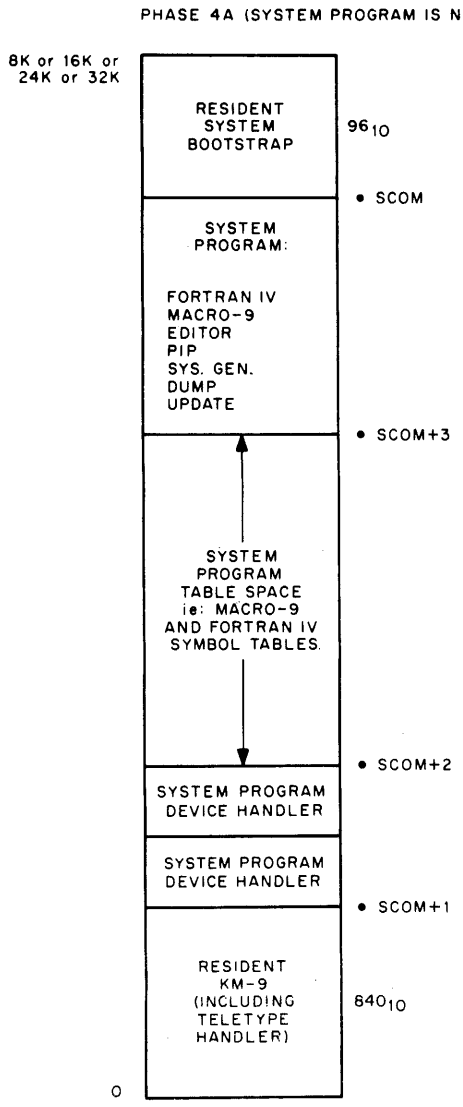
Figure 3-3 Memory Maps: Bulk Storage Systems (cont)



The Keyboard Monitor loads (DUMP mode) the System Loader and the system device handler from the system device.

The System Loader, during loading of a system program from the system device, builds the loader (GLOBAL) symbol table down from .SCOM+3 and the programs up from .SCOM+2.

Figure 3-3 Memory Maps: Bulk Storage Systems (cont)

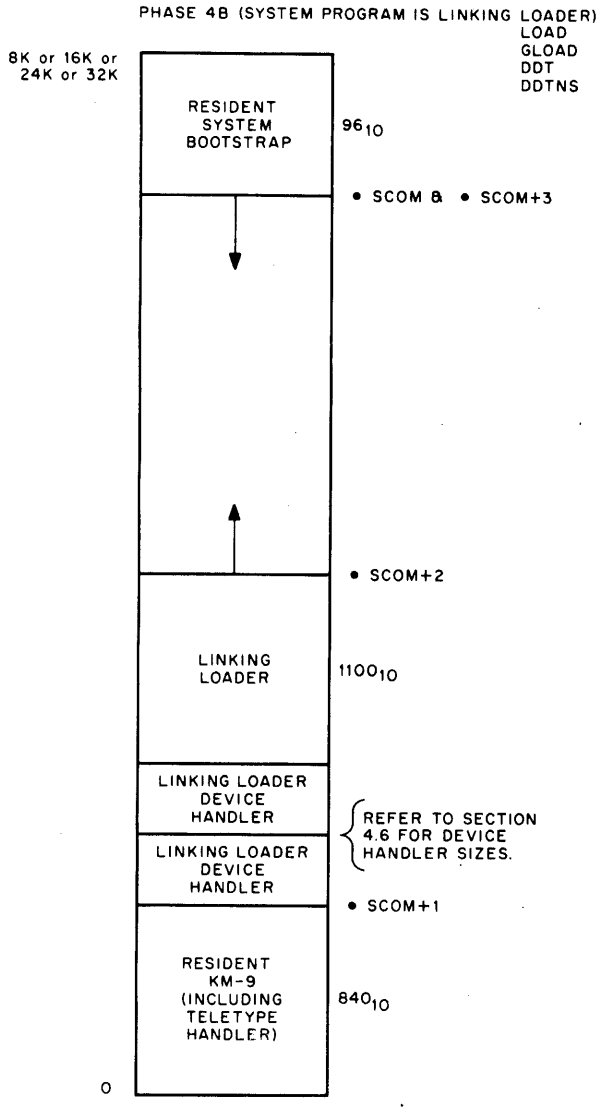


The System Loader learns which I/O handlers are required by the requested system program from its table of .IODEV info for system programs, loads the handlers relocatably just above the resident KM-9 and then modifies the System Bootstrap to bring in the system program in Dump mode just below the Bootstrap.

.EXIT from the system program takes the process back to PHASE 2 where the system bootstrap reinitializes the Keyboard Monitor.

Refer to Section 4.6 for the sizes of the device handlers that may be associated with the .DAT slots used by the System program.

Figure 3-3 Memory Maps: Bulk Storage Systems (cont)



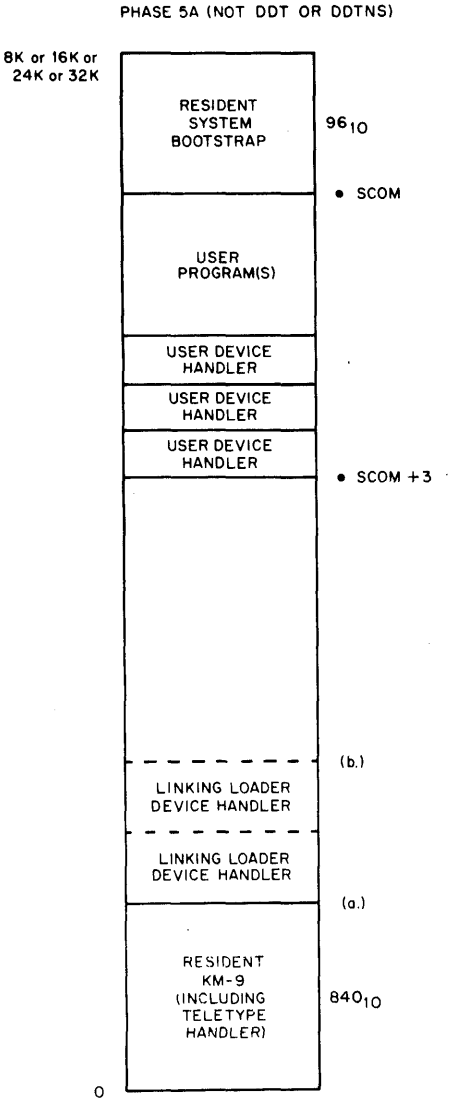
The System Loader learns which I/O handlers are required by the Linking Loader, loads them relocatably and then loads the Linking Loader relocatably.

The Linking Loader, during loading of user programs down from .SCOM+3, builds the loader (GLOBAL) and DDT (if DDT) symbol tables up from .SCOM+2. DDT symbol table will not be built if a LOAD, GLOAD, or DDTNS load.

If a DDT load, the Linking Loader just prior to giving control to DDT moves the DDT symbol table down in core so that is overlays all of the Linking Loader except for the small routine that makes the block transfer.

The Linking Loader will not load a device handler that is already in core for its own use.

Figure 3-3 Memory Maps: Bulk Storage Systems (cont)



.EXIT from the user program takes the process back to PHASE 2 where the system bootstrap re-initializes the Keyboard Monitor.

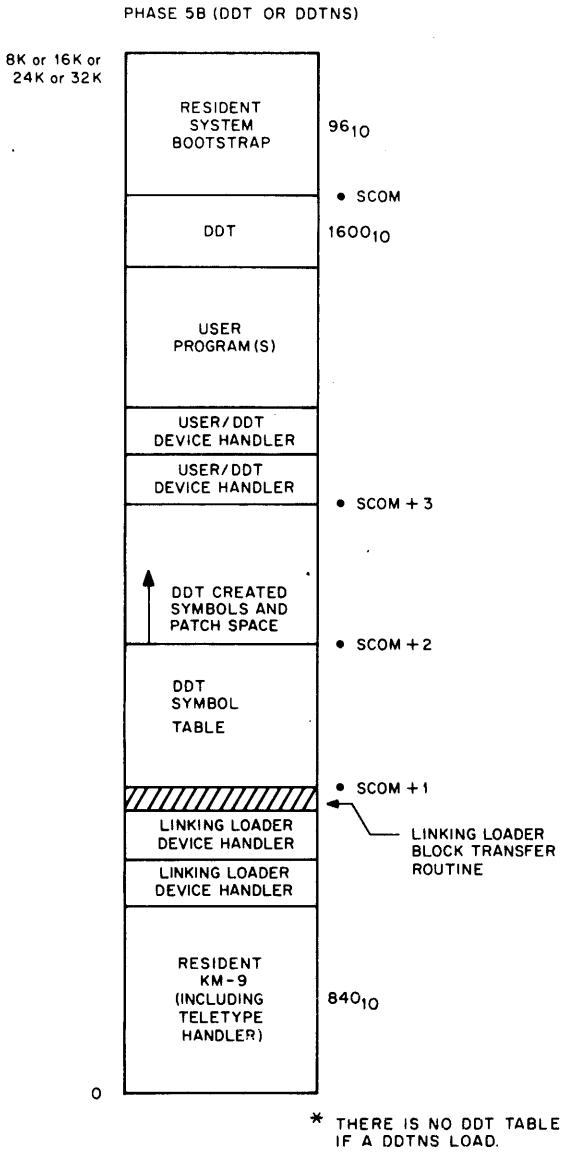
Refer to Section 4.6 for sizes of device handlers.

.SCOM+1 and .SCOM+2 both point to one of two places and non-BLOCK DATA COMMON (FORTRAN IV or MACRO-9) output may make use of core as low as they point.

a. If the user program did not have any device handlers in common with the Linking Loader.

b. If the user program did have at least one device handler in common with the Linking Load.

Figure 3-3 Memory Maps: Bulk Storage Systems (cont)



.EXIT from the user program takes the process back to PHASE 2 where the system bootstrap re-initializes the Keyboard Monitor

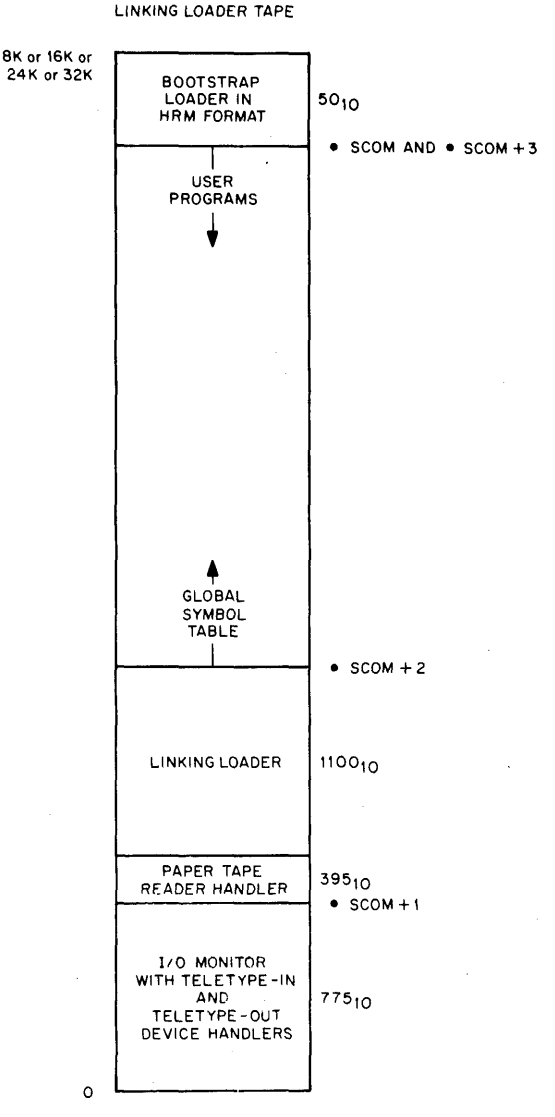
Refer to Section 4.6 for sizes of device handlers.

Non BLOCK DATA COMMON (FORTRAN IV of MACRO-9 output) may make use of core as low as the DDT symbol table*. However, trouble will occur if the user requests DDT to create symbols or make patches that cause overlaying of the COMMON area.

The Linking Loader device handlers would have been used to satisfy user device requests.

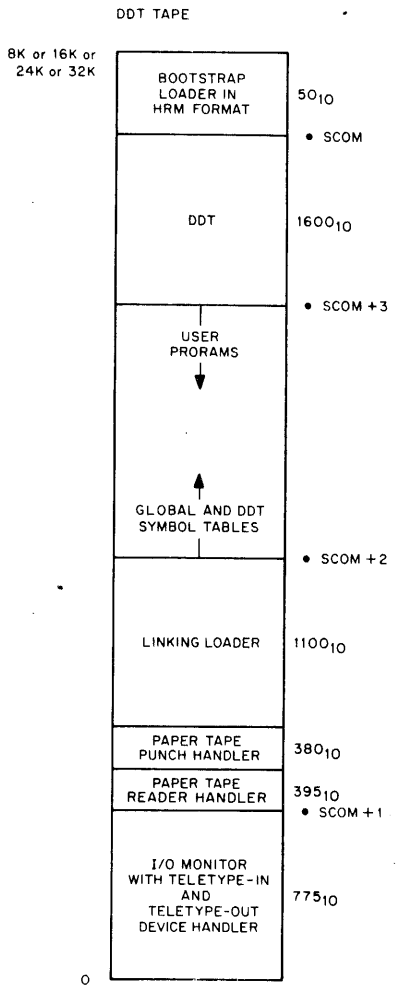
*There is no DDT table if a DDTNS load.

Figure 3-3 Memory Maps: Bulk Storage Systems (cont)



Refer to memory map 5A of Bulk Storage Systems for results of Link Loading.

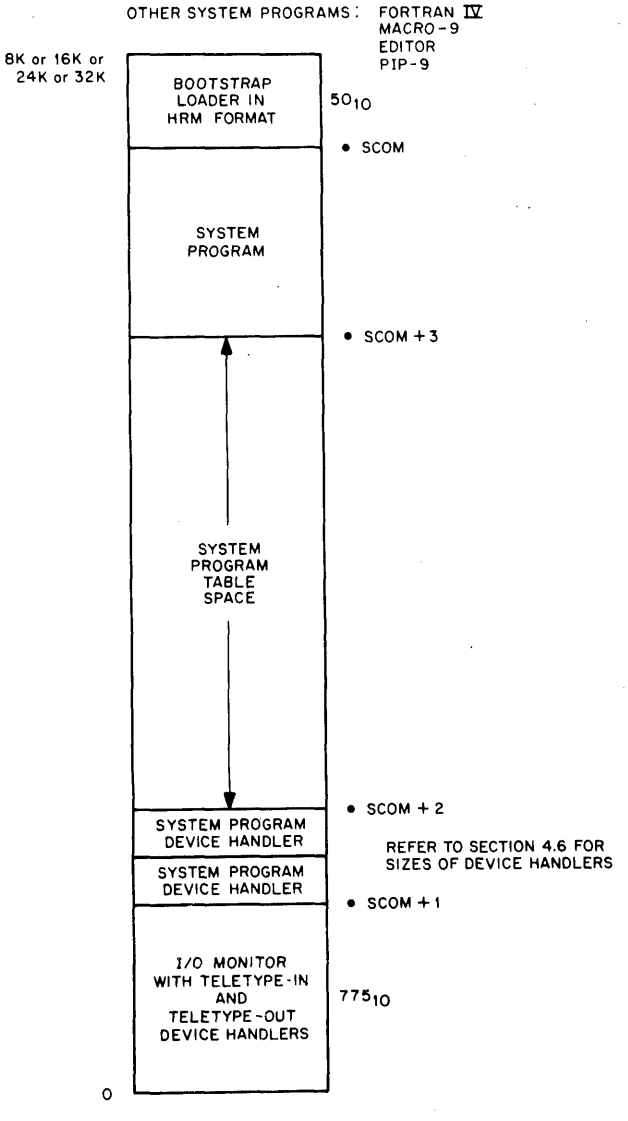
Figure 3-4 Paper Tape System (I/O Monitor)



Refer to memory map 5B of Bulk Storage Systems for results of Link Loading in DDT mode.

Paper Tape Punch Handler is only present in DDT versions with patch file capabilities.

Figure 3-4 Paper Tape System (I/O Monitor) (cont)



Refer to Section 4.6 for sizes of device handlers.

Figure 3-4 Paper Tape System (I/O Monitor) (cont)

CHAPTER 4
SUMMARY OF COMMANDS

4.1 SUMMARY OF USER PROGRAM COMMANDS

To initialize a device and device handler:

```
.INIT␣a,f,r
where   a = ·DAT slot number in octal
        f = 0 for input files; 1 for output files.
        r = user restart address*
```

To read a line of data from a device to a user's buffer.

```
.READ␣a,m,l,w
where   a = ·DAT slot number in octal.
        m = a number, 0-4, specifying the data mode:
              0 = IOPS binary
              1 = Image binary
              2 = IOPS ASCII
              3 = Image alphanumeric
              4 = Dump mode
        l = Line buffer address
        w = word count of the line buffer in decimal, including the
            two word header
```

To write a line of data from the user's buffer to a device:

```
.WRITE␣a,m,l,w
where   a = ·DAT slot number in octal
        m = a number, 0-4, specifying the data mode:
              0 = IOPS binary
              1 = Image binary
              2 = IOPS ASCII
              3 = Image alphanumeric
              4 = Dump mode
        l = Line buffer address
        w = word count of line buffer in decimal, including the
            two word header
```

*Only meaningful when device associated with ·DAT slot a is the Teletype. Typing ↑P on the keyboard will force control to location r.

To detect the availability of a line buffer:

```
.WAIT␣a
where   a = .DAT slot number in octal. After the previous .READ,
           .WRITE or .TRAN command is completed, control is
           returned to the user immediately after the .WAIT.
```

To close a file and update the device directory to include the new file name:

```
.CLOSE␣a
where   a = ·DAT slot number in octal
```

To set the real-time clock to n and start it:

```
.TIMER␣n,c
where   n = number of clock increments in decimal. Each increment
           is 1/60 second (in 60 cycle systems) or 1/50 (in 50 cycle
           systems).
        c = address of subroutine to handle interrupt at end of interval
```

To return control to Keyboard Monitor or I/O Monitor:

```
.EXIT
```

Mass Storage Commands for DECtape, Magnetic Tape, Disk and Drum Only

To search for a file, and position the device for subsequent .READ commands:

```
.SEEK␣a,d
where   a = ·DAT slot number in octal
        d = address of user directory entry block
```

To examine a file directory and find a free directory entry block:

```
.ENTER␣a,d
where   a = ·DAT slot number in octal
        d = address of user directory entry block
```

To clear a file directory to zero:

```
.CLEAR␣a
where   a = ·DAT slot number in octal
```

To rewind, backspace, skip, write end-of-file or write blank tape on non-file-oriented magnetic tape:

```
.MTAPE␣a,xx
where   a  = .DAT slot number in octal
        xx = a number, 00-07, specifying one of the functions shown below:
                00 = Rewind to load point*
                02 = Backspace one record*
                03 = Backspace one file
                04 = Write end-of-file
                05 = Skip one record
                06 = Skip forward one file
                07 = Skip to logical end-of-tape
             or a number, 10-16, to describe the tape configuration:
                10 = 7-channel, even parity, 200 BPI
                11 = 7-channel, even parity, 556 BPI
                12 = 7-channel, even parity, 800 BPI
                13 = 9-channel, even parity, 800 BPI
                14 = 7-channel, odd parity, 200 BPI
                15 = 7-channel, odd parity, 556 BPI
                16 = 7-channel, odd parity, 800 BPI
                17 = 9-channel, odd parity, 800 BPI
```

To read from, or write to any user file-structured mass storage device:

```
.TRAN␣a,d,b,l,w
where,   a = .DAT slot number in octal
         d = Transfer direction:
                0 = Input forward
                1 = Output forward
                2 = Input reverse (DECtape only)
                3 = Output reverse (DECtape only)
         b = Device address in octal, such as block number for DECtape
         l = core starting address
         w = word count in decimal
```

To delete a file:

```
.DLETE␣a,d
where   a = .DAT slot number in octal
        d = starting address of the 3-word block of storage in user area
            containing the file name and extension of file to be deleted
            from the device.
```

*May be used with any non-file structured mass storage device.

To rename a file:

.RENAM␣a,d

where a = ·DAT slot number in octal

d = starting address of two 3-word blocks of storage in user area containing the file names and extensions of the file to be renamed, and the new name, respectively.

To determine if a file is present on a device:

.FSTAT␣a,d

where a = ·DAT slot number

d = starting address of 3-word block in user area containing the file name and extension of the file whose status is desired.

4.2 SYSTEM COMMUNICATION TABLE (.SCOM)

Beginning at Location 100_8

Word	Purpose
.SCOM + 0	First free register below resident portion of System Bootstrap (constant)
.SCOM + 1	First free register above resident KM-9 (constant)
.SCOM + 2	First free register
.SCOM + 3	Last free register
.SCOM + 4	Hardware options available
.SCOM + 5	System program starting location
.SCOM + 6	User starting location
.SCOM + 7-11_8	Device numbers of Linking Loader's devices. These are used to avoid loading user handlers already in core for the Loader itself.
.SCOM + 12-15_8	Transfer vectors associated with API software level channel registers 40-43_8.
.SCOM + 16	Contains PC on Keyboard interrupts.
.SCOM + 17	Contains AC on Keyboard interrupts.

4.3 MAXIMUM LINE BUFFER SIZES (INCLUDING 2-WORD HEADER)

Device	Maximum Line Buffer Size	Data Modes*	Notes
PR (paper tape reader)	52_{10}	All	34_{10} sufficient if A mode only. Headers accepted for B; generated for A, I, Z
PP (paper tape punch)	52_{10}	All	34_{10} sufficient if A mode only. Headers output for B only.
TT (Teletype)	34_{10}	A, Z only	Allows for 80_{10} characters. Headers generated on input. Headers not output on output.
CD (card reader)	52_{10}	All	52_{10} for B mode; 82_{10} for I, Z modes. Headers accepted for B; generated for A, I, Z.
LP (line printer)	52_{10}	A only	Allows for 125_{10} characters. No headers output.
DT0-7 (DECtape)	255_{10}	All	IOPS mode allows for several line buffers or logical records/physical block.
MT0-7 (MAG. Tape)	255_{10}	All	Where IOPS ASCII and binary lines may be intermixed on a given IOPS mode block. Headers generated only for IOPS modes. Modes I, Z allow for only one line buffer or logical record/physical block.
DK (disk)	255_{10}	All	
DR (drum)	255_{10}	All	

* Data Modes are:
A = IOPS ASCII
B = IOPS Binary
D = Dump Mode
I = Image Binary
Z = Image Alphanumeric

4.4 SUMMARY OF KEYBOARD COMMANDS

Form	Page
LOG any comment (ALT mode)	
SCOM	
INSTRUCT XXXXXX	
REQUEST or REQUEST XXXXXX	
ASSIGN DEVn a, b, etc/DEVm, X, Y, etc.	
DIRECT or DIRECT n	
NEWDIR n	
LOAD	
GLOAD	
EDIT	
MACRO	
DDT	
DDTNS	
PIP	
F4	
SGEN	
DUMP	
UPDATE	
CONV	
GET N or GET N XXXXX or GET N HALT	
↑S	
↑C	
↑T	
↑Q␣	
↑P	
SDUMP	
↑R	
HALT	

4.5 MONITOR/IOPS COMMAND TABLE

.COMTB contains n 1-word entries, where each entry is a 15-bit address of the entry point of the corresponding Monitor/IOPS handler.

.COMTB	−n
1	.INIT
2	.DLETE, .RENAM .FSTAT
3	.SEEK
4	.ENTER
5	.CLEAR
6	.CLOSE
7	.MTAPE
10	.READ
11	.WRITE
12	.WAIT
13	.TRAN
14	.TIMER
15	.EXIT

−n = 1s complement of table length.

4.6 SUMMARY OF STANDARD I/O HANDLER FEATURES

4.6.1 LPA:(647 LINE PRINTER)

A. Function

1.	.INIT	(a) Return standard line buffer size (52_{10})
		(b) .SETUP --- API channel register 56_8
		(c) Clear line printer buffer
		(d) Form feed
2.	.DLETE .RENAM .FSTAT	Ignore
3.	.SEEK	Illegal function
4.	.ENTER	Ignore
5.	.CLEAR	Ignore
6.	.CLOSE	(a) Allow previous output to terminate
		(b) Form feed
		(c) Allow form feed to terminate
7.	.MTAPE	Ignore
10.	.READ	Illegal function
11.	.WRITE	(a) Allow previous output to terminate
		(b) Output line
12.	.WAIT	Allow previous output to terminate
13.	.TRAN	Illegal function

B. Legal Data Modes

1. IOPS ASCII

C. Vertical Control Characters (when first character of line)

12	Print every line
21	Print every second line
22	Print every third line
13	Print every sixth line
23	Print every tenth line
24	Print every twentieth line
20	Overprint
14	Form feed

D. Horizontal Control Characters (anywhere in line)

11	Horizontal tab -- converted to N spaces, where N is the number necessary to have the next character in column 11, 21, 31, 41, ...

E. Recoverable Errors

1. Device not ready	Monitor error message, .IOPS 4. Make device ready, then type ↑R to continue.

F. Unrecoverable Errors

1. Illegal function	Monitor error message, .IOPS 06 XXXXXX, where XXXXXX is address of error CAL. (a) .SEEK (b) .READ (c) .TRAN
2. Illegal data mode	Monitor error message, .IOPS 07 XXXXXX, where XXXXXX is address of error CAL. Any mode other than IOPS ASCII.

I. Program Size

275 (decimal) registers.

4.6.2 TTA:(TELETYPE)

A. Functions - All function descriptions (except READ and WRITE) refer to action taken when either the teleprinter or the keyboard is addressed.

Wait for previous I/O to finish on all functions.

1. .INIT
   (a) Return standard buffer size (34_{10}).
   (b) Assign return addresses for certain control characters from contents of CAL ADDRESS + 2. Bits 0-1 in CAL + 2 address are set to designate caller:

B0, 1 = 01	caller = KM9
B0, 1 = 10	caller = DDT
B0, 1 = 00	caller = any other user

   (c) Set I/O UNDERWAY indicator.
   (d) Print CR/LF.

2. .DLETE
   .RENAM
   .FSTAT
   Ignore

3.	.SEEK	Ignore
4.	.ENTER	Ignore
5.	.CLEAR	Ignore
6.	.CLOSE	(a) Set I/O UNDERWAY indicator. (b) Print CR/LF. (c) Wait on .CLOSE for completion of I/O.
7.	.MTAPE	Ignore
10.	.READ	(a) Set I/O UNDERWAY indicator. (b) Set up to accept characters from keyboard.
11.	.WRITE	(a) Set I/O UNDERWAY indicator. (b) Print line.
12.	.WAIT	Ignore
13.	.TRAN	Illegal function.

B. Legal Data Modes

1. IOPS ASCII
2. IMAGE ASCII

C. Vertical Carriage Control Characters

1. Output
   (a) Line feed (12_8)
      (1) IOPS ASCII: Ignore all leading line feeds; otherwise output
      (2) IMAGE: Output
   (b) Others (vertical tab, form feed) output
2. Input — Inserted in buffer.

D. Horizontal Carriage Control Characters

Tab (11_8) in or out

1. IOPS ASCII
   (a) Model 33 - output sufficient number of spaces to place the next typed character in column 11, ..., 71. Insert only 11_8 in buffer on input.
   (b) Model 35 - input, insert 11_8 in buffer. Output, print tab.
2. IMAGE — Input, insert 11_8 in buffer. Output, print tab.

E. Program Control Characters-IN

1. Stop current I/O to Teletype.
2. Decode character and echo on Teleprinter.*
   (a) ↑C transfers control to the address specified as return in the .INIT performed by KM9.
   (b) ↑P transfers control to the address specified as return in the .INIT performed by the user (other than KM9 or DDT).
   (c) ↑T transfers control to the address specified as return in the .INIT performed by DDT.
   (d) ↑S transfers control to the address specified in .SCOM + 6.
   (e) ↑Q transfers control to KM9SAV.

F. Data Control Characters-IN

1. IMAGE Mode — All characters inserted in buffer
2. IOPS ASCII Mode
   (a) Rubout. Delete previous character typed. Type out reverse slash (\).
   (b) ↑U delete entire line typed so far. Type out commercial at (@). If output is UNDERWAY, printing is terminated and a CR/LF is output.

G. Data Control Characters-OUT (both modes)

Ignore rubout (177_8) and null (00).

In image alpha mode, a rubout should be used to fill the last word pair when an odd number of characters is to be output

H. Errors (no program-initiated recovery)

1. Illegal data mode — Error code 7
2. Illegal function — Error code 6

I. Program Size

410_{10} registers (this is included in resident MONITOR)

J. Teletype I/O - Can be requested only from mainstream in API systems, since the Teletype is not connected to the API.

*Character will be ignored (not echoed) in cases (a), (b), and (c) if respective .INIT has not been performed.

4.6.3 PP (PAPER TAPE PUNCH)

A. Functions

1.	.INIT	(a) Return standard buffer size (52_{10}). (b) .SETUP - no API. (c) Punch two fanfolds of leader.
2.	.DLETE .RENAM .FSTAT	ignore
3.	.SEEK	Illegal function.
4.	.ENTER	Ignore
5.	.CLEAR	Ignore
6.	.CLOSE	(a) Allow previous output to terminate. (b) Punch EOF if IOPS Binary (c) Punch two fanfolds of trailer (d) Allow trailer punching to terminate
7.	.MTAPE	Ignore
10.	.READ	Illegal function
11.	.WRITE	(a) Allow previous output to terminate. (b) Output buffer
12.	.WAIT	Allow previous output to terminate.
13.	.TRAN	Illegal function

B. Legal Data Modes

1. IOPS Binary
2. IMAGE Binary
3. IOPS ASCII
4. IMAGE ASCII
5. Dump

C. Vertical Control Characters (IOPS ASCII only)

May appear only as first character of line, if elsewhere in line will be ignored; if no vertical control character at beginning of line, a line feed (012) will be used.

1.	012	Line feed
2.	013	Vertical tab, followed by four deletes (177)
3.	014	Form feed, followed by 40_8 nulls (000)

D. Horizontal Control Characters (IOPS ASCII only)

1.	011	Horizontal tab, followed by one delete (177)

E. Recoverable Errors

1.	No tape in punch	Monitor error code 4
		(a) Put tape in punch
		(b) Type ↑R

F. Unrecoverable Errors

1.	Illegal function	Monitor error code 6
		(a) .SEEK
		(b) .READ
		(c) .TRAN
2.	Illegal data mode	Monitor error code 7

I. Program Size

PPA. (all data modes)	361 decimal registers
PPB. (all except IOPS ASCII)	254 decimal registers
PPC. (IOPS binary only)	194 decimal registers

J. In API systems, the paper tape punch can be called only from mainstream, since the punch is not connected to the API.

4.6.4 PR (PAPER TAPE READER)

A. Functions

1.	.INIT	(a) Return standard line buffer size (52_{10})
		(b) .SETUP API channel register 50_8
		(c) Clear I/O UNDERWAY indicator
2.	.DLETE .RENAM .FSTAT	Ignore
3.	.SEEK	Ignore
4.	.ENTER	Illegal function (error code 6)
5.	.CLEAR	Illegal function (error code 6)
6.	.CLOSE	Allow previous input to finish and then clear I/O UNDERWAY indicator.
7.	.MTAPE	Ignore

10. .READ
    (a) Allow previous input to be completed.
    (b) Input line or block of data (see modes below).
11. .WRITE
    Illegal function (error code 6).
12. .WAIT
    Allow previous input to be completed before allowing user program to continue.
13. .TRAN
    Illegal function.

B. Legal Data Modes (A11)

1. IOPS ASCII
    (a) Constructs line buffer header, computing:
        (1) Word pair count
        (2) Data mode
        (3) Data validity bits
    (b) Packs characters into the line buffer in 5/7 ASCII, checking parity (eighth bit, even) on each character.
    (c) Allows vertical form control characters (FF, LF, VT) only in character position 1 of the line buffer. Otherwise, ignored.
    (d) Terminates reading on CR or line buffer overflow. In the latter case, tape is moved past the next CR to be encountered.
2. IOPS BINARY
    (a) Reads binary data in alphanumeric mode, checking parity (seventh hole, odd) on each frame.
    (b) Accepts line buffer header at head of input data, modifying data validity bits if parity or checksum errors (or short line) have occurred.
    (c) Terminates reading on overflow of word pair count in line buffer header or word count in .READ macro, whichever is smaller, moving tape to end of line or block if necessary.
3. IMAGE ASCII
    (a) Constructs line buffer header, computing:
        (1) Word pair count
        (2) Data mode
    (b) Stores characters, without editing, or parity checking in the line buffer, one per register.
    (c) Terminates reading as a function of .READ macro word count.
4. IMAGE BINARY
    Same as 3 (a), (b), (c) above; however, a binary read is issued to the PTR.
5. DUMP
    Same as 3 (b), (c) above. A binary read is issued to the PTR. No header is constructed; loading begins at the core address specified in the .READ macro.

NOTE: An end of tape condition causes the PTR interrupt service routine to terminate the input line, turning off the I/O UNDERWAY program indicator and marking the header (data mode bits) as an EOM (end of medium) for all modes except DUMP.

C. Unrecoverable Errors

1. Illegal Function — Monitor error code 6
   (a) .ENTER
   (b) .CLEAR
   (c) .WRITE
   (d) .TRAN
2. Illegal Data Mode — Monitor error code 7

I. Program Size

PRA. (all data modes)	420 decimal registers
PRB. (IOPS ASCII only)	271 decimal registers

4.6.5 DT (DECTAPE)

A. Function

1. .INIT
   (a) Return standard line buffer size (255_{10})
   (b) .SETUP - API channel register 44_8
   (c) Set direction switch (input or output).

   Note: in order to change transfer direction when operating in a file oriented environment, a new .INIT must first be executed.

2. .OPER (.DLETE)
   (.RENAM)
   (.FSTAT)
   (a) .DLETE
      (1) Examines specified Directory for presence of desired file name. If not found, AC = 0 upon return to user.
      (2) Deletes file name (clears to 0) from the Directory of the specified unit.
      (3) Clears file bit map corresponding to deleted entry.
      (4) Clears corresponding occupancy bits in Directory bit map.
      (5) Records modified Directory and file bit map block on specified unit.

(b) .RENAM

(1) Examines specified Directory for presence of desired file name. If not found, AC = 0 upon return to user.

(2) Changes file name in Directory to new one specified by user program (no change is made to first block pointers).

(3) Records modified Directory on specified unit.

(c) .FSTAT

(1) Examines specified Directory for presence of desired file name. If not found, AC = 0 upon return to user. If found, AC = first block number of file. Also, bits 0 - 2 of CAL ADDRESS + 2 = 1 = DECTAPE Directory type.

(d) Other .OPER codes are illegal.

3. .SEEK

(a) Loads into core the Directory of the unit specified if the Directory is not already in core.

(b) Checks for presence of named file. (Error return to Monitor if not found.)

(c) Begins transfer of first block of file into handler buffer area, overlaying Directory Entry Section but not Directory Bit Map.

(d) Declares unit to be file oriented.

4. .ENTER

(a) Loads into core the Directory of the unit specified if the Directory is not already in core.

(b) Checks for presence of named file. If present, pointer to that entry is saved for update at .CLOSE time. If not present, empty slot is found for file name insertion at .CLOSE time.

(c) Examine Directory Bit Map for free block and saves that block number for first transfer out and for insertion in Directory Entry Section at .CLOSE time.

(d) Declares unit to be file oriented.

5. .CLEAR

(a) Zero's out File Bit Map blocks (3) on specified DECTAPE unit.

(b) Zero's out DECTAPE Directory block.

6. CLOSE

(a) On input, clears internal program switches. On output, writes 2 cell EOF line as last line in output buffer (IOPS ASCII and binary only) and outputs last data buffer with the data link = 777777.

(b) Loads into core the file bit map corresponding to the Directory Entry in order to clear the Directory Bit Map of bits for blocks formerly occupied by this file.

(c) Records newly constructed file bit map.

(d) Loads Directory into memory, enters new entry and records Directory again with new entry and updated Directory Bit Map.

(e) Clears internal program switches.

7. MTAPE (REWIND) (BACKSPACE)

(a) REWIND

(1) Sets internal switches such that data transfer will begin at block 0 in the forward direction.

(2) Declares the unit as non-file oriented, i.e., data will be recorded (starting at block 0) every fourth block. At EOT, recording continues in the reverse direction, etc. Four passes will record all 1100_8 blocks of a DECTAPE.

(b) BACKSPACE

(2) Decrements (by 1) the internal pointer to the next block to be transferred.

(c) Other .MTAPE functions - ignored.

10. .READ

(a) Input line from DECTAPE handler buffer or block of data to user area. (see B below for data modes.)

(b) Initiate input of next DECTAPE block when preceding block has been emptied.

11. .WRITE

(a) Transfers line or block of data from user area to DECTAPE handler buffer.

(b) Outputs buffer when full, examing Directory Bit Map for free block number to store as Data Link (word 377_8) of current block output.

12. .WAIT

Allow previous transfer to be completed before allowing user program to continue.

13. .TRAN

Transfer (in or out) the number of words specified by the user's word count to/from the core area indicated in the .TRAN macro to/from the specific block(s) desired by the user. Data will be transferred to/from contiguous DECTAPE blocks in the forward or reverse direction (also declared by the user). On input, transfer stops on word count overflow. On output, transfer also stops on word count overflow; however, if the word count is not equivalent to an integral number of DECTAPE blocks, the remainder of the last block will be filled with zeros.

B. Legal Data Modes

1. IOPS ASCII
2. IOPS Binary
3. Image Alphanumeric
4. Image Binary
5. Dump

C. Recoverable Errors

1. Select Error* — Monitor Error Code 4
   (a) Ready the desired DECTAPE unit
   (b) Type ↑R on the Teletype.

D. Unrecoverable Errors

1. Illegal Function — Monitor Error Code 6
   (a) See E below for illegal functions
2. Illegal Data Mode — Monitor Error Code 7
   (a) .SEEK with .INIT for output
   (b) .ENTER with .INIT for input.
   (c) See E below for .READ, .WRITE illegal data modes.
3. File Still Active — Monitor Error Code 10
   (a) .SEEK, .ENTER, .CLEAR or .OPER when last file has not been closed.
4. .SEEK, .ENTER Not Executed — Monitor Error Code 11
   (a) .READ or .WRITE executed prior to .SEEK or .ENTER (or .MTAPE - REWIND).
5. DECTAPE Error — Monitor Error Code 12
   (a) Mark Track Error
   (b) EOT during read or write
6. File Not Found — Monitor Error Code 13
   (a) File name not found in Directory on a .SEEK.
7. DECTAPE Directory Full — Monitor Error Code 14
   (a) Directory Entry Section found full (24 files) on an .ENTER

*A "Select" error is equivalent to a hardware not ready condition. See the PDP-9 Users Handbook, F95, 5-12, for a detailed description.

8. DECTAPE Full — Monitor Error Code 15
   (a) All DECTAPE blocks occupied on a .WRITE
9. Output Buffer Overflow — Monitor Error Code 16
   (a) Output line (IOPS ASCII or Binary) greater than 255_{10} cells (including header).
   (b) Output block (Image Binary or Image Alphanumeric) greater than 255_{10} cells (excluding header).
10. Excessive Number of Files Referenced — Monitor Error Code 17
    See Section E for file reference limitations.

E. Subprogram Description and Size

1. DTA (which requires 2100_{10} locations)
   DTA is the most general DECTAPE handler issued with the PDP-9 ADVANCED Software System. DTA has a simultaneous 3-file capacity, either input or output. All files may be referenced on the same or different DECTAPE units. All data modes are handled as well as all IOPS functions. Three 256_{10}-word data buffers, three 32_{10}-word Directory Bit Maps and three 32_{10} word File Bit Maps are included in the body of the handler.

2. DTB (which requires 1664_{10} locations)
   DTB has a simultaneous 2-file capacity, either input or output. Both files may be on the same or different units. DTB transfers data only in IOPS ASCII or IOPS binary data modes. Included in the handler is space for two 256_{10} word data buffers, two 32_{10}-word Directory Bit Maps and two 32_{10}-word File Bit Maps. Functions included are:

   .INIT .ENTER .READ .WAIT
   .SEEK .CLOSE .WRITE

3. DTC (which requires 645_{10} locations)
   DTC is the most limited (and conservative in terms of core allocation) DECTAPE handler in the PDP-9 ADVANCED Software System. DTC is a READ ONLY handler with a 1-file capacity requiring no space for bit maps and only one 256_{10}-word DECTAPE data buffer to handle either IOPS ASCII or IOPS binary input (and no other). Functions included are:

   .INIT .CLOSE .WAIT
   .SEEK .READ

4. DTD (which requires 1400_{10} locations)
   DTD has full IOPS function capabilities; however, it allows for one and only one file reference, either input or output, at any given time. Sequential file references are permitted. All data modes are acceptable to DTD. One 256_{10}-word data buffer, one 32_{10}-word Directory Bit Map and one 32_{10}-word File Bit Map are included.

4.6.6 CD (Card Readers CR01E and CR02B)

A. Functions

1.	.INIT	(a) Return standard buffer size (52_{10}) (b) Call .SETUP to update skip chain with PIC servicer addresses for column ready and card done flags and to place API servicer address in location 55_8 (API channel 13).
2.	.DLETE .RENAM .FSTAT	Ignored
3.	.SEEK	Ignored
4.	.ENTER	Illegal function
5.	.CLEAR	Illegal function
6.	.CLOSE	Allow previously requested input to terminate.
7.	.MTAPE	Ignored
10.	.READ	(a) Allow previously requested input to terminate. (b) Ensure that device is ready. (c) Initiate input of next card.
11.	.WRITE	Illegal function
12.	.WAIT	Allow previously requested input to terminate.
13.	.TRAN	Illegal function

B. LEGAL Data Modes

1. Alphanumeric Input Modes

a. IOPS ASCII (Mode 2) (36_{10} locations required to store an 80-column card)

Eighty card columns are read and interpreted as Hollerith data, mapped into the corresponding 64-graphic subset of ASCII, and stored in the user's line buffer in 5/7 format. Compression of internal blanks to tabs and truncation of trailing blanks is not performed; all 80 characters appearing on the card are delivered to the caller's line buffer. In addition, a carriage return (015_8) character is appended to the input line; a total of 81 ASCII characters are thus returned by the handler in IOPS ASCII mode.

All illegal punch configurations (i.e., those not appearing in the 029 Hollerith set) are represented in the user's line buffer area as null (00) characters. In addition, the parity error bit is set in the line buffer header to indicate the illegal punch condition. There is no possibility of confusion between those nulls representing illegal punch combinations and nulls to pad a word-pair containing fewer than five characters. The reason for this lies in the fact that padding nulls are used only in the last pair of the line, and these are always (in IOPS ASCII mode) preceded by a carriage return. Thus any nulls which appear before the handler-supplied carriage return must be considered to represent illegal punches.

The single addition to the Hollerith set, one made necessary by the constraints of system programs, is the provision for the internal generation of the ALT mode terminator. The appearance of a 12-1-8 punch (multiple-punched A/8) on the card is mapped into the standard PDP-9 ALT mode character (175_8) in the user's line.

When card processing is complete, word 1 of the header is constructed and stored in the caller's line buffer area. Word 2 of the header, the checksum location, is never disturbed by the card reader handler in IOPS ASCII mode.

Attention is called to Appendix 1 ("PDP-9 ASCII-Hollerith Correspondence") for a delineation of legal Hollerith codes and their corresponding ASCII graphics.

b. Image Alphanumeric (Mode 3) (82_{10} locations required to store an 80-column card)

Eighty card columns are read and interpreted as Hollerith data, mapped into the corresponding 64-graphic subset of ASCII, and stored in the user's line buffer area as 80 right-adjusted 7-bit characters, one per word, with leading zero bits. No editing takes place (except in the case of illegal punch combinations), and no terminator is added to the input line.

When an illegal (non-Hollerith) punch combination is encountered on the card being read, the corresponding position in the caller's line buffer is set to contain a null (00) character. In addition, the parity-error bit in the line buffer header is raised to indicate the condition.

2. Binary Input Modes

a. Image Binary (Mode 1) (82_{10} locations required to store an 80-column card)

Eighty card columns are read as 12-bit binary numbers and stored one per word in the caller's line buffer. The column data appears right-adjusted with leading zero bits.

b. IOPS Binary (Mode 0) (52_{10} locations required to store 78 data columns)

Seventy-eight card columns (columns 1-78) are read as 12-bit bytes and stored, 3 bytes in each 2-word group, in the caller's line buffer area. Punches appearing in columns 79 and 80 are ignored in this mode.

Data punched on the card are taken to represent full 18-bit words, each divided into one 6-bit segment and one 12-bit segment. The high-order (leftmost) bit in each column appears in the 12-row of that column; the low-order bit is punched in the 9-row. The organization of words on the card is represented schematically in Figure 4-1.

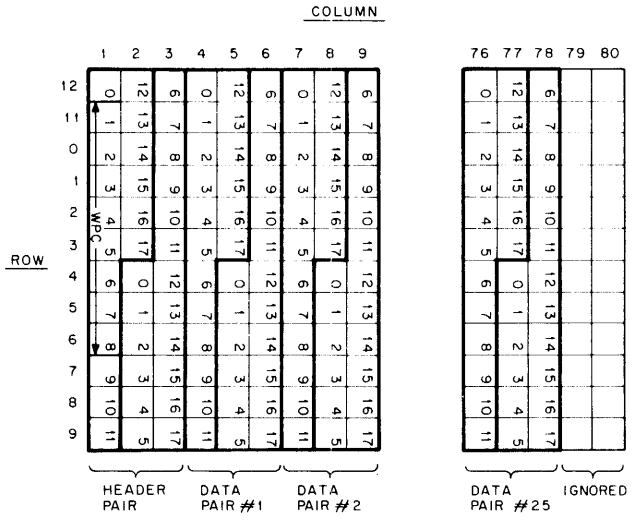


Figure 4-1 IOPS Binary Input Card Format

The line header pair, punched in columns 1, 2, and 3 of the card, consists of a header word (word 1) and a checksum word (word 2).

Header word 1 (column 1; column 2, rows 12-3) includes the following data fields:

Bit 0 (column 1, row 12): Ignore checksum indicator; may be punched.

Bits 1-8 (column 1, rows 11-6): Word Pair Count; must be punched. The handler will halt data transfer upon fulfillment of (1) the word count in the .READ sequence, (2) the word pair count in the card, or (3) 52_{10} words transferred, whichever occurs first.

Bits 9-11 (column 1, rows 7-9): Rows 7 and 9 must be punched.

Bits 12-13 (column 2, rows 12-11): Validity Code. This field is ignored by the handler (except for checksum computation) and the punches appearing in it are not passed on to the caller. The handler sets this field in the user's line buffer as dictated by conditions resulting from the read request.

Bits 14-17 (column 2, rows 0-3): Mode; this field may contain either no punches or punches in rows 0 and 3 to indicate logical end-of-file. If rows 0 and 3 are punched, columns 4-80 are ignored.

Header word 2 (column 2, rows 4-9; column 3) includes only, in bits 0-17, the checksum word for the card. The checksum must be the 2's complement of the 18-bit, unsigned, arithmetic sum of all the data words (column 4-78) on the card and word 1 of the header.

c. Dump (Mode 4) (80_{10} locations required to store an 80-column card)

Identical to Image Binary (2.a), except that no header pair is stored in the line buffer.

C. Unrecoverable Errors

1. Illegal Function (Monitor Error Code 6)
   a. .ENTER
   b. .CLEAR
   c. .WRITE
   d. .TRAN
2. Illegal Data Mode (Monitor Error Code 7)
   a. CDA.: Not applicable; all modes are legal for this version.
   b. CDB,CDC.: IOPS ASCII only is legal for these versions. A request for data transfer in any other mode results in an error return to the Monitor.

D. Program Size

1. CDA. (All modes): Approximately 450_{10} locations.
2. CDB. (IOPS ASCII only): Approximately 360_{10} locations.
3. CDC. (IOPS ASCII only): Approximately 270_{10} locations.

E. Program Descriptions

Both CDA. and CDB. utilize 80-word internal buffers for the temporary storage of column data as it is encountered; remapping in these two handlers is performed after all 80 columns have been read. This scheme guarantees protection against data loss resulting from the service requirements of

other active I/O devices. CDC., on the other hand, remaps each column as it appears, thus doing away with the need for 80 words of temporary storage. There is some, though slight, possibility of data loss in the process, since the column data is presented at fixed time intervals which cannot be program-specified. If data loss does occur during reading, the checksum error indicator is set in the validity field of the header for the line in which loss was detected.

CDC. is designed for use with programs which have large core requirements but relatively low I/O rates (e.g., FORTRAN 4, MACRO).

4.6.7 MT (Magnetic Tape)

Not Yet Implemented

4.6.8 DK (Disk)

Not Yet Implemented

4.6.9 DR (Drum)

Not Yet Implemented

4.6.10 I/O Handlers Acceptable To System Programs

This section lists the ·DAT slot requirements of system programs, the uses made of the ·DAT slots and which I/O handlers may be assigned to each. It is imperative that one and only one I/O handler for a device be in core at the same time; i.e, DTA. and DTB. should not be brought in together since there is no communication between the two interrupt handlers.

System Program	·DAT Slot	Use	Handler
FORTRAN IV	-11	Input	TTA. PRA. PRB. (recommended) DTA. (required if 3 DT files open) DTB. (recommended if 2 DT files open) DTC. (recommended if DT input only) DTD. CDE. CDB. MT DK DR
	-12	Listing	TTA. LPA. PPA.

System Program	.DAT Slot	Use	Handler
			DTA. (required if 3 DT files open) DTB. (recommended if 2 DT files open) DTD. (recommended if DT listing output only) MT DK DR
	-13	Output	PPA. PPB. PPC. (recommended) DTA. (required if 3 DT files open) DTB. (recommended if 2 DT files open) DTD. (recommended if DT output only) MT DK DR

MACRO-9

Identical to FORTRAN IV, with two exceptions (1) if .ABS binary output is requested on .DAT slot -13, PPC. and DTB. cannot be used. (2) .DAT slot -10 is the secondary input device (P option in command string) and should be attached to a non bulk storage handler:

TTA.
PRA.
PRB.
CDE.
CDB.

DDT-9	-6	Output (paper tape only)	PPA. PPB. PPC. (recommended)
	-10	Input (paper tape only)	PRA.

NOTE: As DDT handlers can be serially shared with the user program, choice of the DDT output handler (.DAT slot -6) should be a function of the user paper tape output requirements.

EDIT-9	-10	Secondary Input	TTA. PRA. PRB. (recommended) CDE. CDB.

System Program	.DAT Slot	Use	Handler
	−14	Input	TTA. PRA. PRB. (recommended) DTA. (required if DT input and output) DTD. (DT input only) CDE. CDB. MT DK DR
Linking Loader	−1	System Library	PRA. DTA. DTB. DTC. (recommended if no user DT I/O) DTD. CDE. CDB. MT DK DR
	−4	Input	(same as for .DAT −1)
	−5	External User Library	(same as for .DAT −1)

NOTE: As Linking Loader handlers can be used by the user program, choice of the DECtape handler for the loader should be a function of the user DECtape requirements.

	−15	Output	LPA. TTA. PPA. DTA. (required if DT input and output) DTD. (DT output only) MT DK DR

PIP-9

PIP-9 uses all the positive .DAT slots (1 - 10)

Prior to PIP-9 use, any non-standard peripheral device assignments should be made via the ASSIGN command to the PDP-9 Keyboard Monitor (KM-9). If several PIP functions are to be used with a variety of peripherals, assignment of all these peripherals to Device Assignment Table (DAT) slots (1-10) will avoid returning to KM-9 for reassignment of DAT slots and reloading PIP and its new IOPS routines. into memory. The following should be carefully noted in using the ASSIGN command to set up DAT slots for PIP-9. Since the PDP-9 ADVANCED Software System includes more than one device handler for

certain peripherals, those used with PIP-9 should normally be the ones with the fullest capabilities, e.g., PRA, PPA and DTA. If both input and output are to occur to the same type device (e.g., DECtape), separate slots must be assigned. Both, however, must be assigned to the same handler, i.e., erroneous results will occur if DTA is assigned to one slot and DTB to another, since there is no communication between the interrupt service routines. The user must be certain to clear (ASSIGN NONE A, B, C↵), where A, B, and C are the DAT slots to be cleared), undesirable assignments.

System Program	.DAT Slot	Use	Handler
System Generator	−14 and −10	Input	DTA. (I/O both DT) MT DK DR
	−15	Output	DTA. (I/O both DT) MT DK DR
DUMP-9	−12	Listing	LPA. TTA. PPA.
	−14	Input	DTA. DTD. (recommended) MT DK DR
7-TO-9 CONVERTER	−12	Listing	TTA. LPA. PPA. DTA. (required if 3 DT files open) DTB. (recommended if 2 DT files open) DTD. (recommended if 1 DT file open) MT DK DR
	−14	Input	TTA. PRA. PRB. (recommended) DTA. (required if 3 DT files open) DTB. (recommended if 2 DT files open) DTC. (recommended if DT input only) DTD. CDE. CDB. MT DK DR

System Program	.DAT Slot	Use	Handler
	-15	Output	LPA. TTA. PPA. DTA. (required if 3 DT files open) DTB. (recommended if 2 DT files open) DTD. (recommended if DT output only) MT DK DR
Library Update	-10	Secondary Input	PRA. DTA. CDE. CDB. MT DK DR
	-12	Listing	LPA. TTA. PPA. DTA. MT DK DR
	-14	Input	PRA. DTA. MT DK DR CDE. CDB.
	-15	Output	PPA. PPC. PPB. DTA. MT DK DR

NOTE: DTA. can only handle three files simultaneously. Thus it should not be assigned to all four Library Update .DAT slots.

4.7 DESCRIPTION OF I/O HARDWARE AND API SOFTWARE LEVEL HANDLERS

4.7.1 I/O Device Handlers

All communications between user programs and I/O device handlers are made via CAL instructions (see Section 2.2) followed by pertinent argument lists. The CAL Handler in the MONITOR

(.IOPS), performs preliminary setups, checks on the CAL calling sequence, and then transfers control via a JMP instruction to the entry point of the device handler. When the control transfer occurs, the AC contains the address of the CAL in bits 3-17 and bits 0, 1, and 2 indicate the status of the Link, extend mode and memory protect, respectively, at the time of the CAL. Note that the content of the AC at the time of the CAL is not preserved.

On machines that have an API, the execution of a CAL instruction automatically raises the priority to the highest software level (level 4). Control passes to the handler while it is still at level 4, allowing the handler to complete its reentrant procedures before debreaking (DBK) from level 4. This permits the handler to receive reentrant calls from software levels higher than the priority of this call. If a device handler does not contain reentrant procedures, system failure caused by inadvertent reentries can be prevented by remaining at level 4 until control is returned to the user.

If the non-reentrant method is used, the debreak and restore (DBR) instruction should be executed just prior to the JMP* which returns control to the user, allowing debreak from level 4 and restoring the conditions of the Link, extend mode, and memory protect. Any IOTs issued at the CAL level (level 4 if API present, mainstream if no API) should be executed immediately before the DBR exit sequence to ensure that the exit takes place before the interrupt from the issued IOT occurs.
JMP*

The CAL instruction must not be used at any hardware priority level (API or PIC), since interrupts to these levels are not closed out by the execution of a CAL and recovery is not possible from such sequences of events as

a. An I/O flag coming up during a CAL at level 7,

b. Control going to the I/O device handler at level 3,

c. The handler at level 3 CALing and thus destroying the content of location 00020 for the previous CAL.

The highest API software level (level 4) is also used for processing CALs and extreme care must be taken when executing CALs at this level. For example, a routine that is CALd from level 4 must know that if a debreak (DBR or DBK) is issued control will return to the calling program at a level lower than 4. The calling routine will also debreak; however, this second debreak will not be from level 4 but from the next highest active level. Much confusion can arise as to which level you are on.

4.7.1.1 Setting Up the Skip Chain and API (Hardware) Channel Registers - When the Monitor is loaded, the program interrupt control (PIC) skip chain and the automatic priority interrupt (API) channels are set up to handle the Teletype keyboard, teleprinter and clock interrupts, only. The skip chain contains the other skip IOT instructions, but indirect jumps to an error routine result if a skip occurs, as follows:

```
SKP DTA          /Skip if DECtape flag.
SKP
JMP*INT1         /INT1 contains error address.
```

```
        SKP LPT                 /Skip if line printer flag.
        SKP
        JMP*INT2                /INT2 contains error address.
        SKP TTI                 /Skip if Teletype flag.
        SKP
        JMP TELINT              /To Teletype interrupt handler.
          :
          :
```

All unused API channels also contain JMP's to the error address.

When a device handler is called for the first time via an .INIT user program command, it must call a Monitor routine (.SETUP) to set up its skip chain entry or entries and API channel, prior to performing any I/O functions. The calling sequence is as follows.

```
        CAL  N                  /N = API channel register 40-77, (see section
                                /4.7.3 for standard channel assignments), 0 if de-
                                /vice not connected to API.
        16                      /.SETUP function code.
        SKP IOT                 /Skip IOT for this device.
        DEVINT                  /Address of interrupt handler.
        (normal return)
```

DEVINT exists in the device handler in the following format.

```
DEVPIC  DAC     DEVAC           /SAVE AC.
        LAC*    (0
        DAC     DEVOUT          /SAVE PC, LINK, EX.MODE, MEM.PROT.
        LAC     DEVION          /FORCE ION AT DISMISSAL.
        JMP     DVSTON
DEVINT  JMP     DEVPIC          /PIC ENTRY.
        DAC     DEVAC           /API ENTRY, SAVE AC.
        LAC     DEVINT
        DAC     DEVOUT          /SAVE PC, LINK, EX.MODE, MEM.PROT.
        IORS                    /CHECK STATUS OF PIC
        SMA!CLA                 /FOR RESTORATION AT DISMISSAL.
        LAW     17740           /PIC OFF, BUILD IOF IOT.
        TAD     DEVION          /PIC ON, BUILD ION IOT.
DVSTON  DAC     DVSWCH
        DEVCF                   /CLEAR DEVICE DONE FLAG
DEVION  ION                     /ENABLE PIC SO THAT
                                /OTHER DEVICES AREN'T SHUT
        .                       /OUT.
        .
        .

        IOF                     /DISABLE PIC TO INSURE
        DEVIOT                  /DISMISSAL BEFORE INTERRUPT
        .                       /FROM THIS IOT OCCURS
        .
        .
/DISMISS ROUTINE.
        LAC     (JMP DEVPIC     /RESTORE DEVINT IN
        DAC     DEVINT          /CASE API DISABLED.
        LAC     DEVAC           /RESTORE AC
```

```
DVSWCH    ION                    /ION OR IOF
          DBR                    /DEBREAK AND RESTORE CONDITIONS
          JMP*   DEVOUT          /OF LINK,EX.MODE AND MEM.PROT.
```

Since the auto-index registers and EAE registers are not used by the standard I/O device handlers, it is not necessary to save and restore them.

The Monitor routine (.SETUP) checks the skip chain for the instruction which matches SKP IOT, if there is a match it places the address, DEVINT, in the appropriate transfer vector (INTn) and places JMS* INTn in the corresponding API channel register. If a match cannot be found, .IOPS outputs the following error message,

```
.IOPS   05   XXXXXX
```

indicating that the skip IOT in the CAL calling sequence at location XXXXXX was not in the skip chain.

Refer to the operating procedures of the System Generator for the method of incorporating new handlers and associated skip chain entries into the Monitor.

4.7.2 API Software Level Handlers

(This section assumes complete familiarity with chapter 9 of the PDP-9 User Handbook.)

4.7.2.1 Setting Up API Software Level Channel Registers - When the Monitor is loaded, the API software-level channel registers (40-43) are initialized to

```
JMS*    .SCOM+12        /LEVEL 4
JMS*    .SCOM+13        /LEVEL 5
JMS*    .SCOM+14        /LEVEL 6
JMS*    .SCOM+15        /LEVEL 7
```

where the .SCOM registers are at absolute locations 00112 through 00115 and contain the address of an error routine.

Therefore, prior to requesting any interrupts at these software priority levels, the user must modify the contents of the .SCOM registers so that they point to the entry point of the user's software level handlers.

EXAMPLE:

```
.SCOM = 100
        LAC     (LV5INT
        DAC*    (.SCOM+13
          :
          :
```

LV5INT exists in the user's area in the following format:

```
LV5INT      0                       /PC, LINK, EX.MODE, MEM.PROT.
            DAC     SAV5AC          /SAVE AC
            /SAVE AUTO INDEX REGISTERS
            /IF LEVEL 5 ROUTINES
            /USE THEM AND LOWER LEVEL
            /ROUTINES ALSO USE THEM.
```

```
/SAVE MQ AND STEP COUNTER
/IF SYSTEM HAS EAE AND IT
/IS USED AT DIFFERENT LEVELS.
        :
        :
/RESTORE SAVED REGISTERS.
DBR                         /DEBREAK FROM LEVEL 5
JMP*    LV5INT              /AND RESTORE L, EX.MODE, MEM.PROT.
```

4.7.2.2 Queueing - High priority/high data rate/short access routines cannot perform complex calculations based on unusual conditions without holding off further data inputs. To perform the calculations, the high priority program segment must initiate a lower priority (interruptable) segment to perform the calculations. Since, in general, many data handling routines will be requesting calculations, there will exist a queue of calculation jobs waiting to be performed at the software level. Each data handling routine must add its job request to the appropriate queue (taking care to raise the API priority level as high as the highest level that manipulates the queue before adding the request) and issue an interrupt request (ISA) at the corresponding software priority level. The general flow chart, Figure 4-2, depicts the structure of a software level handler involved with queued requests.

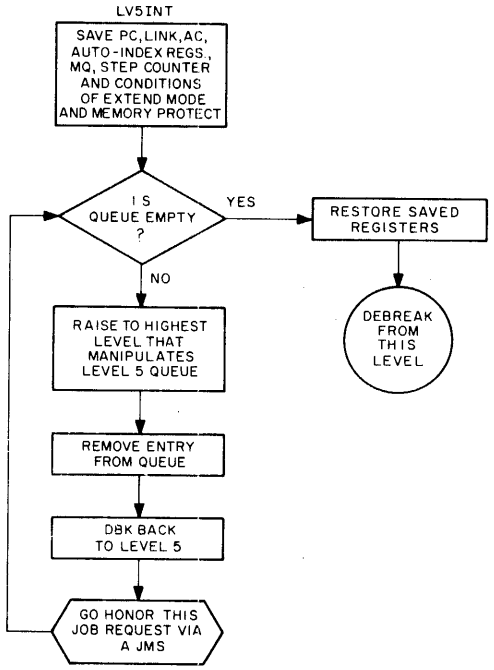


Figure 4-2 Structure of API Software Level Handler

Care must be taken about which routines are called when a software level request is honored; that is, if a called routine is "open" (started but not completed) at a lower level, it must be reentrant or errors will result.

NOTE: The standard hardware I/O device handlers do not contain reentrant procedures and must not be reentered from higher software levels.

4.7.3 Standard API Channel/Priority Assignments

Channel	Device	Option Number	Priority	Channel Register
0	Software priority	--	4	40
1	Software priority	--	5	41
2	Software priority	--	6	42
3	Software priority	--	7	43
4	DECtape	TC02	1	44
5	MAGtape	TC59	1	45
6	Drum	RM09	1	46
7	Disk	--	1	47
8	Paper Tape Reader	--	2	50
9	Clock overflow	--	3	51
10	Power fail	KP09	0	52
11	Parity	MP09	0	53
12	Display (L. P. flag)	34H	2	54
13	Card readers	CR01E CR02B	2 2	55
14	Line Printer	647	2	56
15	A/D	138/139	0	57
16	DB99A/DB98A	DB09A	3	60
17	360 Data Link		3	61

Channels 18-31 still unassigned.

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively, we need user feedback: your critical evaluation of this manual and the DEC products described.

Please comment on this publication. For example, in your judgment, is it complete, accurate, well-organized, well-written, usable, etc?

Did you find this manual easy to use?

What is the most serious fault in this manual?

What single feature did you like best in this manual?

Did you find errors in this manual? Please describe.

Please describe your position.

Name Organization

Street State Zip

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Quality Control
Building 12
146 Main Street
Maynard, Mass. 01754

APPENDIX 1

PDP-9 ASCII/HOLLERITH CORRESPONDENCE

	*00-37	40-77		100-137		*140-177	
	ASCII CHAR.	ASCII CHAR.	HOLLERITH CHAR.	ASCII CHAR.	HOLLERITH CHAR.	ASCII CHAR.	
0	NUL	SP		\	@ 4-8	@	0
1	SOH	!	11-2-8	A	12-1	a	1
2	STX	"	7-8	B	12-2	b	2
3	ETX	#	3-8	C	12-3	c	3
4	EOT	$	11-3-8	D	12-4	d	4
5	ENQ	%	0-4-8	E	12-5	e	5
6	ACK	&	12	F	12-6	f	6
7	BELL	'	5-8	G	12-7	g	7
10	BS	(	12-5-8	H	12-8	h	10
11	HT	)	11-5-8	I	12-9	i	11
12	LF	*	11-4-8	J	11-1	j	12
13	VT	+	12-6-8	K	11-2	k	13
14	FF	,	0-3-8	L	11-3	l	14
15	CR	-	11	M	11-4	m	15
16	SO	.	12-3-8	N	11-5	n	16
17	SI	/	0-1	O	11-6	o	17
20	DLE	0	0	P	11-7	p	20
21	DC1	1	1	Q	11-8	q	21
22	DC2	2	2	R	11-9	r	22
23	DC3	3	3	S	0-2	s	23
24	DC4	4	4	T	0-3	t	24
25	NACK	5	5	U	0-4	u	25
26	SYNC	6	6	V	0-5	v	26
27	ETB	7	7	W	0-6	w	27
30	CNCL	8	8	X	0-7	x	30
31	EM	9	9	Y	0-8	y	31
32	SS	:	2-8	Z	0-9	z	32

*ASCII code 0-37 and 140-177 have no corresponding codes in the Hollerith set.

	*00-37	40-77		100-137		*140-177	
	ASCII CHAR.	ASCII CHAR.	HOLLERITH CHAR.	ASCII CHAR.	HOLLERITH CHAR.	ASCII CHAR.	
33	ESC	;	11-6-8	[	¢ 12-2-8	{	33
34	FS	<	12-4-8	⌒	11-7-8	¬	34
35	CS	=	6-8	]	± 0-2-8	}	35
36	RS	>	0-6-8	^	12-7-8	|	36
37	US	?	0-7-8	_(under-score)	0-5-8	delete	37

*ASCII code 0-37 and 140-177 have no corresponding codes in the Hollerith set.

APPENDIX 2

PDP-9 ASCII CHARACTER SET

Listed below are the ASCII characters interpreted by the PDP-9 Keyboard Monitor and system programs as meaningful data input or as control characters.

	00-37	40-77	100-137	140-177	
	ASCII CHAR.	ASCII CHAR.	ASCII CHAR.	ASCII CHAR.	
0	NUL	SP	\		0
1	SOH (↑A)	!	A		1
2		"	B		2
3	ETX (↑C)	#	C		3
4		$	D		4
5		%	E		5
6		&	F		6
7		'	G		7
10		(	H		10
11	HT	)	I		11
12	LF	*	J		12
13	VT	+	K		13
14	FF	,	L		14
15	CR	-	M		15
16		.	N		16
17	SI (↑O)	/	O		17
20	DLE (↑P)	0	P		20
21		1	Q		21
22	DC2 (↑R)	2	R		22
23	DC3 (↑S)	3	S		23
24	DC4 (↑T)	4	T		24
25	NACK (↑U)	5	U		25
26		6	V		26
27		7	W		27
30	CNCL (↑X)	8	X		30
31		9	Y		31
32	SS (↑Z)	:	Z		32

	00-37	40-77	100-137	140-177	
	ASCII CHAR.	ASCII CHAR.	ASCII CHAR.	ASCII CHAR.	
*33	ESC	;		ESC	33
34		<			34
35		=		ESC	35
36	RS (↑)	>	^ or ↑		36
37		?		delete (RO)	37

*Codes 33, 176, 175 are interpreted as ESC (ALT Mode) and are converted on input to code 175 by IOPS handlers.

digital

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Printed in U.S.A.