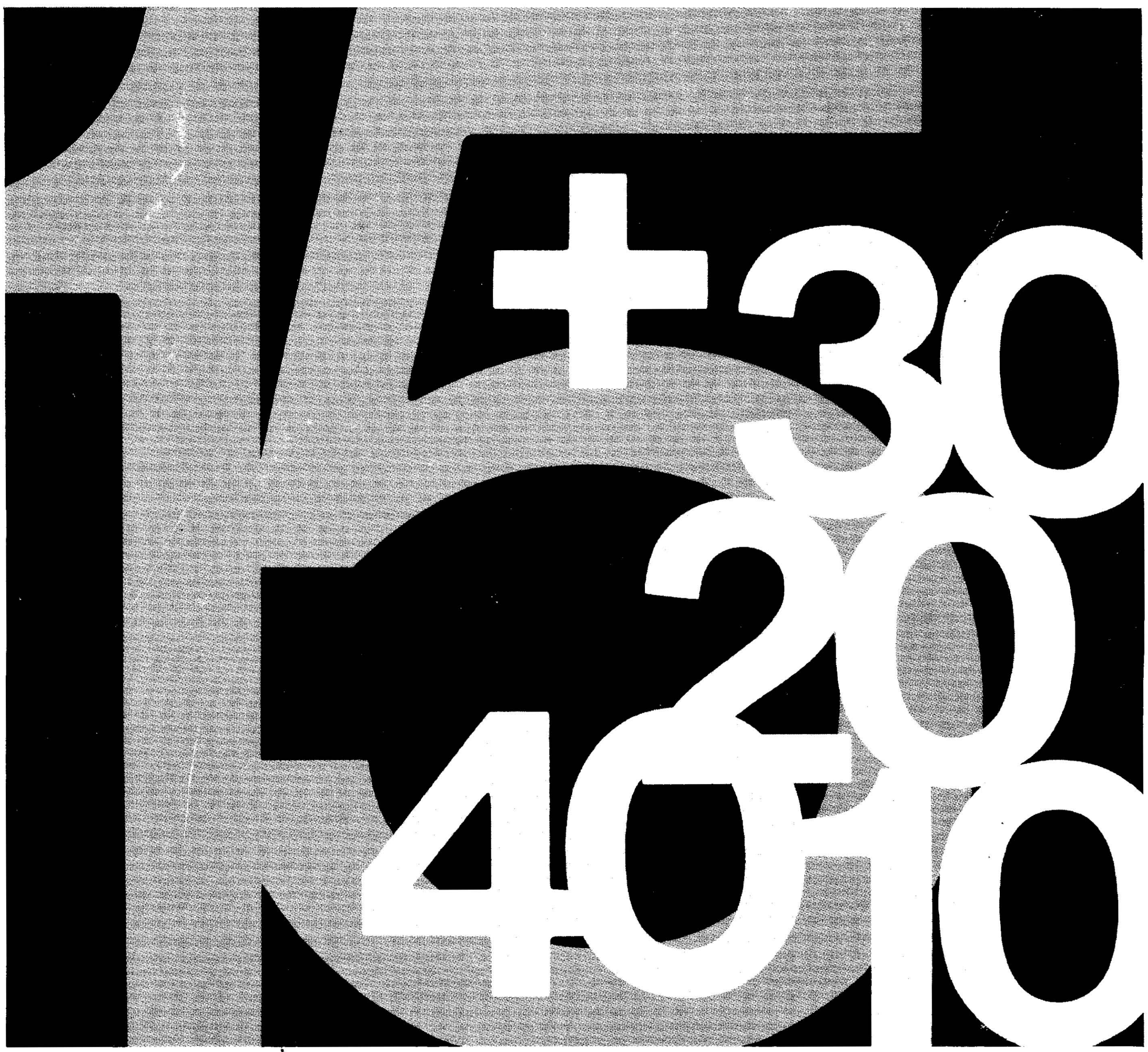
Digital Equipment Corporation
Maynard, Massachusetts

digital

Advanced Monitor Software System for PDP-15/20/30/40



PDP-15
ADVANCED MONITOR SOFTWARE SYSTEM FOR PDP-15/20/30/40 PROGRAMMER'S REFERENCE MANUAL

To obtain additional copies of this manual, order number DEC-15-MR2B -D from the Program Library, Digital Equipment Corporation, Maynard, Massachusetts 01754 Price $5.00

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing January 1970
Revised December 1970

The material in this manual is for information purposes and is subject to change without notice.

ADVANCED MONITOR SOFTWARE SYSTEM

The ADVANCED Monitor software system described in this manual may be obtained in either of two versions: 1) a standard PAGE mode system or 2) an optional BANK mode system.

Page Mode System

The ADVANCED Monitor Page mode system loads and relocates user programs in 4K pages and permits address modification via the index register. The Page mode system is supplied as standard software with the PDP-15/20/30 and /40 systems.

> NOTE
>
> With the exception of Appendix G, or where otherwise referenced, the information presented in this manual concerns only the Page mode ADVANCED Monitor software system.

Bank Mode System

The optional Bank mode system permits direct addressing within 8K banks, but does not permit the use of the index register for address modification. This system is useful to the PDP-15 user who prefers direct addressing up to 8K, or who wishes to take advantage of the extensive library of PDP-9 programs now available from the Digital Equipment Computer Users Society (DECUS).

The Bank mode system is also available to PDP-9 users who have equipment configurations comparable to those of the basic PDP-15/20. This enables the PDP-9 user to avail himself of the many improvements introduced into the PDP-15 ADVANCED Monitor software system.

The differences between the Page and Bank mode systems and procedures for the installation of the Bank mode system are given in Appendix G.

OPTIONAL KEYBOARD MONITOR

The ADVANCED Monitor software system contains a sophisticated interactive keyboard/program monitor which is available in either a standard or a "special" version. The special version (designated KMS15) permits overprinting on the console teleprinter, the standard version does not.

Overprinting, a feature useful to FORTRAN programming, requires a slightly larger monitor (+24_{10} words). If the user desires the KMS15 version, it must be installed into the standard system. Appendix G contains a brief description of the KMS15 monitor and instructions for its installation.

HOW ADVANCED SOFTWARE SYSTEMS ARE SUPPLIED

The ADVANCED Monitor software is supplied to the user (Page or Bank mode) in the form of two DECtapes:

1) a standard system DECtape (Page or Bank mode) which contains all programs considered as standard to the system. The separate Page and Bank mode system DECtapes are identified as follows:

   a) Page Mode DEC-15-SRZB-UC
   b) Bank Mode DEC-15-SWZA-UC

2) a peripheral DECtape common to both Page and Bank mode systems which contains device handlers and routines for Digital-offered optional peripherals and special programs (e.g., KMS 15). The peripheral DECtape is identified as DEC-15-SZZB-UC.

OVERALL PDP-15 DOCUMENTATION STRUCTURE

A tree-type block diagram of the overall "PDP-15 Family of Manuals" is illustrated on the following page. A brief description of the contents and the order number of each manual shown in the diagram are also presented.

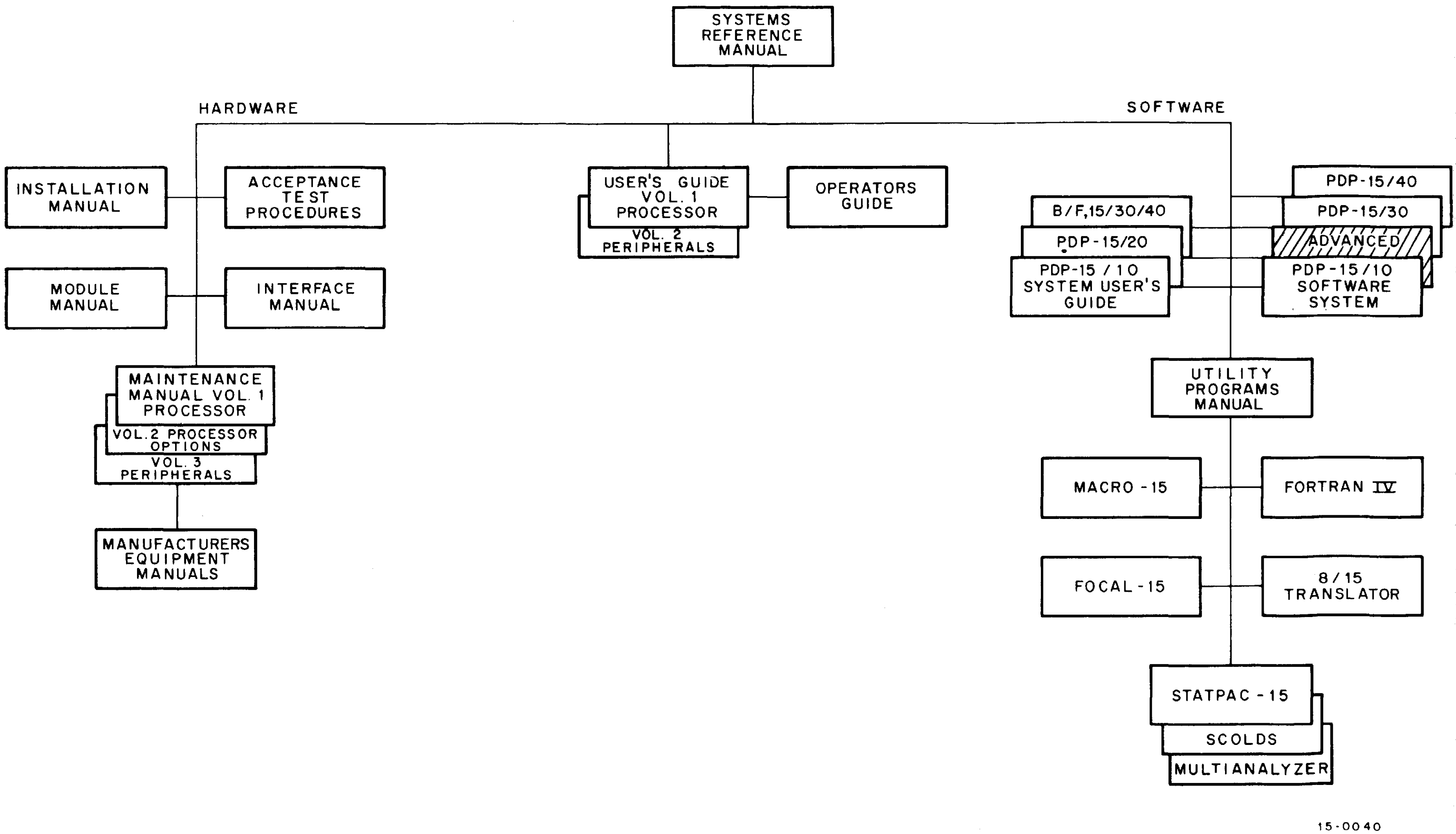
ORGANIZATION OF PDP-15 SOFTWARE MANUALS

There are two basic categories of PDP-15 software manuals:

a. Unique, single-system manuals which contain information concerning only one of the four available PDP-15 systems. This category consists of detailed software system descriptive manuals, each with an associated operational command summary. An example of this class of manual would be the "PDP-15/20 Software System" manual and its associated "PDP-15/20 User's Guide".

b. Common, multi-system manuals that describe utility, language, application and other PDP-15 programs which may be employed in one or more of the four available PDP-15 systems. Some examples of this type of manual are the PDP-15 "Utility", "MACRO-15 Assembler", and "STATPAC" manuals.

PDP-15 FAMILY OF MANUALS



SYSTEM REFERENCE MANUAL - Overview of PDP-15 hardware and software systems and options; instruction repertoire, expansion features and descriptions of system peripherals. (DEC-15-GRAZ-D)

USER'S GUIDE VOLUME 1, PROCESSOR - Principal guide to system hardware includes system and subsystem features, functional descriptions, machine-language programming considerations, instruction repertoire and system expansion data. (DEC-15-H2DA-D)

VOLUME 2, PERIPHERALS - Features functional descriptions and programming considerations for peripheral devices. (DEC-15-H2DA-D)

OPERATOR'S GUIDE - Procedural data, including operator maintenance, for using the operator's console and the peripheral devices associated with PDP-15 Systems. (DEC-15-H2CA-D)

PDP-15/10 SYSTEM USER'S GUIDE - COMPACT and BASIC I/O Monitor operating procedures. (DEC-15-GG1A-D)

PDP-15/20 SYSTEM USER'S GUIDE - ADVANCED Monitor system operating procedures. (DEC-15-MG2B-D)

PDP-15/20/30/40 ADVANCED MONITOR SOFTWARE SYSTEM - ADVANCED Monitor System descriptions; programs include system monitor and language, utility, and application types; operation, core organization, and input/output operations within the monitor environment are discussed. (DEC-15-MR2B-D)

PDP-15/30 and 15/40 BACKGROUND/ FOREGROUND MONITOR SOFTWARE SYSTEM - Background/Foreground Monitor description, including the associated language, utility, and application programs. (DEC-15-MR3A-D)

MAINTENANCE MANUAL VOLUME 1, PROCESSOR - Block diagram and functional theory of operation of the processor logic. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

VOLUME 2, PROCESSOR OPTIONS - Block diagram and functional theory of operation of the processor options. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

VOLUME 3, PERIPHERALS (Set of Manuals - Block diagram and functional theory of operation of the peripheral devices. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

INSTALLATION MANUAL - Power specifications, environmental considerations, cabling, and other information pertinent to installing PDP-15 Systems. (DEC-15-H2AA-D)

ACCEPTANCE TEST PROCEDURES - Step-by-step procedures designed to ensure optimum PDP-15 Systems operation.

MODULE MANUAL - Characteristics, specifications, timing, and functional descriptions of modules used in PDP-15 Systems. (DEC-15-H2EA-D)

INTERFACE MANUAL - Information for interfacing devices to a PDP-15 System. (DEC-15-H0AA-D)

UTILITY PROGRAMS MANUAL - Utility programs common to PDP-15 Monitor Systems. (DEC-15-YWZA-D)

MACRO-15 - MACRO assembly language for the PDP-15. (DEC-15-AMZA-D)

FORTRAN IV - PDP-15 version of the FORTRAN IV compiler language. (DEC-15-KFZB-D)

FOCAL-15 - An algebraic interactive compiler-level language developed by Digital Equipment Corporation. (DEC-15-KJZB-D)

CONTENTS

Page

CHAPTER 1
ADVANCED SOFTWARE SYSTEM

CHAPTER 2
THE ADVANCED MONITOR ENVIRONMENT

CHAPTER 3
SYSTEM MACROS

CHAPTER 4
ADVANCED MONITOR

CHAPTER 5

I/O DEVICE HANDLERS

FIGURES

TABLES

CHAPTER 1

ADVANCED SOFTWARE SYSTEM

1.1 INTRODUCTION

The ADVANCED Software System is a complete integrated system of programs for the preparation, compilation, assembly, debugging, and operation of user programs. A diagram illustrating the structure of the ADVANCED software system is shown in Figure 1-1. As shown, this software system includes:

a. Compiler and assembly language programs
b. A large group of programming and operational aid (Utility) programs
c. A versatile and flexible Input/Output Programming System (IOPS)
d. A sophisticated interactive keyboard/program monitor which permits device-independent programming and automatic creation, calling, and loading of programs.

Upwards-compatibility exists between all PDP-15 Monitor Systems (e.g., programs prepared for the Basic I/O Monitor may be run in the ADVANCED system environment).

1.2 HARDWARE REQUIREMENTS

The minimum equipment configuration for the employment of the ADVANCED Software System is that of the Basic 15/20 system as illustrated in Figure 1-2.

1.3 MONITOR SYSTEMS

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

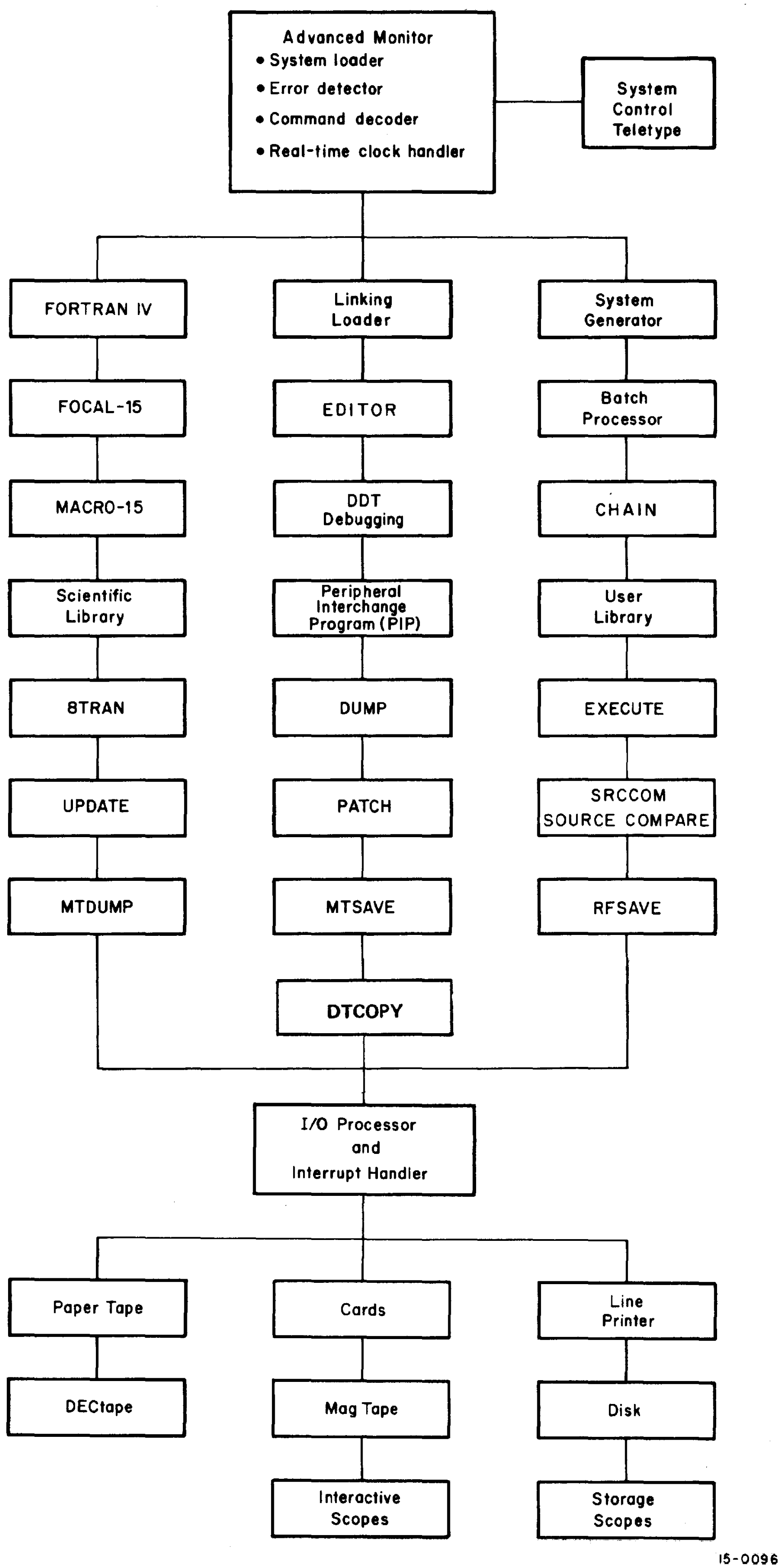


Figure 1-1 ADVANCED Monitor

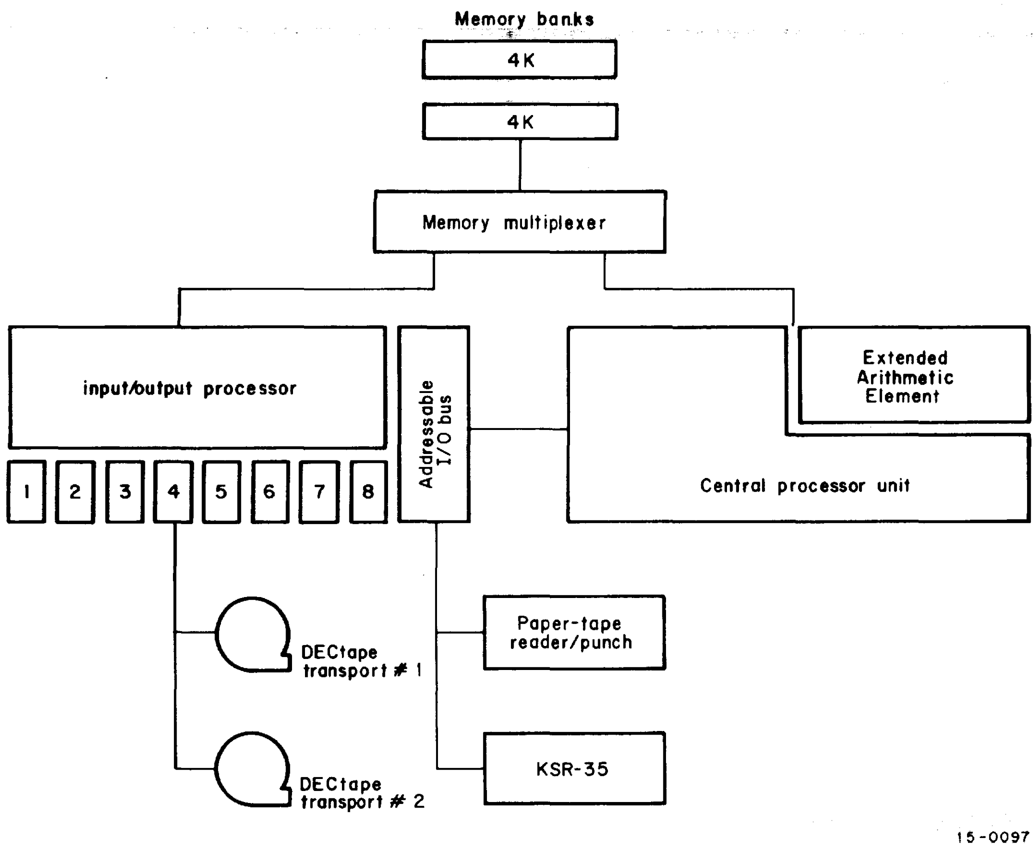


Figure 1-2 PDP-15/20 System Equipment Configuration (Basic)

1.3.1 'ADVANCED Monitor

The ADVANCED Monitor Software System includes all of the facilities of the BASIC I/O Monitor (Paper Tape) plus routines to accept and act upon teleprinter 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, this 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) existing hardware need not result in program reassembly or recompilation.

The 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 mass storage device with a few simple keyboard commands. The Monitor also has a batch processing capability that allows user commands to originate from the paper tape reader or card reader instead of from the teleprinter, thus permitting many programs to be run without operator intervention.

1.4 INPUT/OUTPUT PROGRAMMING SYSTEM (IOPS)

The Input/Output Programming System (IOPS) consists of an I/O control routine (CAL handler) and individual hardware device handling routines (device handlers) that process file and data level commands to the device. These handlers exist for all standard peripherals (see Section 5.4).

The CAL handler 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 opening and closing files, and for performing 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. All device handlers operate either with or without the Automatic Priority Interrupt (API) option.

1.5 SYSTEM PROGRAMS

In addition to IOPS and the ADVANCED Monitor, the ADVANCED Software System contains the following language and utility programs:

FOCAL - Algebraic Language Interpreter (DEC-15-KJZB-D)

FORTRAN IV - Compiler, Object Time System, and Science Library (DEC-15-KFZB-D)

MACRO-15 - PDP-15 Assembler (DEC-15-AMZB-D)

DDT - Dynamic Debugging Technique

EDIT - Text Editor

EDITVP - Text Editor for VP15A Storage Tube Display

PIP - Peripheral Interchange Program

Linking Loader - Loads Relocatable Programs

CHAIN - Program to Construct System of Core Overlays (DEC-15-YWZA-D)

EXECUT - Program to Supervise Execution of CHAIN Built Overlay System

SGEN - System Generator

DUMP - Dump Program

PATCH - System Patch Program

SRCCOM - Source Compare Program

DTCOPY - High-Speed 8K DECtape Copy Program

8TRAN - PDP8 to PDP-15 Translator (DEC-15-ENZA-D)

The following special purpose utility programs are also available:

RFSAVE - DECdisk/DECtape Save MTSAVE - DECdisk/Magtape Save MTDUMP - Magnetic Tape Dump	(DEC-15-YWZA-D)

1.5.1 FOCAL Programs

FOCAL (Formulating On-line Calculations in Algebraic Language) operates in on-line conversational mode, using natural language and arithmetic terms to establish a simplified environment for the computer-aided solution of business and scientific arithmetic problems. Included in FOCAL are such features as:

1. Device independence;
2. Linkage to assembly language (MACRO) routines to establish a user library of commonly used functions;
3. Use of COMMON to facilitate chaining in the same manner as FORTRAN IV.

1.5.2 FORTRAN IV Compiler

The PDP-15 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 provided in the Science Library.
c. Blank Common is treated as named Common except when the 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 program interrupt or API facilities enabled. It generates programs that operate with the Program Interrupt (PI) or Automatic Priority Interrupt (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-15 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 three versions of the FORTRAN IV compiler: (1) F4, the basic compiler; (2) F4I, a compiler which permits DECtape I/O in an 8K system; and (3) F4S, a more powerful version of F4 which has fewer restrictions and an expanded diagnostic capability.

1.5.3 MACRO Assembler

The MACRO Assembler provides users with highly sophisticated macro generating and calling facilities within the context of a symbolic assembler.

Some of the prominent features of MACRO include:

a. The ability to:
   (1) define macros
   (2) define macros within macros (nesting)
   (3) redefine macros (in or out of macro definitions)
   (4) call macros within macro definitions
   (5) have macros call themselves (recursion)
   (6) combine three input files for one assembly
b. Conditional assembly based on the computational results of symbols or expressions
c. Repeat functions
d. Boolean manipulation
e. Optional octal, symbolic, and cross-reference 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 (MACROI) is available for users with 8K systems which permits DECtape input and output.

1.5.4 Dynamic Debugging Technique (DDT) Program

DDT provides on-line debugging facilities within the 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 breakpoint feature allows the insertion and simultaneous use of up to four breakpoints, any 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, 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.

1.5.5 Text Editor Programs, EDIT and EDITVP

The Text Editor of the 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.

The program EDITVP is similar to EDIT except that it permits the text to be displayed on the VP15A storage tube.

1.5.6 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; transfer files; or copy the entire contents of mass storage units.

1.5.7 Linking Loader

The Linking Loader loads any FORTRAN IV or MACRO 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.

1.5.8 8 to 15 Translator (8TRAN)

This program is used as an aid in translating programs written for the Digital PDP-8 computer into MACRO-15 form. The translator does not necessarily produce an executable program, but translates a major portion of the PDP-8 code into equivalent MACRO-15 code and indicates those areas of the 8 program which must be reviewed and processed by the programmer.

1.5.9 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.

1.5.10 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. It also provides the ability to dump areas of mass storage (e.g., a DECtape block) onto any listing device.

1.5.11 Library Update Program

This system program gives the user the capability to examine and update the binary library files on mass storage devices.

1.5.12 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, to examine and change any word in any DECtape or DECdisk block, or to convert relocatable binary programs into system programs (SYS files).

1.5.13 CHAIN and EXECUTE Programs

The programs CHAIN and EXECUTE provide the user with the ability to construct and run a system of core overlays in the ADVANCED Monitor environment.

1.5.14 Source Compare Program (SRCCOM)

The SRCCOM program compares any two symbolic programs (IOPS ASCII) and indicates their differences. This program is useful for program identification and/or verification, proofing an edited program, comparison of old and new versions of the same program, etc.

1.5.15 DECtape Copy (DTCOPY)

This program, designed for 8K system users, permits high speed copying of DECtape onto DECdisk units.

CHAPTER 2

THE ADVANCED MONITOR ENVIRONMENT

2.1 MONITOR FUNCTIONS

The ADVANCED Monitor simplifies the task of programming I/O functions by providing an interface between system or user programs and the external world of I/O devices. The Monitor, by means of IOPS and Program Interrupt (PI) or optional Automatic Program Interrupt (API), permits simultaneous operation of multiple I/O devices along with overlapping computations.

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[1] within the Monitor as a legitimate I/O call. 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

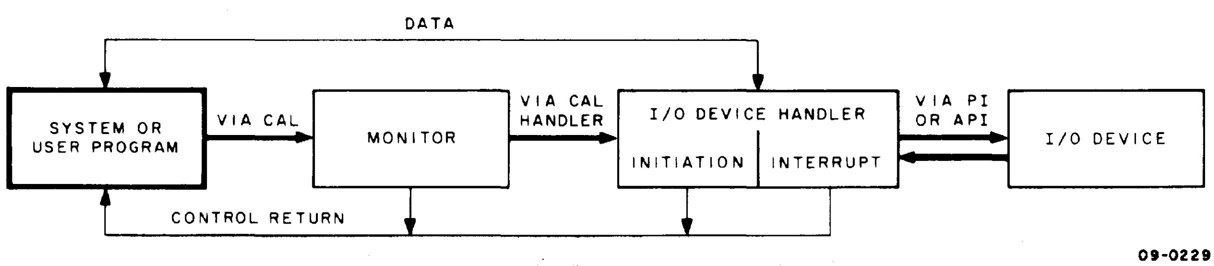


Figure 2-1 General I/O Communication in Monitor Environment

[1]Refer to the PDP-15 Users' Handbook Vol. 1, (DEC-15-H2DA-D) for a description of the CAL handler.

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. Computation 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.

2.1.2 Command, Control, and Data Flow

Figure 2-2 illustrates the data flow and general organization of the ADVANCED Monitor. As shown, the user can initiate a command via the teleprinter.

In the ADVANCED Monitor environment, an expanded set of keyboard commands can be interpreted by a Keyboard Listener (.KLIST) and acted upon by a Monitor Command Decoder (.MCD). This feature greatly extends the capabilities of the Monitor and provides the user with a large repertoire of 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.

ADVANCED Monitor I/O commands (system macros[1]) 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-15-KFZA-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 source programs, Monitor I/O commands are written as system 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.

---

[1]System Macros are predefined system commands which are equivalent to a specific sequence of machine instructions. Refer to Chapter 3 for a description of the ADVANCED system macros.

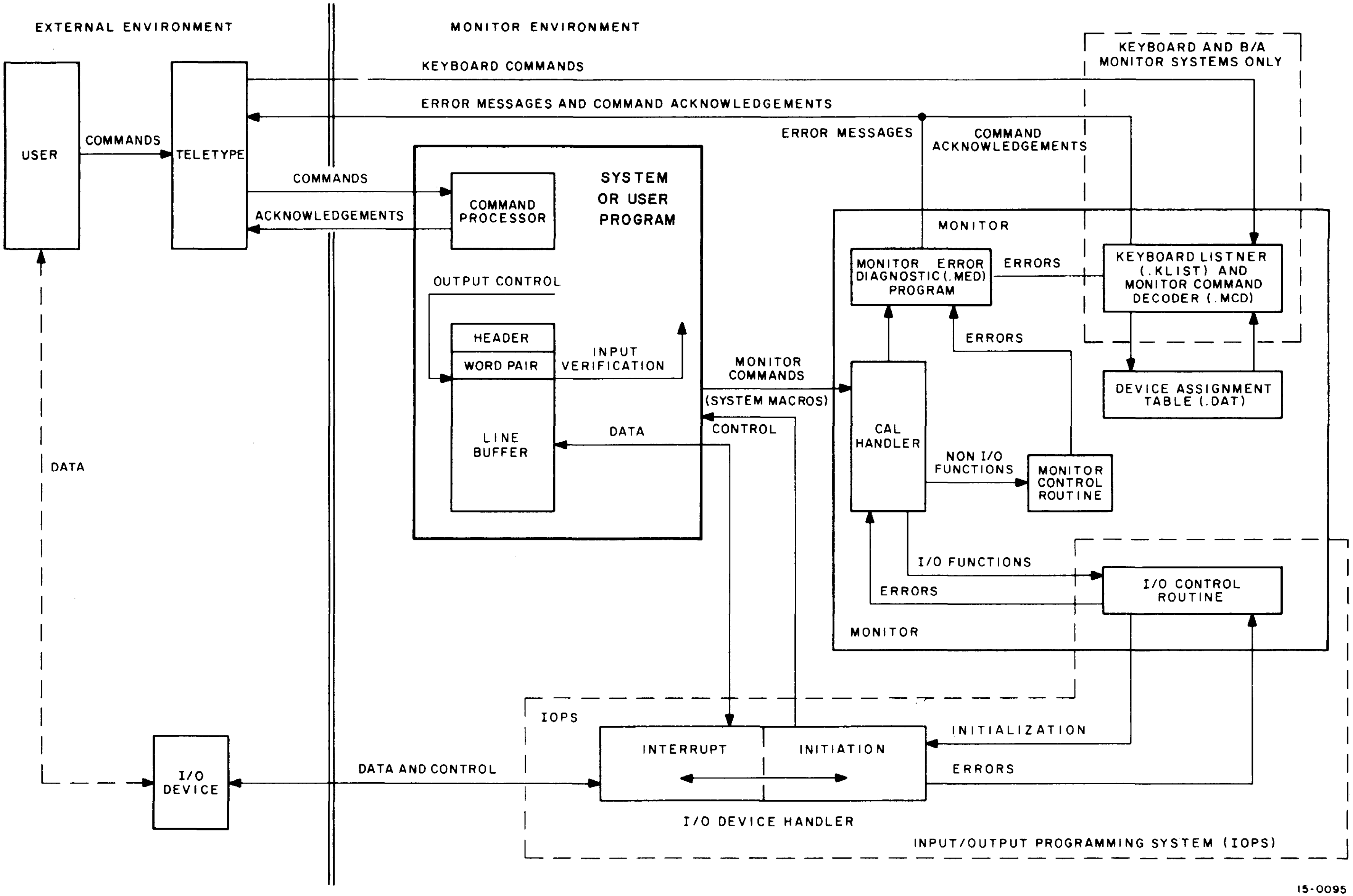


Figure 2-2 Command, Control, and Data Flow in Monitor Environment

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 ADVANCED Monitor environment, device associations can be permanently modified at System Generation time, or dynamically modified 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
	11	.WRITE
	12	.WAIT and .WAITR
	13	.TRAN
Functions processed by Monitor Control Routine	14	.TIMER
	15	.EXIT
	16	.SETUP

Figure 2-3 ADVANCED Monitor Commands and Function Codes

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 (or .WAITR) 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 or to the address specified in the .WAITR.

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 preceding 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 paragraph 2.3.3) or when the .TRAN command is used (see paragraph 3.1.15). 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.

Word 0	First Word of Line Buffer Header
Word 1	Second Word of Line Buffer Header
Word 2	First Word of Data Area
↓	↓
Word n-1	Last Word of Data Area

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 simultaneously. 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 and should be studied carefully. The .BLOCK pseudo operation may be used to reserve space for a line buffer. A tag is required to allow referencing by individual .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 it has 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 in which 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.

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-files.

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.

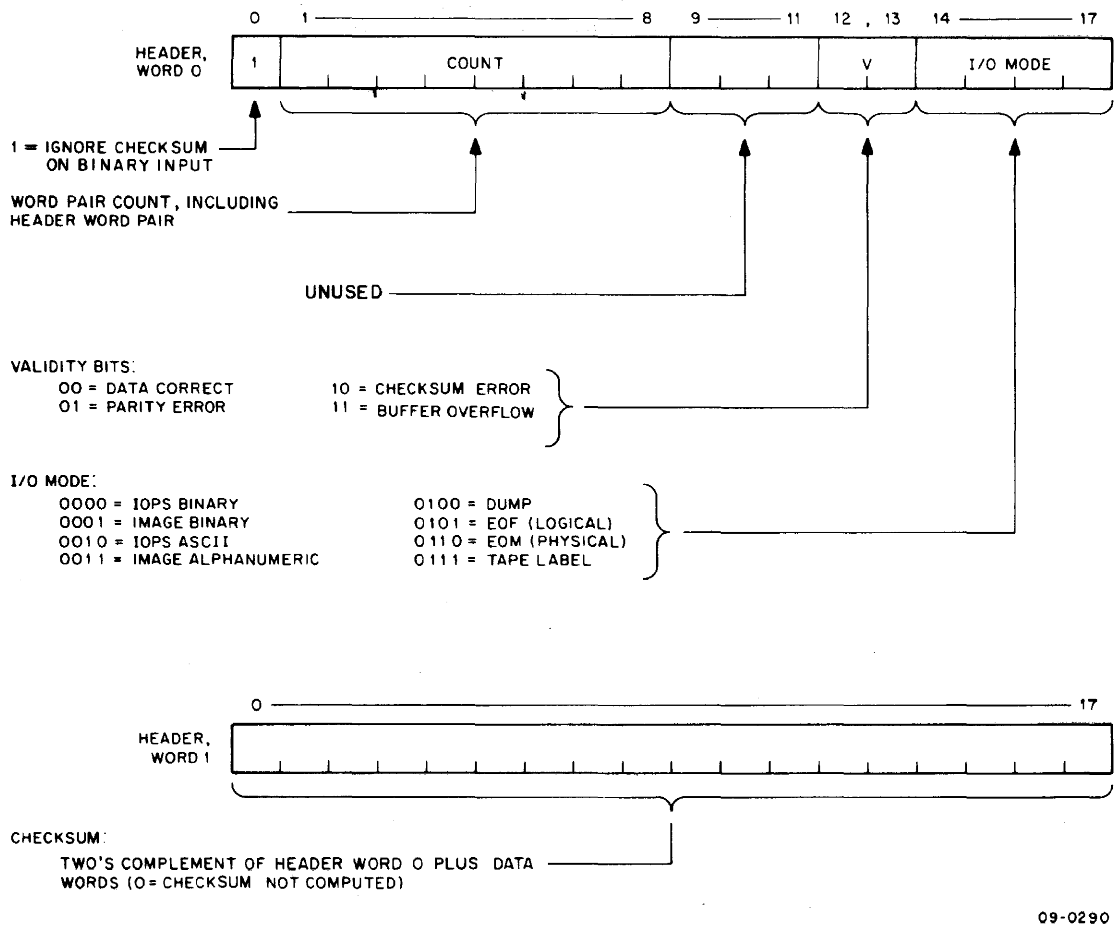


Figure 2-5 Format of Header Word Pair

2.3 DATA MODES

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

Mode	Code[1]	
IOPS Binary	0	
Image Binary	1	
IOPS ASCII	2	
Image Alphanumeric	3	
Dump	4	
9-Channel Dump	5	(Magtape only; see sections 5.3.10, 5.4.6, and 5.4.6.3 (f).)

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

Table 2-1

Maximum Line Buffer Sizes

Device	Maximum Line Buffer Size	Data[1] Modes	Modes
PR (paper tape reader)	52_{10}	All	34_{10} sufficient if mode 2 only. Headers accepted for mode 0; headers generated for modes 1, 2, 3.
PP (paper tape punch)	52_{10}	All	34_{10} sufficient if mode 2 only. Headers output for mode 0 only.
TT (teleprinter)	34_{10}	2,3 only	Allows for 80_{10} characters. Headers generated on input. Headers not generated on output.
CD (card reader)	36_{10}	2 only	Headers generated for mode 2.
LP (line printer)	52_{10}	2 only	Allows for 125_{10} characters. No headers output.
VP (Display)	34_{10}	2,3 only	Allows for 80_{10} characters. Mode 3 requires 80_{10} word buffer. No Header output.
DT (DECtape)	255_{10}	All	IOPS and image modes allow for several line buffers (logical records) per physical block.
MT (magnetic tape)	255_{10}	All	
DK (DECdisk)	255_{10}	All	

[1]See Paragraph 2.3 above.

2.3.1 IOPS Modes

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

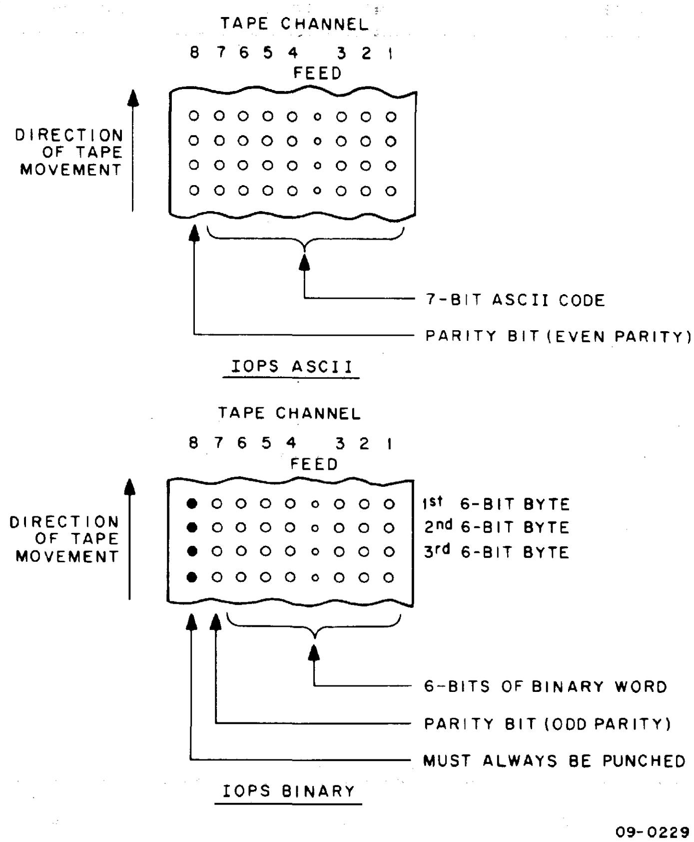


Figure 2-6 IOPS Mode Data on Paper Tape

2.3.1.1 IOPS ASCII - Seven-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 ordinarily input to or output from IOPS via the teleprinter 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.

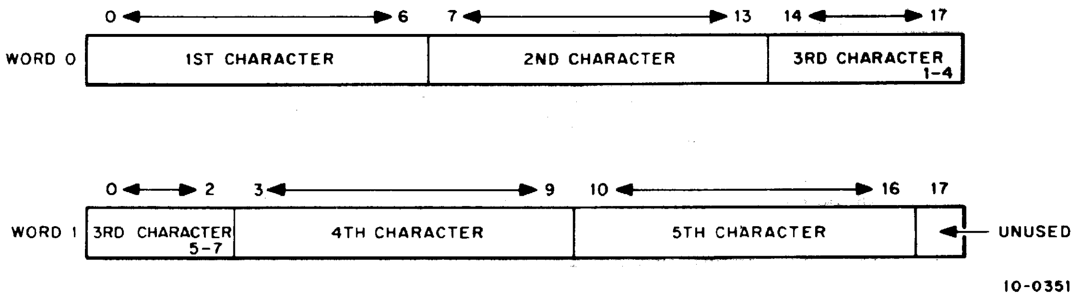


Figure 2-7 5/7 ASCII Packing Scheme

Non-parity IOPS ASCII occurs in data originating at a Model 33, 35, or 37 Teletype[1], 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 optional 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 5.4 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.

---

[1]Teletype is a registered trademark of the Teletype Corporation.

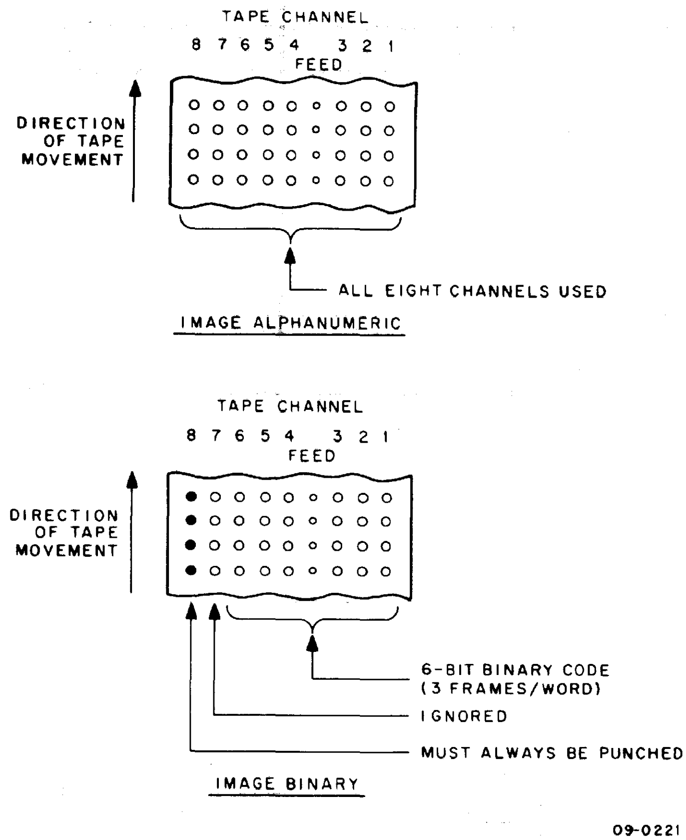


Figure 2-8, Image Mode Data On Paper Tape

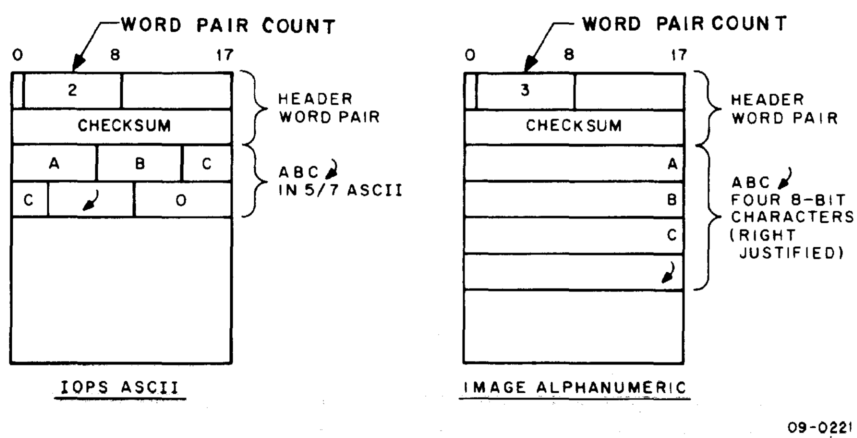


Figure 2-9, IOPS ASCII And Image Alphanumeric Data In Line Buffers And On Mass Storage Devices.

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.

Table 2-2

Input/Output Data Mode Terminators

DATA MODE	INPUT	OUTPUT
IOPS ASCII	Carriage RETURN ALT MODE Word Pair Count[1] EOM Word Count[2] EOF[1]	Carriage RETURN ALT MODE Word Pair Count[3]
IOPS Binary	Word Pair Count EOM Word Count[2] EOF[1]	Word Pair Count
Image Alpha-numeric Image Binary	Word Count EOM EOF[1]	Word Pair Count
Dump[4]	Word Count EOM EOF[1]	Word Count

[1]Bulk storage only.

[2]If word count is exceeded before a terminator is encountered, IOPS sets bits 12 and 13 of Header Word 0 to 3 (Buffer Overflow).

[3]If the Word Pair Count is 1 or less, the line is ignored; if greater than 1, ignore the count and accept Carriage RETURN or ALT MODE (non-file-oriented devices only). Bulk storage devices require a Word Pair Count greater than 1 and less than 177_8, otherwise an IOPS 27 error will occur.

[4]9-Channel Dump data mode is available for magnetic tape; refer to section 5.4.6.3(f) for a description of this data mode.

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 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, the Monitor 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.

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.

2.5 SPECIFYING DEVICES USED TO LINKING LOADER.

When writing a MACRO 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 logical device numbers or .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. The .IODEV pseudo-op has the following form:

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

where the MACRO 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.

FORTRAN IV programs cause the compiler to generate the appropriate Linking Loader code based on the units specified in READ and WRITE statements. Note that use of a constant to specify an I/O unit in a FORTRAN program will cause only the handler assigned to the corresponding .DAT slot to be loaded; whereas if a variable is used, handlers will be loaded for all positive .DAT slots that have handlers assigned.

Table 2-3

System Communication Table (.SCOM) Entries

Word	Purpose
.SCOM	First free register below resident portion of System Bootstrap.
.SCOM+1	First free register above resident monitor (constant)
.SCOM+2	Lowest free register available to user or system program
.SCOM+3	Highest free register available to user or system program
.SCOM+4	Hardware options available: Bit 0 1 = API Bit 1 1 = EAE Bit 2 1 = TTY = 35/37 (0 = KSR33) Bit 3 1 = Non-resident monitor in core Bit 4 Reserved Bit 5 Reserved Bit 6 1 = 9-channel, 0 = 7-channel Magnetic Tape Bit 7 1 = Page Mode Operation, 0 = Bank Mode Operation Bit 8 1 = No ↑Q Area Bit 9 Reserved Bit 10 Reserved Bit 11 1 = Bank Mode Operation Bits 12 and 13 Line Printer Column Size 00 = Unknown 01 = 80 10 = 120 11 = 132 Bit 14 1 = Background/Foreground System Bit 15-17 Reserved
.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. Also used for file name with EXECUTE.
.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.
.SCOM+20	Extra 4K System Information Bit 0 1 = Extra 4K on System Bit 3-17 First free register in extra 4K
.SCOM+21	Magtape Status Register
.SCOM+22	(Reserved for Magtape Handler)
.SCOM+23	Address of Device Assignment Table (.DAT)

2.6 RESERVED WORD LOCATIONS

Word locations ∅ through 77 are dedicated systems locations and cannot be employed by the user. The contents of these locations are described in Table 2-4.

Table 2-4

Reserved Address Locations

ADDRESS	USE: Page Mode	USE: Bank Mode
∅	Stores the contents of the extended PC, link, extend mode status, and memory protect status during a program interrupt.	
1	JMP to Skip Chain	EEM (Enter Extend Mode) instruction for compatibility with PDP-9 systems.
2	Stores system tape (Bank or Page) indicator during Teletype interrupts.	JMP to Skip Chain
3	.MED, entry to Monitor Error Diagnostic routine	
4	JMP to error handler	(Same)
5	(Not used in Page Mode.)	Stores system type (Bank or Page) indicator during Teletype interrupts.
6	Used for API ON/OFF indicator in both systems.	
7	Stores real time clock count.	(Same)
10 - 17	Autoindex registers	(Same)
20	Stores the contents of the extended PC, link, extend mode status, and memory protect status during a program interrupt.	
21	JMP to CAL handler	(Same)
22 - 37	Seven pairs of word counter-current address registers for use with 3-cycle I/O device data channels.	
40 - 77	Store unique entry instructions for each of 32_{10} automatic priority interrupt channels.	

CHAPTER 3

SYSTEM MACROS

3.1 INTRODUCTION

The MACRO-15 assembler permits the development of instructions called "macros" which, when used as a source statement, can cause a specific sequence of instructions to be generated in the object program. For example, consider the following sequence:

```
→|LAC  →|A
→|TAD  →|B
→|DAC  →|C
   .
   .
   .
→|LAC  →|D
→|TAD  →|E
→|DAC  →|F
```

The assembler enables the following basic instruction sequence to be represented in the source program by a single macro instruction. To employ macros, it is first necessary to define the desired coding sequence with dummy arguments as a macro instruction; the defined instruction may then be referenced by name, together with the real arguments, as a single statement each time the equivalent coding sequence is needed in the program. Refer to the PDP-15 MACRO-15 Assembler manual (DEC-15-AMZB-D) for a complete description of macros.

```
→|LAC  →|x
→|TAD  →|y               NOTE:  x, y, and z are dummy arguments.
→|DAC  →|z
```

The ADVANCED Monitor provides the user with access to a set of pre-defined macros (referred to as system macros) as a programming convenience. These system macros are referenced (called) in user programs by writing a statement comprising an assigned macro name followed, if needed, by a list of real arguments separated by commas Macro statements are terminated by either a space (␣), a tab (→|), or a carriage return (↵). For example:

.SEEK →|7, NAME 1↵

3.1.1 Summary of ADVANCED Monitor System MACROs.

The following is a summary of the System MACROs which are recognized by the PDP-15 ADVANCED Monitor. Individual detailed descriptions are provided in paragraphs 3.1.2 through 3.1.17.

Name	
.INIT	Initializes the device and device handler.
.DLETE	Deletes file from file-oriented device.
.RENAM	Renames file on file-oriented device.
.FSTAT	Checks presence of file on file-oriented device.
.SEEK	Locates file on file-oriented device and begins data input.
.ENTER	Primes file-oriented device for output.
.CLEAR	Initializes file structure on file-oriented device.
.CLOSE	Terminates use of a file.
.MTAPE	Provides special commands for industry compatible magnetic tape.
.READ	Transfers data from the device to the user's line buffer.
.WRITE	Transfers data from the user's line buffer to the device.
.WAIT	Checks the 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.
.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.
.TIMER	Calls and uses Real Time Clock option.
.EXIT	Returns control to the Monitor.

The first seven MACROS listed above (excluding .INIT) apply to file-oriented devices (i.e., DECtape, DECdisk, and MAGtape); they are either ignored or treated as illegal (depending upon the function) by non-file-oriented functions of magnetic tape (REWIND, BACKSPACE, etc.). If these non-file-oriented commands are issued to file-oriented devices, they are either ignored or flagged as errors. Two .MTAPE commands (REWIND TO LOAD POINT and BACKSPACE RECORD), however, may be used with DECtape and DECdisk with the appropriate handler version. When so used, these commands preclude the use of .SEEK or .ENTER. Refer to paragraph 5.4 for specific device handler characteristics.

3.1.2 .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[1] (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[2].
LOC + 2	R	
LOC + 3	n	/Maximum size of line buffer /associated with .DAT slot a, for /example, 255_{10} for DECtape.[3]

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.

3.1.3 .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

---

[1]Has meaning only for .INIT commands referencing slots used by the TTY (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 (e.g., another paper tape in the reader.

[2]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.

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

the file to be deleted from the device associated with .DAT slot a.

EXPANSION: LOC CAL + 1000 + a_{9-17}

EXPANSION:	LOC	CAL + 1000 + 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: .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.

3.1.4 .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	

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.

3.1.5 .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[1]

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 reoommended 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 teleprinter.

3.1.6 .SEEK

FORM: .SEEK a,D

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

D = Address of user directory entry block

EXPANSION: LOC   CAL + a_{9-17}

LOC + 1   3   /The CAL handler will place unit /number (if applicable) into bits /0 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; if one is 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)

---

[1]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 (file structuring)
4 = Magnetic tape

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.

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, unless otherwise specified, use predetermined file name extensions in their operation. For example, if MACRO-15 wishes to .SEEK program ABCDEF as source input and the user has not specified an extension, 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. FORTRAN IV is an exception to the above conventions in that it assumes the input file name extension is always SRC.

3.1.7 .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.

3.1.8 .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).

3.1.9 .CLOSE

FORM: .CLOSE a

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

EXPANSION:

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

3.1.10 .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     7          /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 handler that is capable of being 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.

3.1.11 .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
- 5 = 9-channel Dump Mode (MAGtape only)

L = Line Buffer address

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

```
EXPANSION:  LOC         CAL + M6-8 + a9-17

            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 and 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.

3.1.12 .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
- 5 = 9-channel Dump Mode (MAGtape only)

L = Line buffer address

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

```
EXPANSION:  LOC         CAL + M6-8 to a9-17

            LOC + 1     11          /CAL handler will place the unit
                                    /number (if applicable) 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 a .DAT slot a.

.WAIT must be used after a .WRITE command, before the line buffer is is used again, to ensure 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.

3.1.13 .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.

3.1.14 .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.

3.1.15 .TRAN

FORM: .TRAN a, D, B, L, W, P

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

D = Transfer direction

- 0 = Input Forward
- 1 = Output Forward
- 2 = Input Reverse[1]
- 3 = Output Reverse[1]

B[2] = Device address; for example, block number (octal radix) for DECtape

L = Core starting address

W = Word count (decimal radix)

P[3] = High order 3 bits of device address (e.g., RS15 DECdisk platter number, 0-7).

EXPANSION:

LOC	CAL + D_{7-8} + a_{9-17} or CAL + P_{5-7} + D_8 + a_{9-17}	
LOC + 1	13	/The CAL handler will place the unit /number (if applicable) associated /with .DAT slot a into bits 0 /through 2.
LOC + 2	B	
LOC + 3	L	
	.DEC	/Decimal radix
LOC + 4	-W	

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

3.1.16 .TIMER

FORM: .TIMER n, C

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

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

---

[1]DECtape only.

[2]Ignored for magnetic tape.

[3]The argument P is omitted for devices other than the DECdisk. If the argument P is present, the argument D must be either 0 or 1; values of 2 or 3 for D will produce erroneous results.

```
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 $1/60^{th}$ second for 60-Hz systems and $1/50^{th}$ second 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
```

3.1.17 .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 BASIC I/O Monitor environment, it causes a program halt; in the ADVANCED Monitor environment, it causes the non-resident Monitor to be reloaded. When the reloading process has been completed, the Monitor types

```
KM 15 Vnn
$
```

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

CHAPTER 4

ADVANCED MONITOR

4.1 ADVANCED MONITOR FUNCTIONS

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

4.2 PROGRAMMING EXAMPLE

The following example illustrates the use of system macros with MACRO-15 programs in the ADVANCED Monitor Environment. The example inputs a line of data from the teleprinter keyboard, writes it on DECtape, reads it back from DECtape, and outputs it on the teleprinter. 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, ECHOA TST, ECHOB TST, ECHOC 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 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:

Source Listing

```
        .TITLE DTECHO
DECTAPE=7
TTI=6
TTO=5
IN=Ø
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,RESTRT           /AND TELETYPE 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                              /Ø 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 DECTAPE
        .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     (774ØØØ                  /SAVE FIRST SEVEN BITS
        SAD     (544ØØØ                  /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*1ØØØ                 /WPC FOR HEADER WORD Ø
        Ø
        .ASCII "FILE ALREADY "
        .ASCII "PRESENT!!"<15>
MSG2    COM-MSG2/2*1ØØØ                  /WPC FOR HEADER WORD Ø
        Ø
        .ASCII "DO YOU WISH TO KEEP IT ?"
        .ASCII "(Y OR N) AND CR."<15>
COM     .BLOCK  1Ø
BUFFER  .BLOCK  42
NAME    .SIXBT  "ECHO@@TST"
UDSW    Ø
        .END    START
```

Example (continued)

Assembly Listing

```
PAGE    1       DTECHO.SRC       DTECHO

                                          .TITLE DTECHO
                   000007 A    DECTAPE=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   DECTAPE,OUT,RESTRT      /INITIALIZE DECTAPE OUTPUT,
       00000 R 001007 A *G                CAL+OUT*1000 DECTAPE&777
       00001 R 000001 A *G                1
       00002 R 000070 R *G                RESTRT+0
       00003 R 000000 A *G                0
                                          .INIT   TTI,IN,RESTRT           /TELETYPE INPUT,
       00004 R 000006 A *G                CAL+IN*1000 TTI&777
       00005 R 000001 A *G                1
       00006 R 000070 R *G                RESTRT+0
       00007 R 000000 A *G                0
                                          .INIT   TTO,OUT,RESTRT          /AND TELETYPE OUTPUT
       00010 R 001005 A *G                CAL+OUT*1000 TTO&777
       00011 R 000001 A *G                1
       00012 R 000070 R *G                RESTRT+0
       00013 R 000000 A *G                0
                                          .FSTAT  DECTAP,NAME             /IS FILE PRESENT?
       00014 R 003007 A *G                CAL+3000 DECTAP&777
       00015 R 000002 A *G                2
       00016 R 000246 R *G                NAME
       00017 R 740200 A                   SZA                             /NO, INPUT KEYBOARD
       00020 R 600077 R                   JMP     UPDATE                  /YES, OUTPUT MSG1 AND MSG2
       00021 R                 READKB     .READ   TTI,IOPS,BUFFER,34      /INPUT IOPS ASCII FROM KEYBOARD
       00021 R 002006 A *G                CAL+IOPS*1000 TTI&777
       00022 R 000010 A *G                10
       00023 R 000204 R *G                BUFFER
                          *G              .DEC
       00024 R 777736 A *G                -34
                                          .WAIT   TTI                     /WAIT UNTIL INPUT COMPLETE
       00025 R 000006 A *G                CAL TTI&777
       00026 R 000012 A *G                12
                                          .EJECT
```

```
 18   ØØØ27 R 2ØØ251 R               LAC     UDSW                    /TEST UPDATE SWITCH
 19   ØØØ3Ø R 74Ø2ØØ A               SZA                             /Ø REPLACE INPUT FILE
 2Ø   ØØØ31 R 6ØØ132 R               JMP     NEWFIL                  /-1=SAVE INPUT; CREATE NEW OUTPT
 21   ØØØ32 R                 WRITE  .ENTER  DECTAP,NAME             /LOCATE FREE DECTAPE FILE
      ØØØ32 R ØØØØØ7 A *G            CAL DECTAP&777
      ØØØ33 R ØØØØØ4 A *G            4
      ØØØ34 R ØØØ246 R *G            NAME
 22                                  .WRITE  DECTAP,IOPS,BUFFER,34   /OUTPUT DATA ON DECTAPE
      ØØØ35 R ØØ2ØØ7 A *G            CAL+IOPS*1ØØØ DECTAP&777
      ØØØ36 R ØØØØ11 A *G            11
      ØØØ37 R ØØØ2Ø4 R *G            BUFFER
                       *G            .DEC
      ØØØ4Ø R 777736 A *G            -34
 23                                  .WAIT   DECTAP                  /WAIT UNTIL OUTPUT COMPLETED
      ØØØ41 R ØØØØØ7 A *G            CAL DECTAP&777
      ØØØ42 R ØØØØ12 A *G            12
 24                                  .CLOSE  DECTAP                  /CLOSE FILE
      ØØØ43 R ØØØØØ7 A *G            CAL DECTAP&777
      ØØØ44 R ØØØØØ6 A *G            6
 25   ØØØ45 R                 READDT .INIT   DECTAP,IN,RESTRT        /INITIALIZE DECTAPE INPUT
      ØØØ45 R ØØØØØ7 A *G            CAL+IN*1ØØØ DECTAP&777
      ØØØ46 R ØØØØØ1 A *G            1
      ØØØ47 R ØØØØ7Ø R *G            RESTRT+Ø
      ØØØ5Ø R ØØØØØØ A *G            Ø
 26                                  .SEEK   DECTAP,NAME             /LOCATE FILE "NAME"
      ØØØ51 R ØØØØØ7 A *G            CAL DECTAP&777
      ØØØ52 R ØØØØØ3 A *G            3
      ØØØ53 R ØØØ246 R *G            NAME
 27                                  .READ   DECTAP,IOPS,BUFFER,34   /READ INTO BUFFER
      ØØØ54 R ØØ2ØØ7 A *G            CAL+IOPS*1ØØØ DECTAP&777
      ØØØ55 R ØØØØ1Ø A *G            1Ø
      ØØØ56 R ØØØ2Ø4 R *G            BUFFER
                       *G            .DEC
      ØØØ57 R 777736 A *G            -34
 28                                  .WAIT   DECTAP                  /WAIT UNTIL READ COMPLETE
      ØØØ6Ø R ØØØØØ7 A *G            CAL DECTAP&777
      ØØØ61 R ØØØØ12 A *G            12
 29                                  .EJECT
```

```
                                          .WRITE  TTO,IOPS,BUFFER,34      /OUTPUT TO TELETYPE
00062 R 002005 A *G                       CAL+IOPS*1000 TTO&777
00063 R 000011 A *G                       11
00064 R 000204 R *G                       BUFFER
                 *G                       .DEC
00065 R 777736 A *G                       -34
                                          .WAIT   TTO                     /WAIT UNTIL OUTPUT COMPLETE
00066 R 000005 A *G                       CAL TTO&777
00067 R 000012 A *G                       12
00070 R                          RESTRT   .CLOSE  TTO                     /TERMINATE TELETYPE OUTPUT,
00070 R 000005 A *G                       CAL TTO&777
00071 R 000006 A *G                       6
                                          .CLOSE  TTI                     /TELETYPE INPUT,
00072 R 000006 A *G                       CAL TTI&777
00073 R 000006 A *G                       6
                                          .CLOSE  DECTAP                  /AND DECTAPE INPUT/OUTPUT
00074 R 000007 A *G                       CAL DECTAP&777
00075 R 000006 A *G                       6
00076 R 600000 R                          JMP     START                   /LOOP FOR UPDATE OPTION
00077 R                          UPDATE   .WRITE  TTO,IOPS,MSG1,34        /OUTPUT MSG1
00077 R 002005 A *G                       CAL+IOPS*1000 TTO&777
00100 R 000011 A *G                       11
00101 R 000134 R *G                       MSG1
                 *G                       .DEC
00102 R 777736 A *G                       -34
                                          .WAIT   TTO                     /AND MSG2
00103 R 000005 A *G                       CAL TTO&777
00104 R 000012 A *G                       12
                                          .EJECT
```

```
                                    .WRITE  TTO,IOPS,MSG2,34         /ON
   00105 R 002005 A *G              CAL+IOPS*1000 TTO&777
   00106 R 000011 A *G              11
   00107 R 000150 R *G              MSG2
                    *G              .DEC
   00110 R 777736 A *G              -34
                                    .WAIT   TTO                      /TELETYPE
   00111 R 000005 A *G              CAL TTO&777
   00112 R 000012 A *G              12
                                    .READ   TTI,IOPS,COM,8           /READ RESPONSE
   00113 R 002006 A *G              CAL+IOPS*1000 TTI&777
   00114 R 000010 A *G              10
   00115 R 000174 R *G              COM
                    *G              .DEC
   00116 R 777770 A *G              -8
                                    .WAIT   TTI                      /WAIT UNTIL READ COMPLETE
   00117 R 000006 A *G              CAL TTI&777
   00120 R 000012 A *G              12
   00121 R 200176 R                 LAC     COM+2                    /GET FIRST WORD
   00122 R 500252 R                 AND     (774000                  /SAVE FIRST SEVEN BITS
   00123 R 540253 R                 SAD     (544000                  /IS CHAR A Y?
   00124 R 600127 R                 JMP YES
   00125 R 140251 R                 DZM     UDSW                     /NO, SET TO REPLACE INPUT FILE
   00126 R 600021 R                 JMP     READKB                   /LOOP TO READ KEYBOARD
   00127 R 750001 A        YES      CLC                              /SET UPDATE SW, TO SAVE
   00130 R 040251 R                 DAC     UDSW                     /INPUT, CREATE NEW OUTPUT
   00131 R 600021 R                 JMP     READKB                   /LOOP TO READ KEYBOARD
   00132 R 440247 R        NEWFIL   ISZ     NAME+1                   /CHANGE FILE NAME
   00133 R 600032 R                 JMP     WRITE                    /TO CREATE NEW OUTPUT
   00134 R 006000 A        MSG1     MSG2-MSG1/2*1000                 /WPC FOR HEADER WORD 0
   00135 R 000000 A                 0
   00136 R 432231 A                 .ASCII "FILE ALREADY "
   00137 R 442500 A
   00140 R 406312 A
   00141 R 242602 A
   00142 R 422624 A
   00143 R 000000 A
   00144 R 502450 A                 .ASCII "PRESENT!!"<15>
   00145 R 551612 A
   00146 R 472504 A
   00147 R 120432 A
                                    .EJECT
```

```
PAGE   5      DTECHO SRC      DTECHO

  59        00150 R 012000 A     MSG2     COM-MSG2/2*1000              /WPC FOR HEADER WORD 0
  60        00151 R 000000 A              0
  61        00152 R 422364 A              .ASCII "DO YOU WISH TO KEEP IT ?"
            00153 R 054636 A
            00154 R 525012 A
            00155 R 744646 A
            00156 R 441012 A
            00157 R 447500 A
            00160 R 456130 A
            00161 R 550100 A
            00162 R 446504 A
            00163 R 037400 A
  62        00164 R 242624 A              .ASCII "(Y OR N) AND CR."<15>
            00165 R 047644 A
            00166 R 202345 A
            00167 R 120202 A
            00170 R 472104 A
            00171 R 041644 A
            00172 R 270320 A
            00173 R 000000 A
  63        00174 R        A     COM      .BLOCK  10
  64        00204 R        A     BUFFER   .BLOCK  42
  65        00246 R 050310 A     NAME     .SIXBT  "ECHO@@TST"
            00247 R 170000 A
            00250 R 242324 A
  66        00251 R 000000 A     UDSW     0
  67                000000 R              .END    START
            00252 R 774000 A *L
            00253 R 544000 A *L
                SIZE=00254     NO ERROR LINES
```

```
PAGE   6      DTECHO SRC      DTECHO

BUFFER  00204 R    COM      00174 R    DECTAP 000007 A    IN      000000 A
IOPS   000002 A    MSG1     00134 R    MSG2    00150 R    NAME     00246 R
NEWFIL  00132 R    OUT     000001 A    READDT  00045 R    READKB   00021 R
RESTRT  00070 R    START    00000 R    TTI    000006 A    TTO     000005 A
UDSW    00251 R    UPDATE   00077 R    WRITE   00032 R    YES      00127 R
```

```
PAGE    7      DTECHO SRC      DTECHO

IN       000000 A     START    00000 R     OUT      000001 A     IOPS     000002 A
TTO      000005 A     TTI      000006 A    DECTAP   000007 A     READKB    00021 R
WRITE     00032 R     READDT    00045 R    RESTRT    00070 R     UPDATE    00077 R
YES       00127 R     NEWFIL    00132 R    MSG1      00134 R     MSG2      00150 R
COM       00174 R     BUFFER    00204 R    NAME      00246 R     UDSW      00251 R
```

```
PAGE    8     DTECHO CROSS REFERENCE

BUFFER   00204   15    22    27    30    64*
COM      00174   41    43    59    63*
DECTAP  000007    2*    9    12    21    22    23    24    25    26
                 27    28    34
IN      000000    5*   10    25
IOPS    000002    7*   15    22    27    30    36    39    41
MSG1     00134   36    54*   54
MSG2     00150   39    54    59*   59
NAME     00246   12    21    26    52    65*
NEWFIL   00132   20    52*
OUT     000001    6*    9    11
READDT   00045   25*
READKB   00021   15*   48    51
RESTRT   00070    9    10    11    25    32*
START    00000    9*   35    67
TTI     000006    3*   10    15    16    33    41    42
TTO     000005    4*   11    30    31    32    36    37    39    40
UDSW     00251   18    47    50    66*
UPDATE   00077   14    36*
WRITE    00032   21*   53
YES      00127   46    49*
```

4.3 KEYBOARD COMMANDS

The ADVANCED Monitor provides:

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 ADVANCED Monitor initially gets control it outputs:

```
KM15 Vnn
$
```

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

ADVANCED 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.)

4.3.1 System Program Load Commands

Loading commands instruct the ADVANCED 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 Monitor.

Command	System Program Loaded
F4	FORTRAN IV Compiler
F4I	8K FORTRAN IV Compiler (DECtape I/O only)
MACRO	MACRO-15 Assembler
MACROI	8K MACRO-15 Assembler (DECtape only)
PIP	Peripheral Interchange Program
EDIT	Symbolic Text Editor
EDITVP	Symbolic Text Editor (VP15A Display)
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	Program to create a system of core overlays
EXECUTE (E)	Control program to supervise core residency during execution of CHAIN built overlay system
SGEN	System Generator program
SRCCOM	Text Line Comparison program.
DTCOPY	8K high speed DECtape copy program

All commands should be terminated by a Carriage RETURN(↵) or ALT MODE. When the requested program has been loaded and is waiting for keyboard input, an indication is given on the teleprinter with an appropriate message, such as

```
       LOADER  Vnn
       >

or     F4      Vnn
       >

or     EDITOR  Vnn
       >
       etc.            where:  Vnn = current version of the
                                     program
```

4.3.2 Special Function Commands

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

4.3.2.1 LOG (or L) - The LOG command is used to make hard copy records of user comments on the teleprinter. 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)

4.3.2.2 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:

```
KM15 V5A

$SCOM

 SYSTEM INFO - V5A - 7/1/70

 17646 - BOOTSTRAP RESTART ADDR
 17636 - 1ST FREE CELL BELOW BOOTSTRAP
 1745 - 1ST FREE CELL ABOVE RESIDENT MONITOR
 141 - ADDR OF .DAT
 565 - ↑Q ADDRESS FOR MANUAL DUMP
 101 - START BLOCK FOR ↑Q DUMP AREA
 255 - KM15 START WITH RESTART ADDRESS IN CELL 0
 SYSTEM HAS EAE
 7 CHANNEL MAGTAPE ASSUMED BY HANDLERS
 I/O HANDLERS AVAILABLE:
 TTA  TELETYPE: I/O, ASCII MODES, ALL FUNCTIONS
 PRA  TAPE READER: INPUT, ALL MODES, ALL FUNCTIONS
 PRB  TAPE READER: INPUT, IOPS ASCII MODE, ALL FUNCTIONS
 PPA  PUNCH: OUTPUT, ALL MODES, ALL FUNCTIONS
 PPB  PUNCH: OUTPUT, ALL MODES LESS IOPS ASCII, ALL FUNCTIONS
 PPC  PUNCH: OUTPUT, IOPS BINARY MODE, ALL FUNCTIONS
 DTA  DECTAPE: 3 FILES, I/O, ALL MODES, ALL FUNCTIONS
 DTB  DECTAPE: 2 FILES, I/O, IOPS MODES, LIM FUNCTIONS
 DTC  DECTAPE: 1 FILE, INPUT, IOPS MODES, LIMITED FUNCTIONS
 DTD  DECTAPE: 1 FILE, I/O, ALL MODES, ALL FUNCTIONS
 DTE  DECTAPE: 1 FILE, I/O, ALL MODES, ALL FUNCTIONS EXCEPT .MTAPE
 DTF  DECTAPE: NON-FILE ORIENTED FOR F4 .OTS
 DKA  DECDISK: 3 FILES, I/O, ALL MODES, ALL FUNCTIONS
 DKB  DECDISK: 2 FILES, I/O, IOPS MODES, LIM FUNCTIONS
 DKC  DECDISK: 1 FILE, INPUT, IOPS MODES, LIM FUNCTIONS
 DKD  DECDISK: 1 FILE, I/O, ALL MODES, ALL FUNCTIONS
 DKE  DECDISK: 1 FILE, I/O, ALL MODES, ALL FUNCTIONS EXCEPT .MTAPE
 DKF  DECDISK: NON-FILE ORIENTED FOR F4 .OTS
 MTA  MAGTAPE: 3 FILES, I/O, ALL MODES, ALL FUNCTIONS
 MTC  MAGTAPE: 1 FILE, INPUT, IOPS MODES, ALL FUNCTIONS
 MTF  MAGTAPE: NON-FILE ORIENTED FOR F4 .OTS
 LPA  LINE PRINTER: OUTPUT, IOPS ASCII MODE, ALL FUNCTIONS
 CDB  CARD READER: INPUT, IOPS ASCII MODE, ALL FUNCTIONS
 VPA  VP DISPLAY: OUTPUT, ASCII AND DUMP MODES, ALL FUNCTIONS
 SKIP CHAIN ORDER
     SPFAL
     DTDF
     DSSF
     MTSF
     SDDF
     RCSF
     RCSD
     LSDF
     CLSF
     RSF
     PSF
     KSF
     TSF
     DTEF
     MPSNE
     MPSK
     SPE
```

4.3.2.3 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
```

4.3.2.4 QDUMP (or ↑Q)[1] - In the event of an unrecoverable error, this command conditions the Monitor to dump memory on the "save, or CTRL Q, area" of one of the units of the system device.

QDUMP forces automatic execution of the CTRL Q command (described in Paragraph 4.3.3) 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 ENABLE switches on the system device should be enabled in case of error; otherwise, an IOPS 4 (not ready) error will follow the initial error.

4.3.2.5 HALT (or H)[1] - 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 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.)

4.3.2.6 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 teleprinter; INSTRUCT ERRORS causes a summary of system error messages to be printed.

Example:

```
$I

 KM15 COMMANDS:
    LOG(L): USER COMMENTS TERMINATED BY ALTMODE
    SCOM(S): SYSTEM INFO
    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 MODS
    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
```

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

```
    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
    CHANNEL 7/9: SETUP DEFAULT ASSUMPTION FOR MAGTAPE
    ↑C: RESTORE KM15
    ↑P: USER RESTART
 KM15 PROG LOADING COMMANDS AND PROGNAM FOR REQUEST COMMAND
    LOAD: LINK LOAD AND WAIT FOR ↑S
    GLOAD: LINK LOAD AND GO
    DDT: LINK LOAD WITH SYMBOLS AND GO TO DDT
    DDTNS: LINK LOAD W/O SYMBOLS AND GO TO DDT
    MACRO: SYMBOLIC MACRO ASSEMBLER
    MACROI: 8K DECTAPE I/O MACRO ASSEMBLER
    F4: FORTRAN IV COMPILER
    F4I: 8K DECTAPE I/O FORTRAN IV COMPILER
    EDIT: TEXT EDITOR
    PIP: PERIPHERAL INTERCHANGE PROG
    SGEN: SYSTEM GENERATOR
    DUMP: BULK STOR DEV DUMP
    UPDATE: LIBR FILE UPDATE
    SRCCOM: SOURCE COMPARE
    EDITVP: SCOPE EDITOR
    PATCH: SYSTEM TAPE PATCH ROUTINE
    EXECUTE(E) FILE: LOAD AND RUN FILE XCT
    CHAIN: XCT CHAIN BUILDER
 KM15: BATCH
    BATCH(B) DV: ENTER BATCH MODE WITH DV AS BATCH DEV
        DV: PR = PAPER TAPE READER
            CD = CARD READER
    $JOB: CONTROL COMMAND WHICH SEPARATES JOBS
    $DATA: BEGINNING OF DATA
    $END: END OF DATA
    $EXIT: LEAVE BATCH MODE
    ↑T: SKIP TO NEXT JOB
    ↑C: LEAVE BATCH MODE
```

4.3.2.7 REQUEST (or R) - The REQUEST command allows examination of the .DAT slots associated with various programs[1]. 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.

[1]See Paragraph 5.3 for .DAT slots used by system programs, their uses, and acceptable I/O handlers.

Examples:

```
$REQUEST

 .DAT   DEVICE  USE

  -15   DTA2    OUTPUT
  -14   DTA1    INPUT
  -13   PPCØ    OUTPUT FOR MACRO, F4
  -12   TTAØ    LISTING
  -11   PRBØ    INPUT FOR MACRO, F4
  -1Ø   TTAØ    INPUT
  -7    DTCØ    SYS DEV FOR .SYSLD
  -6    DTB2    OUTPUT FOR CHAIN
  -5    NONE    USER LIBR FOR .LOAD
  -4    DTC2    SYS INPUT
  -3    TTAØ    TTY OUT
  -2    TTAØ    TTY IN
  -1    DTCØ    SYS DEV 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

  -14   DTA1    INPUT
  -13   PPCØ    OUTPUT
  -12   TTAØ    LISTING
  -11   PRBØ    INPUT
  -1Ø   TTAØ    SECONDARY INPUT
  -3    TTAØ    CONTROL AND ERROR MES
  -2    TTAØ    COMMAND STRING
```

4.3.2.8 ASSIGN (or A) - The ASSIGN command allows temporary reassignment of .DAT slots to devices other than those set at system generation (SGEN program). 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./DEV. 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[1]). If the

[1]See Paragraph 5.3 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.

third letter of a handler name is omitted, the letter A is assumed.

n, m are unit numbers (if non 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
```

DEVn can be replaced by NON to clear .DAT slots.

```
$ASSIGN NON 4, 5, 10
```

.DAT slots -2 and -3 are permanent and can not be modified.
.DAT slot -7 is automatically modified only at system generation time to the smallest system device handler.

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

The command takes the following form:

```
DIRECT N
```

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

Example:

```
KM15 V5A
$DIRECT

 DIRECTORY LISTING
 .LOAD  BIN    36
 DDT    BIN    37
 EXECUT BIN    40
 INTEAE BIN    41
 INTNON BIN    47
 RELEAE BIN    54
 RELNON BIN   104
 .LIBR  BIN   105
 FOCAL  BIN   122
 8TRAN  BIN   244
```

```
FNEW    SRC   300
TIME    BIN   333
TIME10  BIN   340
FOCAL   XCT   345
FOCAL   XCU   352
KM15    SYS     0
SKPBLK  SYS    42
IOBLK   SYS    46
SGNBLK  SYS    52
SYSHAN  SYS    56
SYSBLK  SYS    61
.SYSLD  SYS    62
BITMAP  SYS    71
DIRECT  SYS   100
F4I     SYS   141
MACROI  SYS   201
EDIT    SYS   627
EDITVP  SYS   641
PIP     SYS   656
MACRO   SYS   676
CHAIN   SYS   734
F4      SYS   754
DUMP    SYS  1007
DTCOPY  SYS  1013
PATCH   SYS  1016
UPDATE  SYS  1025
SRCCOM  SYS  1035
SGEN    SYS  1047
114   FREE BLOCKS
```

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

4.3.2.11 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 of the system device which contains the CTRL Q area to be retrieved, and xxxxx is a program starting address.

GET retrieves the core image (including the Monitor) stored on unit n of the system device by CTRL Q commands and restores it to memory. Control is transferred to address xxxxx, if specified. If HALT was specified, the computer halts to permit the starting address to be placed in the ADDRESS switches. Execution is initiated by pressing the START button (PIC and API are enabled). If neither xxxxx nor HALT is specified, the core image is restored in memory and the Resident Monitor waits in a teleprinter loop with API and/or PI on for one of the following commands to be typed:

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

4.3.2.12 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, then 9 channel is assumed. This default condition can also be set at system generation time by answering yes or no to the question

"7 CHANNEL MAGTAPE?"

4.3.3 Control Character Commands

All of the ADVANCED Monitor control character commands (except RUBOUT) are formed by holding down the CTRL key while striking the appropriate letter key. The commands, the character(s) echoed on the teleprinter, and the resulting actions are summarized in Table 4-1.

4.4 OPERATING THE ADVANCED MONITOR SYSTEM

Detailed operating procedures for utilizing the system programs in the ADVANCED Monitor environment are given in the PDP-15/20 Users Guide (DEC-15-MG2B-D). The following paragraphs present general descriptions of the operations involved in loading the ADVANCED Monitor, system generating, assigning devices, loading programs, and error detecting and handling.

The ADVANCED Software System is supplied to all users in the form of a DECtape reel. Special DECtape-to-Disk and Magnetic Tape-to-Disk utility routines are provided to users who purchase optional disk storage units and use disk as the system device.

4.4.1 Loading the ADVANCED Monitor

Each installation employing the DECtape or DECdisk version of the ADVANCED Monitor must reserve unit 0 as the system device. The ADVANCED Monitor, the Input/Output Programming System, and all system and library programs needed by the user will reside on this unit.

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,

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 C	↑C	Forces control back to Monitor, which types KM15 Vnn $ to indicate that it is awaiting a keyboard command. 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.
CTRL P	↑P	Forces control to address specified in the last .INIT command referencing teleprinter. 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 ENABLE 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 teleprinter (input or output).
RUBOUT	\	Cancels last character input from teleprinter (not applicable with DDT).

depressing STOP and RESET, and then the READIN switch, the bootstrap is loaded into upper core. It clears all flags, disables the Program Interrupt (and the Automatic Priority Interrupt, if available), loads the Monitor from the system device into lower core, and transfers control to it. The Monitor types

```
KM15   Vnn
$
```

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 STOP and RESET, and then START.

4.4.2 System Generation

The System Generator (SGEN) is a standard system program used to create new system software configurations, either on DECtape or DECdisk. Upon receiving a PDP-15/20 system, the user should immediately create a standard system for his installation. This is done by loading the System Bootstrap, which loads the Monitor into core from the DECtape supplied with the system, and then calling the System Generator via the teleprinter. SGEN will create a new system on the device associated with .DAT slot -15. The ASSIGN command must be used prior to calling SGEN to assign a bulk storage device to .DAT slot -15 and the old system device to .DAT slot -15 as follows:

```
$ASSIGN DTA0 -14DTA2 -15(or DKD2[1]-15)⤶
$SGEN⤶
```

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

a. The device on which 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.

---

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

```
$ASSIGN DKA0 -15/DTD2 -15⤶
```

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, 12, 16, 20, 24, 28, 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 PDP-15 Utility Programs Manual.

The following paragraphs are intended to assist 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 obtaining 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.

4.4.2.1 DECtape or DECdisk Systems - All users having either DECtape or DECdisk receive the ADVANCED Monitor system as an 8K, EAE, non-API, KSR-35 DECtape system. Each user having a core configuration greater than 8K, the API option, DECdisk, or a KSR-33 teleprinter, should perform a system generation in order to tailor his software system for maximum efficient use. Also users who, upon examining the .SCOM printout, discover devices or options listed that are not present in their system may wish to eliminate the irrelevant skips from the skip chain. Those with non-standard devices (A/D converter, for example) must expand the skip chain. The standard 8K DECtape skip chain is as follows:

SPFAL	Power Fail
DTDF	DECtape Done
DSSF	DECdisk Done
MTSF	Magnetic Tape Done or Error
SDDF	VP15A Display Done
RCSF	Card Column Ready
RCSD	Card Done
LPSF	Line Printer Done or Error
CLSF	Clock Done
RSF	Reader 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

It is important that the above order remain intact even if deletions or additions are to be made. For example, given a system without the Power Fail, Parity VP-15A Displays, or Memory Protect options, and without 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	PPC0	OUTPUT
-12	TTA0	LISTING
-11	PRB0	INPUT
-10	TTA0	INPUT
-7	DTC0	SYSTEM DEVICE FOR .SYSLD
-6	DTB2	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
1	DTA0	USER
2	DTA1	USER
3	DTA2	USER
4	TTA0	USER
5	PRA0	USER
6	PPA0	USER
7	DTA1	USER
10	DTA2	USER

The following examples are variations on .DAT slot assignments[1] 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-2 and one of the positive slots, for example, 3, 7, or 10. CDB should be assigned to one of the positive slots.

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

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

-15	DKA2	-3	TTA
-14	DKA1	-2	TTA
-13	DKA3	-1	DKA0
-12	TTA	1	DKA1
-11	DKA1	2	DKA2
-10	PRA	3	DKA3
-7	DKC0	4	TTA
-6	DKA3	5	PRA
-5	NONE	6	PPA
-4	DKA3	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	-3	TTA
-14	DTA1	-2	TTA
-13	DTA2	-1	DTA0
-12	TTA	1	DTA0
-11	DTA1	2	DTA1
-10	PRA	3	DTA2
-7	DTC0	3	TTA
-6	DTA2	5	PRA
-5	NONE	6	PPA
-4	DTA2	7	MTF1
		10	MTF2

4.4.3 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 Paragraph 4.3.2.8) is used to attach hardware devices to the slots of the device assignment table. Table 4-2 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, FOCAL and EDITVP are exceptions to this rule in that they use both the positive and negative .DAT slots.

4.4.4 Loading Programs in the ADVANCED Monitor Environment

In the ADVANCED Monitor environment, most system programs are called by unique keyboard commands (see paragraph 4.3.1). User programs and some system programs (e.g., FOCAL and 8TRAN) 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.

Table 4-2

Function of .DAT Slots in the ADVANCED Monitor System

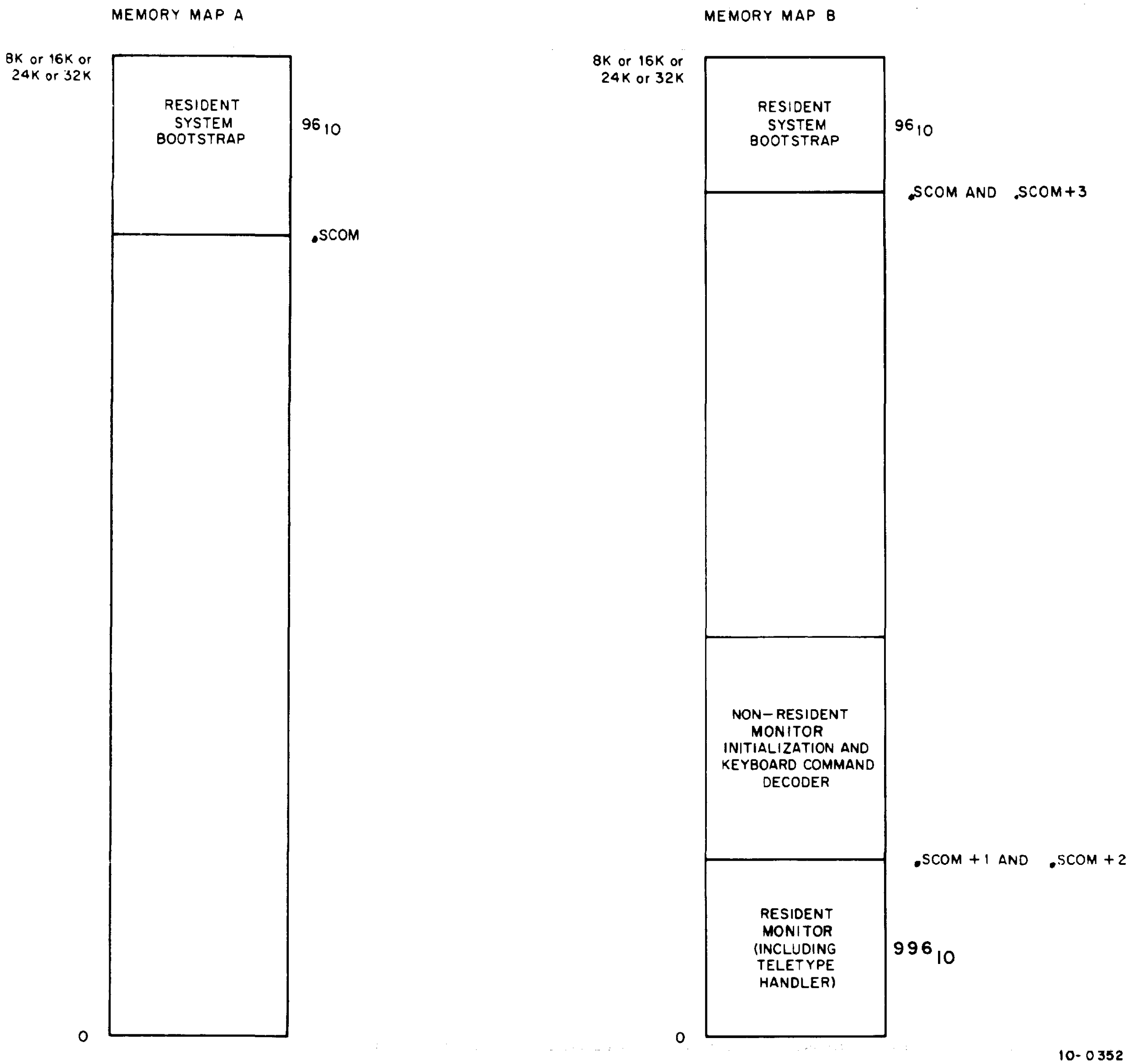
.DAT Slot	Device Handler 8K	Device Handler 12K or greater	Use	
-15	DTA2	DTA2	Output	-UPDATE, SGEN, DTCOP
			Input	-8TRAN, SRCCOM (old file)
			Scratch	-EDIT, EDITVP
-14	DTA1	DTA1	Input	-EDIT, EDITVP, UPDATE, SGEN, DTCOP, SRCCOM (new file), DUMP
			Secondary Input	-MACRO (macro definitions file)
			Input/Output	-PATCH
			Output	-8TRAN
-13	PPC[1]	DTA2	Output	-MACRO, FORTRAN IV
-12	TTA	TTA	Listing Output	-MACRO, FORTRAN IV, UPDATE, DUMP, SRCCOM
-11	PRB0[1]	DTA1	Input	-MACRO, FORTRAN IV
-10	PRA	PRA	Secondary Input	-EDITOR, EDITVP, UPDATE, PATCH, MACRO (parameter file)
-7	DTC0	DTC0	Input	-System device for the System Loader
-6	DTB2	DTA2	Output	-CHAIN (XCT overlay system)
-5	NONE	NONE	External Library	-Linking Loader, CHAIN, DDT (these programs expect the name .LIBR5 BIN if a device is assigned)
-4	DTC2[2]	DTA2	Input	-Linking Loader, DDT, CHAIN, EXECUTE
-3	TTA	TTA	Output	-All system programs
-2	TTA	TTA	Input	-All system programs
-1	DTC0[2]	DTA0	Input (System Library)	-FORTRAN IV, Linking Loader, CHAIN, DDT
1	DTA0	DTA0	Input/Output	-PIP, User
2	DTA1	DTA1	Input/Output	-PIP, User
3	DTA2[3]	DTA2[3]	Input/Output	-PIP, User
			Input	-FOCAL library
4	TTA	TTA	Input/Output	-User
5	PRA[3]	PRA[3]	Input	-PIP, User
			Output	-FOCAL library
6	PPA	PPA	Output	-PIP, User
7	DTA1[3]	DTA1[3]	Input/Output	-PIP, User
			Input	-FOCAL (data files)
10	DTA2[3,4]	DTA2[3,4]	Input/Output	-PIP, User
			Output	-EDITVP (display), FOCAL (data files)

[1]Use MACROI or F4I for 8K DECtape I/O

[2]DTC is sufficient for the Linking Loader; however, since the Linking Loader shares the handler with the program it loads, this handler should be chosen as a function of the requirements of the program to be loaded. The same version of the handler should be used for DECtape assignments when a handler is required for more than one .DAT slot.

[3]Reassign these .DAT slots to appropriate devices when using FOCAL. Use DTE if DECtape I/O is desired in 8K.

[4]Reassign this .DAT slot to VPA if EDITVP is to be used.



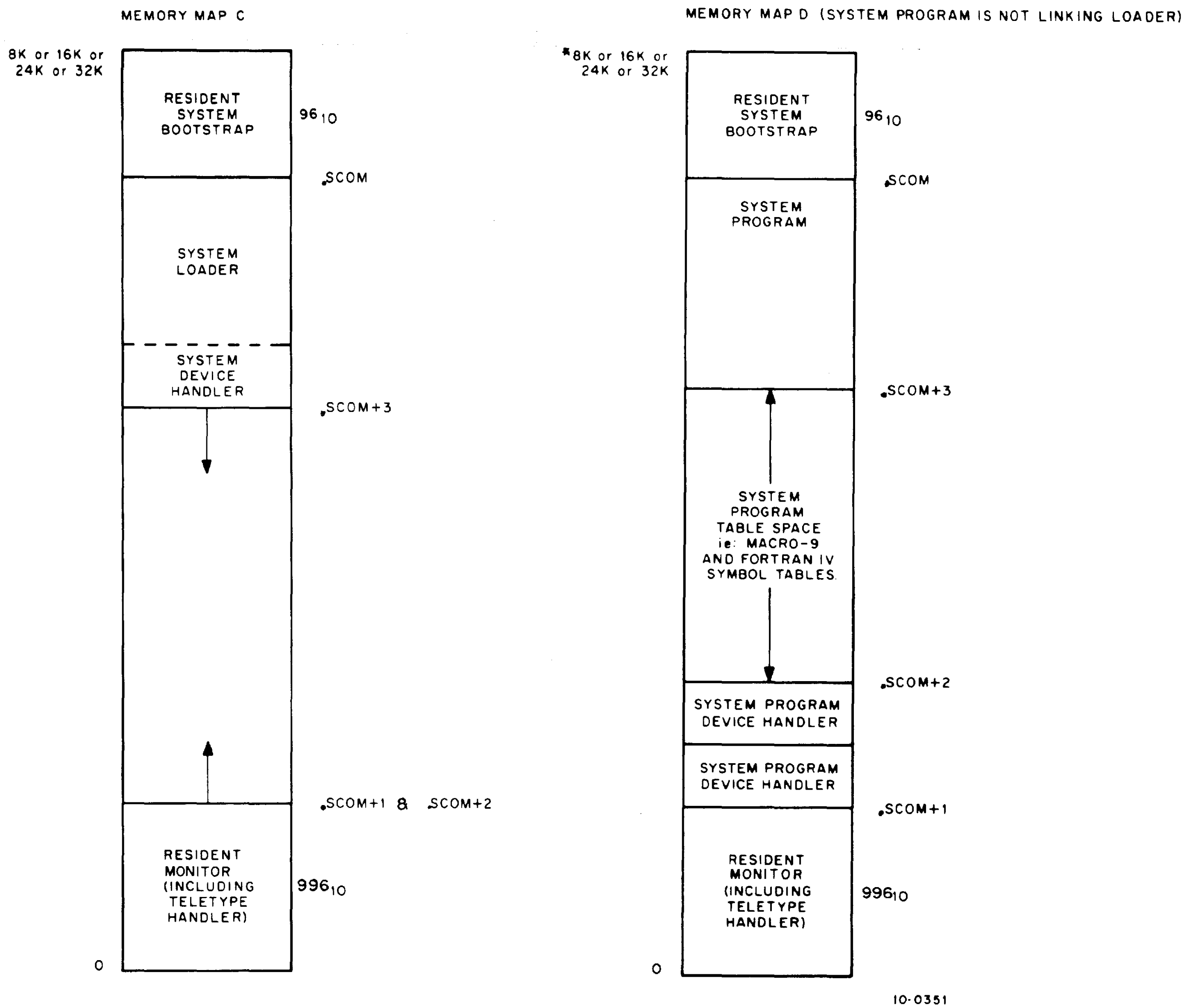
The System Bootstrap is loaded via the paper tape reader in HRM mode.

The System Bootstrap loads the ADVANCED Monitor (resident and nonresident) from the system device.

Figure 4-1 ADVANCED Monitor System Memory Maps

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 4-1 for memory maps of programs loaded in the ADVANCED Monitor System.

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



The Monitor loads the System Loader and the system device handler from the system device via the Bootstrap.

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.

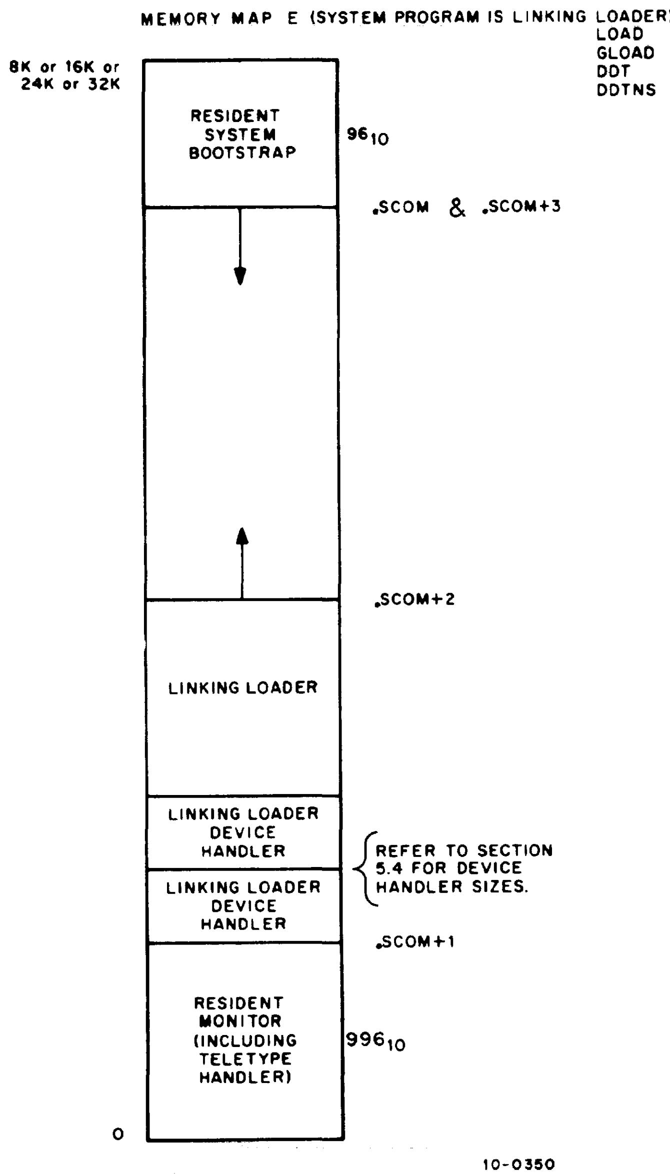
The System Loader learns which I/O handlers are required by the requested system program from its table of .IODEV info for system programs, loads the handlers relocatably just above the resident Monitor*, and then modifies the System Bootstrap to bring in the system program in just below the Bootstrap.

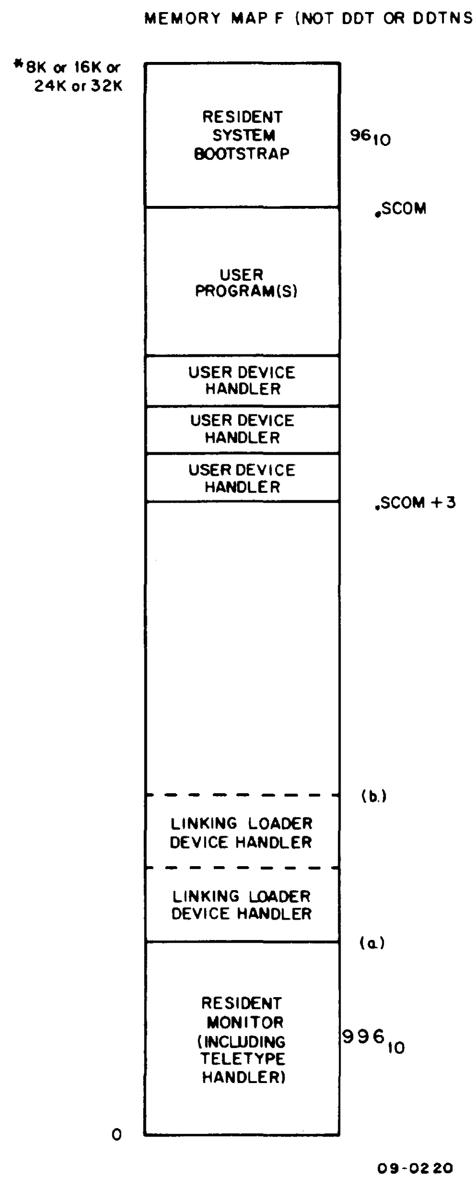
.EXIT from the system program takes the process back to Memory Map B where the system bootstrap reinitializes the ADVANCED Monitor.

Refer to Section 5.4 for the sizes of the device handlers that may be associated with the .DAT slots used by the system program.

*For 12K, 20K, or 28K systems, handlers are loaded up into the extra 4K, then up from .SCOM+2 if necessary. .SCOM +20 is set to the first address above the I/O handlers in the extra 4K with the sign bit = 1.

Figure 4-1 ADVANCED 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+3*, builds the loader (GLOBAL) and DDT (if DDT) symbol tables up from .SCOM+2. DDT symbol table will not be built if a LOAD, GLOAD, or DDTNS load.

.EXIT from user program takes the process back to Memory Map B where the system bootstrap reinitializes the ADVANCED Monitor.

Refer to Section 5.4 for sizes of device handlers.

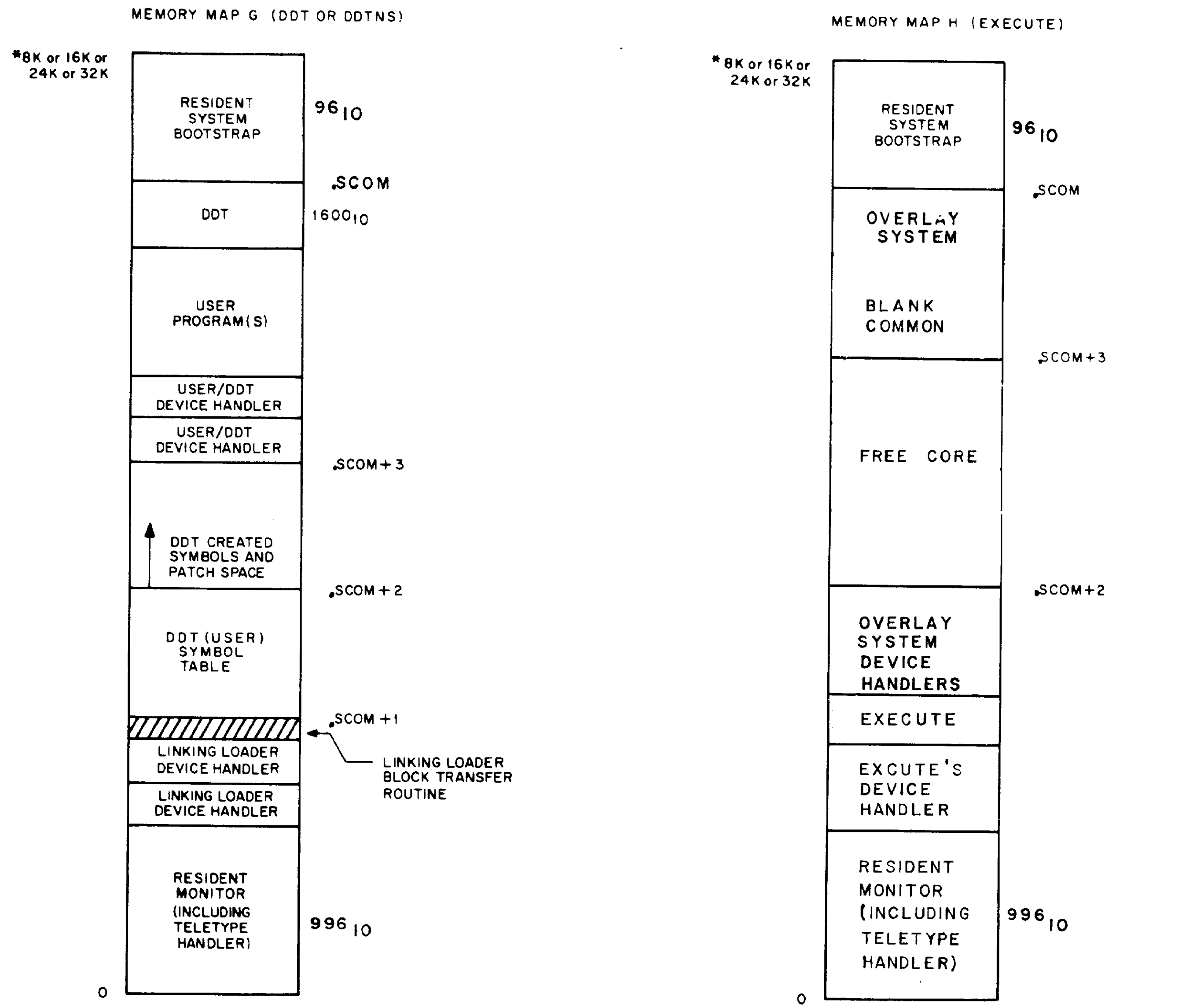
.SCOM+1 and .SCOM+2 both point to one of two places and non-BLOCK DATA COMMON (FORTRAN IV or MACRO) output make use of core as low as they point.

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.

*In 12K, 20K, or 28K systems DDT, user programs and user I/O handlers are first loaded down from the top of the extra 4K. .SCOM+2 and .SCOM+3 bracket free core. .SCOM+20 points to the first free cell below the routines in the extra 4K with SCOM+20, bit 0 = 1.

Figure 4-1 ADVANCED Monitor System Memory Maps (Cont.)



.EXIT fromthe user program takes the process back to Memory Map B where the system bootstrap reinitializes the Monitor.

Refer to Section 5.4 for sizes of device handlers.

Non-BLOCK DATA COMMON (FORTRAN IV or MACRO output) may make use of core as low as the DDT symbol table. (There is no DDT symbol table if a DDTNS load.) 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.

If none of these handlers is used by user program, these handlers are overlaid also by the DDT symbol table.

*In 12K, 20K, or 28K systems DDT, user programs and user I/O handlers are first loaded down from the top of the extra 4K, then down from .SCOM+3 if necessary. When the loading process is complete, .SCOM+2 and .SCOM+3 bracket free core. .SCOM+20 points to the first free cell below the routines in the extra 4K with .SCOM+20, bit 0=1

The System Loader (.SYSLD) opens the XCT file to determine the handlers required by the Overlay System. It then loads the handler required for EXECUTE and then EXECUTE itself, followed by the Overlay System's device handlers. Control is then passed to EXECUTE which loads the Link Table, the Resident Code, and the main program, to which control then passes. .EXIT from the Overlay System initiates a return to the Monitor via Memory Map B.

**If an extra 4K of core is to be used by the Overlay System (e.g., 12K, 20K, etc.), EXECUTE will attempt to load as much of the Overlay System as possible from the top of the extra 4K down and the remainder from .SCOM down.

Figure 4-1 ADVANCED Monitor System Memory Maps (Cont.)

The System Loader is used to bring in most 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 ADVANCED 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 PDP-15/20 Users Guide.

4.4.5 Error Detection and Handling

Comprehensive error checking is provided by the ADVANCED 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 (CTRL P) or user program, dump core (CTRL Q), or return control to the System Bootstrap for re-initialization of the Monitor (see Section 4.3.2.11). 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.

4.5 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 teleprinter. Monitor commands that are peculiar to the Batch Processor include the following:

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

When using the card reader, the following special characters are punched as shown:

Back Arrow (←) = 0-5-8 (029) or 8-2 (026)

ALT MODE = 12-1-8 (029 or 026)

$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 teleprinter.
$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.

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 ADVANCED 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 an ⓧ.

```
KM15 V5A

$X4K ON

$BATCH PR

KM15 V5A

$$JOB TEST BATCH

$PIP

PIP V13A

>N DT1

>T DT1 TEST SRC ← PR
$DATA
$END

>$JOB

KM15 V5A

$R F4

 .DAT   DEVICE  USE

  -13    PPC0    OUTPUT
  -12    TTA0    LISTING
  -11    PR*0    INPUT
  -3     TTA0    CONTROL AND ERROR MES
  -2     PR*0    COMMAND STRING

$ASSIGN DT1 -11/DT2 -13

$F4

F4  V10A
>S,L,B←TEST ⓧ
```

This command causes all subsequent commands to come from the paper tape reader.

The entire program to be compiled below appears on the paper tape between $DATA and $END.

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

FORTRAN program to list numbers from 1 through 10.

```
KM15  V5A

$$JOB

$GLOAD


LOADER  V9A
>P←TEST (x)
P TEST    27734
P BCDIO   24676
P STOP    24663
P SPMSG   24570
P FIOPS   24030
P OTSER   23734
P RELEAE  22672
P .CB     22652
```

Program execution begins here.

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

STOP  012345

KM15  V5A

$$JOB

$$EXIT


KM15  V5A

$
```

Control is returned to Teletype at this point

4.6 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.

4.6.1 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. A .CLOSE terminates recording with a simulated end-of-file consisting of two words: 001005, 776773. Note that the simulated end-of-file is identical whether executing a .CLOSE in a file-oriented or a non-file-oriented environment.

Because braking on DECtape allows for tape roll, staggered recording of blocks is employed in the 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 ($0\text{-}1077_8$).

4.6.2 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 ADVANCED Monitor. A fresh directory may be recorded via the (N)ewdir command in the ADVANCED Monitor or PIP or by the N or S switch in PIP.

The directory of all DECtapes except system tapes occupies all 400_8 words of block 100_8. It is divided into two sections: (1) a 40_8word 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. Words $0\text{-}37_8$ constitute the System Tape Directory Bit Map and words $40\text{-}177_8$ 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 the Monitor, PIP, and SGEN.

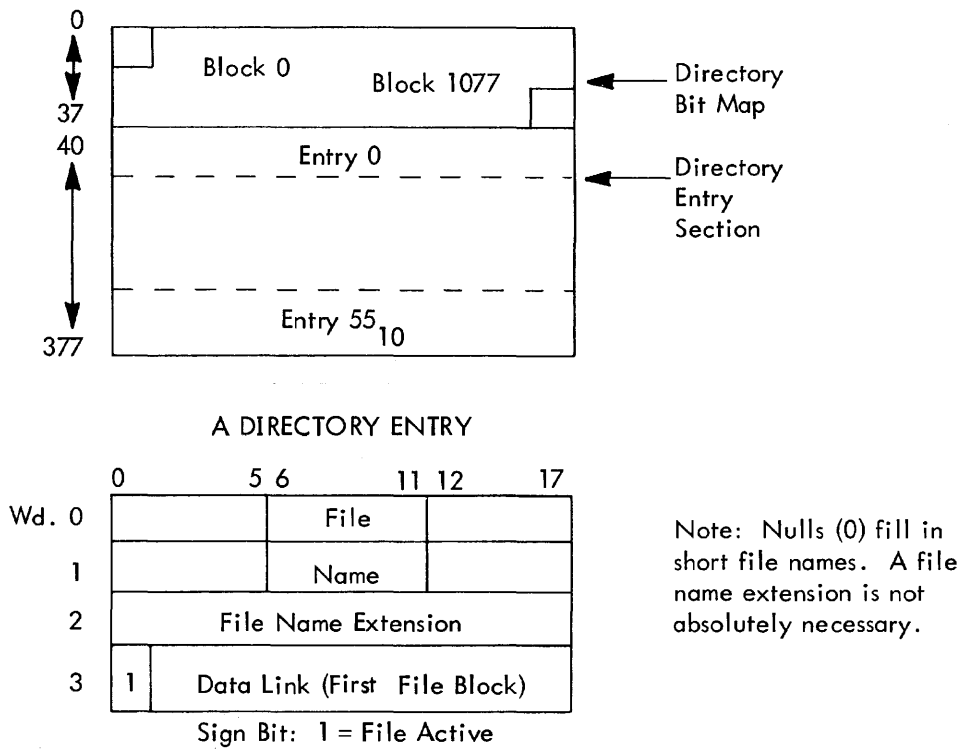


Figure 4-2 DECtape Directory

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 on the .CLOSE of the new file.

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. The last word of each data block recorded contains a data link or pointer to the next block in the file. 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.

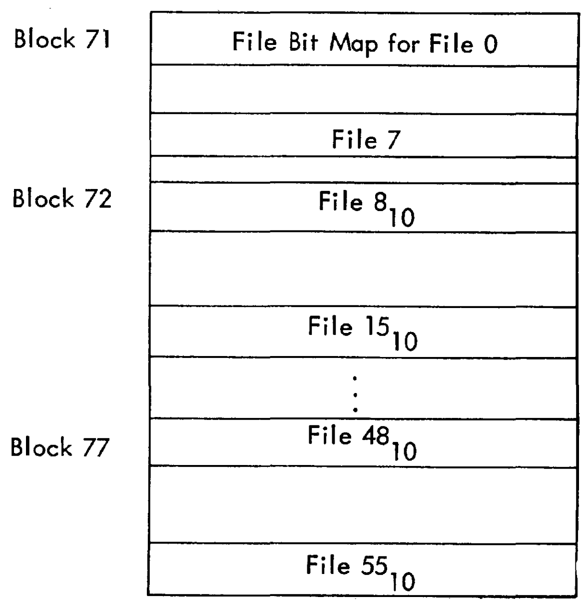


Figure 4-3 DECtape File Bit Map Blocks

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 Alphanumeric, and Image binary and (2) dump mode data - Dump 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 available free block.

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.

4.7 RF15 DECDISK

4.7.1 General Description

The PDP-15/20 DECdisk System accommodates up to four RS15 DECdisk Platters which communicate with the device handlers via the RF15 controller. Each platter is treated as two logical units, each of which contains 512_{10} blocks. Each block contains 256_{10} words.

4.7.2 File Structure

The file structure of the RF15 DECdisk is the same as that used for DECtape (see 4.6 above) with the following exceptions:

a. Each DECdisk unit is 64_{10} blocks shorter than a DECtape. Since the handlers are similar in structure to DECtape, they will always indicate that blocks 512_{10} through 576_{10} are occupied; where, in fact, they do not exist.

b. Reading and writing is performed in one direction only.

4.7.3 Disk File Protection

Selected areas of a disk platter may be write-protected by use of the switches provided on each RS15 Platter control panel. Each switch protects $40{,}000_8$ words.

4.8 MAGNETIC TAPE

The ADVANCED Software provides for industry-compatible magnetic tape as either a file-structured or non-file-structured medium. 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.

There are a number of major differences between magnetic tape and other mass-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 (i.e., .TRAN) 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.

4.8.1 File Organization

The device handlers for magnetic tape allow the use of either a file structured or a non-file structured mode. Handler MTA. enables both reading and writing in the file structured mode; handler MTC. (a subset of MTA.) permits reading in the file structured mode only. Handler MTF. enables reading and writing in the non-file structured mode. The legal functions and data modes for each of the magnetic tape handlers are described in paragraph 5.4.6.

4.8.1.1 Non-File-Structured Data Recording (MTF.) - 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. Each .WRITE (or .READ) issued results, in general, in the transfer of exactly one physical record to (or from) tape.

4.8.1.2 File-Structured Data Recording (MTA., MTC.) - 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.

4.8.1.3 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
00	User-File Header Label
10	User-File Trailer Label
20	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.

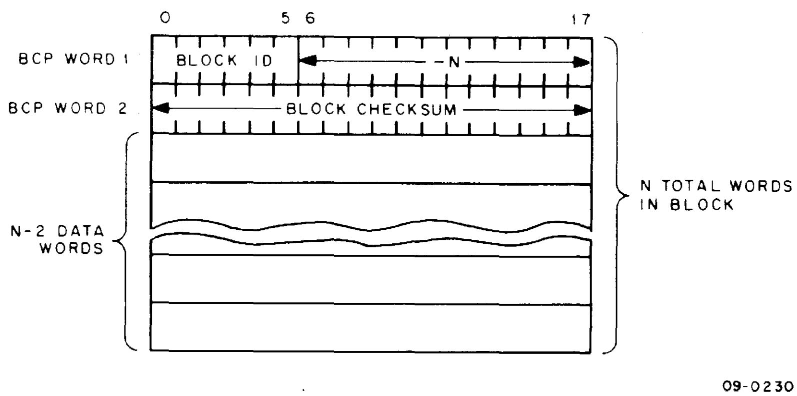


Figure 4-4 Block Format, File-Structured Mode

4.8.2 File Identification and Location

One of the main file-manipulation functions of MTA. and MTC. 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.

4.8.2.1 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 characteristics:

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 specifies 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 4-5.

4.8.2.2 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

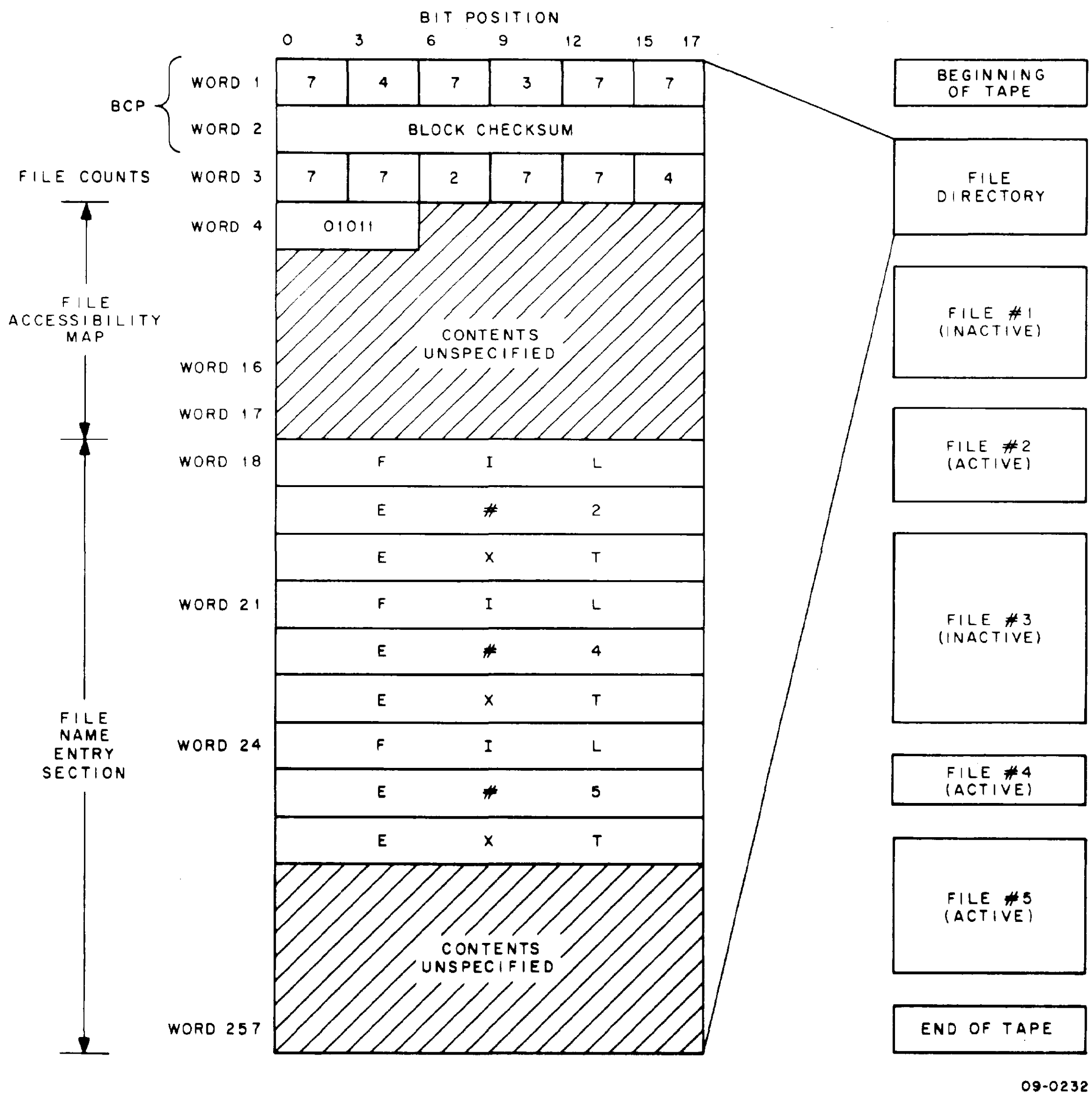


Figure 4-5a. Format of the File Directory Data Block, showing relationship of active and inactive files to file name entries and to Accessibility Map.

Figure 4-5b. Format of file-structured tape, showing directory block and data files.

recorded data block of the file. Each label is 27_{10} words in length. Label format is shown in Figure 4-6.

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.

4.8.2.3 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 .SEEKs. 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.

4.8.3 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 I/O request must be one which requires an implicit .WAIT to ensure successful completion. The handler in processing requests in continuous mode depends on receiving control at the CAL level in order to respond to I/O errors. In addition, an explicit .WAIT command should not be issued (see examples 1 and 2 given below). The functions for which continuous operation is attempted include only the following:

1. .MTAPE	3. .WRITE
2. .READ	4. .TRAN

d. 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 following examples illustrate this principle.

Example 1: Successful Continued Operation

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

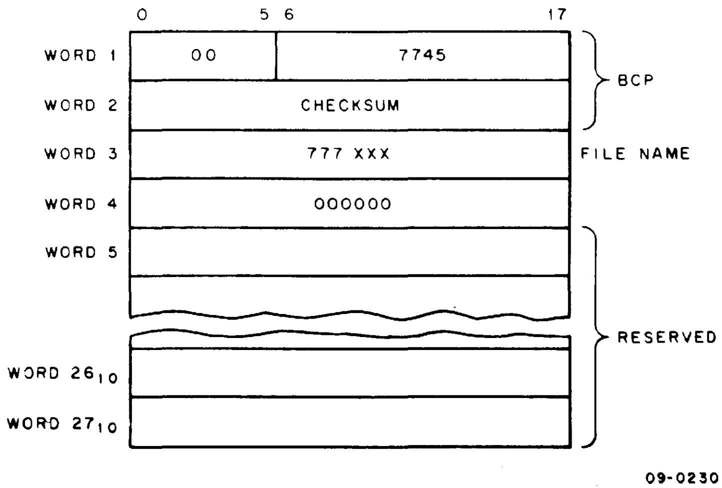


Figure 4-6a. User-File Header Label Format

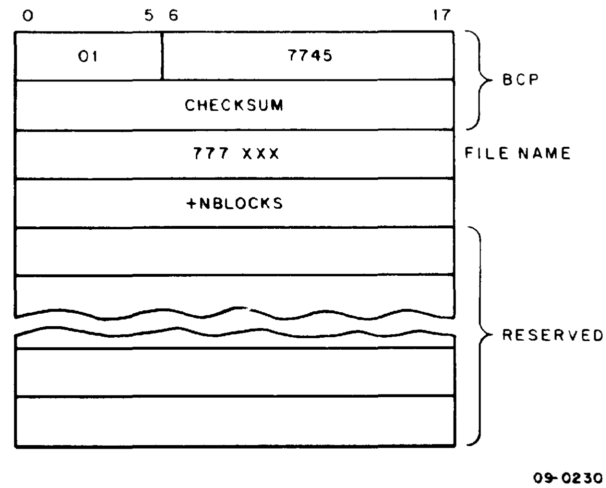


Figure 4-6b. User-File Trailer Label Format

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. Each .TRAN request causes an implicit .WAIT until its operation is completed.

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
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.

4.8.4 Storage Retrieval on File-Structured Magnetic Tape

The use of a file accessibility map as well as block identifiers in MAGtape file directories makes it almost impossible to retrieve the area of a deleted file from a magnetic tape. The execution of the deletion command (i.e., .DLETE) removes the name of the object file from the file directory; however, the file accessibility map will continue to reflect the presence of the file on the tape.

The only circumstance under which a file area may be retrieved is when the deleted file is also the last file, physically, on the tape. Under these conditions, the handler can effect retrieval of the area occupied by the deleted file when the next .ENTER- .WRITE- .CLOSE sequence is executed.

4.8.5 Magnetic Tape Dump (MTDUMP) Utility Program

The MTDUMP program provides the user who employs magnetic tape as a storage medium with the ability to view and manipulate any named portion (i.e., file) of a tape. Some of the features provided by MTDUMP are:

a. Files may be output (dumped) onto any system device in any of four possible formats.

b. Comments may be inserted into any output file.

c. Files may be copied onto another tape.

A complete description of the features and operation of the MTDUMP program is given in the PDP-15 Utility Manual.

CHAPTER 5

I/O DEVICE HANDLERS

This chapter contains information essential to a good understanding of the operation and use of the ADVANCED Monitor I/O device handlers. 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 are included in this chapter.

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

5.1.1 I/O Device Handlers

All communications between user programs and I/O device handlers are made via CAL instructions (see Chapter 3) followed by an argument list. The CAL Handler in the Monitor (see Figure 5-1) 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 (see Figures 5-2 and 5-3), 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, Bank/Page 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 when control is returned to the user.

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 non-reentrant procedures before debreaking (DBK) from level 4. This permits the handler to receive reentrant calls from software levels higher than the priority of the program that contained this call. Device handlers which do not contain reentrant procedures (including all IOPS handlers), may avoid system failure caused by inadvertent reentries 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

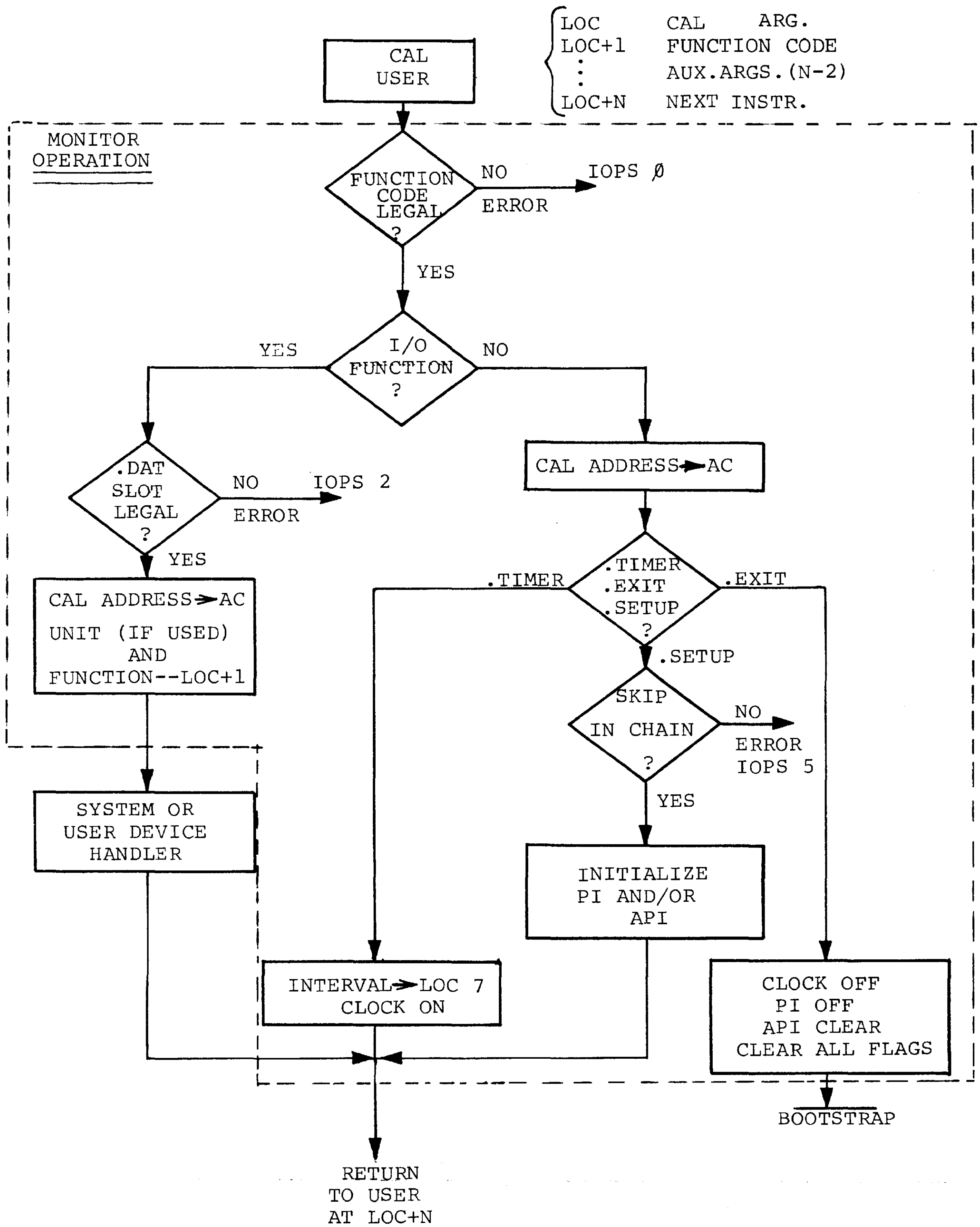


Figure 5-1, CAL Handler Functions.

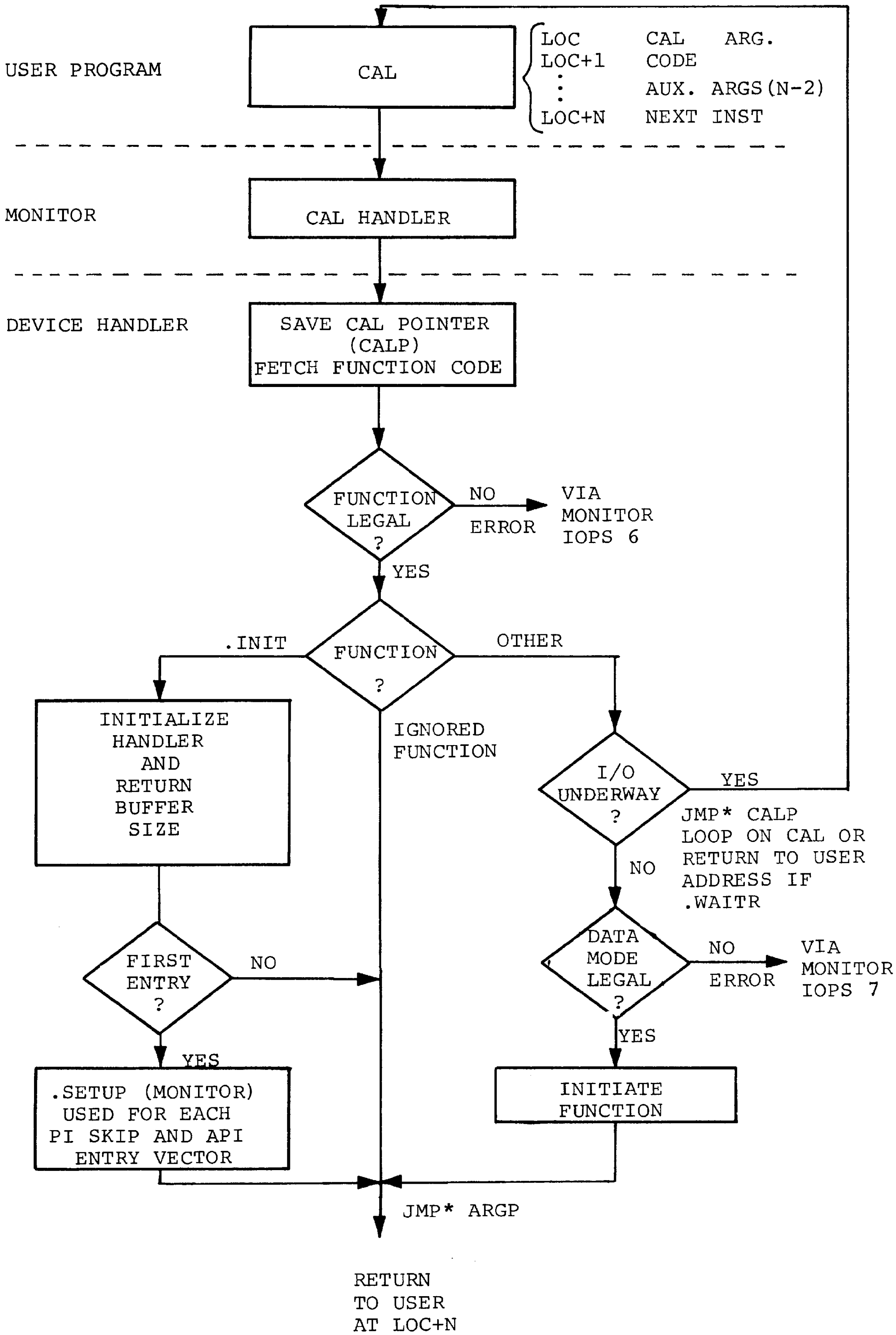


Figure 5-2, CAL Entry To Device Handler.

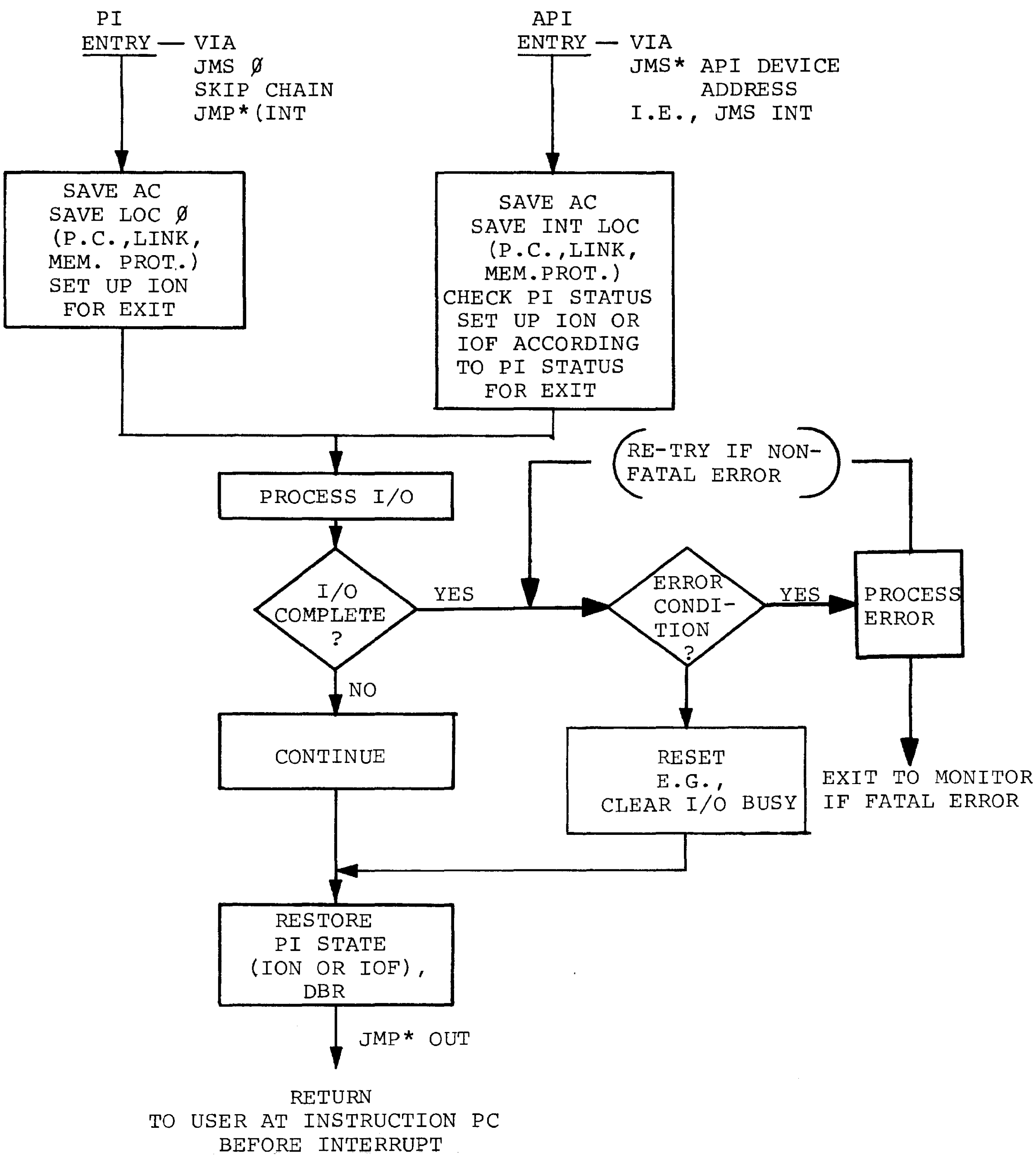


Figure 5-3, PI and API Entries to Device Handlers control to the user, allowing debreak from level 4 and restoring the conditions of the Link, Bank/Page mode, and Memory Protect. Any IOT's issued at the CAL level (level 4 if API present, mainstream if no API) should be executed immediately before the

```
DBR
JMP*
```

exit sequence to ensure that the exit takes place before the interrupt from the issued IOT occurs.

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 sequence 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 CALd from level 4 must know that if a debreak (DBR or DBK) is issued, control will return to the calling program (which had been at level 4) at a level lower than level 4.

5.1.1.1 Setting Up the Skip Chain and API (Hardware) Channel Registers - When the Monitor is loaded, the Program Interrupt (PI) skip chain and the Automatic Priority Interrupt (API) channels are set up to handle the TTY keyboard and printer 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:

```
SKPDTA                    /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.
SKPTTI                    /Skip if teleprinter flag.
SKP
JMP TELINT                /To teleprinter interrupt handler.
    .
    .
    .
```

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

When a device handler is called for the first time via an .INIT user program command, it must call a Monitor routine (.SETUP) to set up its ship 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 5.1.3 for standard channel assign-
                /ments), 0 if device 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 to allow for either API or PI interrupts. Users with PI-only systems may exclude code from DEVINT-1 through DVSTON-1, in which case DEVINT=DEVPIC is required. Users who always expect to use API may exclude code from DEVPIC through DEVINT-1, in which case DEVINT 0 should be substituted.

```
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     DEVAC           /RESTORE AC
DVSWCH  XX                      /ION OR IOF
        DBR                     /DEBREAK AND RESTORE CONDITIONS

        JMP*    DEVOUT          /OF LINK, EX.MODE AND MEM.PROT.
```

Since the Index, Autoincrement, 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.

Refer to Paragraph 5.2 for the method of incorporating new handlers and associated skip chain entries into the Monitor.

5.1.2 API Software Level Handlers

The information presented in the following paragraphs assumes that the reader is familiar with the system input/output considerations described in the PDP-15 User's Handbook Vol 1.

5.1.2.1 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 .SCOM is equal to absolute location 000100 and .SCOM+12 through .SCOM+15 (000112 through 000115) each contains 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,BANK/PAGE MODE,MEM.PROT.
        DAC        SAV5AC       /SAVE AC
        /SAVE INDEX, AUTOINCREMENT AND EAE REGISTERS
        /IF LEVEL 5 ROUTINES
        /USE THEM AND LOWER LEVEL
        /ROUTINES ALSO USE THEM
        /SAVE MQ AND STEP COUNTER
        /IF SYSTEM HAS EAE AND IT
        /IS USED AT DIFFERENT LEVELS.
                .
                .
                .
        /RESTORE SAVED REGISTERS.
        DBR                     /DEBREAK FROM LEVEL 5
        JMP*       LV5INT       /AND RESTORE L, BANK/PAGE MODE,MEM.PROT.
```

5.1.2.2 Queueing - High priority/high data rate/short access routines cannot perform complex calculations based on unusual conditions without holding off further data input. To perform the calculations, the high priority program segment must initiate a lower priority (interruptable) segment to perform the calculations. Since many data handling routines would generally 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 5-4, depicts the structure of a software handler involved with queued requests.

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.
>
> 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 condition 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 (see 5.1.1.1).

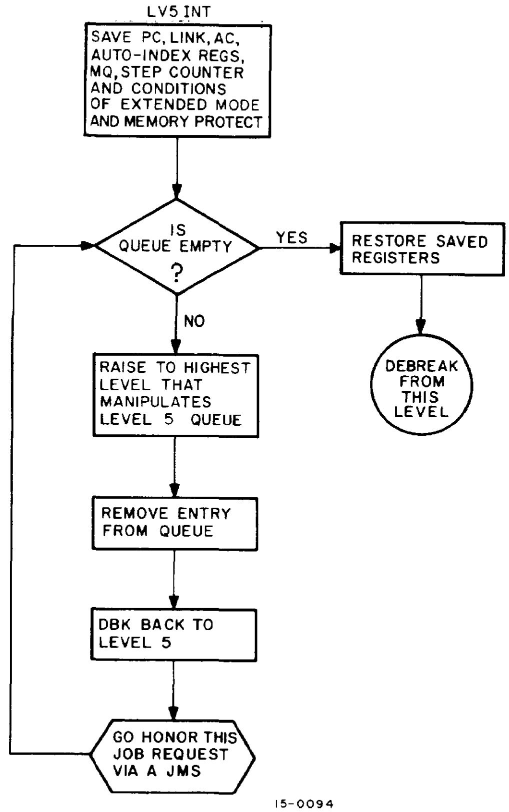


Figure 5-4 Structure of API Software Level Handler

5.1.3 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	TC02D	1	44
5	MAGtape	TC59D	1	45
6	RESERVED	--	--	46
7	RESERVED	--	--	47
10	Paper Tape Reader	PC15	2	50
11	Clock Overflow	KW15	3	51
12	Power Fail	KF15	0	52
13	Parity	MP15	0	53
14	Display (LP flag)	VP15A	2	54
15	Card Reader	CR03B	2	55
16	Line Printer	LP15F/LP15C	2	56
17	A/D	AF01/ADC1/9	0	57
20	Interprocessor Buffer	DB99/DB98	3	60
21	RESERVED	--	--	61
22	Data Phone	DP09A	2	62
23	DECdisk	RF15	1	63
24	DISK Pack	RP15	1	64
25	Plotter	XY15	2	65
32	Multi-Station TTY Control	LT19A (Teleprinter)	2	74
33	Multi-Station TTY Control	LT19A (Keyboard)	2	75
34	DECtape (DCH Channel 36)	TC02[1]	1	76
35	Dataphone	DP09A[1]	2	77

[1] Channel allocated for systems with more than one of the above options.

5.2 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 ADVANCED Monitor System.

The ADVANCED 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.

It is assumed that the user is familiar with Section 5.1.1 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 Bank/Page 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 or at a hardware level for interrupt servicing (if API), and restore the status of the Link, Bank/Page 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. This handler is used to read data from the ADC and store it in the user's line buffer.

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 the Reference Manual, 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.) - Since 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 four of the macros.

(1) .INIT would be used to set up the associated API channel register and the interrupt skip IOT sequence in the Program Interrupt (PIC) ship 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 in Section 5.1.3.

(2) .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.

(3) .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.

(4) .WAITR detects the availability of the user's buffer area as in (3) above. If the buffer is not available, control is returned to a user specified address, which allows other processing to continue.

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 uses 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, and .WAITR were implemented into the ADC handler (Example B) under the subroutine names ADINIT, ADREAD, ADWAIT (.WAIT and .WAITR).

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 in some cases, depending upon the number of arguments following the CAL (See 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, the Monitor must then be modified to recognize the new handler. This is accomplished by the use of the System Generator Program (SGEN) described in the Utility Programs Manual.

Once the system has been generated, the system program UPDATE (refer to the Utility Programs Manual (DEC-15-YWZA-D) 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-15 system.

5.2.1 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 PI skip chain and/or the API channel register. Paragraph 5.1.3 of this manual shows the standard device/API associations.
Part 4	Shows the API and PI 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.

5.2.2 Example A, Skeleton I/O Device Handler

PART 1

```
/SKELETON I/O DEVICE 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 NO IF APPLICABLE
        ISZ     DVARGP                /POINTS TO CAL+2
        TAD     (JMP DSPCH
        DAC     DSPCH                 /DISPATCH WITH
```

PART 2

```
DSPCH   XX                            /MODIFIED JUMP
        JMP     DVINIT                /1 = .INIT
        JMP     DVFSAT                /2 = .FSTAT, .DLETE, .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

/ILLEGAL FUNCTIONS IN ABOVE TABLE CODED AS:
        JMP    DEVERR6
/FUNCTION CODE ERROR
DVERR6  LAW     6                     /ERROR CODE 6
        JMP*    (.MED+1               /TO MONITOR
```

PART 2 (continued)

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

/DEVICE NOT READY
DVERR4  LAC     (RETURN         /RETURN (ADDRESS IN HANDLER)
        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 LEVLE 4
        JMP*    DVARGP          /RETURN AFTER CAL AND
                                /ARGUMENT STRING
```

PART 3

```
/THE DVINIT ROUTINE MUST INCLUDE
/A .SETUP CALLING SEQUENCE FOR
/EACH FLAG CONNECTED TO API
/AND/OR PI A(AT SGEN TIME).
/THE SETUP CALLING SEQUENCE IS:

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

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

```
/THIS SPACE MAY BE USED FOR I/O SUBROUTINES
```

PART 4

```
/INTERRUPT HANDLER FOR API OR PI
DVPIC   DAC     DEVAC           /SAVE AC
        LAC*    (0              /SAVE:PC, LINK, BANK/PAGE MODE
        DAC     DVOUT           /AND MEMORY PROTECT
        LAC     DEVION          /FORCE ION AT DISMISSAL
        JMP     DVSTON
DVINT   JMP     DEVPIC          /PI ENTRY
        DAC     DEVAC           /API ENTRY; SAVE AC
        LAC     DEVINT          /SAVE: PC, LINK, BANK/PAGE MODE
        DAC     DEVOUT          /MEMORY PROTECT
        IORS                    /CHECK STATUS OF PI
        SMA!CLA                 /FOR RESTORATION AT
        LAW     17740           /DISMISSAL
        TAD     DEVION
DVSTON  DAC     DEVSWCH
        DEVCF                   /IOT TO CLEAR FLAG
DEVION  ION                     /ENABLE PI
```

PART 5

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

        IOF                     /DISABLE PI TO INSURE
        DEVIOT                  /DIMISSAL BEFORE INTERRUPT
                                /FROM THIS IOT OCCURS
```

PART 6

```
/INTERRUPT HANDLER DISMISS ROUTE
DVDISM  LAC     DEVAC               /RESTORE AC
DVSWCH  XX                          /ION OR IOF
        DBR                         /DEBREAK AND RESTORE
        JMP*    DEVOUT              /LINK, BANK/PAGE MODE, MEMORY
                                    /PROTECT

/IF THE HANDLER USES THE AUTOINCREMENT , INDEX
/OR EAE REGISTERS, THEIR CONTENTS
/SHOULD BE SAVED AND RESTORED. FUNCTIONS
/POSSIBLY IGNORED SHOULD CONTAIN
/PROPER INDEXING TO BYPASS
/CAL ARGUMENT STRING
/
```

PART 7

```
/CODE TO BYPASS IGNORED FUNCTIONS
/

DVIGN2  ISZ     DVARGP              /BYPASS FILE POINTER
        JMP     DVCK
```

5.2.3 Example B, Special Device Handler for AF01B A/D Converter.

```
PAGE    1      ADC.    SRC

                                /ADC HANDLER
                                /
                     701301 A   ADSF=701301        /SKIP IF CONVERSION FLAG IS SET
                     701304 A   ADSC=701304        /SELECT AND CONVERT (ADC FLAG IS CLEARED
                                                   /AND A CONVERSION IS INITIALISED)
                     701312 A   ADRB=701312        /READ CONVERTER BUFFER INTO AC AND CLEAR FLAG
                                /
                                          .GLOBL   ADC.
                     440000 A   IDX=ISZ
                     000003 A   .MED=3             /MED (MONITOR ERROR DIAGNOSTIC)
                                /
           00000 R 040151 R     ADC.      DAC      ADCALP   /SAVE CAL POINTER
           00001 R 040152 R               DAC      ADARGP   /AND ARGUMENT POINTER
           00002 R 440152 R               IDX      ADARGP   /POINTS TO FUNCTION CODE
           00003 R 220152 R               LAC*     ADARGP   /GET CODE
           00004 R 440152 R               IDX      ADARGP   /POINTS TO CAL + 2
           00005 R 340155 R               TAD      (JMP DSPCH
           00006 R 040007 R               DAC      DSPCH    /DISPATCH WITH
           00007 R 740040 A     DSPCH     XX                /MODIFIED JUMP
           00010 R 600027 R               JMP      ADINIT   /1=.INIT
           00011 R 600074 R               JMP      ADIGN2   /2=.FSTAT,.DLETE,.RENAM
           00012 R 600074 R               JMP      ADIGN2   /3=.SEEK
           00013 R 600023 R               JMP      ADERR6   /4=.ENTER
           00014 R 600023 R               JMP      ADERR6   /5=.CLEAR
           00015 R 600075 R               JMP      ADIGN1   /6=.CLOSE
           00016 R 600075 R               JMP      ADIGN1   /7=.MTAPE
           00017 R 600051 R               JMP      ADREAD   /10=.READ
           00020 R 600023 R               JMP      ADERR6   /11=.WRITE
           00021 R 600044 R               JMP      ADWAIT   /12=.WAIT
           00022 R 600023 R               JMP      ADERR6   /13=.TRAN
                                /
                                /ILLEGAL FUNCTIONS IN ABOVE TABLE CODED AS
                                /         JMP      ADERR6
                                          .EJECT
```

```
                          /
                          /FUNCTION CODE ERROR
                          /
00023 R 760006 A    ADERR6  LAW     6        /ERROR CODE 6
00024 R 620156 R            JMP*    (.MED+1  /TO MONITOR
                          /DATA MODE ERROR
00025 R 760007 A    ADERR7  LAW     7        /ERROR CODE 7
00026 R 620156 R            JMP*    (.MED+1  /TO MONITOR
                          /THE ADINT ROUTINE MUST INCLUDE A .SETUP
                          /FOR EACH FLAG ASSOCIATED WITH THE DEVICE
                          /
00027 R 440152 R    ADINIT  IDX     ADARGP   /IDX TO RETURN BUFF SIZE
                            .DEC
00030 R 200157 R            LAC     (256     /STANDARD BUFFER SIZE (DECIMAL)
                            .OCT
00031 R 060152 R            DAC*    ADARGP   /RETURN IT TO USER
00032 R 440152 R            IDX     ADARGP
00033 R 000057 A    ADCMOD  CAL     57       /57=API CHANNEL
00034 R 000016 A    ADCKSM  16               /.SETUP IOPS FUNCTION CODE
00035 R 701301 A    ADCBP   ADSF             /ADC SKIP IOT
00036 R 000104 R    ADLBHP  ADCINT           /ADDR. OF INTERRUPT
00037 R 200041 R    ADUND   LAC     .+2      /SET-UP ONCE ONLY
00040 R 040033 R    ADWC    DAC     ADCMOD   /SKIP SET-UP CODE IF MORE
00041 R 600042 R    ADWPCT  JMP     ADSTOP   /.INITS ARE DONE
                          /
                          /STOP ADC ROUTINE CLEARS I/O UNDERWAY SWITCH
                          /
00042 R 140037 R    ADSTOP  DZM     ADUND
00043 R 600075 R            JMP     ADIGN1   /RETURN
                          /
                          /THE PREVIOUS TAGS IN THE CAL AREA ARE USED FOR
                          /STORAGE DURING THE ACTUAL .READ FUNCTION
                          /
                          /ADCKSM IS FOR STORING THE CHECKSUM
                          /ADCBP IS THE CURRENT BUFFER POINTER
                          /ADLBHP IS THE LINE BUFFER HEADER POINTER
                          /ADUND IS FOR DEVICE UNDERWAY SWITCH
                          /ADWC IS USED AS THE COUNTER
                          /ADWPCT IS USED TO STORE CURRENT WORD COUNT
                          /
                                    .EJECT
```

```
ØØØ44 R 2ØØØ37 R     ADWAIT  LAC      ADUND
ØØØ45 R 7412ØØ A             SNA
ØØØ46 R 6ØØØ75 R             JMP      ADIGN1
                     /I/O UNDERWAY LOOP
ØØØ47 R 7Ø3344 A     ADBUSY  DBR
ØØØ5Ø R 62Ø151 R             JMP*     ADCALP
                     /
                     /
ØØØ51 R 2ØØØ37 R     ADREAD  LAC      ADUND     /CHECK TO SEE IF I/O IS UNDERWAY
ØØØ52 R 7402Ø1 A             SZA!CMA            /IF NOT SET IT WITH -1
ØØØ53 R 6ØØØ47 R             JMP      ADBUSY    /IT WAS SET,GO BACK TO CAL
ØØØ54 R Ø4ØØ37 R             DAC      ADUND     /SET IT
ØØØ55 R 22Ø151 R             LAC*     ADCALP    /LOOK AT MODE
ØØØ56 R 50Ø16Ø R             AND      (7ØØØ     /BITS 6-8 ONLY
ØØØ57 R 7402ØØ A             SZA                /IOPS BINARY?
ØØØ6Ø R 6ØØØ25 R             JMP      ADERR7    /NO, ERROR
ØØØ61 R 22Ø152 R             LAC*     ADARGP    /GET LINE BUFFER HEADER POINTER
ØØØ62 R Ø4ØØ35 R             DAC      ADCBP     /STORE IT
ØØØ63 R Ø4ØØ36 R             DAC      ADLBHP    /ALSO STORE IT FOR LATER HEADER
ØØØ64 R 44Ø152 R             IDX      ADARGP    /INCREMENT ARG. POINTER
ØØØ65 R 22Ø152 R             LAC*     ADARGP    /GET -L,B,W,C(2'S COMP)
ØØØ66 R Ø4ØØ4Ø R             DAC      ADWC      /STORE IT IN WORD COUNTER
ØØØ67 R 14ØØ41 R             DZM      ADWPCT    /ZERO WORD COUNT REG.
ØØØ7Ø R 14ØØ34 R             DZM      ADCKSM    /ZERO CHECKSUM REG.
ØØØ71 R 44ØØ35 R             IDX      ADCBP     /GET PAST HEADER PAIR
ØØØ72 R 44ØØ35 R             IDX      ADCBP     /NOW POINTING AT BEGINNING OF
                                                /BUFFER
ØØØ73 R 7Ø13Ø4 A             ADSC               /START UP DEVICE
ØØØ74 R 44Ø152 R     ADIGN2  IDX      ADARGP    /INCR. FOR EXIT
ØØØ75 R 7Ø3344 A     ADIGN1  DBR                /BREAK FROM LEVEL 4
ØØØ76 R 62Ø152 R             JMP*     ADARGP    /RETURN AFTER CAL
                     /INTERRUPT HANDLER FOR API OR PIC
                     /
ØØØ77 R Ø4Ø154 R     ADCPIC  DAC      ADCAC     /SAVE AC
ØØ1ØØ R 22Ø161 R             LAC*     (Ø)       /SAVE PC,LINK,EX. MODE
ØØ1Ø1 R Ø4Ø153 R             DAC      ADCOUT    /MEM.PROT.
ØØ1Ø2 R 2ØØ12Ø R             LAC      ADCION    /FORCE ION AT DISMISSAL
ØØ1Ø3 R 6ØØ116 R             JMP      ADSION
                             .EJECT
```

```
00104 R 600077 R    ADCINT  JMP     ADCPIC  /PIC ENTRY
00105 R 040154 R            DAC     ADCAC   /API ENTRY,SAVE AC
00106 R 200104 R            LAC     ADCINT  /SAVE PC,LINK,EX,MODE
00107 R 040153 R            DAC     ADCOUT  /MEM,PROT
00110 R 200162 R            LAC     (JMP ADCPIC /RESTORE PIC ENTRY BECAUSE API
00111 R 040104 R            DAC     ADCINT  /ENTRY IS A JMS, NOT A JUMP
00112 R 700314 A            IORS            /CHECK FOR PIC
00113 R 750100 A            SMA!CLA         /FOR RESTORATION AT
00114 R 777740 A            LAW     17740   /DISMISSAL (IOF-ION)
00115 R 340120 R            TAD     ADCION  /+ION
00116 R 040146 R    ADSION  DAC     ADSWCH
00117 R 701312 A            ADRB            /READ CONVERTER BUFFER
00120 R 700042 A    ADCION  ION             /ENABLE PIC FOR OTHER DEVICES
00121 R 060035 R            DAC*    ADCBP   /STORE DATA IN USER BUFFER
00122 R 440035 R            IDX     ADCBP   /INC. BUFFER POINTER
00123 R 440041 R            IDX     ADWPCT  /INC. WORD PAIR COUNTER
00124 R 340034 R            TAD     ADCKSM  /ADD CHECKSUM
00125 R 040034 R            DAC     ADCKSM  /STORE IT
00126 R 440040 R            ISZ     ADWC    /IS I/O COMPLETE
00127 R 600143 R            JMP     ADCONT  /NO KEEP GOING
00130 R 200041 R            LAC     ADWPCT  /YES,COMPUTE WORD COUNT PAIR
00131 R 740030 A            IAC             /MAY BE ODD
00132 R 742030 A            SWHA            /TO TOP HALF
00133 R 740020 A            RAR             /MAKE WD. PRS.
00134 R 500163 R            AND     (377000 /8 BITS ONLY
00135 R 060036 R            DAC*    ADLBHP  /STORE IN HEADER #1
00136 R 440036 R            IDX     ADLBHP  /INC. TO STORE CKSUM
00137 R 340034 R            TAD     ADCKSM  /ADD WORD PAIR COUNT
00140 R 060036 R            DAC*    ADLBHP  /STORE IN HEADER #2
00141 R 140037 R            DZM     ADUND   /CLEAR DEVICE UNDERWAY
00142 R 600145 R            JMP     ADDISM  /EXIT
00143 R 700002 A    ADCONT  IOF             /DISABLE  PIC TO ENSURE DISMISSAL
00144 R 701304 A            ADSC            /BEFORE INTERRUPT FROM THIS IOT OCCURS
                    /INTERRUPT HANDLER DISMISS RTE
                    /
00145 R 200154 R    ADDISM  LAC     ADCAC   /RESTORE AC
                            .EJECT
```

```
PAGE   5      ADC.    SRC

        00146 R 700042 A    ADSWCH  ION              /ION OR IOF
        00147 R 703344 A            DBR              /DEBREAK AND RESTORE
        00150 R 620153 R            JMP*    ADCOUT   /LINK,EX.MODE,MEM,PROT
        00151 R 000000 A    ADCALP  0                /ADD CAL POINTER
        00152 R 000000 A    ADARGP  0                /ADD ARGUMENT POINTER
        00153 R 000000 A    ADCOUT  0                /PC,L,FM,MP
        00154 R 000000 A    ADCAC   0                /AC SAVED HERE
                            /
                000000 A            .END
        00155 R 600007 R *L
        00156 R 000004 A *L
        00157 R 000400 A *L
        00160 R 007000 A *L
        00161 R 000000 A *L
        00162 R 600077 R *L
        00163 R 377000 A *L
            SIZE=00164     NO ERROR LINES
```

```
PAGE   6      ADC.    SRC

ADARGP  00152 R    ADBUSY  00047 R    ADCAC   00154 R    ADCALP  00151 R
ADCBP   00035 R    ADCINT  00104 R    ADCION  00120 R    ADCKSM  00034 R
ADCMOD  00033 R    ADCONT  00143 R    ADCOUT  00153 R    ADCPIC  00077 R
ADC.    00000 R    ADDISM  00145 R    ADERR6  00023 R    ADERR7  00025 R
ADIGN1  00075 R    ADIGN2  00074 R    ADINIT  00027 R    ADLBHP  00036 R
ADRB   701312 A    ADREAD  00051 R    ADSC   701304 A    ADSF   701301 A
ADSION  00116 R    ADSTOP  00042 R    ADSWCH  00146 R    ADUND   00037 R
ADWAIT  00044 R    ADWC    00040 R    ADWPCT  00041 R    DSPCH   00007 R
IDX    440000 A    .MED   000003 A
```

```
PAGE   7      ADC.    SRC

ADC.    00000 R    .MED   000003 A    DSPCH   00007 R    ADERR6  00023 R
ADERR7  00025 R    ADINIT  00027 R    ADCMOD  00033 R    ADCKSM  00034 R
ADCBP   00035 R    ADLBHP  00036 R    ADUND   00037 R    ADWC    00040 R
ADWPCT  00041 R    ADSTOP  00042 R    ADWAIT  00044 R    ADBUSY  00047 R
ADREAD  00051 R    ADIGN2  00074 R    ADIGN1  00075 R    ADCPIC  00077 R
ADCINT  00104 R    ADSION  00116 R    ADCION  00120 R    ADCONT  00143 R
ADDISM  00145 R    ADSWCH  00146 R    ADCALP  00151 R    ADARGP  00152 R
ADCOUT  00153 R    ADCAC   00154 R    IDX    440000 A    ADSF   701301 A
ADSC   701304 A    ADRB   701312 A
```

```
PAGE    8     ADC.    CROSS REFERENCE

ADARGP  00152   13    14    15    16    46    50    51    92    95
                96   104   106   156*
ADBUSY  00047   80*   86
ADCAC   00154  109   116   150   158*
ADCALP  00151   12    81    88   155*
ADCBP   00035   54*   93   100   101   128   129
ADCINT  00104   55   115*  117   120
ADCION  00120  112   124   127*
ADCKSM  00034   53*   99   131   132   142
ADCMOD  00033   52*   57
ADCONT  00143  134   146*
ADCOUT  00153  111   118   154   157*
ADCPIC  00077  109*  115   119
ADC.    00000    8    12*
ADDISM  00145  145   150*
ADERR6  00023   23    24    28    30    38*
ADERR7  00025   41*   91
ADIGN1  00075   25    26    63    78   105*
ADIGN2  00074   21    22   104*
ADINIT  00027   20    46*
ADLBHP  00036   55*   94   140   141   143
ADRB   701312    6*  126
ADREAD  00051   27    84*
ADSC   701304    4*  103   147
ADSF   701301    3*   54
ADSION  00116  113   125*
ADSTOP  00042   58    62*
ADSWCH  00146  125   152*
ADUND   00037   56*   62    76    84    87   144
ADWAIT  00044   29    76*
ADWC    00040   57*   97   133
ADWPCT  00041   58*   98   130   135
DSPCH   00007   17    18    19*
IDX    440000    9*   14    16    46    51    95   100   101   104
               129   130   141
.MED   000003   10*   39    42
```

5.3 Device Handlers Acceptable to System Programs

The following paragraphs provide listings of .DAT Slot assignments for the various system programs and the I/O device handlers which may be assigned to each. Standard assignments for 8K systems are indicated by an asterisk (*).

It is imperative to note that only one I/O handler for a device may be in core at the same time (i.e., PRA and PRB should not be brought in together since there is no communication between their interrupt handling routines).

5.3.1 FORTRAN IV (F4)

.DAT Slot	Use	Handler
-11	Input	TTA
		PRA
		*PRB
		CDB
		DTA, DKA, MTA (for 3 open files)
		DTB, DKB (for 2 open files)
		DTC, DKC, MTC (1 input file only)
		DTD, DKD DTE, DKE (1 file only)
		DTF, DKF, MTF (non-file-oriented)
-12	Listing	*TTA
		LPA
		VPA
		PPA
		DTA, DKA, MTA (for 3 open files)
		DTB, DKB (for 2 open files)
		DTD, DKD DTE, DKE (1 file only)
		DTF, DKF, MTF (non-file-oriented)
-13	Output	PPA
		PPB
		*PPC
		DTA, DKA, MTA (for 3 open files)
		DTB, DKB (for 2 open files)
		DTD, DKD DTE, DKE (1 file only)
		DTF, DKF, MTF (non-file-oriented)

5.3.2 MACRO-15

.DAT Slot	Use	Handler
-11	Input	TTA
		PRA
		*PRB
		CDB
		DTA, DKA, MTA (for 3 open files)
		DTD, DKD DTE, DKE (1 file only)

5.3.2 (Cont.)

.DAT Slot	Use	Handler
-10	Parameter File Input	TTA *PRA PRB (recommended) CDB DTA, DKA, MTA (for 3 open files) DTD, DKD } (1 file only) DTE, DKE
-14	Macro Def-initions File	TTA PRA PRB CDB *DTA, DKA, MTA (for 3 open files) DTD, DKD } (1 file only) DTE, DKE
-12	Listing Output	*TTA LPA VPA PPA DTA, DKA, MTA (for 3 open files) DTD, DKD } (1 file only) DTE, DKE
-13	Output	PPA PPB *PPB

5.3.3 FOCAL

.DAT Slot	Use	Handler
3	Library Input	TTA PRA PRB CDB *DTA, DKA, MTA (for 3 open files) DTC, DKC, MTC (1 input file only) DTD, DKD (1 file only) DTE, DKE (recommended - 1 file only)
5	Library Output	TTA PPA DTA, DKA, MTA (for 3 open files) DTD, DKD (1 file only) DTE, DKE (recommended - 1 file only) LPA VPA

5.3.3 (Cont.)

.DAT Slot	Use	Handler
7	Data File Input	TTA PRA PRB CDB *DTA, DKA, MTA (for 3 open files) DTC, DKC, MTC (1 input file only) DTD, DKD (1 file only) DTE, DKE (recommended - 1 file only)
10	Data File Output	TTA PPA *DTA, DKA, MTA (for 3 open files) DTD, DKD (1 file only) DTE, DKE (recommended - 1 file) LPA VPA

5.3.4 EDIT and EDITVP

.DAT Slot	Use	Handler
-15	Scratch/ Output	TTA VPA LPA PPA *DTA, DKA, MTA (required for input and output DTD, DKD } (1 file only) DTE, DKE
-14	Input	TTA PRA PRB CDB *DTA, DKA, MTA (required for input and output) DTD, DKD } (1 file only) DTE, DKE
-10	Secondary Input	TTA *PRA PRB (recommended) CDB
10	Display Output (EDITVP only)	VPA

5.3.5 Linking Loader and DDT

.DAT Slot	Use	Handler
-1	System Library Input	PRA DTA, DKA (for 3 open user files) DTB, DKB (for 2 open user files) *DTC, DKC (1 input file only - recommended if no user I/O) DTD, DKD DTE, DKE
-4	User Program Input	Same as specified above for .DAT Slot -1.
-5	External User Library Input	Same as specified above for .DAT Slot -1.

NOTE

Since Linking Loader handlers can be used by the program being loaded, choice of bulk storage handlers should be made in terms of user requirements.

5.3.6 PIP (Peripheral Interchange Program)

PIP uses all the positive .DAT Slots (1 through 10) plus -2 and -3 for TTY I/O. Prior to use, any non-standard device assignments should be made via the ASSIGN command to the Monitor. If several functions are to be used with a variety of peripherals, assignment of these devices all at the same time avoids the necessity for returning to the Monitor to reassign devices and for repeatedly reloading PIP after each operation that requires a new device. Conversely, it may be necessary to clear certain unused .DAT Slots (i.e., ASSIGN NONE n, n,...,etc.) to prevent loading of standardly assigned handlers. This is particularly useful when operating in 8K with non-standard handlers the size of which, in combination with other standard handlers, could cause core overflow during loading of PIP (.SYSLD 1 error).

NOTE

The device handlers used with PIP should normally be those having the greatest capability (i.e., PRA, PPA, DTA, DKA, etc.). If both input and output are to occur on the same device (e.g., DECtape), separate .DAT Slots must be assigned. Both .DAT Slots must be assigned to the same handler; otherwise erroneous results will occur since there is no communication between the interrupt service routines of different handlers (e.g. DTA assigned to one. DAT Slot and DTB assigned to another).

PIP standard (8K) assignments are as follows:

.DAT Slot	Use	Handler
1	I/O	*DTA
2	I/O	*DTA
3	I/O	*DTA
4	I/O	*TTA
5	Input	*PRA
6	Output	*PPA
7	I/O	*DTA
10	I/O	*DTA

5.3.7 SGEN (System Generator)

.DAT Slot	Use	Handler
-15	Output	*DTA, DKA
-14	Input	*DTA, DKA

5.3.8 PATCH

.DAT Slot	Use	Handler
-10	Secondary Input	*PRA DTA, DKA
-14	I/O	*DTA, DKA DTD, DKD DTE, DKE

5.3.9 UPDATE

.DAT Slot	Use	Handler
-14	Input	PRA *DTA, DKA, MTA
-15	Output	PPA PPB PPC *DTA, DKA, MTA
-10	Secondary Input	*PRA DTA, DKA, MTA
-12	Listing	LPA *TTA VPA PPA DTA, DKA, MTA

5.3.10 DUMP

.DAT Slot	Use	Handler
-14	Input	*DTA, DKA, MTA DTD, DKD DTE, DKE
-12	Listing	*TTA LPA VPA PPA DTA, DKA, MTA DTD, DKD DTE, DKE

5.3.11 CHAIN

.DAT Slot	Use	Handler
-4	Input	PRA DTA, DKA, MTA DTB, DKB (recommended) *DTC, DKC, MTC (Use only if no DTD, DKD other DT, DK, or DTE, DKE MT is assigned)
-1	System Library	Same as for .DAT -4
-5	User Library	

NOTE

Use the smallest handlers possible since they are not recoverable as user handlers (i.e., in the overlay system).

5.3.12 EXECUTE

.DAT Slot	Use	Handler
-4	CHAIN-Built Overlay System Input (XCT, XCU Files)	PRA DTA, DKA, MTA DTB, DKB *DTC, DKC, MTC (use only if not shared with overlay system) DTD, DKD DTE, DKE

5.3.13 SRCCOM (Source Compare)

.DAT Slot	Use	Handler
-15	Old File Input	TTA (if not assigned to -14) PRA (if not assigned to -14) CDB (if not assigned to -14) *DTA, DKA, MTA DTD, DKD DTE, DKE
-14	New File Input	TTA (if not assigned to -15) PRA (if not assigned to -15) CDB (if not assigned to -15) *DTA, DKA, MTA DTD, DKD DTE, DKE
-12	Listing	*TTA PPA LPA VPA DTA, DKA, MTA DTD, DKD DTE, DKE

5.3.14 DTCOPY (DECtape Copy)

.DAT Slot	Use	Handler
-14	Input	*DTA, DKA DTD, DKD DTE, DKE
-15	Output	*DTA, DKA DTD, DKD DTE, DKE

5.3.15 8TRAN (PDP-8 to PDP-15 Translator)

.DAT Slot	Use	Handler
-15	Input	PRA CDB TTA *DTA, DKA, MTA DTD, DKD DTE, DKE
-14	Output	PPA LPA TTA VPA *DTA, DKA, MTA DTD, DKD DTE, DKE

5.4 SUMMARY OF STANDARD I/O HANDLER FEATURES

5.4.1 TTA. (Teletypewriter)

5.4.1.1 General Description - TTA. (469_{10} registers) is embedded in the Resident Monitor and provides all necessary functions for teletypewriter I/O. All functions (described below), except .READ and .WRITE, refer to action taken when either the teleprinter or the keyboard is addressed.

5.4.1.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard buffer size (34_{10}) b. Assign return address for certain control characters (CTRL C, CTRL T, CTRL P) from contents of CAL ADDRESS+2. Bits 0 and 1 in CAL+2 are set to designate caller: Bit 0-1 — Caller 01 — Monitor (↑C) 10 — DDT (↑T) 00 — All others (↑P) c. Set I/O UNDERWAY indicator d. Print Carriage RETURN/LINE FEED (CR/LF)
.DLETE, .RENAM, .FSTAT	2	Ignored
.SEEK	3	Ignored
.ENTER	4	Ignored
.CLEAR	5	Ignored
.CLOSE	6	a. Set I/O UNDERWAY indicator b. Print CR/LF c. Wait for completion of I/O.
.MTAPE	7	Ignored
.READ	10	a. Set I/O UNDERWAY indicator b. Set up to accept characters from keyboard
.WRITE	11	a. Set I/O UNDERWAY indicator b. Print
.WAIT, .WAITR	12	Test for I/O UNDERWAY (1) If busy, return to CAL(.WAIT) or to address in CAL+2 (.WAITR) (2) If non-busy, return to CAL+2 (.WAIT) or to CAL+3 (.WAITR)
.TRAN	13	Illegal function IOPS 6

5.4.1.3 Legal Data Modes

IOPS ASCII (Mode 2)
Image Alphanumeric (Mode 3)

5.4.1.4 Function Characters - The following function characters may have special significance when input or output in IOPS ASCII Mode. In Image Alphanumeric Mode, these characters are treated as ordinary ASCII.

Character	Transfer Direction	Action
Carriage RETURN (015_8)	Input	Insert in the user's buffer
	Output	Output Carriage RETURN/LINE FEED operation (015/012)
ALT MODE (175_8)	Input	Accept 33,175 or 176 and map into user's buffer as 175
	Output	Output Carriage RETURN/LINE FEED
FORM Feed (014_8) or VT (Vertical Tab) (013_8)	Input	Insert in user's buffer, as applicable
	Output	Output FORM Feed function
Horizontal TAB (011_8)	Input	Insert in user's buffer
	Output	Model 35 - Output (011_8) TAB function
		Model 33 - Output sufficient number of spaces (040_8) to position the printer at column 9,17,25,...,70.
RUBOUT (177_8)	Input	Delete previous character typed and echo a reverse slash (\\).
	Output	Ignore
LINE FEED (012_8)	Input	Insert in user's buffer
	Output	Ignore all leading LINE FEEDs and output all others.
CTRL U (025_8)	Input	Delete all characters typed since last Carriage RETURN, and echo a commercial at (@) sign. If output is underway, terminate printing and output a Carriage RETURN/LINE FEED operation.
	Output	Ignore
Null (000_8)	Input/Output	Ignore

5.4.1.5 Program Control Characters - The following Program Control Characters, regardless of mode of operation or of transfer direction, are recognized when typed on the keyboard. The current I/O function is stopped and the character is decoded as described below[1]:

[1]Character will be ignored (no echo) for CTRL C, P and T if respective .INIT has not been performed.

Character	Action
CTRL C (echoes ↑C)	Transfer control to the address specified as return in the .INIT (to the TTY) performed by the Monitor.
CTRL P (echoes ↑P)	Transfer control to the address specified as return in the .INIT (to the TTY) performed by the user (other than Monitor or DDT).
CTRL T (echoes ↑T)	Transfer control to the address specified as return in the .INIT (to the TTY) performed by DDT.
CTRL S (echoes ↑S)	Transfer control to the address specified in .SCOM+6 (location 106_8 in the Monitor).
CTRL Q (echoes ↑Q)	Transfer control to Monitor Save routine (KM9SAV).

5.4.1.6 Unrecoverable Errors (No Program Initiated Recovery)

a. Illegal Data Mode - IOPS 7

b. Illegal Function - IOPS 6

5.4.1.7 Restriction - TTY I/O can only be requested from mainstream in API systems, since the teleprinter is not connected to the API.

5.4.2 PP (Paper Tape Punch)

5.4.2.1 General Description - Three handlers are provided for use with the Paper Tape Punch. PPA (377_{10} registers) is the most general and operates in all data modes. PPB (270_{10} registers) accepts data in all modes except IOPS ASCII. PPC (210_{10} registers) accepts IOPS Binary **only.**

5.4.2.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard buffer size (52_{10}). b. .SETUP - no API. c. Punch two fanfolds of leader.
.DLETE .RENAM .FSTAT	2	Ignore
.SEEK	3	Illegal function (IOPS 6)
.ENTER	4	Ignore
.CLEAR	5	Ignore
.CLOSE	6	a. Allow previous output to terminate. b. Punch EOF if IOPS Binary. c. Punch two fanfolds of trailer. d. Allow trailer punching to terminate.

Mnemonic	Code	Action
.MTAPE	7	Ignore
.READ	10	Illegal function (IOPS 6)
.WRITE	11	a. Allow previous output to terminate. b. Output buffer.
.WAIT, .WAITR	12	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).
.TRAN	13	Illegal function (IOPS 6)

5.4.2.3 Legal Data Modes

a. IOPS Binary (Mode 0) PPA., PPB., PPC.
b. IMAGE Binary (Mode 1) PPA., PPB.
c. IOPS ASCII (Mode 2) PPA.
d. IMAGE Alphanumeric (Mode 3) PPA., PPB.
e. Dump (Mode 4) PPA., PPB.

5.4.2.4 Vertical Control Characters (IOPS ASCII only) - May appear as only first character of line and will be ignored if elsewhere in line; if no vertical control character at beginning of line, a line feed (012_8) will be used.

a. LINE FEED (012_8) - Output
b. VT (Vertical Tab 013_8) - Output, followed by four RUBOUTs (177_8)
c. FORM Feed (014_8) - Output, followed by 40_8 Nulls (000_8)

5.4.2.5 Horizontal Control Characters (IOPS ASCII only)

TAB (011_8) - Output followed by one RUBOUT (177_8)

5.4.2.6 Recoverable Errors

No tape in punch	Monitor error IOPS 4 a. Put tape in punch b. Type CTRL R

a.	Illegal function	Monitor error IOPS 6 a. .SEEK b. .READ c. .TRAN
b.	Illegal data mode	Monitor error IOPS 7

5.4.2.8 Restriction - In API systems, the Paper Tape Punch can be called only from mainstream, since the punch is not connected to the API.

5.4.3 PR (Paper Tape Reader)

5.4.3.1 General Description - Two handlers are provided for use with the Paper Tape Reader. PRA. (444_{10} registers) operates in all data modes, while PRB. (294_{10} registers) accepts IOPS ASCII only.

5.4.3.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard line buffer size (52_{10}) b. .SETUP API channel register 50_8 c. Clear I/O UNDERWAY indicator
.DLETE .RENAM .FSTAT	2	Ignore
.SEEK	3	Ignore
.ENTER	4	Illegal function (IOPS 6)
.CLEAR	5	Illegal function (IOPS 6)
.CLOSE	6	Allow previous input to finish and then clear I/O UNDERWAY indicator.
.MTAPE	7	Ignore
.READ	10	a. Allow previous input to be completed. b. Input line or block of data (see modes below).
.WRITE	11	Illegal function (IOPS 6).
.WAIT, .WAITR	12	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)
.TRAN	13	Illegal function (IOPS 6)

5.4.3.3 Legal Data Modes

a. IOPS ASCII (Mode 2) PRA., PRB.

(1) Constructs line buffer header, computing:

(a) Word pair count
(b) Data mode
(c) Data validity bits

(2) Packs characters into the line buffer in 5/7 ASCII, checking parity (eighth bit, even), on each character.

(3) Allows vertical form control characters. (FF, LF, VT) only in character position 1 of the line buffer. Otherwise, ignored.

(4) Terminates reading on CR or line buffer overflow. In the latter case, tape is moved past the next CR to be encountered.

b. IOPS Binary (Mode 0) PRA.

(1) Reads binary data in alphanumeric mode, checking parity (seventh hole, odd) on each frame.

(2) Accepts line buffer header at head of input data, modifying data validity bits if parity or checksum error (or short line) have occurred.

(3) 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.

c. Image Alphanumeric (Mode 3) PRA.

(1) Constructs line buffer header, computing:

(a) Word pair count
(b) Data mode

(2) Stores characters, without editing, or parity checking in the line buffer, one per register.

(3) Terminates reading as a function of .READ macro word count.

d. Image Binary (Mode 1) PRA.

Same as Image Alphanumeric; however a binary read is issued to the PTR.

e. Dump (Mode 4) PRA.

Same as Image Alphanumeric except 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.

5.4.3.4 Unrecoverable Errors

a. Illegal function — Monitor error IOPS 6

(1) .ENTER

(2) .CLEAR

(3) .WRITE

(4) .TRAN

b. Illegal Data Mode — Monitor error IOPS 7

5.4.4 DT (DECtape)

5.4.4.1 General Description - Six handlers are available for DECtape operations.

DTA. (2296_{10} registers) is the most general DECtape handler issued with the 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.

DTB. (1554_{10} registers) has a simultaneous 2-file capacity, one input and one 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 buffers, one 32_{10}-word Directory Bit Map, and one 32_{10}-word File Bit Map. Functions included are: .INIT, .ENTER, .READ, .WAIT, .WAITR, .SEEK, .CLOSE, and .WRITE.

DTC. (689_{10} registers) is the most limited (and conservative in terms of core allocation) DECtape handler in the 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 buffer to handle either IOPS ASCII or IOPS Binary input (and no other). Functions included are: .INIT, .SEEK, .CLOSE, .READ, .WAIT, .WAITR.

DTD. (1593_{10} registers) has full IOPS function capabilities including .MTAPE commands (REWIND, BACKSPACE). 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 DTD.. One 256_{10}-word data buffer, one 32_{10}-word Directory Bit Map, and one 32_{10}-word File Bit Map are included.

DTE. (1468_{10} registers) has the same capabilities as DTD. except the .MTAPE function is not allowed.

DTF. (617_{10} registers) is a non-file-oriented, multi-unit handler which will accommodate (serially) up to eight DECtape units, both input and output. When the last block on a tape has been accessed, IOPS 4 will be typed. The user may continue onto another tape simply by dismounting the current tape, replacing it with another, and typing CTRL R. If the tape is not replaced at this time, and CTRL R is typed, the contents of this tape will be lost if a .WRITE operation is being performed. The handler accepts IOPS ASCII and Binary with no internal buffering. Legal functions are as follows: .CLOSE, .READ, .WRITE, .MTAPE, .INIT, .WAIT, .WAITR.

5.4.4.2 Functions

Mnemonic	Code	Action
.INIT	1	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.

Mnemonic	Code	Action
..DLETE .RENAM .FSTAT	2	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 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.

Mnemonic	Code	Action
.SEEK	3	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.
.ENTER	4	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.
.CLEAR	5	a. Zeroes 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.
.CLOSE	6	a. File-oriented Operation (1) 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. (2) 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. (3) Records newly constructed File Bit Map. (4) Loads Directory into memory, enters new entry and records Directory again with new entry and updated Directory Bit Map. (5) Clears internal program switches.

Mnemonic	Code	Action
		b. Non-file-oriented Operation (DTD. and DTF. only) During output, a three word EOF block is written as the last DECtape block of the logical record, as follows: 001005 776773 000000 The remaining words of the EOF DECtape block are zero.
.MTAPE	7	a. Rewind (1) Sets internal switches such that data transfer will begin at block 0 in the forward direction. (2) Declares the unit to be non-file-oriented (i.e., data will be recorded, beginning at block 0, and continuing every fifth block thereafter). When EOT is reached, recording continues in the reverse direction. Five passes are required to record the entire tape (1100_8 blocks). b. Backspace - Decrements the internal block pointer to the next block to be transferred. c. Space Forward One Record - The block pointer is incremented by 5 (no physical action). d. Other .MTAPE functions ignored.
.READ	10	a. Inputs line from DECtape handler buffer or block of data to user area. (See 5.4.4.3 for data modes.) b. Initiates input of next DECtape block when preceding block has been emptied.
.WRITE	11	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 Line (word 377_8) of current block output.
.WAIT, .WAITR	12	Checks 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)
.TRAN	13	Transfers (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

Mnemonic	Code	Action
		user). On input, transfer 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.

5.4.4.3 Legal Data Modes

IOPS ASCII (Mode 2) DTA., DTB., DTC., DTD., DTE., DTF.

IOPS Binary (Mode 0) DTA., DTB., DTC., DTD., DTE., DTF.

Image Alphanumeric (Mode 3) DTA., DTD., DTE.

Image Binary (Mode 1) DTA., DTD., DTE.

Dump (Mode 4) DTA., DTD., DTE.

5.4.4.4 Recoverable Errors

Select Error[1]	Monitor Error IOPS 4 a. Ready the desired DECtape unit b. Type CTRL R on the TTY.

5.4.4.5 Unrecoverable Errors

a. Illegal Function	Monitor Error IOPS 6 See 5.4.4.1 for legal functions for each DT handler.
b. Illegal Data Mode	Monitor Error IOPS 7 (1) .SEEK with .INIT for output. (2) .ENTER with .INIT for input. (3) See 5.4.4.1 for .READ, .WRITE legal data modes.
c. File Still Active	Monitor Error IOPS 10 .SEEK, .ENTER, .CLEAR or .OPER when last file has not been closed.
d. .SEEK, .ENTER Not Executed	Monitor Error IOPS 11 .READ or .WRITE executed prior to .SEEK or .ENTER (or .MTAPE-REWIND)
e. DECtape Error	Monitor Error IOPS 12 (1) Mark Track Error (2) EOT during read or write
f. File Not Found	Monitor Error IOPS 13 File name not found in Directory on a .SEEK
g. DECtape Directory Full	Monitor Error IOPS 14 Directory Entry Section found full on an .ENTER

[1]A "Select" error is equivalent to a hardware not ready condition.

h.	DECtape Full	Monitor Error IOPS 15 All DECtape blocks occupied on a .WRITE or .ENTER
i.	Output Buffer Overflow	Monitor Error IOPS 16 (1) Output line (IOPS ASCII or Binary) greater than 255_{10} cells (including header). (2) Output block (Image Binary or Image Alphanumeric) greater than 255_{10} cells (excluding header).
j.	Excessive Number of Files Referenced	Monitor Error IOPS 17 See 5.4.4.1 for file reference limitations.
k.	Two output files on same unit	Monitor Error IOPS 22 Two output files open simultaneously on the same unit
l.	Illegal Word Pair Count (WPC	Monitor Error IOPS 23 WPC = 0, or greater than 177

5.4.5 RF (RF15 DECdisk)

The following naming convention is observed with the handlers described below. Although the RF15 handlers are named RFA., RFB., etc., the system software expects handler names such as: DKA., DKB., etc. Therefore, the .GLOBL name given in these handlers is DKn. All user references must be to DK rather than RF.

5.4.5.1 General Description - The following six handlers are provided for DECdisk operation.

RFA. (2269_{10} registers) is the most general Disk handler for the RF/RS Disk issued with the ADVANCED Software System. RFA. has a simultaneous 3-file capacity, either input or output. Files may be referenced on the same or different Disk 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.

RFB. (1536_{10} registers) has a simultaneous 2-file capacity, one input and one output. Both files may be on the same or different units. RFB. 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, .WAITR
.SEEK	.CLOSE	.WRITE	

RFC. (655_{10} registers) is the most limited (and conservative in terms of core allocation) Disk handler in the ADVANCED Software System. RFC. is a read-only handler with a 1-file capacity requiring no space for Bit Maps and only one 256_{10}-word data buffer to handle either IOPS ASCII or IOPS Binary input (and no other). Functions included are:

.INIT	.CLOSE	.WAIT, .WAITR
.SEEK	.READ	

RFD. (1517_{10} registers) has full IOPS function capabilities including .MTAPE commands (REWIND, BACKSPACE). 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 RFD.. One 256_{10}-word data buffer, one 32_{10}-word Directory Bit Map, and one $32_{910})$-word File Bit Map are included.

RFE. (1436_{10} registers) is the same as RFD. except that it will not handle .MTAPE commands.

RFF. (553_{10} registers) is a non-file-oriented, multi-unit handler which will accommodate (serially) up to four RF15 DECdisk platters (eight logical units), both input and output. When either the last block (forward direction) or first block (backspacing) of a unit has been accessed, an IOPS 4 message will be typed. The user may continue onto the next sequential (higher or lower) disk unit by typing CTRL R. The handler accepts both IOPS ASCII and Binary with no internal buffering. Legal functions are as follows:

.READ .WRITE .MTAPE .INIT .WAIT .WAITR .CLOSE

5.4.5.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard line buffer size (255_{10}) b. .SETUP API channel register 63_8 c. Set direction switch (input or output)

NOTE

In order to change direction when operating in a file-oriented environment, a new .INIT must first be executed.

Mnemonic	Code	Action
.DLETE .FSTAT .RENAM	2	a. .DLETE (1) Examines specified Directory for presence of desired file name. If not found, AC = 0 upon return to user.

Mnemonic	Code	Action
		(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
		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 Ø-2 of CAL address +2 = 5 to designate the DECdisk.
.SEEK	3	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.
.ENTER	4	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.

Mnemonic	Code	Action
.CLEAR	5	a. Zeroes out File Bit Map blocks 71_8 through 77_8 on specified disk unit.
		b. Initializes Disk Directory block 100 to indicate that 72_{10} blocks (71_8 through 100_8 and 1000_8 through 1077_8) are occupied.
.CLOSE	6	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 with new entry and updated Directory Bit Map.
		e. Clears internal program switches.
.MTAPE	7	a. Rewind (1) Initializes internal switches to permit data transfer beginning at block 0. (2) Declares the unit to be non-file-oriented (i.e. data will be recorded starting at block 0 and sequentially thereafter.
		b. Backspace - Decrements the internal block pointer to point to the next previous sequential record.
		c. Space Forward One Record - The block pointer is incremented by one to point to the next sequential record.
		d. Other .MTAPE functions ignored.
.READ	10	a. Input line or block of data from handler's buffer to user's buffer.
		b. When handler's buffer is empty, input next block from the disk.
.WRITE	11	a. Output line or block of data from user's buffer to the handler's buffer.
		b. When handler's buffer is full, output a block of data to the disk examining the Directory Bit Map for the next free block number to store as the data link (word 377_8) of the current block output.

.WAIT .WAITR	12	Check I/O UNDERWAY: (1) If busy, return to CAL (.WAIT) or to the address in CAL + 2 (.WAITR). (2) If not busy, return to CAL + 2 (.WAIT) or to CAL + 3 (.WAITR).
.TRAN	13	Transfer the number of words specified (.TRAN arg) to (or from) the core area specified (.TRAN arg) from (or to) the disk block specified (.TRAN arg). Data is transferred from (or to) contiguous disk blocks in the <u>forward</u> direction (.TRAN arg). Transfer terminates when the word count (.TRAN arg) overflows. During output, however, if the word count is not equal to an integral number of disk blocks, the remaining words in the last block are zero-filled.

5.4.5.3 Legal Data Modes

a. IOPS ASCII (Mode 2) RFA., RFB., RFC., RFD., RFE., RFF.

b. IOPS Binary (Mode 0) RFA., RFB., RFC., RFD., RFE., RFF.

c. Image Alphanumeric (Mode 3) RFA., RFD., RFE.

d. Image Binary (Mode 1) RFA., RFD., RFE.

e. Dump (Mode 4) RFA., RFD., RFE.

5.4.5.4 Recoverable Errors

Device Not Ready	Monitor Error IOPS 4 a. WRITE ENABLE the appropriate disk unit. b. Type CTRL R on the TTY.

5.4.5.5 Unrecoverable Errors

a. Illegal Function	Monitor Error IOPS 6 See 5.4.5.1 for legal handler functions.
b. Illegal Data Mode	Monitor Error IOPS 7 (1) .SEEK executed with disk .INITed for output. (2) .ENTER executed with disk .INITed for input. (3) See 5.4.5.1 for legal data modes for .READ and .WRITE.
c. File Still Active	Monitor Error IOPS 10 .SEEK, .ENTER, .CLEAR, .DLETE, .FSTAT or .RENAM executed before a previously opened file has been .CLOSEd.

d.	.SEEK/.ENTER Not Executed	Monitor Error IOPS 11 A .READ or .WRITE has been executed prior to a .SEEK, .ENTER, or .MTAPE (Rewind).
e.	Disk Error	Monitor Error IOPS 12 EOT encountered during a .READ or .WRITE operation.
f.	File Not Found	Monitor Error IOPS 13 File named in a .SEEK not found in disk directory.
g.	Disk Directory Full	Monitor Error IOPS 14 Execution of .ENTER finds directory full.
h.	Disk Full	Monitor Error IOPS 15 No free block can be found during attempt to execute .WRITE or .ENTER.

NOTE

If block 0 is selected as the first block of a file (.WRITE or .ENTER) the disk unit will be declared full (IOPS 15). Otherwise, execution of .FSTAT would produce ambiguous results, since .FSTAT returns either 0's in the AC, if a file is not found, or the first block number of the file, if it is found.

i.	Output Buffer Overflow	Monitor Error IOPS 16 (1) The output line (IOPS Modes) is greater than 255_{10} words (including header). (2) The output block (Image Modes) is greater than 255_{10} (excluding header).
j.	Excessive Number of Files	Monitor Error IOPS 17 Refer to 5.4.5 1 for file reference limitations.
k.	Illegal Disk Address	Monitor Error IOPS 21 (1) Reference made to a nonexistent disk. Bits 15-17 of the CAL address output with the error message indicate the number of the disk platter referenced. (2) An illegal disk address was calculated from a legal initial starting address. The offending logical block number is output with the error message.
l.	Two Output Files on the Same Unit	Monitor Error IOPS 22 Two output files are simultaneously open in the same unit.
m.	Illegal Word Pair Count	Monitor Error IOPS 23 Word pair count is 0 or greater than 177_8.

5.4.6 MT (Magnetic Tape

5.4.6.1 General Description - Three handlers are provided for use with TU20 and TU20A Magnetic Tape Drives.

MTA. (2432_{10} registers) is designed for file-oriented operations only. Up to three files may be concurrently referenced, all on different transports, either input or output. All data modes are allowed and all functions are legal except .MTAPE.

MTC. (683_{10} registers) is a read-only handler having a single file capacity. Legal data modes are IOPS ASCII and Binary. Sequential file references are allowed. Legal functions are:

.SEEK .INIT .CLOSE .READ .WAIT .WAITR

MTF. (624_{10} registers) is a non-file-oriented, multi-unit handler which will accommodate (serially) up to eight concurrently open magnetic tape transports, both input and output. Only non-file-oriented operations are permitted. No internal buffering is performed. Legal data modes are IOPS ASCII and Binary. Legal functions are:

.INIT .READ .WRITE .WAIT .WAITR .MTAPE (Rewind and Backspace)

5.4.6.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard buffer size (255_{10}) b. .SETUP - API channel register 45_8 c. Set transfer direction (input or output) d. If first .INIT to this unit, assign default parity density and track-count settings (i.e., parity is odd, density is 800 BPI and track-count is as specified by .SCOM+4, bit 6: 0 = 7 channel 1 = 9 channel) e. Indicate that the referenced drive is open for I/O transfers.
.DLETE .RENAM .FSTAT	2	a. .DLETE (1) Examine the directory on the referenced unit for a file of the name specified. (2) If file is found, remove the name 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.

Mnemonic	Code	Action
		b. .RENAM (1) Search the directory on the referenced unit for an active file of the name given. If no file is found, return to the user with 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). Also, bits 0 through 2 of CAL address +2 = 4 to designate Magtape.
.SEEK	3	a. Check that no file is open on the referenced unit. Take error return (IOPS 10) if so. b. Check that the referenced unit has been .INIT'ed for input. Take error return (IOPS 7) if not. c. Search directory for a file of the name given. If no file is found, error return (IOPS 13) to Monitor. d. 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). e. Indicate a file open for reading on the referenced unit.
.ENTER	4	a. Check that no file is open on the referenced unit. Take error return (IOPS 10) if so. b. Check that the referenced unit has been declared an output unit. Error return (IOPS 7) if not.

Mnemonic	Code	Action
.ENTER (Cont.)		c. Check that space is available in the File Name Entry Section of the Directory for this file name. Take error return (IOPS 14) if not.
		d. Check that space is available in the Accessibility Map for this file. Take error return (IOPS 42) if not.
		e. Indicate that a file is open for writing on the unit referenced.
.CLEAR	5	a. Rewind and write an empty File Directory at the front of the tape, along with a logical End-Of-Tape indicator.
.CLOSE	6	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 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.

Mnemonic	Code	Action
.MTAPE	7	a. 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 thru 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	9	Even	800 BPI
14	7	Odd	200 BPI
15	7	Odd	556 BPI
16	7	Odd	800 BPI
17	9	Odd	800 BPI

.READ 10

a. Check that referenced unit is input. IOPS 7 if not.

b. Check that a file is open for reading: IOPS 11 if not.

c. Initiate data transfer

d. Read Errors

(1) Parity/Checksum Errors

Mnemonic	Code	Action
.READ (cont.)		(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 (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 Mode 4: 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 user's line buffer area. The format of the 2-word line is as follows: Header word 0: 001006 Header word 1: 776772 (b) Non-File-Structured Environment. Modes 0 - 3: Exactly as described for file-structured environment (3a above). Mode 4: Error return (IOPS 43).
.WRITE	11	a. Check that referenced unit is output. IOPS 7 if not. b. Check that a file is open for writing: IOPS 11 if not. c. Initiate data transfer. 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".

Mnemonic	Code	Action
.WRITE (cont.)		e. Write Errors: Continued attempts are made to rewrite the record in error. The process terminates when EOT is encountered.
.WAIT, .WAITR	12	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).

NOTE

On a non-busy return, the accumulator contains the contents of the magnetic tape status register as it appeared on completion of the latest operation. This is the only facility the user has for checking I/O errors in the .TRANS and dump-mode transfers.

Mnemonic	Code	Action
.TRAN	13	Honor subfunction indicator as follows:

Subfunction	Action
0	Input Forward - Transfer next physical block on tape to user's buffer area.
1	Output Forward - Transfer from user's buffer directly to the next physical block on tape.
2	Illegal Function - Monitor Error IOPS 6
3	Illegal Function - Monitor Error IOPS 6

5.4.6.3 Legal Data Modes

a. IOPS Binary (Mode 0)

(1) Acceptable handlers: MTA., MTC., MTF.

(2) Output

(a) File-structured Tape (MTA.)
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.

(b) Non-File-Structured Tape (MTF.)

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.

(3) Input

(a) File-Structured Tape (MTA., MTC.)

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.

(b) Non-File-Structured Tape (MTF.)

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 (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.

b. Image Binary (Mode 1)

(1) Acceptable Handlers: MTA.

(2) Handler operation is exactly as described for IOPS Binary (above). Headers and data are transferred in file-structured mode. Modifications are limited to the checksum word and the validity field as stated above.

c. IOPS ASCII (Mode 2)

(1) Acceptable Handlers: MTA., MTC., MTF.

(2) Operation is the same as described for IOPS Binary Mode (above).

d. Image Alphanumeric (Mode 3)

(1) Acceptable Handlers: MTA.

(2) 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.

e. Dump (Mode 4)

(1) Acceptable Handlers: MTA.

(2) 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 .READs and .WRITEs is identical in both file-structured and non-file-structured environments.

(a) Output

Data is 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 is 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 is 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.

5.4.6.4 Recoverable Errors

IOPS 4 (Device Not Ready)

a. Cause:
   (1) Transport OFF LINE
   (2) Unit number incorrect
   (3) Attempt to .WRITE with WRITE LOCK set to ON
   (4) 9-Channel I/O request to a 7-channel transport (and vice-versa)

b. Recovery:
   (1) Correct fault
   (2) Type CTRL R

5.4.6.5 Unrecoverable Errors

a. Illegal Function - Monitor Error IOPS 6
   (1) Attempt to execute file-structured to non-file-structured transport (and vice-versa)
   (2) An input request made to an output unit (and vice-versa)
   (3) A .TRAN was attempted in the reverse direction.

b. Illegal Data Mode - Monitor Error IOPS 7

   Illegal data mode for particular handler version (see 5.4.6.4).

c. File Still Active - Monitor Error IOPS 10

   A .SEEK, .ENTER, .CLEAR, .RENAME, .DLETE, or .FSTAT requested while a file is still open on the specified unit.

d. SEEK/ENTER Not Executed - Monitor Error IOPS 11

   A .READ or .WRITE has been requested to a file-structured transport without performing either a .SEEK or a .E TER.

e. EOT Encountered on Read - Monitor Error IOPS 12

   Physical End-Of-Tape encountered during an input operation.

f. File Not Found - Monitor Error IOPS 13

   During the processing of a .SEEK, the requested file name is absent from the File Name Entry Section of the specified Directory.

g. Directory Overflow - Monitor Error IOPS 14

During the processing of .ENTER, the File Name Entry Section of the Directory is discovered to be full.

i. Output Buffer Overflow - Monitor Error IOPS 16

IOPS ASCII or IOPS Binary line exceeds 255_{10} words (including header pair).

j. Word Pair Count Error - Monitor Error IOPS 23

During an IOPS Mode transfer, the Word Pair Count is found to be less than 1 or greater than 177_8.

k. Too Many Files - Monitor Error IOPS 17

An excessive number of files are currently referenced.

l. Header Label Error - Monitor Error IOPS 40

During the processing of a .SEEK, the handler calculated file name is discovered to be different from the name present in the file header label.

m. Directory Format Error - Monitor Error IOPS 41

Illegal or meaningless data was found in the File Directory.

n. Accessibility Map Overflow - Monitor Error IOPS 42

During the processing of a .ENTER, the Accessibility Map is found to be full.

o. Directory Recording Error - Monitor Error IOPS 43

A write error is encountered during the recording of a Directory.

p. Logical EOT Found - Monitor Error IOPS 44

Logical End-Of-Tape encountered during the processing of a .SEEK or a .ENTER.

q. Long Input Record - Monitor Error IOPS 45

Input record read tape is too long to fit in the handler's buffer.

5.4.7 LPA. (LIne Printers LP15C and LP15F)

5.4.7.1 General Description - LPA. (311_{10} registers) is designed to operate Line Printers LP15 (132 columns) and LP15F (80 columns). Legal data modes are IOPS ASCII and Image Alphanumeric. Functions are as follows:

.INIT .WRITE .WAIT .WAITR .CLOSE

5.4.7.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard buffer size: (1) 54_{10} (LP15C) (2) 36_{10} (LP15F) NOTE .SCOM+4, bit 12 determines printer column size. 0 = 80 column 1 = 132 column b. .SETUP - API channel register 56_8 c. Output FORM Feed and determine if subsequent FORM Feeds should be issued every 57 lines. Bit 6 of the .INIT CAL is tested as follows: 0 = FORM Feed every 57 lines 1 = No FORM Feed Bit 6 is set by using a 5 rather than a 1 as the "F" argument of the .INIT (see 3.1.2).
.DLETE .RENAM .FSTAT	2	Ignore
.SEEK	3	Illegal Function - Monitor Error IOPS 6
.ENTER	4	Ignore
.CLEAR	5	Ignore
.CLOSE	6	a. Allow previous output to terminate. b. Output FORM Feed (if not inhibited in the .INIT) and allow it to terminate.
.MTAPE	7	Ignore
.READ	10	Illegal Function - Monitor Error IOPS 6
.WRITE	11	a. Allow previous output to terminate. b. Examine word 0 of the user's header word pair as follows:

Mnemonic	Code	Action
.WRITE (cont.)		

Bit	Signifies
0	0 = Enter Single Line Mode 1 = Enter Multiple Line Mode
1-8	Line count for Multiple Line Mode.
17	0 = IOPS ASCII Mode 1 = Image Alphanumeric Mode

NOTE

The user must explicitly set the data mode (Bit 17) in his line buffer as the handler does not examine the .WRITE macro for this information.

If in Multiple Line Mode, step "c" (below) is not performed.

c. Check the first character of the user's buffer for the following vertical form control characters, all of which are output by the FORTRAN IV Object Time System:

014	Form Feed
020	Overprint
021	Print every second line
012	Line Feed

To effect the Overprint function for FORTRAN users, it is necessary to simulate certain vertical form control characters. If the first character of a line is 012, 014, or 021, the handler automatically enters Multiple Line Mode (by setting bit 0 of the first word in the user's buffer to 1) and prints two lines, the first line being the vertical control character, and the second line being the actual data If the first character is 020 (Overprint), it is replaced in the user's buffer by 015 (Carriage Return) which does not affect the page position and both lines are printed. All other characters cause a Line Feed to be output from the handler's internal buffer followed by the line from the user's buffer. After output, any data in the user's buffer which was changed (i.e., header word 0 or the first data word) is restored.

If the user intends to output to another device from the same line buffer (e.g., two sequential .WRITEs), a .WAIT should be used after the .WRITE referencing the Line Printer to permit the restoration of any data which may have been replaced in the user's buffer by LPA.

Mnemonic	Code	Action
.WRITE (cont.)		d. Output in either Single Line Mode or Multiple Line Mode as applicable.
		e. Restore modified portions of the user's buffer (if changed).
.WAIT .WAITR	12	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).
.TRAN	13	Illegal Function - Monitor Error IOPS 6.

5.4.7.3 Legal Data Modes

a. IOPS ASCII (Mode 2)

b. Image Alphanumeric (Mode 3)

5.4.7.4 Carriage Control Characters - The following vertical control characters, except horizontal TAB, cause line termination except for special cases described under .WRITE (above).

Character	Action	
Line Feed (012_8)	Space one line	
VT (Vertical Tab, 013_8)	Space 20 lines	
Form Feed (014_8)	Move to top of form	
Carriage Return (015_8)	Reset column count to zero (no implicit LINE FEED function)	
DLE (040_8)	Space 30 lines	Refer to Appendix A for equivalent teleprinter characters
DC1 (041_8)	Space 2 lines	
DC2 (022_8)	Space 3 lines	
DC3 (023_8)	Space 1 line	
DC4 (024_8)	Space 10 lines	
ALT MODE (175_8)	Reset column count to zero (no implicit LINE FEED function)	
Horizontal TAB (011_8) (The horizontal control character Horizontal Tab does not terminate the line and may occur anywhere in the line.	Output sufficient number of spaces to position printer at column 9, 17, 25, ... etc.	

5.4.7.5 Recoverable Errors

Printer Not Ready - Monitor Error IOPS 4

a. Cause
   (1) Off Line
   (2) Out of Paper
   (3) Yoke Open
   (4) Alarm Status

b. Recovery - Ready the line printer and type CTRL R.

5.4.7.6 Unrecoverable Errors

a. Illegal Function - Monitor Error IOPS 6
   Attempt to execute .SEEK, .READ, or .TRAN

b. Illegal Data Mode - Monitor Error IOPS 7
   Attempt to output in Data Modes other than IOPS ASCII or Image Alphanumeric.

c. Line Over - Monitor Error IOPS 37
   The 81st or 133rd character (depending upon printer type) has been reached without encountering a vertical control character (Multiple Line Operation only).

d. Illegal Horizontal TAB - Monitor Error IOPS 47
   An attempt has been made to execute a Horizontal TAB (Multiple Line Mode only) causing the column count to exceed that required for the last tab stop (72 or 124 depending upon the printer type).

5.4.8 CDB. (CR03B Card Reader)

5.4.8.1 General Description - CDB. (415_{10} registers) is an IOPS ASCII handler which operates the CR03B Card Reader. The handler is supplied to the user in source form and, when assembled, operates with cards punched in 029 Hollerith Code. By defining the assembly parameter "DEC026+1", the handler can be assembled to accept cards punched in 026 Hollerith Code.

5.4.8.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard buffer size (52_{10}). b. .SETUP API channel register 55_8.

Mnemonic	Code	Action
.DLETE .RENAM .FSTAT	2	Ignore
.SEEK	3	Ignore
.ENTER	4	Illegal Function - Monitor Error IOPS 6
.CLEAR	5	Illegal Function - Monitor Error IOPS 6
.CLOSE	6	Allow previously requested input to terminate.
.MTAPE	7	Ignore
.READ	10	a. Allow previously requested input to terminate. b. Check that device is ready. c. Input next card.
.WRITE	11	Illegal Function - Monitor Error IOPS 6
.WAIT .WAITR	12	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).
.TRAN	13	Illegal Function - Monitor Error IOPS 6

5.4.8.3 Legal Data Modes

IOPS ASCII (Mode 2)

Eighty card columns are read and interpreted as 029 (or 026) Hollerith data, mapped into the corresponding 64-graphic subset of ASCII, and stored in the user's line buffer in 5/7 format (36_{10} locations are required to store an 80 column card). 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 user's buffer). In addition, a Carriage RETURN (015_8) character is appended to the input line; thus, a total of 81 characters are returned to the user.

All illegal punch configurations (i.e., those not appearing in the 029 or 026 character set, as applicable) are interpreted as validity errors and will cause an IOPS 4 error condition. The card containing the error must be repunched.

In addition to the Hollerith character set, the handler recognizes the ALT MODE terminator (necessary for system programs). ALT MODE, recognized as a 12-1-8 code (multiple-punched A8), is mapped in to the standard ALT MODE character (175_8) in the user's buffer.

Each file must be terminated with an EOF card (all punch positions in card column 1 perforated), which may be created by multiple-punching characters: +-0123456789.

When a card has been processed, word 0 of the header word pair is constructed and stored in the user's line buffer. Word 1 of the header (checksum word) is not changed.

Refer to Appendix B for a listing of legal Hollerith codes and corresponding ASCII graphics.

5.4.8.4 Recoverable Errors

Reader Not Ready - IOPS 4

a. Causes
   (1) Hopper Empty
   (2) Stacker Full
   (3) Feed Check (may be hardware failure)
   (4) Read Check (may be hardware failure)
   (5) STOP button depressed
   (6) START button not depressed
   (7) End of card deck. Add more cards or EOF card.
   (8) Validity Error (VALIDITY switch ON) - unrecognizable punch configuration.
   (9) Pick Fail - Card selected, but not passed from hopper to read station.

b. Recovery
   Remedy error condition and type CTRL R.

5.4.8.5 Unrecoverable Errors

a. Illegal Function - Monitor Error IOPS 6
   An attempt was made to execute a .ENTER, .CLEAR, .WRITE, or a .TRAN.

b. Illegal Data Mode - Monitor Error IOPS 7
   A request for transfer was made in a data mode other than IOPS ASCII.

5.4.9 VPA. (VP15A Storage Tube Display)

5.4.9.1 General Description

5.4.9.1 General Description - VPA. (612_{10} registers) operates the VP15A Storage Tube Display. Legal data modes are IOPS ASCII, Image Alphanumeric, and Dump. Handler functions are as follows:

.INIT, .WRITE, .WAIT, .WAITR, .FSTAT, .CLOSE

5.4.9.2 Functions

Mnemonic	Code	Action
.INIT	1	a. Return standard line buffer size (34_{10}) b. .SETUP API channel register (54_8) c. Set X and Y coordinates to position the beam at the top left corner of the screen (one line above the first visible line). d. Set I/O UNDERWAY indicator. e. Erase the screen
.RENAM .DLETE	}	Ignore
.FSTAT	2	Check to see if file-oriented.
.SEEK	3	Illegal Function - Monitor Error IOPS 6
.ENTER	4	Ignore
.CLEAR	5	Ignore
.CLOSE	6	Allow previous output to terminate.
.MTAPE	7	Ignore
.READ	10	Illegal Function - Monitor Error IOPS 6
.WRITE	11	a. Set I/O UNDERWAY indicator b. Allow previous output to terminate c. Display data
.WAIT .WAITR	12	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).

5.4.9.3 Legal Data Modes

IOPS ASCII (Mode 2) Scale 2

IOPS ASCII (Mode 12) Scale 4

Image Alphanumeric (Mode 3)

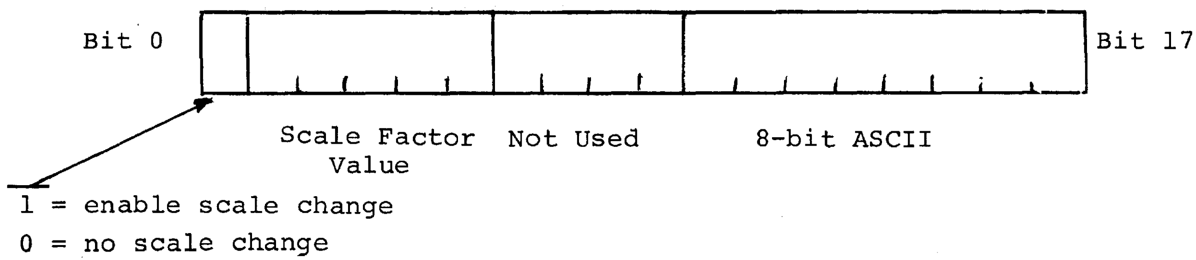
Dump (Mode 4) Store Mode

Dump (Mode 14) Non-store Mode

5.4.9.4 Data Mode Functions

a. IOPS ASCII (Modes 2 and 12) - These data modes allow 5/7 ASCII to be displayed from the addressed line buffer. Header word pair and word pair count must be supplied. Data Mode 2 displays characters using a scale of 2. Data Mode 12 displays characters using a scale of 4.

b. Image Alphanumeric (Data Mode 3) - This data mode allows 7 or 8 bit ASCII stored one character per word in the addressed line buffer to be displayed. Header word pair and word pair count must be supplied. Characters may be displayed at any legal scale (1 - 31_{10}). Each data word may be used to specify a different scale factor, as shown below. If bit 0 is set to 1, the handler determines a new scale factor from bits 1 - 5. If bit 0 is set to 0, bits 1 - 5 are ignored and the previous scale factor is used.



VPA Image ASCII Word Structure

c. Dump (Data Modes 4 and 14) - These data modes allow one point for each data word in the addressed line buffer to be displayed (no header word pair required). Each data word in the buffer is treated as two 9-bit coordinates which describe the location of a point. Bits 0 through 8 represent the X coordinate value while bits 9 through 17 represent the Y coordinate value. Data Mode 4 selects Store Mode and Data Mode 14 selects Non-store Mode which, during assembly, causes Bit 5 of the first word of the .WRITE macro expansion to be set either to 0 (Store Mode) or 1 (Non-store Mode). Points plotted in Store Mode will remain visible for periods up to 15 minutes. Points plotted in Non-store Mode, however, must be refreshed at least 30 times per second to remain visible. This feature is particularly useful for repeatedly displaying a small movable figure such as a cursor. Also, a single Non-store point may be utilized for setting a starting point for ASCII text or Store Mode plots.

5.4.9.5 Special Characters

a. A Carriage RETURN terminates an output character string and automatically initiates a Carriage RETURN/LINE FEED sequence (IOPS only).

b. An ALT MODE terminates an output character string but does not alter the beam position (IOPS only).

c. LINE FEED moves the beam down one line (horizontal position not affected).

d. Horizontal TAB causes a sufficient number of spaces to be output to place the beam in character positions 9, 17, 25, ...70.

e. FORM Feed erases the screen and repositions the beam to the first character position of the first line. It is not a legal terminator and may appear at the beginning of a line.

5.4.9.6 Printing Rules

a. When using a Scale Factor of 2 (default assumption in IOPS ASCII), the VP15A displays 72 characters per line and 56 lines per "page".

b. If the screen has been filled with 56 lines, a subsequent IOPS ASCII .WRITE command will cause the display to be erased and the new line to be displayed at the top of the screen.

c. If the beam has been positioned at the bottom line of the screen by a Dump Mode (non-store) .WRITE and two subsequent ASCII .WRITEs are issued, and the second ASCII .WRITE will cause the display to be erased as in "b" above.

d. When using Image ASCII Mode, the user must set the starting point for the first line to be output after device initialization (.INIT). This may be accomplished either by issuing a Dump Mode .WRITE referencing the desired starting point, or by including a LINE FEED as the first character in the line buffer (first word after the header word pair).

5.4.9.7 Unrecoverable Errors

a. Illegal Function - Monitor Error IOPS 6

An attempt to execute a .SEEK or .READ has been detected.

b. Illegal Data Mode - Monitor Error IOPS 7

See 5.4.9.3.

APPENDIX A

PDP-15 IOPS ASCII CHARACTER SET

Listed below are the ASCII characters interpreted by the ADVANCED 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 (CTRL A)	!	A		1
2		"	B		2
3	ETX (CTRL C)	#	C		3
4	EOT (CTRL D)	$	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		/	O		17
20	DLE (CTRL P)	0	P		20
21	DC1 (CTRL Q)	1	Q		21
22	DC2 (CTRL R)	2	R		22
23	DC3 (CTRL S)	3	S		23
24	DC4 (CTRL T)	4	T		24
25	NACK (CTRL U)	5	U		25
26		6	V		26
27		7	W		27
30	CNCL (CTRL X)	8	X		30
31		9	Y		31
32	SS (CTRL Z)	:	Z		32
33	ESC (ALTMODE)	;			33
34		<			34
35		=		ESC (ALTMODE)	35
36		>	∧ or ↑	ESC (ALTMODE)	36
37		?	← or _ (underscore)	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-15 ASCII/HOLLERITH CORRESPONDENCE

ASCII		HOLLERITH		ASCII		HOLLERITH	
CHAR.	7-BIT CODE	DEC 029	DEC 026	CHAR.	7-BIT CODE	DEC 029	DEC 026
SP	40	None	None	@	100	4-8	8-4
!	41	11-2-8	12-8-7	A	101	12-1	12-1
"	42	7-8	0-8-5	B	102	12-2	12-2
#	43	3-8	0-8-6	C	103	12-3	12-3
$	44	11-3-8	11-8-3	D	104	12-4	12-4
%	45	0-4-8	0-8-7	E	105	12-5	12-5
&	46	12	11-8-7	F	106	12-6	12-6
,	47	5-8	8-6	G	107	12-7	12-7
(	50	12-5-8	0-8-4	H	110	12-8	12-8
)	51	11-5-8	12-8-4	I	111	12-9	12-9
*	52	11-4-8	11-8-4	J	112	11-1	11-1
+	53	12-6-8	12	K	113	11-2	11-2
'	54	0-3-8	0-8-3	L	114	11-3	11-3
–	55	11	11	M	115	11-4	11-4
.	56	12-3-8	12-8-3	N	116	11-5	11-5
/	57	0-1	0-1	O	117	11-6	11-6
0	60	0	0	P	120	11-7	11-7
1	61	1	1	Q	121	11-8	11-8
2	62	2	2	R	122	11-9	11-9
3	63	3	3	S	123	0-2	0-2
4	64	4	4	T	124	0-3	0-3
5	65	5	5	U	125	0-4	0-4
6	66	6	6	V	126	0-5	0-5
7	67	7	7	W	127	0-6	0-6
8	70	8	8	X	130	0-7	0-7
9	71	9	9	Y	131	0-8	0-8
:	72	2-8	11-8-2	Z	132	0-9	0-9
;	73	11-6-8	0-8-2	[	133	12-2-8	11-8-5
<	74	12-4-8	12-8-6	-	134	11-7-8	8-7
=	75	6-8	8-3	]	135	0-2-8	12-8-5
>	76	0-6-8	11-8-6	↑ or Λ	136	12-7-8	8-5
?	77	0-7-8	12-8-2	← or — (underscore)	137	0-5-8	8-2

NOTES: 1. ASCII code 0–37 and 140–177 have no corresponding codes in the Hollerith set and therefore are not shown.

2. ALT Mode is simulated by 12–1–8 punch.

3. The card reader interface actually supplies a direct binary equivalent of the column punch. The octal codes given above are those generated by the handler from the column punches.

APPENDIX C

ADVANCED MONITOR ERROR PRINTOUTS

Errors	Explanation
BAD DEV - 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 PROGNAM	Non-existent program name. Command ignored.
PERM .DAT SLOT	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 - TRY AGAIN	Last command types 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 DEF	Batch device was not designated properly. Should be: CD - for card reader PR - for paper tape reader

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 was not set up by an ASSIGN command.

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 of paper tape. 3. Line printer - off line. 4. Card reader - off line, out of cards, stacks full, or card jam.
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 MTAPE command.

Error Code	Error	Error Data	Comments
12	Unrecoverable DECtape error (MARK TRACK)	DECtape status register B and Unit Number	DECtape error with status register B in bits 0 to 11 and the unit number in bits 15 to 17. Reformat tape.
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	Reserved		
21	Reserved		
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	Reserved		
25	Reserved		
26	Reserved		
27	Reserved		
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	Reserved		
40	Header label errors	CAL address	The internal header label for the currently opened file is incorrect.
41	Directory format error	CAL address	Bad data in file directory.
42	Accessibility Map overflow	CAL address	Too many files recorded in the current MAGtape. Use MTDUMP to retrieve storage occupied by unwanted files.
43	Reserved		
44	Logical EOT	CAL address	An unexpected logical end-of-tape was encountered during the processing of .SEEK or .ENTER.
45	Long Input Record	CAL address	The record read from tape is too large to be accommodated by the handler's buffer.
46	Attempt to delete System File	CAL address	The user has requested deletion of a file whose extension is in "SYS".
47	Illegal Horizontal Tab	CAL address	The line printer received a Horizontal Tab after the 72nd or 128th character (depending on the model). The remainder of the line is lost.
61	Parity Error while reading Directory or File Bit Map Blocks	CAL address	Defective data. DECtape or DECdisk drive (see NOTE below).

NOTE

Recovery procedures:

1. Repeat operation which caused error.
2. Remount DECtape on another drive and repeat step 1.
3. If you are very familiar with DECtape file structure and have a reasonably current directory, proceed as follows:
   a. Use PIP to Block Copy each file on the defective tape to a fresh tape (directory provides starting block number of each file).
   b. Use PATCH to reconstruct a new directory on the new tape (do not write on this tape - it has no file bit maps).
   c. Use PIP to transfer each reconstructed file to another tape (to reconstruct the file bit maps).

APPENDIX F

SUMMARY OF KEYBOARD COMMANDS FOR THE ADVANCED MONITOR ENVIRONMENT

SYSTEM PROGRAM LOAD COMMANDS

Command	System Program Loaded
F4	FORTRAN IV Compiler
F4I	8K FORTRAN IV Compiler (DECtape I/O only)
MACRO	MACRO-15 Assembler
MACROI	8K MACRO Assembler (DECtape I/O only)
PIP	Peripheral Interchange Program
EDIT	Symbolic Text Editor
EDITVP	Symbolic Text Editor using VP15A Display
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	Program which permits the creation of a system of core overlays.
EXECUTE (E)	Control program to load and supervise core residency during the execution of a CHAIN-built overlay system.
SGEN	System Generation program
SRCCOM	Source Compare program
DTCOPY	8K High-speed DECtape Copy 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 any time -- 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.
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.

SPECIAL FUNCTION COMMANDS

Command	Action
API OFF	Disables API.
API ON	Enables API.
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
CHANNEL (or C) 7/9	This command establishes whether the default condition for magnetic tape operation is to be 7-channel or 9-channel.
DIRECT (or D)n	Lists the directory of the System Device unit n (0-6).
GET (or G)n	Restores core image from the system device medium, if available, on unit n (0-7).
GET (or G)n address	Restores core image from the system device medium, if available, on unit n and restarts at specified address.
GET (or G)n HALT (or H)	Restores core image from the system device medium, if available, on unit n and halts.
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.
LOG (or L)	Can be followed by any comment and terminated by ALT MODE.
NEWDIR (or N)n	Writes empty directory onto the system device, unit n (units 1-7 only).
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.
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↵
SCOM (or S)	Causes typeout of system configuration information, including available device handlers.
X4K ON	Permits the Monitor and programs run under the Monitor to use an extra 4K of core (i.e., 12K, 20K, 28K).
X4K OFF	Terminates use of extra 4K page of core.
33TTY ON	Permits the Monitor to properly interface to a Model 33 Teletype unit (convert tabs to spaces).
33TTY OFF	Permits the Monitor to interface to a Model 35 or 37 Teletype unit.

APPENDIX G

OPTIONAL ADVANCED SOFTWARE

PAGE/BANK MODE SYSTEM PROGRAMS

The differences between the Page and Bank mode System Programs, as they exist in the V5A version of the ADVANCED Monitor Software System, are described in the following paragraphs. If no description of a System Program is given in the following, then the program is the same in both the Page and Bank mode systems.

Keyboard Monitor

The ADVANCED Keyboard Monitor for the Bank mode system (identified as KM9-15 V5A) operates exclusively in the Bank mode. This monitor has an EEM instruction in location 1 of the program and the needed JMP to the skip chain in location 2. The Keyboard monitor for the Page mode has the JMP instruction in location 2 since, in this mode, the PDP-15 is always in the extended mode.

System Loader - Bank Mode

In a Bank mode system, the System Loader (.SYSLD) loads all programs (both handlers and System programs) in Bank mode. Specifically, the .SYSLD, DDT and EXECUTE programs operate in Bank mode as do the user programs which they load.

CHAIN (V5A) and EXECUTE (V4A)

The CHAIN System program assumes, unless otherwise instructed, that the program to be built will run in Bank mode (Page mode option is not used). The program EXECUTE, loads and runs overlay systems in Bank mode only. EXECUTE itself runs in Bank mode. Program units 8K or smaller may be handled.

89TRAN

The Bank mode system contains a relocatable binary for the translation of PDP-8 assembly language to PDP-9 assembly language. Users of PDP-15 systems should delete this program since the 8 to 15 translator program (8TRAN) is also on the tape.

BANK MODE RB09 DISK SYSTEM

In order to generate a Bank mode system which will utilize the RB09 disk as the system device, it is necessary to make patches (PATCH) to the standard system and to insert (UPDATE) the RB disk handlers into the system library. The following procedure must be performed:

A. Use Patch to change the 4 locations as follows:

```
$A DTAØ -14)
$PATCH)

PATCH V7A
>B 42)
>L 3)
ØØØØ3/7Ø7ØØ1>7Ø7121<ALTMODE>    /IOT IN THE SKIP CHAIN.

>B 52)
>L 3)                            /MAKE BIT 1Ø=1
ØØØØ3/3Ø112Ø>3Ø132Ø<ALTMODE>    /IN .SCOM+4 IN
>L 127)                          /SGNBLK.
ØØ127/7Ø7ØØ1>7Ø7121<ALTMODE>    /CHANGE DISK SKIP
                                 /IOT IN SGNBLK.
>KM9-15)
>L 1Ø4)                          /CHANGE BIT 1Ø TO 1
ØØ1Ø4/3Ø112Ø>3Ø132Ø<ALTMODE>    /IN .SCOM+4 IN MONITOR
>EXIT)
```

B. Use the following UPDATE procedure to put the RB Disk handlers into the System Library.

NOTE

The peripheral tape should be on DECtape unit 1 and a clean scratch tape on unit 2.

```
$A DTAØ -14/DTA1 -1Ø/DTA2 - 15)
$UPDATE)
UPDATE V8A
>US←<ALTMODE>
>R RFC.,DKC.)                   /INSERT THE RB DISK
>R RFA.,DKA.)                   /HANDLERS AND
>R RFB.,DKB.)                   /DELETE THE RF DISK
>R RFD.,DKD.)                   /HANDLERS.
>D RFE.)
>D RFF.)
>C)
```

On completion of the above,use PIP to replace the Library on the system tape with the one just generated.

```
$PIP)
PIP V13A
>D DTØ .LIBR BIN)
>T DTØ←DT2 .LIBR BIN)
>V DTØ .LIBR BIN<ALTMODE>
```

C. It is now possible to do a system generation from the modified DECtape system to the Disk. Use SGEN. with DTAØ assigned to -14 and DKDØ to -15, to create a system that will reside on disk unit Ø.

NOTE

> If the special Keyboard Monitor (KMS915 is to be used, it should be placed on the DECtape system tape first (step A only of the procedure) and then the above procedure followed if a RB Disk system is desired.

KMS15: SPECIAL ADVANCED MONITOR

The special ADVANCED monitor, located on the V5A Peripheral tape, contains a Teletype handler with overprint capability for use with FORTRAN IV programs. Although aimed at users with 12K or greater systems, KMS can be used in 8K; it is 24_{10} words longer than the standard monitor.

Updating KMS15 Into the Standard V5A System

The standard monitor is first replaced with KMS15 (or KMS915 for the Bank mode version) using PATCH. Once the monitor is on the tape, it is necessary to do a system generation to restore .DAT slot information and to tie the device handlers to monitor text strings in the SCOM command.

The following example illustrates the entire process for updating KMS:

```
A.      KM15 V5A
        $A DTØ -14/DT1 -1Ø)          (SYSTEM ON UNIT Ø, PERIPHERAL
        $PATCH)                       TAPE ON UNIT 1)

        PATCH V7A
        >KM15)                       (KMS915 ON THE BANK-MODE TAPE)
        >READ KMS15)                  (KMS915 ON THE BANK-MODE TAPE)
        >EXIT)

B.      KMS15 V5A
        $A DTØ -14,-15)
        $SGEN)

        SGEN V4A
        NEW SYSTEM? N
        MODIFY SYSTEM ON DTØ
        API? (N) N
        33 TTY? (N) N
        A.  ALTER I/O DEVICES OR HANDLERS ? N
        B.  DISPLAY SKIP CHAIN? N
            CHANGE SKIP CHAIN ORDER? N
            7 CHANNEL MAGTAPE? (Y) Y
            LINE PRINTER LINE SIZE (8Ø,12Ø,132) [8Ø]
        C.  DISPLAY .DAT SLOTS? N
            ALTER .DAT SLOTS > (AS DESIRED BY THE USER)
            SGEN COMPLETE
```

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12

Digital Software News for the PDP-11

Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library. Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning DEC software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms which are available without charge from the Program Library, should be fully filled out and accompanied by Teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

New and revised software and manuals, Software Performance Report forms, and software price lists are available from the Program Library. When ordering, include the document number and a brief description of the program or manual requested. Revisions of programs and documents will be announced in the newsletters. Direct all inquiries and requests to:

Program Library
Digital Equipment Corporation
146 Main Street, Bldg. 1-2
Maynard, Massachusetts 01754

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

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.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

Did you find errors in this manual?

How can this manual be improved?

DEC also strives to keep its customers informed of current DEC software and publications. Thus, the following periodically distributed publications are available upon request. Please check the appropriate boxes for a current issue of the publication(s) desired.

- ☐ Software Manual Update, a quarterly collection of revisions to current software manuals.
- ☐ User's Bookshelf, a bibliography of current software manuals.
- ☐ Program Library Price List, a list of currently available software programs and manuals.

Please describe your position. ____________

Name ____________ Organization ____________

Street ____________ Department ____________

City ____________ State ____________ Zip or Country ____________

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 Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754