Digital Equipment Corporation
Maynard, Massachusetts

digital

PDP-9

ADVANCED SOFTWARE SYSTEM MONITORS

ADVANCED SOFTWARE SYSTEM MONITORS

For additional copies order No. DEC-9A-MAD0-D from Program Library,
Digital Equipment Corporation, Maynard, Massachusetts. Price $5.00

DIGITAL EQUIPMENT CORPORATION □ MAYNARD, MASSACHUSETTS

Printed in U.S.A.

1st Printing March 1967
2nd Printing Revised February 1968
3rd Printing Revised May 1968
4th Printing Revised January 1968

Instruction times, operating speeds and the like are included in this manual for reference only; they are not to be taken as specifications.

CONTENTS

CONTENTS (cont)

CONTENTS (cont)

CONTENTS (cont)

ILLUSTRATIONS

TABLES

TABLES (cont)

PREFACE

This manual contains information required to prepare programs for operation under control of PDP-9 ADVANCED Software System Monitors. The manual is organized as follows:

The I/O Monitor (Chapter 4) is used with paper tape systems; the more sophisticated Keyboard Monitor (Chapter 5) is used with bulk storage systems; and the Background/Foreground Monitor (Chapter 6) is used with time-shared and real-time systems. Upward compatibility exists between the monitor systems. The Keyboard Monitor contains all the features of the I/O Monitor and also provides for Teletype keyboard commands. The Background/Foreground Monitor is an extension of the Keyboard Monitor and provides for concurrent BACKGROUND and FOREGROUND processing.

An I/O Monitor Guide and a Keyboard Monitor Guide, prepared especially for convenient use at the computer console, are now available (Order Numbers DEC-9A-MIPA-D and DEC-9A-MKFA-D, respectively). These console manuals summarize the essential information required to operate the I/O and Keyboard Monitors, and include detailed operating procedures for each of the system programs.

It should be noted that all material presented in this manual for the Background/Foreground Monitor is preliminary and subject to change.

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.

Three monitor systems are available with PDP-9 ADVANCED software. The Input/Output Monitor operates on a basic PDP-9 with 8192 (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.

The more sophisticated Keyboard Monitor is available for systems with auxiliary bulk storage units. It allows for device-independent programming and automatic creation, calling, and loading of programs. Since upward compatibility exists between the Input/Output Monitor and the Keyboard Monitor, all programs prepared for the I/O Monitor can also be run using the Keyboard Monitor.

The Background/Foreground Monitor is an extension of the Keyboard Monitor. It allows for the concurrent, time-shared use of a PDP-9 system by a protected FOREGROUND user program and an unprotected BACKGROUND system or user program. This provides the user with optimum utilization of system hardware and processing time.

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. 8192 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)

For extra memory, the Type KE09A Extended Arithmetic Element and the Type KF09A Automatic Priority Interrupt can also be used with this system. The Type CR02B Card Reader can be used for input in addition to the paper tape reader; a Type 647 Line Printer can be used for listings.

Some form of bulk storage must be added to the basic PDP-9 to use the Keyboard Monitor:

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

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

c. Type RC09 Fixed-Head Disk System

Input/output routines are provided for these devices as required. In addition, this system can take full advantage of extra memory, central processor options, and additional I/O options.

The Background/Foreground Monitor requires all the options itemized for the Keyboard Monitor plus the following:

a. Type LT09A** Multi-station Teletype Control, and one Type LT09B Line Unit for each external Teletype.

b. Type KSR 33 Teletype

c. Type KX09A Memory Protection Option

d. Type KG09A Memory Extension Control

e. Type MM09A Memory Module (8K)

1.3 MONITOR SYSTEMS

PDP-9 monitor systems simplify the handling of input/output functions and 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 and Background/Foreground Monitors) device-independent programming, 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. The Background/Foreground Monitor effectively provides the user with two systems; on-line data acquisition and control can be performed in the FOREGROUND while user program compilation, debugging, etc., can be accomplished in the BACKGROUND environment.

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 provides for simultaneous I/O and computation.

---

* Background/Foreground systems cannot use magnetic tape as a system device.

**If the API option is available, a Type LT19A Teletype Control and a Type LT19B Line Unit are required instead of the LT09A and LT09B.

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 Program System.

c. The CAL Handler, which is used to dispatch to the appropriate Monitor and I/O subroutines.

d. Device handlers for the Teletype and clock.

The I/O Monitor resides in lower core and occupies about 960_{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 Teletype 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 true device independence to the user. Programs may be modified simply and quickly 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. Programs can be created, stored, retrieved, loaded, debugged, and operated at the keyboard console. Both system and user programs can be called from the bulk storage device with a few simple keyboard commands. The Keyboard Monitor also has a batch processing capability that allows user commands to come from the paper tape reader (or card reader) instead of the Teletype, permitting many programs to be run without operator intervention.

1.3.3 Background/Foreground Monitor

The Background/Foreground Monitor is designed to control processing and I/O operations in a real-time or time-shared environment. FOREGROUND programs are defined as the higher priority, debugged programs that interface with the real-time environment. They normally operate under Program Interrupt (PI) or Automatic Priority Interrupt (API) control. At load time they have top priority in selection of core memory and I/O devices, and at execution time they have priority (according to the assigned priority levels) over processing time and use of shared I/O.

BACKGROUND processing is essentially the same as processing normally performed under control of the Keyboard Monitor. That is, it could be an assembly, compilation, debugging run,

production run, editing task, etc. BACKGROUND programs may use any facilities (core, I/O, or processing time) that are available and not simultaneously required by the FOREGROUND job.

1.4 INPUT/OUTPUT PROGRAMMING SYSTEM (IOPS)

The Input/Output Programming System (IOPS) consists of an I/O control routine and individual hardware device handling subroutines that process file and data level commands to the devices. These handlers exist for all standard PDP-9 peripherals (see Section 7.4).

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 which they are incapable of handling (for example, trying to rewind a card reader, or skipping files on a non-file-oriented device). All device handlers operate either with or without the Automatic Priority Interrupt (API) option.

1.5 SYSTEM PROGRAMS

PDP-9 ADVANCED software systems include either the Input/Output Monitor, the Keyboard Monitor, or the Background/Foreground Monitor in addition to an Input/Output Programming System and the following system programs:

FORTRAN IV Compiler, Object Time System, and Science Library
MACRO-9 Assembler
Dynamic Debugging Technique (DDT) Program
Text Editor Program
Peripheral Interchange Program
Linking Loader
PDP-7 to MACRO-9 Assembly Language Converter
Chain Builder Program
Chain Execute Program
System Generator
Dump Program
Library Update Program
System Patch Program

} With Keyboard and Background/Foreground Monitor systems only

The following special-purpose utility programs are also available:

Disk/DECtape Save
Disk/Paper Tape Save

} for Disk users

PUNCH9 - for paper tape systems

1.5.1 FORTRAN IV Compiler

The PDP-9 FORTRAN IV compiler is a two-pass system that accepts statements written in the FORTRAN IV language and produces a relocatable object program 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 8192 words of core storage:

a. Complex arithmetic is not legal.

b. Adjustable array dimensions are not allowed at source level, but may be implemented by calling dimension-adjustment subroutines.

c. Blank Common is treated as named Common except when object program is used in chaining.

d. The implied DO feature is not included for the DATA statement.

e. Specification statements must be strictly positioned and ordered.

The FORTRAN IV compiler operates with the PDP-9 program interrupt or API facilities enabled. It generates programs that operate with the program interrupt or API 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.

There are two versions of the FORTRAN IV compiler; F4 is the basic compiler, and F4A is an abbreviated version that allows for DECtape I/O in an 8K system. F4A does not provide for object code listing, symbol table listing, EQUIVALENCE statements, ASSIGN statements, assigned GOTO statements, or EXTERNAL statements.

The PDP-9 FORTRAN IV Compiler, Object Time System, and Science Library are described fully in the FORTRAN IV Manual (DEC-9A-KFZA-D).

1.5.2 MACRO-9 Assembler

The MACRO-9 Assembler provides PDP-9 users with highly sophisticated macro generating and calling facilities within the context of a symbolic assembler. MACRO-9 is described in detail in the MACRO-9 Assembler Manual (DEC-9A-AMZA-D). Some of the prominent features of MACRO-9 include:

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.

i. Ability to call input/output system macros that expand into IOPS calling sequences.

A shorter version of the assembler (MACRO A) is available to enable users with 8K systems to use DECtape for input and output. Conditional pseudo-ops and .ABS, .FULL, .REPT, and .DEFIN are not allowed.

1.5.3 Dynamic Debugging Technique (DDT) Program

DDT provides on-line debugging facilities within the PDP-9 ADVANCED software system, enabling the user to load and operate his program in a real-time environment while maintaining strict control over the running of each section. DDT 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, 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 or all of an object program with a 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 Program Manual (DEC-9A-GUAB-D).

1.5.4 Text Editor Program

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, 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) or ALT MODE 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 on the Text Editor, refer to the PDP-9 Utility Programs Manual (DEC-9A-GUAB-D).

1.5.5 Peripheral Interchange Program (PIP)

The primary function of PIP is to facilitate the manipulation and transfer of data files from any input device to any output device. It can be used to refresh mass storage file directories, list file directory contents, delete, insert, segment, or combine files, perform code conversions, and copy tapes.

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

1.5.6 Linking Loader

The Linking Loader loads any PDP-9 FORTRAN IV or MACRO-9 object program which exists in relocatable format (or absolute format if pseudo-ops .ABS and .FULL are not used). Its tasks include loading and relocation of programs, loading of called subroutines, retrieval and loading of implied subroutines, and building and relocation of the necessary symbol tables. Its operation is discussed in the PDP-9 Utility Program Manual (DEC-9A-GUAB-D).

1.5.7 PDP-7 to MACRO-9 Assembly Language Converter

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

1.5.8 System Generator

The System Generator (SGEN) 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 and details of its use, are given in the Keyboard Monitor Guide (DEC-9A-MKFA-D).

1.5.9 Dump Program

This system program gives the user the ability to output on any listing device, specified core locations that have been preserved on a bulk storage file via the CTRL Q Keyboard Monitor dump command. A more complete description of the Dump program is given in the Keyboard Monitor Guide (DEC-9A-MKFA-D).

1.5.10 Library Update Program

This system program gives the user the capability to examine and update the binary library files on file-oriented devices. A more complete description of the Library Update program is given in the Keyboard Monitor Guide (DEC-9A-MKFA-D).

1.5.11 System Patch Program

The System Patch program is used to make corrections to the binary version of non-relocatable system programs on the system device. A more complete description of the System Patch program and details of its use, are given in the Keyboard Monitor Guide (DEC-9A-MKFA-D).

1.5.12 Chain Builder and Execute Programs

The Chain Builder and Execute programs provide the user with a capability for program segmentation which allows for multiple core overlap of executable code and certain types of data areas. A more complete description of the Chain Builder and Execute programs is given in the Keyboard Monitor Guide (DEC-9A-MKFA-D).

CHAPTER 2
THE PDP-9 MONITOR ENVIRONMENT

2.1 MONITOR FUNCTIONS

PDP-9 ADVANCED Software System Monitors greatly simplify the task of programming I/O functions by providing an interface between system or user programs and the external world of I/O devices. Upward compatibility exists between the Monitor systems; programs written to operate under control of the I/O Monitor will also operate, without modification, under control of the Keyboard and Background/Foreground Monitors. The Monitors, by means of the Input/Output Programming System (IOPS) and Program Interrupt (PI) or Automatic Priority Interrupt (API), allow simultaneous operation of multiple I/O devices along with overlapping computations.

Certain features such as the general monitor environment, data handling, and logical/physical I/O device associations, are common to all three monitors. These features are discussed at length in this chapter. It is recommended that the reader become thoroughly familiar with the contents of this chapter before reading chapters that apply to each of the monitors.

2.1.1 General I/O Communication

The general communication required to accomplish an I/O task is the same for all three monitor systems (see Figure 2-1). A system or user program initiates an I/O function by means of a monitor command (system macro), which is interpreted by a CAL handler within the monitor as a legitimate I/O call. (See the PDP-9 User Handbook for a description of the CAL instruction.) The I/O call includes a logical I/O device number as one of its arguments. The monitor establishes the logical/physical I/O device association by means of its Device Assignment Table (.DAT). When this has been accomplished, the monitor passes control to the appropriate device handler subroutine to initiate the I/O function and return control to the system or user program. The system or user program retains control until an interrupt (PI or API) occurs, at which time it relinquishes control to the device handler to perform and/or complete the specified I/O function. Computations or other processing can be performed by the system or user program while waiting for an interrupt. This feature allows the programmer to make optimum use of available time.

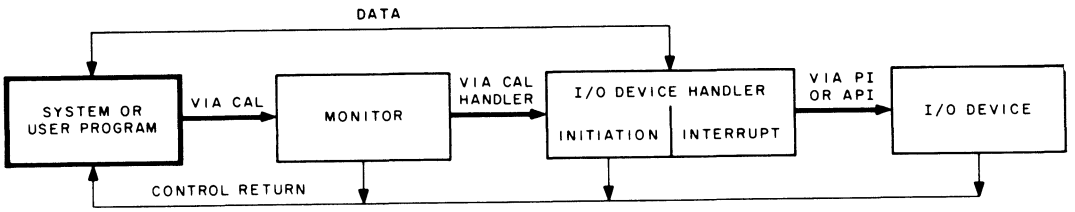


Figure 2-1 General I/O Communication in Monitor Environment

2.1.2 Command, Control, and Data Flow

Figure 2-2 provides a more detailed representation of the monitor environment, with emphasis on command, control, and data flow. As shown, the user can initiate a command via the Teletype. In the I/O Monitor environment, this command can be interpreted only by a Command Processor within the system program (or user program if so designed). In the Keyboard and Background/Foreground Monitor environments, an expanded set of keyboard commands can also be interpreted by a Keyboard Listener (.KLIST) and acted upon by a Monitor Command Decoder (.MCD). This feature greatly extends the capabilities of the monitors and provides the user with a large repertoire of keyboard commands. The monitor shown in Figure 2-2 can represent any one of the monitor systems, except that the I/O Monitor does not contain the .KLIST and .MCD programs to interpret and act upon Teletype keyboard commands. The .KLIST and .MCD programs are nonresident in the sense that they are overlaid by user and system programs.

Each system or user program must internally set up line buffers (except when using Dump mode, discussed later) to be used in transmitting data to or from the external environment. Each line buffer of n words consists of a two-word header (referred to as a header word pair) and n -2 words of data. The system or user program can exercise control on output by modifying the header word pair, or it can verify on input by examining the header word pair. The use of line buffers is discussed in more detail later in this chapter.

Monitor I/O commands (system macros) are written as part of the system or user program. In FORTRAN IV source programs, these commands are in the form of READ and WRITE statements (refer to the FORTRAN IV Manual, DEC-9A-AF4B-D). These statements are translated by the compiler into the proper calling sequences for the FORTRAN Object Time System which provides the required monitor calls at execution time. In MACRO-9 source programs, monitor I/O commands are written as system

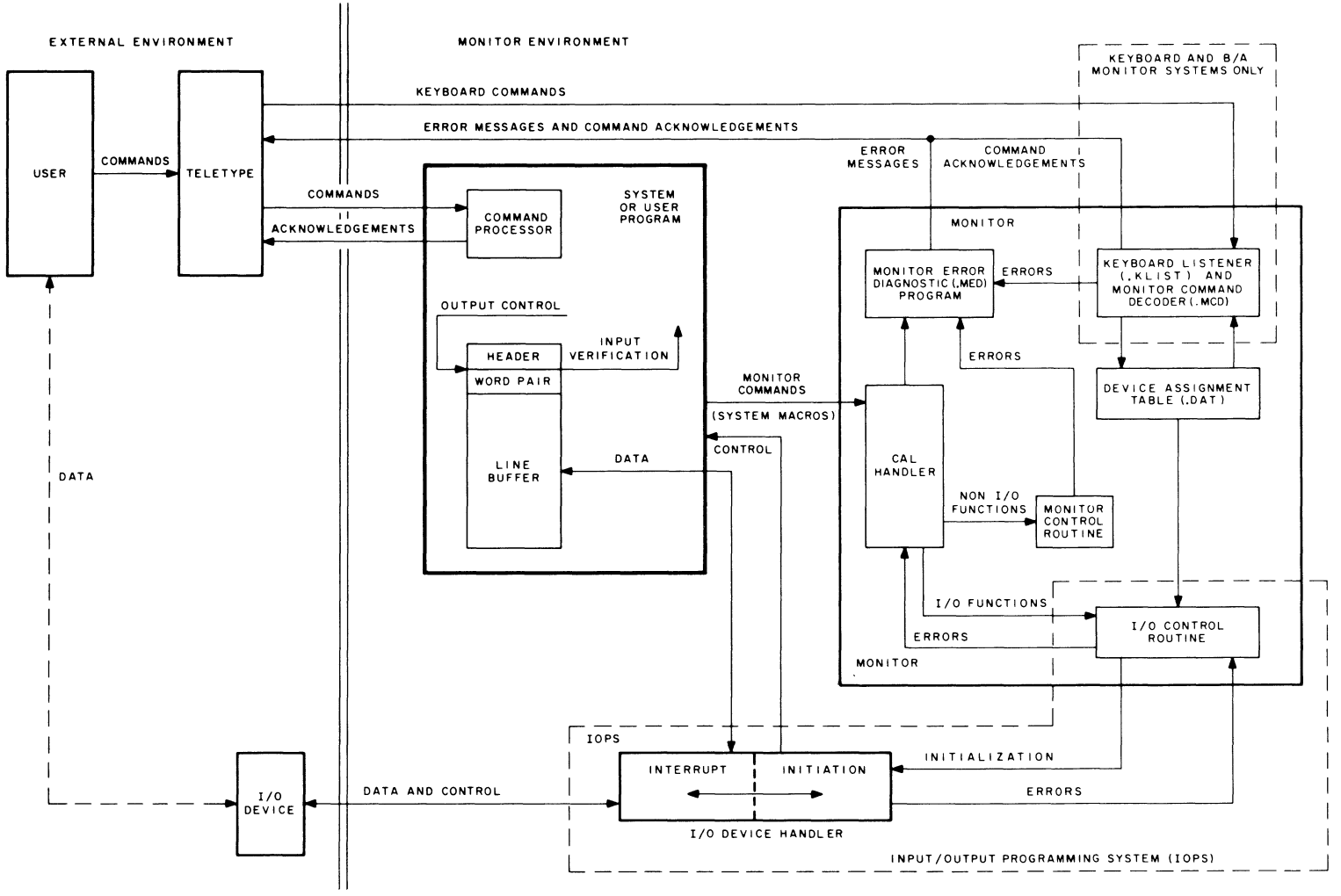


Figure 2-2 Command, Control, and Data Flow in Monitor Environment

macros within the system or user program. These system macros are expanded at assembly time and include a CAL initiated monitor call that contains the logical device number as one of the arguments.

At execution time, monitor calls are processed by the CAL Handler within the monitor. Non-I/O functions are then further processed by the Monitor Control routine, and I/O functions are processed by the I/O Control routine (see Figures 2-2 and 2-3). A complete description of each of these commands is given in Chapter 3. If the original command involved is an I/O function, the I/O control routine checks the Device Assignment Table to associate the logical I/O device (specified by the system macro) to a physical I/O device. In the I/O Monitor environment, the logical/physical device associations can be modified only by reassembly. In the Keyboard and Background/Foreground Monitor environments, device associations can be modified at System Generation time, or by means of the ASSIGN keyboard command just prior to loading a system or user program. This capability adds true device independence to the monitor systems.

	Function Code	Command
Functions processed by I/O control routine	1	.INIT
	2	.DLETE, .RENAM, and .FSTAT
	3	.SEEK
	4	.ENTER
	5	.CLEAR
	6	.CLOSE
	7	.MTAPE
	10	.READ and .REALR
	11	.WRITE and .REALW
	12	.WAIT and .WAITR
	13	.TRAN
Functions processed by monitor control routine	14	.TIMER
	15	.EXIT
	16	.SETUP
	17	.IDLE

Figure 2-3 Monitor Commands and Function Codes

NOTE

.INIT, .READ, .WRITE, .WAIT, .WAITR, .CLOSE, .TIMER, and .EXIT are recognized by all three monitors.

.SEEK, .ENTER, .FSTAT, .RENAM, .DLETE, .TRAN, .CLEAR, and .MTAPE are recognized by the Keyboard and Background/Foreground Monitors (and are ignored by the I/O Monitor).

.SETUP is used by the Monitors in setting up the I/O skip chain and API channel registers (see Chapter 7).

.IDLE is recognized by the Background/Foreground Monitor only.

When the logical/physical I/O device association has been established, the monitor passes control to the appropriate I/O device handler which initializes itself, initiates I/O, and returns control to the system or user program. As mentioned previously, the system or user program retains control until the specified device causes an interrupt (PI or API). At this point, it relinquishes control to the device handler to continue or complete the specified I/O operation. In either case, control is returned to the system or user program at the point where it was interrupted. The system or user program, by means of a .WAIT system macro (described in Chapter 3), can determine whether an input or output operation has been completed. If the transfer of data from or to the system or user program line buffer has been completed, program execution continues; if the transfer has not been completed, control is returned to the .WAIT macro.

Additional buffering is provided by the individual device handlers as required. All device handlers are non-resident in the sense that only those handlers required by the system or user program are loaded into core.

2.2 LINE BUFFERS

As mentioned in the preceeding general description of the monitor environment, each system or user program must internally set up line buffers to be used in transmitting data to or from the external environment. An exception to this rule is when data is transmitted in the Dump mode (described in Section 2.3.3). Each line buffer of n words (always even) should be set up to consist of a two-word header (termed a header word pair) followed by n-2 words of data as shown in Figure 2-4.

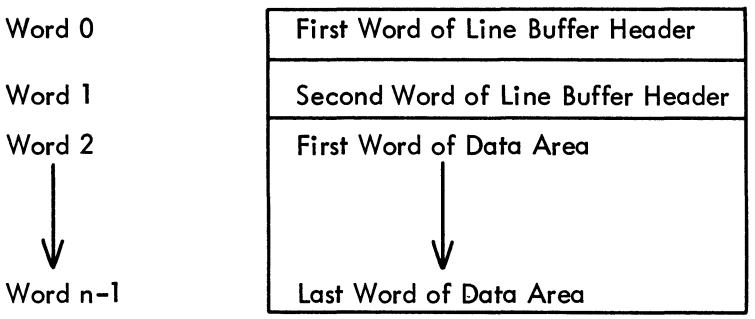


Figure 2-4 Line Buffer Structure

A system or user program should contain at least one line buffer for each device that is to be used. This buffer is used to set up output lines before transmittal to an output device, or to receive input lines from the associated input device. The monitor accepts commands (system macros) from system or user programs to initiate input to the line buffers and to write out the contents of line buffers. Complete descriptions of these commands are given in Chapter 3. Line buffers are internal to, and must be defined by, each system or user program. The header word pair within a line buffer is detailed in Figure 2-5. The .BLOCK pseudo operation may be used to reserve space for a line buffer. A tag is required to allow referencing by indivdual .READ and .WRITE macros. For example:

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

Before output, the user must set the appropriate word pair count in bits 1 through 8 of word zero in the line buffer if they have not already been set by a device handler on input. This count overrides the word count passed to IOPS by the .WRITE macro. (The word count must still be specified in the .WRITE macro for each data mode; however, it only has meaning in Dump mode since there is no header word pair.) In IOPS binary mode (discussed in Paragraph 2.3.1.2), bits 9 through 11 should be set to 101 if the output will ultimately be on cards. The checksum word, the second word in the header, need not be set by the user since checksums are computed by IOPS.

Before input, the user should not be concerned with the header word pair since they will be set by IOPS to enable the user to determine what has happened after input has terminated.

On input, the word count specified in the .READ macro is used by IOPS to determine the maximum number of locations to be occupied by the data being read. If the word count is exceeded before input is terminated, or if there is a parity or checksum error, IOPS sets the appropriate validity bits in header word 0 to indicate the error.

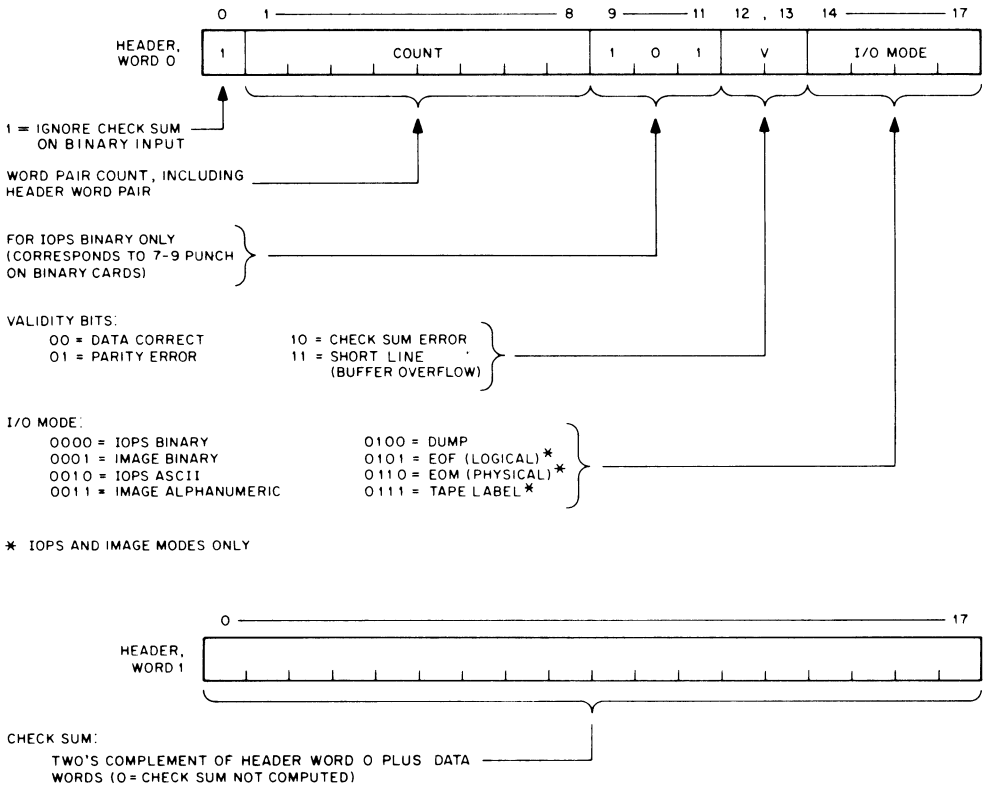


Figure 2-5 Format of Header Word Pair

After input, the user should check the validity bits in word 0 of the line buffer header to determine if the data was read without error. If multiple errors are detected, priority is given to a parity error over a checksum error. IOPS ignores checksum errors on binary input if bit 0 of word 0 of the line buffer header is set to 1. IOPS sets the I/O mode bits (bits 14 through 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), or 5(0101_2) if it senses a logical end-of-file during an IOPS binary read.

When choosing a word count (that is, the maximum line buffer size) to specify in system macros, both the set of possible devices and the mode of data transmission must be considered. The maximum line buffer sizes (including 2-word header) for standard peripheral devices, along with applicable data modes, are listed in Table 2-1.

Table 2-1
Maximum Line Buffer Sizes

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, H
PP (paper tape punch)	52_{10}	All	34_{10} sufficient if A mode only. Headers output for B only.
TT (Teletype)	34_{10}	A,H 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, H modes. Headers accepted for B; generated for A, I, H.
LP (line printer)	52_{10}	A only	Allows for 125_{10} characters. No headers output.
DT0-7 (DECtape)	255_{10}	All	IOPS and image modes allow for several line buffers (logical records) per physical block.
MT0-7(magnetic tape)	255_{10}	All	
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
H = Image Alphanumeric

2.3 DATA MODES

The Input/Output Programming System allows data transmission to or from a system or user program in five different modes.

Mode	Code *
IOPS Binary	0
Image Binary	1
IOPS ASCII	2
Image Alphanumeric	3
Dump	4

* Bits 14 through 17 of Header Word 0, specified by system macro and set by IOPS.

2.3.1 IOPS Modes

The two IOPS data modes include IOPS ASCII and IOPS binary as shown in Figure 2-6 on paper tape, and described in the following paragraphs.

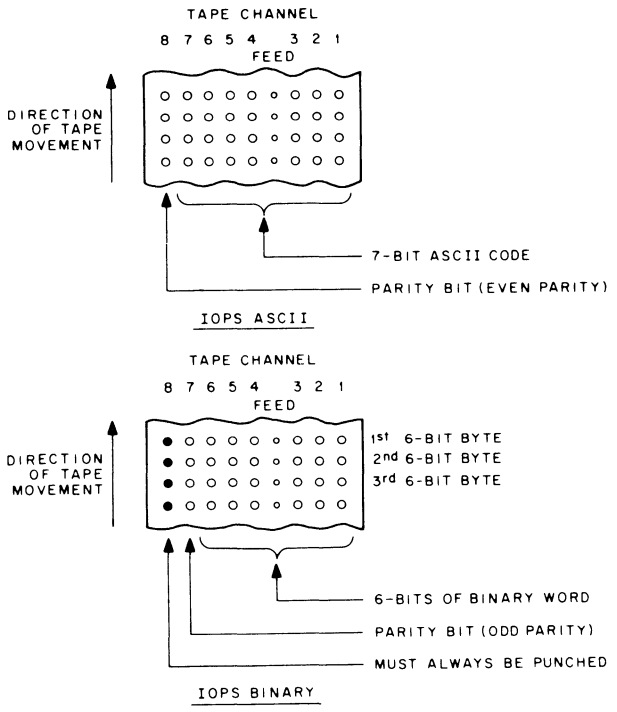


Figure 2-6 IOPS Mode Data on Paper Tape

2.3.1.1 IOPS ASCII - 7-bit ASCII is used by IOPS to accommodate the entire 128-character revised ASCII set (Appendix A). All alphanumeric data, whatever its original form on input (ASCII, Hollerith, etc.) or final form on output, is converted internally and stored as 5/7 ASCII. "5/7 ASCII" refers to the internal packing and storage scheme. Five 7-bit ASCII characters are packed in two contiguous locations as shown in Figure 2-7 and can be stored as binary data on any bulk storage device. Input requests involving IOPS ASCII should be made with an even word count to accommodate the paired input.

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 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.

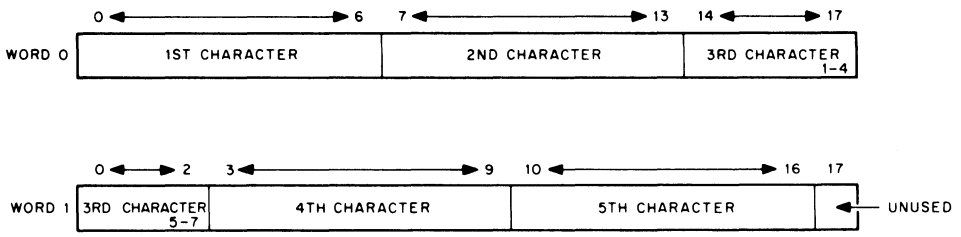


Figure 2-7 5/7 ASCII Packing Scheme

Non-parity 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-parity IOPS ASCII data identically.

An alphanumeric line consists of an initial form control character (line feed, vertical tab, or form feed), 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 by some device handlers for editing or control purposes (see Section 7.4 for effects of control characters on specific devices).

2.3.1.2 IOPS Binary - IOPS Binary data is blocked in an even number of words, with each block preceded by a two-word header. On paper tape (see Figure 2-6), IOPS binary uses six bits per frame, with the eighth channel always set to 1, and the seventh channel containing the parity bit (odd parity) for channels 1 through 6 and channel 8. The parity feature supplements the checksumming as a data validity provision in paper tape IOPS binary.

2.3.2 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, as shown in Figures 2-8 and 2-9. No conversion, checking, or packing is permitted, and character scanning is generally omitted.

2.3.3 Dump Mode

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

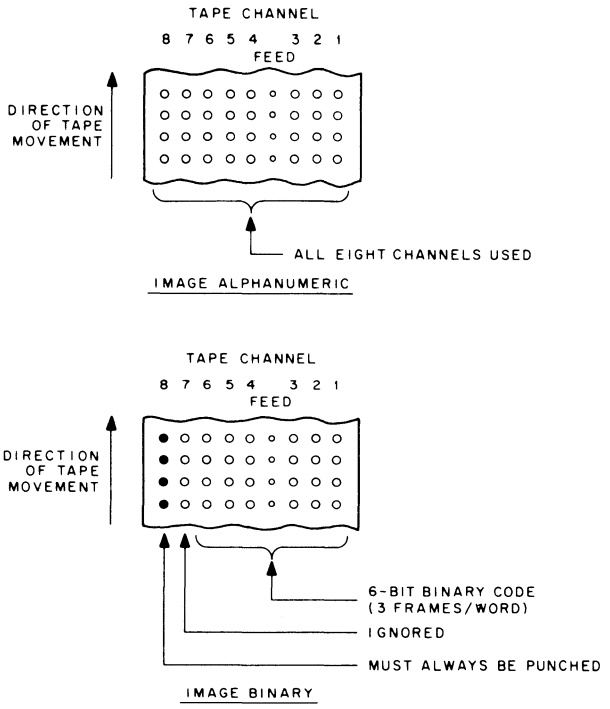


Figure 2-8 Image Mode Data on Paper Tape

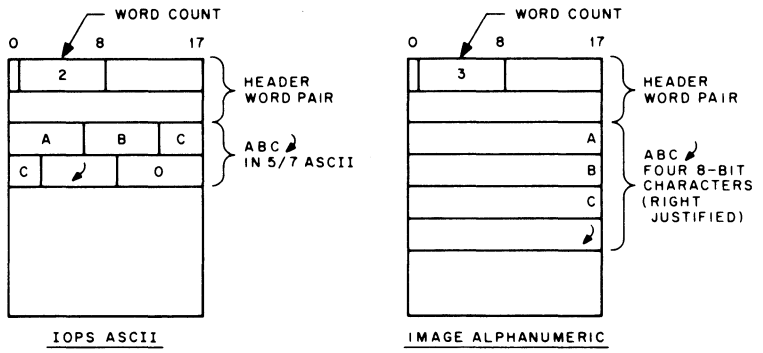


Figure 2-9 IOPS ASCII and Image Alphanumeric Data in Line Buffers

2.3.4 Input/Output Data Mode Terminators

Input/output terminators for each of the data modes are summarized in Table 2-2.

Table 2-2
Input/Output Data Mode Terminators

	IOPS ASCII	IOPS Binary	Image Alphanumeric	Image Binary	Dump
INPUT	Carriage Return ALT MODE Word Pair Count** End of Medium Word Count* End of File**	Word Pair Count End of Medium Word Count* End of File**	Word Count End of Medium End of File**	Word Count End of Medium End of File**	Word Count End of Medium End of File**
OUTPUT	Carriage Return ALT MODE Word Pair Count*** (except on Teletype)	Word Pair Count	Word Pair Count	Word Pair Count	Word Count

*A short Line Indicator will be placed by IOPS into the validity bits (12 and 13) of Header Word 0 if the maximum size of the line buffer is reached before an End-of-File.

**Bulk storage only.

***If the word pair count is not greater than 1, the output line is ignored. If the word pair count is greater than 1, it has no effect and a carriage return or ALT MODE are the only legal line terminators.

2.4 SYSTEM TABLES

System tables used by each of the monitor systems include the Device Assignment Table (.DAT), and the System Communication Table (.SCOM). These tables are discussed in the following paragraphs.

2.4.1 Device Assignment Table (.DAT)

Both FORTRAN IV and MACRO-9 coded user programs, as well as the system programs, specify I/O operations with commands to logical I/O devices. One of the monitor's functions is to relate these logical units to physical devices. To do this, each of the monitors contains a Device Assignment Table (.DAT) which has "slot" numbers that correspond directly to logical I/O device numbers. Each .DAT slot contains the physical device unit number (if applicable) along with a pointer to the appropriate device handler.

All I/O communication in the monitor environment is accomplished by the logical/physical device associations provided by the Device Assignment Table. The use of the Device Assignment Table differs for each of the monitor systems, and is discussed separately for each of the monitors (Chapters 4, 5, and 6).

2.4.2 System Communication Table (.SCOM)

The System Communication Table (.SCOM) provides a list of registers that can be referenced by the monitor, IOPS, and system programs. A complete list of .SCOM entries, and the purpose of each, is given in Table 2-3. The System Communication Table begins at location 100_8.

Table 2-3
System Communication Table (.SCOM) Entries

Word	Purpose
.SCOM	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: Bit 0 1 = API Bit 1 1 = EAE Bit 2 1 = TTY is 35/37 Bit 3 1 = Non-resident KM-9 in core Bit 4 1 = VC38 Character Table Bit 5 1 = 339 Pushdown Table Bit 6 1 = 9-channel, 0 = 7-channel Magnetic tape Bits 15-17 Drum Size: 1 = 32K (RM09A) 2 = 65K (RM09B) 3 = 131K (RM09C) 4 = 262K (RM09D) 5 = 524K (RM09E)
.SCOM + 5	System program starting location
.SCOM + 6	User starting location (bits 3 through 17), and: Bit 0 1 = DDT Load Bit 1 1 = G Load Bit 2 1 = No-symbol-table Load
.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 through 43_8.
.SCOM + 16	Contains PC on keyboard interrupts.
.SCOM + 17	Contains AC on keyboard interrupts.

2.5 SPECIFYING DEVICES USED TO LINKING LOADER

When writing a MACRO-9 program that uses monitor commands (system macros), it is necessary to use the .IODEV pseudo operation somewhere in the program to specify to the Linking Loader which .DAT slots are to be used. The .IODEV pseudo-op causes a code to be generated that is recognized by the Linking Loader and used to load device handlers associated with specified .DAT slots. FORTRAN IV programs cause the compiler to generate this code based on the units specified in READ and WRITE statements. (If a variable is used in a FORTRAN program to specify an I/O unit, handlers will be loaded for all positive .DAT slots that have handlers assigned.) The .IODEV pseudo-op has the following form

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

where the MACRO-9 program containing this statement can use .DAT slots 3, 5, and 6. An error message is generated if a slot called for by a program is unassigned.

CHAPTER 3
USER PROGRAM COMMANDS (SYSTEM MACROS)

All user program commands or system macros are described in this chapter for convenient reference. All commands that apply to the I/O Monitor are presented first and are followed by descriptions of the additional commands that apply to the Keyboard and Background/Foreground Monitors, respectively. Because of the upward compatibility of monitor systems, all I/O Monitor commands (system macros) are also used in the Keyboard and Background/Foreground Monitor environments. All Keyboard Monitor commands (system macros) are also used in the Background/Foreground Monitor environment. Note that all information presented in this manual for the Background/Foreground Monitor is preliminary and subject to change.

NOTE

When executing a system macro, the monitor makes no attempt to save the user's accumulator and link bit.

3.1 I/O MONITOR COMMANDS (SYSTEM MACROS)

The following commands are available for use in programs that are to operate in the I/O Monitor environment. Each command is described in detail in the paragraphs that follow.

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	Checks availability of the user's line buffer and waits if busy.
.WAITR	Checks availability of the user's line buffer, and provides transfer address for busy return.
.CLOSE	Terminates use of a file.
.TIMER	Calls and uses real-time clock.
.EXIT	Returns control to the Monitor.

3.1.1 .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* (should be in every .INIT statement)

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 through 2 of this word.**
LOC + 2	R	
LOC + 3	n	/Maximum size of line buffer associated with .DAT /slot a, for example, 255_{10} for DECtape.***

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. Each initialized .DAT slot constitutes an open file to the device handler and must be .CLOSEd. 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.

3.1.2 .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

---

*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.

L = Line Buffer address

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

EXPANSION:

LOC	CAL + M_{6-8} + $9-17$	
LOC + 1	10	/CAL Handler will place unit number (if applicable) /into bits 0 through 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 the operation, M defines the mode of the data to be transferred; L is the address of the line buffer; and W is 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 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 through 13 of the first word of the line buffer header to ascertain that the line was read without error. Bits 14 through 17 should be checked for end-of-medium and end-of-file conditions.

3.1.3 .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} to a_{9-17}	
LOC + 1	10	/CAL Handler will place the unit number (if appli- /cable) associated with .DAT slot a into bits /0 through 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 transfer to the device has been completed.

On non-bulk storage devices, headers are output along with the data in IOPS binary mode only (bit 9 and 11 of header word 0 should be set to 1). On bulk storage devices, headers are output along with the data in all modes except dump mode. 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 over maximum size (or word count) 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 or disk may be occupied by a single dump command. A subsequent .WRITE in dump mode will utilize the unfilled portion of the last block.

3.1.4 .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 through 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 of data has not been completed, control is returned to the .WAIT macro. .WAIT must also be used after the .TRAN command.

3.1.5 .WAITR

FORM: .WAITR a, ADDR

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

ADDR = Address to which control is passed if line buffer is not available for use.

EXPANSION:

LOC	CAL + 1000_8 + a_{9-17}	
LOC + 1	12	/The CAL Handler will place the unit number (if /applicable) associated with .DAT slot a into bits / 0 through 2.
LOC + 2	ADDR	

DESCRIPTION: .WAITR is also used to detect the availability of the user's line buffer. If the buffer is available, control is returned to the user immediately after the .WAITR macro expansion (LOC + 3). If the transfer of the data has not been completed however, control is given to the instruction at ADDR. It is the user's responsibility to return to the .WAITR to again check the availability of the buffer.

3.1.6 .CLOSE

FORM: .CLOSE a

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

EXPANSION:

LOC	CAL + a_{9-17}	
LOC + 1	6	/The CAL Handler will place the unit (if applicable) /associated with .DAT slot a into bits 0 through 2.

DESCRIPTION: When 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 (end-of-file) line is output before the file is finally closed. If a refers to a file-oriented device, any earlier file of the same name and extension, as currently referenced, is deleted from its directory after the new file is written.

3.1.7 .TIMER

FORM: .TIMER n,C

VARIABLES: 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 + 3	-n	

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

C + 1 is the location to which control is given when the Monitor services the clock interrupt. The coding at C should be in subroutine form; for example,

```
C       0           /C + 1 is reached via JMS
        DAC SAVEAC
        :  } Must not contain any Monitor CALs
        :  } in I/O or Keyboard Systems.
        LAC C       /Restore Link
        RAL
        LAC SAVEAC  /Restore AC
XIT     JMP* C
```

so that control will return to the originally-interrupted sequence when the interval-handling routine has been completed. The Monitor automatically reenables the interrupt system before transferring control to C + 1. If the user wishes to initiate another interval at the completion of the previous interval in the subroutine specified to .TIMER, he may do so as follows:

```
        LAC  (desired interval in 2's complement)
        DAC* (7
        LAC C       /Restore Link
        RAL
        LAC SAVEAC  /Restore AC
        CLON        /Turn on clock
        JMP* C
```

3.1.8 .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 non-resident 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. In the Background/Foreground Monitor environment, the effect of the .EXIT depends upon whether it occurs in a BACKGROUND or a FOREGROUND job (see Section 6.4.5).

3.2 KEYBOARD MONITOR COMMANDS (SYSTEM MACROS)

The commands listed below are available for use in programs that are to operate in the Keyboard Monitor environment. Each command is described in detail in the following paragraphs. Refer to Chapter 5 for a complete description of bulk storage file organization.

Name	Purpose
.SEEK	Locates file on file-oriented device and begins data input.
.ENTER	Primes file-oriented device for output.
.FSTAT	Checks presence of file on file-oriented device.
.RENAM	Renames file on file-oriented device.
.DLETE	Deletes file from file-oriented device.
.TRAN	Reads or records user-specified block on bulk storage devices, providing the user with the capability to determine the structure of the files on the device.
.CLEAR	Initializes file structure on file-oriented device.
.MTAPE	Provides special commands for IBM-compatible magnetic tape.

The first seven of the eight macros listed above apply to the file-oriented devices - DECtape (DT), disk (DK), drum (DR) and magnetic tape (MT); they are ignored by nonfile-oriented devices, depending upon the device handler used. The eighth macro, .MTAPE, handles the nonfile-oriented functions of magnetic tape (REWIND, BACKSPACE, etc.). If these nonfile-oriented commands are given to file-oriented devices, they are ignored by the device handling routines. To the the .MTAPE commands, however, (REWIND TO LOAD POINT, BACKSPACE RECORD), may be used with disk drum or DECtape; when so used, they preclude the use of .SEEK or .ENTER (see Section 5.6).

3.2.1 .SEEK

FORM .SEEK a,D

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

D = Address of user directory entry block

EXPANSION:

```
LOC       CAL + a9-17
LOC + 1   3          /The CAL Handler will place unit number (if appli-
                     /cable) into bits 0 through 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 (that is, 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 allows the user to check the directory for a named file and to retain control if 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 nulls (0).

File Name Extension: Up to three 6-bit trimmed ASCII characters, padded with nulls. (The symbol @ produces a zero when using SIXBT.)

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.

3.2.2 .ENTER

FORM: .ENTER a, D

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

D = Address of user directory entry block

EXPANSION:

```
LOC       CAL + a9-17
LOC + 1   4          /The CAL Handler will place the unit number (if
                     /applicable) associated with .DAT slot a into
                     /bits 0 through 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 any earlier file 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.

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

DESCRIPTION: .FSTAT 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 returned to the Monitor error routine to output an IOPS 13 error code on the Teletype.

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

---

*Bits 0 through 2 of LOC + 2 must be set to zero prior to the execution of the CAL at LOC. On return, bits 0 through 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 through 7 to be specified)

If the contents of the AC are 0 on return from .FSTAT (indicating that the file was not found), bits 0 through 2 of LOC + 2 should be checked because if they are still 0, the device was non-file-oriented.

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 cannot be found.

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

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 cannot be found.

3.2.6 .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 for example, block number (octal radix) for DECtape

L = Core starting address

W = Word count (decimal radix)

EXPANSION:

```
LOC       CAL + D7-8 + a9-17
LOC + 1   13         /The CAL Handler will place the unit number (if
                     /applicable) associated with .DAT slot a into bits
                     /0 through 2.
LOC + 2   B
```

*DECtape only.

```
LOC + 3     L
            .DEC            /Decimal radix
LOC + 4     -W
```

DESCRIPTION: .TRAN is employed when the user desires total freedom in data structuring of 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.

3.2.7 .CLEAR

FORM: .CLEAR a

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

EXPANSION:

```
LOC         CAL + a9-17
LOC + 1     5           /The CAL Handler will place the unit number (if
                        /applicable) associated with .DAT slot a into bits
                        /0 through 2.
```

DESCRIPTION: .CLEAR is used to initiate the IOPS file structuring of the device referenced by .DAT slot a by initializing its existing directory. The directory area and file bit map blocks on the file-structured device are set to 0 (except for those bits in the directory bit map referring to the directory itself and the file bit maps).

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; otherwise, control is transferred to the monitor error handling routine to output an IOPS 10 error code (file still active).

3.2.8 .MTAPE

FORM: .MTAPE a,(XX)

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

XX = Number of magnetic tape function or configuration:

00 = Rewind to load point
02 = Backspace record
03 = Backspace file
04 = Write end-of-file
05 = Skip record
06 = Skip file
07 = Skip to logical end-of-tape
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

EXPANSION: LOC CAL + XX_{5-8} + a_{9-17}

LOC + 1 /The CAL Handler will place the unit number (if /applicable) associated with .DAT slot a into bits /0 through 2.

DESCRIPTION: .MTAPE is used to perform functions unique to non-file-oriented bulk storage devices. In general, these functions are intended for 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.

3.3 BACKGROUND/FOREGROUND MONITOR COMMANDS (SYSTEM MACROS)

The following commands are available for use in programs that are to operate in the Background/Foreground Monitor environment. Each command is described in detail in the paragraphs that follow. It should be noted that this information is preliminary and subject to change.

Name	Purpose
.REALR	Real time transfer of data from device to line buffer.
.REALW	Real time transfer of data from line buffer to device.
.IDLE	Allows FOREGROUND job to indicate that control can be given to lower levels of the FOREGROUND job or to the BACKGROUND job.
.IDLEC	Allows FOREGROUND Mainstream to give control to BACKGROUND job and subsequently continue processing after the .IDLEC.
.TIMER	Calls and uses real-time clock and allows priority level to be established.
.RLXIT	Accomplishes the exit from all real-time subroutines that were entered via .REALR, .REALW or .TIMER requests.

3.3.1 .REALR

FORM: .REALR a, M, L, W, ADDR,p

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

ADDR = 15-bit address of closed subroutine that is given control when the request made by the .REALR is completed.

p = API priority level at which to go to ADDR

P	Priority Level
0	Mainstream
4	Level of .REALR
5	API software level 5
6	API software level 6
7	API slftware level 7

EXPANSION:

```
LOC       CAL + 10000 + M6-8 + a9-17
LOC + 1   10
LOC + 2   L
          .DEC            /Decimal radix
LOC + 3   -W
          .OCT            /Octal radix
LOC + 4   ADDR + p00000
```

DESCRIPTION: The .REALR 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, L is the address of the line buffer, W is the number of words in the line buffer (including the two-word header), and ADDR is the address of a closed subroutine which should be constructed as shown in the following example.

```
ADDR    0
        DAC       SAVEAC        /Save AC
        :   } Any system macro may be issued
        .   } at this point.
        LAC       SAVEAC        /Restore AC
        .RLXIT    ADDR          /Return to interrupted point via monitor CAL**
```

3.3.2 .REALW

FORM:

```
.REALW  a, M, L, W, ADDR, p
```

*The subroutine specified by a .REALR should not be used at more than one priority level. The subroutine is entered via a JMS and thus cannot be protected against re-entry.

**.RLXIT is described in Section 3.3.6.

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

ADDR = 15-bit address closed subroutine that is given control when the request made by the .REALW is completed.

p = API priority level at which to go to ADDR

P	Priority Level
0	Mainstream
4	Level of .REALW
5	API software level 5
6	API software level 6
7	API software level 7

EXPANSION:

```
LOC       CAL + 10000 + M6-8 + a9-17
LOC + 1   11
LOC + 2   L
          .DEC          /Decimal radix
LOC + 3   -W
          .OCT          /Octal radix
LOC + 4   ADDR + p00000
```

DESCRIPTION: The .REALW command is used to transfer the next line of data from the line buffer in the user's program to the device assigned to .DAT slot a. In this operation, M defines the mode of the data to be transferred, L is the address of the line buffer, W is the count of the number of words in the line buffer (including the two-word header), and ADDR is the address of a closed subroutine which should be constructed as shown in the following example.

EXAMPLE:

```
ADDR    0
        DAC       SAVEAC        /Save AC
        :  }
        :  }  Any system macro may be
        .  }  issued at this point.
        LAC       SAVEAC        /Restore AC
        .RLXIT    ADDR          /Return to interrupted point via Monitor CAL**
```

*The subroutine specified by a .REALW should not be used at more than one priority level. The subroutine is entered via a JMS and thus cannot be protected against re-entry.

**.RLXIT is described in Section 3.3.6.

3.3.3 .IDLE

FORM: .IDLE

EXPANSION: LOC CAL

LOC + 1 17

DESCRIPTION: The FOREGROUND job in a Background/Foreground environment can indicate that wishes to relinquish control to lower levels of the FOREGROUND job or to the BACKGROUND job by executing this command. This is useful when the FOREGROUND job is waiting for the completion of real-time I/O from any one of a number of I/O requests that it has initiated or when it is waiting for completion of .TIMER requests.

The .IDLE is the logical end of the current level's processing; that is, control never returns LOC + 2. If the .IDLE is issued at a FOREGROUND API software level, it effects a DBR from the handler at that level. Other routines in the APIQ for that level are deferred until a hardware interrupt is requested for that level again. If the .IDLE is issued at FOREGROUND mainstream, it effects an I/O BUSY situation (except that no BUSY flag is set) and control goes to the BACKGROUND JOB. If the .IDLE is issued at BACKGROUND mainstream level, it effects an I/O BUSY situation and control is returned to the .IDLE CAL.

3.3.4 .IDLEC

FORM: .IDLEC

EXPANSION: LOC CAL+1000

LOC + 1 17

DESCRIPTION: Identical to .IDLE except when issued at FOREGROUND mainstream level. In this case, control goes to the BACKGROUND job, and LOC + 2 is saved as the FOREGROUND mainstream return pointer. The next time control returns to FOREGROUND (at any priority level), FOREGROUND mainstream processing will CONTINUE at LOC + 2 when mainstream becomes the highest active FOREGROUND level.

3.3.5 .TIMER

FORM: .TIMER n, C, p

VARIABLES: n = Number of clock increments (decimal radix)*

C = Address of subroutine to handle interrupt at end of interval**

p = API priority level at which to go to C

*To transfer control to subroutine C at priority level p immediately, n should be set equal to zero.
**The subroutine specified should not be used at more than one priority level. The subroutine is entered via a JMS and thus cannot be protected against re-entry.

P	Priority Level
0	Mainstream
4	Level of .TIMER
5	API software level 5
6	API software level 6
7	API software level 7

```
EXPANSION:      LOC         CAL
                LOC + 1     14
                LOC + 2     C + p00000
                            .DEC            /Decimal radix
                LOC + 3     -n
```

DESCRIPTION: .TIMER is used to set the real-time clock to n increments and to start it. Each clock increment represents 1/60s for 60 Hz systems and 1/50s for 50 Hz systems. When the monitor services the clock interrupt, it passes control to location C + 1 with the priority level set to p. The coding at C should be in subroutine form, for example,

```
        C       0
                DAC         SAVEAC          /C + 1 is reached via JMS

                :   } The restriction that applies to non B/F monitors does
                    } not apply here. Any system macro (including .TIMER)
                    } may be issued at this point.

                LAC         SAVEAC          /Restore AC

                .RLXIT      C               /Return to interrupted point via monitor CAL
```

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

3.3.6 .RLXIT

```
FORM:           .RLXIT      ADDR
VARIABLES:      ADDR = entry point address of real-time subroutine that is to be exited.
EXPANSION:      LOC         CAL         ADDR
                LOC + 1     20
```

DESCRIPTION: .RLXIT is used to exit from all real-time subroutines that were entered via .REALR, .REALW or .TIMER requests. The instruction just preceding the .RLXIT call should restore the AC with the value of the AC on entrance to this subroutine. .RLXIT will restore the link from bit 0 of ADDR.

.RLXIT protects against re-entrance to BACKGROUND or FOREGROUND mainstream real-time subroutines. When the contents of ADDR is non-zero, the subroutine is assumed active; .RLXIT sets the contents of ADDR to 0 thus making it available again. NOTE: Real-time subroutines should initially have their entry point register set to 0.

CHAPTER 4
INPUT/OUTPUT MONITOR

4.1 INPUT/OUTPUT MONITOR FUNCTIONS

The I/O Monitor of the PDP-9 ADVANCED Software System simplifies the programming of input and output functions in the basic paper-tape environment. It serves as an interface between the system and user programs and the external world of device hardware, relying 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. Since upward compatibility exists between the monitor systems, user programs that are written to operate under control of the I/O Monitor will also operate without modification, under control of the Keyboard and Background/Foreground Monitors.

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.

The reader is referred to Chapter 2 for a general discussion of the monitor environment, and to Chapter 3 for detailed descriptions of user program commands (system macros) available in the I/O Monitor environment.

4.2 PROGRAMMING EXAMPLE

The following example illustrates the use of system macros with MACRO-9 programs in the I/O Monitor environment. The example inputs a line of data from the Teletype keyboard, and outputs the same line of data to the Teletype. The arguments used by the system macros are given symbolic names (via MACRO-9 direct assignment statements) to facilitate recall for the programmer, and to change the arguments easily, if desired. Note the use of the pseudo ops .TITLE, .IODEV, .BLOCK, and .END, in addition to the system macros. The assembly listing that follows the example shows how the system macros are expanded at assembly time. (The reader may wish to compare these expansions with the system macro descriptions in Chapter 3.)

```
                .TITLE ECHO
TTI=2
TTO=410
OUT=1
IN=0
IOPS=2
```

```
          .IODEV    2,4
START     .INIT     TTO,OUT,RESTRT      /INITIALIZE TELETYPE OUTPUT
          .INIT     TTI,IN,RESTRT       /AND INPUT
BEGIN     .READ     TTI,IOPS,BUFFER,34  /INPUT IOPS ASCII FROM TELETYPE
          .WAIT     TTI                 /WAIT UNTIL INPUT COMPLETE
          .WRITE    TTO,IOPS,BUFFER,34  /OUTPUT SAME DATA ON TELETYPE
          .WAIT     TTO                 /WAIT UNTIL OUTPUT COMPLETE
          JMP       BEGIN               /LOOP TO INPUT AGAIN
RESTRT    .CLOSE    TTI                 /TERMINATE INPUT
          .CLOSE    TTO                 /TERMINATE OUTPUT
          JMP       START               /RETURN TO REINITIALIZE
BUFFER    .BLOCK    42                  /SET UP TELETYPE BUFFER (34 DEC)
          .END      START               /END OF ECHO PROGRAM
```

ASSEMBLY LISTING:

```
ECHO      PAGE   1

                                                .TITLE    ECHO
                  000007   A          TTI=2
                  000010   A          TTO=4
                  000001   A          OUT=1
                  000000   A          IN=0
                  000002   A          IOPS=2
                                                .IODEV    2,4
     00000   R                        START     .INIT     TTO,OUT,RESTRT
     00000   R   001010   A GEN*                CAL+OUT*1000 TTO&777
     00001   R   000001   A GEN*                1
     00002   R   000025   R GEN*                RESTRT
     00003   R   000000   A GEN*                0
                                                .INIT     TTI,IN,RESTRT
     00004   R   000007   A GEN*                CAL+IN*1000 TTI&777
     00005   R   000001   A GEN*                1
     00006   R   000025   R GEN*                RESTRT
     00007   R   000000   A GEN*                0
     00010   R                        BEGIN     .READ     TTI,IOPS,BUFFER,34
     00010   R   002007   A GEN*                CAL+IOPS*1000 TTI&777
     00011   R   000010   A GEN*                10
     00012   R   000032   R GEN*                BUFFER
                          GEN*                .DEC
     00013   R   777736   A GEN*                -34
                                                .WAIT     TTI
     00014   R   000007   A GEN*                CAL TTI&777
     00015   R   000012   A GEN*                12
                                                .WRITE    TTO,IOPS,BUFFER,34
     00016   R   002010   A GEN*                CAL+IOPS*1000 TTO&777
     00017   R   000011   A GEN*                11
     00020   R   000032   R GEN*                BUFFER
                          GEN*                .DEC
     00021   R   777736   A GEN*                -34
                                                .WAIT     TTO
     00022   R   000010   A GEN*                CAL TTO&777
     00023   R   000012   A GEN*                12
```

```
00024  R  600010  R                    JMP       BEGIN
00025  R                      RESTRT    .CLOSE    TTI
00025  R  000007  A  GEN*              CAL  TTI&777
00026  R  000006  A  GEN*              6
                                       .CLOSE    TTO
00027  R  000010  A  GEN*              CAL  TTO&777
00030  R  000006  A  GEN*              6
00031  R  600000  R                    JMP       START
00032  R          A          BUFFER    .BLOCK    42
          000000  R                    .END      START
                                       NO ERROR LINES
```

4.3 OPERATING THE I/O MONITOR SYSTEM

The reader is referred to the I/O Monitor Guide (DEC-9A-MIFA-D) for detailed operating procedures for system programs in the I/O Monitor environment. The following PDP-9 ADVANCED Software System manuals contain additional detailed information on system programs.

Manual	Document Number
Utility Programs	DEC-9A-GUAB-D
MACRO-9 Assembler	DEC-9A-AMZA-D
FORTRAN IV	DEC-9A-KFZA-D

This section contains descriptions of loading programs, device assignments, and error detection and handling.

4.3.1 Loading Programs in the I/O Monitor Environment

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, is punched on a separate paper tape in absolute format. Each PDP-9 core configuration (8, 16, 24 and 32K) requires a separate set of paper tape system programs. The ten system tapes supplied are:

FORTRAN IV
MACRO-9
PIP-9
Text Editor
Linking Loader
DDT (without Patch File capabilities)
DDT (with Patch File capabilities)
7 to 9 CONVERTER (CONV)
CHAIN
EXECUTE

In addition, the utility program PUNCH 9, which provides the ability to dump an executable core load and .ABS loader onto paper tape, is provided with all paper tape systems. See Figure 4-1 for memory maps of I/O Monitor System.

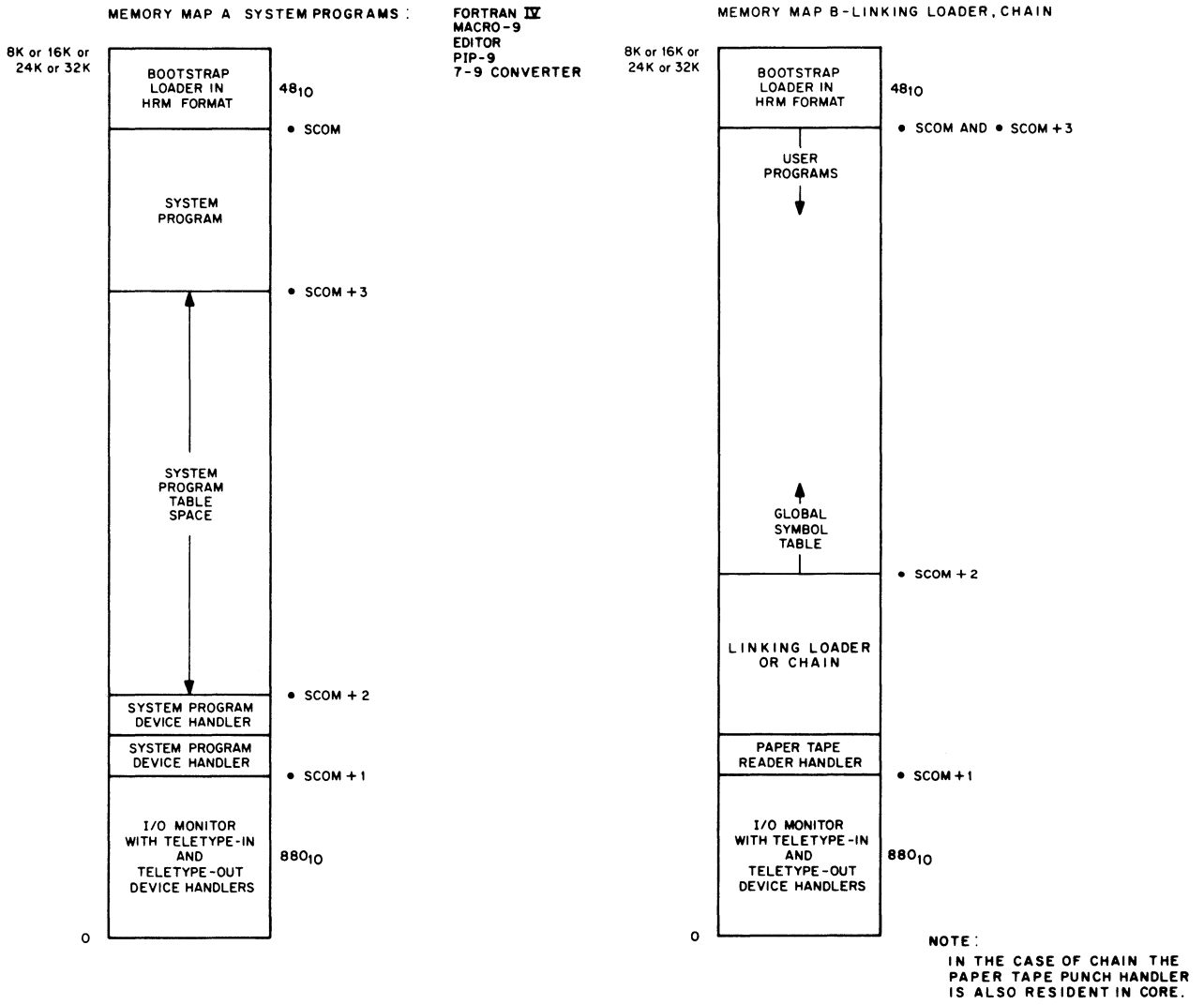
At the beginning of each tape is a Bootstrap Loader in hardware READIN mode. By setting the starting address of the Loader* on the console address switches, depressing I/O RESET, and then depressing the READIN switch, these system tapes may be loaded.

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

Once the system program (FORTRAN IV, MACRO-9, Text Editor, CONV, PIP, CHAIN, EXECUT or Linking Loader/DDT) has been loaded and takes control, the individual system program operating procedures come into use. (See I/O Monitor Guide, DEC-9A-MIPA-D.)

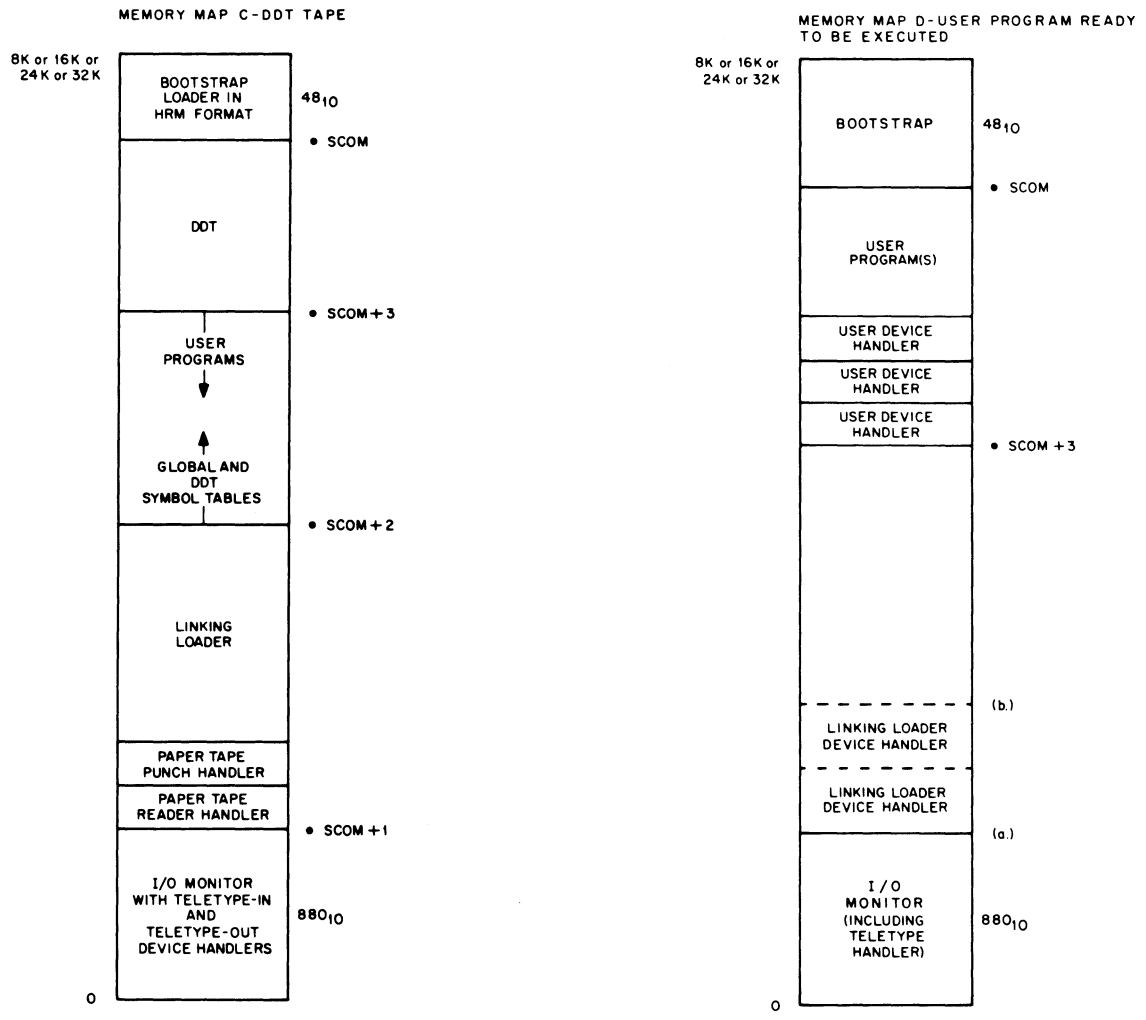
User programs, however, normally exist in relocatable form, as output from FORTRAN IV or MACRO-9; these tapes do not contain copies of the I/O Monitor. To load these programs, a copy of the Linking Loader or DDT should be loaded first. The user should then initiate loading of his main program followed by all required subprograms. By loading subprograms in order of size (largest first, smallest last), the user has a better chance of satisfying core requirements for his program in systems with extended core memory. The version of the I/O Monitor (including the device handlers) contained on the Linking Loader or DDT tape may be used with user programs, and the Linking Loader or DDT can be used to load the necessary device handlers as well as the user's object programs.

*17720 for 8192 word systems, 37720 for 16,384 words, 57720 for 24,576 words, and 77720 for 32,768 words.



Refer to Section 7.4 for sizes for device handlers. Refer to Memory Map D for results of Linking Loader.

Figure 4-1 I/O Monitor System Memory Maps



Refer to Memory Map E for results of Link Loading in DDT mode.

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

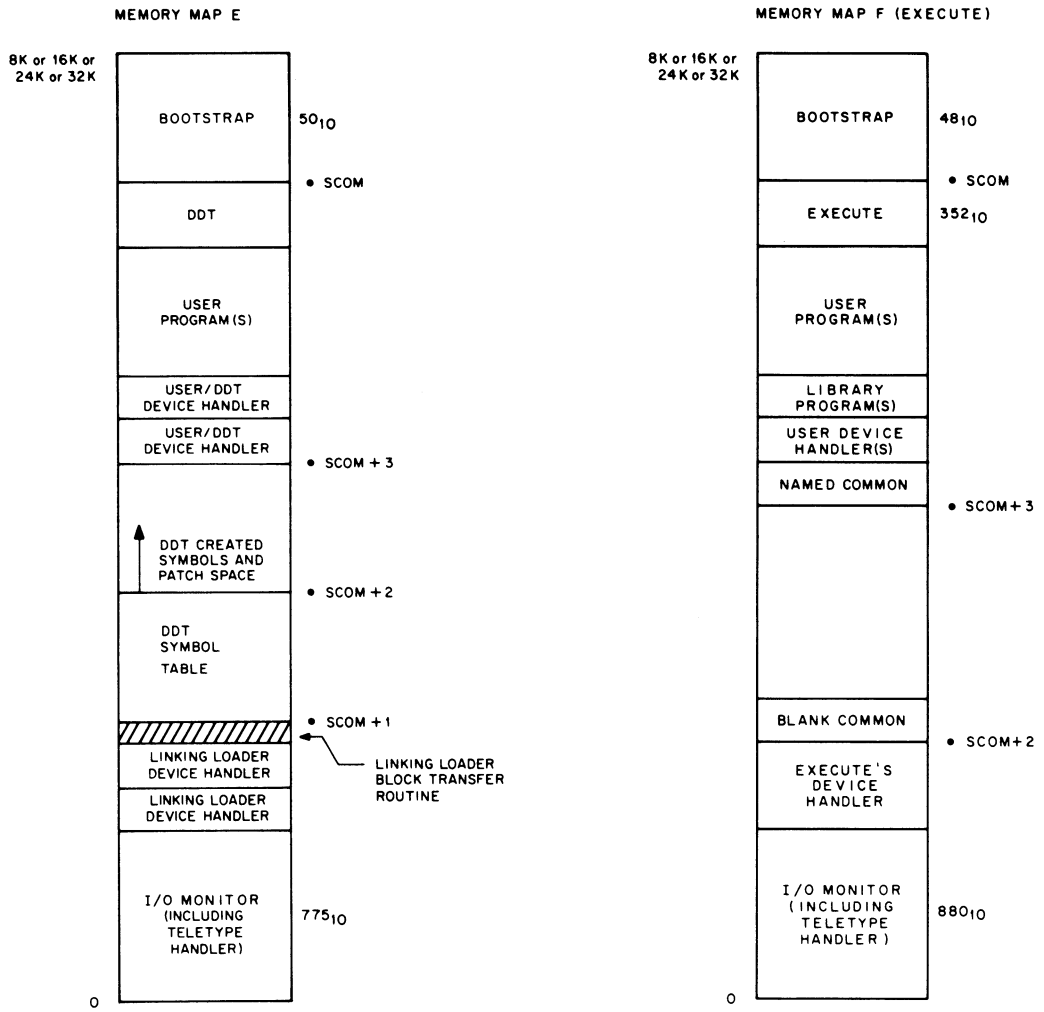
Refer to Section 7.4 for sizes of device handlers.

.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 Loader.

Figure 4-1 I/O Monitor System Memory Maps (Cont)



Refer to Section 7.4 for sizes for 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.

When the user types in the name of the XCT File to be run, .EXECUTE brings in the first chain from paper tape.

FORTRAN programs pass on data in blank common starting at .SCOM + 2. Macro programs pass on data between .SCOM + 2 and .SCOM + 3.

A call from the running chain to bring in another chain is effected by transferring control back to Execute.

Figure 4-1 I/O Monitor System Memory Maps (Cont)

4.3.2 Device Assignments

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

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 standard assignments for the device assignment table for user programs and system programs other than PIP are shown in Figure 4-2. Figure 4-3 illustrates PIP assignments, which include the Card Reader (CR01E, CR02B) and Line Printer as standard devices.

	.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
	7	Paper Tape Punch	(PPA.)	Scratch
	10	Paper Tape Reader	(PRA.)	Scratch
.DATND =.				

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

*See Section 7.4 for a description of the handlers.

The negative .DAT slot assignments for use by system programs may be changed by DEC. For example, .DAT slot -10 might be associated with a card reader, while .DAT slot -12 could be assigned to a line printer. Provision for changing positive .DAT slot assignments for use by relocatable user programs is included in the PUNCH9 utility program (see section 4.3.5). For example, a magnetic tape handler (MTF) or one of the drum handlers (DRA, DRB, DRC, DRD) could be added.

.DAT Slot	Device	Handler	Use
1	Teletype	(TTA.)	Input/Output
2	Teletype	(TTA.)	Input/Output
3	Paper Tape Reader	(PRA.)	Input
4	Line Printer	(LPA.)	Output
5	Paper Tape Punch	(PPA.)	Output
6	Card Reader	(CDB.)	Input
7	Paper Tape Punch	(PPA.)	Output
10	Paper Tape Reader	(PRA.)	Input

Figure 4-3 Device Assignment Table (.DAT) for PIP

4.3.3 Error Detection and Handling

Comprehensive error checking is provided by the Linking Loader and the Input/Output Programming System. Detailed lists of errors that may occur are given in Appendix D and E, respectively. The other system programs also provide comprehensive error checking. Refer to the appropriate PDP-9 ADVANCED Software System manual (listed at the beginning of this section) for detailed information on these errors.

4.3.4 Control Character Commands in the I/O Monitor Environment

All control character commands recognized by the I/O Monitor are summarized in Table 4-1. These commands (except RUBOUT) are formed by holding down the CTRL key while striking a letter key. The character or characters echoed on the Teletype and the resulting action is given in the table for each command.

Table 4-1
Control Character Commands

Command	Echo	Action
CTRL S	↑ S	Starts user program after Linking Loader has brought it into core via a LOAD command.
CTRL T	↑ T	CTRL T is applicable only when using DDT and forces control back to DDT which types DDT > to indicate its readiness for another DDT command. All previous DDT conditions remain intact (breakbpoints, register modifications, etc.).
CTRL R	↑ R	Allows the user to continue when an IOPS 4 (device not ready) error occurs. The user must first ready the device, and then type CTRL R.
CTRL P	↑ P	Forces control to last address specified in the .INIT command referencing Teletype. Used by system programs to reinitialize or restart.
CTRL U	@	Cancels current line on Teletype (input or output).
RUBOUT	\	Cancels last character input from Teletype (not applicable with DDT).

4.3.5 Modifying System Programs and Building Executable User Core Loads in the I/O Monitor Environment

The capability of modifying or patching system programs in the I/O Monitor environment is provided by the utility program PUNCH9. PUNCH9 allows for producing an executable core load on paper tape in .ABS format with the standard .ABS loader (HRM 17720 of the highest available core bank) on the front of the tape. This is particularly useful when the core load consists of a relocatable main program, subroutines and library routines, the repetitive loading of which tends to be time consuming. It is further possible to specify the number of .ABS tapes to be output (1-9) for convenience of tape handling.

The areas of memory output by PUNCH9 are 0 up to .SCOM + 2 (the first free cell) and .SCOM + 3 (last free cell) up to the .ABS loader (17720 module 8K). PUNCH9 is loaded (I/O RESET and READIN) at 17720 of the highest core bank available. It loads and relocates part II of itself in free core, i.e., from the cell in .SCOM + 2 and up. When loaded it types the following message on Teletype:

```
P, T or S? ↵
>
```

The user is expected to type ↑P or ↑T or ↑S to define the starting location of the program to be punched followed by carriage return (1 tape) or a number (1-9) specifying the total number of tapes into which binary output is to be divided.

↑T should be used if DDT is part of the core load.

↑P should be used for all other system programs and user programs which have already initialized (.INIT) the Teletype with a RESTART address at the time PUNCH9 was executed.

↑S should be used only if the core load was output by PUNCH9 after Linking Loader operation at the moment when the loader itself was expecting the ↑S command. All tapes output by PUNCH9 are loaded by hitting I/O RESET and the READIN key with ADDRESS switches set to 17720 of the highest core bank.

4.3.5.1 Modifying User .DAT Slots in the I/O Monitor Environment - Although it is not possible to reassign negative .DAT slots at load time in the I/O Monitor environment for system programs (reassembly is required), PUNCH9 provides this capability for the user or positive .DAT slots.

For example, Figure 4-2 lists the standard .DAT slot assignments. A relocatable user program wanting to use drum (handlers DRA., DRB., DRC. or DRD.) or magnetic tape (MTF.) or card reader (CDB.) or line printer (LPA.) whose handlers are included in the paper tape library has no way of doing so unless he can modify the positive .DAT table. The modifications procedure is as follows:

1. Load the Linking Loader (or DDT) tape into core (HRM 17720 modulo 8K).
2. Stop the computer and modify the appropriate .DAT slot cell (.DAT = 135) according to the Loader - I/O correspondence table below. For example, if .DAT slot 7 is to be assigned to drum unit 1 using handler DRD., cell 144 should be changed to 100013.*
3. Load PUNCH9 into core (HRM 17720 modulo 8K).
4. Type ↑P in response to the query from PUNCH: "P, T or S?"
5. Load the resultant punched tape into core (HRM 17720 modulo 8K).

*Note: The unit number is stored in bits 0-2.

6. The Loader will restart, ready for acceptance of typed program names to be loaded. If .DAT slot 7 is referenced by the user program (e.g., IODEV 7), DRD. for drum unit 1 will be loaded from the I/O Library.

Loader - I/O Correspondence

Handler	.DAT Slot Value
TTA.	1
PRA.	2
PRB.	3
PPA.	4
PPB.	5
PPC.	6
LPA.	7
CDB.	11
MTF.	13**
DRA.	14**
DRB.	15**
DRC.	16**
DRD.	17**

** With unit number in bits 0-2.

CHAPTER 5
KEYBOARD MONITOR

5.1 KEYBOARD MONITOR FUNCTIONS

The Keyboard Monitor is designed to operate with a PDP-9 that has some form of bulk storage (see Hardware Requirements, Section 1.2). It includes all elements of the Input/Output Monitor in addition to routines that accept and interpret Teletype keyboard commands, change device assignments, and automatically load and initiate system and user programs.

The reader is referred to Chapter 2 for a general discussion of the monitor environment, and to Chapter 3 for a detailed description of user program commands (system macros) available in the Keyboard Monitor environment. The Keyboard Monitor Guide (DEC-9A-MKFA-D) contains detailed operating procedures for system programs mentioned in this chapter and a good example (Appendix J) of an actual keyboard session using system software.

5.2 PROGRAMMING EXAMPLE

The following example illustrates the use of system macros with MACRO-9 programs in the Keyboard Monitor Environment. The example inputs a line of data from the Teletype keyboard, writes it on DECtape, reads it back from DECtape, and outputs it on the Teletype. Before subsequent keyboard inputs, the program prints the messages:

```
FILE ALREADY PRESENT!!
DO YOU WISH TO KEEP IT? (Y OR N) AND CR.
```

By typing Y on the keyboard, the file is saved and a new file is created for the next line of input from the keyboard. By typing N on the keyboard, the next line of data input from the keyboard is written on DECtape with the same file name given to the previous line.

The name of the file is initially ECHO TST. The file name for each new file (providing that the previous file was not deleted) is obtained by incrementing location NAME+1. This produces a series of files, ECHO TST, ECHO A TST, ECHO B TST, ECHO C TST,...etc., (since the alphabet in .SIXBIT begins 01_8, 02_8, 03_8, etc).

The arguments used by the system macros are given symbolic names by means of MACRO-9 direct assignment statements at the beginning of the program to facilitate recall for the programmer, and to change the arguments readily. The partial assembly listing that follows the example shows how the first several system macros are expanded at assembly time. (The reader may wish to compare these expansions with the system macro descriptions in Chapter 3.)

Example:

```
          .TITLE DTECHO
DECTAPE=7
TTI=6
TTO=5
IN=0
OUT=1
IOPS=2
          .IODEV    5,6,7
START     .INIT     DECTAPE,OUT,RESTRT  /INITIALIZE DECTAPE OUTPUT,
          .INIT     TTI,IN,RESTRT       /TELETYPE INPUT,
          .INIT     TTO,OUT,RESRT       /AND TELETYE OUTPUT
          .FSTAT    DECTAP,NAME         /IS FILE PRESENT?
          SZA                           /NO, INPUT KEYBOARD
          JMP       UPDATE              /YES, OUTPUT MSG1 AND MSG2
READKB    .READ     TTI,IOPS,BUFFER,34  /INPUT IOPS ASCII FROM KEYBOARD
          .WAIT     TTI                 /WAIT UNTIL INPUT COMPLETE
          LAC       UDSW                /TEST UPDATE SWITCH
          SZA                           /0 REPLACE INPUT FILE
          JMP       NEWFIL              /-1=SAVE INPUT; CREATE NEW OUTPT
WRITE     .ENTER    DECTAP,NAME         /LOCATE FREE DECTAPE FILE
          .WRITE    DECTAP,IOPS,BUFFER,34        /OUTPUT DATA ON DECTAP
          .WAIT     DECTAP              /WAIT UNTIL OUTPUT COMPLETED
          .CLOSE    DECTAP              /CLOSE FILE
READDT    .INIT     DECTAP,IN,RESTRT    /INITIALIZE DECTAPE INPUT
          .SEEK     DECTAP,NAME         /LOCATE FILE "NAME"
          .READ     DECTAP,IOPS,BUFFER,34        /READ INTO BUFFER
          .WAIT     DECTAP              /WAIT UNTIL READ COMPLETE
          .WRITE    TTO,IOPS,BUFFER,34  /OUTPUT TO TELETYPE
          .WAIT     TTO                 /WAIT UNTIL OUTPUT COMPLETE
RESTRT    .CLOSE    TTO                 /TERMINATE TELETYPE OUTPUT,
          .CLOSE    TTI                 /TELETYPE INPUT,
          .CLOSE    DECTAP              /AND DECTAPE INPUT/OUTPUT
          JMP       START               /LOOP FOR UPDATE OPTION
UPDATE    .WRITE    TTO,IOPS,MSG1,34    /OUTPUT MSG1
          .WAIT     TTO                 /AND MSG2
          .WRITE    TTO,IOPS,MSG2,34    /ON
          .WAIT     TTO                 /TELETYPE
          .READ     TTI,IOPS,COM,8      /READ RESPONSE
          .WAIT     TTI                 /WAIT UNTIL READ COMPLETE
          LAC       COM+2               /GET FIRST WORD
          AND       (774000             /SAVE FIRST SEVEN BITS
          SAD       (544000             /IS CHAR A Y?
          JMP       YES                 /YES
          DZM       UDSW                /NO, SET TO REPLACE INPUT FILE
          JMP       READKB              /LOOP TO READ KEYBOARD

YES       CLC                           /SET UPDATE SW. TO SAVE
          DAC       UDSW                /INPUT, CREATE NEW OUTPUT
          JMP       READKB              /LOOP TO READ KEYBOARD
NEWFIL    ISZ       NAME+1              /CHANGE FILE NAME
          JMP       WRITE               /TO CREATE NEW OUTPUT
MSG1      MSG2-MSG1/2*1000              /WPC FOR HEADER WORD 0
          0
          .ASCII "FILE ALREADY"
          .ASCII "PRESENT!!"<15>
MSG2      COM-MSG2/2*1000               /WPC FOR HEADER WORD 0
          0
```

```
          .ASCII "DO YOU WISH TO KEEP IT?"
          .ASCII "(Y OR N) AND CR."<15>
COM       .BLOCK    10                    /RESPONSE BUFFER
BUFFER    .BLOCK    42                    /DATA BUFFER (34 DECIMAL)
NAME      .SIXBIT "ECHO@@TST"             /FILE NAME
UDSW      0                               /UPDATE SWITCH
          .END      START
```

ASSEMBLY LISTING:

```
DTECHO    PAGE   1

                                          .TITLE    DTECHO
                  000007  A       DECTAP=7
                  000006  A       TTI=6
                  000005  A       TTO=5
                  000000  A       IN=0
                  000001  A       OUT=1
                  000002  A       IOPS=2
                                          .IODEV    5,6,7
     00000  R                     START     .INIT     DECTAP,OUT,RESTRT
     00000  R   001007  A GEN*              CAL+OUT*1000 DECTAP&777
     00001  R   000001  A GEN*              1
     00002  R   000070  R GEN*              RESTRT
     00003  R   000000  A GEN*              0
                                          .INIT     TTI,IN,RESTRT
     00004  R   000006  A GEN*              CAL+IN*1000 TTI&777
     00005  R   000001  A GEN*              1
     00006  R   000070  R GEN*              RESTRT
     00007  R   000000  A GEN*              0
                                          .INIT     TTO,OUT,RESTRT
     00010  R   001005  A GEN*              CAL+OUT*1000 TTO&777
     00011  R   000001  A GEN*              1
     00012  R   000070  R GEN*              RESTRT
     00013  R   000000  A GEN*              0
                                          .FSTAT    DECTAP,NAME
     00014  R   003007  A GEN*              CAL+3000 DECTAP&777
     00015  R   000002  A GEN*              2
     00016  R   000246  R GEN*              NAME
     00017  R   740200  A                   SZA
     00020  R   600077  R                   JMP       UPDATE
     00021  R                     READKB    .READ     TTI,IOPS,BUFFER,34
     00021  R   002006  A GEN*              CAL+IOPS*1000 TTI&777
     00022  R   000010  A GEN*              10
     00023  R   000204  R GEN*              BUFFER
                        GEN*              .DEC
     00024  R   777736  A GEN*              -34
                                          .WAIT     TTI
     00025  R   000006  A GEN*              CAL TTI&777
     00026  R   000012  A GEN*              12
     00027  R   200251  R                   LAC       UDSW
     00030  R   740200  A                   SZA
     00031  R   600132  R                   JMP       NEWFIL
     00032  R                     WRITE     .ENTER    DECTAP,NAME
     00032  R   000007  A GEN*              CAL DECTAP&777
     00033  R   000004
     00034  R
```

5.3 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 only the dollar sign ($) to indicate readiness. In both cases, the keyboard command should be typed on the same line as the dollar sign ($).

Keyboard Monitor commands fall into three categories:

a. Commands that load system programs (terminated with a carriage return (↵) or ALT MODE).

b. Commands to perform special functions.

c. Control character commands, formed by holding down the CTRL key while striking a letter key. These commands are used during the running of system or user programs. (System programs echo control character commands by typing an up arrow (↑) followed by the associated character.)

5.3.1 System Program Load Commands

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 which follow are available to the user for loading systems programs via the Keyboard Monitor.

Command	System Program Loaded
F4	FORTRAN IV Compiler
F4A	Abbreviated FORTRAN IV Compiler
MACRO	MACRO-9 Assembler
MACROA	Abbreviated MACRO-9 Assembler
PIP	Peripheral Interchange Program
EDIT	Symbolic Text Editor
CONV	7-to-9 Converter
LOAD	Linking Loader
GLOAD	Linking Loader (set to load and go)

Command	System Program Loaded
DDT	Dynamic Debugging Technique program
DDTNS	DDT program with no user symbol table
UPDATE	Library File Update program
DUMP	Program to dump saved area (see CTRL Q and QDUMP commands)
PATCH	System tape Patch program
CHAIN	Modified version of Linking Loader (allows for chaining)
EXECUTE(E)	Control program to load and execute chained programs
SGEN	System Generation program

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 Teletype with an appropriate message; such as

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

5.3.2 Special Function Commands

The special-function keyboard commands available in the Keyboard Monitor environment are described in the following paragraphs.

5.3.2.1 LOG (or L) - The LOG command is used to make hard copy records of user comments on the Teletype. When the LOG command is encountered, the Monitor ignores all typing up to and including the next ALT MODE.

Example:

```
$LOG THIS IS AN EXAMPLE.  (ALT MODE)
```

5.3.2.2 SCOM (or S) - The SCOM command causes typeout of system configuration information, including available device handlers, the skip chain order, and manual restart and dump procedures.

Example:

```
$SCOM

 SYSTEM INFORMATION - V4B  -  9/30/68

 17646 - BOOTSTRAP RESTART ADDRESS
 17636 - 1ST FREE LOCATION BELOW BOOTSTRAP
 1635 - 1ST FREE LOCATION ABOVE RESIDENT MONITOR
 135 - ADDRESS OF .DAT
 536 - ↑Q ADDRESS FOR MANUAL DUMP
 101 - START BLOCK FOR ↑Q DUMP AREA
 250 - KM9 START WITH RESTART ADDRESS IN LOCATION 0
 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
 DTA  DECTAPE: 3 FILES, INPUT/OUTPUT, ALL MODES, ALL FUNCTIONS
 DTB  DECTAPE: 2 FILES, INPUT/OUTPUT, IOPS MODES, LIMITED FUNCTIONS
 DTC  DECTAPE: 1 FILE, INPUT, IOPS MODES, LIMITED FUNCTIONS
 DTD  DECTAPE: 1 FILE, INPUT/OUTPUT, ALL MODES, ALL FUNCTIONS
 LPA  LINE PRINTER: OUTPUT, IOPS ASCII MODE, ALL FUNCTIONS
 CDB  CARD READER: INPUT, IOPS ASCII MODE, ALL FUNCTIONS
 SKIP CHAIN ORDER
      SPFAL
      DTDF
      RCSF
      CLSF
      LSDF
      RCSD
      RSF
      PSF
      KSF
      TSF
      DTEF
      SPE
      MPSNE
      MPSK
```

5.3.2.3 API ON/OFF - This command controls the status of the Automatic Priority Interrupt if available in the system. API ON enables the API; API OFF disables the API.

Example:

```
$API OFF
```

5.3.2.4 QDUMP (or ↑Q)* - In the event of an unrecoverable error, this command conditions the monitor to dump memory on the "save area" of the system tape (or other unit at the system device if used).

*The QDUMP and HALT commands are mutually exclusive and have no effect if a DDT load.

QDUMP forces automatic execution of the CTRL Q command (described later) on all non-recoverable error calls to the Monitor Error Diagnostic (.MED) program. It must be issued prior to the LOAD, GLOAD, DDT, or DDTNS command used to load the user program. (QDUMP issued prior to a GET, has no effect after the GET since the Monitor at CTRL Q time overlays the monitor primed by QDUMP.) 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.

5.3.2.5 HALT (or H)* - This command conditions the Monitor to print an error message and halt in the event of an unrecoverable IOPS error. Depressing the CONTINUE button reloads the Monitor. HALT must be issued prior to the LOAD, GLOAD, DDT, or DDTNS command. (HALT is issued prior to a GET, has no effect after the GET since the Monitor at CTRL Q time overlays the Monitor primed by the HALT command.)

5.3.2 6 INSTRUCT (or I) - The INSTRUCT command can be used in two ways: INSTRUCT alone causes a summary of Monitor commands to be printed on the Teletype; INSTRUCT ERRORS causes a summary of system error messages to be printed.
Example:

```
$INSTRUCT

 MONITOR: INFORMATION AND MODIFICATION COMMANDS
    LOG(L): USER COMMENTS TERMINATED BY ALTMODE
    SCOM(S): SYSTEMS INFORMATION
    INSTRUCT(I): LIST OF MONITOR COMMANDS
    INSTRUCT(I) ERRORS: DESCRIPTION OF ERROR CODES
    REQUEST(R), REQUEST(R) PRGNAM: .DAT SLOT USAGE
    REQUEST(R) USER: POSITIVE .DAT SLOT USAGE
    ASSIGN(A) DEVN A,B,.../ETC.: .DAT SLOT MODIFICATIONS
    DIRECT(D), DIRECT(D) M: DIRECTORY OF UNIT Ø OR M OF SYSTEM DEVICE
    NEWDIR(N) M: CLEAR DIRECTORY OF UNIT M OF SYSTEM DEVICE
    QDUMP(Q): SET TO SAVE CORE (↑Q) ON .IOPS ERROR
    HALT(H): SET TO HALT ON .IOPS ERROR
    ↑QN: SAVE CORE ON UNIT N
    GET(G) N: RESTORE CORE FROM UNIT N AND RESTART
    GET(G) N X: RESTORE CORE FROM UNIT N AND START AT X
    GET(G) N HALT(H):RESTORE CORE FROM UNIT N AND HALT
    API ON/OFF: CHANGE STATE OF API
    ↑C: RESTORE MONITOR
    ↑P: USER RESTART
```

*The QDUMP and HALT commands are mutually exclusive and have no effect if a DDT load.

```
MONITOR: PROGRAM LOADING COMMANDS AND PRGNAM FOR REQUEST COMMAND
   LOAD: LINK LOADER AND STOP
   GLOAD: LINK LOADER AND GO
   DDT: LINK LOADER WITH SYMBOLS AND GO TO DDT
   DDTNS: LINK LOADER WITHOUT SYMBOLS AND GO TO DDT
   MACRO: SYMBOLIC MACRO ASSEMBLER
   MACROA: ABBREVIATED SYMBOLIC MACRO ASSEMBLER
   F4: FORTRAN IV COMPILER
   F4A: ABBREVIATED FORTRAN IV COMPILER
   EDIT: SYMBOLIC CONTEXT EDITOR
   PIP: PERIPHERAL INTERCHANGE PROGRAM
   SGEN: SYSTEM GENERATOR
   DUMP: BULK STORAGE DEVICE DUMP
   UPDATE: LIBRARY FILE UPDATE
   CONV: 7-TO-9 CONVERTER
   PATCH: SYSTEM TAPE PATCH ROUTINE
   EXECUTE(E) FILE: LOAD AND RUN FILE XCT
   CHAIN: XCT CHAIN BUILDER
MONITOR: BATCH PROCESSOR
   BATCH(B) DV: ENTER BATCH MODE WITH DV AS BATCH DEVICE
        DV: PR = PAPER TAPE READER
            CD = CARD READER
   $JOB: CONTROL COMMAND WHICH SEPARATES JOBS
   $DATA: BEGINNING OF DATA - NOT ECHOED ON TELETYPE
   $END: END OF DATA
   $EXIT: LEAVE BATCH MODE
   ↑T: SKIP TO NEXT JOB
   ↑C: LEAVE BATCH MODE
```

5.3.2.7 REQUEST (or R) - The REQUEST command allows examination of the .DAT slots associated with various programs.* The command takes the following form:

REQUEST XXXXXX

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

Examples:

```
$REQUEST

 .DAT      DEVICE     USE

  -15       DTA2      OUTPUT
  -14       DTA1      INPUT
  -13       PPC0      OUTPUT
  -12       TTA0      LISTING
  -11       DTC1      INPUT
  -10       TTA0      INPUT
  -7        DTC0      SYSTEM DEVICE FOR .SYSLD
  -6        NONE      OUTPUT
```

*See Section 7.3 for .DAT slots used by system programs, their uses, and acceptable I/O handlers.

```
  -5        NONE      EXTERNAL LIBRARY FOR .LOAD
  -4        PRAØ      SYSTEM INPUT
  -3        TTAØ      TELEPRINTER OUTPUT
  -2        TTAØ      KEYBOARD INPUT
  -1        DTCØ      SYSTEM DEVICE FOR .LOAD
   1        DTAØ      USER
   2        DTA1      USER
   3        DTA2      USER
   4        TTAØ      USER
   5        PRAØ      USER
   6        PPAØ      USER
   7        DTA1      USER
   1Ø       DTA2      USER

$REQUEST MACRO

 .DAT      DEVICE    USE

  -13       PPCØ      OUTPUT
  -12       TTAØ      LISTING
  -11       DTC1      INPUT
  -1Ø       TTAØ      SECONDARY INPUT
  -3        TTAØ      CONTROL AND ERROR MESSAGES
  -2        TTAØ      COMMAND STRING
```

5.3.2.8 ASSIGN (or A) - 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 when 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*). If the third letter of a handler name is omitted, the letter A is assumed.

n, m are unit numbers (if none specified, 0 is assumed)

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

Examples:

$ASSIGN DTA0 -10, -6/PRA -5
(An equivalent command would be $ASSIGN DT -10, -6/PR -5)

$ASSIGN PPB -6/DTB2 3/DTB3 5

$ASSIGN DTA1 6, 7, 10

*See Section 7.3 for .DAT slots used by system programs, their uses, and acceptable I/O handlers. 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 since confusion occurs because of the lack of communication between the two interrupt handlers.

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

$ASSIGN NONE 4, 5, 10

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

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

5.3.2.9 DIRECT (or D) - The DIRECT command allows printout of the directory associated with any unit on the system device control (that is, eight units, 0 through 7, on DECtape control).

The command takes the following form:

DIRECT N

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

Example:

```
$DIRECT

 DIRECTORY LISTING
 .LIBR   BIN    141
 DDT9    BIN    142
 .LOAD   BIN    143
 CHAIN   BIN    144
 INTEGE  EAE    145
 INTEGE  NON    152
 REAL    EAE    164
 REAL    NON    215
 KM9     SYS      0
 .SGEN2  SYS     36
 SKPBLK  SYS     44
 UPDATE  SYS     45
 .SYSLD  SYS     56
 EXECUT  SYS     67
 ↑QAREA  SYS    101
 PATCH   SYS    652
 EDIT    SYS    657
 PIP     SYS    671
 F4      SYS    711
 MACRO   SYS    742
 F4A     SYS    771
 MACROA  SYS   1020
 DUMP    SYS   1044
 .SGEN1  SYS   1050
 CONV    SYS   1064
 70    FREE BLOCKS
```

5.3.2.10 NEWDIR (or N) n - This command refreshes the directory on the specified unit (n) of the system device control (unit 0 illegal).

5.3.2.11 GET (or G) - This command has three forms as follows: GET n, GET n xxxxx, or GET n HALT. The letter n is the number (0 through 7) of a unit on system device control that contains the ↑Q area to be retrieved, and xxxxx is a program starting address.

GET retrieves the core image (including the Monitor) stored on unit n on the system device control by CTRL Q commands, and restores it to memory. Control is transferred to xxxxx, if specified; execution halts if HALT was specified. To start in this case, the extended memory switch should be raised if the machine is greater than 8K, and the starting address should be placed in the ADDRESS switches and the START button depressed (PIC and API are enabled). If neither xxxxx not HALT are specified, the job is restored in memory and the resident Monitor waits in a Teletype loop with API and/or PI on for one of the following commands to be typed:

	↑P	(restart any system program and user programs which have issued an .INIT to the Teletype with a restart address.
or	↑T	(restarts DDT)
or	↑S	(starts a relocatable user program - used only if ↑Q had been executed at the completion of a link load when the loader was waiting for ↑Q to by typed.)

5.3.2.12 CHANNEL (or C) 7/9 - This command causes the default operation bit (.SCOM + 4, bit 6) to be cleared or set. If this bit is 0, then 7 channel operation is assumed by the MAGtape handler. If it is 1, the 9 channel is assumed. This default condition can also be set at system generation time by answering yes or no to the question

"SHOULD DEFAULT ASSUMPTION BE 7 CHANNEL MAGTAPE?"

5.3.2.13 339 (or 3) ON/OFF - This command tells the monitor that a 339 handler is to be loaded. A 30_8 register block is reserved for the push-down list. If a 339 display handler is loaded and the push-down list has not been reserved, an .IOPS 24 error will occur on the first .INIT to that handler. The default condition of the 339 ON/OFF bit (.SCOM+4, Bit 5) can be set at system generation time by answering YES to the question,

"SHOULD DEFAULT ASSUMPTION BE A 339 LOAD?"

5.3.2.14 VC 38 (or V) ON/OFF - This command causes the character display table for the VC 38 option to be set up prior to loading of any system or user programs. If this table is not set up and the user does not specify a character table in the first .INIT to the handler, the display handler will assume the presence of a VC 38 for text manipulation. The default condition of the VC 38 ON/OFF bit (.SCOM+4, bit 4) can be set at system generation time by answering YES to the question,

"SHOULD VC 38 CHARACTER TABLE BE LOADED?"

NOTE

If 339 is ON (bit 5 of .SCOM+4=1), then the address of the push-down list is initially set to zero. However, if VC 38 is ON (bit 4 of .SCOM+4=1), then the first register of the push-down list points to the starting address of the VC 38 table.

5.3.3 Control Character Commands

All control character commands recognized by the Monitor are summarized in Table 5-1. These commands (except RUBOUT) are formed by holding down the CTRL key while striking a letter key. The character or characters echoed on the Teletype and the resulting action is given in the table for each command.

Table 5-1
Control Character Commands

Command	Echo	Action
CTRL S	↑S	Starts user program after Linking Loader has brought it into core via a LOAD command.
CTRL C	↑C	Forces control back to Monitor which types MONITOR $ to indicate that it is awaiting a keyboard command. If the non-resident section of the Monitor was in core, the Monitor is not reinitialized; thus, previous conditions, such as .DAT slot assignments, remain as they were prior to CTRL C. If the non-resident section of the Monitor was not in core, it is brought in and all conditions revert to the standard.
CTRL T	↑T	CTRL T is applicable only when using DDT or when operating in the BATCH mode. If DDT is being used, CTRL T forces control back to DDT which types DDT > to indicate its readiness for another DDT command. All previous DDT conditions remain intact (for example, breakpoints, register modifications, etc.). When operating in BATCH mode, CTRL T causes a skip to the next job.
CTRL R	↑R	Allows the user to continue when an IOPS 4 (device not ready) error occurs. The user must first ready the device, and then type CTRL R.

Table 5-1 (Cont)
Control Character Commands

CTRL P	↑P	Forces control to address specified in the last .INIT command referencing Teletype. Used by system programs to reinitialize or restart.
CTRL Q n	↑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, CTRL Q requests can be made to DECtape only. The core image may be retrieved and reloaded by the GET command or examined by using the DUMP command to load the system Dump program. CTRL Q is honored whenever typed.
CTRL U	@	Cancels current line on Teletype (input or output).
RUBOUT	\	Cancels last character input from Teletype (not applicable with DDT).

5.4 OPERATING THE KEYBOARD MONITOR SYSTEM

The reader is referred to the Keyboard Monitor Guide (DEC-9A-MKFA-D) for detailed operating procedures for system programs in the Keyboard Monitor environment. The following PDP-9 ADVANCED Software System manuals contain additional detailed information on system programs.

Manual	Document Number
Utility Programs	DEC-9A-GUAB-D
MACRO-9 Assembler	DEC-9A-AMZA-D
FORTRAN IV	DEC-9A-KFZA-D

This section contains descriptions of loading the Keyboard Monitor, system generation, assigning devices, loading programs, and error detection and handling.

5.4.1 Loading the Keyboard Monitor

Each installation employing the DECtape version of the Keyboard Monitor must reserve tape unit 0 as the system device. This unit will contain the system tape, which includes the Monitor, the Input/Output Programming System, and all system and library programs needed by the user.

A System Bootstrap is supplied on 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 from the system device into lower core, and transfers control to it. The Monitor types

```
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, by setting the ADDRESS switches to 17646 of the highest memory bank, depressing I/O RESET and then START.

Ease of operation in an 8K DECtape environment will be enhanced by noting the following system characteristics, all of which are concerned with a judicious choice of I/O handlers (see Section 7.3 and 7.4) for system program operations:

a. .DAT slots -11 and -13 have been standardly assigned to handler DTB. to facilitate use of MACROA and FORTRAN IV A with DECtape input and DECtape output. Consequently, a reassignment of .DAT slots -11 and/or -13 is required in order to use MACRO or FORTRAN 4. If this reassignment is not performed the user may expect a .SYSLD 1 (core overflow) error.

b. .DAT slot 6 has been assigned to PPB (the punch handler for all data modes except IOPS ASCII) in order to facilitate loading PIP with the optimum combination of I/O handlers. Since paper tape reproduction of any paper tape may be accomplished using the I data mode, the user is seldom inconvenienced. If, however, PPA. is sometimes desired, the other positive .DAT slots will have to be examined for appropriate reassignment.

5.4.2 System Generation

PDP-9 installations with Disk but not DECtape receive PDP-9 ADVANCED software as a set of paper tapes which can be used to create a system tape. Installations that have DECtape will receive a system tape on DECtape.

System Generator (.SGEN) is a standard system program (bulk storage systems only) used to create new system tapes. Upon receiving 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; 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, that is,

```
$ASSIGN DTA0 -14, -10/DTA2 -15 (or DKD2 -15*) ↵
$SGEN ↵
```

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

---

*It is imperative that the Disk "D" handler be used when generating from DECtape to Disk to avoid core overflow. Conversely, generating from Disk to DECtape requires:

```
$ASSIGN DKA0 -14, -10/DTD2 -15 ↵
```

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. The PIC skip chain and API channels can be set up for the system device.

b. All device skips 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).

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 through -15) and slots 1 through 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. A good example of a complete system generation session is given in the Keyboard Monitor Guide (DEC-9A-MKFA-D).

The following paragraphs are intended to assist Keyboard Monitor users in their initial efforts at "tailor making" a system for their installation. The first and foremost rule before system generation is attempted involves getting a .SCOM printout ($S ↵ to the Monitor) and a .DAT slot printout ($R ↵ to the Monitor) in order to assist in determining two basic elements in the system; (1) skip chain content and order; and (2) .DAT slot assignments.

5.4.2.1 DECtape or DECtape/Disk Systems

An 8K, non-EAE, non-API, KSR33 DECtape system is sent to all DECtape or DECtape/Disk customers. Each customer with a core configuration of greater than 8K or who has either EAE or API or a KSR35 Teletype will want to go through system generation in order to tailor his installation for maximum efficient use. All customers who, upon examining the .SCOM printout, discover devices or options listed that are not present in their system may want to eliminate the irrelevant skips from the chain. Those with non-standard devices (A/D, for example) will want to expand the chain.

Listed below is the skip chain as it appears in the standard 8K DECtape system:

SPFAL	Power Fail
DTDF	DECtape Done
DSSF	Disk Done
DRSF	Drum Done
MTSF	Magnetic Tape Done on Error
LSDF	Line Printer Done
RCSF	Card Column Ready
RCSD	Card Done
CLSF	Clock Done

RSF	Ready Done
PSF	Punch Done
KSF	Keyboard Done
TSF	Teleprinter Done
DTEF	DECtape Error
MPSNE	Non-Existent Memory Reference
MPSK	Memory Protect Violation
SPE	Memory Parity Error
*DRNEF	Drum No Error

It is important that the above order remain intact even if deletions or additions are to be made. For example, given a PDP-9 without the Power Fail, Parity or Memory Protect options and having either card reader, line printer or magnetic tape, the skip chain should be generated as follows:

```
DTDF
DSSF
CLSF
RSF
PSF
KSF
TSF
DTEF
```

The position of a skip to be added to the chain varies with the nature of the device. For example, high data rate devices might best be placed at the top of the chain.

Listed below are the .DAT slot assignments as they appear in the standard 8K DECtape system:

.DAT	DEVICE	USE
-15	DTA2	OUTPUT
-14	DTA1	INPUT
-13	DTB2	OUTPUT
-12	TTA0	LISTING
-11	DTB1	INPUT
-10	TTA0	INPUT
-7	DTC0	SYSTEM DEVICE FOR .SYSLD
-6	NONE	OUTPUT
-5	NONE	EXTERNAL LIBRARY FOR .LOAD
-4	DTC2	SYSTEM INPUT
-3	TTA0	TELEPRINTER OUTPUT
-2	TTA0	KEYBOARD INPUT
-1	DTC0	SYSTEM DEVICE FOR .LOAD

*DRNEF, Skip on drum error flag not raised, is a good example of a negative skip, i.e., a skip on a flag not being raised. When specifying such a skip to .SGEN, a minus sign must precede the skip. It should be carefully noted that negative skips should only be included when the device is physically present in the PDP-9 system since the skip IOT otherwise becomes an effective NOP causing execution of the next instruction which is a JMP to the Monitor error routine (IOPS 3).

1	DTA0	USER
2	DTA1	USER
3	DTA2	USER
4	TTA0	USER
5	PRA0	USER
6	PPB0	USER
7	DTA1	USER
10	DTA2	USER

The following examples are variations on .DAT slot assignments* as a function of either core size or different peripherals.

a. Given an 8K system with line printer and card reader: LPA should be assigned to .DAT slot -12 and one of the positive slots, for example, 3, 7, or 10. CDB should be assigned to one of the positive slots.

b. Given a 16K (or greater) Disk/DECtape system, a suggested list of assignments might be as follows:

-15	DKA6
-14	DKA4
-13	DKA5
-12	TTA
-11	DKA4
-10	PRA
-7	DKC0
-6	DKA5
-5	NONE
-4	DKA5
-3	TTA
-2	TTA
-1	DKA0
1	DKA4
2	DKA5
3	DKA6
4	TTA
5	PRA
6	PPA
7	DTA1
10	DTA2

c. Given a 16K (or greater) DECtape system with magnetic tape, a suggested list of assignments might be as follows:

-15	DTA2
-14	DTA1
-13	DTA2
-12	TTA
-11	DTA1

* All installations with 16K or more core should assign the A versions of handlers to all .DAT slots.

-10	PRA
-7	DTC0
-6	DTA2
-5	NONE
-4	DTA2
-3	TTA
-2	TTA
-1	DTA0
1	DTA0
2	DTA1
3	DTA2
4	TTA
5	PRA
6	PPA
7	MTF1
10	MTF2

5.4.2.2 Paper Tape/Disk Systems

Three trays of paper tapes (24) representing an 8K, non-EAE, non-API, KSR33 Disk system are sent to all disk customers who do not have DECtape. Installations with greater than 8K, API, EAE, KSR35 or non-standard options will want to generate a system after the initial loading of their 8K system.

The discussion about skip chain order appearing in Section 5.4.2.1 (DECtape/Disk system) applies and should be read carefully.

Listed below are the .DAT slot assignments as they appear in the standard 8K Paper Tape/ Disk system:

.DAT	DEVICE	USE
-15	DKA6	OUTPUT
-14	DKA4	INPUT
-13	DKB5	OUTPUT
-12	TTA0	LISTING
-11	DKB4	INPUT
-10	TTA0	INPUT
-7	DKC0	SYSTEM DEVICE FOR .SYSLD
-6	NONE	OUTPUT
-5	NONE	EXTERNAL LIBRARY FOR .LOAD
-4	DKC5	SYSTEM INPUT
-3	TTA0	TELEPRINTER OUTPUT
-2	TTA0	KEYBOARD INPUT
1	DKC0	SYSTEM DEVICE FOR .LOAD
1	DKA4	USER
2	DKA5	USER
3	DKA6	USER
4	TTA0	USER
5	PRA0	USER
6	PPA0	USER
7	DKA1	USER
10	DKA2	USER

The following example is a variation of .DAT slot assignments* as a function of both increased core size and additional peripherals:

Given a 16K system with line printer, card reader and magnetic tape, a suggested list of assignments follows:

-15	DKA6
-14	DKA4
-13	DKA5
-12	LPA
-11	DKA4
-10	PRA
-7	DKC0
-6	DKA5
-5	NONE
-4	DKA5
-3	TTA
-2	TTA
-1	DKA0
1	DKA4
2	DKA5
3	DKA6
4	TTA
5	PRA
6	PPA
7	CDB
10	MTF1

5.4.3 Assigning Devices

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

The ASSIGN command (see Section 5.3.2.8) is used to attach hardware devices to the slots of the device assignment table. Figure 5-1 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 through 10) and system slot -2, -3 for Teletype I/O.

	.DAT Slot	Device Handler*	Unit	Use
.DATBG	-15	DTA	2	Output (EDITOR, UPDATE, CONVERTER, SYSGEN)
	-14	DTA	1	Input (EDITOR, UPDATE, CONVERTER, SYSGEN, DUMP)
	-13	DTB	2	Output (MACRO-9, FORTRAN IV)

*All installations with 16K or more core should assign the A version of handlers to all .DAT slots.

	-12	TTA		Listing (MACRO-9, FORTRAN IV, UPDATE, DUMP, CONVERTER)	
	-11	DTB	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	DTC	2	Input (Linking Loader)	All system programs
	-3	TTA		Teleprinter Output	
	-2	TTA		Keyboard Input	
	-1	DTC	0	System Library (Linking Loader)	
.DAT	.DAT				
	1	DTA0		User and PIP-9 .DAT slots	
	2	DTA1			
	3	DTA2			
	4	TTA			
	5	PRA			
	6	PPB			
	7	DTA1			
	10	DTA2			
.DATND=.					

Figure 5-1 Function of .DAT Slots in Keyboard Monitor System

5.4.4 Loading Programs in the Keyboard Monitor Environment

In the Keyboard Monitor environment, all system programs are called by unique keyboard commands (F4, MACRO, etc.). User programs are called by loading the Linking Loader or DDT (via LOAD, GLOAD, DDTNS, or DDT commands) and requesting it to load the desired program. In loading user programs, the main program is loaded first, followed by all required subprograms. By loading subprograms in the order of size (largest first, smallest last), the user has a better chance of satisfying core requirements for his programs in systems with extended core memory. (See Figure 5-2 for memory maps of programs loaded in the Keyboard Monitor system.)

When a keyboard command requests a new system 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.

*See Section 7.4 for a description of the handlers.

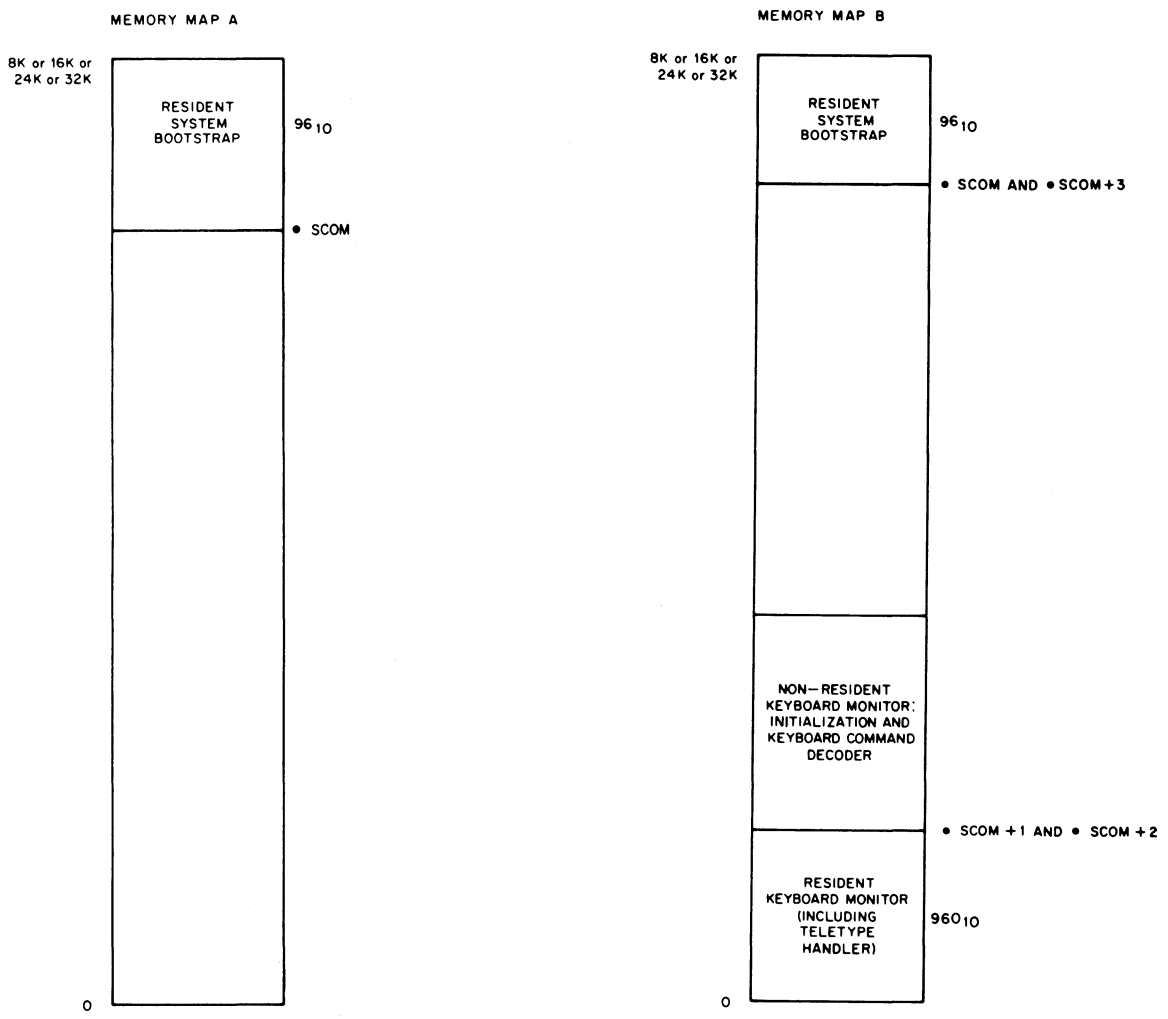
The System Loader is used to bring in all system programs, including the Linking Loader, and their associated device handlers. Once loaded, the Linking Loader is used to bring in user programs, their subroutines and device handlers.

The System Loader can print the same error messages as the Linking Loader (see Appendix D), 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. Detailed operating procedures for each system program are given in the Keyboard Monitor Guide (DEC-9A-MKFA-D).

5.4.5 Error Detection and Handling

Comprehensive error checking is provided by the Keyboard Monitor, the loaders, and the Input/Output Programming System. Detailed lists of errors that may occur are given in Appendices C, D, and E, respectively. After error messages are output, the user may optionally restart the system or user program, dump core, or return control to the System Bootstrap for re-initialization of the Keyboard Monitor (see Section 5.3.2.11). If QDUMP has been issued prior to execution of this program, an automatic CTRL Q (dump the current job, in core image, onto prespecified blocks of the system device) takes place before control is returned to the System Bootstrap. The number of the unit to be used as the dump device must be typed by the user after the error typeout. This dumped file can be selectively listed by the system Dump program. If HALT has been typed prior to program execution, the program stops after error message typeout allowing manual memory cell examination, manual restart or core dump.

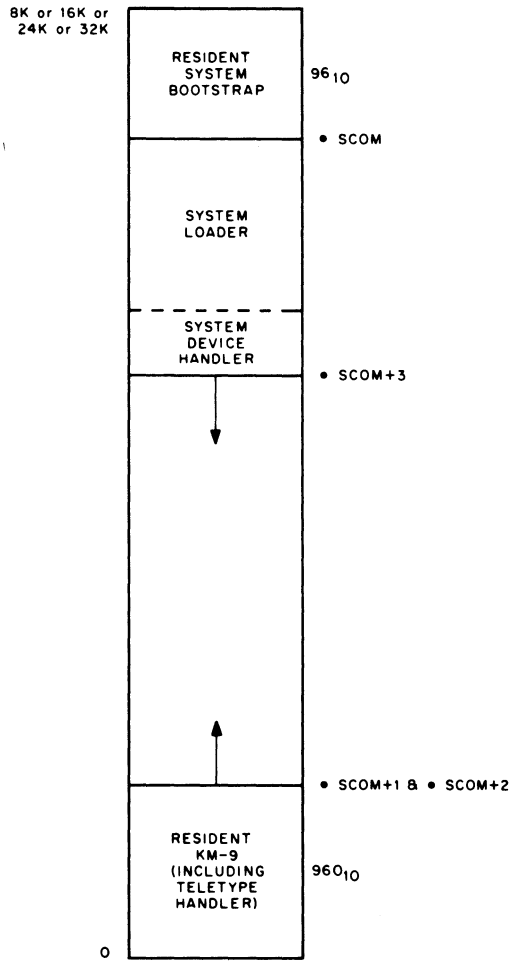


The System Bootstrap is loaded via the paper tape reader in HRM mode.

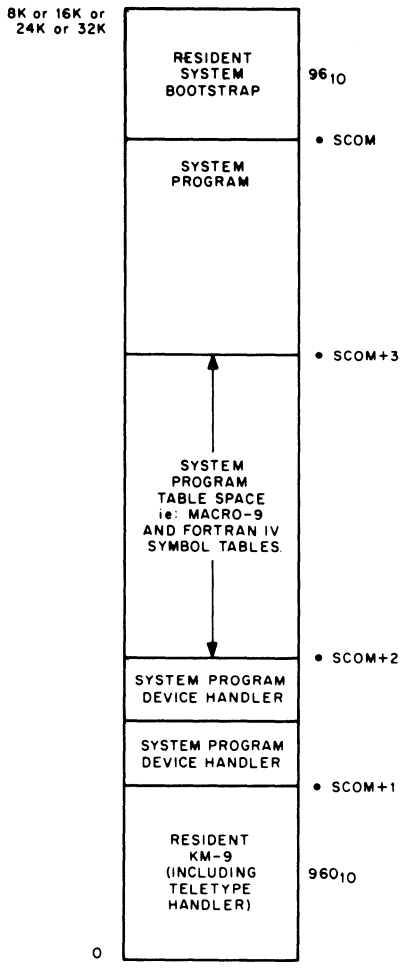
The System Bootstrap loads (DUMP mode) the Keyboard Monitor (resident and non-resident) from the system device.

Figure 5-2 Keyboard Monitor System Memory Maps

MEMORY MAP C



MEMORY MAP D (SYSTEM PROGRAM IS NOT LINKING LOADER)



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

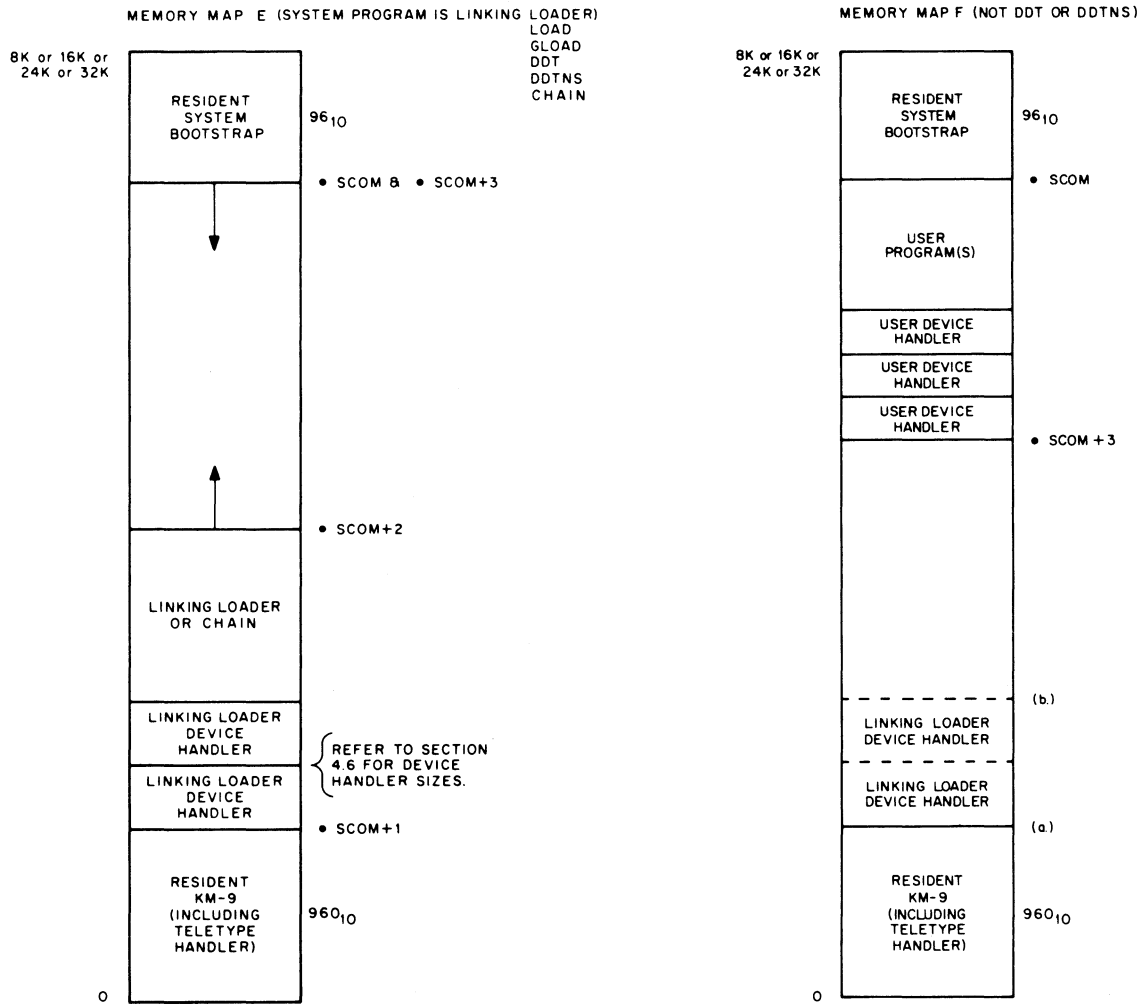
The System Loader, during loading of a system program from the system device, builds the loader (GLOBAL) symbol table down from .SCOM+3 and the programs up from .SCOM+2.

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 Memory Map B where the system bootstrap reinitializes the Keyboard Monitor.

Refer to Section 7.4 for the sizes of the device handlers that may be associated with the .DAT slots used by the System program.

Figure 5-2 Keyboard Monitor System Memory Maps (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.

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

The Linking Loader, during loading of user programs down from .SCOM+2, 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.

.EXIT from user program takes the process back to Memory Map B where the system bootstrap re-initializes the Keyboard Monitor.

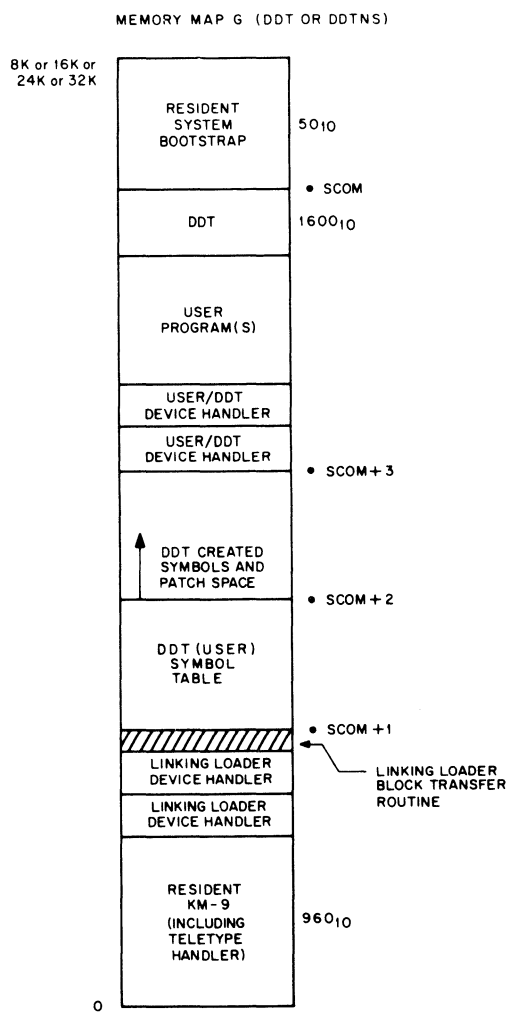
Refer to Section 7.4 for sizes of device handlers.

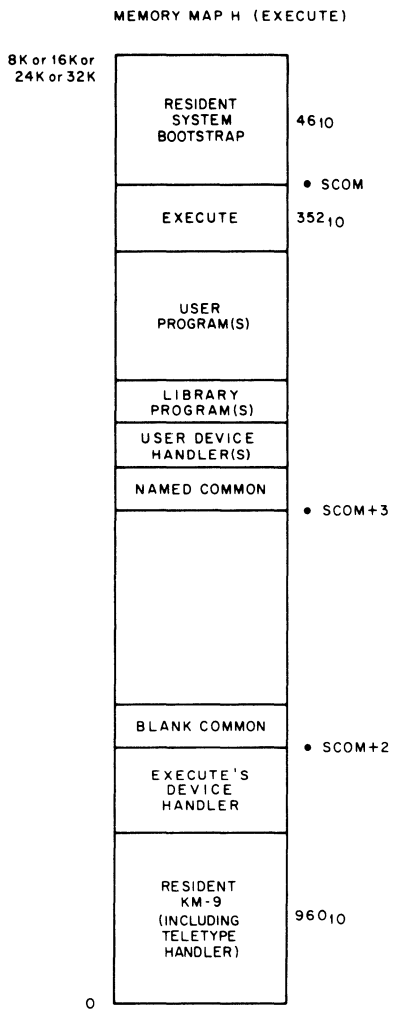
.SCOM+1 and .SCOM+2 both point to one of two places and non-BLOCK DATA COMMON (FORTRAM IV or MACRO-9) output may make use of core as low as they point.

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 Loader.

Figure 5-2 Keyboard Monitor System Memory Maps (Cont)





.EXIT from the user program takes the process back to Memory Map B where the system bootstrap re-initializes the Keyboard Monitor.

Refer to Section 7.4 for sizes of device handlers.

Non-BLOCK DATA COMMON (FORTRAN IV) or 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 symbol table if a DDTNS load.

The System Loader loads the I/O handler needed by EXECUTE above the resident portion of the Monitor. It then modifies the System Bootstrap to bring in EXECUTE in dump mode just below the Bootstrap.

When the user typed in the name of the XCT file to be run, the non-resident portion of the Monitor stored the file name in .SCOM + 7, 10 and 11. .EXECUTE now locates this file and brings in the first chain.

FORTRAN programs pass on data in blank common starting at .SCOM + 2. Macro programs pass on data between .SCOM + 2 and .SCOM + 3.

A call from the running chain to bring in another chain is effected by transferring control back to Execute.

.EXIT from the user chain takes the process back to PHASE 2 where the System Bootstrap re-initializes the Keyboard Monitor.

Figure 5-2 Keyboard Monitor System Memory Maps (Cont)

5.5 BATCH PROCESSING

The Batch Processor portion of the Monitor allows user commands to come from the paper tape reader or card reader instead of the Teletype, allowing many programs to be run without operator intervention. All Monitor commands read on the batch device are echoed on the Teletype. Monitor commands that are peculiar to the Batch Processor include the following:

Command	Function
BATCH (B) dv	Enter Batch mode with dv as batch device; dv can be typed as PR, for paper tape reader, or CD, for card reader NOTE The following special characters can be keypunched using the indicated card codes. Back arrow (←) = 0-5-8 punch ALT MODE (ESC) = 12-1-8 punch
$JOB	Used to separate jobs (the loading of any system or user program constitutes a single job).
$DATA	Beginning of data - all inputs up to $END are not echoed on the Teletype.
$END	End of data.
$EXIT	Leave Batch mode.

NOTE

The following commands are illegal when operating in Batch mode: QDUMP, HALT, GET (all forms), BATCH, LOAD, DDT, and DDTNS.

Special Batch Processor control characters include the following:

CTRL T (echos ↑T)	Skip to next job.
CTRL C (echos ↑C)	Leave Batch mode.

To use the Batch Processor, proceed as follows.

a. Load the batch tape or deck into the batch device.

b. Type BATCH (or B) dv on the keyboard, where dv is PR or CD.

When operating in Batch mode, the Keyboard Monitor has the following operational changes.

a. Any ASSIGN command that references the batch device (any handler) will be assigned to the batch device handler.

b. Any REQUEST command will print the batch device handler PR* or CD* (whichever applies).

c. When the non-resident Monitor is reloaded, it interprets batch communication bits in the top register of core (177777, 377777, 577777, or 777777):

Bit 0	1 = Batch mode 0 = Non-Batch mode
Bit 1	1 = $JOB command in 0 = Search for $JOB
Bit 2	1 = CD is batch device 0 = PR is batch device

When an error occurs in a job, the non-resident Monitor is reloaded and the Batch Processor skips to the next $JOB command on the batch device.

The following example was produced under control of the Batch Processor. Underlined commands are on paper tape. ALT MODE termination is indicated with a ⊗.

```
$↑C

MONITOR V4B

$BATCH PR

MONITOR V4B                          This command causes all subsequent commands
                                     to come from the paper tape reader.
$$JOB TEST BATCH

$PIP

PIP V7A

>N DK4

>T DK4 TEST SRC (A) ← PR             The entire program to be compiled below ap-
$DATA                                pears on the paper tape between $DATA and
$END                                 $END.

MONITOR V4B

$$JOB

$ASSIGN TT -12

$R F4

 .DAT      DEVICE     USE

  -13       DKA5      OUTPUT
  -12       TTAØ      LISTING
  -11       DKA4      INPUT
  -3        TTAØ      CONTROL AND ERROR MESSAGES
  -2        PR*Ø      COMMAND STRING

$F4
```

```
FORTRAN 4 V4A
>S,L,B←TEST(X)

END PASS1

C
C      TEST OF BATCH PROCESSOR
C
            DO 1 I=1,10
1           WRITE (4,100) I
100         FORMAT (6X,I3)
            STOP 12345
            END


   TEST    17777
   .1      00012
   I       00043
 * .FW     00036
   .100    00021
 * .FE     00037
 * .FF     00040
 * .ST     00041
 * .FP     00042

MONITOR V4B

$$JOB

$GLOAD

LOADER V3A
>TEST (X)
 TEST    37573
 BCDIO   34600
 STOP    34565
 SPMSG   34472
 FIOPS   33736
 OTSER   33642
 REAL    32654

        1
        2
        3
        4
        5
        6
        7
        8
        9
       10
```

FORTRAN program to list numbers from 1 through 10.

Program execution begins here.

```
STOP  Ø12345
MONITOR V4B
$$JOB
$$EXIT

MONITOR V4B                    Control is returned to Teletype at this point.
$
```

5.6 DECTAPE FILE ORGANIZATION

DECtape can be treated either as a non file-oriented medium or as a file-oriented medium, as described in the following paragraphs.

5.6.1 Non-File-Oriented DECtape

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 of 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 (256 th) 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: 001005, 776773. The data link of this EOF DECtape block is calculated in case another file is to be recorded. Note that the simulated end-of-file is identical whether executing a .CLOSE in a file-oriented or non-file-oriented environment.

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

5.6.2 File-Oriented DECtape

Just as a REWIND command declares 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 containing file information exists on the DECtape. A directory listing of any DECtape so recorded is available via the (L)ist command in PIP or the (D)irect command in the Keyboard Monitor. A fresh directory may be recorded via the (N)ewdir command in the Keyboard Monitor or PIP or by the

N or S switch in PIP.

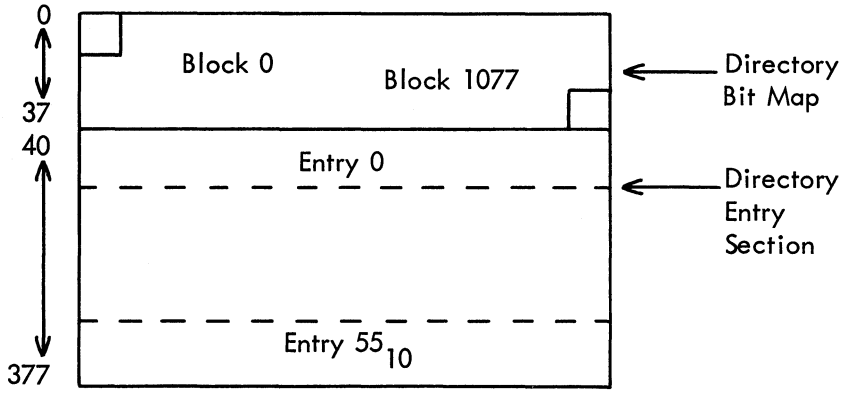
The directory of all DECtapes except system tapes occupies all 400_8 cells of block 100_8. It is divided into two sections: (1) a 40_8 word Directory Bit Map and (2) a 340_8 word Directory Entry Section.

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.

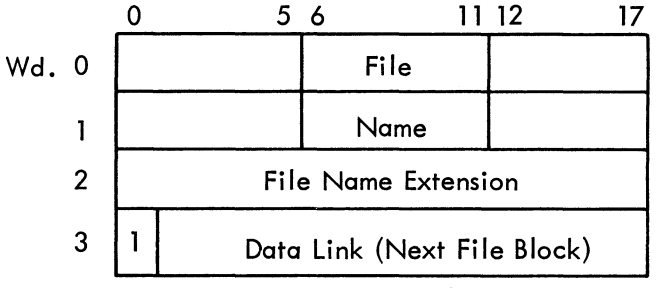
The Directory Entry Section provides for a maximum of 56_{10} files on a DECtape (24_{10} on a system tape). A four-word entry exists for each file on DECtape, where each entry includes the 6-bit trimmed ASCII file name (6 characters maximum), and file name extension (3 characters maximum), a pointer to the first DECtape block of the file, and a file active or present bit.

On a system tape only the first 200_8 words are used as a 24 file directory. Cells $0\text{-}37_8$ constitute the System Tape Directory Bit Map and cells $40\text{-}177_8$ contain 24 file Directory Entry Section. The second 200_8 words of DECtape block 100_8 contain basic system directory information (blocks occupied by system programs), used by KM-9, PIP-9 and SGEN-9.

DECTAPE DIRECTORY



A DIRECTORY ENTRY

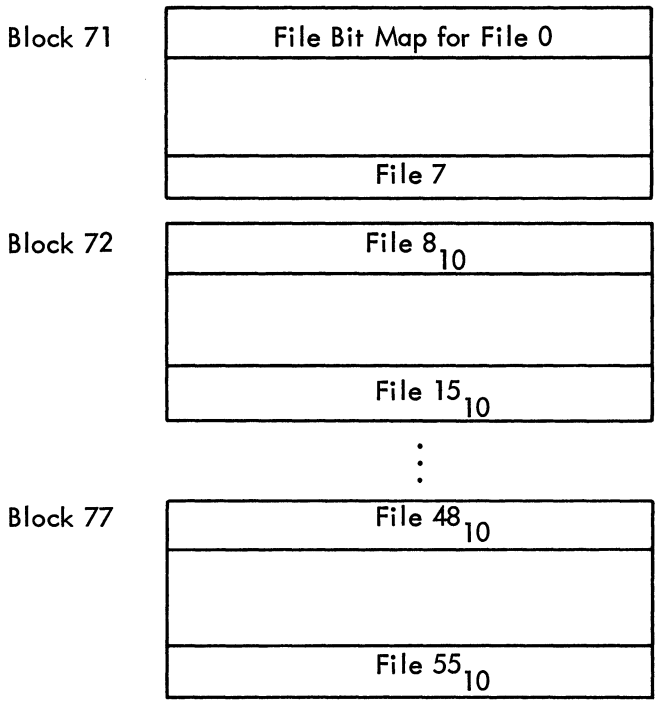


Sign Bit: 1 = File Active

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

Additional file information is stored in blocks 71 through 77 of every file-oriented DECtape (blocks 71 through 73 of a system tape). These are the File Bit Map Blocks. For each file in the directory, a 40_8 word File Bit Map is reserved in block 71 through 77 as a function of file name position in the Directory Entry Section of block 100. Each block 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, that is, on the .CLOSE of the new file.

File Bit Map Blocks



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

Staggered recording (at least every fifth 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 five 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 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.

Section 2.3.1 of this manual discusses 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 and (2) dump mode data - Data Mode.

In IOPS or image modes, each line (including header) is packed into the DECtape buffer. A 2's 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, that is, if the buffer will not contain it, the line 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.

5.7 INTERIM DISK SYSTEM

The PDP-9 Interim Disk System (Storage Unit RB09 and Disk Control RC09) is an adaptation of the PDP-9 ADVANCED Software System for DECtape and includes the Keyboard Monitor, all PDP-9 ADVANCED Software System programs, a disk system bootstrap, disk handlers DKA, DKB, DKD, and DKC, and either DSKSAV or DSKPTR, utility programs to save and restore disk data to/from DECtape or paper tape, respectively.

In the interim system, the disk is treated as six logical units (0, 1, 2, 4, 5, 6), each containing 576_{10} blocks of 256_{10} words per block. Since each of the RB09 disk tracks contains 80 sectors of 64_{10} words, a track is equivalent to 20 logical DECtape blocks. The table below lists unit-track correspondence:

Disk Unit	Track, Sector (Inclusive)
DK0	0,0-28,63
DK1	30,0-58,63
DK2	60,0-88,63
DK4	100,0-128,63
DK5	130,0-158,63
DK6	160,0-188,63

The minimum hardware configuration required for the PDP-9 Disk System is a basic PDP-9 (8K, Teletype, high-speed reader/punch), RB09 Disk Unit with Disk Control RC09, and Direct Memory Access Channel Multiplexer Adapter, Type DM09A.

5.7.1 The Disk as System Device

The PDP-9 ADVANCED Software System resides on tracks 0 through 28 (Disk Unit 0) with the Keyboard Monitor starting at tracks 0, sector 0. All absolute system programs, directory and File Bit Map information occupy the same logical block positions on disk as they occupy on DECtape. Blocks on tracks 0 through 28 not used for system storage are available for user files provided that the disk protection switches have not been activated. It is recommended however, that once the disk system is loaded, the protection switches be activated for tracks 0 through 29.

Much of the description on DECtape file organization (Section 5.4.1), applies to the Interim Disk System as well, since the four disk handlers (DKA, DKB, DKD, and DKC) are modified versions of DECtape handlers DTA, DTB, DTD, and DTC.

5.7.2 Disk File Organization

In the interim system the PDP-9 Disk is divided into 6 units (each equivalent to a DECtape in length) therefore, each unit has its own directory of files. Each unit, each directory and the file therein is treated as distinct within the system.

Disk files may be recorded in any one of the five standard PDP-9 ADVANCED Software System data modes. MTAPE .READS, .WRITES may be considered as creating a sixth data mode as far as data organization of MTAPE files on Disk is concerned.

Disk file data organization may take one of three forms:

a. Packed lines: IOPS ASCII, IOPS Binary, Image Alphanumeric, Image Binary

b. Packed dump areas: Dump Mode

c. MTAPE transfers

In IOPS or Image modes, lines, including headers, are packed back to back on a logical disk block (256_{10} words or 4 disk sectors). A 2's complement checksum is computed by the disk handler and stored in the header of each line. When a line is encountered which will exceed the current block capacity, that line is placed in the next free logical disk block. A data link to the new block is stored in the final word of the filled block. Unused space in any file block is equal to zero. The final line of the file, whatever the data mode, is an EOF header of two words: 1005, 776773. The data link of the last block of any file except a .MTAPE transfer is 777777.

In Dump Mode, the transfer word count is always taken from the I/O Macro. If a word count greater than 255_{10} is specified (again the last word of a logical block must contain a data link), the disk handler will transfer 255_{10} word increments onto the disk in linked logical blocks. If some number of words less than 255_{10} remain as the final element of the Dump Mode .WRITE, they will be stored on the next logical disk block which will then be filled on the next .WRITE or with an EOF if the next command is .CLOSE. Disk storage is optimized in Dump Mode since data is packed and no headers are used.

An MTAPE .WRITE causes transfer of a 255_{10} word logical disk block starting at logical block 0 of the specified disk unit. A disk unit used for MTAPE transfers contains no directory information. No packing is performed by the disk handler, it is the user's responsibility to format the record in whatever way he chooses. When a .CLOSE is issued, the usual EOF line is recorded as the last block of an MTAPE file. The data link is computed in case another MTAPE file is to be written and is stored in the EOF (or last) block of the present file.

5.8 DISK SYSTEM OPERATION

Operation of the PDP-9 Disk System is essentially the same as operation of the DECtape system. To begin disk system operation, the paper tape Disk Bootstrap (.DKSBT) is loaded into core memory (HRM: 17637) and automatically loads the Keyboard Monitor into core from disk track 0. (The bootstrap restart address is 17646.)

PDP-9 ADVANCED Software for the disk is distributed as an 8K system in one of two forms:

a. 1 DECtape
b. 3 trays (24_{10}) of paper tapes

} 8K, non-API, non-EAE, KSR33 systems

DSKPTR is the utility program for loading the disk system from paper tape, a 30 minute operation. It is additionally capable of loading or punching any logical disk unit or 6K elements thereof as a function of ACS settings on the PDP-9 console.

The data format of tapes generated or loaded via DSKPTR is shown in Figure 5-3. Each tape consists of 6K or 24 logical 256_{10} word sequential disk blocks. Each data block on paper tape is preceded by six blank frames to aid visual identification.

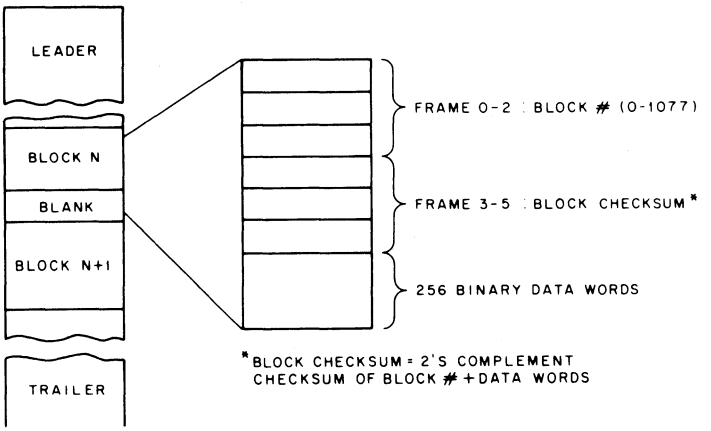


Figure 5-3 Paper Tape Block Format

5.8.1 Paper Tape Load Procedure

a. Place the paper tape of DSKPTR in reader with address switches set to 17720.

b. Press I/O RESET and READIN.

c. When the program is loaded, it will display its title, DSKPTR and skeletal directions on teletype.

SET: *ACS0=0	Paper tape to disk
ACS15-17 = Unit #	0, 1, 2, 4, 5, 6

d. Place the first disk system tape in reader and press CONTINUE.

e. "HIT CONTINUE FOR NEXT TAPE" will be typed on the teletype when the tape has been read correctly. Load the remaining 23 tapes in a similar fashion. Note, any number of tapes up to the full 24 may be loaded in any order.

The following errors may be displayed on the teletype during loading:

1. Unit Error: ACS15-17 = 3 or 7.

   Action:
   a. Reset ACS15-17 = 0, 1, 2, 4, 5, 6
   b. Press CONTINUE

2. Reload Tape: Reader end-of-tape condition at illegal point.

   Action:
   a. Reload tape
   b. Press CONTINUE

3. Input Checksum Error: Block checksum incorrect.

   Action:
   a. Reload tape from beginning, OR position tape to the front of block in error, or leave tape in current position.*

*ACS 0-17 = 0 for full system load onto disk.

b. Press CONTINUE

f. When the disk system is loaded, set the write protect switches for tracks 0 through 29.

g. Load the Disk Bootstrap into the reader with address switches set to 17637.

h. Press I/O RESET and READIN. The PDP-9 Keyboard Monitor will be loaded from the disk and will type:

$ MONITOR

The system is now ready for operation. Refreshing all user disk unit directories is imperative at this point. This is done with the KM-9 Newdir (N) command, for example,

$ N 1 ↵

refreshes disk unit 1. This command should be issued for units 1, 2, 4, 5, and 6 before further operation.

DSKPTR may also be used to punch out 6K areas of the disk. To do so, follow steps a. and b. above under loading procedure.

c. ACS 0 = 1
ACS15-17 = Unit #
**ACS5-14 = Logical Block # 0, 30, 60.... 1050_8

d. Press CONTINUE

e. See step e. above.

The following errors may be displayed on Teletype during disk to punch output:

1. Unit Error: (See loading procedure)

2. Reload Tape: Punch out of tape

   Action:
   a. Reload punch
   b. Press CONTINUE
   c. Splice Tape

3. Disk Error: Probably parity error. AC will contain disk status. Look at disk control panel for AC bit meaning.

   Action: Since DSKPTR will have tried 8 times to read disk
   a. Reset at 16000 or
   b. Press CONTINUE to accept data.

---

*Data block in error will be accepted as is.

**ACS5-14 = 0 causes punching of all 24_{10} tapes for disk unit (ACS15-17). ACS5-14 = NON 0 causes punching of one 6K paper tape starting at the logical block specified. Note, this feature is particularly useful when PATCH is used for system program modification and a single back up paper tape of the associated disk area is desired. See Appendix F for program location on disk.

5.8.2 Disk System Generation

Once the Disk System is loaded and disk units have been refreshed, it may be necessary to generate a system more appropriate to the needs of the installation. For example, since the released system is an 8K system, all 16, 24 and 32K installations will want to generate a system for the appropriate core size.

Operation of SGEN (PDP-9 System Generator Program) is described in Section 5.4.2 of this manual and in the Keyboard Monitor Guide (DEC-9A-MKFA-D). Disk unit 0 should be assigned to .DAT slot -10 and -14. The output device (presently disk unit 6) appears in .DAT slot -15.

When system generation has been completed, PIP-9 should be called into memory to copy disk unit 6 onto disk unit 0 from which the new system may now be operated. The PIP command to be used is

C DK0 (H) ← DK6↵

DSKPTR may be used to produce a backup of the system on paper tape.

5.8.3 Disk System Generation from DECtape

The PDP-9 installations with both disk and DECtape receive the standard Keyboard Monitor system on DECtape. It is an 8K system which when loaded from DECtape, unit 0 with the DECtape bootstrap (I/O RESET followed by READIN with the address switches set to 17637) allows a disk system to be generated.

Once the Monitor is in from DECtape and has typed:

MONITOR

$

the REQUEST and SCOM commands should be issued to the Monitor for system information and the following I/O handler assignments should be made:

$A DTA0-10,-14/DKD0-15↵

System Generator should then be called:

$SGEN↵

Once disk system generation is complete, the write protect switches for tracks 0 through 29 should be activated.

The Monitor system may now be loaded from disk unit 0 using the Disk Bootstrap (HRM:17637). When the Monitor is in core, it is imperative to refresh all user disk unit directories. This is done with the Keyboard Monitor Newdir (N) command, for example,

$N 1↵

refreshes disk unit 1. This command should be issued for units 1, 2, 4, 5, and 6, before further operation.

5.8.4 Disk System Save/Load from DECtape

Once a disk system has been generated, it is valuable to produce a backup system for DECtape for rapid disk system restoration when necessary.

DSKSAV is the utility program for saving or loading the PDP-9 ADVANCED Software Disk System to/from DECtape. System loading or saving with DSKSAV is a 55 second operation. It is additionally capable of loading or saving on DECtape, any other logical disk unit as a function of ACS settings on the PDP-9 console. The save/load procedure is as follows:

a. Place paper tape of DSKSAV in reader with address switches set to 17720. (Restart = 16000)

b. Press I/O RESET and READIN.

c. When the program is loaded, it will type its title, DSKSAV, and brief directions on the console Teletype. The program is stopped to allow ACS settings.

Set:	*ACS0=0	DECtape to Disk (LOAD)
	ACS0=1	Disk to DECTAPE (SAVE)
	ACS15-17=Unit#	0, 1, 2, 4, 5, 6

d. Be sure the DECtape unit selection is identical to ACS 15 through 17 and press CONTINUE.

The following errors may be displayed on the Teletype during loading:

1. Unit Error: ACS15-17 = 3 or 7.

   Action:
   a. Reset ACS 15-17 = 0, 1, 2, 4, 5, 6
   b. Press CONTINUE

2. Disk Error: Probably illegal disk address (for example, attempt to load disk with write protect switch enabled).

   AC contains disk status. Examine disk control panel for AC bit meaning.

   Action: Correct disk problem if possible and press CONTINUE.

3. DECtape Error: Probably end zone error. AC contains DECtape status. Examine DECtape control panel for AC bit meaning.

   Action:
   a. If DECtape is in the forward end zone, press CONTINUE.
   b. If DECtape is in the far end zone (beyond block 1077), position tape out of end zone and press CONTINUE.
   c. If parity or mark track error**, press CONTINUE to restart transfer from the beginning, or set ACS0 through 17 = 0 and press CONTINUE to retry read one more time.

DSKSAV may also be used to save or load other logical units of the disk. This has proved to be very useful since it provides rapid DECtape backup of any or all disk data files. To do so, follow steps (a) and (b) above under loading procedure.

---

*ACS0-17= 0 for full system load onto disk.

** DSKAV will have tried to read four times before the message.

a. ACS0 = 1
   ACS15-17 = Unit #

b. Press CONTINUE

c. See step e. under load procedure.

5.9 MAGNETIC TAPE SYSTEMS

PDP-9 ADVANCED Software provides for IBM-compatible magnetic tape as a file-structured medium and as a full-scale system device. The magnetic tape handlers communicate with a single TC-59 Tape Control Unit (TCU). Up to eight magnetic tape transports may be associated with one TCU; these may include any combination of transports TU-20 and TU-20A.* At least two transports are required for full system operation.

There are a number of major differences between magnetic tape and other bulk-storage devices (for example, DECtape or Disk); these differences affect the operation of the device handlers. Magnetic tape is well-suited for handling data records of variable length; such records, however, must be treated in serial fashion. The physical position of any record may be defined only in relation to the preceding record. Block-addressable devices are most economically used in transferring records having fixed lengths that are hardware-constrained. Using such devices, the absolute physical location of any record is program-specifiable. Because of the serial character of data blocks as they are recorded on magnetic tape and because of the presence of blocks of unknown length, three techniques available in I/O operations to block-addressable devices are not honored by the magnetic tape handlers:

a. The user cannot specify physical block numbers for transfer. In processing I/O requests that have block numbers in their argument lists (.TRAN, for example), the handler ignores the block-number specification.

b. The only area open for output transfers in the file-structured environment is that following the current logical end of tape. The exception to this rule is in the recording of the File Directory as explained below.

c. Only a single file may be open for transfers (either input or output) at any time on a single physical unit.

*A detailed description of the TCU is contained in the TC-59 Instruction Manual (DEC-9A-I3BA-D) and in the PDP-9 User Handbook (F-95).

5.9.1 File Organization

The discussion below applies particularly to MTA, the most general and the largest of the magnetic tape handlers. Response to data-mode specification is identical for handlers MTB., MTC., and MTD., except for the modes that are illegal in these more limited handlers. MTA. is a full device handler that will read and write tape in both the file-structured and non-file-structured fashion. It will honor all CAL functions and requests for transfer in all data modes. The characteristics of data recorded on tape are dependent upon a decision as to the file structuring. This decision, in MTA., is based upon much the same criteria as in other bulk-storage handlers. If the first I/O request after an .INIT is .READ, .WRITE, or any .MTAPE command, the referenced unit is presumed to be non-file-structured. If the first request after an .INIT is .SEEK, .ENTER, .CLEAR, or .OPER, then the referenced unit is treated as a file-structured device.

5.9.1.1 Non-File-Structured Data Recording - The treatment of data to be recorded or read in non-file-structured fashion has two primary objectives. It is intended to satisfy the requirements of the FORTRAN programmer while still providing the assembly language programmer maximum freedom in the design of his tape format.

Magnetic tape data, written in the non-file-oriented environment, differs in two important respects from data recorded by means of file-oriented I/O requests. In the first place, no handler-supplied supplementary information is written on the tape. No reference is made, for example, to a file directory, and block-control data (see below) is never written. Secondly, no blocking (or packing) of lines is performed by the handler. Each .WRITE (or .READ) request causes direct data transfer between the user's line buffer and the TCU. No buffering or editing of any kind is done (in IOPS and Image modes) by MTA. Each .WRITE (or .READ) issued results, in general, in the transfer of exactly one physical record to (or from) tape.*

5.9.1.2 File-Structured Data Recording - The programmer can make the fullest possible use of those features peculiar to magnetic tape by employing non-file-oriented transfer techniques. On the other hand, he has little recourse to the powerful file-manipulation facilities available in the system. File-structured I/O brings to bear the whole body of file-system software, gives true device independence to the magnetic-tape user, and allows extensive use of the storage medium with a minimum of effort.

---

*An exception to this rule is the Dump Mode transfer. The handler will pack output data into, and extract input data from, an internal buffer in this mode.

5.9.1.3 Block Format - Every block recorded by MTA. (with the exception of end-of-file markers, which are hardware-recorded) in file-structured mode includes a two-word Block Control Pair and not more than 255_{10} words of data.

The Block Control Pair serves three functions: it specifies the character of the block (label, data, etc.), provides a word count for the block, and gives an 18-bit block checksum. The Block Control Pair has the following format:

Word 1:

Bits 0 through 5: Block Identifier (BI). This 6-bit byte specifies the block type. Values of BI may range from 0 to 77_8. Current Legal values of BI, for all user files, are as follows:

BI Value	Block Type Specified
0	User-File Header Label
1	User-File Trailer Label
2	User-File Data Block

Bits 6 through 17: Block Word Count (BWC). This 12-bit byte holds the 2's complement of the total number of words in the block (including the Block Control Pair). Legal values of BWC range from -3 to -401_8.

Word 2:

Bits 0 through 17: Block Checksum. The Block Checksum is the full-word, unsigned, 2's complement sum of all the data words in the block and word 1 of the Block Control Pair.

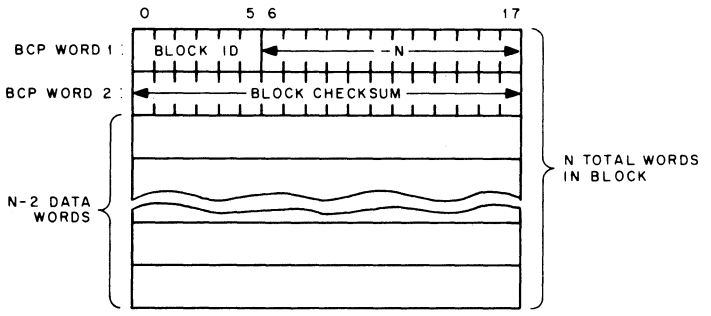


Figure 5-4 Block Format, File-Structured Mode

5.9.2 File Identification and Location

One of the main file-manipulation functions of the handler is that of identifying and locating referenced files. This is carried out by two means: first, names of files recorded are stored in a file directory at the beginning of the tape; and second, labels integral to the file are recorded with the file itself.

5.9.2.1 Magnetic Tape File Directory - The directory, a single-block file (and the only unlabeled file on any file-structured tape), consists of the first recorded data block on the tape. It is a fixed-length block with a constant size of 257_{10} words and the following characters:

a. Block Control Pair (words 1 and 2)

Word 1:

Block Identifier = 74_8 = File Directory Data Block.
Block Word Count = $401_8 = 7377_8$.

Word 2:

Block Checksum: As described.

b. Active File Count (Word 3, Bits 9 through 17) 9-bit one's complement count of the active file names present in the File Name Entry Section (described below).

c. Total File Count (Word 3, Bits 0 through 8) 9-bit one's complement count of all files recorded on the tape, including both active and inactive files, but exclusive of the file directory block.

d. File Accessibility Map (Words 4 through 17): The File Accessibility Map is an array of 252_{10} contiguous bits beginning at bit 0 of word 4 and ending as bit 17 of word 17. Each of the bits in the Accessibility Map refers to a single file recorded on tape. The bits are assigned relative to the zeroth file recorded; that is, bit 0 of word 4 refers to the first file recorded; bit 1, word 4, to the second file recorded; bit 0, word 6, to the 37_{10} file recorded; and so on, for a possible total of 252_{10} files physically present.

A file is only accessible for reading if its bit in the Accessibility Map is set to one. A file is made inaccessible for reading (corresponding bit = 0) by a .DLETE of the file, by a .CLOSE (output) of another file of the same name, or by a .CLEAR. A file is made accessible for reading (corresponding bit = 1) by a .CLOSE (output) of that file. Operations other than those specified above have no effect on the File Accessibility Map.

e. File Name Entry Section (Words 18 through 257): The File Name Entry Section, beginning at word 18 of the directory block, includes successive 3-word file name entries for a possible maximum of 80 entries. Each accessible file on the tape has an entry in this section. Entries consist of the current name of the referenced file in standard DEB format: file name proper in the first two words, extension in the third word; 6-bit trimmed ASCII characters, left-adjusted and, if necessary, zero-filled.

The position of a file name entry relative to the beginning of the section reflects the position of its accessibility bit in the map. That bit, in turn, defines the position of the referenced file on tape with respect to other (active or inactive) files physically present. Only active file names appear in the entry section, and accessibility bits for all inactive files on the tape are always set to zero; accessibility bits for all active files are set to one.

To locate a file on the tape having a name that occupies the second entry group in the File Name Entry Section, the handler must (a) scan the Accessibility Map for the second appearance of a 1-bit, then (b) determine that bit's location relative to the start of the map. That location speci-

fies the position of the referenced file relative to the beginning of the tape. The interaction of the File Name Entry Section and the Accessibility Map is shown in Figure 5-5.

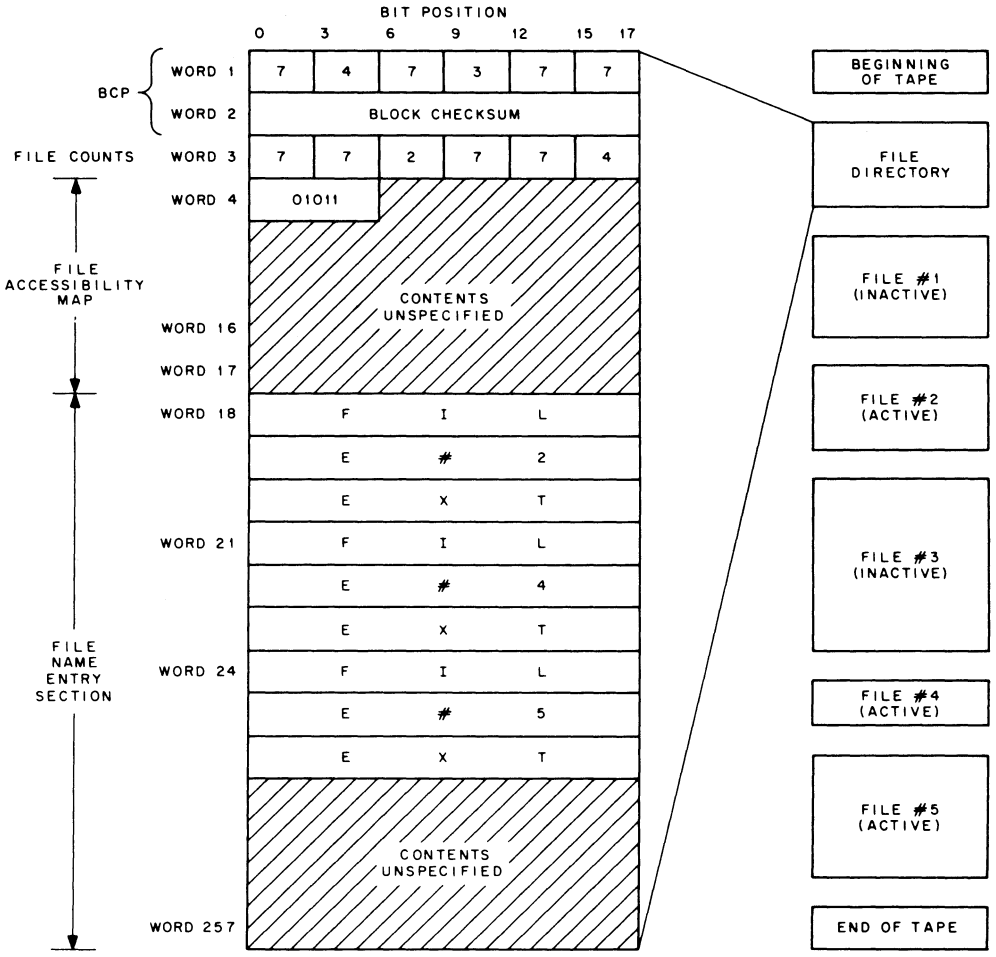


Figure 5-5a. Format of the File Directory Data Block, showing relationship of active and inactive files to file name entries and to Accessibility Map.

Figure 5-5b. Format of file-structured tape, showing directory block and data files.

5.9.2.2 User-File Labels - Associated with each file on tape are two identifying labels. The first is a header label and precedes the first data block of the file; the second, a trailer label, follows the final recorded data block of the file. Each label is 27_{10} words in length. Label format is shown in Figure 5-6.

Note that the trailer label differs from the header label only in the contents of the BI field and in that the former includes an indication (Word 4) of the total blocks recorded in the file. The total includes the two labels themselves.

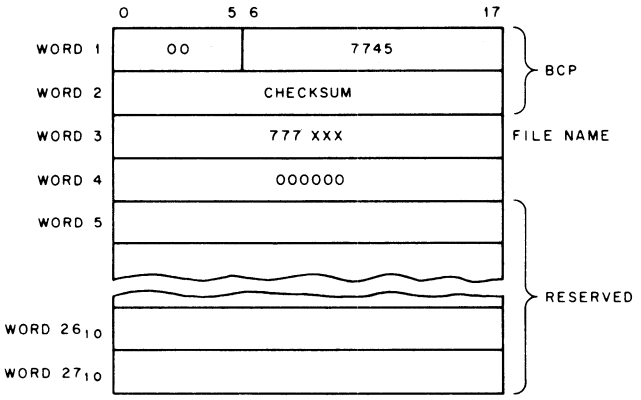


Figure 5-6a. User-File Header Label Format

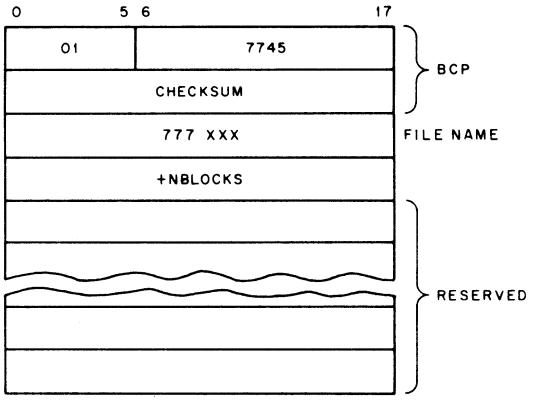


Figure 5-6b. User-File Trailer Label Format

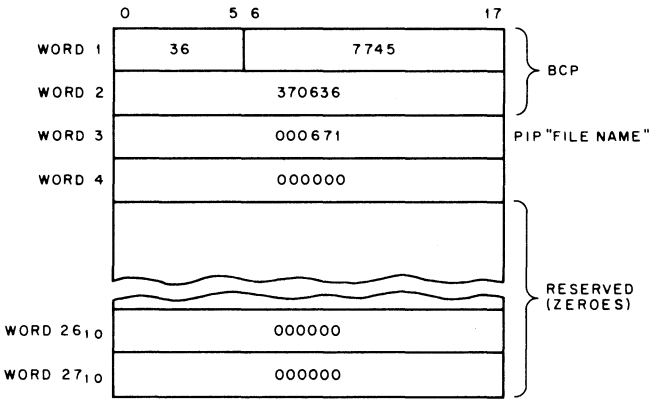
5.9.2.3 File Names in Labels - The handler will supply the contents of the file-name fields (Word 3) in labels. These are used only for control purposes during the execution of .SEEK's. The name consists simply of the two's complement of the position of the recorded file's bit in the Accessibility Map: the "name" of the first file on tape is 777777; that of the third file is 777775; and so on. A unique name is thus provided for each file physically present on the tape. Since there may be a maximum of 252_{10} files present, legal file-name values lie in the range 777777 to 777404.

5.10 MAGNETIC TAPE SYSTEM OPERATION

5.10.1 System File Structure

In general, the system tape differs only incidentally from the user tapes described previously. Directories, labels, and data blocks have like formats and perform identical functions on both user and system tapes. Some identifying information however, is different in order to accommodate those cases in which the tape is bootstrap-accessed (for example, in loading the System Loader).

5.10.1.1 System File Labels - Labels on the system tape serve the same purpose as user-file labels; that is, the identification of the file which follows. They differ only in the contents of the Block Identifier field and of the file name indicator. For header labels, BI = 36_8; for trailer labels, BI = 37_8. The file-name words in both labels contain the block number which begins the file in block-addressable bulk-storage system. In all other respects, the labels are identical to those which appear on user tapes. Typical system tape label format (for PIP) is shown below.



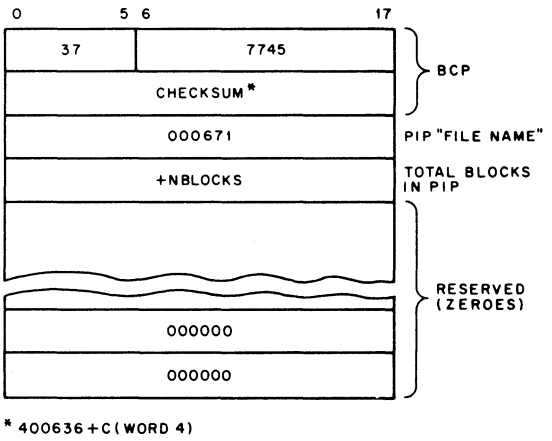


Figure 5-7a. System Program (PIP) Header Label. Figure 5-7b. System Program (PIP) Trailer Label.

All absolute system programs are labeled in the manner described.

5.10.1.2 System File Data Blocks - The Block Control Pair for an absolute system program block serves a purpose somewhat different from that for a user-file data block. The format of the Block Control Pair is as follows:

a. Block Identifier (Word 1, bits 0 through 5): 77_8 = System program data block.

b. Word Count (Word 1, bits 6 through 17): Two's complement count of data words in the block, including the Block Control Pair.

c. Word 2: 15-bit load address-1 for this block.

5.10.2 System Tape Organization

The system tape is organized in the usual fashion, that is, the File Directory data block is recorded first and is followed by labeled data files. The single requirement is that two system files - the Monitor and the System Loader - appear immediately following the Directory. Other system files are recorded following the System Loader.

5.10.3 System Startup

The system tape must be mounted on unit 0 and the transport must be on line and ready. The bootstrap program is loaded via the paper tape reader; the bootstrap will bring the Monitor into core from the system tape and give control to it. The user may then proceed as described in Section 5.4.

5.10.4 Continuous Operation*

Under certain circumstances, it is possible to perform successive I/O transfers without incurring the shut-down delay that normally takes place between blocks. The handler stacks transfer requests, and thus ensures continued tape motion, under the following conditions:

a. The I/O request must be received by the CAL handler before a previously-initiated I/O transfer has been completed.

b. The unit number must be identical to that of the previously-initiated I/O transfer.

c. The previously-requested transfer must be completed without error. In general, successive error-free READS (WRITES) to the same transport will achieve non-stop operation. The examples given below illustrate the principle.

Example 1: Successive Continued Operation.

```
SLOT = 1
INPUT = 0
BLOKNO = 0
READ1           .TRAN SLOT, INPUT, BLOKNO, BUFF1, 257
READ2           .TRAN SLOT, INPUT, BLOKNO, BUFF2, 257
RETURN          JMP READ1
```

The program segment in Example 1 will most probably keep the referenced transport (.DAT slot 1) up to speed. The probability decreases as more time elapses between READ1 and READ2, and between READ2 and RETURN.

Example 2: Unsuccessful Continued Operation.

```
SLOT = 1
INPUT = 0
BLOKNO = 0
READ            .TRAN SLOT, INPUT, BLOKNO, BUFF, 257
STOP            .WAIT SLOT
RETURN          JMP READ
```

The program segment in Example 2 will not keep the tape moving because the explicit .WAIT at location STOP prevents control from returning to location READ until the transfer first initiated at READ has been completed.

Example 3: Unsuccessful Continued Operation

```
SLOT1 = 1
SLOT2 = 2
INPUT = 0
BLOKNO = 0
```

*Control and transport requirements for successful continuous operation are outlined in the TC-59 Instruction Manual (DEC-9A-I3BA-D) and in the PDP-9 User Manual (F-95).

```
READ1       .TRAN SLOT1, INPUT, BLOKNO, BUFF1, 257
READ2       .TRAN SLOT2, INPUT, BLOKNO, BUFF2, 257
RETURN      JMP READ1
```

This program segment will not provide non-stop operation because of the differing unit specification at READ1 and READ2.

5.11 DRUM FILE ORGANIZATION

Drum handlers DRA, DRB, DRC, and DRD are now available in version V4B of the Keyboard Monitor System. The Drum handlers are modified versions of DECtape handlers DTA, DTB, DTC, and DTD, (see Section 7.4) and permit data transfer to/from the RM09 Drum by both user and system programs. It should be clearly noted, however, that availability of Drum handlers does not imply use of the Drum as a system device; that capability does not presently exist.

The PDP-9 Advanced Software Drum handlers may be used with any one of the five RM09 drum options available. Storage capacity is defined in the following table in terms of DECtape storage capacity.

Drum	Size	Number of Units	Number of Blocks (256_{10})
1	32K	1/4	128
2	65K	1/2	256
3	131K	1	512
4	256K	2	1024
5	524K	4	2048

A Drum unit is defined as 512_{10} sectors or blocks of 256_{10} words each (i.e., 64 blocks less than the capacity of a single PDP-9 DECtape). The 32,65 or 131K Drums are referenced as unit 0; the 262K drum, as units 0 and 1; the 524K drum, as units 0, 1, 2, and 3. Referencing a unit number greater than that permitted for a given drum results in an IOPS 26 error. Listed below are the sector limits for each unit of each Drum.

Unit Sector Limits

Drum \ Unit	0	1	2	3
32K	0 - 177	---	---	---
65K	0 - 377	---	---	---
131K	0 - 777	---	---	---
262K	0 - 777	1000 - 1777	---	---
527K	0 - 777	1000 - 1777	2000 - 2777	3000 - 3777

The drum size of a system must be set up in .SCOM+4, bits 15-17. A value of 1-5 is used where 1 = 32K; 2 = 65K, etc. IOPS 35 results if .SCOM+4 is not properly set for drum size.

Although the Drum control is designed to perform full block (256_{10}) transfers rather than variable word count transfers, code has been included in both DRA and DRD to allow variable word count transfers via the .TRAN command (i.e., .TRAN works for Drum as it does for the PDP-9 Disk and DECtape). Furthermore, division of the Drum into logical units does not prevent .TRAN's or .MTAPE READS/WRITES from referencing any and all Drum sectors provided drum unit 0 is used. More simply, if drum units 1-3 are referenced, a check is made such that no transfer is allowed to exceed the sector limits of a unit. If sector size exceeds 777_8, an IOPS 25 error message results. However, if Drum unit 0 is assigned, any sector may be referenced in .TRAN or .MTAPE READ/WRITE commands, provided the physical drum size is not exeeded.

Drum file data organization is basically the same as that for DECtape (see Section 5.6). As mentioned earlier, the file capacity is limited by a unit size of 512_{10} rather than 576_{10}. Sector 100 (1100 for unit 1, etc.), contains the Drum unit directory; sectors 71-77 (1071 - 1077 for unit 1, etc.), contain file bit map information. Unit size is reflected in a directory listing via PIP. For example, when the N (NEWDIR) command is issued, the listing appears as follows:

32K Drum:

```
DIRECTORY LISTING
0 USER FILES
710 SYSTEM BLKS
170 FREE BLKS
```

65K Drum:

```
DIRECTORY LISTING
0 USER FILES
510 SYSTEM BLKS
370 FREE BLKS
```

131, 262 or 524K Drum: (each unit)

```
DIRECTORY LISTING
0 USER FILES
110 SYSTEM BLKS
770 FREE BLKS
```

Because the Drum is not used as a system device, it should be noted that ↑ Q to the Drum cannot be executed. Consequently, the S switch in PIP is inoperative to the Drum and results in execution of the N (NEWDIR) command instead.

Users of Monitor V4B (9-30-68) system tapes will find the Drum handlers included in their I/O library and the skip chain set up accordingly. To adopt the system for the installation Drum size, however, .SCOM+4 (absolute location 104), bits 15-17, must be set (1-5) by use of the system program, PATCH.

Drum users with Monitor tapes earlier than V4B must perform the following sequence:

1. Use SGEN to generate a new system tape being sure to:

   a. Answer "yes" when asked about the presence of new handlers (DRA, DRB, DRC and DRD.)

   b. Answer DRSF, 706101 and
      DRNEF, 706201
   to the new skip IOT question.

   c. Position DRSF high in the skip chain (after DTDF - DECtape or DSSF - Disk).

   d. Position - DRNEF as the very last skip in the chain. Note: DRNEF is a reverse skip (SKIP ON NOT ERROR) and must be preceded by a minus (-) sign.

2. Use UPDATE to incorporate the new Drum handlers into the library of the newly generated tape.

3. Use PATCH to set .SCOM+4, bits 15-17 (absolute cell 104) in KM9 to the appropriate drum size (1-5) for the installation.

CHAPTER 6
BACKGROUND/FOREGROUND MONITOR

6.1 BACKGROUND/FOREGROUND MONITOR FUNCTIONS

The Background/Foreground Monitor is designed to control processing and I/O operations in a real-time or time-shared environment. It is essentially an extension of the Keyboard Monitor (described in Chapter 5) and allows for time-shared use of a PDP-9 by a protected, priority, user FOREGROUND program and an unprotected system or user BACKGROUND program.

The Background/Foreground Monitor greatly expands the capabilities of PDP-9 ADVANCED Software and makes optimum use of all available hardware. It allows for recovery of the free time (or dead time) that occurs between input/output operations, and promotes 100% utilization of central processor time.

The reader is referred to Chapter 2 for a general discussion of the Monitor environment, and to Chapter 3 for a detailed description of user program commands (system macros) available in the Background/Foreground Monitor environment. It should be noted that all material presented in this manual for the Background/Foreground Monitor is preliminary and subject to change.

FOREGROUND programs are defined as the higher-priority, debugged user programs that interface with the real-time environment. They normally operate under Program Interrupt (PI) or Automatic Priority Interrupt (API) control, and are memory protected. At load time they have top priority in selection of core memory and I/O devices, and at execution time they have priority (according to the assigned priority levels) over processing time. Depending upon system requirements, the user's FOREGROUND program could be an Executive capable of handling many real-time programs or subprograms at four levels of priority (with API present).

BACKGROUND processing is essentially the same as the processing normally accomplished under control of the Keyboard Monitor. That is, it could be an assembly, compilation, debugging run, production run, editing task, etc. BACKGROUND programs may use any facilities (for example, core, I/O, and processing time) that are available and not simultaneously required by the FOREGROUND job. Using the Monitor's Batch processing capability optimizes the processing of BACKGROUND jobs under control of the Background/Foreground Monitor.

The Background/Foreground Monitor system is externally a keyboard-oriented system; that is, all FOREGROUND and BACKGROUND requests for systems information, core, I/O devices, programs to be run, etc., are made via the Teletype keyboards (see Figure 6-1). At run time, the Monitor internally controls scheduling and processing of I/O requests, while protecting the two resident users.

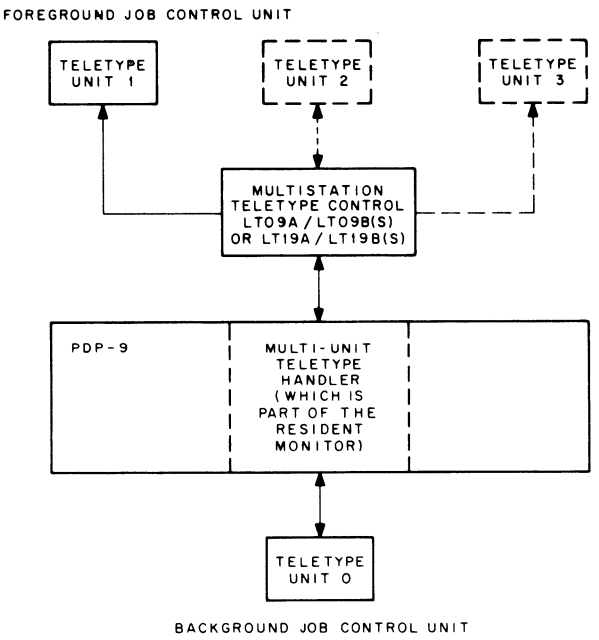


Figure 6-1 Keyboard Communication in Background/Foreground Monitor System

The Background/Foreground Monitor performs the following functions as it controls the time-shared use of the PDP-9 central processor by two co-resident programs:

a. Schedules processing time.

b. Protects the FOREGROUND job's core and I/O devices.

c. Provides for the sharing of multi-unit device handlers, such as DECtape, by both FOREGROUND and BACKGROUND jobs.

d. Allows convenient use of API software levels by FOREGROUND jobs.

e. Provides for convenient and shared use of the system Real Time Clock.

f. Allows communication between the BACKGROUND and FOREGROUND jobs via core-to-core transfers.

6.1.1 Scheduling of Processing Time

At run time, the FOREGROUND job retains control except when it is I/O bound; that is, when completion of an I/O request is required before it can proceed any further, as in the following example.

```
.READ 3,0,LNBUF,48          /READ TO .DAT SLOT 3
    :
    :
.WAIT 3                     /WAIT ON .DAT SLOT 3
```

If the .WAIT is reached before the input requested by the .READ has been completed, control is transferred to a lower priority FOREGROUND segment or to the BACKGROUND job until the input for the FOREGROUND job is completed.

Since multi-unit device handlers can be shared by FOREGROUND and BACKGROUND programs, there is a mechanism by which a FOREGROUND I/O request will cause a BACKGROUND I/O operation to be stopped immediately so that the FOREGROUND operation can be honored. On completion of the FOREGROUND I/O, the BACKGROUND I/O will be restarted with no adverse effects on the BACKGROUND job.

The FOREGROUND program can also indicate that it is I/O bound by means of the .IDLE command (Section 3.3.3). This is useful when the FOREGROUND job is waiting for real-time input from any one of a number of input devices, as in the following example (see Section 3.3.1 for description of real-time read .REALR command).

```
.REALR 1,0,LNBUF1, 32, CTRL1, N          /REAL
.REALR 2,2,LNBUF2, 42, CTRL2, N          /TIME
.REALR 3,3,LNBUF3, 36, CTRL3, N          /READS
    :
    :
.IDLE
```

If the .IDLE is reached before any of the input requests have been satisfied, control is transferred to a lower priority FOREGROUND segment or to the BACKGROUND job, which retains control until one of the FOREGROUND input requests is satisfied. Control is then returned to the FOREGROUND job by executing the subroutine at the specified completion address (CTRL1, CTRL2, CTRL3) and at the priority level specified by N, which may be

0 = Mainstream (lowest level)
4 = Current level
5 = Software level 5
6 = Software level 6
7 = Software level 7

NOTE

If real-time reads (.REALR), real-time writes (REALW), or interval timer requests (.TIMER) are employed in the BACKGROUND, N may be set to 0, 4, 5, 6, or 7, but will be converted to 0 since the BACKGROUND job can run only on the mainstream level. This allows you to set up N as you want it to be in cases where a BACKGROUND program is to be subsequently run in the FOREGROUND.

6.1.2 Protection of FOREGROUND Job Core and I/O

The FOREGROUND job's core is protected by means of the Memory Protection Option (Type KX09A). The BACKGROUND job runs with memory protect enabled; the FOREGROUND job runs with memory protect disabled to allow for execution of IOT's.

Protection of the FOREGROUND job's I/O devices is accomplished with the hardware by means of the memory protect option, which prohibits IOT and Halt instructions in the BACKGROUND area; and with the software by means of the Monitor and IOPS, which screen all I/O requests made by Monitor CAL's. Also, the Linking Loader prevents the BACKGROUND job I/O from conflicting with that of the FOREGROUND job (for example, it would not honor a BACKGROUND request for a paper tape handler being used by the FOREGROUND job).

6.1.3 Sharing of Multi-Unit Device Handlers

The Background/Foreground Monitor allows sharing of multi-unit, mass-storage device handlers (such as DECtape, Magnetic Tape, and Disk) between BACKGROUND and FOREGROUND jobs to reduce core memory requirements for the system. Using these multi-unit handlers, n files can be open simultaneously, where n equals the number of .DAT slots associated with the particular bulk storage device. When this count is not true (because of the .DAT slots not being used simultaneously), the keyboard command FILES (Section 6.3.1) can be used to remedy the situation. Both the FOREGROUND and BACKGROUND jobs can indicate their file requirements by means of the FILES keyboard command.

The multi-unit handlers are capable of stacking one BACKGROUND I/O request. Thus, control is returned to the BACKGROUND job to allow non-I/O related processing when the handler is preoccupied with an I/O request from the FOREGROUND job. For example, if the FOREGROUND job has requested DECtape I/O with a .READ, and is waiting for its completion on a .WAIT, control is returned to the BACKGROUND job. If the BACKGROUND job requests DECtape I/O with a .READ, the

handler will stack the request and return control to the BACKGROUND job following the .READ. The BACKGROUND job can then continue with non-I/O related processing as though the .READ were being honored.

6.1.4 Use of Software Priority Levels

The Background/Foreground Monitor allows convenient use of software priority levels of the API by the FOREGROUND job. The BACKGROUND job can use only the mainstream level.

6.1.5 Use of Real-Time Clock

The Background/Foreground Monitor provides for convenient and shared use of the system real-time clock. It will effectively handle an unlimited number of intervals at the same time, thus the real-time clock can be used simultaneously by both BACKGROUND and FOREGROUND jobs.

6.1.6 Communication Between BACKGROUND and FOREGROUND Jobs

The Background/Foreground Monitor allows communication between BACKGROUND and FOREGROUND jobs via core-to-core transfers. This is accomplished by means of a special "I/O device" handler within IOPS. Complementing I/O requests are required for a core-to-core transfer to be effected; for example, a FOREGROUND .READ (.REALR) from core must be matched with a BACKGROUND .WRITE (.REALW) to core.

Two possible uses of this feature are:

a. The BACKGROUND job could be related to the FOREGROUND job, and as a result of its processing, pass on information that would affect FOREGROUND processing.

b. The BACKGROUND job could be a future FOREGROUND job, and the current FOREGROUND job, being its predecessor, could pass on real-time data to create a true test environment.

6.2 HARDWARE REQUIREMENTS AND OPTIONS

The following hardware is required to operate the Background/Foreground Monitor System.

a. Basic PDP-9 with Teletype,

b. Memory Extension Control, Type KG09A,

c. Additional 8192-Word Core Memory Module, Type MM09A,

d. Memory Protection Option, Type KX09A,

e. External Teletype System, including at least*:

(1) One Teletype Control, Type LT09A,

(2) One Teletype Line Unit, Type LT09B,

(3) One Teletype, Model KSR33 or equivalent,

f. Bulk Storage System, comprising either:

(1) One DECtape Control, Type TC02, and two DECtape Transports, Type TU55, or

(2) One Disk System, Type RB09, or

(3) One Disk System, Type RF09/RS09

The following options may be added to improve system performance (as noted):

Options	Effect
Additional 8192 Word Core Memory Modules, Type MM09B and MM09C	Increase the maximum size of both BACKGROUND and FOREGROUND programs that can be handled by the system.
Automatic Priority Interrupt, Type KF09A	Allows for quicker recognition of requests for service by FOREGROUND devices.
Extended Arithmetic Element, Type KE09A	Increases speed of arithmetic calculations
Additional DECtape Transports, Type TU55, or IBM-compatible Magnetic Tape Transports, Type TU20 or TU20A	Allows greater bulk storage capability, simultaneous use of storage media by more programs. Since only one file may be open at a time on IBM-compatible magnetic tape transports, more then two Type TU20 or TU20A transports may be desirable for some applications.
Automatic Line Printer, Type 647	Provides greater listing capabilities.
200 CPM Card Reader, Type CR03B	Allows card input and control cards for BACKGROUND Batch processing.
Additional Teletype Line Units, Type LT09B, (or LT19B) and Teletypes, Type KSR33 or equivalent	Provides additional output devices if multiple FOREGROUND jobs may require simultaneous output or BACKGROUND jobs wish to use multiple devices.

*The basic system Teletype is assigned to the BACKGROUND environment. One Teletype of the external Teletype system must be reserved for the FOREGROUND job(s); additional Teletypes may be assigned to either BACKGROUND or FOREGROUND functions. If the API option is available, a Type LT19A Teletype Control and a Type LT19B Line Unit are required.

6.3 KEYBOARD COMMANDS

In addition to the keyboard commands available in the Keyboard Monitor environment (see Section 5.3), the Background/Foreground Monitor recognizes and accepts the following commands: FILES, FCORE, FCONTROL, and BCONTROL. Each of these commands is described in the following paragraphs.

6.3.1 FILES

This command is used to conserve core space by indicating the number of bulk storage files that will be open simultaneously. It is normally typed prior to requesting the loading of user programs so that the system or linking loader can allocate sufficient buffer space. The command is typed after the Monitor's dollar sign request.

Example:

```
$FILES DT3
```

In this case, the DECtape handler will be able to accommodate three files open simultaneously.

6.3.2 FCORE

This command is used to define additional core required by the FOREGROUND job, exclusive of the Monitor, the system device handler, the teletype handler and the FOREGROUND user programs. The command is normally typed following the Monitor's dollar sign request and is only applicable before loading the FOREGROUND job. It is followed by a space and some octal number n, which represents the total number of 1K octal increments of core memory required by the FOREGROUND job (area between .SCOM+2 and .SCOM+3). If FCORE is not used, core will be allocated dynamically at load time and there will be no free core available to the FOREGROUND job.

Example:

```
$FCORE  3
```

6.3.3 FCONTROL

This command allows the user to change the Teletype assigned to the FOREGROUND job while the Monitor is operating in the FOREGROUND mode. Initially, keyboard commands are accepted from external Teletype Unit 1. The user could assign external Teletype Unit 2 (if available) to the FOREGROUND job by typing "FCONTROL 2" after the Monitor's dollar sign request. The Monitor would respond by typing CONTROL RELINQUISHED on external Teletype Unit 1, typing

```
MONITOR
$
```

on external Teletype Unit 2, and accepting subsequent keyboard commands in the FOREGROUND mode from external Teletype Unit 2. Control would then remain with Teletype Unit 2 until another FCONTROL is given.

6.3.4 BCONTROL

This command allows the user to change the Teletype assigned to the BACKGROUND job while the monitor is operating in the BACKGROUND mode. When control is first transferred to the BACKGROUND mode, keyboard commands are accepted from Teletype Unit 0 (basic system teletype). The user can assign an external Teletype to the BACKGROUND job, provided that an external Teletype not currently assigned to the FOREGROUND job is available. For example, if external Teletype Unit 2 is available and not assigned to the FOREGROUND program, the user could assign it to the BACKGROUND job by typing "BCONTROL 2" after the Monitor's dollar sign request. The Monitor would respond by typing CONTROL RELINQUISHED on Teletype Unit 0, typing

```
MONITOR
$
```

on Teletype Unit 2, and accepting subsequent keyboard commands in the BACKGROUND mode from external Teletype Unit 2. Control would then remain with Teletype Unit 2 until another BCONTROL is given.

6.4 OPERATING THE BACKGROUND/FOREGROUND MONITOR SYSTEM

The reader is referred to the Keyboard Monitor Guide (DEC-9A-MKFA-D) for detailed operating procedures for system programs, which can be used in BACKGROUND processing. The following PDP-9 ADVANCED Software System manuals contain additional detailed information on system programs.

Manual	Document No.
Utility Programs	DEC-9A-GUAB-D
MACRO-9 Assembler	DEC-9A-AMZA-D
FORTRAN IV	DEC-9A-KFZA-D

This section contains descriptions of loading the Background/Foreground Monitor, assigning devices, loading user FOREGROUND programs, loading system or user BACKGROUND programs, and error detection and handling. Figure 6-2 contains complete memory maps and a summary of all loading procedures.

6.4.1 Loading the Background/Foreground Monitor

The procedure for loading the Background/Foreground Monitor is similar to that for loading the Keyboard Monitor. That is, the System Bootstrap (in hardware READIN format) is loaded at the top of core using the paper tape reader. This is accomplished by placing the tape in the reader, momentarily pressing the tape feed button, setting the Address switches to the lowest address of the bootstrap (37637 for 16K systems), and depressing the I/O RESET and READIN switches. The System Bootstrap automatically loads the resident portion of the Monitor from the system device into lower core. It then transfers control to the Monitor via the .EXIT command, with the FOREGROUND flag set to simulate a FOREGROUND .EXIT. (This will cause subsequent keyboard commands to be accepted from Teletype Unit 1 and interpreted as FOREGROUND requests.) The resident Monitor loads the non-resident Monitor from the system device into upper memory, overlaying the System Bootstrap. It then types

```
MONITOR
$
```

on external Teletype Unit 1 and transfers control to the Keyboard Listener portion of the non-resident Monitor to await keyboard commands from Teletype Unit 1. Initially, since the FOREGROUND flag was set, these commands are interpreted as FOREGROUND commands. While operating in the FOREGROUND mode, the FOREGROUND control Teletype can be changed from external Teletype Unit 1 to any other external Teletype available by means of the FCONTROL keyboard command (see Section 6.3.3).

6.4.2 Assigning Devices

Before loading a user FOREGROUND program or a system or user BACKGROUND program, the user should make all device assignments required for the program to be run. There are two Device Assignment Tables (.DAT) in the Background/Foreground Monitor system: one used by BACKGROUND jobs, and one used by FOREGROUND jobs. The Monitor determines which table is intended or required according to its current operating mode. The .DAT slot assignment for BACKGROUND jobs are the same as normal Keyboard Monitor .DAT slot assignments (see Section 5.4.3). Since only user programs operate in the FOREGROUND mode, all .DAT slots can be used by the FOREGROUND job, except .DAT slots -1, -4, -5 and -7.

6.4.3 Loading User FOREGROUND Programs

When a keyboard command that requests loading of a user FOREGROUND program is encountered, the non-resident Monitor brings in the System Loader, overlaying itself. This is done via

the resident system device handler. For FOREGROUND loads, the System Loader is the Linking Loader. It first loads any other I/O handlers required for loading, and then loads the user program along with its required I/O handlers. It then allocates buffer space to accommodate the number of bulk-storage files either

a. specified by the keyboard command FILES that was typed prior to the load request, or

b. determined from the number of .DAT slots associated with bulk storage devices.

The System Loader sets the memory protect bound above the FOREGROUND system. It then places the address of a routine to perform a .EXIT into the BACKGROUND program counter register (control will go there when the FOREGROUND job becomes I/O bound), and gives control to the FOREGROUND job with memory protect disabled.

When the FOREGROUND job becomes I/O bound, control is transferred to the routine which performs a .EXIT. The Monitor recognizes this as a BACKGROUND .EXIT and loads the non-resident Monitor (via the resident system device handler) into upper memory. It then gives control to the Keyboard Listener which types

```
MONITOR
$
```

on the system Teletype Unit 0 and then waits for BACKGROUND keyboard commands from the system Teletype (Unit 0). While operating in the BACKGROUND mode, the control Teletype can be changed from Unit 0 to any other Teletype available, not currently being used by the FOREGROUND job, by means of the BCONTROL command (see Section 6.3.4).

6.4.4 Loading System or User BACKGROUND Programs

When a keyboard command that requests loading of a system or user BACKGROUND program is encountered, the non-resident Monitor brings in the System Loader, overlaying itself. This is done via the resident system device handler.

If the BACKGROUND program is a system program, the System Loader loads the system program I/O handlers up from the top of the FOREGROUND system*, allocates buffer space to accommodate all .DAT slots associated with bulk storage devices, and then loads the system program into the top of memory. It then sets the memory protect bound above the allocated buffer space and transfers control to the system program.

---

*The System Loader loads I/O handlers required by either a system program or the Linking Loader just above the FOREGROUND job. This is done to make use of any space resulting from the fact that the smallest unit of memory protection is 1K (decimal). Also, multi-unit handlers (such as DECtape) which are already in core for the FOREGROUND job are shared, rather than loaded again.

If the BACKGROUND program is a user program, the System Loader loads the Linking Loader I/O handlers up from the top of the FOREGROUND system, allocates the required buffer space, and then loads the relocatable Linking Loader where the memory protect bound can be set just below it.

The Linking Loader is used to load the user's BACKGROUND program down from the top of core (see Keyboard Monitor Guide, DEC-9A-MKFA-D). The user's I/O handlers are loaded last (relocated to run from the area just above the Loaders I/O handlers). User program handlers that are identical to Loader or FOREGROUND handlers already in core are not loaded. The necessary buffer space is allocated just above the area where the user's handlers are to be relocated. An .EXIT from the Linking Loader causes the user program I/O handlers to be block transferred to their running position, the memory protect bound to be set just above the buffer area, and control to be transferred to the user program.

6.4.5 End of Job

When an .EXIT is encountered in a job, .EXIT is output on the associated control Teletype and additional action is taken depending upon whether the job is a BACKGROUND or a FOREGROUND.

If the job is a BACKGROUND, and the FOREGROUND job has not been completed, the non-resident Monitor is loaded into upper core and outputs

```
MONITOR
$
```

to the current BACKGROUND control Teletype to indicate its readiness to receive BACKGROUND commands. If the job is a BACKGROUND, and the FOREGROUND job has been completed, the

```
MONITOR
$
```

is output on the current FOREGROUND control Teletype to indicate readiness to accept FOREGROUND commands.

If the job is a FOREGROUND, the FOREGROUND completion flag is set and control is returned to the BACKGROUND job. The BACKGROUND job may be aborted at this time by typing CTRL C on the FOREGROUND control Teletype. This will cause ABORT to be typed on the current BACKGROUND control Teletype, the non-resident monitor to be loaded into upper core, and

```
MONITOR
$
```

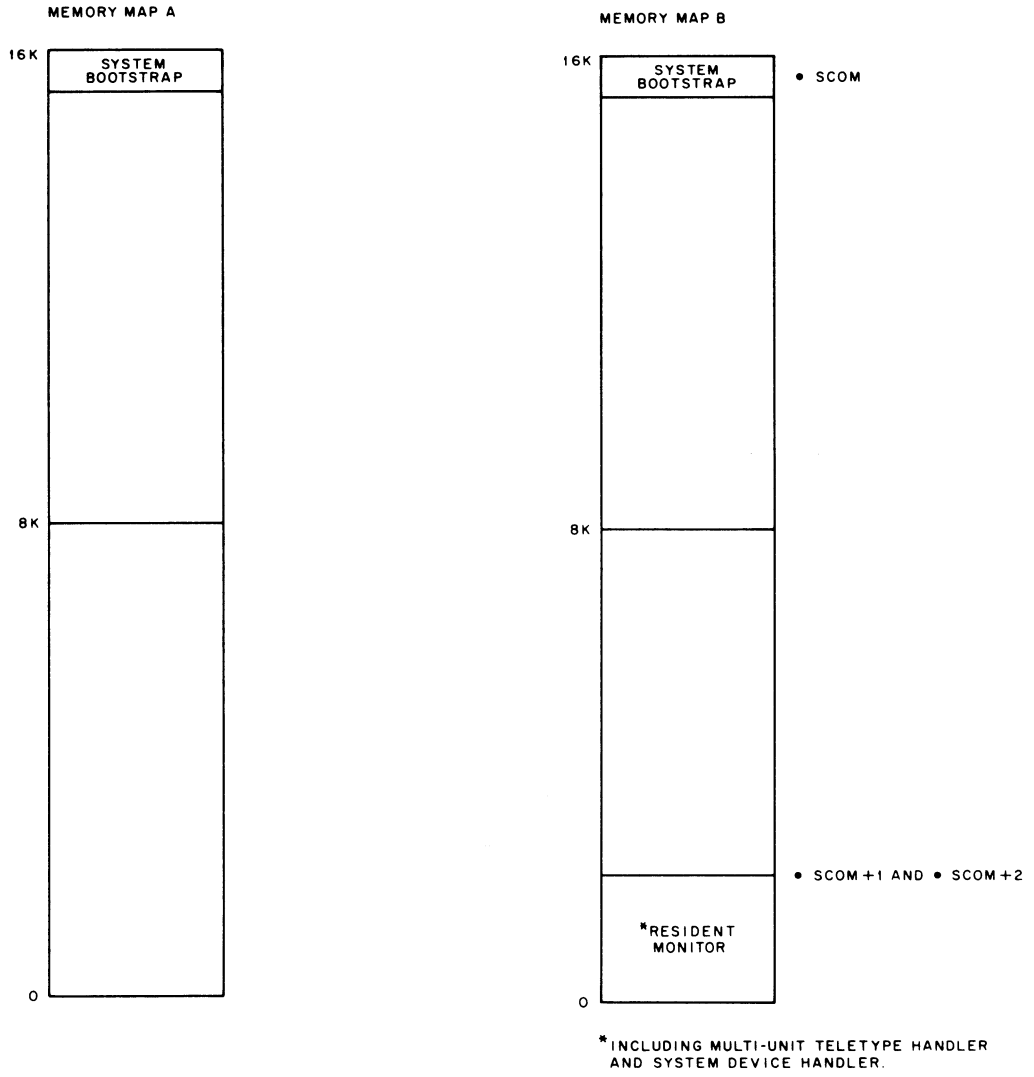
to be typed on the FOREGROUND control Teletype indicating readiness to accept FOREGROUND command.

6.4.6 Error Detection and Handling

Comprehensive error checking is provided by the Background/Foreground Monitor, the loaders, and the Input/Output Programming System. Detailed lists of errors that may occur are given in Appendices C, D, and E, respectively.

After error messages are output, the non-resident Monitor is brought into core to allow initialization of the next job.

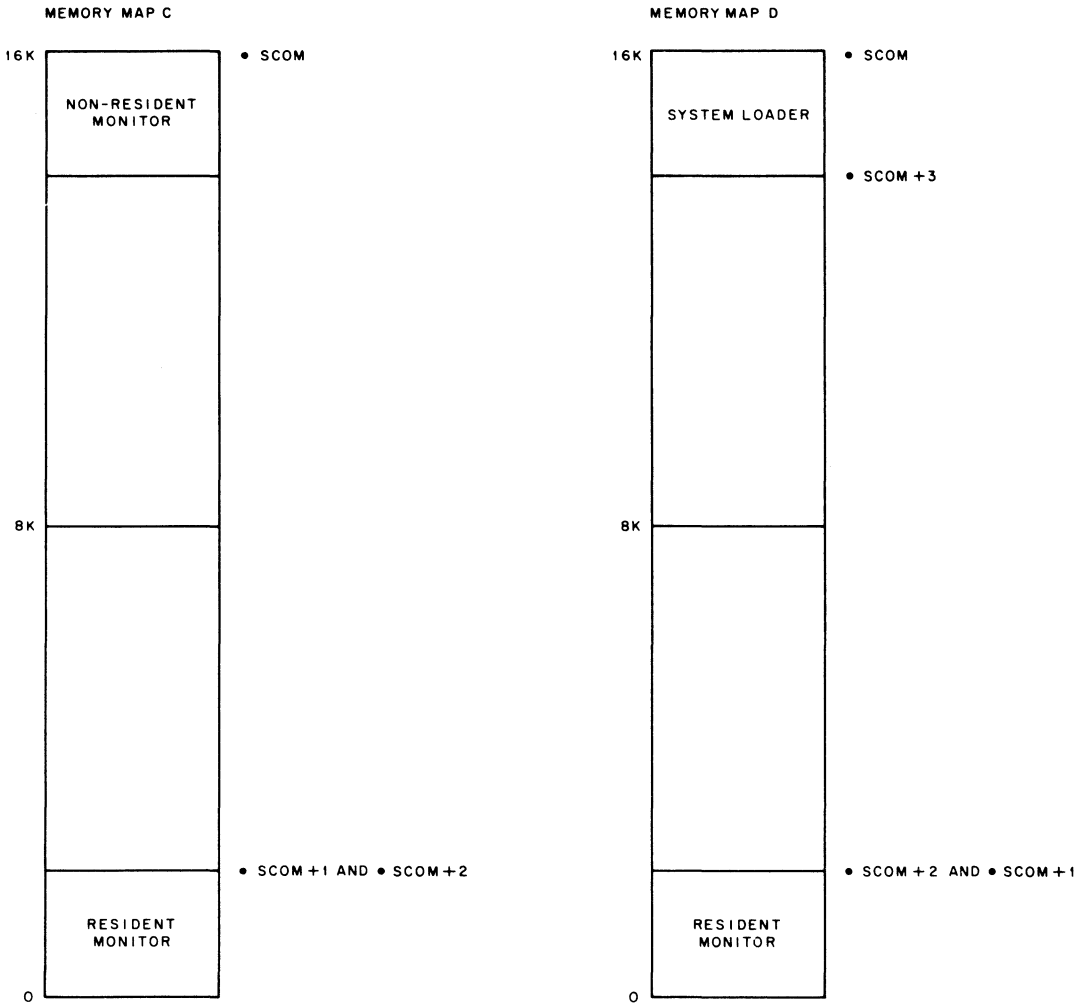
The system programs that operate in the BACKGROUND mode also provide comprehensive error checking. Refer to the appropriate PDP-9 ADVANCED Software System Manual (listed at the beginning of this section) for detailed information on these errors.



The System Bootstrap is loaded at the top of core via the paper tape reader in HRM format.

The System Bootstrap automatically loads the resident Monitor from the system device into lower core.

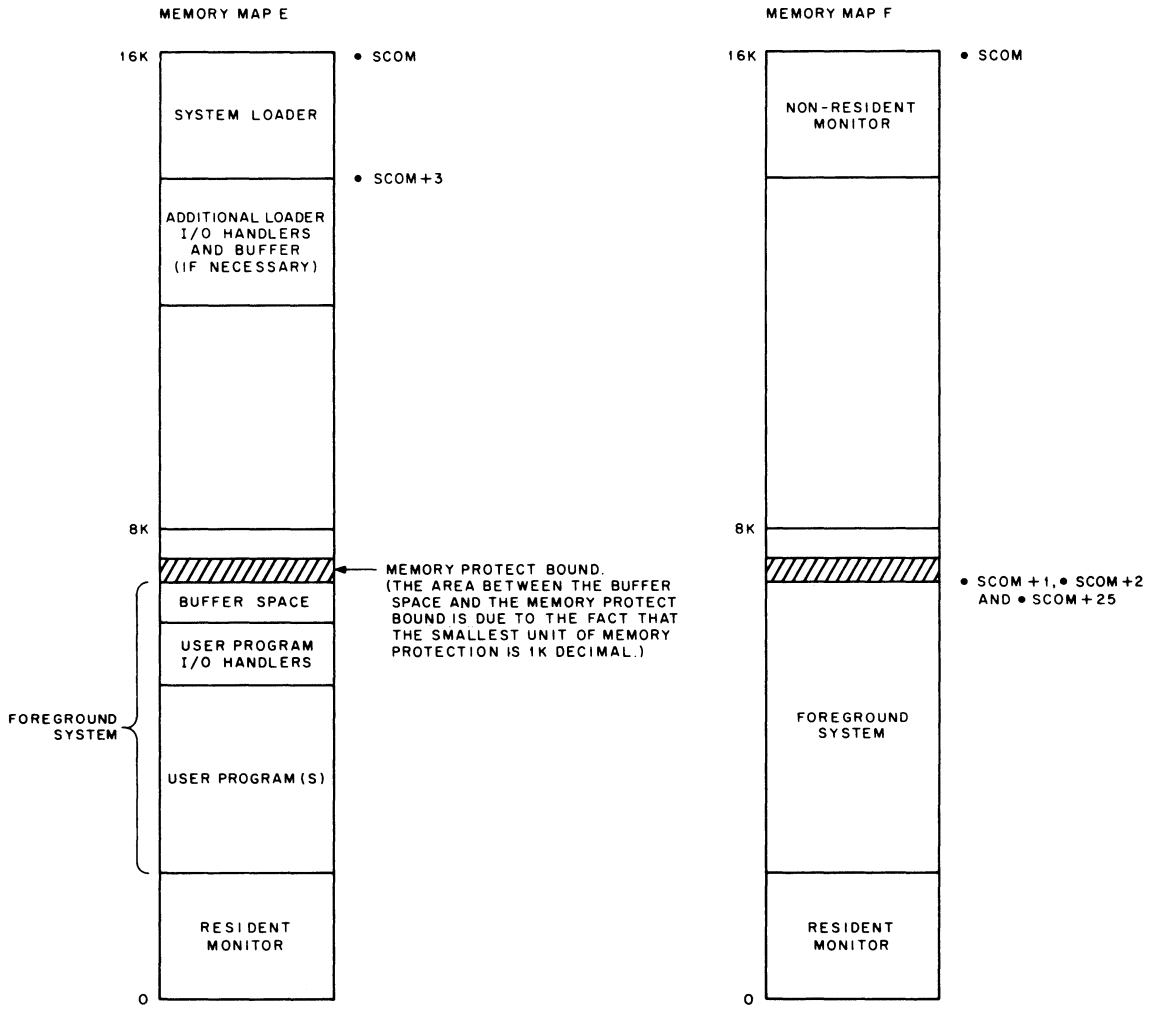
Figure 6-2 Background/Foreground Monitor System Memory Maps



The resident Monitor loads the non-resident Monitor (via the resident system device handler) into upper core, overlaying the System Bootstrap

To load a user FOREGROUND program, the non-resident Monitor brings in the System Loader, overlaying itself. For FOREGROUND loading, the System Loader is the Linking Loader.

Figure 6-2 Background/Foreground Monitor System Memory Maps (Cont)

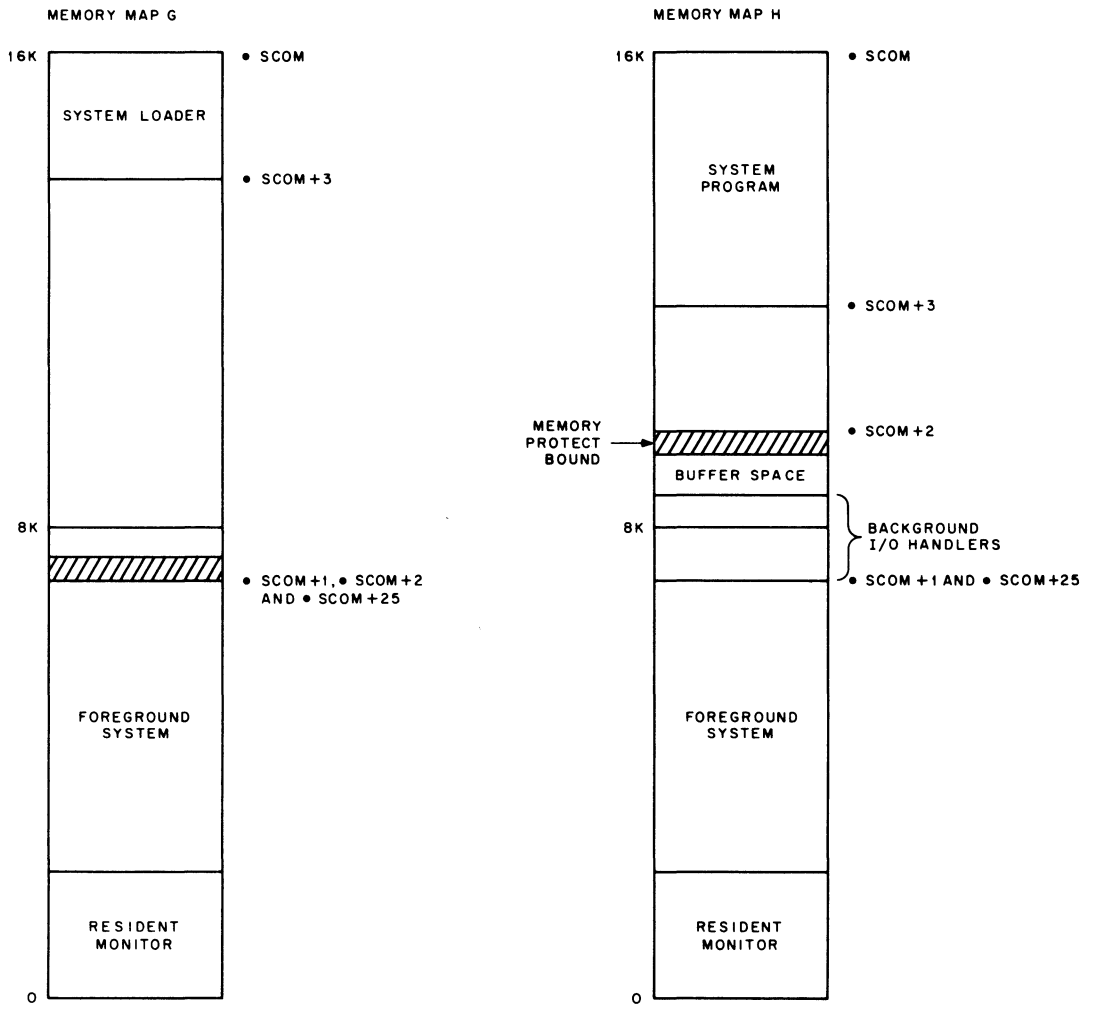


The System Loader first loads any additional I/O handlers required for loading. It then loads the user program and the I/O handlers that it requires, and allocates buffer space.

Memory protect bound. (The area between the buffer space and the memory protect bound is due to the fact that the smallest unit of memory protection is 1K decimal. However, this area can be used for dynamic data storage via a software protect feature.)

When the FOREGROUND job becomes I/O bound, control is transferred to the BACKGROUND job. The resident Monitor loads the non-resident Monitor (via the resident system device handler) into upper core. It then gives control to the Keyboard Listener (within the non-resident Monitor) to await a BACKGROUND keyboard command.

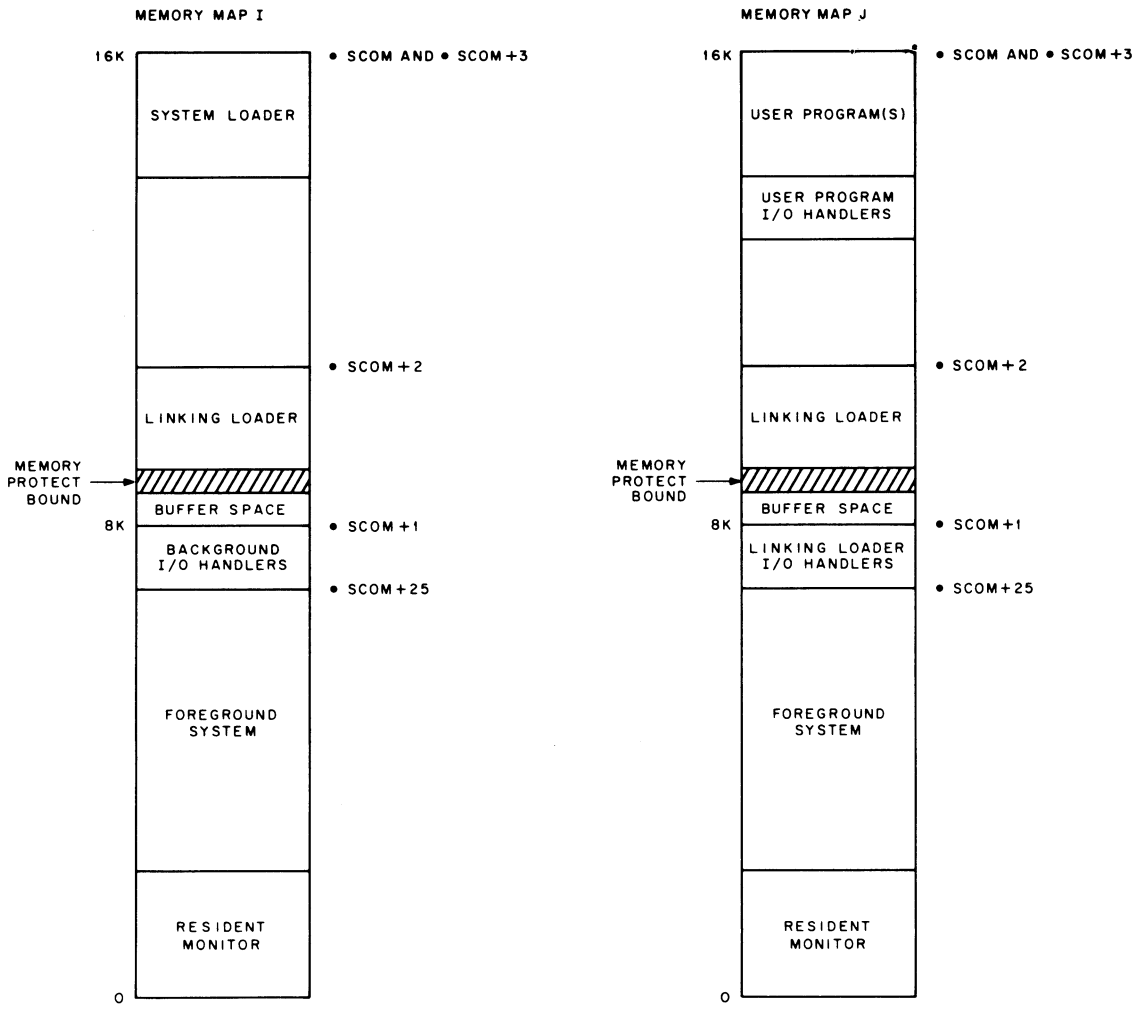
Figure 6-2 Background/Foreground Monitor System Memory Maps (Cont)



When a keyboard command requests loading of a BACKGROUND system or user program, the non-resident Monitor brings in the System Loader, overlaying itself.

If the BACKGROUND program is a system program, the System Loader loads the system program I/O handlers up from the top of the FOREGROUND system, allocates buffer space, and loads the system program at the top of core. It then sets the memory protect bound above the buffer space and gives control to the system program.

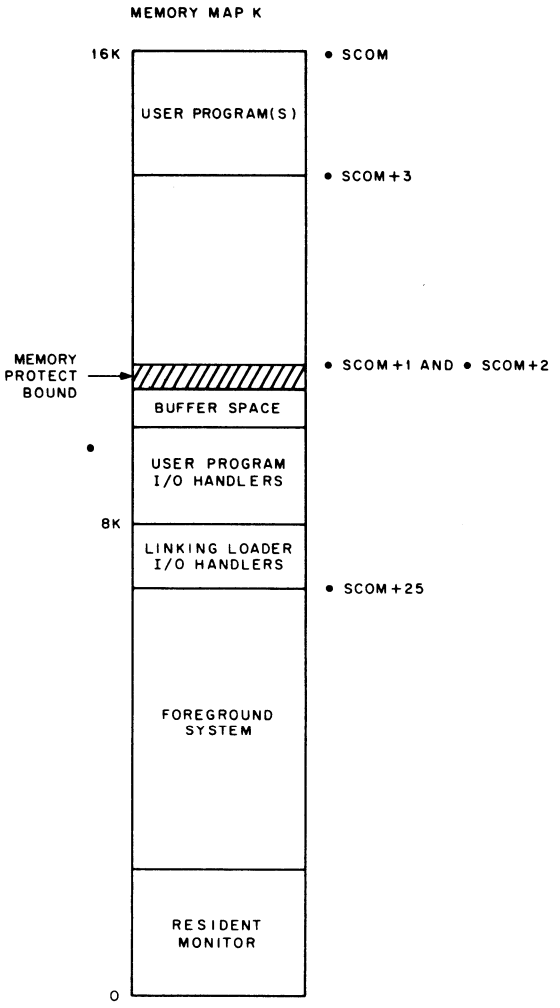
Figure 6-2 Background/Foreground Monitor System Memory Maps (Cont)



If the BACKGROUND program is a user program, the System Loader loads the Linking Loader I/O handlers up from the top of the FOREGROUND system, allocates buffer space, and loads the Linking Loader such that the memory protect bound can be set just below it.

The user's program, followed by its I/O handlers, are loaded down from the top of core. The I/O handlers are relocated to run just above the Linking Loader I/O handlers so that the memory protect bound can be set above them.

Figure 6-2 Background/Foreground Monitor System Memory Maps (Cont)



The .EXIT from the Linking Loader causes the user program I/O handlers to be block transferred to their running position, the memory protect bound to be set just above the buffer space, and control given to the user program.

Figure 6-2 Background/Foreground Monitor System Memory Maps (Cont)

CHAPTER 7
I/O DEVICE HANDLERS

This chapter contains information that is essential for a good understanding and proper use of I/O device handlers for the I/O and Keyboard Monitor systems. Included is a general description of I/O hardware and API software level handlers, a complete section on writing special I/O device handlers, a summary of I/O handlers acceptable to system programs, and a summary of standard I/O handler features.

It is assumed that the reader is familiar with all related material in the PDP-9 User Handbook (F-95), especially Chapter 9, Input/Output Considerations. It is also assumed that the reader is familiar with the PDP-9 Monitor environment and other pertinent information contained in this manual.

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

This section applies to I/O and Keyboard Monitor environments only.

7.1.1 I/O Device Handlers

All communications between user programs and I/O device handlers are made via CAL instructions (see Chapter 3) followed by argument lists. The CAL Handler in the Monitor performs preliminary setups, checks on the CAL calling sequence, and 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 through 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 re-entrant procedures before debreaking (DBK) from level 4. This permits the handler to receive re-entrant calls from software levels higher than the priority of the program that contained this call. If a device handler does not contain re-entrant procedures, system failure caused by inadvertent re-entries 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
XCT   .+1
JMP*
```

exit sequence to ensure that the exit takes place before the interrupt from the issued IOT occurs. (The XCT is necessary to ensure that the 3 cycles requested by the API on a debreak operation occur in the instruction after the DBR.)

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 care must be taken when executing CALs at this level. For example, a routine that is CAL'd 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.

7.1.1.1 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 JMPs 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 through 77 (see section
                                    /7.1.3 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.

7.1.2 API Software Level Handlers

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

7.1.2.1 Setting Up API Software Level Channel Registers - When the Monitor is loaded, the API software-level channel registers (40 through 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
        XCT             .+1
        JMP*            LV5INT          /AND RESTORE L, EX.MODE, MEM.PROT.
```

7.1.2.2 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 7-1 depicts the structure of a software level handler involved with queued requests.

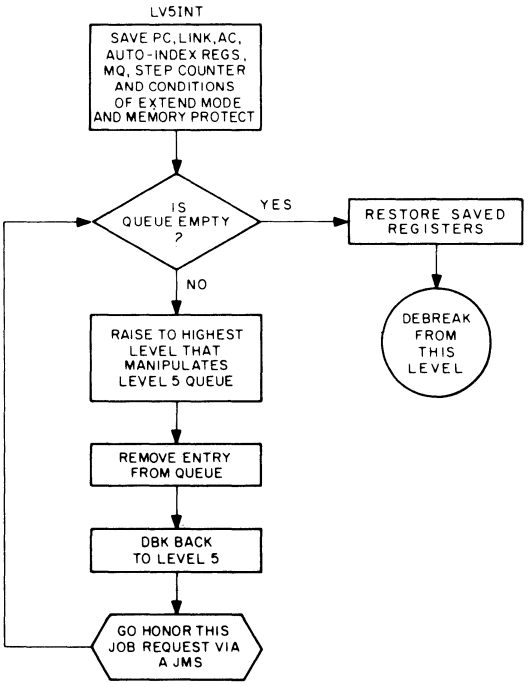


Figure 7-1 Structure of API Software Level Handler

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.

New resident handlers for Power Fail, Memory Parity, nonexistent memory violation, and Memory Protect violation have been incorporated into the system and effect an IOPS error message if the conditon is detected (see Appendix E for IOPS errors). The user can, via a .SETUP, tie his own handler to these skip IOT or API channel registers.

7.1.3 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 through 31 still unassigned.

7.2 WRITING SPECIAL I/O DEVICE HANDLERS

This section contains information prepared specifically to aid those users who plan to write their own special I/O device handlers for the I/O or Keyboard Monitor systems. (Information on special handlers for the Background/Foreground Monitor system will be available at a later date.) The PDP-9 Keyboard Monitor system is designed to enable users to incorporate their own device handlers; however, precautions should be taken when writing the handler to ensure compatibility with the Monitor.

Special handlers cannot be incorporated into the I/O Monitor system, but can be designed to run with user programs in the I/O Monitor environment (see Section 7.2.4). In an I/O Monitor system, if a user wishes to incorporate a special handler into a systems program (for example, a card punch handler for MACRO-9), he must purchase the source tape and assembly listings, modify them symbolically, and reassemble using at least a 16K machine.

It is assumed that the user is familiar with Section 7.1 of this chapter. To summarize, the handler is entered via a JMP from the Monitor as a result of a CAL instruction. The contents of the AC contain the address of the CAL in bits 3 through 17. Bit 0 contains the Link, bit 1 contains the extend mode status, and bit 2 contains the memory protect status. The previous contents of the AC and Link are lost.

To show the steps required in writing an I/O device handler, a complete handler (Example B) was developed with the aid of a skeleton handler (Example A). This handler is a non-reentrant type (discussed briefly at the beginning of this chapter) and uses the Debreak and Restore instruction (DBR) to leave the handler at software priority level 4 (if API), and restore the status of the Link, extend mode, and memory protect. Example A is referenced by part numbers to illustrate the development of Example B, a finished Analog to Digital Converter (ADC) I/O Handler. The ADC handler shown in Example B, was written for the Type AF01B Analog to Digital Converter, discussed on pages 4-26 and 4-27 of the PDP-9 User Handbook. This handler is used to read data from the ADC and store it in the user's line buffer. The handler shown in Example B is for instructional purposes only; it has not been thoroughly tested.

The reader, while looking at the skeleton of a specialized handler as shown in Example A, should make the following decisions about his own handler. (The decisions made in this case are in reference to developing the ADC handler):

a. Services that are required of the handler (flags, receiving or sending of data, etc.). By looking at the ADC IOT's shown in Chapter 4 of the Users Handbook, it can be seen that there are three IOT instructions to be implemented. These instructions are: Skip if Converter Flag Set; Select and Convert; and Read Converter Buffer.

The only service the ADC handler performs is that of receiving data and storing it in user specified areas. This handler will have a standard 256-word buffer.

b. Data Modes used (for example, IOPS ASCII, etc.). As there is only one format of input from the Type AF01B ADC, mode specification is unnecessary in Example C.

c. Which I/O macros are needed for the handler's specific use, that is, .INIT, .CLOSE, .READ, etc. These are fully described in Chapter 3 of this manual. For an ADC, the user would be concerned with three of the macros.

.INIT would be used to set up the associated API channel register and the interrupt skip IOT sequence in the Program Interrupt (PIC) skip chain. This is done by a CAL (N) as shown in Part III of Example A, where (N) is the channel address. The standard device/API channel associations can be found on Pages 12 and 13 of the Users Handbook.

.READ is used to transfer data from the ADC. When the .READ macro is issued, the ADC handler will initiate reading of the specified number of data words and then return control to the user. The analog input data received is in its raw form; it is up to the programmer to convert the data to a usable format.

.WAIT detects the availability of the user's buffer area and ensures that the I/O transfer is completed. It would be used to ensure a complete transfer before processing the requested data.

d. Implementation of the API or PIC interrupt service routine. Example A shows an API or PIC interrupt service routine that handles interrupts, processes the data and initiates new data requests to fully satisfy the .READ macro request. Note that the routines in Example A will operate with or without API. Example B used the routines exactly as they are shown in Example A.

During the actual writing of Example B, consideration was given to the implementation of the I/O Macros in the new handler in one of the following ways:

1. Execute the function in a manner appropriate to the given device as discussed in (c). .INIT, .READ, .WAIT were implemented into the ADC handler (Example B) under the subroutine names ADINIT, ADREAD, ADWAIT.

   Wait for completion of previous I/O. (Example B shows the setting of the ADUND switch in the ADREAD subroutine to indicate I/O underway.)

2. Ignore the function if meaningless to the device. See Example B (.FSTAT results in JMP ADIGN2) in the dispatch table DSPCH. For ignored macros, the return address must be incremented depending upon the argument string after the CAL. The number of arguments for each macro is shown in Chapter 3.

3. Issue an error message in the case where it is not possible to perform the I/O function. (An example would be trying to execute an .ENTER on the paper tape reader.) In Example B the handler jumps to DVERR6 which returns to the Monitor with a standard error code in the AC.

After the handler has been written and assembled, users who have a mass storage device must rebuild the Monitor system using the System Generator (SGEN), conveying to it the following information:

a. Answer "yes" for, "Are any other device handlers present?"

b. Give the total number of additional handlers.

c. Type handler names.

d. Type number of skip IOT's.

e. Type skip IOT's

f. Type skip chain, in the desired order of priority (usually with high speed devices first).

An example of System Generation is shown in the SGEN section of the Keyboard Monitor Guide (DEC-9A-NGBA-D).

When the system has been generated on a mass storage device, the system program UPDATE must be used to add the new handler to the library. At this time, the user is ready to use his specialized device handler in the PDP-9 system.

For the I/O Monitor (paper tape systems), the user must assemble the handler and splice it to the IOPS Library Tape (Number 1). This procedure is described following Example B.

7.2.1 Discussion of Example A by Parts

Part 1 Stores CAL pointer and argument pointer; also picks up function code from argument string.

Part 2 By getting proper function code in Part 1 and adding a JMP DSPCH, the CAL function is dispatched to the proper routine.

Part 3 This is the .SETUP CAL used to set up the API channel register and PIC skip chains. Section 7.1.3 of this manual shows the standard device/API associations.

Part 4 Shows the API and PIC handlers. It is suggested these be used as shown.

Part 5 This area reserved for processing interrupt and performing any additional I/O.

Part 6 Interrupt dismiss routine.

Part 7 Increments argument pointer in bypassing arguments of ignored macro CAL's.

7.2.2 Example A, Skeleton I/O Device Handler

Part 1

```
/SPECIALIZED I/O HANDLER
/CAL ENTRY ROUTINE
          .GLOBL DEV.                    /MUST BE OF FORM AAA.
.MED=3                                   /.MED (MONITOR ERROR DIAGNOSTIC)
DEV.      DAC       DVCALP               /SAVE CAL POINTER
          DAC       DVARGP               /AND ARGUMENT POINTER
          ISZ       DVARGP               /POINTS TO FUNCTION CODE
          LAC*      DVARGP               /GET CODE
          AND       (77777               /REMOVE UNIT # IF APPLICABLE
          ISZ       DVARGP               /POINTS TO CAL + 2
          TAD       (JMP DSPCH)
          DAC       DSPCH                /DISPATCH WITH
DSPCH     XX                             /MODIFIED JUMP
          JMP       DVINIT               /1 = /INIT
          JMP       DVFSAT               /2 = .FSTAT, .DELET, .RENAM
          JMP       DVSEEK               /3 = .SEEK
          JMP       DVENTR               /4 = .ENTER
          JMP       DVCLER               /5 = .CLEAR
          JMP       DVCLOS               /6 = .CLOSE
          JMP       DVMTAP               /7 = .MTAPE
          JMP       DVREAD               /10 = .READ
          JMP       DVWRTE               /11 = .WRITE
          JMP       DVWAIT               /12 = .WAIT
          JMP       DVTRAN               /13 = .TRAN
```

Part 2

```
/ILLEGAL FUNCTIONS IN ABOVE TABLE CODED AS:
/         JMP       DVERR6

/FUNCTION CODE ERROR
DVERR6    LAW 6                          /ERROR CODE 6
          JMP*      (MED+1)              /TO MONITOR

/DATA MODE ERROR
DVERR7    LAW 7                          /ERROR CODE 7
          JMP*      (.MED+1)             /TO MONITOR

/DEVICE NOT READY
DVERR4    LAC       (RETURN)             /RETURN (ADDRESS IN HANDLER)
                                         /TO RETURN TO WHEN NOT READY
                                         /CONDITION HAS BEEN REMOVED)
          DAC*      (.MED)
          LAC (4)                        /ERROR CODE 4
          JMP*      (.MED+1)             /TO MONITOR
```

```
/I/O UNDERWAY LOOP
DVBUSY    DBR                          /BREAK FROM LEVEL 4
          JMP*      DVCALP             /LOOP ON CAL

/NORMAL RETURN FROM CAL
DVCK      DBR                          /BREAK FROM LEVEL 4
          JMP*      DVARGP             /RETURN AFTER CAL AND
                                       /ARGUMENT STRING.

/THE DVINIT ROUTINE MUST INCLUDE
/A. SETUP FOR
/EACH FLAG CONNECTED TO PIC (AT BUILD TIME)
/ONE OF THESE MAY ALSO BE THE API SETUP CALL.
/THE SETUP CALLING SEQUENCE IS:

DVINIT    CAL       N                /N = API CHANNEL REGISTER
                                     /(40-77); 0 IF NOT
                                     /CONNECTED TO API

          16                         /IOPS FUNCTION CODE
          SKP IOT                    /SKIP IOT TO TEST THE FLAG
          DBVINT                     /ADDRESS OF INTERRUPT
                                     /HANDLER (API OR PIC)

/THIS SPACE CAN BE USED FOR I/O SUBROUTINES

/INTERRUPT HANDLER FOR API OR PIC
DEVPIC    DAC       DEVAC            /SAVE AC
          LAC*      (0)              /SACE PC, LINK, EX. MODE,
          DAC       DEVOUT           /MEM. PROT.
          LAC       DEVION           /FORCE ION AT DISMISSAL
          JMP       DVSTON
DEVINT    JMP       DEVPIC           /PIC ENTRY
          DAC       DEVAC            /API ENTRY, SAVE AC
          LAC       DEVINT           /SAVE PC, LINK, EX. MODE,
          DAC       DEVOUT           /MEM. PROT.
          IORS                       /CHECK STATUS OF PIC
          SMA!CLA                    /FOR RESTORATION AT
          LAW       17740            /DISMISSAL
          TAD       DEVION
DVSTON    DAC       DVSWCH
          DEVCF                      /CLEAR FLAG
DEVION    ION                        /ENABLE PIC

/THIS IS THE AREA DEVOTED TO PROCESSING INTERRUPT AND
/PERFORMING ANY ADDITIONAL I/O DESIRED.

          IOF                        /DISABLE PIC TC INSURE
          DEVIOT                     /DISMISSAL BEFORE
                                     /INTERRUPT FROM THIS
                                     /IOT OCCURS

/INTERRUPT HANDLER DISMISS RTE
DVDISM    LAC       (JMP DEVPIC)     /RESTORE PIC ENTRY
          DAC       DEVINT
          LAC       DEVAC            /RESTORE AC
DVSWCH    ION                        /ION OR IOF
          DBR                        /DEBREAK AND RESTORE
          JMP*      DEVOUT           /LINK, EX. MODE, MEM. PROT.
```

```
/IF THE HANDLER USES THE AUTO-INDEX
/OR
/EAE REGISTERS, THEIR CONTENTS
/SHOULD BE
/SAVED AND RESTORED.
/FUNCTIONS POSSIBLY IGNORED SHOULD
/CONTAIN PROPER INDEXING TO BYPASS
/ARGUMENT STRING.
DVIGN2    ISZ       DVARGP              /BYPASS FILE POINTER
          JMP       DVCK
```

Part 7

7.2.3 Example B, Special I/O Handler for Type AF01B A/D Converter

```
/ADC IOT'S
ADSF=701301
ADSC=701304
ADRB=701312
/ADSF=SKIP IF CONVERTER FLAG IS SET
/ADSC=SELECT AND CONVERT(ADC FLAG IS CLEARED
/AND A CONVERSION IS INITIALITED)
/ADRB=READ CONVERTER BUFFER(PUTS CONTENTS IN AC)
/CAL ENTRY ROUTINE
          .GLOBL ADC.                   /ADC. IS GLOBAL NAME FOR HANDLER
.MED=3                                  /MED(MONITOR ERROR DIAGNOSTIC)
ADC.      DAC ADCALP                    /SAVE CAL POINTER
          DAC ADARGP                    /AND ARGUMENT POINTER
          ISZ ADARGP                    /POINTS TO FUNCTION CODE
          LAC* ADARGP                   /GET CODE
          ISZ ADARGP                    /POINTS TO CAL + 2
          TAD (JMP DSPCH)
          DAC DSPCH                     /DISPATCH WITH
DSPCH     XX                            /MODIFIED JUMP
          JMP ADINIT                    /1 = .INIT
          JMP ADIGN2                    /2 = .FSTST, .DELET, .RENAM
          JMP ADIGN2                    /3 = .SEEK
          JMP ADFRR6                    /4 - .ENTER
          JMP ADFRR6                    /5 = .CLEAR
          JMP ADOK                      /6 = .CLOSE
          JMP ADOK                      /7 = .MTAPE
          JMP ADREAD                    /10 = .READ
          JMP ADERR6                    /11 = .WRITE
          JMP ADWAIT                    /12 = .WAIT
          JMP ADERR6                    /13 = .TRAN
/ILLEGAL FUNCTIONS IN ABOVE TABLE CODED AS:
/         JMP ADERR6
/FUNCTION CODE ERROR
ADERR6    LAW 6                         /ERROR CODE 6
          JMP* (.MED+1)                 /TO MONITOR
/DATA MODE ERROR
ADERR7    LAW 7                         /ERROR CODE 7
          JMP* (.MED+1)                 /TO MONITOR
```

```
/THE ADINT ROUTINE MUST INCLUDE A
/.SETUP FOR
/EACH FLAG ASSOCIATED WITH THE
/DEVICE
/THE .SETUP CALLING SEQUENCE IS:
ADINIT    ISZ ADARGP                    /IDX TO RET STNDRD BUFF SIZE
          .DEC
          LAC (256                      /STANDARD BUFFER SIZE (DECIMAL)
          .OCT
          DAC* ADARGP                   /PUT BACK STANDARD BUFFER SIZE
          ISZ ADARGP
          CAL 57                        /57=API CAHNNEL REGISTER
                                        /(40-77); 0 IF NOT
                                        /CONNECTED TO API
ADCKSM    16                            /.SETUP IOPS FUNCTION CODE
ADCBP     ADSF                          /ADC SKIP IOT TEST THE FLAG
ADLBHP    ADCINT                        /ADDRESS OF INTERRUPT
                                        /HANDLER (API OR PIC)
ADUND     LAC .+2                       /ERASES CALL FUNCTION IN CASE
ADWRC     DAC .-5                       /OF FURTHER .INIT'S THEY WILL BE
 ADWPCT   JMP ADSTOP                    /IGNORED BY THIS JMP TO ADSTOP
                                        /WHERE THE I/O UNDERWAY SWITCH
                                        /IS CLEARED AND ALL I/O  IS
                                        /TERMINATED
/THE PREVIOUS SIX TAGS IN THE CAL AREA ARE USED FOR TEMP
/STORAGE DURING THE ACTUAL .READ FUNCTION
/ADCKSM IS FOR STORING THE CHECKSUM
/ADDRP IS THE CURRENT BUFFER POINTER
/ADLRHP IS THE LINE BUFFER HEADER POINTER
/ADUND IS FOR DEVICE UNDERWAY SWITCH
/ADRWC IS USED AS A -WORD COUNT REGISTER
/ADRWCT IS USED TO STORE CURRENT WORD PAIR COUNT
/STOP ADC ROUTINE CLEARS I/O UNDER WAY SWITCH
ADSTOP DZM ADUND
/ADC WAIT LAC ADUND
          SNA
          JMP ADOK
/I/O UNDERWAY LOOP
ADBUSY    DBR
          JMP* ADCALP

ADREAD    LAC ADUND                     /CHECK TO SEE IF I/O IS UNDERWAY
          SZA!CMA                       /IF NOT SET IT WITH -1
          JMP ADBUSY                    /IT WAS SET, GO BACK TO CAL
          DAC ADUND                     /SET IT
          LAC* ADARGP                   /GET LINE BUFFER HEADER POINTER
          DAC ADDBP                     /STORE IT
          DAC ADLBHP                    /ALSO STORE IT FOR LATER HEADER
          ISZ ADARGP                    /INCREMENT ARG. POINTER
          LAC* ADRAGP                   /GET -L.B.W.C.(2'S COMP)
          DAC ADRWC                     /STORE IT IN WORD COUNT REGISTER
          ISZ ADARGP                    /INCREMENT FOR EXIT FROM .READ
          DZM ADWPCT                    /ZERO WORD PAIR COUNT REG.
          DZM ADCKSM                    /ZERO CHECKSUM REG.
          ISZ ADDBP                     /GET PAST HEADER PAIR
          ISZ ADDBP                     /NOW POINTING AT BEGINNING OF
                                        /BUFFER
          ADSC                          /START UP DEVICE
/NORMAL RETURN FROM CAL
ADOK      DBR                           /BREAK FROM LEVEL 4
          JMP* ADARGP                   /RETRUN AFTER CAL
```

```
/INTERRUPT HANDLER FOR API OR PIC
ADCPIC    DAC ADCAC                    /SAVE AC
          LAC* (Ø)                     /SAVE PC, LINK, EX. MODE
          DAC ADCOUT                   /MEM. PROT.
          LAC ADCION                   /FORCE ION AT DISMISSAL
          JMP ADSTON
ADCINT    JMP ADCPIC                   /PIC ENTRY.
          DAC ADCAC                    /API ENTRY, SAVE AC
          LAC ADCINT                   /SAVE PC,LINK,EX. MODE,
          DAC ADCOUT                   /MEM. PROT.
          LAC (JMP ADAPIC)             /RESTORE PIC ENTRY BECAUSE PIC
          DAC ADCINT                   /A JMS CALL NOT A JUMP
          IORS                         /CHECK STATUS OF PIC
          SMA!CLA                      /FOR RESTORATION AT
          LAW 17740                    /DISMISSAL
          TAD ADCION
ADSTON    DAC ADSWCH
          ADRB                         /READ CONVERTER BUFFER
 ADCION   ION                          /ENABLE PIC FOR OTHER DEVICES
          DAC* ADDBP                   /STORE DATA IN USER BUFFER
          ISZ ADDBP                    /INC. BUFFER POINTER
          ISZ ADWPCT                   /INC. WORD PAIR COUNTER
          TAD ADCKSM                   /ADD CHECKSUM
          DAC ADCKSM                   /STORE IT
          ISZ ADRWC                    /IS I/O COMPLETE
          JMP ADCONT                   /NO KEEP GOING
          LAC ADWPCT                   /YES,COMPUTE WORD COUNT PAIR
          TAD (1                       /ADD ONE MAY BE ODD
          RTL                          /DIVIDE BY TWO, AT SAME TIME ADD
          RTL                          /IT,FOR HEADER
          RTL
          RTL
          AND (377000                  /ALL SET
          DAC* ADLBHP                  /STORE IN HEADER #1
          ISZ ADLBHP                   /INC. TO STORE CKSUM
          TAD ADCKSM                   /ADD WORD PAIR COUNT
          DAC* ADLBHP                  /STORE IN HEADER #2
          DZM ADUND                    /CLEAR DEVICE UNDERWAY
          JMP ADDISM                   /EXIT
ADCONT    IOF                          /DISABLE PIC TO INSURE
          ADSC                         /DISMISSAL BEFORE
                                       /INTERRUPT FROM THIS
                                       /IOT OCCURS

/INTERRUPT HANDLER DISMISS RTE
ADDISM    LAC ADCAC                    /RESTORE AC
ADSWCH    ION                          /ION OR IOF
          DBR                          /DEBREAK AND RESTORE
          JMP* ADCOUT                  /LINK,EX.MODE,MEM. PROT.
ADCAIP    Ø                            /ADC CAL POINTER
ADARGP    Ø                            /ADC ARGUEMENT POINTER
ADCOUT    Ø                            /PC,L,FM,MP
ADCAC     Ø                            /AC SAVED HERE

/IF THE HANDLER USES THE AUTO-INDEX
/OR
/EAE REGISTERS, THEIR CONTENTS
/SHOULD BE
/SAVED AND RESTORE.
```

```
/FUNCTIONS POSSIBLY IGNORED SHOULD
/CONTAIN PROPER INDEXING TO BYPASS
/ARGUMENT STRING.
ADIGN2    ISZ ADARGP                    /BYPASS FILE POINTER
          JMP ADOK
          .END
```

7.2.4 Incorporating Special, User-Program I/O Handler into Paper Tape System

Proceed as follows to incorporate a nonstandard device handler into a user program in the I/O Monitor environment.

a. Assemble the new handler in relocatable binary form. After removing the end-of-file block (with the W switch in PIP), attach it to the I/O tape (No. 1) of the library.

b. Write a main program in which the new handler is declared as an external global:

```
.GLOBL    NUHDLR
```

Perform the following sequence of instructions prior to the execution of any monitor calls to the .DAT slot(s) to which this handler is to be attached:

```
LAC       NUHDLR
DAC*      (140          .     /.DAT SLOT #3
DAC*      (141          .     /.DAT SLOT #4
```

c. Do not put these .DAT slot numbers in a .IODEV psuedo-op.

d. On the first .INIT to this handler, the handler should:

1. Place the skip IOT in the skip chain in place of an unused skip IOT (avoid DRNEF)

PAPER TAPE SYSTEM SKIP CHAIN

LOC	SKP IOT	DEVICE
1511	SPFAL	POWER FAIL
1514	DRSF	DRUM DONE
1517	MTSF	MAGTAPE DONE
1522	CLSF	CLOCK OVERFLOW
1525	RCSF	CARD COLUMN READY
1530	RCSD	CARD DONE
1533	LSDF	LINE PRINTER DONE
1536	RSF	PAPER TAPE READER DONE
1541	PSF	PAPER TAPE PUNCH DONE
1544	KSF	KEYBOARD READY
1547	TSF	TELEPRINTER DONE
1552	MPSNE	NON-EXISTENT MEMORY
1555	MPSK	MEMORY PROTECT VIOLATION
1560	SPE	MEMORY PARITY ERROR
1563	DRNEF	NOT DRUM ERROR

An example is:

```
LAC       (skip IOT)        /REPLACE LINE
DAC*      (IE33)            /PRINTER SKIP IOT
```

2. Call .SETUP

```
CAL         AD          /ADDRESS OF API REGISTER
16                      /.SETUP FUNCTION CODE
SKIP IOT
DEVINT                  /ADDRESS OF INTERRUPT SERVICE
```

Where AD is the address of the associated API channel register; it is 0 if this device is not attached to the API.

e. The setup for communication with this nonstandard handler via .DAT slots 3 and 4 is complete.

NOTE

This procedure refers to the current Paper Tape System being distributed where .DAT slot Number 1 is at absolute register 136_8. A more modular verision is to be distributed in the future.

7.3 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 the I/O handlers that 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; that is, DTA. and DTB. should not be brought in together since there is no communication between the two interrupt handlers.

FORTRAN IV

.DAT Slot	Use	Handler
-11	Input	TTA PRA PRB (recommended) DTA, DKA, DRA, or MTA (required if 3 files open) DTB, DKB, DRB, or MTB (recommended if 2 files open) DTC, DKC, DRC, or MTC (recommended if input only) DTD, DKD, DRD, or MTD MTF (non-file-oriented) CDA CDB CDC
-12	Listing	TTA LPA PPA DTA, DKA, DRA, or MTA (required if 3 files open)

.DAT Slot	Use	Handler
		DTB, DKB, DRB, or MTB (recommended if 2 files open) DTD, DKD, DRD, or MTD (recommended if listing output only) MTF (non-file-oriented)
-13	Output	PPA PPB PPC (recommended) DTA, DKA, DRA, or MTA (required if 3 files open) DTB, DKB, DRB, or MTB (recommended if 2 files open) DTD, DKD, DRD, or MTD (recommended if listing output only) MTF (non-file-oriented)

MACRO-9

Identical to FORTRAN IV, with two exceptions (a) if .ABS binary output is requested on .DAT slot -13, PPC. and DTB. cannot be used. (b) .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
CDA
CDB
CDC

EDIT-9

.DAT Slot	Use	Handler
-10	Secondary Input	TTA PRA PRB (recommended) CDA CDB CDC

.DAT Slot	Use	Handler
-14	Input	TTA PRA PRB (recommended) DTA, DKA, DRA, or MTA (required if input and output) DTD, DKD, DRD, or MTD (input only) CDA CDB CDC
-15	Output	LPA TTA PPA DTA, DKA, DRA, or MTA (required if input and output) DTD, DKD, DRD, or MTD (output only)

Linking Loader and DDT

.DAT Slot	Use	Handler
-1	System Library	PRA DTA, DKA, or MTA DTB, DKB, or MTB DTC, DKC, or MTC (recommended if no user I/O) DTD, DKD, or MTD CDA CDB CDC
-4	Input	Same as for .DAT slot -1 plus DRA
-5	External User Library	DRB DRC DRD

NOTE

Since Linking Loader handlers can be used by the user program, choice of the bulk-storage handler should be a function of user requirements.

PIP

PIP uses all the positive .DAT slots (1 through 10) plus .DAT slot -3 for Teletype I/O.

Prior to PIP use, any non-standard peripheral device assignments should be made via the ASSIGN command to the PDP-9 Keyboard or Background/Foreground Monitor. 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 through 10) will avoid returning to the Monitor 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. Since the PDP-9 ADVANCED Software System includes more than one device handler for certain peripherals, those used with PIP should normally be the ones with the fullest capabilities, for example, PRA, PPA and DTA. If both input and output are to occur to the same type device (for example, DECtape), separate slots must be assigned. Both, however, must be assigned to the same handler, that is, 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 Generator

.DAT Slot	Use	Handler
-14, -10	Input	DTA, DKA, or MTA
-15	Output	DTA, DKA, or MTA DTD, DKD, or MTD

DUMP

-12	Listing	LPA TTA PPA
-14	Input	DTA, DKA, DRA, or MTA DTD, DKD, DRD, or MTD (recommended)

7-TO-9 CONVERTER

.DAT Slot	Use	Handler
-12	Listing	TTA LPA PPA DTA, DKA, DRA, or MTA (required if 3 files open) DTB, DKB, DRB, or MTB (recommended if 2 files open) DTD, DKD, DRD, or MTD (recommended if 1 file open)
-14	Input	TTA PRA PRB (recommended) DTA, DKA, DRA, or MTA (required if 3 files open) DTB, DKB, DRB or MTB (recommended if 2 files open) DTC, DKC, DRC, or MTC (recommended if input only) DTD, DKD, DRD, or MTD CDA CDB CDC
-15	Output	LPA TTA PPA DTA, DKA, DRA, or MTA (required if 3 files open) DTB, DKB, DRB, or MTB (recommended if 2 files open) DTD, DKD, DRD, or MTD (recommended if output only)

Libary Update

-10	Secondary Input	PRA DTA, DKA, DRA, or MTA CDA
-12	Listing	LPA TTA PPA DTA, DKA, DRA, or MTA CDA

.DAT Slot	Use	Handler
-14	Input	PRA DTA, DKA, DRA, or MTA
-15	Output	PPA PPB PPC DTA, DKA, DRA, or MTA

NOTE

The A version of bulk-storage handlers (DTA, DKA, etc.) can handle only three files simultaneously; any one of them should never be assigned to all four Library Update .DAT slots.

System Patch

.DAT Slot	Use	Handler
-14	System Device I/O	DTA, DKA, MTA DTD, DKD, MTD
-10	Secondary Input	PRA

NOTE

PATCH uses .DAT slot -3 for teletype output and .DAT slot -2 for input from the keyboard of batch input device. The user should never modify .DAT slots -3 or -2.

Chain Builder

.DAT Slot	Use	Handler
-6	Output	PPA PPB PPC DTA, DKA, DRA, or MTA DTB, DKB, DRB, or MTB (recommended) DTD, DKD, DRD, or MTD (only if no other DT, DK, etc., in -4, -5, and -1)

.DAT Slot	Use	Handler
-5	External Library	PRA CDA DTA, DKA, DRA, or MTA DTB, DKB, DRB, or MTB (recommended) DTC, DKC, DRC, or MTC (only if no other DT, DK, etc. in -6) DTD, DKD, DKD, or MTD (only if no other DT, DK, etc. in -6)
-4	User Programs	(Same as for .DAT -5)
-1	System Library	

NOTE

Use smallest handlers possible since they are not recoverable as user handlers.

Chain Execute

.DAT Slot	Use	Handler
-4	Chain Input	PRA CDA DTA, DKA, DRA, or MTA DTB, DKB, DRB, or MTB DTC, DKC, DRC, or MTC DTD, DKD, DRD, or MTD

NOTE

Use smallest "input-only" handler since user program may not use this handler.

7.4 SUMMARY OF STANDARD I/O HANDLER FEATURES

This section applies to I/O and Keyboard Monitor environments only.

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

297 (decimal) registers.

TTA*(TELETYPE)

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

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 through 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 — Allow input or output to finish.
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) tC transfers control to the address specified as return in the .INIT performed by KM9.
    (b) tP transfers control to the address specified as return in the .INIT performed by the user (other than KM9 or DDT).
    (c) tT transfers control to the address specified as return in the .INIT performed by DDT.
    (d) tS transfers control to the address specified in .SCOM + 6.
    (e) tQ transfers control to KM9SAV.

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

F. Data Control Characters-IN

1.	IMAGE Mode	All characters inserted in buffer as 7-bit characters.
2.	IOPS ASCII Mode	(a) Rubout. Delete previous character typed. Type out reverse slash (\).
		(b) tU 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

469_{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.

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 tR

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)	397 decimal registers
PPB. (all except IOPS ASCII)	270 decimal registers
PPC. (IOPS binary only)	210 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.

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)	436 decimal registers
PRB. (IOPS ASCII only)	287 decimal registers

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 71 through 77 on specified DECtape unit.

(b) Initializes DECtape Directory block 100 to indicate that eight blocks (71 through 100) are occupied.

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, that is, data will be recorded (starting at block 0) every fifth block. At EOT, recording continues in the reverse direction, etc. Five passes will record all 1100_8 blocks of a DECtape.

(b) BACKSPACE

(1) Decrements 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, examining 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 tR 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 on an .ENTER.
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).

---

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

		(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.
11.	Two output files on same unit	Monitor Error Code 22 (a) Two output files open simultaneously on the same unit.
12.	Illegal word pair count (WPC)	Monitor Error Code 23 (a) WPC = 0, or greater than 177.

E. Subprogram Description and Size

1. DTA (which requires 2321_{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. Files may be referenced on the same or different DECtape units, except that two or more output files may not be on the same unit. All data modes are handled as well as all IOPS functions except .MTAPE. 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 1552_{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 680_{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 1569_{10} locations)

DTD has full IOPS function capabilities including .MTAPE commands; 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.

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 (029) 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 (that is, 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 B (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 through 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 7-2.

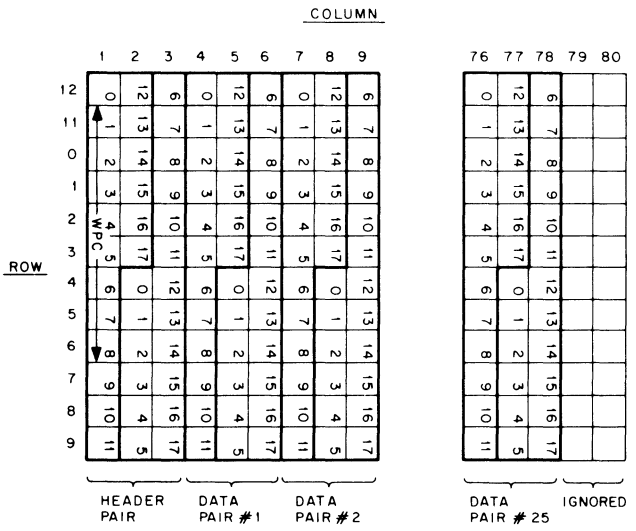


Figure 7-2 IOPS Binary Input Card Format

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 through 8 (column 1, rows 11 through 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 through 11 (column 1, rows 7 through 9): Rows 7 and 9 must be punched.

Bits 12 through 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 through 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 through 80 are ignored.

Header word 2 (column 2, rows 4-9; column 3) includes only, in bits 0 through 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 word (columns 4 through 78) on the card and word 1 of the header.

c. Dump (Mode 4) (52_{10} locations required to store 78 data columns)
Identical to IOPS Binary (2.b), except that no header pair appears on the card.

C. Recoverable Errors

1. Reader Not Ready (Monitor Error Code 4)

a. Hopper Empty
b. Stacker Full
c. Feed Check (may be hardware failure)
d. Read Check (may be hardware failure)
e. Reader not ready:
(1) Stop button depressed.
(2) Start button not depressed.
(3) Validity check with validity button on.

D. 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.

E. Program Size

1. CDA. (All modes): Approximately 450_{10} locations
2. CDB. (IOPS ASCII only): 411_{10} locations.
3. CDC. (IOPS ASCII only): Approximately 270_{10} locations.

F. 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 (for example, FORTRAN 4, MACRO).

Determination of the end of a card deck is performed as follows by CDA, CDB and CDC: on both the CR01E and CR02B, the presence of an EOF card (all 1's in column 1) signals the handler that an EOM (End-of-Medium) condition exists. The handler appropriately returns an EOM header (1006 in word 0) to the calling program.

An alternative method exists on the CR02B. The EOF button, if depressed with the hopper empty, is interpreted as an EOM condition. Typically, as the end of a card deck is reached, assuming no EOF card, an IOPS 4 message is output to the Teletype. Depressing the EOF button on the card reader and striking tR on the Teletype signal the end of the card deck.

DK (DISK)

A. Functions

1. .INIT
   (a) Return standard line buffer size (255_{10})
   (b) .SETUP - API channel register 47_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 = Disk 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 (71 through 77) on specified DISK unit.

(b) Initializes DISK Directory block 100 to indicate that eight blocks (71 though 100) are occupied.

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.

(2) Declares the unit as non-file-oriented, i.e., data will be recorded (starting at block 0) every fifth block. Five passes will record all 1100_8 blocks of a DISK unit.

(b) BACKSPACE

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

(c) Other .MTAPE functions - ignored.

10. .READ

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

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

11. .WRITE

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

(b) Outputs buffer when full, examining 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 DISK 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 DISK 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. Unrecoverable Errors

1. Illegal Function — Monitor Error Code 6

(a) See D 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 D 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. File Not Found — Monitor Error Code 13

(a) File name not found in Directory on a .SEEK.

6.	DISK Directory Full	Monitor Error Code 14
		(a) Directory Entry Section found full on an .ENTER
7.	DISK Unit full	Monitor Error Code 15
		(a) All DISK unit blocks occupied on a .WRITE
8.	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).
9.	Excessive Number of Files Referenced	Monitor Error Code 17
		See Section D for file reference limitations.
10.	Disk Failure	Monitor Error Code 20
		(a) Disk hardware malfunction
11.	Illegal Disk Address	Monitor Error Code 21
		(a) Reference to protected track
		(b) Reference to nonexistent track
12.	Two Output Files on Same Unit	Monitor Error Code 22
		(a) Two output files open simultaneously on the same unit.
13.	Illegal Word Pair Count (WPC)	Monitor Error Code 23
		(a) WPC = 0, or greater than 177
14.	Illegal Disk Unit	Monitor Error Code 27
		(a) Attempt to reference disk unit 3 or 7

D. Subprogram Description and Size

1. DKA (which requires 2272_{10} locations)

   DKA is the most general DISK handler issued with the PDP-9 ADVANCED Software System. DKA has a simultaneous 3-file capacity, either input or output. Files may be referenced on the same or different DISK units except that 2 or more output files may not be on the same unit. All data modes are handled as well as all IOPS functions (except .MTAPE). 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. DKB (which requires 1533_{10} locations)

   DKB has a simultaneous 2-file capacity: one input; one output. Both files may be on the same or different units. DKB transfers data only in IOPS ASCII or IOPS binary data modes. Included in the handler is space for two 256_{10}-word data buffers, one 32_{10}-word Directory Bit Map and one 32_{10}-word File Bit Map. Functions included are:

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

3. DKC (which requires 645_{10} locations)

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

.INIT	.CLOSE	.WAIT
.SEEK	.READ	

4. DKD (which requires 1533_{10} locations)

   DKD 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 DKD. One 256_{10}-word data buffer, one 32_{10}-word Directory Bit Map and one 32_{10}-word File Bit Map are included. DKD. is the only DISK handler which allows .MTAPE transfers.

DR (DRUM)

A. Functions

1. .INIT

   (a) Return standard line buffer size (255_{10}).

   (b) .SETUP - API channel register 46_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 = Drum 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) Examines 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 (71 - 77) on specified Drum unit.

(b) Initializes Drum Directory block 100 to indicate that eight blocks (71 - 100) are occupied.

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.

(2) Declares the unit as non-file-oriented, i.e., data will be recorded (starting at block 0) every fifth block. Five passes will record all 1100_8 blocks of a Drum unit.

(b) BACKSPACE

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

(c) Other .MTAPE functions are ignored.

10. .READ

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

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

11. .WRITE

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

(b) Outputs buffer when full, examining 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 Drum 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 Drum blocks, the remainder of the last block will be filled with 0s.

B. Legal Data Modes

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

C. Unrecoverable Errors

1. Illegal Function — Monitor Error Code 6
   (a) See D 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 D 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. File Not Found — Monitor Error Code 13
   (a) File name not found in Directory on a .SEEK
6. Drum Directory Full — Monitor Error Code 14
   (a) Directory Entry Section found full on an .ENTER
7. Drum Unit Full — Monitor Error Code 15
   (a) All Drum unit blocks occupied on a .WRITE
8. 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).
9. Excessive Number of Files Referenced — Monitor Error Code 17
   See Section D for file reference limitations.
10. Illegal Drum Sector Address — Monitor Error Code 25
   (a) Sector address greater than physical drum size.
   (b) For unit 1 or greater, sector address greater than 777_8
11. Illegal Drum Size — Monitor Error Code 35

(a) .SCOM + 4, Bits 15 - 17, not set up (1-5)

12. Two Output Files on Same Unit

Monitor Error Code 22

(a) Two output files open simultaneously on the same unit.

13. Illegal Word Pair Count (WPC)

Monitor Error Code 23

(a) WPC = 0, or greater than 177

14. Illegal Drum Unit

Monitor Error Code 26

(a) 32, 65 or 131K Drum: Unit #>0

(b) 262K Drum: Unit #>1

(c) 524K Drum: Unit #>3

D. Subprogram Description and Size

1. DRA (which requires 2288_{10} locations)

DRA is the most general Drum handler issued with the PDP-9 ADVANCED Software System. DRA has a simultaneous 3-file capacity, either input or output. Files may be referenced on the same or different Drum units except that 2 or more output files may not be on the same unit. All data modes are handled as well as all IOPS functions (except .MTAPE). 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. DRB (which requires 1473_{10} locations)

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

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

3. DRC (which requires 601_{10} locations)

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

.INIT	.CLOSE	.WAIT
.SEEK	.READ	

4. DRD (which requires 1478_{10} locations)

DRD has full IOPS function capabilities; however, it allows for only one file-reference, either input or output, at any given time. Sequential file references are permitted.

All data modes are acceptable to DRD. One 256_{10}-word data buffer, one 32_{10}-word Directory Bit Map and one 32_{10}-word File Bit Map are included. DRD. is the only Drum handler which allows .MTAPE transfers.

MT (MAGNETIC TAPE)

A. Functions

1. .INIT

(a) Return standard buffer size (255_{10}).

(b) Call .SETUP - API Channel Register 45_8.

(c) Ensure that the referenced transport is not currently open for output transfers in the file-structured environment. If the drive is not open for writing or is non-file-structured, continue. If the drive is open for output and is file-structured, overlay the first block (header label) with a logical end-of-tape indicator.

(d) Set transfer direction (input or output).

(e) Indicate that the decision with respect to file-structuring has not been made.

(f) If first .INIT to this unit, assign default parity, density, and track-count settings for this unit. Parity is odd, density is 800 BPI, and track-count is specified by bit 6 of .SCOM + 4 (0 = 7-channel, 1 = 9-channel).

(g) Indicate that the referenced drive is open for I/O transfers.

2. .OPER (.DLETE)
(.RENAM)
(.FSTAT)

(a) .DLETE

(1) Examine the directory on the referenced unit for a file of the name provided. If no file is found, return to the user with the AC = 0.

(2) If file is found, remove the name found from the Directory, zero the applicable accessibility bit, decrement the active file count, and re-record the Directory.

(3) Return with the AC $\neq$ 0.

(b) .RENAM

(1) Search the Directory for an active file of the name given. If no file is found, return to the user with the AC = 0.

(2) If file is found, replace the Directory file name entry with the new file name and re-record the Directory.

(3) Return with the AC $\neq$ 0.

(c) .FSTAT

(1) Set bits 0 through 2 of CAL + 2 = 4.

(2) Search Directory for a file of the name given. If no file is found, return with the AC = 0.

(3) If a file is found, return with the AC = relative position of the file on tape (1 through 374_8).

3. .SEEK

(a) Determine if the referenced unit is file-structured. If it is not, take error return (IOPS 7). If no decision has been made, declare the unit file-structured.

(b) Ensure that no file is open on the referenced unit. Take error return (IOPS 10) if so.

(c) Ensure that the referenced unit has been .INIT'ed for input. Take error return (IOPS 7) if not.

(d) Search directory for a file of the name given. If no file is found, error return (IOPS 13) to Monitor.

(e) Physically position the tape to read the first data block on the file. The handler-calculated file name must match the file name in the header label (IOPS 40 if not).

(f) Indicate a file open for reading on the referenced unit.

4. .ENTER

(a) Determine if the referenced unit is file-structured. If it is not, take error return (IOPS 7). If no decision has been made, declare the unit to be file-structured.

(b) Ensure that no file is open on the referenced unit. Take error return (IOPS 10) if so.

(c) Ensure that the referenced unit has been declared an output unit. Error return (IOPS 7) if not.

(d) Ensure that space is available in the File Name Entry Section of the Directory for this file name. Take error return (IOPS 14) if not.

(e) Ensure that space is available in the Accessibility Map for this file. Take error return (IOPS 42) if not.

(f) Indicate that a file is open for writing on the unit referenced.

5. .CLEAR

(a) Determine if the referenced unit is file-structured. If not, take error return (IOPS 7). If no decision has been made, declare the unit file-structured.

(b) Rewind and write an empty File Directory at the front of the tape, along with a Logical End-of-Tape indicator.

6. .CLOSE

(a) Input: Indicate that the referenced unit is no longer available for I/O transfers; return to caller.

(b) Output:

(1) Non-File-Structured Tape: Write two end-of-file markers,

then backspace one to position the recording head between the two EOF markers written. Indicate that unit is no longer open for I/O transfers, and return to caller.

(2) File-Structured Tape

a. Write the partial data buffer, if one is present.

b. Write trailer label and logical end-of-tape indicator.

c. Search the File Directory for a name identical to that of the file being closed. If one is found, remove it from the directory and set its accessibility bit to zero.

d. Add the new file name at the bottom of the Directory.

e. Update total and active file counts.

f. Re-record the Directory.

g. Indicate that unit is no longer open for I/O transfers, and return to caller.

7. .MTAPE

(a) Determine whether unit if file-structured; if so, take error return (IOPS 7). If no decision has been made, declare unit non-file-structured.

(b) Honor subfunction specification as follows:

00 Rewind: Issue rewind to drive specified.

01 Undefined: Error Return (IOPS 6).

02 Backspace Record: Issue a single backspace to the drive specified.

03 Backspace File: Backspace until two EOF markers have been passed in reverse, then space forward one record.

04 Write EOF: Write one EOF marker.

05 Space Forward Record: Issue a single space forward to the drive specified.

06 Space Forward File: Space forward until a single EOF marker is passed.

07 Space to Logical EOT: Space forward until two consecutive EOF markers are passed, then backspace one record.

10 through 17 Describe Tape Configuration: Update the tape format descriptor bits for the drive specified. Subsequent I/O transfers (including space) will be performed in the density, parity, and channel-count given in .MTAPE 10 - 17, thus:

Sub-function	Channel Count	Parity	Density
10	7	Even	200 BPI
11	7	Even	556 BPI
12	7	Even	800 BPI
13	7	Odd	200 BPI
14	7	Odd	556 BPI
15	7	Odd	800 BPI
16	9	Even	800 BPI
17	9	Odd	800 BPI

8. .READ

(a) Ensure that referenced unit is input. IOPS 7 if not.

(b) Ensure that a file is open for reading or unit is non-file-structured. IOPS 11 if not.

(c) Initiate data transfer as described in Section 5.9.4.

(d) Read Errors

(1) Parity/Checksum Errors - As described in Section 5.9.4.

(2) EOF Encountered.

a. File-Structured Environment.

Modes 0 - 4: An EOF pseudo-line is constructed and stored in the user's line buffer area. The format of the 2-word line is as follows:

Header word 0: 001005
Header word 1: 776773
Mode 5: Illegal in file-structured environment.

b. Non-File-Structured Environment.

Modes 0 - 3: An EOF pseudo-line is constructed and stored in the user's line buffer area. The format of the line is as follows:

Header word 0: 001005
Header word 1: 776773
Data word 0: 000000
Data word 1: Unchanged

Modes 4 - 5: No indication of End-of-File is currently provided.

(3) EOT Encountered

a. File-Structured Environment.

Modes 0 - 4: An EOM pseudo-line is constructed and stored in the user's line buffer area. The format of the 2-word line is as follows:

Header word 0: 001006
Header word 1: 776772
Mode 5: Illegal in file-structured environment.

b. Non-File-Structured Environment.

Modes 0 - 3: Exactly as described for file-structured environment (3a above).

Modes 4 - 5: Error return (IOPS 43).

9. .WRITE

(a) Ensure that referenced unit is output. IOPS 7 if not.

(b) Ensure that a file is open for writing or that unit is non-file-structured. IOPS 11 if not.

(c) Initiate data transfer as described in Section 5.9.4.

(d) EOT: When physical End-of-Tape is encountered during writing, an error return (IOPS 15) is made to the Monitor. Before control is given to the Monitor, the file being written is added to the directory with the final two characters of the extension as "XX".

(e) Write Errors: Continued attempts are made to rewrite the record in error. The process terminates when EOT is encountered.

10. .WAIT, .WAITR

(a) Check I/O underway.

(1) Busy: Return to CAL (.WAIT) or to address in CAL+2 (.WAITR).

(2) Non-Busy: Return to CAL+2 (.WAIT) or to CAL+3 (.WAITR).

11. .TRAN

Honor subfunction indicator as follows:

0 Input Forward: Transfer next physical block on tape directly to user's buffer area.

1 Output Forward: Transfer directly from user's buffer to the next physical block on tape.

2, 3 Illegal; Take error return (IOPS 6) to Monitor.

B. Legal Data Modes

Mode 0 IOPS Binary

Mode 1 Image Binary

Mode 2 IOPS ASCII

Mode 3 Image Alphanumeric

Mode 4 Dump

Mode 5 9-Channel Dump (Legal for 9-channel transports only)

1. IOPS Binary (Mode 0)

   (a) Output

      (1) File-Structured Tape

      An attempt is made to pack the binary line into a 257_{10}-word buffer internal to MTA. If the line will not fit, the current contents of the buffer are written and the line transmitted begins a new buffer. The line checksum is computed and stored in the second word of the line as it appears in MTA.'s buffer; the user's line-buffer checksum word is undisturbed. The buffer checksum (BCP word 2) is updated. Bits 12-13 in the user's line (in MTA.'s buffer) are set to 00.

      The maximum length of the line buffer, including the header pair, is 254_{10} words. The first word of the header is checked to ensure that the word-pair count is less than or equal to 177_8 and greater than 0. A word-pair count equal to zero or greater than 177_8 results in an error return (IOPS 23) to the Monitor.

      (2) Non-File-Structured Tape

      A check is made to ensure that the word-pair count is greater than zero. A 0 count results in an immediate error return (IOPS 23) to the Monitor. No check is made on the upper limit of the word-pair count; anything from 1-377_8 is legal. The checksum is computed and stored in the second word of the line in the user's line buffer area. Bits 12 - 13 of this first header word are set to zero. The count of words to write is taken from the word-pair count in the header and transfer from the user's area is initiated.

   (b) Input

      (1) File-Structured Tape

      The line called for is unpacked from a 257_{10}-word buffer internal to MTA. If the buffer was emptied by a previous .READ, or if this .READ is the first one, the buffer is refilled from the next physical block on tape. The line is stored in the user's line buffer area. Transmission from MTA.'s buffer stops when (a) the word-pair count in the input line or (b) the word count in the CAL sequence is satisfied, whichever occurs first. In either case, the next-line pointer indicates the true subsequent line. In case of buffer overflow, bits 12 and 13 of the first header word are raised. (If buffer overflow does occur, the untransmitted portion of the line is no longer available to the caller.)

      Whether buffer overflow occurs or not, the validity bits (12 - 13) of the first header word are modified as follows and in the order indicated. First, the checksum for the line is calculated; if it is different from the transmitted checksum, bits 12 - 13 are set to 10. Next, a check is made for successful transfer of the entire block. In this context, "Successful Transfer" means (a) the block was read without hardware-detected error and (b) the block checksum (BCP Word 2) is correct. If transfer was unsuccessful, bits 12 - 13 are set to 01.

(2) Non-File-Structured Tape

The count of words to transfer is taken from CAL sequence, and input is initiated from the next physical block on tape directly to the user's line-buffer area. When the read is complete, the line validity bits are modified under the following conditions and in the order indicated. First, bits 12 - 13 of header word 0 are set if buffer overflow occurred. Next, a checksum is calculated (only if buffer overflow did not occur) and compared with the checksum read. If the two checksums differ, bits 12 - 13 are set to 10. Finally, a check is made to ensure that the line was transferred without hardware-detected error. If an error occurred, bits 12 - 13 are set to 01. If no errors of the types described are encountered, bits 12 - 13 are unchanged.

2. Image Binary (Mode 1) - Handler activity is exactly as described for IOPS binary, above. Headers and data are transferred in both file-structured and non-file-structured modes. Modifications are limited to the checksum word and the validity field as outlined above.

3. IOPS ASCII (Mode 2) - Handler activity is exactly as described for IOPS Binary, above.

4. Image Alphanumeric (Mode 3)

(a) File-Structured Tape

Handler activity is exactly as described for IOPS binary, above. In the file-structured environment, headers and data are transferred and modifications, when applicable, are carried out only on the checksum word and validity field.

(b) Non-File-Structured Tape

Image Alphanumeric Mode in the non-file-structured environment is intended to provide the user with a facility for reading and writing on tape, the alphanumeric character codes of his choice. The Mode 3 line appears in core following a .READ and prior to a .WRITE as two header words and a number of data words as reflected by the word pair count. Only the data portion of the line is transmitted to or from the TCU, however. On output, the header pair is discarded before transfer begins; on input, the header pair is constructed when the read is complete.

The method of transfer depends upon the characteristics of the tape unit referenced. If the unit is 7-channel, then full 18-bit words are transferred. Each of the three 6-bit bytes in the PDP-9 register occupies a single 7-bit frame on the tape. If the unit is 9-channel, then each of two 8-bit bytes (bits 2 - 9 and 10 - 17) occupies a single 9-bit frame. Only two frames (a total of 16 bits) are transferred to or from 9-channel tape.

(1) Output

The count of words (including the header pair) in the line to be written is taken from the first header word. Transfer is initiated directly from the user's line buffer, beginning at the first data word.

(2) Input

The input block is read directly into the user's line buffer area, beginning at the address which is two greater than that given in the CAL sequence. The count given in the CAL sequence is interpreted as the maximum number, less two, of data words to read. When the transfer is complete, a header is constructed and stored in the line buffer. Possible errors include only buffer overflow (validity bits = 11) or parity error (validity bits = 01).

5. Dump (Mode 4) - Dump mode is used to read into or write from specified areas of core, under count control, without the need for line buffers. The action taken by MTA. in honoring dump-mode .READ's and .WRITE's is identical in both file-structured and non-file-structured environments.

(a) Output

Data are taken from the core area specified in the CAL sequence and stored starting in the next available place in MTA.'s buffer. When the buffer is filled, it is written out and transmission to the new buffer continues until the count in the CAL sequence is fulfilled. The partly-filled buffer, if one remains, is not written at the completion of the operation. Data are transferred in 255_{10}-word increments. The dump mode buffer as written includes the BCP for a total block length on tape of 257_{10} words.

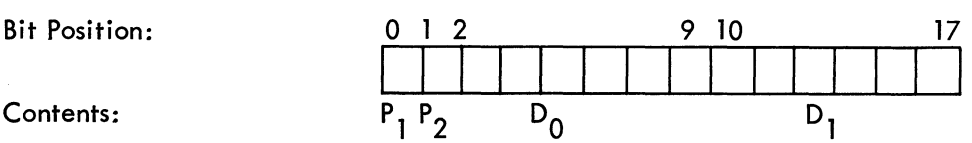
(b) Input

Data are taken from the handler buffer and stored sequentially starting at the core location given in the CAL argument list. Transmission continues until the word count in the CAL sequence is satisfied. If the handler buffer is emptied in the process, it is refilled from the next physical block on tape.

(c) Read/Write Errors

There is presently no facility for indicating I/O errors to the caller while dump mode is being used.

6. 9-Channel Dump (Mode 5) - This mode, device-dependent for 9-channel magnetic tape, is designed to make use of all 8 data bits in each frame of tape, whereas all other modes except 9-channel Image Alphanumeric require three frames for each data word written. Transfers in Mode 5 require only two frames for each word stored. Only 16 bits of each PDP-9 word, however, are read from or written on tape. Word format for Mode 5 is shown below.



Parity Bits (P_0, P_1): Bits 0 and 1 of the PDP-9 word are used as parity bits for the two 8-bit data bytes (D_0 and D_1), respectively, in the low-order portion of the register. During an output (write) transfer, these bits are ignored. The hardware generates the proper parity for each data f rame. During an input (read) transfer, these bits are set to the actual parity values for the two data frames as read from tape.

Data Bytes (D_0, D_1): Bits 2 - 9 and 10 - 17 hold values of two adjacent 8-bit frames. During reads and writes, D_0 is the first frame transferred. If a record containing an odd number of frames is read, the final frame is stored in D_0; D_1 is set to binary 0s. An even number of frames is always written during an output operation.

Mode 5 transfers may be performed only in the non-file-structured environment on 9-channel tape. An attempt to read or write a file-structured 9-channel tape or any 7-channel tape causes an error return (IOPS 7) to the Monitor.

Data are transferred directly between the user's buffer area and the control. No buffering or editing of any kind is performed by MTA. On output, the exact count of words to write is taken from the CAL sequence. On input, the count is interpreted as the maximum number of words to read. Each .WRITE results in exactly one physical block on tape; each .READ results in the input of one record (or as much of that record as will fit).

There is presently no facility for indicating I/O errors to the caller while Mode 5 is being used.

C. Recoverable Errors

1. Transport not ready. IOPS 4

   Results From:

   (a) Write request with write lock.

   (b) Transport off line or not dialed up.

   (c) 9-Channel I/O request to 7-channel transport, and vice-versa.

   Remedy:

   (a) Ready the requested magnetic tape unit.

   (b) Type CONTROL R on the Teletype.

D. Unrecoverable Errors

1. Illegal Function. IOPS 6

   (a) Any file-structured request to a non-file-structured transport.

   (b) Any non-file-structured request to a file-structured transport.

(c) Input request to output unit; output request to input unit.

(d) .TRAN with reverse direction specified.

(e) Additional restrictions outlined below.

2. Illegal Data Mode IOPS 7

   (a) Mode 5 transfer request to file-structured transport.

   (b) Additional restrictions outlined below.

3. File Still Active IOPS 10

   .SEEK, .ENTER, .CLEAR, OR .OPER requested while a file is still open on the specified unit.

4. .SEEK/.ENTER Not Executed IOPS 11

   Before a .SEEK/.ENTER is executed, a .READ/.WRITE is requested to a transport which has been declared file-structured.

5. EOT Encountered on Read IOPS 12

   Physical End-of-Tape encountered during an input operation.

6. File Not Found IOPS 13

   During processing of .SEEK, the requested file name is absent from the File Name Entry Section of the specified Directory.

7. Directory Overflow IOPS 14

   During processing of .ENTER, the File Name Entry Section of the Directory is discovered to be full.

8. EOT Encountered on Write IOPS 15

   Physical End-of-Tape encountered during an output operation.

9. Output Buffer Overflow IOPS 16

   IOPS ASCII or IOPS Binary line exceed 255_{10} words (including header).

10. Too Many Files IOPS 17

    An excessive number of files are concurrently referenced

11. Word Pair Count Error IOPS 23

    During a transfer in IOPS ASCII or IOPS Binary data mode, the Word Pair Count is found to be less than 1 or greater than 177_8.

12. Header Label Error IOPS 40

    During the processing of .SEEK, the handler-calculated file name is discovered to be different from that present in the file header label.

13. Accessibility Map Overflow IOPS 42

    During the processing of .ENTER, the Accessibility Map is found to be full.

14. Directory Recording Error IOPS 43

A write error is encountered during Directory recording.

E. Handler Description

1. MTA.

The most general of the magnetic tape handlers, MTA. allows concurrent reference to a maximum of three files, either input or output. All functions and data modes are honored.

2. MTB.

MTB. is an input-output handler which allows concurrent reference to two files.. Transfers are honored only in IOPS ASCII and IOPS Binary data modes, and only the following functions are allowed:

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

3. MTC.

Designed for input operations, MTC. is a read-only handler with the capacity for operating on a single file. Sequential file references are allowed. Transfers are honored only in IOPS ASCII and IOPS Binary data modes. Legal functions are limited to the following:

.INIT	.CLOSE	.WAIT
.SEEK	.READ	.WAITR

4. MTD.

MTD. honors all IOPS functions and data modes. Only a single file however, may be open for transfers at any time. Sequential file references are permitted.

5. MTF.

Designed for the user operating in the FORTRAN environment, MTF. offers functions and data modes limited to those required by the FORTRAN Object Time system. Data modes include only IOPS ASCII and IOPS Binary. The device is always considered non-file-structured. Functions honored include:

.INIT	.READ	.WAIT	.MTAPE Rewind
.CLOSE	.WRITE	.WAITR	.MTAPE Backspace

MTF. allows a maximum of eight concurrently-open transports.

APPENDIX A

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		S	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
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.

APPENDIX B

PDP-9 ASCII/HOLLERITH CORRESPONDENCE

	*00-37	44-77		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
33	ESC	;	11-6-8	[	c 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 C

KEYBOARD AND BACKGROUND/FOREGROUND MONITOR ERRORS

Errors	Explanation
WHAT?	Unrecognizable command
BAD DEV - IGNORED FROM ERR	Illegal device reference, for example: A PRA 5,6/PPW7/DTA-5 where the command is processed and effective up to the PPW and the remainder of the command is ignored.
BAD .DAT SLOT - IGNORED FROM ERR	Illegal .DAT slot reference, for example: A PRA 5,6/PPA G where the command is processed and effective through A PRA 5,6 but ignored from there on.
BAD PRGNAM	Non-existent program name. Command ignored.
PERMANENT .DAT SLOT - IGNORED FROM ERR	Command attempted to assign a device handler to one of the permanent .DAT slots (-2, -3, or -7).
BAD UNIT - IGNORED FROM ERR	Illegal unit reference (e.g., DTAX)
BAD START LOC	Illegal address given in "GET n address" command.
SYS DEV ERR - CHECK UNIT & TRY AGAIN	Last command typed caused error condition on system device control.
BAD COMMAND IN BATCH MODE	Illegal Batch Processor commands: QDUMP, HALT, GET (all forms), BATCH, LOAD, DDT, or DDTNS.
BAD BATCH DEV	Batch device was not designated properly. Should be: CD - for card reader PR - for paper tape reader
BAD $JOB COMMAND	$JOB command not terminated by space, carriage return, or ALT MODE.

APPENDIX D

LINKING LOADER AND SYSTEM LOADER ERRORS

The following error codes are output by the Linking Loader and by the System Loader. When output by the Linking Loader, the errors are identified as shown below. When output by the System Loader, the errors are identified as ".SYSLD n" instead of ".LOAD n."

Error	Meaning
.LOAD 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.
.LOAD 2	Input data error - parity error, checksum error, illegal data code, or buffer overflow (input line bigger than Loader's buffer).
.LOAD 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.
.LOAD 4	Illegal .DAT slot request - the .DAT slot requested was: a. Out of range of legal .DAT slot numbers, b. Zero, c. Unassigned, that is, was not set up at System Generation Time or (in the case of the Keyboard and Background/Foreground Monitors) was not set up by an ASSIGN command. d. Or, in the Background/Foreground system, the BACKGROUND program requested a .DAT slot which called for an I/O handler and device number which conflicted with the FOREGROUND job's I/O.

APPENDIX E

IOPS ERRORS

Error Code	Error	Error Data	Comments
0	Illegal Function CAL	CAL address	The address points to a CAL which did not have a legal function code (1 to 16) in bits 3 to 17 of the word after the CAL.
1	CAL * illegal	CAL address	The instruction CAL* (Indirect) is an illegal Monitor CAL.
2	.DAT slot error	CAL address	1. The .DAT slot number in Bits 9 to 17 of the CAL was 0, greater than 10, or less than -15. 2. The .DAT slot did not contain a handler address (no .IODEV was given for this .DAT slot).
3	Illegal interrupt	I/O status	An interrupt occurred which did not have an active device handler associated with it. The contents of the IORS word at the time of the interrupt is printed out.
4	Device not ready (type control R when ready)		This error can occur whenever any not ready condition occurs. 1. DECtape or MAGtape - unit not selected or not write enabled. 2. Punch - out paper tape 3. Line printer - off line 4. Card reader - off line, out of cards, stacks full, or card jam 5. Disk - massive failure 6. Drum - attempt to write on locked out area or non-existent track, timing errors.
5	Illegal .SETUP CAL	CAL address	Use of .SETUP when appropriate skip not placed in skip chain at system generation time.
6	Illegal handler function	CAL address	A function (.READ, .WRITE, etc.) was issued to a handler which is incapable of performing that function (.READ to paper tape punch, .WRITE to C version of handler (Read only)).
7	Illegal data mode	CAL address	1. Illegal data mode for this version of the handler used. 2. Use of input commands after device has been .INITed for output.
10	File still active	CAL address	Failure to close a file before another seek or enter on the same .DAT slot.
11	SEEK/ENTER not executed	CAL address	A read or write was issued without a prior SEEK, ENTER, or MTAP command.
12	Unrecoverable DECtape error	DECtape status register B and Unit Number	DECtape error with status register B in bits 0 to 11 and the unit # in bits 15 to 17.

Error Code	Error	Error Data	Comments
13	File not found	CAL address	The file name specified by the directory entry section (pointer to entry is in CAL address plus 2) was not found.
14	Directory full	CAL address	The directory entry section of the current device in use is full.
15	DECtape full	CAL address	All blocks available for file storage are currently full.
16	Output buffer overflow	CAL address	The word pair count on the current .WRITE is greater than 177_8.
17	Too many files for handler	CAL address	Too many files are currently open on the handler referenced by this CAL (e.g., 4 files on DTA will cause error while 2 files on DTD would cause same error).
20	Disk failure	Disk status register	Disk failure with status register printed out.
21	Illegal disk address	CAL address	The CAL pointed to caused the disk control to reference an illegal or write protected address.
22	Two output files on one unit	CAL address	Two concurrent output files have been opened on one unit.
23	Illegal Word Pair Count	Sector address	The word pair count on the current input or output line equals zero or greater than 177_8.
24	339 Handler called without push down list set up	CAL address	An attempt was made to use the 339 handler without reserving an area for the push down list.
25	Illegal sector address	CAL address	The current read or write CAL caused an illegal drum sector address to be used.
26	Illegal Drum unit	CAL address	The unit number in bits 0 to 2 of the word following the CAL is illegal for the drum size specified in bits 15 to 77 of .SCOM + 4.
27	Illegal disk unit	CAL address	The unit number in bits 0 to 2 of the word after the CAL is 3 or 7 which are both non-existent on the Disk.
30	API software level error	API status register	An API break occurred to a software level which did not have the appropriate transfer vector set up in .SCOM + 12 to .SCOM + 15.
31	Non-existent memory reference	Program counter	Non-existent memory reference with protect mode on without a user defined violation routine.
32	Memory protect violation	Program Counter	Reference to a location below the memory protect boundary without a user defined violation routine.
33	Memory parity error	Program counter	Memory parity error without a user defined parity error routine.

Error Code	Error	Error Data	Comments
34	Power fail with no skip setup	Program counter	Power low flag came up but a user defined routine to save appropriate registers not in core.
35	Illegal drum size	CAL address of first .INIT	Drum size in .SCOM + 4 (bits 15 to 17) is not 1 through 5.
40	Header label errors	CAL address	The internal header label for the currently opened file is incorrect.
42	Accessibility map overflow	CAL address	Too many files recorded on the current MAGtape. Copy the tape to retrieve storage occupied by unwanted files.
43	Directory recording error	CAL address	Directory cannot be recorded at the beginning of the tape - use the utility program to re-make the tape.

APPENDIX F

SYSTEM PROGRAM DISK AND DECTAPE ADDRESSES

Program	Logical Block (8)**	Disk Track, Sector	Number of Logical Blocks (8)
KM-9	0	0,0	36
SGEN2	36	1,40	6
KM-9 SKIP BLOCK	44	1,64	1
UPDATE	45	1,68	10
KM-9 I/O BLOCK	55	2,20	1
SYSLD	56	2,24	11
EXECUTE	67	2,60	2
FILE BIT MAPS	71	2,68	7
DIRECTORY	100	3,16	1
†QAREA	101	3,20	40-200*
PATCH	652	21,24	5
EDIT	657	21,44	12
PIP	671	22,04	20
F4	711	22,68	31
MACRO	742	24,08	27
F4A	771	25,20	27
MACROA	1020	26,32	23
DUMP	1044	27,32	4
SGEN1	1050	27,48	14
CONV	1064	28,16	14

*Depends upon installation core size (8-32K).
**Applies to both DECtape and Disk.

APPENDIX G

SUMMARY OF KEYBOARD COMMANDS FOR KEYBOARD AND BACKGROUND/FOREGROUND MONITORS

KEYBOARD MONITOR COMMANDS

System Program Load Commands

Command	System Program Loaded
F4	FORTRAN IV Compiler
F4A	Abbreviated FORTRAN IV Compiler
MACRO	MACRO-9 Assembler
MACROA	Abbreviated MACRO-9 Assembler
PIP	Peripheral Interchange Program
EDIT	Symbolic Text Editor
CONV	7-to-9 Converter
LOAD	Linking Loader
GLOAD	Linking Loader (set to load and go)
DDT	Dynamic Debugging Technique program
DDTNS	DDT program with no user symbol table
UPDATE	Library File Update program
DUMP	Program to dump saved area (see CTRL Q and QDUMP commands)
PATCH	System tape Patch program
CHAIN	Modified version of Linking Loader -- allows for chaining
EXECUTE (E)	Control program to load and execute chained programs
SGEN	System Generation program

Special Function Commands

Command	Action
LOG (or L)	Can be followed by any comment and terminated by ALT MODE.
SCOM (or S)	Causes typeout of system configuration information, including available device handlers.
API OFF	Disables API.
API ON	Enables API.
QDUMP (or Q)	Conditions Monitor to dump memory on the "save area" of the system tape (or other system device medium if available) in the event of an unrecoverable IOPS error.

Command	Action
HALT (or H)	Conditions the Monitor to halt in the event of an unrecoverable IOPS error.
INSTRUCT (or I)	Types list of Monitor commands.
INSTRUCT (or I) ERRORS	Types system error messages.
REQUEST (or R)	Types .DAT slot assignments and use: a. For system program when followed by system program name. Example: R DDT b. For all positive .DAT slots when followed by USER. Example: R USER c. For all .DAT slots when followed by carriage return. Example: R ↵
ASSIGN (or A)	Allows reassignment of .DAT slots to devices other than those set at system generation time. Example: A PRA -10,3/PPA -6,4
DIRECT (or D) n	Lists the directory of DECtape mounted on unit n (0-7).
NEWDIR (or N) n	Writes empty directory on DECtape on unit n (units 1-7 only).
GET (or G) n	Restores core image from DECtape (or other system device medium if available) on unit n (0-7).
GET (or G) n address	Restores core image from DECtape (or other system device medium if available) on unit n and restarts at specified address.
GET (or G) n HALT (or H)	Restores core image from DECtape (or other system device medium if available) on unit n and halts.
CHANNEL (or C) 7/9	This command establishes whether the default condition for magnetic tape operation is to be 7-channel or 9-channel.
339 (or 3) ON/OFF	This command informs the Monitor whether or not a 339 handler is to be loaded.
VC38 (or V) ON/OFF	This command establishes whether a character display table for the VC38 option should be set up prior to loading any system or user program.

Control Character Commands

Command	Echoes	Action
CTRL S	↑ S	Starts user program after loading by linking loader.
CTRL C	↑ C	Returns to Monitor; may be used at anytime -- resets all .DAT slot assignments.
CTRL T	↑ T	a. Returns control to DDT if DDT is being used. b. Skips to next job when in Batch mode.

Command	Echoes	Action
CTRL R	↑R	Allows program to continue after IOPS 4 message.
CTRL P	↑P	a. Reinitializes or restarts system program. b. Returns to location specified in user program's last .INIT referencing the Teletype.
CTRL Q n	↑Q	Saves core image on save area on DECtape (or other system device medium if available) mounted on unit n (may be system device) and returns to Monitor.
CTRL U	@	Cancels current line on Teletype (input or output).
RUBOUT	\	Cancels last character input from Teletype (not applicable with DDT).

Batch Processor Commands

Command	Function
BATCH (B) dv	Enter Batch mode with dv as batch device; dv can be typed as PR, for paper tape reader, or CD, for card reader
$JOB	Used to separate jobs.
$DATA	Beginning of data -- all inputs up to $END are not echoed on the Teletype.
$END	End of data.
$EXIT	Leave Batch mode.

NOTE

The following commands are illegal when operating in Batch mode: QDUMP, HALT, GET (all forms), BATCH, LOAD, DDT, and DDTNS.

Special Batch Processor control characters include the following:

CTRL T (echoes ↑T) Skip to next job.

CTRL C (echoes ↑C) Leave Batch mode.

Command	Function
FILES xxn	Allows conservation of core space by indicating number of files open simultaneously. Normally typed prior to loading user programs so loader can allocate sufficient buffer space. xx = bulk storage device; n = number of files . Example: $FILES DT3
FCORE n	Used to define core requirements of FOREGROUND job prior to loading. n = octal number that represents number of 1K increments required. Example: $FCORE 3
FCONTROL n	Allows FOREGROUND Teletype assignment to be changed (while operating in FOREGROUND mode only). n = number of external Teletype to be assigned to FOREGROUND. Example: $FCONTROL 2
BCONTROL n	Allows BACKGROUND Teletype assignment to be changed (while operating in BACKGROUND mode only). The Teletype to be assigned must not currently be assigned to FOREGROUND. n = number of external Teletype to be assigned. Example: $BCONTROL 3

INDEX

HOW TO OBTAIN REVISIONS AND CORRECTIONS

Notification of changes and revisions to currently available Digital software and of new software manuals is available from the DEC Program Library for the PDP-5, 8, 8/S, 8/I, 8/L, LINC-8, the PDP-4, 7, and 9 is currently published in DECUSCOPE, the magazine of the Digital Equipment Computer User's Society (DECUS). This information appears in a section of DECUSCOPE called "Digital Small Computer News".

Revised software products and documents are shipped only after the Program Library receives a specific request from a user.

DECUSCOPE is distributed periodically to both DECUS members and to non-members who request it. If you are not now receiving this information, you are urged to return the request form below so that your name will be placed on the mailing list.

To: Decus Office,
Digital Equipment Corporation,
Maynard, Mass. 01754

☐ Please send DECUS installation membership information.

☐ Please send DECUS individual membership information.

☐ Please add my name to the DECUSCOPE non-member mailing list.

Name _____

Company _____

Address _____

_____
(Zip Code)

PDP-9 ADVANCED SOFTWARE
SYSTEM MONITORS
DEC-9A-MAD0-D

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

DIGITAL EQUIPMENT CORPORATION ▫ MAYNARD, MASSACHUSETTS