DEC-10-MRRC-D

decsystem10
MONITOR CALLS

This manual reflects the software as of the 5.06 release of the monitor.

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing June 1971
2nd Printing (Rev) January 1972
3rd Printing (Rev) June 1972
4th Printing (Rev) March 1973

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

Actual distribution of the software described in this specification will be subject to terms and conditions to be announced at some future date by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

The software described in this manual is furnished to purchaser under a license for use on a single computer system and can be copied (with inclusion of DEC's copyright notice) only for use in such system, except as may otherwise be provided in writing by DEC.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

ILLUSTRATIONS

TABLES

TABLES (Cont)

ALPHABETICAL LIST OF MONITOR CALLS

FOREWORD

DECsystem-10 Monitor Calls is a complete reference document describing the monitor programmed operators (UUOs) and is intended for the experienced assembly language programmer. The information presented in this manual reflects the 5.06 release of the monitor. The monitor calls are grouped in a manner that facilitates easy learning, and once they are mastered, the user can refer to the end of the Table of Contents and to the index for an alphabetical list of the UUOs.

DECsystem-10 Monitor Calls does not include reference material on the operating system commands. This information can be found in DECsystem-10 Operating System Commands (DEC-10-MRDC-D). Included in DECsystem-10 Operating System Commands are discussions on commands processed by both the monitor command language interpreter and the programs in the Batch system. The two manuals, DECsystem-10 Monitor Calls and DECsystem-10 Operating System Commands, supersede the Time-sharing Monitors manual (DEC-T9-MTZD-D) and all of its updates.

A third manual, Introduction to DECsystem-10 Software (DEC-10-MZDA-D), is a general overview of the DECsystem-10. It is written for the person, not necessarily a programmer, who knows computers and computing concepts and who desires to know the relationship between the various components of the DECsystem-10. This manual is not intended to be a programmer's reference manual, and therefore, it is recommended that it be read at least once before reading the above-mentioned reference documents.

SYNOPSIS OF DECsystem-10 MONITOR CALLS

Chapter 1 discusses the format of memory and briefly describes the job data area. Chapter 2 introduces all of the monitor programmed operators available to a user program and the various processor modes in which a user program operates. The UUOs available for non-I/O operations are presented in Chapter 3. These programmed operators are used to obtain execution, core, and segment control; program identification; environmental information; and real-time status. An introduction to I/O programming is presented in Chapter 4; the services the monitor performs for the user and how the user

program obtains these services are also discussed. I/O programming specific to the nondirectory devices and directory devices is explained in Chapters 5 and 6, respectively. Algorithms of the monitor, described in Chapter 7, give the user an insight into system operation. The appendices contain supplementary reference material and tables.

CONVENTIONS USED IN DECsystem-10 MONITOR CALLS

The following conventions have been used throughout this manual:

dev:	Any logical or physical device name. The colon must be included when a device is used as part of a file specification.
list	A single file specification or a string of file specifications. A file specification consists of a filename (with or without a filename extension), a device name, a directory name, and a protection.
job n	A job number assigned by the system.
file.ext	Any legal filename and filename extension.
core	Decimal number of 1K blocks of core (KA10). Decimal number of pages of core (KI10).
adr	An octal address.
C(adr)	The contents of an octal address.
[proj,prog]	Project-programmer numbers; the square brackets must be included in the command string.
fs	Any legal file structure name or abbreviation.
(ESC)	The symbol used to indicate the ESCAPE Key.
tx	A control character obtained by depressing the CTRL key and then the character key x.
←	A back arrow used in command string to separate the input and output file specifications.
=	A equal sign used in a command string to separate the input and output file specifications.
*	The system program response to a command string.
.	The monitor response to a command string.
(RET)	The symbol used to indicate that the user should depress the RETURN key. This key must be used to terminate every command to the monitor command language interpreter.
____	Underscoring used to indicate computer typeout.
n	A decimal number.
[directory]	A designation identifying a particular disk area. This designation can be in the form [proj, prog] which identifies a UFD or [proj,prog,sfd, sfd,...] which identifies a sub-file directory path branching from a UFD. The square brackets are required.

CHAPTER 1
MEMORY FORMAT

1.1 MEMORY PROTECTION AND RELOCATION

Each user program is run with the processor in a special mode called the user mode; in this mode the program must operate within an assigned area in core, and certain operations are illegal. Because every user has an assigned area in core, the rest of core is unavailable to him. He cannot gain access to a protected area for either storage or retrieval of information.

The assigned area of each user can be divided into two segments. If this is the case, the low segment (impure segment) is unique for a given user and can be used for any purpose. The high segment (pure segment) can be used by one user or it can be shared by many users. If the high segment is shared, the program is a reentrant program. The monitor usually write-protects the high segment so that the user cannot alter its contents. This is done, for example, when the high segment is a pure procedure to be used reentrantly by many users. One high pure segment can be used with any number of low impure segments. Any user program that attempts to write in a write-protected high segment is aborted and receives an error message. If the monitor defines two segments but does not write-protect the high segment, the user has a two-segment non-reentrant program (refer to Paragraph 3.2.4).

The DECsystem-10 monitor defines the size and position of a user's area. On KA10-based systems (DECsystem-1040, -1050, and -1055), the monitor uses relocation by specifying protection and relocation addresses for the low and high segments. The protection address is the maximum relative address the user can reference. The relocation address is the absolute core address of the first location in the segment, as seen by the monitor in the hardware. The monitor defines these addresses by loading four 8-bit registers (two 8-bit registers in a KA10 based system with the KT10 option instead of the KT10A option), each of which correspond to the left eight bits of an 18-bit PDP-10 address. Thus, segments always contain a multiple of 1024 words.

On KI10-based systems (DECsystem-1070 and -1077), the user's area is page mapped. This means that each page (a page consists of 512 words) of the user's area is associated with a page of physical core memory. Because the assignment of physical pages of core does not need to be contiguous, the monitor has greater freedom in allocating core. The monitor associates (maps) pages in the user's area with physical pages in core in such a way that the user appears to have one or two segments as with a KA10-based system. Therefore, in most cases, the user does not need to be concerned with the type

of processor on which his program is running. However, the unit of core allocation is different on the two processors. The unit of allocation on the KA10 is 1024 words (1K) and on the KI10, is 512 words (1 page).

In general, the term mapping refers to both relocation (KA10) and page mapping (KI10). On either processor, a user address is called a relative or virtual address before it is mapped, and an absolute or physical address after it is mapped.

To take advantage of the fast accumulators, memory addresses $0-17_8$ are not mapped and all users have access to the accumulators. Therefore, relative locations $0-17_8$ cannot be referenced by a user's program. The monitor saves the user's accumulators in this area when the user's program is not running and while the monitor is servicing a UUO from the user. Refer to the PDP-10 System Reference Manual for a more complete description of the relocation and mapping hardware.

1.2 USER'S CORE STORAGE

A user's core storage consists of blocks of memory, the sizes of which are an integral multiple of 1024 (2000_8) words on the KA10-based system and 512_{10} (1000_8) words on the KI10-based system. In a non-reentrant monitor, the user's core storage is a single contiguous block of memory. After mapping, the first address in a block is a multiple of 2000_8 or 1000_8. The relative user and relocated address configurations on the KA10 are shown in Figure 1-1, where P_L, R_L, P_H, and R_H are the protection and relocation addresses for the low and high segments, respectively. If the low segment is more than half the maximum memory capacity ($P_L \geq 400000$), the high segment starts at the first location after the low segment (at P_L + 2000). The high segment is limited to 128K. The relative user address configurations on the KI10 are shown in Figure 1-2, where P_L and P_H are the protection addresses for the low and high segments, respectively.

Two methods are available to the user for loading his core area. The simplest way is to load a core image stored on a retrievable device (refer to RUN and GET commands). The other method is to use the relocatable binary loader to link-load binary files. The user can then write the core image on a retrievable device for future use (refer to SAVE command).

1.2.1 Job Data Area (JOBDAT)

The first 140 octal locations of the user's core area are always allocated to the job data area (refer to Table 1-1). Locations in this area are given mnemonic assignments where the beginning characters are .JB. The job data area provides storage for specific information of interest to both the monitor and the user. Some locations, such as .JBSA and .JBDDT, are set by the user's program for use by the monitor. Other locations, such as .JBREL, are set by the monitor and are used by the user's program. In particular, the right half of .JBREL contains the highest legal address set by the monitor when the user's core allocation changes.

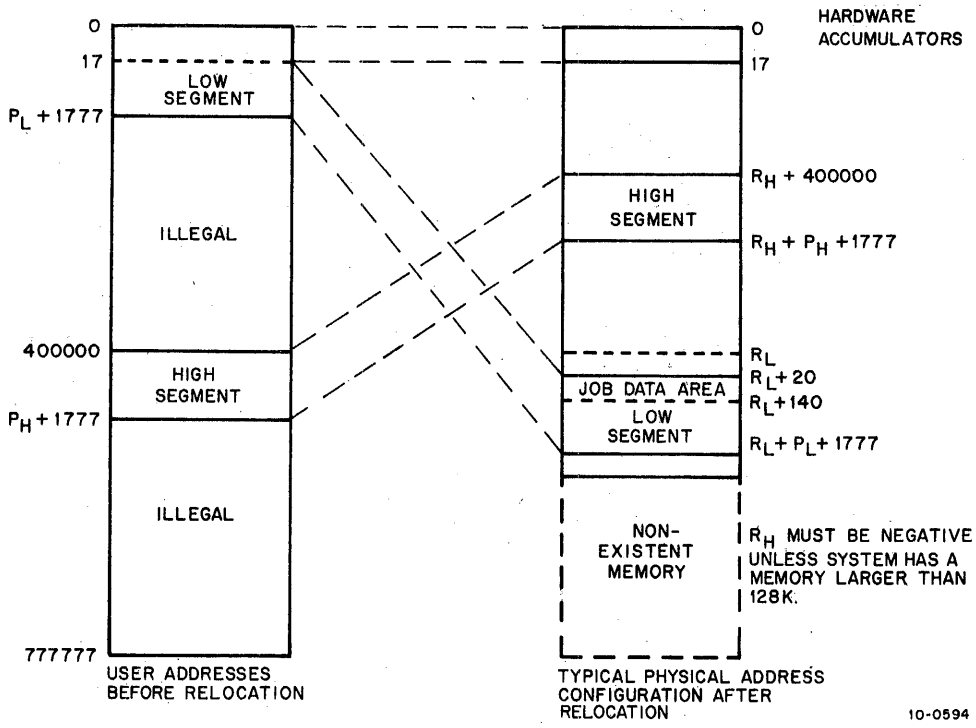


Figure 1-1 KA10 User Address Mapping

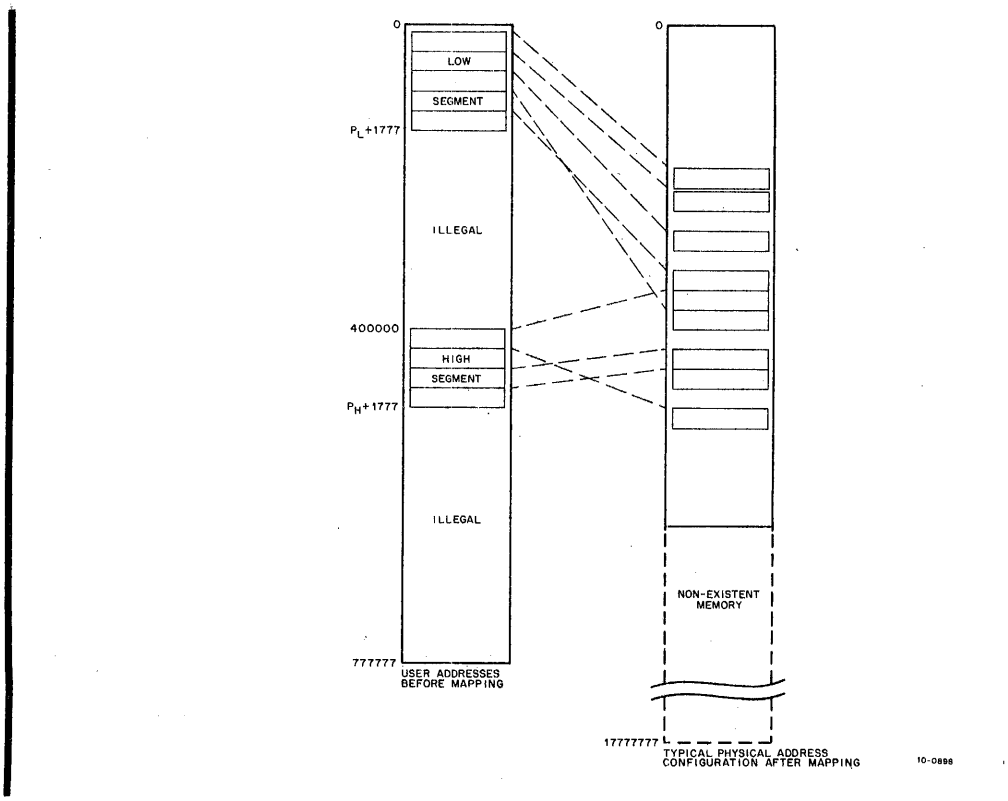


Figure 1-2 K110 User Address Mapping

Table 1-1
Job Data Area Locations
(for user-program reference)

Name	Octal Location	Description
.JBUUO	40	User's location 40₈. Used by the hardware when processing user UUOs (001 through 037) for storing op code and effective address.
.JB41	41	User's location 41₈. Contains the beginning address of the user's programmed operator service routine (usually a JSR or PUSHJ).
.JBERR	42	Left half: Unused. Right half: Accumulated error count from one system program to the next. System programs should be written to look at the right half only.
.JBREL	44	Left half: Zero. Right half: The highest relative core location available to the user (i.e., the contents of the memory protection register when this user is running).
.JBBLT	45	Three consecutive locations when the LOADER puts a BLT instruction and a CALLI UUO to move the program down on top of itself. These locations are destroyed on every executive UUO by the executive pushdown list.
.JBDDT	74	Left half: The last address of DDT. Right half: The starting address of DDT. If contents are 0, DDT has not been loaded.
.JBCN6	106	Six temporary locations used by the CHAIN program (refer to the LOADER manual) after it releases all I/O channels. .JBCN6 is defined to be in .JBJDA.
.JBPFI	114 (value)	All user I/O must be to locations greater than .JBPFI.
.JBHRL	115	Left half: First relative free location in the high segment (relative to the high segment origin so it is the same as the high segment length). Set by the LOADER and subsequent GETs, even if there is no file to initialize the low segment. The left half is a relative quantity because the high segment can appear at different user origins at the same time. The SAVE command uses this quantity to know how much to write from the high segment. Right half: Highest legal user address in the high segment. Set by the monitor every time the user starts to run or does a CORE or REMAP UUO. The word is ⩾401777 unless there is no high segment, in which case it will be zero. The proper way to test if a high segment exists is to test this word for a non-zero value.

Table 1-1 (Cont)
Job Data Area Locations
(for user-program reference)

Name	Octal Location	Description
.JBSYM	116	Contains a pointer to the symbol table created by linking loader. Left half: Negative of the length of the symbol table. Right half: Lowest address used by the symbol table.
.JBUSY	117	Contains a pointer to the undefined symbol table created by linking loader or defined by DDT. This location has the same format as .JBSYM. There are no undefined symbols if the contents is ≥ 0.
.JBSA	120	Left half: First free location in low segment (set by loader). Right half: Starting address of the user's program.
.JBFF	121	Left half: Zero. Right half: Address of the first free location following the low segment. Set to C $(.JBSA)_{LH}$ by RESET UUO.
.JBREN	124	Left half: Unused. Right half: REENTER starting address. Set by user or by loader and used by REENTER command as an alternate entry point.
.JBAPR	125	Left half: Zero. Right half: Set by user program to trap address when user is enabled to handle APR traps such as illegal memory, pushdown overflow, arithmetic overflow, and clock. See APRENB UUO.
.JBCNI	126	Contains state of APR as stored by CONI APR when a user-enabled APR trap occurs.
.JBTPC	127	Monitor stores PC of next instruction to be executed when a user-enabled APR trap occurs.
.JBOPC	130	The previous contents of the job's last user mode program counter are stored here by monitor on execution of a DDT, REENTER, START, or CSTART command. After a user program HALT instruction followed by a START, DDT, CSTART, or REENTER command, .JBOPC contains the address of the HALT. To proceed at the address specified by the effective address, it is necessary for the user or his program to recompute the effective address of the HALT instruction and to use this address to start. Similarly, after an error during execution of a UUO followed by a START, DDT, CSTART, or REENTER command, .JBOPC points to the address of the UUO. For example, in DDT to continue after a HALT, type .JBOPC/10000,,3010 JRST @ .$X

Table 1-1 (Cont)
Job Data Area Locations
(for user-program reference)

Name	Octal Location	Description
.JBCHN	131	Left half: Zero or the address of first location after first FORTRAN IV loaded program. Right half: Address of first location after first FORTRAN IV Block Data.
.JBCOR	133	Left half: Highest location in low segment loaded with non-zero data. No low file written on SAVE or SSAVE if less than 140. Set by the LOADER. Right half: User argument on last SAVE or GET command. Set by the monitor.
.JBINT	134	Left half: Reserved for the future. Right half: Zero or the address of the error-intercepting block (refer to Paragraph 3.1.3.2).
.JBOPS	135	Reserved for all operating systems.
.JBCST	136	Reserved for customers.
.JBVER	137	Program version number. The bits are defined as follows: Bits 0-2 The group who last modified the program 0 = Digital development group. 1 = Other Digital employees. 2-4 = Reserved for customers. 5-7 = Reserved for customer's users. Bits 3-11 Digital's major version number. Usually incremented by 1 after a release. Bits 12-17 Digital's minor version number. Usually 0, but may be used if an update is needed after work has begun on a new major version. Bits 18-35 Edit number. Usually not reset. The VERSION and the SET WATCH VERSION commands output the version number in standard format. Refer to DECsystem-10 Operating System Commands.
.JBDA	140	The value of this symbol is the first location available to the user.

NOTE: Only those JOBDAT locations of significant importance to the user are given in this table. JOBDAT locations not listed include those that are used by the monitor and those that are unused at present. User programs should not refer to any locations not listed above because such locations are subject to change.

JOBDAT is loaded automatically, if needed, during the loader's library search for undefined global references, and the values are assigned to the mnemonics. JOBDAT exists as a .REL file on device SYS: for loading with user programs that symbolically refer to the locations. User programs should reference locations by the assigned mnemonics, which must be declared as EXTERN references to the assembler. All mnemonics in this manual with a .JB prefix refer to locations in the job data area.

1.2.2 Vestigial Job Data Area

A few constant data in the job data area may be loaded by a two-segment, one-file program without using instructions on a GET command (.JB41, .JBREN, .JBVER), and some locations are loaded by the monitor on a GET (.JBSA, .JBCOR, .JBHRL). The vestigial job data area (the first 10 locations of the high segment) is reserved for these low-segment constants; therefore, a high-segment program is loaded into 400010 instead of 400000 (refer to Table 1-2). With the vestigial job data area in the high segment, the loader automatically loads the constant data into the job data area without requiring a low file on a GET, R, or RUN command, or a RUN UUO. SAVE will write a low file for a two-segment program only if the LH of .JBCOR is 140_8 or greater.

Table 1-2
Vestigial Job Data Area Locations

Symbol	Octal Location†	Description
.JBHSA	0	A copy of .JBSA.
.JBH41	1	A copy of .JB41.
.JBHCR	2	A copy of .JBCOR.
.JBHRN	3	LH: restores the LH of .JBHRL, RH: restores the RH of .JBREN.
.JBHVR	4	A copy of .JBVER.
.JBHNM	5	High segment name set on a SAVE.
.JBHSM	6	A pointer to the high-segment symbols, if any.
	7	Reserved for future use.
.JBHDA	10	First location not used by vestigial job data area.

†Relative to origin of high segment, usually .JBHGH = 400000_8.

CHAPTER 2
INTRODUCTION TO USER PROGRAMMING

2.1 PROCESSOR MODES

In a single-user, non-timesharing system, the user's program is subject only to those conditions inherent in the hardware. The program must

a. Stay within the memory capacity.
b. Observe the hardware restrictions placed on the use of certain memory locations.
c. Observe the restriction on interrupt instructions.

With timesharing, the hardware limits the central processor operations to one of three modes: user mode, user I/O mode, and executive mode.

2.1.1 User Mode

Normally, user programs run with the processor in user mode and must operate within an assigned area of core. In user mode, certain instructions are illegal. User mode is used to guarantee the integrity of the monitor and each user program. The user mode of the processor is characterized by the following:

a. Automatic memory protection and mapping (refer to Chapter 1).
b. Trap to absolute location 40 in the monitor on any of the following:
   (1) Operation codes 040 through 077 and operation code 00,
   (2) Input/output instructions (DATAI, DATAO, BLKI, BLKO, CONI, CONO, CONSZ, and CONSO),
   (3) HALT (i.e., JRST 4,),
   (4) Any JRST instruction that attempts to enter executive mode or user I/O mode.
c. Trap to relative location 40 in the user area on execution of operation codes 001 through 037.

2.1.2 User I/O Mode

The user I/O mode (bits 5 and 6 of PC word = 11) of the central processor allows privileged user programs to be run with automatic protection and mapping in effect, as well as the normal execution of

all defined operation codes. The user I/O mode provides some protection against partially debugged monitor routines and permits infrequently used device service routines to be run as a user job. Direct control of special devices by the user program is particularly important in real-time applications.

To utilize this mode, the user must have bit 15 (JB.TRP) set in the privilege word. RESET AC, or CALLI 0 terminates user I/O mode. User I/O mode is not used by the monitor and is normally not available to the timesharing user (refer to Paragraph 3.8.3).

2.1.3 Executive Mode

The monitor operates with the processor in executive mode, which is characterized by special memory protection and mapping (refer to Chapter 1) and by the normal execution of all defined operation codes.

User programs run in user mode; therefore, the monitor must schedule user programs, service interrupts, perform all input and output operations, take action when control returns from a user program, and perform any other legal user-requested operations that are not available in user mode. The services the monitor makes available to user-mode programs and how a user program obtains these services, are described in Chapters 3 and 4.

2.2 PROGRAMMED OPERATORS (UUOs)

Operation codes 000 through 077 in the PDP-10 are programmed operators, sometimes referred to as UUOs. They are software-implemented instructions because from a hardware point of view, their function is not pre-specified. Some of these op-codes trap to the monitor, and the rest trap to the user program.

After the effective address calculation is complete, the contents of the instruction register, along with the effective address, are stored, and an instruction is executed out of the normal sequence.

Although there is one operating system for all configurations of the DECsystem-10, some UUOs may not be included in each DECsystem-10. This is especially true of the DECsystem-1040, the basic system intended for small installations who do not want all of the system's features because of a constraint on core. UUOs are deleted from the DECsystem-1040 by feature test switches defined at MONGEN time. In the standard DECsystem-1040, many of these switches are off, and therefore, the corresponding UUOs are not available. This saves core but limits various features of the operating system. In the UUO descriptions that follow, footnotes indicate if the switch is normally absent in the DECsystem-1040. If not stated, the UUO is available on all configurations of the DECsystem-10.

2.2.1 Operation Codes 001-037 (User UUOs)

Operation codes 001 through 037 do not affect the mode of the central processor; thus, when executed in user mode, they trap to user location 40, which allows the user program complete freedom in the use of these programmed operators.

If a user's undebugged program accidentally executes one of these op-codes when the user did not intend to use it, the following error message is normally issued:

HALT AT USER PC addr

This message is given because the user's relative location 41 contains HALT (unless his program has overtly changed it) which is provided by the loader; addr is the location of the user UUO.

2.2.2 Operation Codes 040-077 and 000 (Monitor UUOs)

Operation codes 040 through 077 and 000 trap to absolute location 40, with the central processor in executive mode. These programmed operators are interpreted by the monitor to perform I/O operations and other control functions for the user's program.

Operation code 000 always returns the user to monitor mode with the error message:

?ILLEGAL UUO AT USER PC addr

Table 2-1 lists the operation codes 040 through 077 and their mnemonics.

Table 2-1
Monitor Programmed Operators

Op Code	Call	Function
040	CALL AC, [SIXBIT/NAME/] , or NAME AC,	Programmed operator extension (refer to Paragraph 2.2.2.1).
041	INIT D, MODE SIXBIT/DEV/ XWD OBUF, IBUF error return normal return	Select I/O device (refer to Paragraph 4.2.3).
042		No operation } Reserved for installation-dependent definition.
043		No operation
044		No operation
045		No operation
046		No operation

Table 2-1 (Cont)
Monitor Programmed Operators

Op Code	Call	Function
047	CALLI AC, N	Programmed operator extension (refer to Paragraph 2.2.2.1).
050	OPEN, D, E error return normal return E: EXP STATUS SIXBIT /DEV/ XWD OBUF, IBUF	Select I/O device (refer to Paragraph 4.2.3).
051	TTCALL AC, ADR	Extended operations on job-controlling terminal (refer to Paragraph 5.10.3).
052		Reserved for future expansion by DEC.
053		Reserved for future expansion by DEC.
054		Reserved for future expansion by DEC.
055	RENAME D, E error return normal return E: SIXBIT /FILE/ SIXBIT /EXT/ EXP <PROT> B8+DATE XWD PROJ, PROG	Rename or delete a file (see Section 4.4.3).
056	IN D, normal return error or EOF return	INPUT and skip on error or EOF (see Section 4.5).
057	OUT D, normal return error return	OUTPUT and skip on error or EOT (see Section 4.5).
060	SETSTS D, STATUS	Set file status (see Section 4.6.2).
061	STATO D, BITS R0: NO SELECTED BITS = 1 R1: SOME SELECTED BITS = 1	Skip if file status bits = 1 (see Section 4.6.1).
062	GETSTS D, E	Copy file status to E (see Section 4.6.1).
063	STATZ D, BITS R0: SOME SELECTED BITS = 1 R1: ALL SELECTED BITS = 0	Skip if file status bits = 0 (see Section 4.6.1).
064	INBUF D, N	Set up input buffer ring with N buffers (refer to Paragraph 4.3.2).
065	OUTBUF D, N	Set up output buffer ring with N buffers (refer to Paragraph 4.3.2).
066	INPUT D,	Request input or request next buffer (refer to Paragraph 4.5).

Table 2-1 (Cont)
Monitor Programmed Operators

Op Code	Call	Function
067	OUTPUT D,	Request output or request next buffer (refer to Paragraph 4.5).
070	CLOSE D,	Terminate file operation (refer to Paragraph 4.7).
071	RELEAS D,	Release device (refer to Paragraph 4.8.1).
072	MTAPE D, N	Perform tape positioning operation (refer to Paragraphs 5.5.3 and 6.1.6.5).
073	UGETF D,	Get next free block number on DECtape (refer to Paragraph 6.1.6.3).
074	USETI D, E	Set next input block number (refer to Paragraphs 6.1.6.1 and 6.2.9.2).
075	USETO D, E	Set next output block number (refer to Paragraphs 6.1.6.2 and 6.2.9.2).
076	LOOKUP D, E error return normal return E: SIXBIT /FILE/ SIXBIT /EXT/ 0 XWD PROJ, PROG	Select a file for input (refer to Paragraph 4.4.1).
077	ENTER D, E error return normal return E: SIXBIT /FILE/ SIXBIT /EXT/ 0 XWD PROJ, PROG	Select a file for output (refer to Paragraph 4.4.2).
100	UJEN	Dismiss real-time interrupt (refer to Paragraph 3.8.4).

2.2.2.1 CALL and CALLI - Operation codes 040 through 077 limit the monitor to 40_8 operations. The CALL operation extends this set by specifying the name of the operation by the contents of the location specified by the effective address (e.g., CALL [SIXBIT /EXIT/]). This capability provides for indefinite extendability of the monitor operations, at the overhead cost to the monitor of a table lookup.

The CALLI operation eliminates the table lookup of the CALL operation by having the programmer or the assembler perform the lookup and specify the index to the operation in the effective address of the CALLI. Table 2-2 lists the monitor operations specified by the CALL and CALLI operations.

Table 2-2
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, -2 ... -n		Customer defined	Reserved for definition by each customer installation.
CALLI AC, -1	LIGHTS	CALL AC, [SIXBIT/LIGHTS/]	Display AC in console lights (refer to Paragraph 3.6.4.2).
CALLI AC, 0	RESET	CALL [SIXBIT/RESET/] return	Reset I/O device (refer to Paragraph 4.1.2).
CALLI AC, 1	DDTIN	MOVEI AC, BUFFER CALL AC, [SIXBIT/DDTIN/] only return	DDT mode console input (refer to Paragraph 5.9.2).
CALLI AC, 2	SETDDT	MOVEI AC, DDT-start-adr CALL AC, [SIXBIT/SETDDT/] only return	Set protected DDT starting address (refer to Paragraph 3.1.1.1).
CALLI AC, 3	DDTOUT	MOVEI AC, BUFFER CALL AC, [SIXBIT/DDTOUT/] only return	DDT mode console output (refer to Paragraph 5.9.2).
CALLI AC, 4	DEVCHR	MOVE AC, [SIXBIT/dev/] or MOVEI AC, channel no. CALL AC, [SIXBIT/DEVCHR/] only return C(AC) = 0 if no such device C(AC) = DEVMOD word of device data block if device is found.	Get device characteristics (refer to Paragraph 4.10.2).
CALLI AC, 5	DDTGT	CALL AC, [SIXBIT/DDTGT/] only return	No operation, historical UUO.
CALLI AC, 6	GETCHR	AC: = SIXBIT/DEV/ CALL AC, [SIXBIT/GETCHR/] only return	Same as CALLI AC, 4.
CALLI AC, 7	DDTRL	CALL AC, [SIXBIT/DDTRL/] only return	No operation; historical UUO.
CALL AC, 10	WAIT	AC field is software channel number. CALL AC, [SIXBIT/WAIT/] only return	Wait until device is inactive (refer to Paragraph 4.5.3).
CALLI AC, 11	CORE	MOVE AC, [XWD HIGH ADR or 0, LOW ADR or 0] CALL AC, [SIXBIT/CORE/] error return, assignment unchanged normal return, new assignment AC: = max. core available (in 1K blocks) on error or normal return.	Allocate core (refer to Paragraph 3.2.3).

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 12	EXIT	CALL AC, [SIXBIT/EXIT/] return If AC ≠ 0, devices are not released and CONT and CCONT commands are effective.	Stop job, may release devices depending on contents of AC (refer to Paragraph 3.1.2.3).
CALLI AC, 13	UTPCLR	AC field is software channel number CALL AC, [SIXBIT/UTPCLR/] only return	Clear DECtape directory (refer to Paragraph 6.1.6.4).
CALLI AC, 14	DATE	CALL AC, [SIXBIT/DATE/] only return AC: = date in compressed format	Return date (refer to Paragraph 3.6.1.1).
CALLI AC, 15	LOGIN††	MOVE AC, [XWD -N, LOC] CALL AC, [SIXBIT/LOGIN/] R0: return Does not return if C(R0) is a HALT instruction.	Privileged UUO in that the calling job must not be logged in. Is a no-op if executed by a job already logged-in.
CALLI AC, 16	APRENB	MOVEI AC, BITS CALL AC, [SIXBIT/APRENB/] return	Enable central processor traps (refer to Paragraph 3.1.3.1).
CALLI AC, 17	LOGOUT††	CALL AC, [SIXBIT/LOGOUT/] no return	Privileged UUO available only to system-privileged programs. Is treated like an EXIT UUO if executed by a non-system-privileged program.
CALLI AC, 20	SWITCH	CALL AC, [SIXBIT/SWITCH/] return AC: contents of console data switches	Read console data switches (refer to Paragraph 3.6.4.1).
CALLI AC, 21	REASSI	MOVE AC, job number MOVE AC+1, [SIXBIT/DEV/] CALL AC, [SIXBIT/REASSI/] return If C(AC) = 0 on return, the job specified has not been initialized. If C(AC+1) = 0 on return, the device is not assigned to calling job, or device is TTY.	Reassign device (refer to Paragraph 4.8.3).
CALLI AC, 22	TIMER	CALL AC, [SIXBIT/TIMER/] return AC: = time in jiffies, right justified.	Read time of day in clock ticks (refer to Paragraph 3.6.1.2).
CALLI AC, 23	MSTIME	CALL AC, [SIXBIT/MSTIME/] return AC: = time in milliseconds, right-justified.	Read time of day in milliseconds (refer to Paragraph 3.6.1.3).

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 24	GETPPN	CALL AC, [SIXBIT/GETPPN/] normal return alternate return AC: = XWD proj. no., prog. no. of this job. Alternate return is taken only if job is privileged and the same proj-prog number occurs twice in the table of jobs logged in.	Return project-programmer number of job (refer to Paragraph 3.6.2.3).
CALLI AC, 25	TRPSET	MOVE AC, [XWD N, LOC] CALL AC, [SIXBIT/TRPSET/] error return normal return LOC: JSR TRAP	Set trap for user I/O mode (refer to Paragraph 3.8.3).
CALLI AC, 26	TRPJEN	CALL [SIXBIT/TRPJEN/]	Illegal UUO; replaced by UJEN (op code 100).
CALLI AC, 27	RUNTIM	MOVE AC, job number or 0 CALL AC, [SIXBIT/RUNTIM/] only return AC: = running time of job AC: = 0 if non-existent job	Return the jobs running time in milliseconds (refer to Paragraph 3.6.2.1).
CALLI AC, 30	PJOB	CALL AC, [SIXBIT/PJOB/] return AC: = job number, right-justified	Return job number (refer to Paragraph 3.6.2.2).
CALLI AC, 31	SLEEP	MOVE AC, time to sleep in seconds CALL AC, [SIXBIT/SLEEP/] return	Stop job for specified time in seconds (refer to Paragraph 3.1.4.1).
CALLI AC, 32	SETPOV	CALL AC, [SIXBIT/SETPOV/] return	Superseded by APRENB UUO.
CALLI AC, 33	PEEK	MOVEI AC, exec adr CALL AC, [SIXBIT/PEEK/] return AC: = C(exec-adr)	Return contents of executive address (refer to Paragraph 3.6.3.1).
CALLI AC, 34	GETLIN	CALL AC, [SIXBIT/GETLIN/] return AC: = SIXBIT TTY name, left-justified (e.g., CTY, TTY27)	Return SIXBIT name of attached terminal (refer to Paragraph 5.9.4).
CALLI AC, 35	RUN	MOVSI AC, start adr increment HRRI AC, E RUN AC, error return normal return	Transfer control to selected program (refer to Paragraph 3.3.1).

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 35 (continued)	RUN	E: SIXBIT/DEVICE/ SIXBIT/FILE/ SIXBIT/EXT/ 0 XWD proj no. prog no XWD 0; optional core assignment	
CALLI AC, 36	SETUWP	MOVEI AC, BIT SETUWP AC, error return normal return	Set or clear user mode write protect for high segment (refer to Paragraph 3.2.4).
CALLI AC, 37	REMAP	MOVEI AC, highest adr. in low seg REMAP AC, error return normal return	Remap top of low segment into high segment (refer to Paragraph 3.3.3).
CALLI AC, 40	GETSEG	MOVEI AC, E GETSEG AC, error return normal return E: SIXBIT/DEVICE/ SIXBIT/FILE/ SIXBIT/EXT/ 0 XWD proj no, prog no 0	Replace high segment in user's addressing space (refer to Paragraph 3.3.2).
CALLI AC, 41	GETTAB	MOVSI AC, job no. or index no. HRRI AC, table no. GETTAB AC, error return normal return C(AC) unchanged on error return AC: = table entry if table is defined and index is in range.	Return contents of monitor table or location (refer to Paragraph 3.6.3.4).
CALLI AC, 42	SPY	MOVEI AC, highest physical adr. desired SPY AC, error return normal return	Make physical core be high segment for examination of monitor (refer to Paragraph 3.6.3.2).
CALLI AC, 43	SETNAM	MOVE AC, [SIXBIT/NAME/] SETNAM AC, return	Set program name in monitor job table (refer to Paragraph 3.4.1).
CALLI AC, 44	TMPCOR	MOVE AC, [XWD CODE, BLOCK] TMPCOR, error return normal return	Allow temporary in-core file storage for job (refer to Paragraph 3.5.1).

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 44 (continued)	TMPCOR	BLOCK: XWD NAME, 0 IOWD BUFFLEN,BUFFER AC: = value depending on CODE and whether error or normal return is taken.	
CALLI AC, 45	DSKCHR	MOVE AC, [XWD+N, LOC] DSKCHR AC, error return normal return AC: = XWD status, configuration LOC: = SIXBIT/NAME/ 0 0 0 } values returned	Return disk characteristics (refer to Paragraph 6.2.9.13).
CALLI AC, 46	SYSSTR	MOVEI AC, 0 or MOVE AC, [SIXBIT/FSNAME/] SYSSTR AC, error return - not a file structure normal return AC: = next file structure name in SIXBIT, left-justified	Return next file structure name (refer to Paragraph 6.2.9.10).
CALLI AC, 47	JOBSTR	MOVE AC, [XWD N, LOC] JOBSTR AC, error return normal return AC: = argument Contents — Use LOC/SIXBIT/NAME/or - 1 arg LOC+1/XWD proj no. prog. no. — value LOC+2/status bits — value	Return next file structure name in the jobs search list (refer to Paragraph 6.2.9.8).
CALLI AC, 50	STRUUO	MOVE AC, [XWD N, LOC] STRUUO AC, error return normal return AC: = status or error code Contents — Use LOC:function numbers — arg LOC+1: — arg depending on function number	Manipulate file structures (refer to Paragraph 6.2.9.7).
CALLI AC, 51	SYSPHY	MOVEI AC, 0 or last unit name SYSPHY AC, error return normal return	Return all physical disk units (refer to Paragraph 6.2.9.11).

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 52	FRECHN		Reserved for future use.
CALLI AC, 53	DEVTYP	MOVE AC, [SIXBIT/dev/] or MOVEI AC, channel no. DEVTYP AC, error return normal return	Return properties of device (refer to Paragraph 4.10.3).
CALLI AC, 54	DEVSTS	MOVEI AC, channel no. of device DEVSTS AC, error return normal return	Return hardware device status word (refer to Paragraph 4.10.1).
CALLI AC, 55	DEVPPN	MOVE AC, [SIXBIT/DEV/] DEVPPN AC, error return normal return AC: = XWD proj-prog. number on a normal return	Return the project programmer number associated with a device (refer to Paragraph 6.2.9.12).
CALLI AC, 56	SEEK†††	AC is software channel number SEEK AC, return	Perform a SEEK to current selected block for software channel AC (refer to Paragraph 6.2.9.3).
CALLI AC, 57	RTTRP	MOVEI AC, RTBLK RTTRP AC, error return normal return	Connect real-time devices to PI system (refer to Paragraph 3.8.1).
CALLI AC, 60	LOCK	MOVE AC, [XWD high seg code, low seg code] LOCK AC, error return normal return	Lock job in core (refer to Paragraph 3.2.2).
CALLI AC, 61	JOBSTS	MOVEI AC, channel no. or MOVNI AC, job JOBSTS AC, error return normal return	Return status information about device TTY and/or controlled job (refer to Paragraph 5.9.4.4).
CALLI AC, 62	LOCATE	MOVEI AC, location LOCATE AC, error return normal return	Change the job's logical station (refer to Paragraph 3.4.3).
CALLI AC, 63	WHERE	MOVEI AC, channel no. or MOVE AC, [SIXBIT/dev/] WHERE AC, error return normal return	Return the physical station of the device (refer to Paragraph 4.10.5).

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 64	DEVNAM	MOVEI AC, channel no. or MOVE AC, [SIXBIT/dev/] DEVNAM AC, error return normal return	Return physical name of device obtained through generic INIT/OPEN or logical device assignment (refer to Paragraph 4.10.6).
CALLI AC, 65	CTLJOB	MOVE AC, job number CTLJOB AC, error return normal return	Return job number of controlling job (refer to Paragraph 5.9.4.5).
CALLI AC, 66	GOBSTR	MOVE AC, [XWD N,LOC] GOBSTR AC, error return normal return LOC: job number LOC+1: XWD proj no, prog no LOC+2: SIXBIT/NAME/or-1 LOC+3: 0 LOC+4: Status bits	Return next file structure name in an arbitrary job's search list (refer to Paragraph 6.2.9.9).
CALLI AC, 67	ACTIVATE		Reserved for the future.
CALLI AC, 70	DEACTIVATE		
CALLI AC, 71	HPQ	MOVE AC, high-priority queue no. HPQ AC, error return normal return	Place job in high priority scheduler's run queue (refer to Paragraph 3.8.5).
CALLI AC, 72	HIBER	MOVSI AC, enable bits HRRI AC, sleep time HIBER AC, error return normal return	Allow job to become dormant until the specified event occurs (refer to Paragraph 3.1.4.2).
CALLI AC, 73	WAKE	MOVE AC, job no. WAKE AC, error return normal return	Allow job to activate the specified dormant job (refer to Paragraph 3.1.4.3).
CALLI AC, 74	CHGPPN††	MOVE AC, new proj. prog. no. CHGPPN AC, error return normal return	Change project-programmer number. Gives an error return if executed by a job already logged-in.
CALLI AC, 75	SETUUO	MOVE AC, [XWD function, argument] SETUUO AC, error return normal return	Set system and job parameters (refer to Paragraph 3.4.2).
CALLI AC, 76	DEVGEN		Reserved for the future.

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 77	OTHUSR	OTHUSR AC, non-skip return skip return AC: = proj. prog. no.	Determine if another job is logged in with same project-programmer number (refer to Paragraph 3.6.2.4).
CALLI AC, 100	CHKACC	MOVE AC, [EXP LOC] CHKACC AC, error return normal return LOC: XWD action, protection LOC+1: directory proj-prog no. LOC+2: user proj-prog no.	Check user's access to the file specified (refer to Paragraph 6.2.9.6).
CALLI AC, 101	DEVSIZ	MOVE AC, [EXP LOC] DEVSIZ AC, error return normal return LOC: EXP STATUS LOC+1: SIXBIT/dev/	Determine buffer size for the specified device (refer to Paragraph 4.10.4).
CALLI AC, 102	DAEMON	MOVE AC, [XWD + length, adr of arg. list] DAEMON AC, error return normal return	Request DAEMON to perform a specified task (refer to Paragraph 3.7).
CALLI AC, 103	JOBPEK††	MOVE AC, adr of arg block JOBPEK AC, error return normal return	Read or write another job's core. Gives the error return if executed by a non-system-privileged program.
CALLI AC, 104	ATTACH††	MOVE AC, [XWD line no., job no.] ATTACH AC, error return normal return	Attach the job to the specified TTY line number. Gives the error return if executed by a non-system-privileged program.
CALLI AC, 105	DAEFIN††	MOVE AC, [XWD + length, adr of arg. list] DAEFIN AC, error return normal return	Indicate that the request to the DAEMON program has been completed. Gives the error return if executed by a non-system-privileged program.
CALLI AC, 106	FRCUUO††	MOVE AC, [XWD + length, adr of arg. list] FRCUUO AC, error return normal return	Force a command for a job. Gives the error return if executed by a non-system-privileged program.

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 107	DEVLNM	MOVE AC, [SIXBIT/dev/] or MOVEI AC, channel no. MOVE AC+1, [SIXBIT/logical name/] DEVLNM AC, error return normal return	Set a logical name for this specified device (refer to Paragraph 4.8.4).
CALLI AC, 110	PATH.	MOVE AC, [XWD + length, adr. of argument list] PATH. AC, error return normal return ADR: N or SIXBIT/NAME/ ADR+1: Scan switch ADR+2: PPN ADR+3: SFD name ADR+4: SFD name ⋮	Read or modify the default directory path or read the current path of a file OPEN on a channel. Refer to Paragraph 6.2.9.1.
CALLI AC, 111	METER.	MOVE AC, [XWD N, LOC] METER. AC, error return normal return LOC: function code LOC+1: argument depends LOC+2: on function code used. ⋮ LOC+N-1:	Provide performance analysis and metering of dynamic system variables. Refer to Paragraph 3.9.
CALLI AC, 112	MTCHR.	MOVEI AC, channel no. or MOVE AC, [SIXBIT/dev/] MTCHR. AC, error return normal return	Return characteristics of the magnetic tape. Refer to Paragraph 5.5.3.2.
CALLI AC, 113	JBSET.††	MOVE AC, [2,,BLOCK] JBSET. AC, error return normal return BLOCK: 0,, job number BLOCK+1: function,, value	Execute the specified function of SETUUO for a particular job.
CALLI AC, 114	POKE.	MOVE AC, [3,,BLOCK] POKE. AC, error return normal return BLOCK: location BLOCK+1: old value BLOCK+2: new value	Alter the specified location in the Monitor. Refer to Paragraph 3.6.3.3.

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI† Mnemonic	CALL	Function
CALLI AC, 115	TRMNO.	MOVE AC, job number TRMNO. AC, error return normal return	Return number of the terminal currently controlling the specified job. Refer to Paragraph 5.10.5.
CALLI AC, 116	TRMOP.	MOVE AC, [XWD N, ADR] TRMOP. AC, error return normal return ADR: function code ADR+1: terminals's universal index . Following arguments depend : on function used.	Perform miscellaneous terminal functions. Refer to Paragraph 5.10.6.
CALLI AC, 117	RESDV.	MOVE AC, channel no. RESDV. AC, error return normal return	Reset the specified channel. Refer to Paragraph 4.8.2.

CALLI AC, 120	UNLOK.	MOVSI AC,1
		MOVSI AC,0
		HRRI AC,1
		HRRI AC,0
		UNLOK. AC,
		error return
		normal return
			Unlock a locked job in core. Refer to Paragraph 3.2.2.4.

CALLI AC, 121	DISK.	MOVE AC, [XWD function, ADR] DISK. AC, error return normal return	Set or read a disk or file system parameter (e.g., set the disk priority for a channel or the job). Refer to Paragraph 6.2.9.14.
CALLI AC, 122	DVRST.††	MOVE AC, [SIXBIT/dev/] or MOVEI AC, channel no. DVRST. AC, error return normal return	Restrict the specified device to a privileged job.
CALLI AC, 123	DVURS.††	MOVE AC, [SIXBIT/dev/] or MOVEI AC, channel no. DVURS. AC, error return normal return	Remove the restricted status of the specified device.

†The CALLI mnemonics are defined in a separate MACRO assembler table, which is scanned whenever an undefined OP CODE is found. If the symbol is found in the CALLI table, it is defined as though it had appeared in an appropriate OPDEF statement, that is

RETURN : EXIT

If EXIT is undefined, it will be assembled as though the program contained the statement

OPDEF EXIT [CALLI 12]

This facility is available in MACRO V.43 and later.

Table 2-2 (Cont)
CALL and CALLI Monitor Operations

††This CALLI is a system-privileged UUO available only to users logged in under [1,2] or to programs running with the JACCT bit set. Complete documentation for system-privileged UUOs appears in the Specifications section of the DECsystem-10 Software Notebooks.

†††All CALLI's above CALLI 55 do not have a corresponding CALL with a SIXBIT argument. This is to save monitor table space.

The customer is allowed to add his own CALL and CALLI calls to the monitor. A negative CALLI effective address (-2 or less) should be used to specify such customer-added operations.

2.2.2.2 Suppression of Logical Device Names - Some system programs, e.g., LOGOUT, require I/O to specific physical devices regardless of the logical name assignments. Therefore, for any CALLI, if bit 19 (UU.PHS) in the effective address of the CALLI is not equal to bit 18, only physical names will be used; logical device assignments will be ignored. This suppression of logical device names will be ignored. This suppression of logical device names is helpful, for example, when using the results of the DEVNAM UUO where the physical name corresponding to a logical name is returned.

2.2.2.3 Restriction on Monitor UUOs in Reentrant User Programs - A number of restrictions on UUOs that involve a high segment prevent naive or malicious users from interferring with other users while sharing segments and minimize monitor overhead in handling two-segment programs. The basic rules are as follows:

a. All UUOs can be executed from the low or high segment although some of their arguments cannot be in or refer to the high segment.

b. No buffers, buffer headers, or dump-mode command lists may exist in the high segment for reading from or writing to any I/O device.

c. No I/O is processed into or out of the high segment except via the SAVE and SSAVE commands.

d. No STATUS, CALL or CALLI UUO allows a store in the high segment.

e. The effective address of the LOOKUP, ENTER, INPUT, OUTPUT, and RENAME UUOs cannot be in the high segment. If any rule is violated, an address check error message is given.

f. As a convenience in writing user programs, the monitor makes a special check so that the INIT UUO can be executed from the high segment, although the calling sequence is in the high segment. The monitor also allows the effective address of the CALL UUO, which contains the SIXBIT monitor function name, and the effective address of the OPEN UUO, which contains the status bits, device name, and buffer header addresses, in the high segment. The address of TTCALL 1, and TTCALL 3, may be in the high segment for convenience in typing messages.

2.2.3 Operation Codes 100-127 (Unimplemented Op Codes)

Op code 100(UJEN)	Dismiss real-time interrupt from user mode (refer to Paragraph 3.8.4).
Op codes 101-107 114-117 123	Monitor prints ?ILL INST. AT USER n and stops the job.
Op codes 110-113 120-122 124-127	These op codes are valid on the KI10. If used on the KA10, the monitor prints ?KI10 ONLY INST. AT USER n and stops the job.

2.2.4 Illegal Operation Codes

The eight I/O instructions (e.g., DATAI) and JRST instructions with bit 9 or 10 = 1 (e.g., HALT, JEN) are interpreted by the monitor as illegal instructions (refer to the System Reference Manual in the Software Notebooks). The job is stopped and a question mark is printed immediately. A carriage return-line feed is then output, followed by an error message. For example, a DATAI instruction would produce the following:

```
?
? ILL INST AT USER addr
```

2.2.5 Naming Conventions for Monitor Symbols

The names of the monitor's data base symbols contain dots or percent signs so that they can be made user-mode symbols without conflicting with previously-coded user programs. Data symbols can be divided into five classes:

1) numbers
2) masks
3) UUO names
4) GETTAB arguments
5) error codes.

Symbols defining numbers begin with a dot, followed by a two-letter prefix indicating the type of number, and end with a three-character abbreviation representing the specific number. Numbers are 18-bit quantities and include core addresses and function codes. The following are examples of names of various numbers:

.JBxxx	Job Data Area
.GTxxx	GETTAB table numbers
.RBxxx	Extended arguments for LOOKUP, ENTER, RENAME

Names for masks start with a two-letter prefix indicating the individual word, followed by a dot, and end with three characters representing the specific mask. Masks are 36-bit quantities and include bits and fields. The following are examples of names of masks:

JP.xxx	Privilege word bits
JW.xxx	WATCH word bits
PC.xxx	PC word bits

Names for UUOs implemented after the 5.03 release of the monitor are five or less characters followed by a dot. For example,

PATH.	UUO to modify directory path
TRMOP.	UUO to perform terminal functions.

Individual words within a GETTAB table start with a percent sign, followed by two characters representing the generic name of the table, and end with three characters identifying the specific word. For example,

%NSCMX	CORMAX word in the nonswapping data table.
%CNSTS	States word in the configuration table.

Names of bytes and bits within a GETTAB word begin with two characters representing the word, followed by a percent sign, and end with three characters designating the specific byte.

ST % DSK	Byte representing disk system; contained in the states word.
ST % SWP	Byte indicating swapping system; contained in the states word.

Error codes returned on a UUO error have names with the following pattern: two characters indicating the UUO, three characters designating the failure type, and a terminating percent sign.

DMILF%	DAEMON error; illegal function.
RTDIU%	RTTRP error; device in use.
LKNLP%	LOCK error; no locking privileges.

Many of the values useful in user programming are encoded in the parameter file C.MAC for the convenience of writing and modifying programs.

CHAPTER 3
NON-I/O UUOS

3.1 EXECUTION CONTROL

3.1.1 Starting

A user program may start another program only by using the RUN or GETSEG UUOs (refer to Paragraphs 3.3.1 and 3.3.2). A user at a terminal may start a program with the monitor commands RUN, START, CSTART, CONT, CCONT, DDT, and REENTER (refer to DECsystem-10 Operating System Commands). The starting address either appears as an argument of the command or is stored in the user's job data area (refer to Chapter 1).

3.1.1.1 SETDDT AC, or CALLI AC, 2 - This UUO causes the contents of the AC to replace the DDT starting address, which is stored in the protected job data area location .JBDDT. The starting address is used by the monitor command, DDT.

3.1.2 Stopping

Any of the following procedures can stop a running program:

a. One ↑C from the user's terminal if the user program is in a TTY input wait; otherwise, two ↑Cs from the user's terminal (refer to DECsystem-10 Operating System Commands);

b. A monitor detected error;

c. Program execution of HALT, CALL [SIXBIT/EXIT/], or CALL [SIXBIT/LOGOUT/].

3.1.2.1 Illegal Instructions (700-777, JRST 10, JRST 14) and Unimplemented OP Codes (101-127)- Illegal instructions trap to the monitor, stop the job, and print:

?ILL INST. AT USER adr or ?KI ONLY INST. AT USER adr

Refer to Paragraph 2.2.3 for an explanation of op codes 101-127. Note that the program cannot be continued by typing the CONT or CCONT commands.

3.1.2.2 HALT or JRST 4 - The HALT instruction is an exception to the illegal instructions; it traps to the monitor, stops the job, and prints:

?HALT AT USER adr

where n is the location of the HALT instruction. If the HALT instruction is in location 41 and the program executed a user UUO (operation codes 001-037), the address in the error message is that of the user UUO instead of address 41.

However, the CONT and CCONT commands are still valid, and, if typed, will continue the program at the effective address of the HALT instruction. After a user program HALT instruction followed by a START, DDT, CSTART, or REENTER command, . JBOPC contains the address of the HALT. To proceed at the address specified by the effective address, it is necessary for the user or his program to recompute the effective address of the HALT instruction and to use this address to start (refer to .JBOPC description, Table 1-1 in Paragraph 1.2.1). HALT is not the instruction used to terminate a program (refer to Paragraph 3.1.2.3). HALT is useful for indicating impossible error conditions.

3.1.2.3 EXIT AC, or CALLI AC, 12 - When the value of AC is zero, all I/O devices (including real-time devices) are RELEASed (refer to Paragraph 4.8.1); the job is unlocked from core; the user mode write protect bit (UWP) for the high segment is set; the APR traps are reset to 0; the PC flags are cleared; and the job is stopped. If timesharing was stopped (refer to Paragraph 3.8.3), it is resumed. In other words, after releasing all I/O devices that close out all files, a RESET is done (refer to Paragraph 4.1.2). The carriage-return/line-feed is performed, and

EXIT

is printed on the user's terminal, which is left in monitor mode. The CONT and CCONT commands cannot continue the program.

When the value of AC is nonzero, the job is stopped, but devices are not RELEASed and a RESET is not done. Instead of printing EXIT, only a carriage-return and line-feed is performed, and a period is printed on the user's terminal. The CONT and CCONT commands may be used to continue the program. In other words, this form of EXIT does not affect the state of the job except to stop it and return the terminal to monitor mode. Programs using EXIT 1, (MONRT.) as a substitute for EXIT (to eliminate the typing of EXIT) should RELEASE all devices first.

3.1.3 Trapping

3.1.3.1 APRENB AC, or CALLI AC, 16 - APR trapping allows a user to handle any and all traps that occur while his job is running on the central processor, including illegal memory references, non-existent memory references, pushdown list overflow, arithmetic overflow, floating-point overflow, and clock flag. To enable for trapping, a APRENB AC, or CALLI AC, 16 is executed, where the AC contains the central processor flags to be tested on interrupts, as defined below:

Name		AC Bit	Trap On
AP.REN	18	400000	Repetitive enable
AP.POV	19	200000	Pushdown overflow
AP.ILM	22	20000	Memory protection violation
AP.NXM	23	10000	Nonexistent memory flag
AP.PAR	24	4000	Parity error
AP.CLK	26	1000	Clock flag
AP.FOV	29	100	Floating-point overflow
AP.AOV	32	10	Arithmetic overflow

When one of the specified conditions occurs while the central processor is in user mode, the state of the central processor is CONditioned Into (CONI) location, .JBCNI, and the PC is stored in location .JBTPC in the job data area (refer to Table 1-1 in Paragraph 1.2.1). Then control is transferred to the user trap-answering routine specified by the contents of the right half of .JBAPR, after the arithmetic and floating-point overflow flags are cleared. (However, the job is stopped if the PC is equal to the first or second instruction in the user's trap routine.) The user program must set up location .JBAPR before executing the APRENB UUO. To return control to his interrupted program, the user's trap-answering routine must execute a JRSTF @ .JBTPC which clears the bits that have been processed and restores the state of the processor.

The APRENB UUO normally enables traps for only one occurrence of any selected condition and must be re-issued after each condition of a trap. To disable this feature, set bit 18 to a 1 when executing the UUO. However, even with bit 18 = 1, clock interrupts must be re-enabled after each trap.

If the user program does not enable traps, the monitor sets the PDP-10 processor to ignore arithmetic and floating-point overflow, but enables interrupts for the other error conditions in the list above. If the user program produces such an error condition, the monitor stops the user job and prints one of the following appropriate messages:

```
?PC OUT OF BOUNDS AT USER PC addr
?ILL MEM REF AT USER PC addr
?NON-EX MEM AT USER PC addr
?PDL OV AT USER PC addr
?MEM PAR ERROR AT USER PC addr
```

The CONT and CCONT commands will not succeed after such an error.

3.1.3.2 Error Intercepting - When certain conditions occur in the program, the monitor intercepts the condition and examines location .JBINT in the job data area. Depending on the contents of this location, control is either retained by the user program or is given to the monitor for action. If this location is zero, the job is stopped and the user and possibly the operator are notified by appropriate messages, if any. If location .JBINT is non-zero, the contents is interpreted as the address of a block with the following format:

```
LOC: XWD N, INTLOC
LOC+1: XWD BITS, CLASS
LOC+2: 0
LOC+3: 0
```

where N is the number of words in the block (N >3).

INTLOC is the location at which the program is to be restarted.

BITS is a set of bits interpreted as follows:

If bit 0 = 1, an error message, if any, is not to be typed on the user's terminal or, in some cases, the operator's terminal.

If bit 0 = 0, an error message, if any, will be typed on the user's terminal and possibly the operator's terminal.

CLASS is a set of bits interpreted as follows:

For each type of error, CLASS has a specific bit. For a given error, the job will be interrupted if the appropriate bit is 1 and the content of LOC+2 is zero. The job will be stopped if either the appropriate bit is 0 or the appropriate bit is 1 and the content of LOC+2 is not zero. By requiring LOC+2 to be zero, the possibility of a loop occurring is prevented.

The monitor examines the CLASS bits and the contents of LOC+2 to determine if the job is to be stopped or interrupted on the particular error. If the job is interrupted, the following information is then stored in LOC+2 and LOC+3:

LOC+2	The last user PC word.
LOC+3	RH = the channel number.
	LH = the error bit as defined in CLASS (see below).

The job is then restarted at location INTLOC.

The CLASS bits are defined as follows:

Device Errors

Bit 35[1] (ER.IDV) represents device errors that can be corrected by human intervention. The appropriate message returned to the user is

DEVICE xxx OPR zz ACTION REQUESTED

where xxx is the device name, and zz is the number of the station at which the operator is located. The operator receives the message

%PROBLEM ON DEVICE xxx FOR JOB n

[1] This bit depends on FTOPRERR which is normally off in the DECsystem-1040.

where xxx is the device name, and n is the number of the job that is stopped. When the operator has corrected the error, he starts the job with the JCONT command and the message

CONT BY OPER

appears on the user's terminal to signify that the error has been corrected.

↑C Intercept

Bit 34[1] (ER.ICC) indicates a ↑C intercept. This intercept allows the user's program to process a ↑C itself instead of allowing the job to automatically return to monitor level. If this bit is 1, the job does not return to monitor level on two ↑Cs (or on one ↑C if the job is in TTY input wait), but instead traps to the user's interrupt routine. There are no messages associated with this bit. When enabled for ↑C, the program should normally exit immediately by releasing any special resources and issuing an EXIT UUO (MONRT. or CALLI 1, 12).. If the user types .CONT, the job continues.

```
        TITLE   CONCIN -- SAMPLE FOR CONTROL-C INTERCEPT

;THIS ROUTINE SHOWS HOW TO ENABLE FOR A CONTROL-C INTERCEPT
;AND HANDLE IT CORRECTLY.  THE IDEA IS TO GET THE USER TO
;MONITOR LEVEL AS QUICKLY AS POSSIBLE.

        LOC     134             ;SET POINTER IN .JBINT
        EXP     INTBLK          ;  TO THE INTERRUPT BLOCK
        RELOC

INTBLK: XWD     4,INTLOC        ;4 WORDS LONG,,PLACE TO START
        XWD     Ø,2             ;NO MESSAGE CONTROL,,TYPE 2 (↑C)
        Z                       ;GETS LAST USER PC
        Z                       ;LH GETS INTERRUPT TYPE

;THE INTERRUPT ROUTINE STARTS HERE

INTLOC: MOVEM   1,TEMP1         ;SAVE AC 1
        HLRZ    1,INTBLK+3      ;GET REASON FOR INTERRUPT
        CAIE    1,2             ;SEE IF CONTROL-C
        HALT    .               ;ERROR IF NOT
                        ;RELEASE ANY SPECIAL RESOURCES HERE
                        ;  BUT BE CAREFUL THAT THIS DOES NOT
                        ;  TAKE VERY LONG OR CAUSE A LOOP.
        EXIT    1,              ;RETURN TO MONITOR
        MOVE    1,INTBLK+2      ;GET RETURN PC
        EXCH    1,TEMP1         ;RESTORE AC
        PUSH    P,INTBLK+2      ;SAVE RETURN ADDRESS
        SETZM   INTBLK+2        ;CLEAR INTERUPT TO ALLOW ANOTHER ONE
        POPJ    P,              ;RETURN TO PROGRAM WHERE STOPPED

TEMP1:  Z                       ;TEMPORARY
```

[1] This bit depends on FTCCIN which is normally off in the DECsystem-1040

The following example illustrates user ↑C processing by a program which will not let users reach monitor level by means of a ↑C.

```
        LOC     134             ;SET UP .JBINT TO POINT TO
        EXP     INTBLK          ;  THE INTERRUPT BLOCK
        RELOC

INTBLK: XWD     3,INTLOC        ;3 WORDS LONG,,PLACE TO START
        XWD     0,2             ;NO MESSAGE CONTROL,,TYPE 2 (↑C)
        Z                       ;GETS LAST USER PC
        Z                       ;LH GETS INTERRUPT TYPE

; THE INTERRUPT ROUTINE

INTLOC: SKIPL   RENFLA          ;OK TO FAKE A REENTER?
        JRST    .+3             ;NO, CURRENT ROUTINE CANNOT BE
                                ;INTERRUPTED

        SETZM   INTBLK+2        ;YES, RE-ENABLE INTERRUPT AND GO
        JRST    REENRT          ;  TO INTERRUPT ROUTINE

        SETOM   RENSWH          ;SET FLAG TO SAY "REENTER AS SOON AS
                                ;  YOU CAN"
        PUSH    P,INTBLK+2      ;GET LAST PC. PUSH/POP
        SETZM   INTBLK+2        ;RE-ENABLE INTERRUPT
        POPJ    P,              ;GO BACK TO INTERRUPTED ROUTINE
                                ;   NOTE THAT IF A CONTROL-C IS
                                ;   TYPED AFTER THE SETZM, THE
                                ;   INTERRUPTS NEST.
```

Off-line Disk Unit

Bit 33 (ER.OFL) indicates a disk unit has dropped off-line. The operator is given the message

UNIT xxx WENT OFF-LINE (FILE UNSAFE)
PLEASE POWER DOWN AND THEN TURN IT ON AGAIN

immediately and then once every minute. The user receives the message

DSK IS OFF-LINE. WAITING FOR OPERATOR
ACTION. TYPE ↑C TO GET A HUNG MESSAGE
(IN 15 SECONDS). DONT TYPE ANYTHING TO WAIT
FOR THE OPERATOR TO FIX THE DEVICE.

If the user has a system resource, he receives the additional message:

THE SYSTEM WILL DO NO USEFUL WORK UNTIL
THE DRIVE IS FIXED OR YOU TYPE ↑C

Full File Structure

Bit 32 (ER.FUL) indicates that a file structure has filled up with data (i.e., there are no free blocks). There are no messages associated with this bit.

Exhausted Disk Quota

Bit 31 (ER.QEX) indicates that the user's disk quota has been exhausted. The user receives the message

[EXCEEDING QUOTA file structure name]

Exceeded Time Limit

Bit 30[1] (ER.TLX) indicates that the user's run time limit (as set by a previous SET TIME command) has been exceeded. This bit is used only by non-batch jobs. The user receives the message

?TIME LIMIT EXCEEDED

3.1.4 Suspending

3.1.4.1 SLEEP AC, or CALLI AC, 31 - This UUO temporarily stops the job and continues it automatically after the elapsed real-time (in seconds) indicated by the contents of the AC. There is an implied maximum of approximately 68 sec (82 sec in 50-Hz countries) or 1 min. A program that requires a longer SLEEP or HIBER time should use the HIBER UUO with no clock request and then call DAEMON, via the .CLOCK function (refer to Paragraph 3.7.2), to wake it.

3.1.4.2 HIBER AC, or CALLI AC, 72[2] - The HIBERNATE UUO allows a job to become dormant until a specified event occurs. The possible events that can wake a hibernating job are: 1) input activity from the user's TTY or any TTY INITed by this job (both line mode and character mode), 2) PTY activity for any PTY currently INITed by this job, 3) the time-out of a specified amount of sleep time, or 4) the issuance of a WAKE UUO directed at this job either by some other job with wake-up rights or by this job at interrupt level.

The HIBERNATE UUO must contain in the left half of AC the wake-condition enable bits and in the right half the number of ms for which the job is to sleep before it is awakened.

[1] This bit depends on FTTLIM which is normally off in the DECsystem-1040.

[2] This UUO depends on FTHIBWAK which is normally off in the DECsystem-1040.

The call is as follows:

```
MOVSI AC, enable bits      ;get HIBERNATE conditions
HRRI AC, sleep time        ;number of ms to sleep
HIBER AC,                  ;or CALLI AC, 72
error return
normal return
```

The HIBERNATE UUO enable condition codes are as follows:

Bits	Meaning
18-35	Number of ms sleep time. It is rounded up to an even multiple of jiffies (maximum being 2^{12} jiffies). Zero means no clock request (i.e., infinite sleep).
15-17	WAKE UUO protection code: Bit 17 (HB.RWT) = 1, project codes must match. Bit 16 (HB.RWP) = 1, programmer codes must match. Bit 15 (HB.RWJ) = 1, only this job can wake itself.
13-14	Wake on TTY input activity: Bit 14 (HB.RTC) = 1, wake on character ready. Bit 13 (HB.RTL) = 1, wake on line of input ready.
12	(HB.RPT) Wake on PTY activity since last HIBERNATE.
0	(HB.SWP) Causes job to be swapped out immediately.

An error return is given if the UUO is not implemented. The SLEEP UUO should be used in this case. A normal return is given after an enabled condition occurs.

Jobs either logged-in as [1,2] or running with the JACCT bit on can wake any hibernating job regardless of the protection code. This allows privileged programs, which are the only jobs that can wake certain system jobs, to be written.

A RESET UUO always clears the protection code and wake-enable bits for the job. Therefore, until the first HIBERNATE UUO is called, there is no protection against wake-up commands from other jobs. To guarantee that no other job wakes the job, a WAKE UUO followed by a HIBERNATE UUO with the desired protection code should be executed. The WAKE UUO ensures that the first HIBERNATE UUO always returns immediately, leaving the job with the correct protection code.

3.1.4.3 WAKE AC, or CALLI AC, 73[1] - The WAKE UUO allows one job to activate a dormant job when some event occurs. This feature can be used with Batch so that when a job wants a core dump taken, it can wake up a dump program. Also, real-time process control jobs can cause other process control jobs to run in response to a specific alarm condition. The WAKE UUO can be called for a RTTRP job running at interrupt level, thereby allowing a real-time job to wake its background portion

[1] This UUO depends on FTHIBWAK which is normally off in the DECsystem-1040.

quickly in order to respond to some real-time condition. (Refer to Paragraph 3.8.1.2 for the restrictions on accumulators when using the RTTRP UUO at interrupt level.)

The call is as follows:

```
MOVE AC, JOBNUM          ;number of job to be awakened
WAKE AC,                 ;or CALLI AC, 73
error return
normal return
```

An error return is given if the proper wake privileges are not specified. There is a wake bit associated with each job. If any of the enabled conditions specified in the last HIBERNATE UUO occurs, then the wake bit is set. The next time a HIBERNATE UUO is executed, the wake bit is cleared and the HIBERNATE UUO returns immediately. The wake bit eliminates the problem of a job going to sleep and missing any wake conditions.

On a normal return, the job has been awakened and has started at the location of the normal return of the HIBER UUO that caused it to become dormant.

3.2 CORE CONTROL

For various reasons, privileged jobs may desire to be locked in core so that they are never to be considered for swapping or shuffling. Some examples of these jobs are as follows:

Real-time jobs	These jobs require immediate access to the processor in response to an interrupt from an I/O device.
Display jobs	The display must be refreshed from a display buffer in the user's core area in order to keep the display picture flicker-free.
Batch	Batch throughput may be enhanced by locking the Batch job controller in core.
Performance analysis	Jobs monitoring the activities of the system need to be locked in core so that they can be invoked quickly with low overhead in order to record activities of the monitor.

3.2.1 Definitions

In swapping and non-swapping systems, unlocked jobs can occupy only the physical core not occupied by locked jobs. Therefore, locked jobs and timesharing jobs contend with one another for physical core memory. In order to control this contention, the system manager is provided with a number of system parameters as described below.

Total User Core is the physical core that can be used for locked and unlocked jobs. This value is equal to total physical core minus the monitor size.

CORMIN is the guaranteed amount of contiguous core that a single unlocked job can have. This value is a constant system parameter and is defined by the system manager at monitor generation time using

MONGEN. It can be changed at monitor startup time using the ONCE ONLY dialogue. This value can range from 0 to Total User Core.

CORMAX is the largest contiguous size that an unlocked job can be. It is a time-varying system parameter that is reduced from its initial setting as jobs are locked in core. In order to satisfy the guaranteed size of CORMIN, the monitor never allows a job to be locked in core if this action would result in CORMAX becoming less than CORMIN. The initial setting of CORMAX is defined at monitor generation time using MONGEN and can be changed at monitor startup time using the ONCE ONLY dialogue. CORMAX can range from CORMIN to Total User Core. A guaranteed amount of core available for locked jobs can be made by setting the initial value of CORMAX to less than Total User Core.

3.2.2 LOCK AC, or CALLI AC, 60[1]

This UUO provides a mechanism for locking jobs in user memory. The user may specify if the high segment, low segment, or both segments are to be locked, and whether the core is to be physically contiguous. Note that on KA10-based systems, core is always allocated contiguously, and that the job may be moved to an extremity of user core before it is locked.

A job may be locked in core if all of the following are true:

a. The job has the LOCK privilege (set from the accounting file ACCT.SYS by LOGIN).

b. The job, when locked, would not prevent another job from expanding to the guaranteed limit, CORMIN.

c. The job, when locked, would not prevent an existing job from running. Note that unlocked jobs can exceed CORMIN.

d. The job when mapped, if specifying exec mapping, would not exceed the maximum amount of exec virtual address space available for locking (KI10 only).

The call is:

```
MOVE AC, [XWD high seg. code, low seg. code]
LOCK AC                                        ;or CALLI AC, 60
error return                                   ;AC contains an error code
normal return
```

The segment codes are a series of bits which specify the way in which the high segment (LH code) and the low segment (RH code) are to be locked. The order and position of the bits in the left half correspond to the order and position of the bits in the right half; that is, to obtain the bit number for the high segment, subtract 18 from the corresponding bit for the low segment. The bits are shown below.

[1] This UUO depends on FTLOCK which is normally off in the DECsystem-1040.

Bit 17 (high segment) Bit 35 (low segment)	If 1, lock the segment in the manner indicated by the following bits. If 0, do not lock the segment; the following bits are ignored.
Bit 16 (high segment) Bit 34 (low segment)	If 0, map contiguously in the exec virtual memory (always implied on the KA10). This causes the segment to be added to the exec virtual address space so that it can be executed in exec mode. For example, this is required when exec mode real-time trapping (RTTRP) is used. On the KI10, the amount of exec virtual address space used by locked jobs is a limited resource with a defined maximum per processor. If mapping the segment would cause the maximum to be exceeded, the LKNEM% error return is given. The maximum amount available can be obtained from the CPU variable GETTAB table for each processor (GETTAB word %CVEVM). The current amount used can also be obtained from the table (%CVEVU). If 1, do not map in exec virtual memory.
Bit 15 (high segment) Bit 33 (low segment)	If 0, lock in contiguous physical memory locations (always implied on the KA10). This causes the segment to be moved and remapped, if necessary, so that its physical core is contiguous. On the KA10 system, the segment is also moved to one end of user core in order to minimize fragmentation of memory. If 1, do not attempt physical contiguity.

If the user requests a segment to be locked in contiguous physical memory, the monitor attempts to lock the segment as low in physical memory as possible. When the segment is locked below 112K, physical and virtual contiguity are equivalent, and thus in this case, virtual contiguity does not require the exec virtual memory resource to achieve contiguity.

On a KA10-based system, physical memory is always allocated contiguously and user segments are directly addressable in exec mode, and therefore, bit codes 1,3,5 and 7 are synonomous.

The setting of bits 33 and 34 (bits 15 and 16) is compatible with the implementation of the LOCK UUO on a KA10-based system. That is, code 1 is the most restrictive, so that a program coded for the KA10 system that implicitly uses these properties will also run on the KI10 system. Applications that do not require all properties can add the appropriate bits to the LOCK UUO's calling sequence.

On a normal return, the job is locked in core. If there is a high segment, the LH of AC contains its absolute address in units of pages (one page is 512 words). The value can be converted to a word address by shifting it left nine bits. If there is no high segment, the LH of AC contains zero. The RH of AC contains the absolute address of the low segment, shifted right nine bits.

On an error return, the job is not locked in core and AC either is unchanged or contains an error code. The AC is unchanged when the LOCK UUO is executed in monitors previous to the implementation of the UUO. An error code indicates the condition that prevented the job from being locked. The error codes are as follows:

Error Code	Name	Explanation
0	LKNIS%	The UUO is not included in this system because it has not been defined with MONGEN or because the appropriate feature test switch is off.
1	LKNLP%	The job does not have locking privileges, or RTTRP privileges, if required.
2	LKNCA%	If the job were locked in core, it would not be possible to run the largest existing non-locked job. (Applies only to swapping systems.)
3	LKNCM%	If the job were locked in core, it would not be possible to meet the guaranteed largest size for an unlocked job, that is, CORMAX would be less than CORMIN.
4	LKNEM%	The mode of locking requested exec virtual memory mapping but the allowable amount of exec mapping has been exhausted.

NOTE

The CORE UUO may be given for the high segment of a locked job only if it is removing the high segment from the addressing space. When the segment is locked in core, the CORE UUO and the CORE command with a non-zero argument cannot be satisfied and, therefore, always give an error return. The program should determine the amount of core needed for the execution and request this amount before executing the LOCK UUO.

Although memory fragmentation is minimized by both the LOCK UUO and the shuffler, the locking algorithm always allows job locking, even though severe fragmentation may take place, as long as

1) all existing jobs can continue to run, and
2) at least CORMIN is available as a contiguous space (see Figure 3-1E).

Therefore, it is important that system managers use caution when granting locking privileges. The following are guidelines for minimizing fragmentation when using the LOCK UUO.

3.2.2.1 KA10 Systems - The guidelines for KA10 systems are:

a. There is no memory fragmentation if two jobs or less are locked in core.
b. There is no fragmentation if the locked jobs do not relinquish their locked status (i.e., no job terminates that has issued a LOCK UUO). In general, jobs with locking privileges should be production jobs.

c. If a job issuing a LOCK UUO is to be debugged and production jobs with locking privileges are to be run, the job to be debugged should be initiated and locked in core first, since it will be locked at the top of core. Then, the production jobs should be initiated since they will all be locked at the bottom of core. This procedure reserves the space at the top of core for the job being debugged and guarantees that there is no fragmentation as it locks and unlocks.

d. With a suitable setting of CORMIN and the initial setting of CORMAX in relation to Total User Core, the system manager can establish a policy which guarantees

1) a maximum size for any unlocked job (CORMIN),

2) a minimum amount of total lockable core for all jobs (Total User Core - CORMAX), and

3) the amount of core which locked and unlocked jobs can contend for on a first-come-first-serve basis (Total User Core - initial CORMAX + CORMIN).

3.2.2.2 Core Allocation Resource - Because routines that lock jobs in core use the swapping and core allocation routines, they are considered a sharable resource. This resource is the semipermanent core allocation resource (mnemonic=CA). When a job issues a LOCK UUO and the system is currently engaged in executing a LOCK UUO for another job, the job enters the queue associated with the core allocation resource. Because a job may share a queue with other jobs and because swapping and shuffling may be required to position the job to where it is to be locked, the actual execution time needed to complete the process of locking a job might be on the order of seconds.

When it has been established that a job can be locked, the low segment number and the high segment number (if any) are stored as flags to activate the locking routines when the swapper and shuffler are idle. The ideal position for the locked job is also stored as a goal for the locking routines. In KA10 swapping systems, the ideal position is always achieved to guarantee minimum fragmentation. In nonswapping systems, minimum fragmentation is achieved only if the ideal position does not contain an active segment (see Figure 3-1).

In swapping systems, after the job is locked in core, the locking routine determines the size of the new largest contiguous region available to unlocked jobs. This value will be greater than or equal to CORMIN. If this region is less than the old value of CORMAX, then CORMAX is set equal to the size of the new reduced region. Otherwise, CORMAX remains set to its old value.

3.2.2.3 UNLOK. AC, or CALLI AC, 120[1] - This UUO provides a mechanism for a job to unlock itself without doing a RESET UUO. The user can specify if one or both segments are to be unlocked. The call is:

```
MOVSI AC, 1          ;if high segment is to be unlocked
MOVSI AC, 0          ;if no high segment, or if high segment
                     ;is not to be unlocked
HRRI AC, 1           ;if low segment is to be unlocked.
HRRI AC, 0           ;if low segment is not to be unlocked.
UNLOK. AC,           ;or CALLI AC, 120
error return
normal return
```

[1] This UUO depends on FTLOCK which is normally off in the DECsystem-1040.

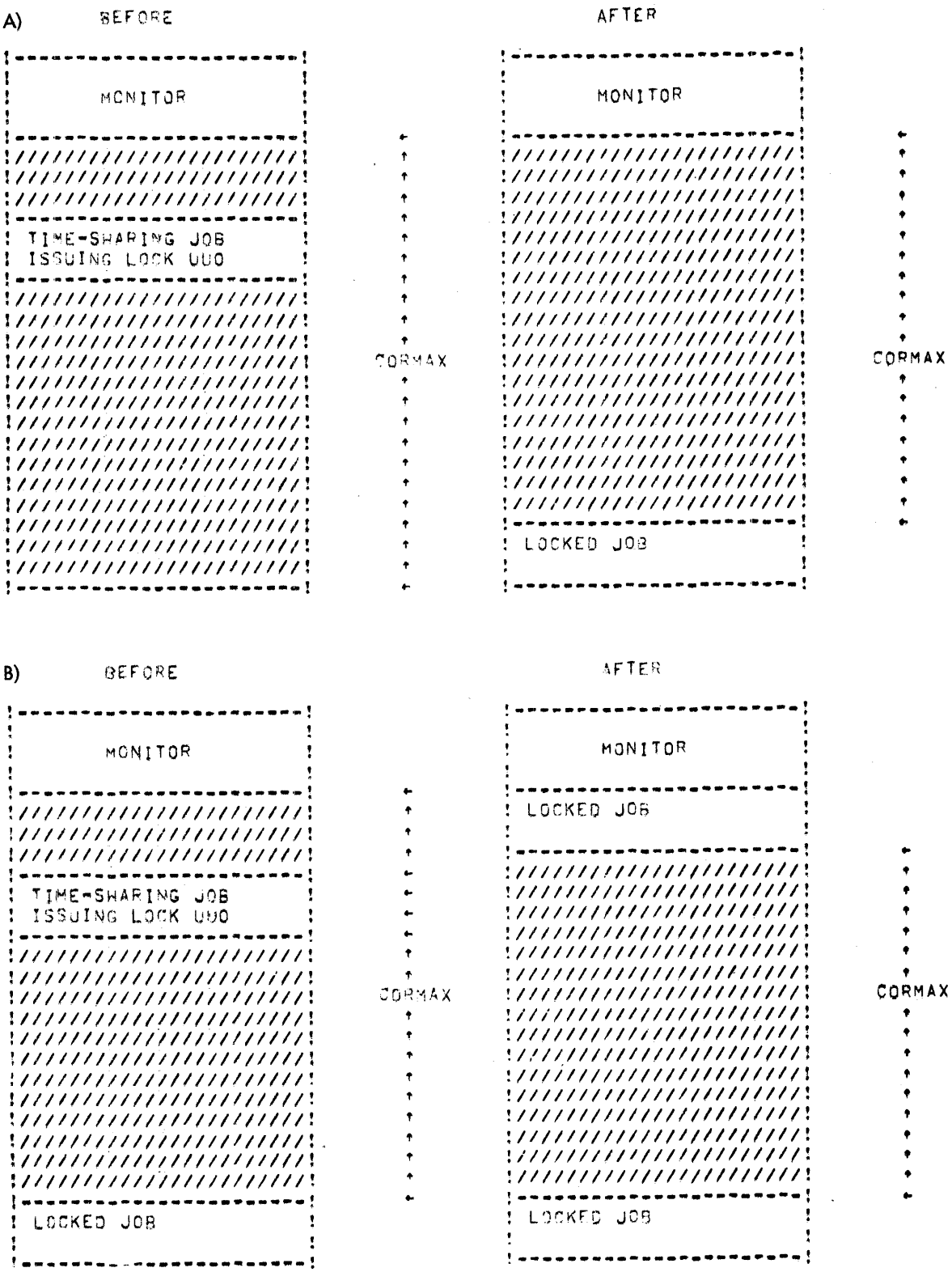


Figure 3-1 Locking Jobs In Core on KA10 Systems

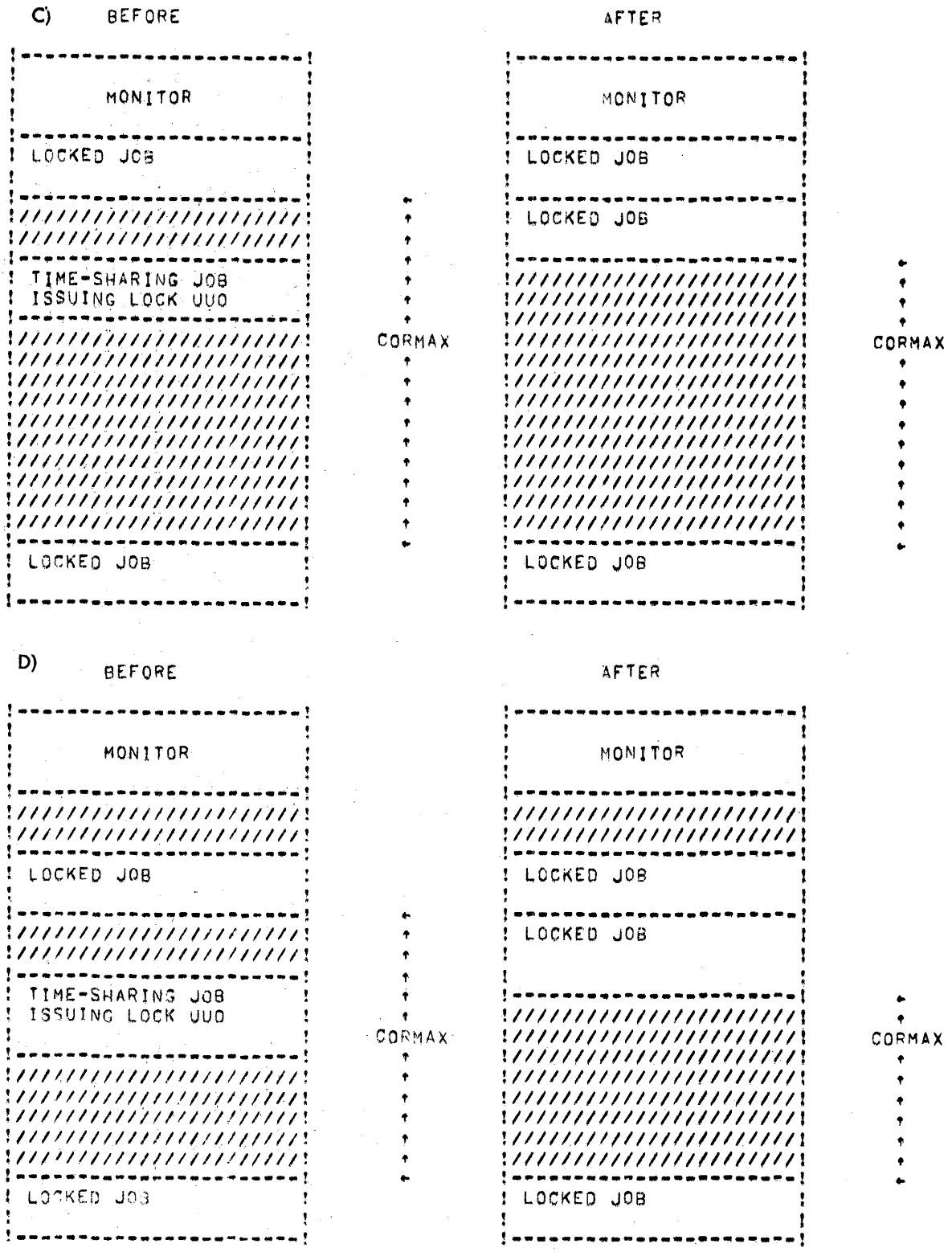


Figure 3-1 Locking Jobs In Core on KA10 Systems (Cont)

E) Unlikely Fragmentation Case

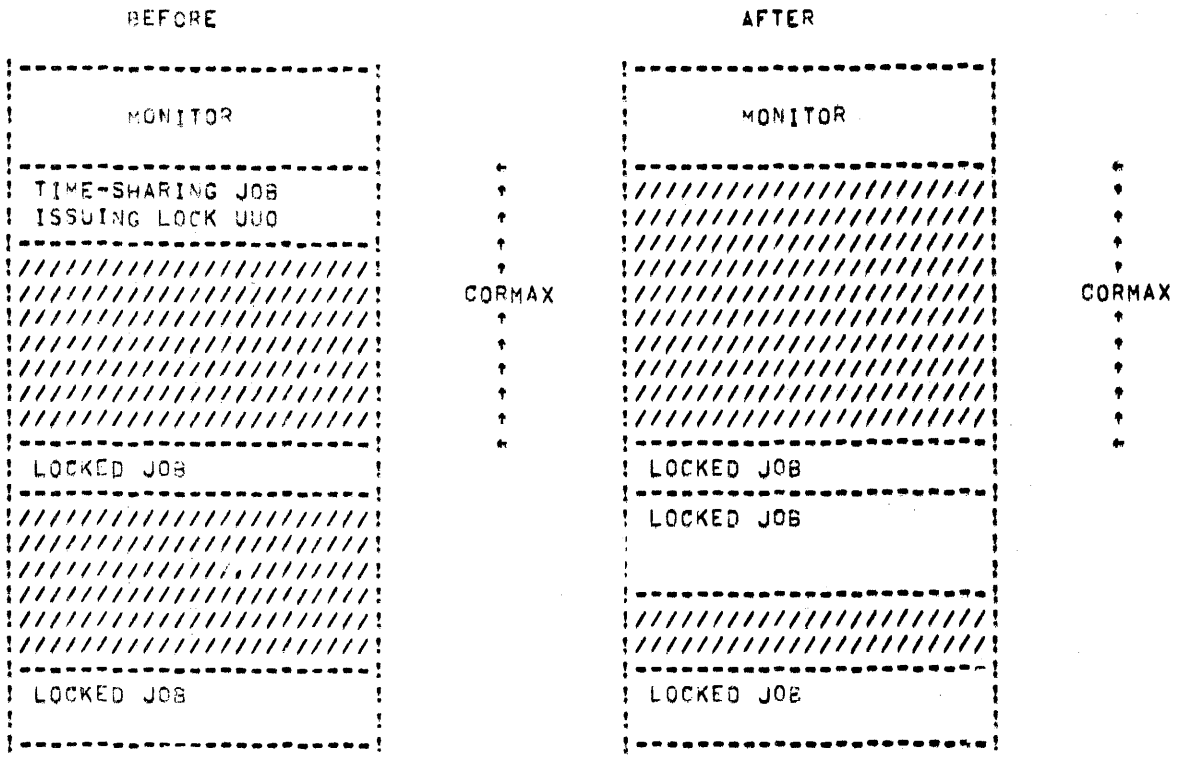


Figure 3-1 Locking Jobs In Core on KA10 Systems (Cont)

An error return is given if the UUO is not implemented. If this is the case, a job can relinquish its locked status when either the user program executes an EXIT or RESET UUO, or the monitor performs an implicit RESET for the user. Implicit RESETS occur when

a. The user program issues a RUN UUO, or

b. The user types any of the following monitor commands: R, RUN, GET, SAVE, SSAVE, CORE 0, and any system program-invoking command.

NOTE

If several jobs are sharing a locked high segment, the high segment is unlocked only when the SN%LOK bit is turned off for all jobs sharing the segment (i.e., when all jobs which executed the LOCK UUO have performed the unlock function) (refer to GETTAB table 14).

On a normal return, the segment (or job) is unlocked and becomes a candidate for swapping and shuffling. Any meter points (METER.UUO) are deactivated and, if the low segment is unlocked, any real-time devices are RESET. CORMAX is increased to reflect the new size of the largest contiguous region available to unlocked jobs. However, CORMAX is never set to a greater value than its initial setting.

3.2.3 CORE AC, or CALLI AC, 11

This UUO provides a user program with the ability to expand and contract its core size as its memory requirements change. To allocate core in either or both segments, the left half of AC is used to specify the highest user address to be assigned to the high segment and the right half is used to specify the highest user address in the low segment. The monitor will assign the smallest amount of core which will satisfy the request. If the left half of AC contains 0, the high segment core assignment is not changed. If the left half of AC is non-zero and is either less than 400000 or the length of the low segment, whichever, is greater, the high segment is eliminated. If this is executed from the high segment, an illegal memory error message is printed when the monitor attempts to return control to the illegal address.

A RH of 0 leaves the low segment core assignment unaffected. The monitor clears new core before assigning it to the user; therefore, privacy of information is ensured.

The error return is given if:

1) The LH is greater than or equal to 400000 and the system does not have a two-segment capability.
2) The LH is greater than or equal to 400000 and the user has been meddling without write access privileges (refer to Paragraph 6.2.3).
3) The LH and the RH are both zero.

In swapping systems, this programmed operator returns the maximum number of 1K core blocks (all of core minus the monitor, unless an installation chooses to restrict the amount of core) available to the user. By restricting the amount of core available to users, the number of jobs in core simultaneously is increased. In nonswapping systems, the number of free and dormant 1K blocks is returned; therefore, the CORE UUO and the CORE command return the same information.

For compatibility, the KI10 also returns the number of 1K blocks available even though core is allocated in 512-word pages. The value returned is truncated to the nearest multiple of 1K (e.g., if 21 pages are available, the value returned in 10K). If it is necessary to obtain the exact amount of core available in units of pages, the user can examine the monitor location CORMAX (in GETTAB table 12) with the GETTAB UUO (refer to Paragraph 3.6.3.4). CORMAX is the maximum number of words available to the user and thus can be converted to either pages or K.

The call is:

```
MOVE AC [XWD HIGH ADR or 0, LOW ADDR or 0]
CORE AC,                                        ;or CALLI AC, 11
error return
normal return
```

The CORE UUO re-assigns the low segment (if RH is non-zero) and then re-assigns the high segment (if LH is non-zero). If the sum of the new low segment and the old high segment exceeds the maximum amount of core allowed to a user, the error retum is given, the core assignment is unchanged, and the maximum core available to the user for high and low segments (in 1K blocks) is returned in the AC. In a nonswapping system, the number of free and dormant 1K blocks is returned.

If the sum of the new low segment and the new high segment exceeds the maximum amount of core allowed to a user, the error retum is given, the new low segment is assigned, the old high segment remains, and the maximum core available to the user in 1K blocks is returned in the AC. Therefore, to increase the low segment and decrease the high segment at the same time, two separate CORE UUOs should be used to reduce the chances of exceeding the maximum size allowed to a user job. If the new low segment extends beyond 377777, the high segment shifts up into the virtual addressing space instead of being overlaid. If a long low segment is shortened to 377777 or less, the high segment shifts from the virtual addressing space to 400000 instead of growing longer or remaining where it was. If the high segment is a program, it does not execute properly after a shift unless it is a self-relocating program in which all transfer instructions are indexed.

If the high segment is eliminated by a CORE UUO, a subsequent CORE UUO, in which the LH is greater than 400000, will create a new, nonsharable segment rather than re-establishing the old high segment. This segment becomes sharable after it has been:

a. Given an extension .SHR.
b. Written onto the storage device.
c. Closed so that a directory entry is made.
d. Initialized from the storage device by GET, R, or RUN commands or RUN or GETSEG UUOs.

The loader and the SAVE and GET commands use the above sequence to create and initialize new sharable segments.

A user program which expands core should compare its highest desired address with its highest legal address obtained from the Job Data Area location .JBREL (refer to Chapter 1). If the desired address is greater than the highest legal address, the program should execute a CORE UUO for the new desired address (not for the highest old legal address plus 512 or 1024). The monitor then updates .JBREL by the number of words in its basic core allocation unit (i.e., 1024 words on the KA10 processor or 512 words on the KI10 processor). Subsequent compares of the desired address and the highest legal address do not cause a CORE UUO until the next increase of core is required. If used this way, a CORE UUO will execute on both the KA10 and KI10 processors and will require less monitor CPU time because the number of CORE UUOs needed will be minimized.

The following example illustrates the method for obtaining core only when needed.

```
;SUBROUTINE TO GET CORE ONLY WHEN NEEDED
;CALL:  MOVE    T1, HIGHEST DESIRED ADDRESS
;       PUSHJ   P,CHKCOR
;       RETURN HERE UNLESS NO MORE CORE

CHKCOR: CAMLE   T1,.JBREL##     ;GREATER THAN HIGHEST LEGAL ADDRESS?
        POPJ    P,              ;NO, PRESENT CORE BIG ENOUGH.
        CORE    T1,             ;YES, GET NEXT INCREMENT OF CORE.
          JRST   ERROR          ;NOT AVAILABLE.
        POPJ    P,              ;NEXT INCREMENT ASSIGNED.
```

3.2.4 SETUWP AC, or CALLI AC, 36

This UUO allows a user program to set or clear the hardware user-mode write protect bit and to obtain the previous setting. It must be used if a user program is to modify the high segment.
The call is:

```
SETUWP AC,                  ;OR CALLI AC, 36
error return
normal return
```

If the system has a two-register capability, the normal return will be given unless the user has been meddling without write privileges, in which case an error return will be given. A normal return is given whether or not the program has a high segment, because the reentrant software is designed to allow users to write programs for two-register machines, which will run under one-register machines. Compatibility of source and relocatable binary files is, therefore, maintained between one-register and two-register machines.

If the system has a one-register capability, the error return (bit 35 of AC=0) is given. This error return allows the user program to find out whether or not the system has a two-segment capability. The user program specifies the setting of the user-mode write protect bit in bit 35 of AC (write protect = 1, write privileges = 0). The previous setting of the user-mode write protect bit is returned in bit 35 of AC, so that any user subroutine can preserve the previous setting before changing it. Therefore, nested user subroutines, which either set or clear the bit, can be written, provided the subroutines save the previous value of the bit and restore it on returning to its caller.

3.3 SEGMENT CONTROL

3.3.1 RUN AC, or CALLI AC, 35

This UUO has been implemented so that programs can transfer control to one another. Both the low and high segments of the user's addressing space are replaced with the program being called.

The call is:

```
MOVSI AC, starting address increment
HRRI AC, adr of six-word argument block
RUN AC,                                  or CALLI AC, 35
error return (unless HALT in LH)
[normal return is not here, but to start-
ing address plus increment of new program]
```

The arguments contained in the six-word block are:

E: SIXBIT/logical device name/ SIXBIT/filename/	;for either or both high and low files
SIXBIT/ext. for low file/	;if LH = 0, .LOW is assumed if high segment exists, .SAV is assumed if high segment does not exist.
0 XWD proj. no., prog. no. XWD 0, optional core assignment	;if = 0, use current user's proj, prog ;RH = new highest user address to be assigned to low segment. LH is ignored rather than setting high segment.

A user program usually will specify only the first two words and set the others to 0. The RUN UUO destroys the contents of all of the user's ACs and releases all the user's I/O channels; therefore, arguments or devices cannot be passed to the next program.

The RUN UUO to certain system programs (e.g., LOGIN, LOGOUT) automatically sets the appropriate privileged bits (JACCT and JLOG). These bits are not set (or are turned off if they were set) for programs that are not privileged programs from device SYS or for programs whose starting address offset is greater than 1. Assigning a device as SYS does not cause these bits to be set.

The RUN UUO clears all of core. However, programs should not count on this action, and must still initialize core to the desired value to allow programs to be restarted by a ↑C, START sequence without having to do I/O.

Programs on the system library should be called by using device SYS with a zero project-programmer number instead of device DSK with the project-programmer number [1, 4]. The extension should also be 0 so that the calling user program does not need to know if the called system program is reentrant or not.

The LH of AC is added to and stored in the starting address (.JBSA) of the new program before control is transferred to it. The command ↑C followed by the START command restarts the program at the location specified by the RUN UUO, so that the user can start the current system program over again.

The user is considered to be meddling with the program (refer to Paragraph 3.3.5) if the LH of AC is not 0 or 1 unless the program being run is execute-only for this job. In this case, the offset is treated as 0.

Programs accept commands from a terminal or a file, depending on how they were started, due to control by the program calling the RUN UUO. The following convention is used with all of DEC's standard system programs: 0 in LH of AC means type an asterisk and accept commands from the terminal. A 1 means accept commands from a command file, if it exists; if not, type an asterisk and accept commands from the terminal. The convention for naming system program command files is that

the filename be of the form

###III.TMP

where III are the first three (or fewer if three do not exist) characters of the name of the program doing the LOOKUP, and ### is the decimal character expansion (with leading zeroes) of the binary job number. The job number is included to allow a user to run two or more jobs under the same project-programmer number. For example,

009PIP.TMP
039MAC.TMP

Decimal numbers are used so that a user listing his directory can see the same number as the PJOB command types. These command files are temporary and may, therefore, be deleted by the KJOB program (refer to KJOB command and Appendix C in DECsystem-10 Operating System Commands).

At times it is necessary to remember the arguments that a user typed in to invoke a program (i.e, the arguments on a GET or RUN command). For example, the COBOL program needs these arguments in order to GETSEG the next overlay from the same place. In all monitors, when the program is first started, this information can be obtained from the following accumulators:

AC0 (.SGNAM) contains the filename.
AC7 (.SGPPN) contains the directory name.
AC11 (.SGDEV) contains the device name.
AC17 (.SGLOW) contains the extension of the low segment.

Note that the starting address should be changed by the program so that a ↑C, START sequence will not destroy the remembered arguments in the ACs. This information should not be used when desiring to save the current segment name (GETTAB should be used in this case), but rather when obtaining the call arguments before calling the next segment.

The RUN UUO can give an error return with an error code in AC if any errors are detected; thus, the user program may attempt to recover from the error and/or give the user a more informative message on how to proceed. Some user programs do not go to the bother of including error recovery code.

The monitor detects this and does not give an error return if the LH of the error return location is a HALT instruction. If this is the case, the monitor simply prints its standard error message for that type of error and returns the user's terminal to monitor mode. This optional error recovery procedure also allows a user program to analyze the error code received and then execute a second RUN UUO with a HALT if the error code indicates an error for which the monitor message is sufficiently informative or one from which the user program cannot recover.

The error codes are an extension of the LOOKUP, ENTER, and RENAME UUO error codes and are defined in the S.MAC monitor file. Refer to Appendix E for an explanation of the error codes.

The monitor does not attempt an error return to a user program after the high or low segment containing the RUN UUO has been overlaid. The UUO should be placed in the low segment in case the error is discovered after the high segment has been released.

To successfully program the RUN UUO for all size systems and for all system programs with a size that is not known at the time the RUN UUO is coded, it is necessary to understand the sequence of operations the RUN UUO initiates. Assume that the job executing the RUN UUO has both a low and a high segment. (It can be executed from either segment; however, fewer errors can be returned to the user if it is executed from the high segment.)

The sequence of operations for the RUN UUO is as follows:

1. Does a high segment already exist with desired name?
   If yes, go to 30.
   INIT and LOOKUP filename .SHR. If not found, go to 10.
   Read high file into top of low segment by extending it. (Here the old segment and new high segment and old high segment together may not exceed the maximum user core legally available to this job at the time of the UUO nor may it cause the total amount of virtual core assigned to all users to exceed the size of the swapping space.)

   REMAP the top of low segment replacing old high segment in logical addressing space.
   If high segment is sharable (.SHR) store its name so others can share it.
   Always go to 40 or return to user if GETSEG UUO.

10. LOOKUP filename .HGH. If not found, go to 41 or error return to user if GETSEG UUO.
    Read high file into top of low segment by extending it. (The old low segment and new high segment and old high segment together may not exceed the maximum user core legally available to this job at the time of the UUO nor may it cause the total amount of virtual core assigned to all users to exceed the size of the swapping space.)
    Check for I/O errors. If any, error return to user unless HALT in LH of return.
    Go to 41.

30. Remove old high segment, if any, from logical addressing space.
    Place the sharable segment in user's logical addressing space. Go to 40 or return to user if GETSEG UUO.

35. Remove old high segment, if any, from logical addressing space.
    (Go to 41).

40. Copy vestigial job data area into job data area.
    Does the new high segment have a low file
    (LH of .JBCOR >137)?
    If not, go to 45.

41. LOOKUP filename .SAV or .LOW or user specified extension. Error if not found. Return to user if there is no HALT in LH of error return, provided that if the CALL is from the high segment, it is still the original high segment and has not been removed from the user's addressing space. Otherwise, the monitor prints one of the following error messages:

?NOT A SAVE FILE
?filename .SAV NOT FOUND
?TRANSMISSION ERROR
?LOOKUP FAILURE n
?nK OF CORE NEEDED
?NO START ADR

and stops the job.
Reassign low segment core according to size of file or user specified core argument, whichever is larger. Previous low segment is overlaid. Read low file into beginning of low segment. Check for I/O errors. If there is an error print error message and do not return to user. If there are no errors, perform START.

45. Reassign low segment core according to larger of user's core argument or argument when file saved (RH of .JBCOR).

NOTE

To be guaranteed of handling the largest number of errors, the cautious user should remove his high segment from high logical addressing space (use CORE UUO with a one in LH of AC). The error handling code should be put in the low segment along with the RUN UUO and the size of the low segment reduced to 1K. A better idea would be to have the error handling code written once and put in a seldom used (probably nonsharable) high segment, which could be gotten in high segment using GETSEG UUO (see below) when an error return occurs to low segment on a RUN UUO.

3.3.2 GETSEG AC, or CALLI AC, 40

This UUO has been implemented so that a high segment can be initialized from a file or shared segment without affecting the low segment. It is used for shared data segments, shared program overlays, and run-time routines such as FORTRAN or COBOL object time systems. This programmed operator works exactly like the RUN UUO with the following exceptions:

a. No attempt is made to read a low file.

b. The accumulators are not preserved. The only change made to JOBDAT is to set the left half of .JBHRL to 0 (a SAVE command then saves all of the high segment) and the right half to the highest legal user address.

c. If an error occurs, control is returned to the location of the error return, unless the left half of the location contains a HALT instruction.

d. On a normal return, the control is returned to two locations following the UUO, whether it is called from the low or high segment. It should be called from the low segment unless the normal return coincides with the starting address of the new high segment.

e. User channels 1 through 17 are not released so the GETSEG UUO can be used for program overlays, such as the COBOL compiler. Channel 0 is released because it is used by the UUO.

f. .JBSA and .JBREN are zeroed if they point to a high segment that is being removed. This produces the message:

?NO START ADDRESS

if a START or REENTER command is given.

Refer to steps 1 through 30 of the RUN UUO description (Paragraph 3.3.1) for details of GETSEG UUO operation.

3.3.3 REMAP AC, or CALLI AC, 37

This UUO takes the top part of a low segment and remaps it into the high segment. The previous high segment (if any) will be removed from the user's addressing space. The new low segment will be the previous low segment minus the amount remapped.

The call is:

```
MOVEI AC, desired highest adr in low segment
REMAP AC,                                      ;or CALLI AC, 37
error return
normal return
```

The monitor rounds up the address to the nearest core allocation unit of either 1024_{10} (2000_8) words on KA10-based systems or 512_{10} (1000_8) words on KI10-based systems. If the argument exceeds the length of the low segment, remapping will not take place, the high segment will remain unchanged in the user's addressing space, and the error return will be taken. The error return will also be taken if the system does not have a two-register capability. The content of AC is unchanged. The content of .JBREL (refer to Paragraph 1.2.1) is set to the new highest legal user address in the low segment. The LH of .JBHRL is set to 0 (a SAVE command then saves all of the high segment) and the RH is set to the highest legal user address in the high segment (401777 or greater or 0). The hardware relocation will be changed, and the user-mode write protect bit will be set.

This UUO is used by the LOADER to load reentrant programs, which make use of all of physical core. Otherwise, the LOADER might exceed core in assigning additional core and moving the data from the low to the high segment with a BLT instruction. The GET command also uses this UUO to perform I/O into the low segment instead of the high segment.

3.3.4 Testing for Sharable High Segments

Occasionally, it is desirable for a program to determine whether its high segment is sharable. If the high segment is sharable, the program may decide not to modify itself. The following code tests the high segment whether or not 1) the system has a high segment capability or 2) the job has a high segment.

```
HRROI   T, .GTSGN       ;see if high segment is sharable
GETTAB  T,              ;look at monitor .GTSGN table
JRST    .+2             ;table or UUO not present
TLNN    T, (SN%SHR)     ;is sharable bit on ?
JRST    NOTSHR          ;no, go ahead and modify here
                        ;if high segment is sharable.
```

3.3.5 Modifying Shared Segments and Meddling

A high segment is usually write-protected, but it is possible for a user program to turn off the user write-protect bit or to increase or decrease a shared segment's core assignment by using the SETUWP or CORE UUO. These UUOs are legal from the high or low segment if the sharable segment has not been "meddled" with, unless the user has write privileges for the file that initialized the high segment. Even the malicious user can have the privilege of running such a program, although he does not have the access rights to modify the file used to initialize the sharable segment.

Meddling is defined as any of the following, even if the user has privileges to write the file which initialized the sharable segment.

a. START or CSTART commands with an argument.
b. DEPOSIT command in the low or high segment.
c. RUN UUO with anything other than a 0 or 1 in LH of AC as a starting address increment.
d. GETSEG UUO.

It is not considered meddling to perform any of the above commands or UUOs with a nonsharable program. It is never considered meddling to type ↑C followed by START (without an argument), CONT, CCONT, CSTART (without an argument), REENTER, DDT, SAVE, or E command.

When a sharable program is meddled with, the monitor sets the meddle bit for the user. An error return is given when the clearing of the user write-protect bit is attempted with the SETUWP UUO or when the reassignment of core for the high segment (except to remove it completely) is attempted with the CORE UUO. An attempt to modify the high segment with the DEPOSIT command causes the message

OUT OF BOUNDS

to be printed. If the user write-protect bit was not set when the user meddled, it will be set to protect the high segment in case it is being shared. The command and the two UUOs are allowed in spite of meddling, if the user has the access privileges to write the file which initialized the high segment.

A privileged programmer is able to supersede a sharable program, which is in the process of being shared by a number of users. When a successful CLOSE, OUTPUT, or RENAME UUO is executed for a file with the same directory name and filename (previous name if the RENAME UUO is used) as the segment being shared, the name of the segment is set to 0. New users do not share the older version, but they do share the newer version. This requires the monitor to read the newly created file only once to initialize it. The monitor deletes the older version when all users are finished sharing it.

Users with access privileges are able to write programs that access sharable data segments via the GETSEG UUO (which is meddling) and then turn off the user write-protect bit using SETUWP UUO. With DECtape, write privileges exist if it is assigned to the job (cannot be a system tape) or is not assigned to any job and is not a system tape.

When control can be transferred only to a small number of entry points (two), which the shared program is prepared to handle, then the shared program can do anything it has the privileges to do, although the person running the program does not have these privileges.

The ASSIGN (and the DEASSIGN, DISMOUNT/REMOV, FINISH, KJOB commands if the device was previously assigned by console) command clears all shared segment names currently in use, which were initialized for the device, if the device is removable (DTA, MTA). Otherwise, new users could continue to share the old segment indefinitely, even if a new version were mounted on the device. Therefore, it is possible to update the library during regular timesharing, if the programmer has access privileges.

3.4 PROGRAM AND PROFILE IDENTIFICATION

3.4.1 SETNAM AC, or CALLI AC, 43

This UUO is used by the LOADER. The content of AC contains a left-justified SIXBIT program name, which is stored in a monitor job table. The information in the table is used by the SYSTAT program (refer to Table 3-1 in Paragraph 3.6.3.3). This UUO clears the "SYS:" program bit JB.LSY (used by Batch), clears the execute-only bit, and outputs a SET WATCH VERSION number (refer to DECsystem-10 Operating System Commands).

3.4.2 SETUUO AC, or CALLI AC, 75[1]

This UUO is used to set various system or job parameters. To set system parameters, the user must be logged in under [1, 2] or the job must be running with the JACCT bit set. Refer to the Specifications section of the DECsystem-10 Software Notebooks for a complete description of the privileged functions.

The contents of AC contain a function code in the left half and an argument in the right half. The call is:

```
MOVE AC, [XWD function, argument]
SETUUO AC,                          ;or CALLI AC, 75
error return
normal return
```

[1] This UUO depends on FTSET which is normally off in the DECsystem-1040. If FTSET is on, individual functions depend on the other feature test switches as noted in the text.

The functions and arguments are as follows:

Function	Name	Argument
0	.STCMX	CORMAX. Privileged function.
1	.STCMN	CORMIN. Privileged function.
2	.STDAY	DAYTIME. Privileged function (FTSEDAT).
3	.STSCH	SCHED. Privileged function.
4	.STCDR	CDR (input name counter for this job). Not a privileged function. If AC is non-zero, the content is the same as the next input name. If AC is 0, the current counter is returned in AC (FTSPL).
5	.STSPL	SPOOL for this job. Not a privileged function unless the user is unspooling devices. Bits are bits 31-35 of .GTSPL (FTSPL). Bit 35 JS.PLP line printer spooling Bit 34 JS.PPL plotter spooling Bit 33 JS.PPT paper tape punch spooling Bit 32 JS.PCP card punch spooling Bit 31 JS.PCR card reader spooling
6	.STWTC	WATCH for this job. Not a privileged function. Bits are bits 1-6 of .GTWCH (FTWATCH). Bit 1 JW.WDY watch time of day Bit 2 JW.WRN watch run time Bit 3 JW.WWT watch wait time Bit 4 JW.WDR watch disk reads Bit 5 JW.WDW watch disk writes Bit 6 JW.WVR watch version numbers.
7	.STDAT	DATE. Privileged function (FTSEDAT).
10	.STOPR	OPR. Privileged function.
11	.STKSY	KSYS. Privileged function (FT5UUO).
12	.STCLM	CORE limit. Privileged function (FTTLIM).
13	.STTLM	TIME limit for this job. Privileged function (FTTLIM).
14	.STCPU	CPU specification for this job. The following bits select the CPU on which the job is allowed to run. Bit 35 SP.CR0 CPU0 Bit 34 SP.CR1 CPU1 Bit 33 SP.CR2 CPU2 Bit 32 SP.CR3 CPU3 Bit 31 SP.CR4 CPU4 Bit 30 SP.CR5 CPU5
15	.STCRN	CPU runnability. Privileged function.
16	.STLMX	LOGMAX. Privileged function.

Function	Name	Argument
17	.STBMX	BATMAX. Privileged function.
20	.STBMN	BATMIN. Privileged function.
21	.STDFL	DSKFUL for this job. Not a privileged function. An argument of 0 (.DFPSE) causes a pause and an argument of 1 (.DFERR) causes an error when the disk is full or the user's quota is exceeded. The current setting can be determined by issuing an argument other than 0 or 1. The value returned is either 0 or 1 depending on whether PAUSE or ERROR is set. The initial setting is ERROR.

The error return is given if 1) the UUO is not implemented, 2) the user does not have the correct privileges for the function specified, or 3) the argument specified is invalid.

On a normal return, AC remains unchanged.

3.4.3 LOCATE AC, or CALLI AC, 62[1]

This UUO is used to change the logical station associated with the user's job. The call is:

```
MOVEI AC, station number
LOCATE AC,                    ;or CALLI AC, 62
error return
normal return
```

The station number requested is contained in AC as follows:

- -1 changes the job's location to the physical station of the job's controlling terminal.
- 0 changes the job's location to the central station.
- n changes the job's location to remote station n.

The normal return is taken if the UUO is implemented, the station is defined, and the station is in contact. Subsequent generic device specifications are at the new station. The error return is taken if the UUO is not implemented or the specified station is illegal or not in contact.

3.5 INTER-PROGRAM COMMUNICATION

3.5.1 TMPCOR AC, or CALLI AC, 44[2]

This UUO allows a job to leave several short files in core from the running of one user program or system program to the next. These files are referenced by a three-character filename and are unique

[1] This UUO depends on FTREM which is normally off in the DECsystem-1040.

[2] This UUO depends on FTTMP which is normally off in the DECsystem-1040.

to each job. All files are deleted when the job is killed. This system of temporary storage improves response time and reduces the number of disk operations. If this UUO fails, the file specification DSK:nnnNAM.TMP, where nnn is the job number and NAM is the three-character filename, should be used for temporary disk storage.

Each temporary file appears to the user as one dump mode buffer. The actual size of the file, the number of temporary files a user can have, and the total core a user can use for temporary storage are parameters determined at MONGEN time. All temporary files reside in a fixed area, but the space is dynamically allocated among different jobs and several different files for any given job.

The call is:

```
        MOVE AC, [XWD CODE, BLOCK]
        TMPCOR AC,                          ;or CALLI AC, 44
        error return
        normal return
        .
        .
        .
BLOCK:  XWD NAME, 0                         ;NAME is filename
        IOWD BUFLEN, BUFFER                 ;user buffer area
                                            ;(zero for no buffer)
```

The AC must be set by the user program prior to execution of the UUO and is changed by the UUO on return to a value that depends on the particular function performed. Functions of the TMPCOR UUO are presented in the following paragraphs.

3.5.1.1 CODE = 0 (.TCRFS), Obtain Free Space - This is the only form of the UUO that does not use a two-word parameter block and, therefore, the contents of AC are ordinarily set to 0. A normal return is given (unless the UUO is not implemented), and the number of the free words available to the user is returned in AC.

3.5.1.2 CODE = 1 (.TCRRF), Read File - If the specified file is not found, the number of free words available for temporary files is returned in AC and the error return is taken. If the specified file is found, the length of the file in words (that is, the length in BUFLEN when writing the file rounded up to the next highest multiple of four) is returned in AC, and as much of the file as possible is copied into the user's buffer. The user may check for truncation of the file by comparing the contents of AC with BUFLEN.

3.5.1.3 CODE = 2 (.TCRDF), Read and Delete File - This function is similar to CODE = 1, except that if the specified file is found, it is deleted and its space is reclaimed.

3.5.1.4 CODE = 3 (.TCRWF), Write File - If a file exists with the specified name, it is deleted and its space reclaimed. The requested size of the file is the value in BUFLEN rounded up to the next highest multiple of four. If there is enough space

a. The file is written.

b. The number of remaining blocks is returned in AC.

c. The normal return is taken.

If there is not enough space to completely write the file

a. The file is not written.

b. The number of free words available to the user is returned in AC.

c. The error return is taken.

3.5.1.5 CODE = 4 (.TCRRD), Read Directory - The number of different files in the temporary file area of the job is returned in AC. An entry is made for each file in the user's buffer area until either there is no more space or all files have been listed. The error return is never taken. The user may check for truncation of the entries by comparing the contents of AC with BUFLEN. The format of a directory entry is as follows:

XWD NAME, SIZE

where NAME is the filename and SIZE is the file length in words.

3.5.1.6 CODE = 5 (.TCRDD), Read and Clear Directory - This function is similar to CODE = 4, except that any files in the temporary storage area of the job are deleted and their space is reclaimed. This UUO is used by the LOGOUT program.

3.6 ENVIRONMENTAL INFORMATION

3.6.1 Timing Information

The 5.05 and later monitors use two time and two date standards. The time accounting is performed by two clocks. The APR clock, driven by the power source frequency (60 Hz in North America, 50 Hz in most other countries), is accurate over long periods of time. For this reason, it is used to keep the time of day, e.g., for the TIMER UUO. It can also be used for runtime accounting measurement (i.e., keeping track of the processor time each job uses). However, there will be some loss of accuracy since the time intervals in which a job runs are often less than the period of the APR clock.

The DK10 clock, a 100000 Hz clock, is accurate over short periods of time. It is used to perform runtime accounting, and thereby achieves greater accuracy than the APR clock.

The traditional DECsystem-10 date (returned with the DATE UUO) is a 15-bit integer. This integer is incremented by 1 each day, by 31 each month (regardless of the actual number of days in the month), and by 12*31 each year (also regardless of the actual number of days in the year). This date format is easy to resolve into year-month-day; however, the difference between two dates in this format is not necessarily the actual number of days between them.

A universal date-time standard (GETTAB table 11, item 53) is also used in which the left half of the word is the date and the right half is the time. The date is uniformly incremented each day (at midnight, Greenwich Mean Time) with 1 being November 18, 1858. This date is consistent with the Smithsonian Astronomical Date Standard and other computer systems. The time is a fraction of a day. Thus, the 36-bit quantity is in units of days with a binary point between the left and right halves. The resolution is approximately 1/3 of a second; that is, the least significant bit (bit 35) represents approximately 1/3 of a second. Since the time is Greenwich Mean Time (GMT), all installations have a date-time reference which is independent of location and local time conventions.

For convenience, the monitor maintains a set of GETTAB values which gives the local date and time in terms of year, month, day, hours, minutes, and seconds (GETTAB table 11, items 56-63).

3.6.1.1 DATE AC, or CALLI AC, 14 - A 15-bit binary integer computed by the formula

date=((year-1964)x12+(month-1))x31+day-1

represents the date.
This integer representation is returned right justified in AC.

3.6.1.2 TIMER AC, or CALLI AC, 22 - This UUO returns the time of day, in clock ticks (jiffies), right justified in AC. A jiffy is 1/60 of a second (16.6 milliseconds) for 60-cycle power and 1/50 of a second (20 milliseconds) for 50-cycle power. The MSTIME UUO should normally be used so that the time is not a function of the cycle.

3.6.1.3 MSTIME AC, or CALLI AC, 23 - This UUO returns the time of day, in milliseconds, right justified in AC.

3.6.2 Job Status Information

3.6.2.1 RUNTIM AC, or CALLI AC, 27 - The accumulated running time (in milliseconds) of the job number specified in AC is returned right justified in AC. If the job number in AC is zero, the running time of the currently running job is returned. If the job number in AC does not exist, zero is returned.

3.6.2.2 PJOB AC, or CALLI AC, 30 - This UUO returns the job number right justified in AC.

3.6.2.3 GETPPN AC, or CALLI AC, 24 - This UUO returns in AC the project-programmer pair of the job. The project number is a binary number in the left half of AC, and the programmer number is a binary number in the right half of AC. If the program has the JACCT bit set, a skip return is given if the old project-programmer number is also logged in on another job.

3.6.2.4 OTHUSR AC, or CALLI AC, 77 - This UUO is used to determine if another job is logged in with the same project-programmer number as the job executing the UUO. The non-SKIP return is given if

1) the UUO is not implemented, in which case the AC remains unchanged, or
2) the UUO is implemented and no other jobs are logged in with the same project-programmer number, in which case the AC contains the project-programmer number of the job executing the UUO.

The SKIP return is given if the UUO is implemented and other jobs are logged in with the same project-programmer number. The AC contains the project-programmer number of the job executing the UUO. This UUO is used by KJOB.

3.6.3 Monitor Examination

3.6.3.1 PEEK AC, or CALLI AC, 33 - This UUO allows a user program to examine any location in the monitor. It is used by SYSTAT, FILDDT, and DATDMP and could be used for on-line monitor debugging. The PEEK UUO requires bit 16 (JP.SPA - examine all of core) and/or bit 17 (JP.SPM - examine the monitor) to be set in the privilege word . GTPRV.
The call is:

```
MOVEI AC, exec address          ;TAKEN MODULO SIZE OF MONITOR
PEEK AC,                        ;OR CALLI AC, 33
```

This call returns with the contents of the monitor location in AC.

3.6.3.2 SPY AC, or CALLI AC, 42 - This UUO is used for efficient examination of the monitor during timesharing. Any number of K of physical core (not limited to the size of the monitor) is

placed into the user's logical high segment. This amount cannot be saved with the monitor SAVE command (only the low segment is saved), cannot be increased or decreased by the CORE UUO (error return taken), or cannot have the user-mode write-protect bit cleared (error return taken). The call is:

```
        MOVEI AC, highest physical core location desired
        SPY AC,                          ;or CALLI AC, 42
        error return
        normal return
```

Any program that is written to use the SPY UUO should try the PEEK UUO if it receives an error return. The SPY UUO requires bit 16 (JP.SPA - examine all of core) and/or bit 17 (JP.SPM - examine the monitor) to be set in the privilege word .GTPRV.

3.6.3.3 POKE. AC, or CALLI AC, 114[1] - This UUO is used by a privileged user to alter one location in the monitor at a time. The POKE. UUO requires bit 4 (JP.POK) to be set in the privilege word .GTPRV.

The call is:

```
        MOVE AC, [3,,ADR]
        POKE. AC,                        ;or CALLI AC, 114
        error return
        normal return
ADR:    monitor location
        old value
        new value
```

The error return is given if:

The user is not privileged; AC contains 0.

The value specified in ADR+1 as the old value is not the same as the actual value contained in the monitor location; AC contains 1.

The address specified is not a valid monitor address; AC contains 2.

3.6.3.4 GETTAB AC, or CALLI AC, 41 - This UUO provides a mechanism which will not vary from monitor to monitor for user programs to examine the contents of certain monitor locations. The call is:

```
        MOVE AC, [XWD index, table number]
        GETTAB AC,                       ;or CALLI AC, 41
        error return
        normal return
```

[1] This UUO depends on FTPOKE which is normally off in the DECsystem-1040.

The left half of AC contains a job number or some other index to a table. Some job numbers may refer to high segments of programs by using arguments greater than the highest job number for the current monitor. A LH of -1 indicates the current job number. A LH of -2 references the job's high segment. An error return is given if there is no high segment or if the hardware and software are non-reentrant. The right half of AC contains a table number from the list of monitor data tables and parameters in Table 3-1. The entries in these tables are globals in the monitor subroutine COMMON. The actual values of the core addresses of these locations are subject to change and can be found in the LOADER storage map for the monitor. The complete description of these globals is found in the listing of COMMON.

The customer is allowed to add his own GETTAB tables to the monitor. A negative right half should be used to specify such customer-added tables.

An error return leaves the AC unchanged and is given if the job number or index number in the left half of AC is too high, the table number in the right half of AC is too high, or the user does not have the privilege of accessing the specified table.

A normal return supplies the contents of the requested table in AC, or a zero if the table is not defined in the current monitor.

The SYSTAT program makes frequent use of this UUO.

NOTE

Many GETTAB tables have information in the undescribed bits. This information is likely to change and should be ignored. Although the field may currently be zero, there is no reason to believe that it will always be zero.

Table 3-1
GETTAB Tables

Table Numbers (RH of AC)	Table Names	Explanation
00	.GTSTS	Job status word; index by job or segment number.
01	.GTADR	Job relocation and protection; index by job or segment number
02	.GTPPN	Project and programmer numbers; index by job or segment number
03	.GTPRG	User program name; index by job or segment number.
04	.GTTIM	Total run time used in units of jiffies; index by job number. The value of a jiffy can be obtained from bit 6 of the STATES word (item 17 in the .GTCNF table).
05	.GTKCT	Kilo-Core ticks of job; index by job number.
06	.GTPRV	Privilege bits of job; index by job number, refer to Paragraph 3.6.3.4.1.
07	.GTSWP	Swapping parameters of job; index by job or segment number.
10	.GTTTY	Terminal-to-job translation; index by job number.
11	.GTCNF	Configuration table; index by item number, refer to Paragraph 3.6.3.4.2.
12	.GTNSW	Nonswapping data; index by item number, refer to Paragraph 3.6.3.4.3.
13	.GTSDT	Swapping data; index by item number, refer to Paragraph 3.6.3.4.4.
14	.GTSGN	High segment table; index by job number. Bit 0 = 0, then bits 18-35 are index of high segment (if bits 18-35 = 0, then there is no high segment). Bit 0 = 1, then bits 18-35 are number of K to spy on. Bit 1 (SN%SHR) = 1 if job has a high segment that is sharable. Bit 5 (SN%LOK) = 1 if job has a high segment that is locked.
15	.GTODP	Once-only disk parameters; index by item number, refer to Paragraph 3.6.3.4.5.
16	.GTLDV	5-series monitor disk parameters; index by item number, refer to Paragraph 3.6.3.4.6.

Table 3-1 (Cont)
GETTAB Tables

Table Numbers (RH of AC)	Table Names	Explanation
17	.GTRCT	Disk blocks read by job; used by DSK command: a. Bits 0-11 = incremental blocks b. Bits 12-35 = total blocks since start of job. Index by job number. Job 0 indicates the number of blocks swapped in.
20	.GTWCT	Disk blocks written by job: a. Bits 0-11 = incremental blocks. b. Bits 12-35 = total blocks since start of job. Index by job number. Job 0 indicates the number of blocks swapped out.
21	.GTDBS	Reserved for future.
22	.GTTDB	Reserved for future.
23	.GTSLF	Table of GETTAB addresses (GETTAB immediate); index by GETTAB table number, refer to Paragraph 3.6.3.4.7.
24	.GTDEV	Device or file structure name of sharable high segment. Index by high segment number.
25	.GTWSN	Two-character SIXBIT names for job queues; index by item numbers, refer to Paragraph 3.6.3.4.8.
26	.GTLOC	Job's logical station; index by job number.
27	.GTCOR	Physical core allocation. One bit per one K of core if system does not include LOCK UUO. Two bits per entry if system includes LOCK UUO. A non-zero entry indicates core in use.
30	.GTCOM	Table of SIXBIT names of monitor commands.
31	.GTNM1	First half of name of user in SIXBIT; index by job number.
32	.GTNM2	Last half of name of user in SIXBIT; index by job number.
33	.GTCNO	Job's charge number; index by job number.
34	.GTTMP	Job's TMPCOR pointers; index by job number.
35	.GTWCH	Job's WATCH bits; index by job number, refer to Paragraph 3.6.3.4.9.
36	.GTSPL	Job's spooling control bits; index by job number, refer to Paragraph 3.6.3.4.10.
37	.GTRTD	Job's real-time status word; index by job number.

Table 3-1 (Cont)
GETTAB Tables

Table Numbers (RH of AC)	Table Names	Explanation
40	.GTLIM	Job's time limit in jiffies and Batch status; index by job number. a. Bits 1-9 (JB.LCR) = job's core limit. b. Bit 10 = 1 (JB.LBT) if a Batch job. c. Bit 11 = 1 (JB.LSY) if program comes from SYS. Set on R command or equivalent. Cleared on R command (or equivalent) or SETNAM UUO. d. Bits 12-35 (JB.LTM) = job's time limit.
41	.GTQQQ	Timesharing scheduler's queue headers.
42	.GTQJB	Timesharing scheduler's queue that job is in; index by job number.
43	.GTCM2	Table of SET command names.
44	.GTCRS	Status of hardware taken on a crash. 0: CR.SAP = CONI APR, 1: CR.SPI = CONI PI, 2: CR.SSW = DATAI APR The remainder of the table contains the status of the various devices.
45	.GTISC	Swapper's input scan list of queues.
46	.GTOSC	Swapper's output scan list of queues.
47	.GTSSC	Scheduler's scan list of queues.
50	.GTRSP	Response counter table. Time in jiffies when user started to wait for his job to run. This time is cleared when the job is first given to the processor by the scheduler.
51	.GTSYS	System variables which are independent of CPU. Word 0 (%SYERR) = system wide hardware error count. Word 1 (%SYCCO) = number of times COMCNT was off. Word 2 (%SYDEL) = number of error-logging disabled errors. Word 3 (%SYSPC) = LH is a 3-letter code of the last STOPCD. RH is the address +1 of the last STOPCD executed. Word 4 (%SYNDS) = number of debug STOPCDs. Word 5 (%SYNJS) = number of job STOPCDs. Word 6 (%SYNCP) = total number of commands processed by the system since it was started.
52	.GTWHY	Operator why comments in ASCIZ.
53	.GTTRQ	Total time job was in run queues whether it was running or not.

Table 3-1 (Cont)
GETTAB Tables

Table Numbers (RH of AC)	Table Names	Explanation
54	.GTSPS	Job status word of second processor. Bit 29 (SP.SC0) = SET CPU command can be used. Bit 35 (SP.CR0) = SET CPU UUO can be used. Bits for other processors can be obtained by shifting left 1 bit per processor.
55	.GTC0C	CPU0 CDB constants; index by item number, refer to Paragraph 3.6.3.4.11.
56	.GTC0V	CPU0 CDB variables; index by item number, refer to Paragraph 3.6.3.4.12.
57	.GTC1C	CPU1 CDB constants; index by item number; see .GTC0C.
60	.GTC1V	CPU1 CDB variables; index by item number; see .GTC0V.
61	.GTC2C	CPU2 CDB constants; index by item number; see .GTC0C.
62	.GTC2V	CPU2 CDB variables; index by item number; see .GTC0V.
63	.GTC3C	CPU3 CDB constants; index by item number; see .GTC0C.
64	.GTC3V	CPU3 CDB variables; index by item number; see .GTC0V.
65	.GTC4C	CPU4 CDB constants; index by item number; see .GTC0C.
66	.GTC4V	CPU4 CDB variables; index by item number; see .GTC0V.
67	.GTC5C	CPU5 CDB constants; index by item number, see .GTC0C.
70	.GTC5V	CPU5 CDB variables; index by item number; see .GTC0V.
71	.GTFET	Current setting of all features defined in F.MAC, index by item number, refer to Paragraph 3.6.3.4.14.
72	.GTEDN	Table of ersatz device names (e.g., NEW, LIB) with their corresponding project-programmer numbers. The search lists of these devices can be obtained from the PATH.UUO.

3.6.3.4.1 Entries in Table 6- .GTPRV (Privilege Table)

Each job has a one-word entry to indicate job privileges. The privilege bits are as follows:

Bit	Mnemonic	Meaning
1-2	JP.DPR	Highest disk priority for this job.
3	JP.MET	Job is allowed to execute the METER.UUO.
4	JP.POK	Job is allowed to execute the POKE.UUO.
5	JP.CCC	Job is allowed to change its CPU specification via a command or UUO.
6-9	JP.HPQ	Highest high-priority queue available to this job.
10	JP.NSP	Job is allowed to unspool devices.
13	JP.RTT	Job is allowed to execute the RTTRP UUO.
14	JP.LCK	Job is allowed to execute the LOCK UUO.
15	JP.TRP	Job is allowed to execute the TRPSET UUO.
16	JP.SPA	Job is allowed to PEEK and SPY on all of core.
17	JP.SPM	Job is allowed to PEEK and SPY on the monitor.

3.6.3.4.2 Entries in Table 11 - .GTCNF (Configuration Table)

Item	Location	Use
0	%CNFG0	Name of system in ASCIZ.
-		
4	%CNFG4	
5	%CNDT0	Date of system in ASCIZ.
6	%CNDT1	
7	%CNTAP	Name of system device (SIXBIT).
10	%CNTIM	Time of day in jiffies.
11	%CNDAT	Today's date (15-bit format).
12	%CNSIZ	Highest location in monitor +1.
13	%CNOPR	Name of OPR TTY (SIXBIT).
14	%CNDEV	LH is start of DDB (device-data-block) chain.
15	%CNSJN	LH=-# of high segments, RH=+# of jobs (counting NULL job).
16	%CNTWR	Non-zero if system has two-register hardware and software.

Item	Location	Use
17	%CNSTS	Location describing feature switches of this system in LH, and current state in RH. Assembled according to MONGEN dialog and S.MAC: Bit 0=1 if disk system (ST%DSK) Bit 1=1 if swap system (ST%SWP) Bit 2=1 if LOGIN system (ST%LOG) Bit 3=1 if full duplex software (ST%FTT) Bit 4=1 if privilege feature (ST%PRV) Bit 5=1 if assembled for choice of reentrant or non-reentrant software at monitor load time (ST%TWR) Bit 6=1 if clock is 50 cycle instead of 60 cycle (ST%CYC) Bit 7-9 type of disk system (ST%TDS): if 0, 4-series disk system. if 1, 5-series disk system. if 2, spooled disk. Bit 10=1 if independent programmer numbers between project (INDPPN is non-zero) (ST%IND) Bit 11=1 if image mode on terminal (8-bit SCNSER) (ST%IMG) Bit 12=1 if dual processor system (ST%DUL) Bit 13=1 if multiple RIBs supported (ST%MRB) Bit 14=1 if high precision time accounting (ST%HPT) Bit 15 = 1 if overhead excluded from time accounting (ST%EMO) Bit 16=1 if real-time clock (ST%RTC) Bit 17=1 if built to handle FOROTS (ST%MBF) Set by the privileged operator command, SET SCHED: Bit 27=1 means no operator (ST%NOP) Bit 28=1 means unspooling devices (ST%NSP) Bit 29=1 means assigning devices (ST%ASS) Bit 33=1 means only Batch jobs may LOGIN (except from CTY or OPR) (ST%BON) Bit 34=1 means no remote LOGINs (ST%NRL) Bit 35=1 means no more LOGINs except from CTY or OPR (ST%NLG)
20	%CNSER	Serial number of PDP-10 processor. Set by MONGEN dialog.
21	%CNNSM	Number of nanoseconds per memory cycle for memory system. If the GETTAB fails, the number of nanoseconds per memory cycle is † D1000. Used by SYSTAT to compute shuffling time.

Item	Location	Use
22	%CNPTY	PTY parameters for Batch. LH = the number of the first invisible terminal (which is one greater than the number of the CTY) RH = the number of PTY's in the system configuration.
23	%CNFRE	AOBJN word to use bit map in monitor for allocating 4-word core blocks.
24	%CNLOC	LH=0, RH=address in monitor for free 4-word core block areas. (This is never changed while monitor runs).
25	%CNSTB	Link to STB chain for remote Batch.
26	%CNOPL	Address of the line data block (LDB) of the operator's terminal.
27	%CNTTF	Pointer to TTY free chunks.
30	%CNTTC	LH=number of TTY chunks. RH=address of first TTY chunk.
31	%CNTTN	Number of free TTY chunks.
32	%CNLNS	Pointer to current TTY as seen by the command decoder.
33	%CNLNP	Pointer to examine TTY line table, including remote terminals. LH= -total number of TTY lines. RH=beginning of line table.
34	%CNVER	Version of monitor. (Stored in location 137 of monitor as a save file when monitor is not running.) Bits 0-17 reserved for customer. Bits 18-23 monitor level (e.g., 5) Bits 24-29 monitor release (e.g., 3). Bits 30-35 used for internal development. If the GETTAB fails, the monitor is a version previous to 5.03
35	%CNDSC	Pointer to data set control table. LH = -length of table. RH = beginning of control table.
36	%CNDLS	Last received interrupt from the DC10.
37	%CNCCI	Last received interrupt from the 680I.
40	%CNSGT	Last dormant segment which was deleted to free a segment number.
41	%CNPOK	Address of last location changed in monitor by the POKE.UUO.

Item	Location	Use
42	%CNPUC	LH = the number of the job which successfully executed the POKE.UUO last. RH = the number of successful POKE.UUOs executed.
43	%CNWHY	The reason for the last reload (SIXBIT unabbreviated operator answer). Refer to ONCE in the DEC-system-10 Software Notebooks.
44	%CNTIC	The number of clock ticks per second. This is the time-of-day clock. The number is obtained by conducting a simple experiment at monitor load time. A different clock can be used for incremental run time accounting (refer to %CNRTC below).
45	%CNPDB	The pointer to the process data block (PDB) pointer tables.
46	%CNRTC	The run time clock rate (jiffies per second). That is, the rate of the clock used to measure the run time of the job and the system statistics (null, lost, and overhead time). This is the precision of the measurement, not the units of measurement.
47	%CNCHN	The pointer to the list of channel (DF10) data blocks. LH = the address of the 1st channel data block. RH = unused.
50	%CNLMX	LOGMAX. The maximum number of jobs allowed to LOGIN.
51	%CNBMX	BATMAX. The maximum number of Batch jobs allowed to LOGIN.
52	%CNBMN	BATMIN. The guaranteed number of Batch jobs (i.e., the number of jobs reserved for Batch).
53	%CNDTM	The universal date-time standard (refer to Paragraph 3.6.1).
54	%CNLNM	LOGNUM. The number of jobs currently logged-in.
55	%CNBNM	BATNUM. The number of Batch jobs currently logged-in.
56	%CNYER	LOCYER. The year.
57	%CNMON	LOCMON. The month (Jan = 1, Feb = 2, etc.).
60	%CNDAY	LOCDAY. The local day of the month (1, 2, 3,...).
61	%CNHOR	LOCHOR. The local hour in 24-hr format.
62	%CNMIN	LOCMIN. Minutes(0, 1, . . . , 59).
63	%CNSEC	LOCSEC. Seconds (0, 1, . . ., 59).

Item	Location	Use
64	%CNGMT	Offset for the universal date-time standard in order to convert it to local time from Greenwich Mean Time (not yet implemented).
65	%CNDBG	Debugging status word. Bit 0=1 System debugging (ST%DBG). Bit 1=1 Reload on debug stop code (ST%RDC). Bit 2=1 Reload on job stop code (ST%RJE).
66	%CNFRU	Amount of free core currently in use by the monitor.

3.6.3.4.3 Entries in Table 12 - .GTNSW (Nonswapping Data)

With the 5.05 and later monitors, no new entries will be added to the .GTNSW table because many of the parameters in this table are dependent upon the processor used and therefore are different for each processor in a multiprocessor system. GETTAB tables 51-70 exist for new parameters as well as the .GTNSW parameters.

Item	Location	Use
0		Obsolete,
-		unspecified data.
7		
10	%NSCMX	CORMAX. Size in words of largest legal user job +1 (Low seg+high seg).
11	%NSCLS	Byte pointer to last free block.
12	%NSCTL	Total free+dormant+idle K physical core left (virtual core).
13	%NSSHW	Job number shuffler has stopped.
14	%NSHLF	Absolute address of job above lowest hole, 0 if no job.
15	%NSUPT	Time system has been up in jiffies.
16	%NSSHF	Total number of words shuffled by system.
17	%NSSTU	Number of job using SYS if not a disk.
20	%NSHJB	Highest job number currently assigned.
21	%NSCLW	Total number of words cleared by system.
22	%NSLST	Total number of clock ticks when null job ran and other jobs wanted to but could not because: a. Swapped out or on way in or out. b. Monitor waiting for I/O to stop so it can shuffle or swap. c. Job being swapped out because of expanding core.

Item	Location	Use
23	%NSMMS	Size of physical memory in words.
24	%NSTPE	Total number of user parity errors (memory) since system was loaded.
25	%NSSPE	Total number of spurious (refer to Paragraph 7.7) parity errors (memory).
26	%NSMPC	Total number of multiple parity errors (memory).
27	%NSMPA	The absolute location of the last user mode memory parity error.
30	%NSMPW	The contents of the last user mode memory parity error.
31	%NSMPP	The user PC of the last user mode memory parity error.
32	%NSEPO	Total number of PDL OVR's at UUO level in exec mode which were not recovered.
33	%NSEPR	Number of PDL OVR's at UUO level which were recovered by assigning extended list.
34	%NSNXM	Highest legal value of CORMAX.
35	%NSKTM	Count-down timer for SET KSYS UUO.
36	%NSCMN	Amount of core guaranteed to be available after locking jobs in core (CORMIN).
37	%NSABC	Count of number of address breaks handled.
40	%NSABA	Contents of data switches on last address break.
41	%NSLJR	Last job that ran if different from the current job.
42	%NSACR	Accumulated CPU response. Total number of jiffies that all users waited for their jobs to initially run after either a command was issued which ran a job (program) or terminal input was given that removed the job from a TTY input wait state.
43	%NSNCR	Number of CPU responses for all users waiting for jobs to run (refer to %NSACR above). Dividing the value of %NSACR by the value of %NSNCR gives the average response time since system startup.
44	%NSSCR	Accumulated squares of the CPU response times obtained from %NSACR.

3.6.3.4.4 Entries in Table 13 - .GTSDT (Swapping Data)

Item	Location	Use
0	%SWBGH	Number of K in biggest hole in core.
1	%SWFIN	-Job number of job being swapped out, +Job number of job being swapped in.
2	%SWFRC	Job being forced to swap out.
3	%SWFIT	Job waiting to be fit into core.
4	%SWVRT	Amount of virtual core left in system in K (initially set to number of K of swapping space).
5	%SWERC	LH=number of swap read or write errors, RH=error bits (bits 18-21 same as status bits) + number of K discarded.
6	%SWPIN	-1 if job swapped in (monitors which swap process data blocks (PDBs) only).

3.6.3.4.5 Entries in Table 15 - .GTODP (Once-Only Disk Parameters)

Item	Location	Use
0	%ODSWP	Unused, contains zero in 5-series monitors.
1	%ODK4S	K of disk words set aside for swapping on all units in active swapping list.
2	%ODPRT	In-core protect time multiplies size of job in K-1.
3	%ODPRA	In-core protect time added to above result after multiply.

3.6.3.4.6 Entries in Table 16 - .GTLVD (5-series Monitor Disk Parameters)

Item	Location	Use
0	%LDMFD	Project-programmer number for UFDs only [1,1].
1	%LDSYS	Project-programmer number for device SYS [1,4]. In 4-series monitors [1,1].
2	%LDFFA	Project-programmer number for FAILSAFE [1,2].
3	%LDHLP	Project-programmer number for SYSTAT and HELP [2,5].
4	%LDQUE	Project-programmer number for spooling programs [3,3].
5	%LDSPB	a. LH=address of first PPB block. b. RH=address of next PPB block to be scanned.

Item	Location	Use
6	%LDSTR	a. LH=address of first file structure data block. b. RH=relative address of next file structure data block, i.e., the address within the data block which points to the actual address of the next data block.
7	%LDUNI	a. LH=address of data block of first unit in system. b. RH=relative address of data block of next unit in system.
10	%LDSWP	a. LH=address of first unit for swapping in system. b. RH=relative address of next unit for swapping in system.
11	%LDCRN	Number of 4-word access blocks for disk systems allocated at ONCE - only time.
12	%LDSTP	Standard file protection code (057), can be changed by installation. In 4-series monitors (055).
13	%LDUFP	Standard UFD protection code (775), can be changed by installation. In 4-series monitors (055).
14	%LDMBN	Number of monitor buffers allocated at once-only time (2). In 4-series monitors, 1.
15	%LDQUS	SIXBIT name of file structure containing 3,3.UFD for spooling and OMOUNT queues. In 4-series monitors, DSK.
16	%LDCRP	UFD used for storing system crashes. In 4-series monitors, [10,1].
17	%LDSFD	Maximum number of nested SFD's which the monitor allows to be created.
20	%LDSPP	Protection of spooled output files (bits 0-7).
21	%LDSYP	Standard protection for files in SYS: (155) except for files with an extension of .SYS.
22	%LDSSP	Standard protection for files in SYS: with an extension of .SYS (157).
23	%LDMNU	Maximum negative argument to USETI which reads extended RIBs.
24	%LDMXT	Maximum number of blocks transferred with one I/O operation (one IOWD). Normally 100000 but can be defined at MONGEN to be smaller so that a job doing high priority disk I/O will be locked out for a shorter period of time (since it can be locked out for as long as the channel is busy).
25	%LDNEW	Project-programmer number for experimental SYS [1,5].

Item	Location	Use
26	%LDOLD	Project-programmer number for library of superseded system programs [1,3].
27	%LDUMD	Project-programmer number for user mode diagnostics [6,6].
30	%LDNDB	Default number of disk buffers in a buffer ring.

3.6.3.4.7 Entries in Table 23 - .GTSLF (GETTAB Immediate)

This table is useful for a program that uses the SPY UUO for efficiency and needs the core address of the monitor tables. Absolute location 151 in the monitor contains the address of the beginning of this table.

The format of each entry is as follows:

LH=Bits 0-8 = maximum item number in table.
Bit 9 = data may be process data.
Bit 10 = data may be segment data.
Bits 14-17 = a monitor AC.

RH=executive-mode address of table (item 0).

Examples:

```
XWD ITEM + JBTMXL, JOBSTS
XWD ITEM + TTPMXL, TTYTAB
```

3.6.3.4.8 Entries in Table 25 - .GTWSN (Two-character SIXBIT names for job queues)

Word 0

Bits 0-11 = contain the two SIXBIT character mnemonic of job state code 0.
Bits 12-23 = contain the two SIXBIT character mnemonic of job state code 1.
Bits 24-35 = contain the two SIXBIT character mnemonic of job state code 2.

Word 1

Bits 0-11 = contain the mnemonics of job state code 3.
Bits 12-23 = contain the mnemonics of job state code 4.
Bits 24-35 = contain the mnemonics of job state code 5.
etc.

The job state codes for a disk system are as follows:

RN - one of the run queues.
WS - I/O wait satisfied.
TS - TTY I/O wait satisfied.
DS - disk I/O wait satisfied.
AU - disk alter UFD wait.

MQ	-	disk monitor buffer wait.
DA	-	disk storage allocation wait.
CB	-	disk core block scan wait.
D1	-	DECtape control wait.
D2	-	second DECtape control wait.
DC	-	data control wait.
MT	-	magnetic tape control wait.
CA	-	core allocation wait (to be locked).
IO	-	I/O wait.
TI	-	TTY I/O wait.
DI	-	disk I/O wait.
SL	-	sleep wait.
NU	-	null state.
ST	-	stop (↑C) state.
JD	-	DAEMON wait.

These state codes are printed by SYSTAT. Note that SYSTAT displays other codes based on analysis, such as the following:

TO	-	TTY output.
↑C	-	job stopped.
↑W	-	command wait.
OW	-	operator wait.
HB	-	hibernate.

3.6.3.4.9 Entries in Table 35 - .GTWCH (WATCH Table)

Each job has a one-word entry to indicate the WATCH bits. The bits for each word are as follows:

Bit	Mnemonic	Meaning
1	JW.WDY	Watch time of day.
2	JW.WRN	Watch run time.
3	JW.WWT	Watch wait time.
4	JW.WDR	Watch disk reads.
5	JW.WDW	Watch disk writes.
6	JW.WVR	Watch versions.

3.6.3.4.10 Entries in Table 36 - .GTSPL (Spooling Table)

Each job has a one-word entry to indicate the spooling control bits. These bits are as follows:

Bit	Mnemonic	Meaning
35	JS.PLP	Line printer spooling.
34	JS.PPL	Plotter spooling.
33	JS.PPT	Paper tape punch spooling.
32	JS.PCP	Card punch spooling.
31	JS.PCR	Card reader spooling.
24-26	JS.PRI	Disk priority.

3.6.3.4.11 Entries in Table 55 - .GTC0C (CPU0 CDB constants table)

The items in this table correspond to the items in the constants table for each processor.

CPU1	Table 57 - .GTC1C
CPU2	Table 61 - .GTC2C
CPU3	Table 63 - .GTC3C
CPU4	Table 65 - .GTC4C
CPU5	Table 67 - .GTC5C

Item	Location	Use
0	%CCPTR	LH=pointer to next CDB, or 0 if this is the last CDB. RH=unused.
1	%CCSER	APR serial number.
2	%CCOKP	If less than or equal to zero, CPU is running ok. If greater than zero, CPU has stopped running correctly. Contents of word is the number of jiffies CPU has been stopped.
3	%CCTOS	Trap offset for KA10 interrupt locations (0 or 100).
4	%CCLOG	Logical CPU name in SIXBIT (CPUn).
5	%CCPHY	Physical CPU name in SIXBIT (CPAn, CPIn, or CP6n).
6	%CCTYP	Type of processor (LH for customers, RH for DEC) 1 (.CC166) = PDP-6 2 (.CCKAX) = KA10 3 (.CCKIX) = KI10
7	%CCMPT	Relative GETTAB pointer to memory parity bad address subtable. Refer to Paragraph 3.6.3.4.13. Bits 0-8 maximum relative entry in subtable Bits 18-35 relative address of first word in subtable in CPU variable GETTAB (.GTC0V). If word is 0, the subtable has been conditionally assembled out of the monitor.
10	%CCRTC	Real time clock (DK10) DDB. If word is 0, there is no real time clock on this CPU.
11	%CCRTD	Real time clock DDB if high precision time accounting. If 0, there is no high precision time accounting on this CPU.

Item	Location	Use
12	%CCPAR	Relative GETTAB pointer to memory parity subtable. Refer to Paragraph 3.6.3.4.13. Bits 0-8 maximum relative entry in subtable. Bits 18-35 relative address of first word in subtable in CPU variable GETTAB. (.GTC0V). If word is 0, the subtable has been conditionally assembled out of the monitor.
13	%CCRSP	Relative GETTAB pointer to response subtable. Refer to Paragraph 3.6.3.4.13. Bits 0-8 Maximum relative entry in subtable. Bits 18-35 relative address of first word in subtable in CPU variable GETTAB (.GTC0V).

3.6.3.4.12 Entries in Table 56 - .GTC0V (CPU0 CDB Variable Table)

The items in this table correspond to the items in the variables table for each processor.

CPU1 Table 60 - .GTC1V
CPU2 Table 62 - .GTC2V
CPU3 Table 64 - .GTC3V
CPU4 Table 66 - .GTC4V
CPU5 Table 70 - .GTC5V

Item	Location	Use
5	%CVUPT	Uptime in jiffies for this CPU.
12	%CVLST	Lost time in jiffies for this CPU.
14	%CVTPE	Total memory parity error words detected during all CPU sweeps on this CPU while processor was in exec or user mode. If the system halts, this location has already been updated.
15	%CVSPE	Total spurious memory parity errors detected on this CPU (i.e., errors which did not reoccur when the CPU swept through core). Can occur on a read-pause-write which rewrites memory or on a channel-detected parity not found on the sweep (refer to %CVPCS in parity subtable.)
16	%CVMPC	Multiple memory parity errors for this CPU. That is, the number of time the operator pushed CONTINUE after a serious memory parity halt. LH = 1 if serious error on this bad parity (must halt). LH is cleared on CONTINUE or STARTUP.

Item	Location	Use
17	%CVMPA	Memory parity address for this CPU. That is, first bad physical memory address found when the monitor swept through core after processor or channel detected first parity error.
20	%CVMPW	Memory parity word for this CPU. That is, contents of first bad word found by monitor when it swept through core after the processor or channel detected first bad parity.
21	%CVMPP	Memory parity PC for this CPU. That is, PC of last memory parity (not counting sweep through core).
27	%CVABC	Address break count on this CPU.
30	%CVABA	Address break address on this CPU.
31	%CVLJR	Last job run on this CPU including the null job.
32-34		Obsolete. Refer to items 20-23 in the Response Subtable.
35	%CVSTS	Stop timesharing on this CPU. Contains job number which performed the TRPSET UUO.
36	%CVRUN	Operator-controlled scheduling for this CPU (OPSER: SET RUN command). Bit 0 (CV%RUN) = 1 do not run jobs on this CPU.
37	%CVNUL	Null time in jiffies for this CPU.
40	%CVEDI	LH = exec PC so that offending instruction can be corrected. RH = number of exec "don't care" interrupts (i.e., user enabled APR interrupts which monitor causes (AOV, FOV).
41	%CVJOB	Current job running on this CPU (0 is null job).
42	%CVOHT	Overhead time in jiffies for this CPU. Includes clock queue processing, short command processing, swapping and scheduling decisions, and software context switching. Does not include UUO execution or I/O interrupt time, since these times are not overhead.
43	%CVEVM	(KI10 only) Maximum amount of exec virtual address space to be used for mapping user segments on a LOCK UUO.
44	%CVEVU	(KI10 only) Current amount of exec virtual address space being used for mapping user segments on a LOCK UUO.
45	%CVLLC	On a dual processor system, the count of the number of times a CPU has looped in the CPU interlock while waiting for it to be relinquished by the second CPU.

Item	Location	Use
46	%CVTUC	Total number of UUOs executed on this CPU from exec and user mode.
47	%CVTJC	Total number of job context switches from one job to a different job, including the null job, on this CPU.

3.6.3.4.13 GETTAB Subtables - Via the GETTAB mechanism, GETTAB subtables make monitor-collected data available to user programs and, at the same time, allow the installation to decide if it wants to use more monitor table space without invalidating any user programs. These subtables are included in all systems except the DECsystem-1040. However, they may be excluded by changing the appropriate conditional assembly switches with MONGEN. It is anticipated that only installations that need the core space for other uses will decide to exclude the subtables.

To reference a subtable, the user program first does a GETTAB UUO to obtain the pointer to the subtable (refer to Paragraph 3.6.3.4.11). Then the program does a second GETTAB to get the desired item in the subtable. If the pointer is zero, the desired subtable is not included in the system.

The following example illustrates the method for obtaining the accumulated response times for CPU N for all users that waited for their jobs to initially run after TTY input was given.

```
%CCRSP==XWD 13,55        ;WORD AND TABLE NUMBER FOR RESPONSE SUBTABLE
%CVRAI==3                ;SUBTABLE INDEX FOR ACCUMULATED TTY INPUT UUO
                         ;   RESPONSE.
.GTC0V==56               ;GETTAB TABLE FOR CPU0 VARIABLES
    MOVEI T1,N           ;CPU NUMBER (0,1,...,5)
    LSH T1,N             ;CONSTANTS TABLE GETTAB INDEX MOVES UP BY TWOS.
    MOVE T2,[%CCRSP]     ;RELATIVE GETTAB POINTER WORD FOR RESPONSE
                         ;   SUBTABLE FOR CPU0.
    ADD T2,T1            ;FORM GETTAB ARGUMENT FOR CPU N.
    GETTAB T2,           ;GET RELATIVE POINTER TO RESPONSE SUBTABLE.
      JRST NONE          ;NOT THERE (MONITOR IS ONE BEFORE 5.05)
    JUMPE T2,NONE        ;IF 0, SUBTABLE NOT INCLUDED IN THIS
                         ;   LOAD OF THE MONITOR.
    ADDI T2,%CVRAI       ;FORM DESIRED INDEX IN SUBTABLE WITH
                         ;   RESPECT TO VARIABLE GETTAB.
    HRL T2,T2            ;RELATIVE ADDRESS OF SUBTABLE WITH
                         ;   RESPECT TO VARIABLE TABLE.
    HRRI T2,.GTC0V(T1)   ;FORM PROPER GETTAB FOR CPU VARIABLES.
    GETTAB T2,           ;GET RESPONSE TIME.
      JRST NONE          ;NOT THERE. THIS SHOULD NOT HAPPEN SINCE
                         ;   ZERO TEST ON RELATIVE POINTER FAILED.
    HERE WITH RESPONSE IN T2
```

Response Subtable

The response subtable is pointed to by %CCRSP in the constants table for each processor. This subtable is under the conditional assembly switch FTRSP. Refer to Paragraph 3.6.3.4.3 for additional response information.

Item	Location	Use
0	%CVRSO	Accumulated TTY output UUO responses. That is, the total number of jiffies users have spent waiting for their jobs to do a TTY output UUO (on CPU0) after either a command was issued which ran a job or terminal input was given that removed the job from a TTY input wait state.
1	%CVRNO	Number of TTY output UUO responses for this CPU.
2	%CVRHO	The high-order sum of the squares of TTY output UUO responses. Used for computing standard deviation.
3	%CVRLO	The low-order sum of the squares of TTY output UUO responses.
4	%CVRSI	Accumulated TTY input UUO responses for this CPU. That is, the total number of jiffies users have spent waiting for their jobs to do a TTY input UUO (on CPU0) after either a command was issued which ran a job or terminal input was given that removed the job from a TTY input wait state.
5	%CVRNI	Number of TTY input UUO responses for this CPU.
6	%CVRHI	The high-order sum of the squares of TTY input UUO responses. Used for computing standard deviation.
7	%CVRLI	The low-order sum of the squares of TTY input UUO responses.
10	%CVRSR	Accumulated CPU quantum requeue responses. That is, total number of jiffies users spent waiting for their jobs to exceed the CPU quantum on this CPU after either a command was issued which ran a job or terminal input was given that removed the job from a TTY input wait state.
11	%CVRNR	Number of CPU quantum requeue responses for this CPU.
12	%CVRHR	The high-order sum of the squares of CPU quantum requeue response. Used for computing standard deviation.
13	%CVRLR	The low-order sum of the squares of CPU quantum requeue response.
14	%CVRSX	Accumulated response terminated by the first occurrence of one of the above 3 events (TTY output, TTY input, or CPU quantum requeue).
15	%CVRNX	Number of such responses in %CVRSX.
16	%CVRHX	The high-order sum of the squares of responses in %CVRSX. Used for computing standard deviation.

Item	Location	Use
17	%CVRLX	The low-order sum of the squares of responses in %CVRSX.
20	%CVRSC	Accumulated CPU responses on this CPU. Total number of jiffies that users waited for their jobs to run after either a command was issued which ran a job or terminal input was given that removed the job from a TTY input state.
21	%CVRNC	Number of CPU responses for all users waiting for their jobs to run. Dividing this value into the value of %CVRSC gives the average response time since the system was started.
22	%CVRHC	The high-order sum of the squares of CPU responses on this CPU.
23	% CVRLC	The low-order sum of the squares of CPU responses on this CPU.

Parity Subtable

The parity table is pointed to by %CCPAR in the constants table for each processor. This subtable is under the conditional assembly switch FTMEMPAR. Refer to Paragraphs 3.6.3.4.3 and 7.7 for additional parity information.

Item	Location	Use
0	%CVPLA	Highest bad memory parity address on last sweep of memory. Used to tell operator the range of bad addresses.
1	%CVPMR	Relative address (not virtual address) in the high or low segment of the last memory parity error.
2	%CVPTS	Number of parity errors on the last sweep of core. Set to 0 at beginning of the sweep.
3	%CVPSC	Number of parity sweeps by the monitor.
4	%CVPUE	Number of user-enabled parity errors. Refer to Paragraph 3.1.3.1.
5	%CVPAA	The AND of bad addresses on the last memory parity sweep.
6	%CVPAC	The AND of bad contents on the last memory parity sweep.
7	%CVPOA	The OR of bad addresses on the last memory parity sweep.
10	%CVPOC	The OR of bad contents on the last memory parity sweep.
11	%CVPCS	Number of spurious parity errors. (The APR sweep found no bad parity but the channel had requested the sweep rather than the processor). This indicates a channel memory port problem.

Bad Address Subtable

The bad address table is pointed to by %CCMPT in the constants table for each processor. This subtable is under the conditional assembly switch FTMEMPAR and contains the bad addresses on the last memory parity sweep. It is not cleared and the number of valid entries is kept in %CVPTS in the parity subtable.

3.6.3.4.14 Entries in Table 71 - .GTFET (Feature Table)

This table provides the user with a mechanism for determining the current settings of all features defined in F.MAC

Item	Location	Use
0	%FTUUO	UUOs Bit 26 = 1 if control-C intercept (F%CCIN). Bit 27 = 1 if JOBSTS and CTLJOB UUOs are implemented (F%PTYU). Bit 28 = 1 if PEEK UUO implemented (F%PEEK). Bit 29 = 1 if POKE. UUO implemented (F%POKE). Bit 30 = 1 if JOB continue (F%JCON). Bit 31 = 1 if spooling supported (F%SPL). Bit 32 = 1 if job privileges supported (F%PRV). Bit 33 = 1 if DAEMON supported (F%DAEM). Bit 34 = 1 if GETTAB exists (F%GETT). Bit 35 = 1 if 2-register relocation (F%2REL).
1	%FTRTS	Real time and scheduling features Bit 27 = 1 if swapper (F%SWAP). Bit 28 = 1 if shuffler (F%SHFL). Bit 29 = 1 if DK10 service (F%RTC). Bit 30 = 1 if LOCK UUO implemented (F%LOCK). Bit 31 = 1 if TRPSET UUO implemented (F%TRPS). Bit 32 = 1 if real-time traps implemented (F%RTTR). Bit 33 = 1 if SLEEP UUO implemented (F%SLEE). Bit 34 = 1 if HIBER and WAKE UUOs supported (F%HIBW). Bit 35 = 1 if high priority queues supported (F%HPQ)
2	%FTCOM	Commands Bit 23 = 1 if COMPIL commands (F%CCL). Bit 24 = 1 if COMPIL-class (F%CCLX). Bit 25 = 1 if QUEUE (F%QCOM). Bit 26 = 1 if SET UUO and command (F%SET). Bit 27 = 1 if VERSION (F%VERS). Bit 28 = 1 if Batch control file commands (F%BCOM). Bit 29 = 1 if SET DAYTIME and SET DATE (F%SEDA). Bit 30 = 1 if WATCH (F%WATC).

Item	Location	Use
		Bit 31 = 1 if FINISH and CLOSE (F%FINI). Bit 32 = 1 if REASSIGN (F%REAS). Bit 33 = 1 if E and D (F%EXAM). Bit 34 = 1 if SEND (F%TALK). Bit 35 = 1 if ATTACH (F%ATTA).
3	%FTACC	Accounting information Bit 31 = 1 if time and core limits (F%TLIM). Bit 32 = 1 if charge number (F%CNO). Bit 33 = 1 if user name (F%UNAM). Bit 34 = 1 if kilo-core-ticks accumulation (F%KCT). Bit 35 = 1 if run-time computation (F%TIME).
4	%FTERR	Error control and internal options Bit 26 = 1 if swapping process data block (F%PDBS). Bit 27 = 1 if KI10 features at startup time (F%KI10). Bit 28 = 1 if METER. UUO supported (F%METR). Bit 29 = 1 if execute-only files (F%EXON). Bit 30 = 1 if illegal instruction message checks for for KI10 instructions (F%KII). Bit 31 = 1 if code to load BOOTS from disk (F%BOOT). Bit 32 = 1 if more than one swapping device (F%2SWP). Bit 33 = 1 if DAEMON error logging (F%EL). Bit 34 = 1 if multi-processor code loaded (F%MS). Bit 35 = 1 if memory parity error recovery (F%MEMP).
5	%FTDEB	Debugging features Bit 28 = 1 if response time measurement (F%RSP). Bit 29 = 1 if why reload code (F%WHY). Bit 30 = 1 if patch space left in tables (F%PATT). Bit 31 = 1 if back-tracking information left in COMMON (F%TRAC). Bit 32 = 1 if monitor halts on error (F%HALT). Bit 33 = 1 if redundancy checking for internal errors (F%RCHK). Bit 34 = 1 if monitor write protected (F%MONP). Bit 35 = 1 if monitor check summing (F%CHEC).
6	%FTSTR	File structure parameters Bit 21 = 1 if NUL device (F%NUL). Bit 22 = 1 if LIB/SYS/NEW (F%LIB). Bit 23 = 1 if disk priority transfers (F%DPRI). Bit 24 = 1 if append to last block (F%APLB). Bit 25 = 1 if append implies read (F%AIR). Bit 26 = 1 if generic device search (F%GSRC). Bit 27 = 1 if rename across directories (F%DRDR). Bit 28 = 1 if SEEK UUO (F%DSEK). Bit 29 = 1 if super USETI/USETO (F%DSUP). Bit 30 = 1 if disk quotas (F%DQTA).

Item	Location	Use
		Bit 31 = 1 if multiple file structures (F%STR). Bit 32 = 1 if 5-series UUOs (F%5UUO). Bit 33 = 1 if physical-only I/O (F%PHYO). Bit 34 = 1 if sub-file directories (F%SFD). Bit 35 = 1 if STRUUO functions (F%MOUN).
7	%FTDSK	Internal disk parameters Bit 21 = 1 if DEBUG CB interlock (F%CBDB) Bit 22 = 1 if LOGIN system (F%LOGI). Bit 23 = 1 if disk system (F%DISK). Bit 24 = 1 if race-condition prevention in FILFND (F%FFRE). Bit 25 = 1 if swap read error recovery (F%SWPE). Bit 26 = 1 if bad block marking (F%DBBK). Bit 27 = 1 if UFD compressor (F%DUFC). Bit 28 = 1 if disk error simulation (F%DETS). Bit 29 = 1 if extended RIBs supported (F%DMRB). Bit 30 = 1 if smaller allocation for disk core blocks (F%DSMC). Bit 31 = 1 if allocation optimization (F%DALC). Bit 32 = 1 if disk usage statistics (F%DSTT). Bit 33 = 1 if hung disk recovery (F%DHNG). Bit 34 = 1 if disk off-line recovery (F%DBAD). Bit 35 = 1 if latency optimization (F%DOPT).
10	%FTSCN	Scanner options Bit 27 = 1 if TTY BLANK command (F%TBLK). Bit 28 = 1 if page and display knowledge (F%TPAG). Bit 29 = 1 if automatic dialer supported (F%DTAL). Bit 30 = 1 if special line control (F%SCLC). Bit 31 = 1 if hardware (DC10 or DC68) scanner (F%SCNR). Bit 32 = 1 if modem control (F%MODM). Bit 33 = 1 if single scanner 630 (F%630H). Bit 34 = 1 if U.K. modem supported (F%GPO2). Bit 35 = 1 if real half-duplex terminals (F%HDPX).
11	%FTPER	I/O Parameters Bit 27 = 1 if CDP trouble intercept (F%CPTR). Bit 28 = 1 if CDR trouble intercept (F%CRTR). Bit 29 = 1 if CTY1 supported (F%CTY1). Bit 30 = 1 if remote station supported (F%REM). Bit 31 = 1 if LPT error recovery (F%LPTR). Bit 32 = 1 if device errors to operator (F%OPRE). Bit 33 = 1 if CDR super-image mode (F%CDRS). Bit 34 = 1 if MTA density and buffer size (F%MTSE). Bit 35 = 1 if TMPCOR area (F%TMP).

3.6.4 Configuration Information

3.6.4.1 SWITCH AC, or CALLI AC, 20 - This UUO returns the contents of the central processor data switches in AC. Caution must be exercised in using the data switches because they are not an allocated resource and are always available to all users.

3.6.4.2 LIGHTS AC, or CALLI AC, -1 - This UUO displays the contents of AC in the console lights.

3.7 DAEMON AC, OR CALLI AC, 102[1]

This UUO requests the DAEMON program to perform a specified function for the user program. The call is:

```
        MOVE AC, [XWD length (n+1), ADR]
        DAEMON AC,                          ;or CALLI AC, 102
        error return
        normal return
ADR:    function
        arg1
        arg2
        .
        .
        .
        arg_n
```

The length of the argument list can be zero if the number of arguments is fixed. The first word of the argument list is the code for the requested function. Non-privileged functions of the DAEMON UUO are presented in the following paragraphs. Refer to the Specifications section of the DECsystem-10 Software Notebooks for a description of the privileged functions.

3.7.1 .DCORE Function

This function causes DAEMON to write a dump file of the job's core area. The call is:

```
ADR:  1                             ;.DCORE function
      SIXBIT/dev/
      SIXBIT/file/
      SIXBIT/ext/
      0
      XWD ppn
      SIXBIT/SFD1/
            .
            .
      SIXBIT/SFDN/
```

[1] This UUO depends on FTDAEM which is normally off in the DECsystem-1040.

If an argument is omitted, the default is the same as in the DCORE command (refer to DECsystem-10 Operating System Commands).

3.7.2 .CLOCK Function

This function causes DAEMON to enter a request in the clock queue in order to wake the job after the specified number of seconds has elapsed. The UUO returns as soon as the request is entered. The HIBER UUO with no clock request (refer to Paragraph 3.1.4.2) should then be used to place the job in the sleep queue.

The call is:

```
        MOVEI AC, BLOCK
        DAEMON AC,
          JRST ERROR
        SETZ AC,
        HIBER AC,
          JRST ERROR

ERROR: ...                      ;simulate the DAEMON UUO
                                ; with the SLEEP UUO.
BLOCK: 2                        ;.CLOCK function
       +seconds                 ;number of seconds to sleep.
```

If the job already has a request in the clock queue, the new request supersedes the current request. Thus, jobs desiring to be awakened several times should issue one request for the soonest wake time.

There is no maximum on the amount of time a job can sleep and therefore, this UUO is useful when a sleep time of more than 63 seconds is desired (the SLEEP and HIBER UUOs have an implied maximum of 63 seconds). A request specifying 0 seconds clears the job's entry in the clock queue and immediately wakes the job. Note that the resolution of the timer may be several seconds slow if the system is heavily loaded.

3.7.3 Returns

The error return is given if the UUO is not implemented, DAEMON is not running, or DAEMON cannot complete the requested function. If the UUO is not implemented or DAEMON is not running, AC remains unchanged. If DAEMON cannot complete the request, AC contains one of the following error codes:

1	DMILF%	Illegal function.
2	DMACK%	Address check. The argument block is outside of user core or in the job data area.
3	DMWNA%	Wrong number of arguments.
4	DMSNH%	Impossible UUO failure (should never happen).

(Cont on next page)

5	DMCWF%	Cannot write file. An OPEN or INIT failed.
6	DMNPV%	No privileges. An attempt was made to write in the accounting files without having the proper privileges.
7	DMFFB%	FACT format is bad.
10	DMPTH%	Invalid path specification.

The normal return is taken if the requested function is successfully completed.

3.8 REAL-TIME PROGRAMMING

3.8.1 RTTRP AC, or CALLI AC, 57[1]

The real-time trapping UUO is used by timesharing users to dynamically connect real-time devices to the priority interrupt system, to respond to these devices at interrupt level, to remove the devices from the interrupt system, and to change the PI level on which the devices are associated. The RTTRP UUO can be called from UUO level or from interrupt level. This is a privileged UUO that requires the job to have real-time privileges (granted by LOGIN) and to be locked in core (accomplished by LOCK UUO). These real-time privileges are assigned by the system manager and obtained by the monitor from ACCT. SYS. The privilege bits required are:

1) JP.LCK (Bit 14) - allows the job to be locked in core.
2) JP.RTT (Bit 13) - allows the RTTRP UUO to be executed.

WARNING

> Improper use of features of the RTTRP UUO can cause the system to fail in a number of ways. Since design goals of this UUO were to give the user as much flexibility as possible, some system integrity had to be sacrificed. The most common errors are protected against since user programs run in user mode with all ACs saved. It is recommended that debugging of real-time programs not be done when system integrity is important. However, once these programs are debugged, they can run simultaneously with batch and timesharing programs.

Real-time jobs control devices one of two ways: block mode or single mode. In block mode, an entire block of data is read before the user's interrupt program is run. In single mode, the user's interrupt program is run every time the device interrupts. Furthermore, there are two types of block mode: fast block mode and normal block mode. These differ in response times. The response time to read a block of data in fast block mode is 6.5 μs per word and in normal block mode, 14.6 μs per word. (This is the CPU time to complete each data transfer.) In all modes, the response time measured from the receipt of the real-time device interrupt to the start of the user control program is 100 μs.

---

[1] This UUO depends on FTRTTRP which is normally off in the DECsystem-1040.

The RTTRP UUO allows a real-time job to either put a BLKI or BLKO instruction directly on a PI level (block mode) or add a device to the front of the monitor PI channel CONSO skip chain (single mode). Since the BLKI and BLKO are executed in exec mode, a KI10-based system requires that the job be mapped in exec virtual memory, in addition to being locked (refer to the LOCK UUO). When an interrupt occurs from the real-time device in single mode or at the end of a block of data in block mode, the monitor saves the current state of the machine, sets the new user virtual memory and APR flags, and traps to the user's interrupt routine. The user services his device and then returns control to the monitor to restore the previous state of the machine and to dismiss the interrupt.

In fast block mode the monitor places the BLKI/BLKO instruction directly in the PI trap location followed by a JSR to the context switcher. This action requires that the PI channel be dedicated to the real-time job during any transfers. In normal block mode the monitor places the BLKI/BLKO instruction directly after the real-time device's CONSO instruction in the CONSO skip chain (refer to Chapter 7).

Any number of real-time devices using either single mode or normal block mode can be placed on any available PI channel. The average extra overhead for each real-time device on the same channel is 5.5 μs per interrupt.

The call is:

```
        MOVEI AC, RTBLK         ;AC contains address of data block.
        RTTRP AC,               ;or CALLI AC, 57, put device on PI level.
        error return            ;AC contains an error code.
        normal return           ;PI is set up properly.
```

The data block depends on the mode used. In single mode the data block is:

```
RTBLK:  XWD PICHL, TRPADR       ;PI channel (1-6) and trap address.
        EXP APRTRP              ;APR trap address.
        CONSO DEV, BITS         ;CONSO chain instruction.
        0                       ;no BLKI/BLKO instruction.
```

The data block in fast block mode is:

```
RTBLK:  XWD PICHL, TRPADR       ;PI and trap address when BLKO done.
        EXP APRTRP              ;APR trap address.
        BLKO DEV, BLKADR        ;BLKI or BLKO instruction.
        0                       ;BLKADR points to the IOWD of
                                ;block to be sent.
```

The data block in normal block mode is:

```
RTBLK:  XWD PICHL, TRPADR       ;channel and trap address.
        EXP APRTRP              ;APR trap address.
        CONSO DEV, @BITMSK      ;control bit mask from user area.
        BLKI DEV, BLKADR        ;BLKI instruction.
```

On multiprocessor systems, the real-time trap UUO applies only to the processor specified by the job's CPU specification (refer to the SET CPU command or the SET UUO). If the specification indicates more than one processor, the specification is changed to indicate CPU0. Note that the PI channel (PICHL) and processor traps (APRTRP) are only for the indicated CPU.

3.8.1.1 Data Block Mnemonics - The following mnemonics are used in describing the data block associated with the RTTRP UUO.

PICHL - PICHL is the PI level on which the device is to be placed. Levels 1-6 are legal depending on the system configuration. If PICHL = 0, the device is removed from all levels. When a device is placed on a PI level, normally all other occurrences of the device on any PI level are removed. If the user desires the same device on more than one PI level simultaneously (i.e., a data level and an error level), he can issue the RTTRP UUO with PICHL negative. This indicates to the system that any other occurrence of this device (on any PI level) is not to be removed. Note that this addition to a PI level counts as a real-time device, occupying one of the possible real-time device slots.

TRPADR - TRPADR is the location trapped to by the real-time interrupt (JRST TRPADR). Before the trap occurs, all ACs are saved by the monitor and can be overwritten without concern for their contents.

APRTRP - APRTRP is the trap location for all APR traps. When an APR trap occurs, the monitor simulates a JSR APRTRP. The user gains control from an APR trap on the same PI level that his real-time device is on. The monitor always traps to the user program on illegal memory references, non-existent memory references, and push-down overflows. This allows the user to properly turn off his real-time device if needed. The monitor also traps on the conditions specified by the APRENB UUO (see Paragraph 3.1.3.1). No APR errors are detected if the interrupt routine is on a PI level higher than or equal to the APR interrupt level.

DEV - DEV is a real-time device code.

BITS - BITS is the bit mask of all interrupt bits of the real-time device and must not contain any other bits. If the user desires control of this bit mask from his user area, he may specify one level of indirection in the CONSO instruction (no indexing), i.e., CONSO DEV, @ MASK where MASK is the location in the user area of the bit mask. MASK must not have any bits set in the indirect or index fields.

BLKADR - BLKADR is the address in the user's area of the BLKI/BLKO pointer word. Before returning to the user, the monitor adds the proper relocation factor to the right half of the pointer word. Data can only be read into the low segment above the protected job data area, i.e., above location 114.

Since the pointer word is in the user's area, the user can set up a new pointer word when the word count goes to 0 at interrupt level. This allows fast switching between buffers. When the user desires to set up his own pointer word, the right half of the word must be set as an exec virtual instead of a user virtual address. The job's relocation value is returned from both the LOCK UUO and the first RTTRP UUO executed for setting the BLKI/BLKO instruction. If this pointer word does not contain a legal address, a portion of the system might be overwritten. A check should be made to determine if the negative word count in the left half of the pointer word is too large. If the word count extends beyond the user's own area, the device may cause a non-existent memory interrupt, or may overwrite a timesharing job. If all of the above precautions are taken, this method of setting up the pointer word is much faster and more flexible than issuing the RTTRP UUO at interrupt level.

3.8.1.2 Interrupt Level Use of RTTRP - The format of the RTTRP UUO at interrupt level is similar to the format at user level except for two restrictions:

1) AC 16 and AC 17 cannot be used in the UUO call (i.e., CALLI 16, 57 is illegal at interrupt level).
2) All ACs are overwritten when the UUO is executed at interrupt level. Therefore, the user must save any desired ACs before issuing the RTTRP UUO. This restriction is used to save time at interrupt level.

CAUTION

If an interrupt level routine executes a RTTRP UUO that affects the device currently being serviced, no additional UUOs of any kind (including RTTRP) can be executed during the remainder of the interrupt. At this point, any subsequent UUO dismisses the interrupt.

3.8.1.3 RTTRP Returns - On a normal return, the job is given user IOT privileges. These privileges allow the user to execute all restricted instructions including the necessary I/O instructions to control his device.

The IOT privilege must be used with caution because improper use of the I/O instructions could halt the system (i.e., HALT on the KA10; CONO APR, 0; DATAO APR, 0; CONO PI, 0 on the KA10 and KI10; and CONO PAG, 0 or DATAO PAG, 0 on the KI10). Note that a user can obtain just the user IOT privilege by issuing the RTTRP UUO with PICHL = 0.

An error return is not given to the user until RTTRP scans the entire data block to find as many errors as possible. On return, AC may contain the following error codes.

Name	Code	Value	Meaning
RTJNP%	Bit 24=1	4000	Job not privileged.
RTCPU%	Bit 25=1	2000	Not permitted on CPU1. (This is a temporary error condition reflecting the fact that the initial release of the 5.05 monitor will not support the RTTRP UUO on CPU1).
RTDIU%	Bit 26=1	1000	Device already in use by another job.
RTIAU%	Bit 27=1	400	Illegal AC used during RTTRP UUO at interrupt level.
RTJNL%	Bit 28=1	200	Job not locked in core.
RTSLE%	Bit 29=1	100	System limit for real-time devices exceeded.
RTILF%	Bit 30=1	40	Illegal format of CONSO, BLKO, or BLKI instruction.
RTPWI%	Bit 31=1	20	BLKADR or pointer word illegal.
RTEAB%	Bit 32=1	10	Error address out of bounds.
RTTAB%	Bit 33=1	4	Trap address out of bounds.
RTPNB%	Bit 34=1	2	PI channel not currently available for BLKI/BLKO's.
RTPNA%	Bit 35=1	1	PI channel not available (restricted use by system).

3.8.1.4 Restrictions -

1) Devices may be chained onto any PI channel that is not used for BLKI/BLKO instructions by the system or by other real-time users using fast block mode. This includes the APR channel. Normally PI levels 1 and 2 are reserved by the system for magnetic tapes and DECtapes. PI level 7 is always reserved for the system.

2) Each device must be chained onto a PI level before the user program issues the CONO DEV, PIA to set the device onto the interrupt level. Failure to observe this rule or failure to set the device on the same PI level that was specified in the RTTRP UUO could hang the system.

3) If the CONSO bit mask is set up and one of the corresponding flags in a device is on, but the device has not been physically put on its proper PI level, a trap may occur to the user's interrupt service routine. This occurs because there is a CONSO skip chain for each PI level, and if another device interrupts whose CONSO instruction is further down the chain than that of the real-time device, the CONSO associated with the real-time device is executed. If one of the hardware device flags is set and the corresponding bit in the CONSO bit mask is set, the CONSO skips and a trap occurs to the user program even though the real-time device was not causing the interrupt on that channel. To avoid this situation the user can keep the CONSO bit mask in his user area (refer to Paragraph 3.8.1.1). This procedure allows the user to chain a device onto the interrupt level, keeping the CONSO bit mask zero until the device is actually put on the proper PI level with a CONO instruction. This situation never arises if the device flags are turned off until the CONO DEV, PIA can be executed.

4) The user should guard against putting programs on high priority interrupt levels which execute for long periods of time. These programs could cause real-time programs at lower levels to lose data.

5) The user program must not change any locations in the protected job data area (locations 20-114), because the user is running at interrupt level and full context switching is not performed.

6) If the user is using the BLKI/BLKO feature, he must restore the BLKI/BLKO pointer word before dismissing any end-of-block interrupts. This is accomplished with another RTTRP UUO or by directly modifying the absolute pointer word supplied by the first RTTRP UUO. Failure to reset the pointer word could cause the device to overwrite all of memory.

3.8.1.5 Removing Devices from a PI Channel - When PICHL=0 in the data block (see Paragraph 3.8.1.1), the device specified in the CONSO instruction is removed from the interrupt system. If the user removes a device from a PI chain, he must also remove the device from the PI level (CONO DEV, 0).

A RESET, EXIT, or RUN UUO from the timesharing levels removes all devices from the interrupt levels (see Paragraph 3.2.2.4). These UUOs cause a CONO DEV, 0 to be executed before the device is removed. Monitor commands that issue implicit RESETS also remove real-time devices (e.g., R, RUN, GET, CORE 0, SAVE, SSAVE).

3.8.1.6 Dismissing the Interrupt - The user program must always dismiss the interrupt in order to allow monitor to properly restore the state of the machine. The interrupt may be dismissed with any UUO other than the RTTRP UUO or, on the KA10, any instruction that traps to absolute location 60. The standard method of dismissing the interrupt is with a UJEN instruction (op code 100). This instruction gives the fastest possible dismissal.

3.8.1.7 Examples

```
********** EXAMPLE 1 **********
          SINGLE MODE

TITLE RTSNGL - PAPER TAPE READ TEST USING CONSO CHAIN

PIOFF=400                ;TURN PI SYSTEM OFF
PION=200                 ;TURN PI SYSTEM ON
TAPE=400                 ;NO MORE TAPE IN READER IF TAPE=0
BUSY=20                  ;DEVICE IS BUSY READING
DONE=10                  ;A CHARACTER HAS BEEN READ

PDATA:  Z                ;LOCATION WHERE DATA IS READ INTO
```

```
PTRTST: RESET                   ;RESET THE PROGRAM
        MOVE [XWD 1,1]          ;LOCK BOTH HIGH AND LOW SEGMENTS
        LOCK                    ;LOCK THE JOB IN CORE
        JRST FAILED             ;LOCK UUO FAILED
        SETZM PTRCSO            ;MAKE SURE CONSO BITS ARE ZERO
        SETZM DONFLG            ;INITIALIZE DONE FLAG
        MOVEI RTBLK             ;GET ADDRESS OF REAL TIME DATA BLOCK
        RTTRP                   ;PUT REAL TIME DEVICE ON THE PI LEVEL
        JRST FAILED             ;RTTRP UUO FAILED
        MOVEI 1,DONE            ;SET UP CONSO BIT MASK
        HLRZ 2,RTBLK            ;GET PI NUMBER FROM RTBLK
        TRO 2,BUSY              ;SET UP CONO BITS TO START TAPE GOING
        CONO PI,PIOFF           ;GUARD AGAINST ANY INTERRUPTS
        MOVEM 1,PTRCSO          ;STORE CONSO BIT MASK
        CONO PTR,(2)            ;TURN PTR ON
        CONO PI,PION            ;ALLOW INTERRUPTS AGAIN
        MOVEI 5                 ;SET UP TO SLEEP FOR 5 SECONDS
        SLEEP
        SKIPN DONFLG            ;HAVE WE FINISHED READING THE TAPE
        JRST .-3                ;NO GO BACK TO SLEEP
        EXIT

RTBLK:  XWD 5,TRPADR            ;PI CHANNEL AND TRAP ADDRESS
        EXP APRTRP              ;APR ERROR TRAP ADDRESS
        CONSO PTR,@PTRCSO       ;INDIRECT CONSO BIT MASK = PTRCSO
        Z                       ;NO BLKI/O INSTRUCTION

PTRCSO: Z                       ;CONSO BIT MASK
DONFLG: Z                       ;PI LEVEL TO USER LEVEL COMM.
RTBLK1: Z                       ;DATA BLOCK TO REMOVE PTR
        Z                       ;FROM PI CHANNEL
        CONSO PTR,0
        Z

TRPADR: CONSO PTR,TAPE          ;END OF TAPE?
        JRST TDONE              ;YES, GO STOP JOB
        DATAI PTR,PDATA         ;READ IN DATA WORD
        UJEN                    ;DISMISS THE INTERRUPT

APRTRP: Z                       ;APR ERROR TRAP ADDRESS
TDONE:  MOVEI RTBLK1            ;SET UP TO REMOVE PTR
        CONO PTR,0              ;TAKE DEVICE OFF HARDWARE PI LEVEL
        RTTRP                   ;REMOVE FROM SOFTWARE PI LEVEL
        JFCL                    ;IGNORE ERRORS
        SETOM DONFLG            ;MARK THAT READ IS OVER
        SETZM PTRCSO            ;CLEAR CONSO BIT MASK
        UJEN                    ;DISMISS THE INTERRUPT

FAILED: TTCALL 3,[ASCIZ/RTTRP UUO FAILED!/]
        EXIT

        END PTRTST
```

```
********** EXAMPLE 2 **********
        FAST BLOCK MODE

        TITLE RTFBLK - PAPER TAPE READ TEST IN BLKI MODE

        TAPE=400                ;NO MORE TAPE IN READER IF TAPE=0
        BUSY=20                 ;DEVICE IS BUSY READING
        DONE=10                 ;A CHARACTER HAS BEEN READ
BLKTST: RESET                   ;RESET THE PROGRAM
        MOVE [XWD 1,1]          ;LOCK BOTH HIGH AND LOW SEGMENTS
        LOCK                    ;LOCK THE JOB IN CORE
        JRST FAILED             ;LOCK UUO FAILED
        SETZM DONFLG            ;INITIALIZE DONE FLAG
        MOVEI RTBLK             ;GET ADDRESS OF REAL TIME DATA BLOCK
        RTTRP                   ;PUT REAL TIME DEVICE ON THE PI LEVEL
        JRST FAILED             ;RTTTP UUO FAILED
        HLRZ 2,RTBLK            ;GET PI NUMBER FROM RTBLK
        TRO 2,BUSY              ;SET UP CONO BITS TO START TAPE GOING
        CONO PTR,(2)            ;TURN PTR ON
        MOVEI 5                 ;SET UP TO SLEEP FOR 5 SECONDS
        SLEEP
        SKIPN DONFLG            ;HAVE WE FINISHED READING THE TAPE
        JRST .-3                ;NO GO BACK TO SLEEP
        EXIT

RTBLK:  XWD 6,TRPADR            ;PI CHANNEL AND TRAP ADDRESS
        EXP APRTRP              ;APR ERROR TRAP ADDRESS
        BLKI PTR,POINTR         ;READ A BLOCK AT A TIME
        Z

POINTR: IOWD 5,TABLE            ;POINTER FOR BLKI INSTRUCTION
OPOINT: IOWD 5,TABLE            ;ORIGINAL POINTER WORD FOR BLKI
TABLE:  BLOCK 5                 ;TABLE AREA FOR DATA BEING READ
DONFLG: Z                       ;PI LEVEL TO USER LEVEL COMM.
RTBLK1: Z                       ;DATA BLOCK TO REMOVE PTR
        Z                       ;FROM PI CHANNEL
        CONSO PTR,0
        Z

TRPADR: CONSO PTR,TAPE          ;END OF TAPE?
        JRST TDONE              ;YES, GO STOP JOB
        MOVE OPOINT             ;GET ORIGINAL POINTER WORD
        MOVEM POINTR            ;RESTORE BLKI POINTER WORD
        UJEN                    ;DISMISS THE INTERRUPT

APRTRP: Z                       ;APR ERROR TRAP ADDRESS
TDONE:  MOVEI RTBLK1            ;SET UP TO REMOVE PTR
        CONO PTR,0              ;TAKE DEVICE OFF HARDWARE PI LEVEL
        RTTRP                   ;REMOVE FROM SOFTWARE PI LEVEL
        JFCL                    ;IGNORE ERRORS
        SETOM DONFLG            ;MARK THAT READ IS OVER
        UJEN                    ;DISMISS THE INTERRUPT

FAILED: TTCALL 3,[ASCIZ/RTTRP UUO FAILED!/]
        EXIT

        END BLKTST
```

```
          ********** EXAMPLE 3 **********
                  NORMAL BLOCK MODE

          TITLE RTNBLK - PAPER TAPE READ TEST IN BLKI MODE

          TAPE=400                ;NO MORE TAPE IN READER IF TAPE=0
          BUSY=20                 ;DEVICE IS BUSY READING
          DONE=10                 ;A CHARACTER HAS BEEN READ

BLKTST:   RESET                   ;IO RESET
          MOVE [XWD 1,1]          ;LOCK BOTH HIGH AND LOW SEGMENTS
          LOCK                    ;LOCK THE JOB IN CORE
          JRST FAILED             ;LOCK UUO FAILED
          MOVEI RTBLK1            ;GET ADDRESS OF REAL TIME BLOCK
          RTTRP                   ;GET USER IOT PRIVILEGE
          JRST FAILED             ;UUO FAILED!
          CONO PTR,0              ;CLEAR ALL PTR FLAGS
          SETZM DONFLG            ;INITIALIZE DONE FLAG
          MOVEI RTBLK             ;GET ADDRESS OF REAL TIME DATA BLOCK
          RTTRP                   ;PUT REAL TIME DEVICE ON THE PI LEVEL
          JRST FAILED             ;RTTRP UUO FAILED
          MOVE POINTR             ;GET RELOCATED POINTER WORD FOR LATER
          MOVEM OPOINT            ;STORE FOR INTERRUPT LEVEL USE
          HLRZ 2,RTBLK            ;GET PI NUMBER FROM RTBLK
          TRO 2,BUSY              ;SET UP CONO BITS TO START TAPE GOING
          CONO PTR,(2)            ;TURN PTR ON
          MOVEI 5                 ;SET UP TO SLEEP FOR 5 SECONDS
          SLEEP
          SKIPN DONFLG            ;HAVE WE FINISHED READING THE TAPE
          JRST .-3                ;NO GO BACK TO SLEEP
          EXIT

RTBLK:    XWD 6,TRPADR            ;PI CHANNEL AND TRAP ADDRESS
          EXP APRTRP              ;APR ERROR TRAP ADDRESS
          CONSO PTR,DONE          ;WAIT ONLY FOR DONE FLAG
          BLKI PTR,POINTR         ;READ A BLOCK AT A TIME

POINTR:   IOWD 5,TABLE            ;POINTER FOR BLKI INSTRUCTION
OPOINT:   Z
TABLE:    BLOCK 5                 ;TABLE AREA FOR DATA BEING READ
DONFLG:   Z                       ;PI LEVEL TO USER LEVEL COMM.
RTBLK1:   Z                       ;DATA BLOCK TO REMOVE PTR
          Z                       ;FROM PI CHANNEL
          CONSO PTR,0
          Z

TRPADR:   CONSO PTR,TAPE          ;END OF TAPE?
          JRST TDONE              ;YES, GO STOP JOB
          MOVE OPOINT             ;GET ORIGINAL POINTER LOCATION
          MOVEM POINTR            ;STORE IN POINTER LOCATION
          UJEN                    ;DISMISS THE INTERRUPT

APRTRP:   Z                       ;APR ERROR TRAP ADDRESS
TDONE:    MOVEI RTBLK1            ;SET UP TO REMOVE PTR
          CONO PTR,0              ;TAKE DEVICE OFF HARDWARE PI LEVEL
          RTTRP                   ;REMOVE FROM SOFTWARE PI LEVEL
          JFCL                    ;IGNORE ERRORS
          SETOM DONFLG            ;MARK THAT READ IS OVER
          UJEN                    ;DISMISS THE INTERRUPT

FAILED:   TTCALL 3,[ASCIZ/RTTRP UUO FAILED!/]
          EXIT

          END BKJTST
```

3.8.2 RTTRP Executive Mode Trapping

In special cases, the real-time user requires a faster response time than that offered by the RTTRP UUO when executed in user mode. To accommodate these cases, the user can specify a special status bit in the RTTRP UUO call, which gives the program control in exec mode (refer to Paragraph 2.1.3). Exec-mode trapping gives response times of less than 10 μs to real-time interrupts. To use this exec-mode trapping, the job must have real-time privileges (granted by LOGIN) and be locked in core (accomplished by the LOCK UUO). On KI10-based systems, the job must also be mapped contiguously in exec virtual memory (refer to the LOCK UUO). The privilege bits required are:

1) JP.TRP (Bit 15)
2) JP.LCK (Bit 14)
3) JP.RTT (Bit 13)

Several restrictions must be placed on user programs in order to achieve this level of response. On receipt of an interrupt, program control is transferred to the user's real-time program without saving ACs and with the processor in exec mode. Therefore, the user program must save and restore all ACs that are used, must not execute any UUOs, and cannot leave exec mode. This means that the programs must be self-relocating (i.e., through the use of an index or base register).

CAUTION

Improper use of the exec mode feature of the RTTRP UUO can cause the system to fail in a number of ways. Unlike the user mode feature of RTTRP, errors are not protected against since the programs run in exec mode with no ACs saved.

To specify RTTRP exec-mode trapping, bit 17 of the second word in the data block (RTBLK) must be set to 1. This implies that no context switching is to be done and that a JSR TRPADR is to be used to enter the user's real-time interrupt routine. The user program must save and restore all ACs and should dismiss the interrupt with a JRSTF @ TRPADR. This instruction must be set up prior to the start of the real-time device as an absolute or unrelocated instruction. This can be done because the LOCK UUO returns the absolute addresses of the low and high segments after the job is locked in a fixed place in memory.

The exec-mode trapping feature can be used with any of the standard forms of the RTTRP UUO: single mode, normal block mode, and fast block mode.

3.8.2.1 Example

```
        TITLE RTEXEC

        PIA=5
        DONE=10
        BUSY=20
        TAPE=400
        I=1
        AC=2
        OPDEF HIBERNATE [CALLI 72]

RTEXEC: RESET                   ;RESET THE PROGRAM
        SETZM DONFLG            ;INITIALIZE THE DONE FLAG
        MOVE AC,[XWD 1,1]
        LOCK AC,                ;LOCK THE JOB IN CORE
                                ;ABSOLUTE ADDRESS OF JOB IS RETURNED
                                ;IN AC
        JRST FAILED             ;ERROR RETURN
        HRRZS AC                ;GET ONLY LOW SEGMENT ADDRESS
        LSH AC,9                ;JUSTIFY ADDRESS
        MOVEM AC,INDEX          ;SAVE BASE ADDRESS FOR USE AT INTERRUPT
                                ;LEVEL
        ADDM AC,EXCHWD          ;RELOCATE INTERRUPT LEVEL PROGRAM
        ADDM AC,JENWD           ;RELOCATE RETURN INSTRUCTION
        MOVEI AC,RTBLK          ;CONNECT REAL TIME DEVICE
        RTTRP AC,               ;TO THE PI SYSTEM
        JRST FAILED             ;RTTRP UUO FAILED
        CONO PTR,20+PIA         ;START REAL TIME DEVICE READING
SLEEP:  MOVEI AC,^D1000         ;SLEEP
        HIBERNATE AC,           ;FOR 10 MILLISECONDS
        JRST FAILED             ;FAILED
        SKIPN DONFLG            ;IS THE INTERRUPT LEVEL PROGRAM DONE
        JRST SLEEP              ;NO, GO BACK TO SLEEP
        EXIT                    ;YES, EXIT

RTBLK:  XWD PIA,TRPADR
        XWD 1 APRTRP            ;BIT 17 SAYS TRAP IN EXEC MODE
        CONSO  TR,DONE
        0

TRPADR: 0                       ;JSR TRPADR IS DONE UPON INTERRUPT
EXCHWD: EXCH I,INDEX            ;SET UP INDEX REGISTER
        CONSO PTR,TAPE          ;TAPE FINISHED?
        JRST TDONE(I)           ;YES, STOP THE READER
        DATAI PTR,PDATA(I)      ;NO, READ IN THE NEXT CHARACTER
RETURN: EXCH I,INDEX(I)         ;RESTORE AC'S USED
JENWD:  JRSTF @TRPADR           ;DISMISS THE INTERRUPT

APRTRP: 0                       ;APR ERRORS WILL TRAP HERE
TDONE:  CONO PTR,0              ;TAKE THE READER OFF LINE
        SETOM DONFLG(I)         ;MARK THAT THE TAPE IS FINISHED
        JRST RETURN(I)          ;GO DISMISS THE INTERRUPT

FAILED: TTCALL 3,[ASCIZ/UUO FAILURE/]
        EXIT

DONFLG: 0                       ;FLAG TO SPECIFY END OF JOB
PDATA:  0                       ;DATA WORD
INDEX:  0                       ;BASE INDEX REGISTER

        END RTEXEC
```

3.8.3 TRPSET AC, or CALLI AC, 25[1]

The TRPSET feature may be used to guarantee some of the fast response requirements of real-time users. In order to achieve fast response to interrupts, this feature temporarily suspends the running of other jobs during its use. This limits the class of problems to be solved to cases where the user wants to transfer data in short bursts at predefined times. Therefore, because the data transfers are short, the time during which timesharing is stopped is also short, and the pause probably will not be noticed by the timesharing users.

The TRPSET UUO allows the user program to gain control of the interrupt locations. If the user does not have the TRPSET privileges (JP.TRP, bit 15), an error return to the next location after the CALLI is always given, and the user remains in user mode. Timesharing is turned back on. If the user has the TRPSET privileges, the central processor is placed in user I/O mode. If AC contains zero, timesharing is turned on if it was turned off. If the LH of AC is within the range 40 through 57 of the central processor, all other jobs are stopped from being scheduled and the specified executive PI location (40-57) is patched to trap directly to the user. In this case, the monitor moves the contents of the relative location specified in the right half of AC, converts the user virtual address to the equivalent exec virtual address, and stores the address in the specified executive PI location. On a KI10-based system, this requires that the user segment accessed during the interrupt be locked and mapped contiguously in the exec virtual memory (refer to the LOCK UUO). If the segment does not meet these requirements, the error return is given.

On a multiprocessor system, the TRPSET UUO applies to the processor specified by the job's CPU specification (refer to the SET CPU command or the SET UUO). If the specification indicates only CPU1, an error return is given if the job is not locked in core. When the specification indicates more than one processor, the specification is changed to indicate CPU0 (the master processor).

Thus, the user can set up a priority interrupt trap into his relocated core area. On a normal return, AC contains the previous contents of the address specified by LH of AC, so that the user program may restore the original contents of the PI location when the user is through using this UUO. If the LH of AC is not within the range 40 through 57, an error return is given just as if the user did not have the privileges. The basic call is:

```
        MOVE AC,[XWD N, ADR]
        TRPSET AC,
        ERROR RETURN
        NORMAL RETURN
        .
        .
        .
ADR:    JSR TRAP                ;Instruction to be stored
                                ;in exec PI location
                                ;after relocation added to it.
TRAP:   0                       ;Here on interrupt from exec.
```

[1] This UUO depends on FTTRPSET which is normally off in the DECsystem-1040.

The monitor assumes that user ADR contains either a JSR U or BLKI U, where U is a user virtual address; consequently, the monitor adds a relocation to the contents of location U to make it an absolute IOWD (i.e., an exec virtual address). Therefore, a user should reset the contents of U before every TRPSET call.

A RESET UUO returns the user to normal user mode. The following instruction sequence is used to place the real-time device RTD on channel 3.

```
INT46:  BLKI RTD,INBLOK                 ;relocation constant
                                        ;for user is added
INT47:  JSR XITINT                      ;to RH when instructions
        .                               ;are placed into 46 and 47.
        .
        .
START:  MOVEI AC,INT46
        HRLI AC,46
        TRPSET AC,
        JRST EXITR                      ;error return
        MOVE AC, [XWD 47, INT47]        ;normal return
        TRPSET AC,
        JRST EXITR                      ;error return
        .                               ;normal return
        .
        .
XITINT: 0                               ;PC saved
        . . .                           ;interrupt dismiss routine
```

To maintain compatibility between a KA10-based system and a KI10-based system, the interrupt routine should be executed in exec mode. However, for convenience, the routine can be executed in user mode in order to avoid relocation to exec virtual memory. This is possible on KA10-based systems if care is taken when dismissing the interrupt (see example below). On KI10-based systems, if there is a possibility that the interrupt may occur during the job's background processing, the interrupt routine must be executed in exec mode (and thus must be locked and exec-mapped with the LOCK UUO). In particular, if the job is executing a UUO at background level, the use of UJEN at interrupt level may cause an error. On KI10-based systems it is recommended that the TRPSET interrupt routines always be coded to run in exec mode (refer to the RTTRP UUO for programming techniques.)

On KA10-based system, the interrupt routine can be coded to run in user mode if the following procedure is observed. If the interrupt occurs while some other part of the user's program is running, the user may dismiss from the interrupt routine with a JEN @ XITINT. However, if the machine is in exec mode, a JEN instruction issued in user mode does not work because of memory relocation. This is solved by a call to UJEN (op code 100). This UUO causes the monitor to dismiss the interrupt from exec mode. In this case, the address field of the UJEN instruction is the user location when the return PC is stored (i.e., UJEN XITINT). The following sequence enables the user program to decide whether it can issue a JEN to save time or dismiss the interrupt with a UUO call.

Example (KA10-based system only):

```
XITINT:  0                              ;PC with bits in LH

         JRST 1,.+1                     ;essential instruction.
                                        ;returns machine to
                                        ;user mode.
         MOVEM AC, SAVEAC               ;save accumulator AC
         .                              ;service interrupt here
         .
         .

         MOVE AC, XITINT                ;get PC with bits
         SETZM EFLAG

         TLNN AC, 10000                 ;was machine in user
                                        ;mode at entry?
         SETOM EFLAG                    ;no
         MOVE AC, SAVEAC                ;RESTORE saved AC
         SKIPE EFLAG
         UJEN XITINT                    ;not in user mode at entry
         JEN @ XITINT

SAVEAC:  0

EFLAG:   0
```

On entering the routine from absolute 47 with a JSR to XITINT + REL (where REL. is the relocation constant), the processor enters exec mode. The first executed instruction in the user's routine must, therefore, reset the user mode flag, thereby enabling relocation and protection. The user must proceed with caution when changing channel interrupt chains under timesharing, making certain that the real-time job can co-exist with other timesharing jobs.

3.8.4 UJEN (Op Code 100)

This op code dismisses a user I/O mode interrupt if one is in progress. If the interrupt is from user mode, a JRST 12, instruction dismisses the interrupt. If the interrupt came from executive mode, however, this operator is used to dismiss the interrupt. The monitor restores all accumulators, and executes JEN @ U where user location U contains the program counter as stored by a JSR instruction when the interrupt occurred.

3.8.5 HPQ AC, or CALLI AC, 71[1]

The HPQ UUO is used by privileged users to place their jobs in a high-priority scheduler run queue. These queues are always scanned by the scheduler before the normal run queues, and any runnable job

[1] This UUO depends on FTHPQ which is normally off in the DECsystem-1040.

in one of these queues is executed before all other jobs in the system. In addition, these jobs are given preferential access to sharable resources (e.g., shared device controllers). Thus, real-time associated jobs can receive fast response from the timesharing scheduler.

Jobs in high-priority queues are not examined for swap-out until all other queues have been scanned. If a job in a high-priority queue must be swapped, the lowest priority job is transferred first, and the highest priority job last. If the highest priority job is swapped, then that job is the first to be swapped in for immediate execution. Therefore, in addition to being scanned before all other queues for job execution, the high-priority queues are examined after all other queues for swap-out and before all queues for swap-in.

The HPQ UUO requires as an argument the high-priority queue number of the queue to be entered. The lowest high-priority queue is 1, and the highest priority queue is equivalent to the number of queues that the system is built for. The call is as follows:

```
MOVE AC, HPQNUM          ;get high-priority queue number
HPQ AC,                  ;or CALLI AC, 71
error return
normal return
```

On an error return, AC contains -1 if the user did not have the correct privileges. The privilege bits are 6 through 9 in the privilege word (.GTPRV). These four bits specify a number from 0-17 octal, which is the highest priority queue attainable by the user.

On a normal return, the job is in the desired high-priority queue. A RESET or an EXIT UUO returns the job to the high-priority queue specified in the last SET HPQ command. A queue number of 0 as an argument places the job back to the timesharing level.

3.9 METER.AC, OR CALLI AC, 111[1]

This UUO provides a mechanism for system performance metering by allowing privileged users to dynamically select and collect performance statistics from the monitor. The multifunction UUO controls all aspects of the metering facility in order that the user can collect, present, or reduce data for performance analysis or can tune individual jobs or the entire system. The METER. UUO requires JP.MET (bit 3) to be set in the privilege word .GTPRV.

---

[1] This UUO depends on FTMETR which is normally off in the DECsystem-1040.

The general call is:

```
MOVE AC, [XWD N, ADR]
METER.AC,                    ;or CALLI AC, 111
error return
normal return
```

where

N is the number of arguments in the argument list.
ADR is the beginning of the argument list.

If N is 0, the default number of arguments depends on the particular function used. Arguments in the list can be 1) arguments for the monitor, 2) values returned from the monitor, or 3) a combination of both. The first word of the argument block is the code for the particular function. The detailed descriptions of the various functions of the METER. UUO are presented in the METER. Specification in the Software Notebooks; the following is a list of the functions available.

Function Code	Name	Description
0	.MEFCI	Initialize meter channel
1	.MEFCS	Obtain meter channel status
2	.MEFCR	Release meter channel
3	.MEFPI	Initialize meter points
4	.MEFPS	Obtain meter point status
5	.MEFPR	Release meter points

On an error return, the appropriate error code is returned in AC. Refer to the METER. Specification for the error codes.

On a normal return, AC is perserved.

CHAPTER 4
I/O PROGRAMMING

All user-mode I/O programming is controlled by monitor programmed operators. I/O is directed by

a. Associating a device and a ring of buffers with one of the user's I/O channels (INIT, OPEN).

b. Optionally selecting a file (LOOKUP, ENTER).

c. Passing buffers of data to or from the user program (IN, INPUT, OUT, OUTPUT).

Device specification may be delayed from program-generation time until program-run time because the monitor

a. Allows a logical device name to be associated with a physical device (ASSIGN or MOUNT monitor command).

b. Treats operations that are not pertinent to a given device as no-operation code.

For example: a rewind directed to a line printer does nothing, and file selection operations for devices without a filename directory are always successful.

4.1 I/O ORGANIZATION

4.1.1 Files

A file is an ordered set of data on a peripheral device. The extent of a file on input is determined by an end-of-file condition dependent on the device. For example: a file is terminated by reading an end-of-file gap from magnetic tape, by an end-of-file card from a card reader, or by depressing the end-of-file switch on a card reader (refer to Chapter 5). The extent of a file on output is determined by the amount of information written by the OUT or OUTPUT programmed operators up through and including the next CLOSE or RELEAS operator.

4.1.2 Job I/O Initialization

The monitor programmed operator

CALL [SIXBIT /RESET/] or CALLI 0

should normally be the first instruction in each user program. It immediately stops all I/O transmissions on all devices without waiting for the devices to become inactive. All device allocations made by the INIT and OPEN operators are cleared and, unless the devices have been assigned by the ASSIGN or MOUNT monitor command, the devices are returned to the monitor facilities pool. The content of the left half of .JBSA (program break) is stored in the right half of .JBFF so that the user buffer area is reclaimed if the program restarts. The left half of .JBFF is cleared. Any files that have not been closed are deleted on disk. Any older version with the same filename remains. The user-mode write-protect bit is automatically set if a high segment exists, whether it is sharable or not; therefore, a program cannot inadvertently store into the high segment. Additional functions of the RESET UUO include 1) unlocking the job if it was locked, 2) releasing any real-time devices, 3) resetting any high-priority queues set by the HPQ UUO to the value set by the HPQ command, 4) resuming timesharing if it was stopped as a result of a TRPSET UUO with a non-zero argument, 5) resetting the action of the HIBER and APRENB UUOs, and 6) clearing all PC flags except USRMOD.

4.2 DEVICE SELECTION

For all I/O operations, a specific device must be associated with a software I/O channel. This specification is made by an argument of the INIT or the OPEN programmed operators. The INIT or the OPEN programmed operators may specify a device with a logical name that is associated with a particular physical device by the ASSIGN or MOUNT monitor command. Some system programs, e.g., LOGOUT, require I/O to specific physical devices regardless of what logical names have been assigned. Therefore, on an OPEN UUO, if the sign bit of word 0 of the OPEN block is 1 (UU.PHS), the device name is taken as a physical name only, and logical names are not searched. A given device remains associated with a software I/O channel until released (refer to Paragraph 4.8.1) or until another INIT or OPEN is performed for that channel. Devices are separated into two categories: those with no filename directory (refer to Chapter 5) and those with at least one filename directory (refer to Chapter 6).

Assignable devices (i.e., non-disk and non-spooled devices) in the monitor's pool of available resources are designated as being either unrestricted or restricted. An unrestricted device can be assigned directly by any job via the ASSIGN command or INIT or OPEN UUO. A restricted device can be assigned directly only by a privileged job (i.e., a job logged in under [1,2] or running with the JACCT bit set). However, a non-privileged user can have a restricted device assigned to him indirectly via the MOUNT command. This command allows operator intervention for the selection or denial of a particular device; thus the operator can control the use of assignable devices for the non-privileged user. This is particularly useful when there are multiprogramming batch and interactive jobs competing for the same devices. The restricted status of a device is set or removed by the operator with the OPSER commands :RESTRICT and :UNRESTRICT, which use the privileged UUOs, DVRST. and DVURS. (refer to UUOPRV in the DECsystem-10 Software Notebooks).

4.2.1 Nondirectory Devices

For nondirectory devices (e.g., card reader and punch, line printer, paper-tape reader and punch, and user terminal), selection of the device is sufficient to allow I/O operations over the associated

software channel. All other file specifiers, if given, are ignored. Magnetic tape, a nondirectory device, requires, in addition to the name, that the tape be properly positioned. It is advisable to use the programmed operators that select a file, so that a directory device may be substituted for a nondirectory device at run time.

4.2.2 Directory Device

For directory devices (e.g., DECtape and disk), files are addressable by name. If the device has a single file directory (e.g., DECtape) the device name and filename are sufficient information to determine a file. If the device has multiple file directories (e.g., disk) the name of the file directory must also be specified. These names are specified as arguments to the LOOKUP, ENTER, and RENAME programmed operators.

4.2.3 Device Initialization

The OPEN (operation code 050) and INIT (operation code 041) programmed operators initialize a device and associate it with a software I/O channel number for the job. These UUOs perform almost identical functions; the OPEN UUO is a reentrant form of INIT and is preferred for this reason. In addition to the device name, these programmed operators accept, as arguments, an initial file status and the location of the input and output buffer headers. The calls are:

```
OPEN D,SPEC                    INIT D,STATUS
error return                   SIXBIT /dev/
normal return                  XWD OBUF, IBUF
      :                        error return
      :                        normal return
      .
SPEC: EXP STATUS
      SIXBIT /dev/
      XWD OBUF, IBUF
```

The normal return is taken if a device is selected, and if the device is associated with a software I/O channel. The error return is taken if the requested device is in use, if the requested device does not exist, or if the device is restricted and has not been assigned with the MOUNT command.

4.2.3.1 Data Channel - These programmed operators establish a correspondence between the device and a 4-bit channel number, D. Most of the other input/output operators require this channel number as an argument. If a device is already assigned to channel D, it is released (refer to Paragraph 4.8.1).

4.2.3.2 Device Name - The device name, dev, is either a logical or physical device name, with logical names taking precedence over physical names. With multiple stations, the method of device selection depends on the format of the specified SIXBIT device name.

If devn (e.g., LPT7, CDR3) is specified, the monitor attempts to select the device specifically requested.

If devSnn (e.g., CDPS14, PTPS12) is specified, the monitor attempts to select any device of the desired type at the requested station. If a device of the desired type has been previously assigned to this job at the requested station and is not INITed on another channel, it will be selected in preference to an unassigned device.

If dev (e.g., LPT, DTA) is specified, the monitor attempts to select a device of the desired type at the job's logical station. If all devices of this type are in use, the error return is taken. If no device of the desired type exists at the user's logical station, the monitor attempts to select the device at the central station. If the desired type of device has already been assigned to the job at the appropriate station (either the job's logical station or the central station) and is not INITed on another channel, it will be selected instead of an unassigned device.

In non-disk systems, if the specified device is the system device SYS, the job is placed into a system device wait queue and continues to run when SYS becomes available. In disk systems where the system device is one or more file structures, control returns immediately.

The job may pause with the message

?STATION nn NOT IN CONTACT

if the requested station is not in contact with the central station. After station nn has established contact with the central station, the user types CONTINUE for a return to job execution.

4.2.3.3 Initial File Status - The file status, including the data mode, is set to the value of the symbol STATUS. Thereafter, bits are set by the monitor and may be tested and reset by the user via monitor programmed operators. Bits 30-35 of the file status are normally set by an OPEN or INIT UUO. Refer to Table 4-3 in Paragraph 4.6.2 for the file status bits. If the data mode is not legal (refer to Chapters 5 and 6) for the specified device, the job is stopped and the monitor prints

ILL DEVICE DATA MODE FOR DEVICE dev AT USER addr,

where dev is the physical name of the device and addr is the location of the OPEN or INIT operator on the user's terminal. The terminal is left in monitor mode.

4.2.3.4 Data Modes - Data transmissions are either unbuffered or buffered. (Unbuffered mode is sometimes referred to as dump mode.) The mode of transmission is specified by a 4-bit argument to the INIT, OPEN, or SETSTS programmed operator. Tables 4-1 and 4-2 summarize the data modes.

Table 4-1
Buffered Data Modes

Octal Code	Name	Meaning
0	.IOASC	ASCII. Seven bit bytes packed left justified, five characters per word.
1	.IOASL	ASCII line. Same as 0, except that the buffer is terminated by a FORM, VT, LINE-FEED, or ALTMODE character. Differs from ASCII on TTY (half-duplex software) and PTR only.
2-7		Unused.
10	.IOIMG	Image. A device dependent mode. Thirty-six bit bytes. The buffer is filled with data exactly as supplied by the device.
11-12		Unused.
13	.IOIBN	Image binary. Thirty-six bit bytes. This mode is similar to binary mode, except that no automatic formatting or checksumming is done by the monitor.
14	.IOBIN	Binary. Thirty-six bit bytes. This is blocked format consisting of a word count, n (the right half of the first data word of the buffer), followed by n 36-bit data words. Checksum for cards and paper tape.

Table 4-2
Unbuffered Data Modes

Octal Code	Name	Meaning
15	.IOIDP	Image dump. A device dependent dump mode. Thirty-six bit bytes.
16	.IODPR	Dump as records without core buffering. Data is transmitted between any contiguous blocks of core and one or more standard length records on the device for each command word in the command list. Thirty-six bit bytes.
17	.IODMP	Dump one record without core buffering. Data is transmitted between any contiguous block of core and exactly one record of arbitrary length on the device for each command word in the command list. Thirty-six bit bytes.

4.2.3.5 Buffer Header - Symbols OBUF and IBUF, if non-zero specify the location of the first word of the 3-word buffer ring header block for output and input, respectively. Buffered data modes utilize a ring of buffers in the user area and the priority interrupt system to permit the user to overlap computation with his data transmission. Core memory in the user's area serves as an intermediate buffer

between the user's program and the device. The buffer storage mechanism consists of a 3-word buffer ring header block for bookkeeping and a data storage area subdivided into one or more individual buffers linked together to form a ring. During input operations, the monitor fills a buffer, makes that buffer available to the user's program, advances to the next buffer in the ring, and fills that buffer if it is free. The user's program follows the monitor, either emptying the next buffer if it is full or waiting for it to fill.

During output operations, the user's program and the monitor exchange roles; the user program fills the buffers and the monitor empties them. Only the headers that will be used need to be specified. For instance, the output header need not be specified, if only input is to be done. Also, data modes 15, 16, and 17 require no header. If either of the buffer headers or the 3-word block starting at location SPEC lies outside the user's allocated core area†, the job is stopped and the monitor prints

ILLEGAL UUO AT USER addr

(addr is the address of the OPEN or INIT operator) on the user's terminal, leaving the terminal in monitor mode.

The first and third words of the buffer header are set to zero. The left half of the second word is set up with the byte pointer size field in bits 6 through 11 for the selected device-data mode combination.

If the same device (other than disk) is INITed on two or more channels, the monitor retains only the buffer headers mentioned in the last INIT (a 0 specification does not override a previous buffer header specification). Other I/O operations to any of the channels involved act on the buffers mentioned in the last INIT previous to the I/O operations.

4.3 RING BUFFERS

4.3.1 Buffer Structure

The ring buffer (see Figure 4-1) is comprised of a buffer ring header block and buffer rings.

4.3.1.1 Buffer Ring Header Block - The location of the 3-word buffer ring header block is specified by an argument of the INIT and OPEN operators. Information is stored in the header by the monitor in response to the user execution of monitor programmed operators. The user's program finds all the information required to fill and empty buffers in the header. Bit position 0 of the first word of the header is a flag, which, if 1, means that no input or output has occurred for this ring of buffers. The

† Buffer headers may not be in the user's ACs; however, the buffer headers may be in location above .JBPFI (refer to Table 1-1 in Paragraph 1.2.1).

right half of the first word is the address of the second word of the buffer currently used by the user's program. The second word of the header contains a byte pointer to the current byte in the current buffer. The byte size is determined by the data mode. The third word of the header contains a number of bytes remaining in the buffer. A program may not use a single buffer header for both input and output, nor may a single buffer ring header be used for more than one I/O function at a time. Users cannot use the same buffer ring for simultaneous input and output; only one buffer ring is associated with each buffer ring header.

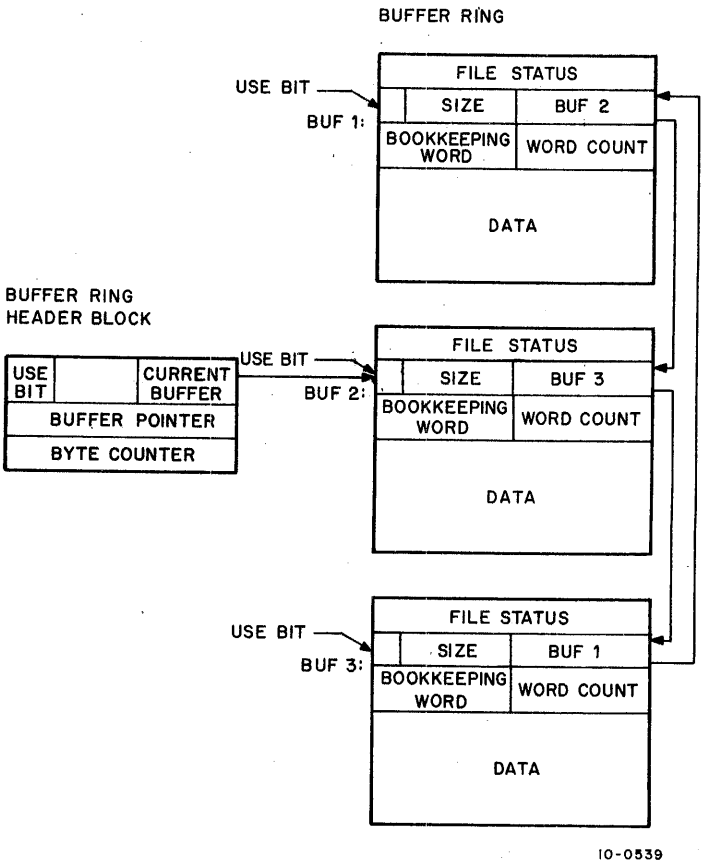


Figure 4-1 User's Ring of Buffers

4.3.1.2 Buffer Ring - The buffer ring is established by the INBUF and OUTBUF operators, or, if none exists when the first IN, INPUT, OUT, or OUTPUT operator is executed, a 2-buffer ring is set up. The effective address of the INBUF and OUTBUF operators specifies the number of buffers in the ring. The location of the buffer ring is specified by the contents of the right half of .JBFF in the user's job data area. The monitor updates .JBFF to point to the first location past the storage area.

All buffers in the ring are identical in structure. The right half of the first word contains the file status when the monitor advances to the next buffer in the ring (see Figure 4-2). Bit 0 of the second word of a buffer, the use bit, is a flag that indicates whether the buffer contains active data. This bit is set to 1 by the monitor when the buffer is full on input or being emptied on output, and set to 0 when the buffer is empty on output or is being filled on input. In other words, if the use bit = 0, the buffer is available to the filler; if the use bit = 1, the buffer is available to the emptier. The use bit prevents the monitor and the user's program from interfering with each other by attempting to use the same buffer simultaneously. Buffers are advanced by the UUOs and not by the user's program. The use bit in each buffer should never be changed by the user's program except by means of the UUOs. Bits 1 through 17 of the second word of the buffer contain the size of the data area of the buffer plus 1. The size of this data area depends on the device. The right half of the third word of the buffer is reserved for a count of the number of words that actually contain data. The left half of this word is reserved for other bookkeeping purposes, depending on the particular device and the data mode.

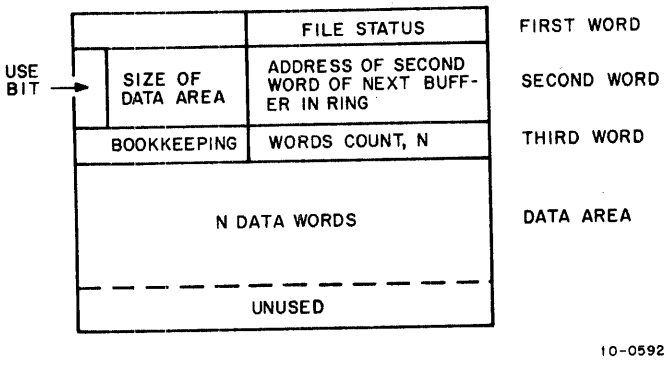


Figure 4-2 Detailed Diagram of Individual Buffer

4.3.2 Buffer Initialization

Buffer data storage areas may be established by the INBUF and OUTBUF programmed operators, or by the first IN, INPUT, OUT, or OUTPUT operator, if none exists at that time, or the user may set up his own buffer data storage area.

4.3.2.1 Monitor Generated Buffers - Each device has an associated standard buffer size (refer to Chapters 5 and 6). The monitor programmed operators INBUF D,n (operation code 064) and OUTBUF D,n (operation code 065) set up a ring of n standard size buffers associated with the input and output buffer headers, respectively, specified by the last OPEN or INIT operator on data channel

D. If n = 0 on either INBUF or OUTBUF, the default number of buffers for the specified device is set up. If no OPEN or INIT operator has been performed on channel D, the monitor stops the job and prints

I/O TO UNASSIGNED CHANNEL AT USER addr

(addr is the location of the INBUF or OUTBUF operator) on the user's terminal leaving the terminal in the monitor mode.

The storage space for the ring is taken from successive locations, beginning with the location specified in the right half of .JBFF. This location is set to the program break, which is the first free location above the program area, by RESET. If there is insufficient space to set up the ring, the monitor automatically attempts to expand the user's core allocation by 1K. If this fails, the monitor stops the job and prints

ADDRESS CHECK FOR DEVICE dev AT USER addr

(dev is the physical name of the device associated with channel D and addr is the location of the INBUF or OUTBUF operator) on the user's terminal, leaving the terminal in monitor mode.

This message is also printed when an INBUF (OUTBUF) is attempted if the last INIT or OPEN UUO on channel D did not specify an input (output) buffer header.

The ring is set up by setting the second word of each buffer with a zero use bit, the appropriate data area size, and the link to the next buffer. The first word of the buffer header is set with a 1 in the ring use bit, and the right half contains the address of the second word of the first buffer.

4.3.2.2 User Generated Buffers - The following code illustrates an alternative to the use of the INBUF programmed operator. Analogous code may replace OUTBUF. This user code operates similarly to INBUF. SIZE must be set equal to the greatest number of data words expected in one physical record.

```
GO:     OPEN I,OPNBLK                       ;INITIALIZE ASCII MODE
          JRST NOTAVL                       ;THE 400000 IN THE LEFT HALF
                                            ;MEANS THE BUFFER WAS NEVER
        MOVE 0, [XWD 400000,BUF1+1]         ;REFERENCED.

                                            ;SET UP NON-STANDARD BYTE
        MOVEM 0, MAGBUF                     ;SIZE
        MOVE 0, [POINT BYTSIZ,0,35]

        MOVEM 0, MAGBUF+1                   ;MAGNETIC TAPE UNIT 0
        JRST CONTIN                         ;INPUT ONLY
```

(continued on next page)

```
OPNBLK:  0
         SIXBIT/MTA0/
         XWD 0,MAGBUF                ;GO BACK TO MAIN SEQUENCE
MAGBUF:  BLOCK 3                     ;SPACE FOR BUFFER RING HEADER

BUF1:    0                           ;BUFFER 1, 1ST WORD UNUSED
         XWD SIZE+1,BUF2+1           ;LEFT HALF CONTAINS DATA AREA
                                     ;SIZE+1, RIGHT HALF HAS
                                     ;ADDRESS OF NEXT BUFFER
                                     ;SPACE FOR DATA, 1ST WORD
                                     ;RECEIVES WORD-COUNT. THUS
         BLOCK SIZE +1               ;ONE MORE WORD IS RESERVED
                                     ;THAN IS REQUIRED FOR DATA
                                     ;ALONE
                                     ;SECOND BUFFER

BUF2:    0                           ;THIRD BUFFER
         XWD SIZE+1,BUF3+1           ;RIGHT HALF CLOSES THE RING
         BLOCK SIZE+1
BUF3:    0
         XWD SIZE+1,BUF1+1
         BLOCK SIZE+1
```

4.4 FILE SELECTION (LOOKUP and ENTER)

The LOOKUP (operation code 076) and ENTER (operation code 077) programmed operators select a file for input and output, respectively. These operators are not necessary for nondirectory devices; however, it is good programming practice to always use them so that directory devices may be substituted at run time (refer to ASSIGN command). The monitor gives the normal return for a LOOKUP or ENTER to a nondirectory device; therefore, user programs can be coded in a device-independent fashion.

4.4.1 The LOOKUP Operator

LOOKUP selects a file for input on channel D.

```
         LOOKUP D,E
         error return
         normal return
           .
           .
           .
E:  SIXBIT/file/                     ;filename, 1 to 6 characters, left-justified
    SIXBIT/ext/                      ;filename extension, 0 to 3
                                     ;characters, left-justified
      —                              ;The remaining words in the argument block
      —                              ;are ignored for nondirectory devices. Refer
                                     ;to Paragraph 6.1.5.1 for the DECtape
                                     ;LOOKUP and Paragraph 6.8.2.1 for the
                                     ;disk LOOKUP.
```

If no device has been associated with channel D by an INIT or OPEN UUO, the monitor stops the job, prints

I/O TO UNASSIGNED CHANNEL AT USER LOC addr

and returns the user's terminal to monitor mode. The input side of channel D is closed if not already closed. The output side is not affected.

On DECtape, LOOKUP searches the device directory as specified by an INIT. On disk, the user's file directory as specified by the contents of location E+3 is searched. Refer to Paragraph 6.1.5.1 for details of a DECtape LOOKUP and Paragraph 6.8.2.1 for details of a disk LOOKUP.

If the device is a directory device and the file is found, the normal return is taken and information concerning the file is returned in location E+1 through E+3. The normal return is always taken if the device associated with the channel D does not have a directory. The error return is taken if 1) the file is not found, 2) the file is found but the user does not have access to it (refer to Paragraph 6.2.3 for the description of file access codes), or 3) the device associated with channel D is a non-input device. Refer to Appendix E for the error codes returned in bits 18-35 of location E+1.

4.4.2 The ENTER Operator

ENTER selects a file for output on channel D.

```
        ENTER D, E
        error return
        normal return
        :
        :
E:      SIXBIT/file/        ;filename, 1 through 6
                            ;characters, left-justified
        SIXBIT/ext/         ;filename extension, 0 through 3 characters,
                            ;left-justified
          ___               ;The remaining words in the argument block are
          ___               ;ignored for nondirectory devices. Refer to
                            ;Paragraph 6.1.5.2 for the DECtape ENTER
                            ;and Paragraph 6.8.2.1 for the disk ENTER.
```

If no device has been associated with channel D by an INIT or OPEN UUO, the monitor stops the job, prints

I/O TO UNASSIGNED CHANNEL AT USER LOC addr

and returns the user's terminal to monitor mode. The output side of channel D is now closed (if it was not closed); the input side is not affected. On DECtape, ENTER searches the device directory as specified by an INIT. On disk, the user's file directory, as specified by the contents of location E+3, is searched.

If the device does not have a directory, the normal return is always taken. On directory devices, if the file is found and is not being written or renamed, the file is deleted (the user must have access privileges to the file), and the storage space on the device is reclaimed. On DECtape, this deletion must occur immediately on ENTER to ensure that space is available for writing the new version of the

file. On disk, the deletion of the previous version does not occur until output CLOSE time, provided bit 30 of CLOSE is 0 (refer to Paragraph 4.7.7). Consequently, if the new file is aborted when partially written, the old version remains. The normal return is taken, and the monitor makes the file entry, and records file information.

The error return is taken if:

a. The filename in location E is 0.

b. The file is found but is being written or renamed.

c. The user does not have access to the file, as supplied by the file if it exists or by the UFD if the file does not exist.

d. The device associated with channel D is a non-output device.

Refer to Paragraph 6.8.2.1 for details of a disk ENTER and Paragraph 6.1.5.2 for details of a DECtape ENTER. Refer to Appendix E for the error codes returned in bits 18-35 of location E+1.

4.4.3 RENAME Operator

The RENAME (operation code 055) programmed operator is used

a. To alter the filename, filename extension, and file access privileges

b. To delete a file associated with channel D on a directory device

```
        RENAME D,E
        error return
        normal return
        :
        :
E:      SIXBIT/file/        ;filename, 1 to 6 characters
        SIXBIT/ext/         ;filename extension, 0 to 3
                            ;characters.
          ---               ;The remaining words in the
          ---               ;argument block are ignored
                            ;for nondirectory devices.
                            ;Refer to Paragraph 6.1.5.3
                            ;for the DECtape RENAME
                            ;and Paragraph 6.8.2.1 for
                            ;the disk RENAME.
```

If no device has been associated with channel D, the monitor stops the job, prints

I/O TO UNASSIGNED CHANNEL AT USER LOC addr

and returns the user's terminal to monitor mode.

The normal return is given if:

a. The device specified is a nondirectory device.

b. If the filename specified in location E is 0, the file is deleted after all read references are completed.

c. If the filename specified in location E and the filename extension specified in the left half of location E+1 are the same as the current filename and filename extension, the access protection bits are set to the contents of bits 0 to 8 of location E+2.

d. If the filename/filename extension specified differ from the current filename/filename extension, a search is made for the specified filename and filename extension. If a match is not found (1) the filename is changed to the filename in location E, (2) the filename extension is changed to the filename extension in the left half of location E+1, (3) the access protection bits are changed to the contents of bits 0-8 of location E+2, and (4) the access date is unchanged.

The error return is given if:

a. No file is selected on channel D.

b. The specified file is not found.

c. The file is found but is being written, superseded, or renamed.

d. The file is found but the user does not have the privileges to RENAME the file.

e. The filename/filename extension specified differ from the current filename/filename extension, a search is made for the specified filename and filename extension. If a match is found, the error return is taken.

f. The UFD is deleted.

Refer to Appendix E for the error codes returned in bits 18-35 of location E+1. Refer to Paragraph 6.1.5.3 for details of a DECtape RENAME and Paragraph 6.8.2.1 for details of a disk RENAME.

Examples

General Device Initialization

```
INIDEV:  0                          ;JSR HERE
         OPEN 3,OPNBLK              ;CHANNEL 3
           JRST NOTAVL              ;WHERE TO GO IF DTA5 IS BUSY
;FROM HERE DOWN IS OPTIONAL DEPENDING ON THE DEVICE AND PROGRAM
;REQUIREMENTS
         MOVE 0, JOBFF
         MOVEM 0, SVJBFF            ;SAVE THE FIRST ADDRESS OF THE BUFFER
                                    ;RING IN CASE THE SPACE MUST BE
                                    ;RECLAIMED
         INBUF 3,4                  ;SET UP 4 INPUT BUFFERS
         OUTBUF 3,1                 ;SET UP 1 OUTPUT BUFFER
         LOOKUP 3, INNAM            ;INITIALIZE AN INPUT FILE
           JRST NOTFND              ;WHERE TO GO IF THE INPUT FILENAME IS
                                    ;NOT IN THE DIRECTORY
         ENTER 3, OUTNAME           ;INITIALIZE AN OUTPUT FILE
           JRST NOROOM              ;WHERE TO GO IF THERE IS NO ROOM IN
                                    ;THE DIRECTORY FOR A NEW FILENAME
         JRST @INIDEV               ;RETURN TO MAIN SEQUENCE
OPNBLK:  14                         ;BINARY MODE
         SIXBIT/DTA5/               ;DEVICE DECTAPE UNIT 5
         XWD OBUF,IBUF              ;BOTH INPUT AND OUTPUT
OBUF:    BLOCK 3                    ;SPACE FOR OUTPUT BUFFER HEADER
IBUF:    BLOCK 3                    ;SPACE FOR INPUT BUFFER HEADER
```

(continued on next page)

```
INNAM:  SIXBIT/NAME/        ;FILE NAME
        SIXBIT/EXT/         ;FILE NAME EXTENSION (OPTIONALLY 0),
                            ;RIGHT HALF WORD RECEIVES THE
                            ;FIRST BLOCK NUMBER
        Ø                   ;RECEIVES THE DATE
        Ø                   ;UNUSED FOR NONDUMP I/O
OUTNAM: SIXBIT/NAME/        ;SAME INFORMATION AS IN INNAM
        SIXBIT/EXT/
        Ø
        Ø
```

4.5 DATA TRANSMISSION

The programmed operators

INPUT D,E and IN D,E
normal return
error return

transmit data from the file selected on channel D to the user's core area. The programmed operators

OUTPUT D,E and OUT D,E
normal return
error return

transmit data from the user's core area to the file selected on channel D. If specified, E is the effective address of the next buffer to be written. If E is not specified, the next buffer in the sequence is implied.

If no OPEN or INIT operator has been performed on channel D, the monitor stops the job and prints

I/O TO UNASSIGNED CHANNEL AT USER addr

(addr is the location of the IN, INPUT, OUT, or OUTPUT programmed operator) on the user's terminal and the terminal is left in monitor mode. If the device is a multiple-directory device and no file is selected on channel D, bit 18 of the file status is set to 1, and control returns to the user's program. Control always returns to the location immediately following an INPUT (operation code 066) and an OUTPUT (operation code 067). A check of the file status for end-of-file and error conditions must then be made by another programmed operator. Note that to trap on a hardware write-locked device, the user should use location .JBINT (refer to Paragraph 3.1.3.2). Following an INPUT, the user program should check the word count of the next buffer to determine if it contains data. Control returns to the location immediately following an IN (operation code 056) if no end-of-file or error condition exists (i.e., if bits 18 through 22 of the file status are all 0). Control returns to the location immediately following an OUT (operation code 057) if no error condition or end-of-tape exists (i.e., if bits 18 through 21 and bit 25 are all zero). Otherwise, control returns to the second location following the IN or OUT. Note that IN and OUT UUOs are the only ones in which the error return is a skip and the normal return is not a skip.

4.5.1 Unbuffered Data Modes

Data modes 15, 16, and 17 utilize a command list to specify areas in the user's allocated core to be read or written. The effective address E of the IN, INPUT, OUT, and OUTPUT programmed operators point to the first word of the command list. Three types of entries may occur in the command list.

a. IOWD n, loc - Causes n words from loc through loc+n-1 to be transmitted. The next command is obtained from the next location following the IOWD. The assembler pseudo-op IOWD generates XWD -n, loc-1.

b. XWD 0, y - Causes the next command to be taken from location y. Referred to as a GOTO word. Up to three consecutive GOTO words are allowed in the command list. After three consecutive GOTO words, an I/O instruction must be written.

c. 0 - Terminates the command list.

Each IOWD which causes data to be transferred writes a separate record. Thus, for devices other than DECtape, the following two examples produce the same result.

```
1)  OUTPUT  D, [IOWD 100, BUF1
                IOWD 100, BUF2
                Z]
2)  OUTPUT  D, [IOWD 100, BUF1
                Z]
    OUTPUT  D, [IOWD 100, BUF2
                Z]
```

For DECtape (where space is an important consideration), the first example writes one block, and the second writes two.

The monitor does not return program control to the user until the command list has been completely processed. If an illegal address is encountered while processing the list, the job is stopped and the monitor prints

ADDRESS CHECK AT USER addr

on the user's terminal and the terminal is left in monitor mode.

Example: Dump Output

Dump input is similar to dump output. This routine outputs fixed-length records.

```
DMPINI: 0                           ;JSR HERE TO INITIALIZE A FILE
        OPEN 0,0PNBLK               ;CHANNEL 0
          JRST NOTAVL               ;WHERE TO GO IF MTA2 IS BUSY
        JRST @ DMPINI               ;RETURN

DMPOUT: 0                           ;JSR HERE TO OUTPUT THE OUTPUT AREA
        OUTPUT 0,OUTLST             ;SPECIFIES DUMP OUTPUT ACCORDING
                                    ;TO THE LIST AT OUTLST
        STATZ 0, 740000             ;CHECK ERROR BITS
          CALL[SIXBIT /EXIT/]       ;QUIT IF AN ERROR OCCURS
        JRST @DMPOUT                ;RETURN
```

```
DMPDON:  0                          ;JSR HERE TO WRITE AN END OF FILE
         CLOSE 0,                   ;WRITE THE END OF FILE
         STATZ 0, 740000            ;CHECK FOR ERROR DURING WRITE
                                    ;END OF FILE OPERATION
           CALL[SIXBIT /EXIT/]      ;QUIT IF ERROR OCCURS
         RELEAS 0,                  ;RELINQUISH THE DEVICE
         JRST @DMPDON               ;RETURN
OPNBLK:  16                         ;DUMP MODE
         SIXBIT /MTA2/              ;MAGNETIC TAPE UNIT 2
         0                          ;NO RING BUFFERS
OUTLST:  IOWD BUFSIZ,BUFFER         ;SPECIFIES DUMPING A NUMBER OF
                                    ;WORDS EQUAL TO BUFSIZ, STARTING
         0                          ;AT LOCATION BUFFER
                                    ;SPECIFIES THE END OF THE COMMAND
                                    ;LIST
BUFFER:  BLOCK BUFSIZ               ;OUTPUT BUFFER, MUST BE CLEARED
                                    ;AND FILLED BY THE MAIN PROGRAM
```

4.5.2 Buffered Data Modes

In data modes 0, 1, 10, 13, and 14 the effective address E of the INPUT, IN, OUTPUT and OUT programmed operators may be used to alter the normal sequence of buffer reference. If E is 0, the address of the next buffer is obtained from the right half of the second word of the current buffer. If E is non-zero, it is the address of the second word of the next buffer to be referenced. The buffer pointed to by E can be in an entirely separate ring from the present buffer. Once a new buffer location is established, the following buffers are taken from the ring started at E. Since buffer rings are not changed if I/O activity is pending, it is not necessary to issue a WAIT UUO.

4.5.2.1 Input - If no input buffer ring is established when the first INPUT or IN is executed, a 2-buffer ring is set up (refer to Paragraph 4.3.2).

Buffered input may be performed synchronously or asynchronously at the option of the user. If bit 30 of the file status is 1, each INPUT and IN programmed operator performs the following:

(1) Clears the use bit in the second word of the buffer with an address in the right half of the first word of the buffer header, thereby making the buffer available for refilling by the monitor.

(2) Advances to the next buffer by moving the contents of the second word of the current buffer to the right half of the first word of the 3-word buffer header.

(3) Returns control to the user's program if an end-of-file or error condition exists. Otherwise, the monitor starts the device, which fills the buffer and stops transmission.

(4) Computes the number of bytes in the buffer from the number of words in the buffer (right half of the first data word of the buffer) and the byte size, and stores the result in the third word of the buffer header.

(5) Sets the position and address fields of the byte pointer in the second word of the buffer header, so that the first data byte is obtained by an ILDB instruction.

(6) Returns control to the user's program.

Thus, in synchronous mode, the position of a device (e.g., magnetic tape), relative to the current data, is easily determined. The asynchronous input mode differs in that once a device is started, successive buffers in the ring are filled at the interrupt level without stopping transmission until a buffer

whose bit is 1 is encountered. Control returns to the user's program after the first buffer is filled. The position of the device, relative to the data currently being processed by the user's program, depends on the number of buffers in the ring and when the device was last stopped.

Example: General Subroutine to Input One Character

```
;GET -- ROUTINE TO GET ONE BYTE FROM THE INPUT FILE
;       NULLS (Ø) WILL BE DISCARDED
;CALL:  JSP     A,GET
;         END-OF-FILE RETURN
;       RETURN WITH BYTE IN C

GET:    SOSGE   IB+2            ;DECREMENT THE BYTE COUNT
        JRST    GETBF           ;BUFFER EMPTY--GET ANOTHER ONE
        ILDB    C,IB+1          ;SOMETHING THERE--GET IT
        JUMPN   C,1(A)          ;RETURN IF NOT NULL
                                ;** IF NULLS ARE SIGNIFICANT, THIS
                                ;   SHOULD BE A JRST  1(A)
        JRST    GET             ;NULL--LOOP FOR ANOTHER CHARACTER

;HERE WHEN INPUT BUFFER IS EMPTY
;ASK THE MONITOR FOR THE NEXT BUFFER AND JUMP BACK
;RETURN TO USER IF END-OF-FILE

GETBF:  IN      I,              ;GET BUFFER FROM MONITOR
          JRST  GET             ;NO ERRORS OR EOF--JUMP BACK
        GETSTS  I,C             ;GET ERROR STATUS
        TRNN    C,74B23         ;SEE IF ANY ERRORS
        JRST    GETBFE          ;NO--GO CHECK EOF
                        ;** INSERT ERROR ROUTINE HERE
                        ;   FOR EXAMPLE, TYPE C IN OCTAL
                        ;   WITH MESSAGE GIVING FILE NAME, ETC.
        TRZ     C,74B23         ;CLEAR ERROR BITS
        SETSTS  I,(C)           ;TELL MONITOR

GETBFE: TRNE    C,1B22          ;SEE IF END OF FILE
        JRST    (A)             ;YES--GIVE NON-SKIP RETURN
        JRST    GET             ;NO--JUMP BACK TO PROCESS DATA
```

4.5.2.2 Output - If no output buffer ring has been established (i.e., if the first word of the buffer header is 0), when the first OUT or OUTPUT is executed, a 2-buffer ring is set up (refer to Paragraph 4.3.2). If the ring use bit (bit 0 of the first word of the buffer header) is 1, it is set to 0, the current buffer is cleared to all 0s, and the position and address fields of the buffer byte pointer (the second word of the buffer header) are set so that the first byte is properly stored by an IDPB instruction. The byte count (the third word of the buffer header) is set to the maximum of bytes that may be stored in the buffer, and control is returned to the user's program. Thus, the first OUT or OUTPUT initializes the buffer header and the first buffer, but does not result in data transmission.

If the ring use bit is 0 and the bit 31 of the file status is 0, the number of words in the buffer is computed from the address field of the buffer byte pointer (the second word of the buffer header) and the buffer pointer (the first word of the buffer header), and the result is stored in the right half of the third

word of the buffer. If bit 31 of the file status is 1, it is assumed that the user has already set the word count in the right half of the third word. The buffer use bit (bit 0 of the second word of the buffer) is set to 1, indicating that the buffer contains data to be transmitted to the device. If the device is not currently active (i.e., not receiving data), it is started. The buffer header is advanced to the next buffer by setting the buffer pointer in the first word of the buffer header. If the buffer use bit of the new buffer is 1, the job is put into a wait state until the buffer is emptied at the interrupt level. The buffer is then cleared to 0s, the buffer byte pointer and byte count are initialized in the buffer header, and control is returned to the user's program.

Example: General Subroutine to Output One Character

```
;PUT -- ROUTINE TO PUT ONE BYTE INTO THE OUTPUT FILE
;CALL:  MOVE    C,BYTE
;       JSP     A,PUT
;       RETURN

PUT:    SOSG    OB+2            ;ADVANCE BYTE COUNTER
        JRST    PUTBF           ;JUMP IF BUFFER FULL (OR FIRST CALL)
PUTC:   IDPB    C,OB+1          ;PUT BYTE INTO BUFFER
        JRST    (A)             ;RETURN TO CALLER

;JUMP HERE WHEN BUFFER IS FULL AND THE NEXT ONE IS NEEDED
;GIVE THE MONITOR THE BUFFER AND JUMP BACK

PUTBF:  OUT     0,              ;GIVE BUFFER TO MONITOR
          JRST  PUTC            ;NO ERRORS--JUMP BACK
        MOVEM   C,SAVEC#        ;ERROR--SAVE AC FOR STATUS CHECKING
        GETSTS  0,C             ;GET ERROR STATUS
                        ;** INSERT OUTPUT ERROR ROUTINE HERE
                        ;   FOR EXAMPLE, TYPE C IN OCTAL
                        ;   WITH MESSAGE GIVING FILE NAME, ETC.
        TRZ     C,74823         ;CLEAR ERROR BITS
        SETSTS  0,(C)           ;TELL MONITOR
        MOVE    C,SAVEC         ;RESTORE CHARACTER
        JRST    PUTC            ;JUMP BACK TO PROCESS CHARACTER
```

4.5.3 Synchronization of Buffered I/O

In some instances, such as recovery from transmission errors, it is desirable to delay until a device completes its I/O activities. The programmed operator

WAIT D, or CALLI D, 10

returns control to the user's program when all data transfers on channel D have finished. This UUO does not wait for a magnetic tape spacing operation, since no data transfer is in progress. An MTAPE D, 0 (refer to Paragraph 5.5.3) should be used to wait for the magnetic tape controller to be freed after completing spacing and I/O activity on magnetic tape. In addition, the UUO does not wait for physical I/O to the terminal to be completed; it waits only until the user's buffer is empty. Therefore, the usual motive for the WAIT UUO, error recovery, does not apply to the terminal. If no device is associated with data channel D, control returns immediately. After the device is stopped, the position of the device relative to the data currently being processed by the user's program can be determined by the buffer use bits.

4.6 STATUS CHECKING AND SETTING

The file status is a set of 18 bits (right-half word), which reflects the current state of a file transmission. The initial status is a parameter of the INIT and OPEN operators. Thereafter, bits are set by the monitor, and may be tested and reset by the user via the STATZ, STATO, and SETSTS monitor programmed operators. Table 4-3 defines the file status bits. All bits, except the end-of-file bit, are set immediately by the monitor as the conditions occur, rather than being associated with the buffer currently being used. However, the file status is stored with each buffer so that the user can determine which bufferful produced an error. The end-of-file bit is set when the user attempts to read past the last block of data (i.e., it is set on an IN or INPUT UUO for which there is no corresponding data;

Table 4-3
File Status Bits

Bit	Meaning
18	Improper mode (IO.IMP). Attempt to write on a software write-locked tape or file structure, or a software detected redundancy failure occurred. Usually set by monitor.
19	Hard device detected error (IO.DER), other than data parity error. This is a search, power supply, or channel memory parity error. The device is in error rather than the data on the medium. However, the data read into core or written on the device is probably incorrect. Usually set by monitor.
20	Hard data error (IO.DTE). The data read or written has incorrect parity as detected by hardware (or by software on CDR, PTR). The user's data is probably non-recoverable even after the device is fixed. Usually set by monitor.
21	Block too large (IO.BKT). A block of data from a device is too large to fit in a buffer; a block number is too large for the unit; the file structure (DSK) or unit (DTA) has filled; or the user's quota on the file structure has been exceeded. Usually set by monitor.
22	End of file (IO.EOF). The user program has requested data beyond the last record or block with an IN or INPUT UUO, or USETI has specified a block beyond the last data block of the file. When set, no data has been read into the input buffer. Usually set by monitor.
23	I/O active (IO.ACT). The device is actively transmitting or receiving data. Always set by monitor.
24-29	Device dependent parameters. Refer to Chapters 5 and 6 and Appendix D for detailed information about each device. Usually set by user.
30	Synchronous input (IO.SYN). Stops the device after each buffer is filled. Usually set by user.
31	User word count (IO.UWC). Forces the monitor to use the word count in the third word of the buffer (output only). The monitor normally computes the word count from the byte pointer in the buffer header. Usually set by user.
32-35	Data mode (IO.MOD). Refer to Tables 4-1 and 4-2. Usually set by user.

the previous IN or INPUT UUO obtained the end of the data). Therefore, when this bit is set, no data has been placed in the input buffer.

The programmed operators discussed in this section are the software equivalents of the hardwired instructions CONO, CONI, CONSO, and CONSZ. A more thorough description of bits 18 through 29 for each device is given in Chapters 5 and 6 and in Appendix D.

4.6.1 File Status Checking

The file status (refer to Table 4-3) is retrieved by the GETSTS (operation code 062) and tested by the STATZ (operation code 063) and STATO (operation code 061) programmed operators. In each case, the accumulator field of the instruction selects a data channel. If no device is associated with the specified data channel, the monitor stops the job and prints

I/O TO UNASSIGNED CHANNEL AT USER addr

(addr is the location of the GETSTS, STATZ, or STATO programmed operator) on the user's terminal and the terminal is left in monitor mode.

GETSTS D,E stores the file status of data channel D in the right half and 0 in the left half of location E.

STATZ D,E skips, if all file status bits selected by the effective address E are 0.

STATO D,E skips, if any file status bit selected by the effective address E is 1.

4.6.2 File Status Setting

The initial file status is a parameter of the INIT and OPEN programmed operators; however, the file status may be changed by the SETSTS (operation code 060) programmed operator. Error status bits IO.ERR (IO.IMP, IO.DER, IO.DTE, and IO.BKT) must be cleared by this programmed operator if the user is attempting an error recovery. In addition, the SETSTS UUO can be used to clear the end-of-file bit, but this is not sufficient to clear the end-of-file condition. Further inputs will not occur until the end-of-file condition (determined by an internal monitor flag IOEND) is cleared by a CLOSE or INIT UUO.

SETSTS D,E waits until the device on channel D stops transmitting data and replaces the current file status, except bit 23, with the effective address E. If the new data mode, indicated in the right four bits of E, is not legal for the device, the job is stopped and the monitor prints

ILL DEVICE DATA MODE FOR DEVICE dev AT USER addr

(dev is the physical name of the device and addr is the location of the SETSTS operator) on the user's terminal and the terminal is left in monitor mode. If the user program changes the data mode, it must

also change the byte size for the byte pointer in the input buffer header (if any) and the byte size and item count in the output buffer header (if any). The output item count should be changed by using the count already placed there by the monitor and dividing or multiplying by the appropriate conversion factor, rather than assuming the length of a buffer. Incorrect I/O may result if a data mode change requires a different buffer length. SETSTS does not change buffer lengths. The mode specified in the INIT is used to determine buffer sizes even though the buffer ring has not been created.

4.7 FILE TERMINATION

File transmission is terminated by the CLOSE D,N (operation code 070) programmed operator. N is usually zero, but individual options may be selected independently to control the effect of the CLOSE.

Usually a given channel is OPEN for file transmission in only one direction, and CLOSE has the effect of either closing input if INPUTs have been done or closing output if OUTPUTs have been done. However, disk and DECtape may have a single channel OPEN for both INPUT and OUTPUT, in which case the first two options below are useful.

4.7.1 CLOSE D,0

The output side of channel D is closed (bit 35=0). In unbuffered data modes, the effect is to execute a device dependent function. In buffered data modes, if a buffer ring exists, the following operations are performed:

a. All data in the buffers that has not been transmitted to the device is written.
b. Device dependent functions are performed.
c. The ring use bit (bit 0 of the first word of the buffer header) is set to 1 indicating that the buffer ring is available.
d. The buffer byte count (the third word of the buffer header) is set to 0.
e. Control returns to the user program when transmission is complete.

The input side of channel D is also closed (bit 34=0). The end-of-file flag is always cleared. Further action depends on the data mode in unbuffered data modes, the effect is to execute a device dependent function. In buffered data modes, if a ring buffer exists, the following operations are performed:

a. Wait until device is inactive.
b. The use bit of each buffer (bit 0 of the second word) is cleared indicating that the buffer is empty.
c. The ring use bit of the buffer header (bit 0 of the first word of the buffer header) is set to 1 indicating that the buffer ring is available.
d. The buffer byte count (the third word of the buffer header) is set to 0.
e. Control returns to the user program.

On output CLOSE, the unwritten blocks at the end of a disk file are automatically deallocated (bit 33=0). On input CLOSE, the access date of a disk file is updated (bit 32=0).

4.7.2 CLOSE D,1 (Bit 35=1, CL.OUT)

The closing of the output side of channel D is suppressed. Other actions of CLOSE are unaffected.

4.7.3 CLOSE D,2 (Bit 34=1, CL.IN)

The closing of the input side of channel D is inhibited; other actions of CLOSE are unaffected.

4.7.4 CLOSE D,4 (Bit 33=1, CL.DLL) †

The unwritten blocks at the end of a disk file are not deallocated. This capability is provided for users who specifically allocate disk space and wish to retain it.

4.7.5 CLOSE D,10 (Bit 32=1, CL.ACS) †

The updating of the access date on CLOSE input is inhibited. This capability is intended for use with FAILSAFE, so that files can be saved on magnetic tape without causing the disk copy to appear as if it has been accessed.

4.7.6 CLOSE D,20 (Bit 31=1, CL.NMB) †

The deleting of the NAME block and the access tables in monitor core on CLOSE input is inhibited if a LOOKUP was done without subsequent INPUT. This bit is used by the COMPIL program to retain the core block in order to speed up the subsequent access by the system program called by COMPIL.

4.7.7 CLOSE D,40 (Bit 30=1, CL.RST) †

The deleting of the original file, if any, is inhibited if an ENTER which creates or supersedes was done. The new copy of the file is discarded. This bit is used by the queue manager (QMANGR) to create a file or a unique name and not supersede the original file.

4.7.8 CLOSE D,100 (Bit 29=1,CL.DAT) †

The NAME block and access tables are deleted from the disk data base and the space is returned to free core.

---

† Meaningful with disk files only, ignored with non-disk files.

Any combinations of the above bit settings are legal.

Example: Terminating a File

```
DROPDV:  0                          ;JSR HERE
         CLOSE 3,                   ;WRITE END OF FILE AND TERMINATE
                                    ;INPUT
         STATZ 3, 740000            ;RECHECK FINAL ERROR BITS
           JRST OUTERR              ;ERROR DURING CLOSE
         RELEAS 3,                  ;RELINQUISH THE USE OF THE
                                    ;DEVICE, WRITE OUT THE DIRECTORY
         MOVE 0, SVJBFF
         MOVEM 0, JOBFF             ;RECLAIM THE BUFFER SPACE
         JRST @ DROPDV              ;RETURN TO MAIN SEQUENCE
```

4.8 DEVICE TERMINATION AND REASSIGNMENT

4.8.1 RELEASE

When all transmission between the user's program and a device is finished, the program must relinquish the device by performing a

RELEASE D,

RELEASE (operation code 071) returns control immediately, if no device is associated with data channel D. Otherwise, both input and output sides of data channel D are CLOSEd and the correspondence between channel D and the device, which was established by the INIT or OPEN programmed operators, is terminated. Any errors that occurred are recorded in the BAT block if a super USETI/USETO was used with channel D. If the device is neither associated with another data channel nor assigned by the ASSIGN or MOUNT monitor command, it is returned to the monitor's pool of available facilities. Control is returned to the user's program.

4.8.2 RESDV. AC, or CALLI AC, 117

This UUO allows a user program to reset a single channel. It is similar to the RELEASE UUO except any files and buffers are not closed. Files that are open on the channel are deleted; any older version with the same filename remains. All I/O transmissions on the channel are stopped, and device allocations made by the INIT or OPEN UUOs on the specified channel are cleared. The device is returned to the monitor pool unless it has been assigned by the ASSIGN or MOUNT monitor command. The call is:

```
MOVEI AC, channel number
RESDV. AC,                  ; or CALLI AC, 117
error return
normal return
```

On an error return, either the AC is unchanged if the UUO is not implemented, or AC contains -1 if there is no device associated with the channel.

On a normal return, the channel is reset.

4.8.3 REASSIGN AC, or CALLI AC, 21

This UUO reassigns a device under program control to the specified job and clears the directory currently in core, but does not clear the logical name assignment. A device can be reassigned if it is assigned to the current job, or if it is both not assigned to any job and is not detached. A RELEASE UUO is performed unless the job issuing the UUO is reassigning the device to itself by specifying -1 in AC or is deassigning the device by specifying 0 in AC. If the device is restricted when it is deassigned with a 0 in AC, it is returned to the restricted pool of devices and can be reassigned to a non-privileged job by a privileged job. (This is the method by which the MOUNT command is implemented.)

The call is:

```
MOVE AC, job number
MOVE AC+1, [SIXBIT /DEVICE/]      ;or MOVEI AC+1, channel number
REASSIGN AC,                      ;or CALLI AC, 21
return                            ;error and normal
```

If on return the contents of AC = 0, the specified job has not been initialized. If the contents of AC+1=0, the device has not been assigned to the new job, the device is the job's controlling terminal, the logical name is duplicated, or the logical name is a physical name in the system and the job reassigning the device is either logged in under a different project-programmer number or is not the operator.

4.8.4 DEVLNM AC, or CALLI AC, 107[1]

This UUO sets the logical name for the specified device. Upon call of the UUO, AC contains either the device name or the channel number associated with the device. The call is:

```
MOVE AC, [SIXBIT /dev/]           ;or MOVEI AC, channel no.
MOVE AC+1, [SIXBIT /log.name/]
DEVLNM AC,                        ;or CALLI AC, 107
error return
normal return
```

On an error return, AC contains one of the following:

AC = unchanged if the UUO is not implemented.
AC = -1 if a non-existent device or channel number was specified.
AC = -2 if the logical name is already in use.
AC = -3 if device is neither assigned by a console command (ASSIGN, MOUNT) nor by the program (INIT, OPEN).

On a normal return, AC and AC+1 are unchanged.

[1] This UUO depends on FT5UUO which is normally off in the DECsystem-1040.

4.9 EXAMPLES

4.9.1 File Reading

The following UUO sequence is required to read a file:

OPEN	Establishes a file structure-channel correspondence (or a set of file structure channel correspondences).
LOOKUP	Establishes a file-channel correspondence. Invokes a search of the UFD. Returns information from the file system.
INBUF	(Optional) Sets up 1 to N ring buffers in the top of core, expand core if necessary.
INPUT	Sets up 2-buffer ring if no INBUF was done.
•	
•	
•	
INPUT	Requests buffers of data from the monitor.
CLOSE	Breaks file-channel correspondence.
RELEASE	Breaks device-channel correspondence.

4.9.2 File Writing

The following UUO sequence is required to write a file:

OPEN	Forms file structure-channel correspondence (or a set of file structure channel correspondences).
ENTER	Forms file-channel correspondence. The monitor creates some temporary storage for interlocking and shared access purpose for the filename. No directory entry is made.
OUTPUT	
•	
•	
•	
OUTPUT	Passes buffers of data to monitor for transmission to storage device. Should not be used for the final buffer because CLOSE completes the action of ENTER.
CLOSE	Completes the action of ENTER. Adds filename to file system. Normally returns allocated, but unused, blocks to the file system.
RELEASE	Breaks device-channel correspondence.

4.9.3 File Reading/Writing

```
          TITLE     FILTRN -- SAMPLE I/O PROGRAM

;A PROGRAM THAT READS 7-BIT ASCII CHARS FROM FILE INFILE.DAT
;ON DEVICE DATA AND OUTPUTS THEM TO FILE OUTFIL.LST ON DEVICE LIST
;NOTE THAT DEVICES DATA AND LIST ARE LOGICAL NAMES.  THUS
;THE PHYSICAL NAMES ARE DETERMINED AT RUN TIME TO PROVIDE DEVICE
;INDEPENDENCE.
;BOTH INPUT AND OUTPUT FILES ARE ACCESSED SEQUENTIALLY.

START:  RESET                          ;DEVICE RESET (IN CASE PROGRAM
                                       ;  IS RESTARTED)
        OPEN    1,[     1              ;CONNECT DEVICE DATA TO PROG ON CH 1
                   SIXBIT /DATA/
                   XWD 0,IBUF1]        ;INBUF1 IS THE INPUT BUFFER HEADER
          HALT  .                      ;ERROR RETURN
        OPEN    2,[     1              ;CONNECT DEVICE LIST TO CH 2
                   SIXBIT /LIST/
                   XWD OBUF2,0]        ;OBUF2 IS OUTPUT BUFFER HEADER
          HALT  .
        LOOKUP  1,L1                   ;OPEN FILE INFILE.DAT FOR INPUT
          HALT  .                      ;ERROR RETURN
        ENTER   2,E2                   ;OPEN FILE OUTFIL.LST FOR OUTPUT
          HALT  .
        INBUF   1,3                    ;CREATE 3 INPUT BUFFERS
                                       ;SINCE NO BUFFERS SPECIFIED FOR OUTPUT
                                       ;  ON FIRST OUTPUT THE MONITOR WILL
                                       ;  MAKE 2

;THIS IS THE BASIC I/O LOOP FOR THE JOB

NEWCHR: JSR     GET                    ;GO GET ONE INPUT CHARACTER
        JSR     PUT                    ;GO PUT THE CHARACTER RECEIVED
        JRST    NEWCHR                 ;LOOP FOR NEXT ONE

;GET -- ROUTINE TO GET ONE CHARACTER FROM THE INPUT
;IT ENDS THE PROGRAM AT INPUT END-OF-FILE

GET:    Z                              ;ENTRY/EXIT
GET1:   SOSGE   IBUF1+2                ;IS INPUT BUFFER EMPTY?
        JRST    GETBF                  ;YES--INPUT FROM DEVICE
        ILDB    3,IBUF1+1              ;GET A CHARACTER FROM INPUT BUFFER
        JUMPE   3,GET1                 ;IF NULL, THROW IT AWAY AND GET NEXT
                                       ;  CHARACTER.  THIS IS CONVENTIONAL FOR
                                       ;  ASCII DATA.
        JRST    @GET                   ;RETURN WITH CHARACTER IN AC 3

GETBF:  IN      1,                     ;DO INPUT FROM DEVICE
          JRST  GET1                   ;LOOP IF NO ERRORS AND NOT EOF
        STATZ   1,74B23                ;SEE IF ERROR READING
          HALT  .                      ;YES--GIVE UP

FINISH: CLOSE   1,                     ;EOF--CLOSE INPUT
        CLOSE   2,                     ;CLOSE OUTPUT
        RELEAS  1,                     ;RELEASE DEVICE DATA
        RELEAS  2,                     ;RELEASE DEVICE LIST
        EXIT                           ;EXIT TO MONITOR
```

(continued on next page)

```
;PUT--ROUTINE TO PUT ONE CHARACTER ONTO THE OUTPUT

PUT:    Z                       ;ENTRY/EXIT
        SOSG    OBUF2+2         ;IS OUTPUT BUFFER FULL?
        JRST    PUTBF           ;YES--GO OUTPUT IT
PUTC:   IDPB    3,OBUF2+1       ;PUT CHARACTER IN BUFFER
        JRST    @PUT            ;RETURN

PUTBF:  OUT     2,              ;OUTPUT BUFFER TO DEVICE
          JRST  PUTC            ;OK, NOW STORE CHARACTER IN BUFFER
        HALT                    ;GIVE UP IF OUTPUT ERROR

;DATA STORAGE AREA

L1:     SIXBIT /INFILE/         ;INPUT FILE NAME
        SIXBIT /DAT/            ;INPUT EXTENSION
        Z                       ;PROTECTION AND CREATION DATE RETURNED
        Z                       ;INPUT DIRECTORY.  Ø MEANS MY OWN

E2:     SIXBIT /OUTFIL/         ;OUTPUT FILE NAME
        SIXBIT /LST/            ;OUTPUT EXTENSION
        Z                       ;PROTECTION CAN GO HERE.  Ø MEANS STD.
        Z                       ;OUTPUT DIRECTORY.  Ø MEANS MY OWN

IBUF1:  BLOCK   3               ;INPUT BUFFER HEADER
OBUF2:  BLOCK   3               ;OUTPUT BUFFER HEADER

        END     START
```

4.10 DEVICE INFORMATION

4.10.1 DEVSTS AC, or CALLI AC, 54[1]

This UUO retrieves the DEVSTS word of the device data block for an INITed device. The DEVSTS word is used by a device service routine to save the results of a CONI after each interrupt from the device. Refer to Appendix D for the device status bits. Devices that use the DEVSTS UUO are the following: CDR, CDP, MTA, DTA, PTR, PTP, DSK, LPT, and PLT.

The call is:

```
MOVEI AC, channel number of device    ;or MOVE AC, [SIXBIT /dev/]
DEVSTS AC,                            ;or CALLI AC, 54
error return                          ;UUO not implemented for any devices
normal return                         ;AC contains the DEVSTS
                                      ;word of the DDB.
```

On return, the contents of the DEVSTS word is returned in AC. Therefore, if the device service routine does not store a CONI, useless information may be returned to user. Note that an error return is not indicated if the device service routine does not use the DEVSTS word for its intended purpose. Devices with both a control and data interrupt store the controller CONI (MTS, DTS, DSK, DSK2, DPC, DPC2).

---

[1] This UUO depends on FT5UUO which is normally off in the DECsystem-1040.

The DEVSTS UUO is not meaningful when used in asynchronous buffered I/O mode unless a WAIT UUO (see Paragraph 4.5.3) is issued first to ensure synchronization of the actual data transferred with the device status returned.

4.10.2 DEVCHR AC, or CALLI AC, 4

This UUO allows the user to determine the physical characteristics associated with a device name. When the UUO is called, AC must contain either the logical or physical device name as a left-justified SIXBIT quantity, or the channel number of the device as a right-justified quantity.

The call is:

```
MOVE AC, [SIXBIT/DEV/]          ;or MOVEI AC, channel number of device
DEVCHR AC,                      ;or CALLI AC,4
return
```

If the device is not found or the channel is not INITed, the contents of AC is zero on return. If the device is found, the following information is returned in AC.

Name	Bit	Explanation
DV.DRI	Bit 0 = 1	DECtape directory is in core. This bit is cleared by an ASSIGN or DEASSIGN to that unit.
DV.DSK	Bit 1 = 1	Device is a disk.
DV.CDR	Bit 2 = 1	Device is a card reader (DV.IN = 1) or card punch (DV.OUT = 1).
DV.LPT	Bit 3 = 1	Device is a line printer.
DV.TTA	Bit 4 = 1	TTY is controlling a job.
DV.TTU	Bit 5 = 1	TTY is in use as a user terminal (even if detached).
DV.TTB	Bit 6 = 1	Free bit left from SCNSRF.
DV.DIS	Bit 7 = 1	Device is a display.
DV.LNG	Bit 8 = 1	Device has a long dispatch table (that is, UUOs other than INPUT, OUTPUT, CLOSE, and RELEASE perform real actions).
DV.PTP	Bit 9 = 1	Device is a paper-tape punch.
DV.PTR	Bit 10 = 1	Device is a paper-tape reader.
DV.DTA	Bit 11 = 1	Device is a DECtape.
DV.AVL	Bit 12 = 1	Device is available to this job or is already assigned to this job.
DV.MTA	Bit 13 = 1	Device is a magnetic tape.
DV.TTY	Bit 14 = 1	Device is a TTY.
DV.DIR	Bit 15 = 1	Device has a directory (DTA or DSK).

(continued on next page)

Name	Bit	Explanation
DV.IN	Bit 16 = 1	Device can perform input.
DV.OUT	Bit 17 = 1	Device can perform output.
DV.ASC	Bit 18 = 1	Device is assigned by a console command.
DV.ASP	Bit 19 = 1	Device is assigned by program (INIT or OPEN).
	Remaining bits	If bit 35-n contains a 1, then mode n is legal for that device. The mode number (0 through 17) must be converted to decimal (e.g., mode 17_8 is represented by bit $35-15_{10}$ or bit 20).

4.10.3 DEVTYP AC, or CALLI AC, 53

The device-type UUO is used to determine properties of devices. This UUO accepts, as an argument, a device name in SIXBIT or a right-justified channel number. The call is:

```
MOVE AC, [SIXBIT/dev/]          ;or MOVEI AC, channel no.
DEVTYP AC,                      ;or CALLI AC, 53
error return
normal return
```

The error return is given if the UUO is not implemented. In this case, the DEVCHR UUO should be used. On a normal return, if AC=0, the specified device does not exist or the channel is not INITed. If the device exists, the following information is returned in AC.

Name	Bit	Explanation
TY.MAN	Bit 0 = 1	LOOKUP/ENTER mandatory.
	Bits 1-11	Reserved for the future.
TY.AVL	Bit 12 = 1	Device is available to this job.
TY.SPL	Bit 13 = 1	Spooled on disk. (Other bits reflect properties of real device, except variable buffer size.)
TY.INT	Bit 14 = 1	Interactive device (output after each break character).
TY.VAR	Bit 15 = 1	Capable of variable buffer size (user can set his own buffer lengths).
TY.IN	Bit 16 = 1	Capable of input.
TY.OUT	Bit 17 = 1	Capable of output.
TY.JOB	Bits 18-26	Job number that currently has device INITed or ASSIGNed.

(continued on next page)

Name	Bit	Explanation
	Bits 27-28	Reserved for the future.
TY .RAS	Bit 29	Device is a restricted device (i.e., can be assigned only by a privileged job or the MOUNT command).
TY .DEV	Bits 30-35	Device type code.

Code 0 (.TYDSK)	Disk of some sort
Code 1 (.TYDTA)	DECtape
Code 2 (.TYMTA)	Magnetic tape
Code 3 (.TYTTY)	TTY or equivalent
Code 4 (.TYPTR)	Paper-tape reader
Code 5 (.TYPTP)	Paper-tape punch
Code 6 (.TYDIS)	Display
Code 7 (.TYLPT)	Line printer
Code 10 (.TYCDR)	Card reader
Code 11 (.TYCDP)	Card punch
Code 12 (.TYPTY)	Pseudo-TTY
Code 13 (.TYPLT)	Plotter
Code 14-57	Reserved for Digital
Code 60-77	Reserved for customer

4.10.4 DEVSIZ AC, or CALLI AC, 101

This UUO is used to determine the buffer size for a device if the user wants to allocate core himself. The call is:

```
        MOVE AC, [EXP LOC]
        DEVSIZ AC,                      ;or CALLI AC, 101
        error return
        normal return

LOC:    EXP STATUS                      ;first word of the OPEN block
LOC+1:  SIXBIT /dev/                    ;second word of the OPEN block
```

The error return is given if the UUO is not implemented. On a normal return, AC contains one of the following values:

If the mode is illegal, AC contains -2.

If the device does not exist, AC contains -1.

If the device exists, but its data mode is dump mode, AC contains 0.

If the device exists and the data mode is legal, AC contains in bits 0-17 the default number of buffers, and in bits 18-35 the default buffer size (including the first three words of the buffer).

4.10.5 WHERE AC, or CALLI AC, 63[1]

This UUO returns the physical station number of the specified device. When the UUO is called, AC contains either the channel number of the device as a right-justified quantity, or the device name as a left-justified SIXBIT quantity. The call is:

```
MOVE AC, [SIXBIT /dev/]          ;or MOVEI AC, channel no.
WHERE AC,                        ;or CALLI AC, 63
error return
normal return
```

If OPR is specified as the device name, the station number at which the job is logically located is returned; if OPR0 is specified, the station number of the central station is returned; and if TTY is specified, the station number at which the job's TTY is located is returned.

On a normal return, the LH of AC contains the station's status, and the RH of AC contains the station number associated with the device. The station's status is represented by the following bits:

Bit 13 = 1 if the station is dial-up (RM.SDU).
Bit 14 = 1 if the station is loaded (.RMSUL).
Bit 15 = 1 if the station is in the loading procedure (.RMSUG).
Bit 16 = 1 if the station is down (.RMSUD).
Bit 17 = 1 if the station is not in contact (.RMSUN).

The error return is taken if the UUO is not implemented, the specified channel is not INITed, or the requested device does not exist.

4.10.6 DEVNAM AC, or CALLI AC, 64

This UUO returns the physical name of a device obtained through either a generic INIT/OPEN or a logical device assignment. When the UUO is called, AC contains either channel number of the device as a right-justified quantity, or the device name as a left-justified SIXBIT quantity. The call is:

```
MOVE AC, [SIXBIT /dev/]          ;or MOVEI AC, channel no.
DEVNAM AC,                       ;or CALLI AC, 64
error return
normal return
```

The normal return is taken if the specified device is found, and AC contains the SIXBIT physical device name.

The error return is taken if the UUO is not implemented (AC is unchanged), the specified channel is not INITed, or no such device exists.

---

[1] This UUO depends on FTREM which is normally off in the DECsystem-1040.

CHAPTER 5
I/O PROGRAMMING FOR NONDIRECTORY DEVICES

This chapter explains the unique features of each standard nondirectory I/O device. Each device accepts the programmed operators explained in Chapter 4, unless otherwise indicated. Table 5-1 is a summary of the characteristics of all nondirectory devices. Buffer sizes are given in octal and include three bookkeeping words. The user may determine the physical characteristics associated with a logical device name by calling the DEVCHR UUO (refer to Paragraph 4.10.2).

Table 5-1
Nondirectory Devices

Device	Physical Name	Controller Number	Unit Number	Programmed Operators	Data Modes	Buffer Size (Octal)†
Card Punch	CDP	-	CP10A	OUTPUT, OUT	A, AL, I, IB, B	35
Card Reader	CDR, CDR1	-	CR10A 461 (PDP-6)	INPUT, IN	A, AL, I, IB, B, SI	36
Console Terminal	CTY	-	LT33A, LT33B LT35A, LT37AC 626 (PDP-6)	INPUT, IN OUTPUT, OUT	A, AL, I	23
Display	DIS	-	VR30, VP10 340B, 30	INPUT, OUTPUT	ID	Dump only
Line Printer	LPT, LPT1		LP10C	OUTPUT	A, AL, I	37
Magnetic Tape	MTA0, MTA1, ..., MTA7	TM10A TM10B 516(PDP-6)	TU20A, TU20B TU30A, TU30B	INPUT, IN OUTPUT, OUT MTAPE	A, AL, I IB, B DR, D	203††
Paper-Tape Punch	PTP	-	PC09 761(PDP-6)	OUTPUT, OUT	A, AL, I IB, B	43
Paper-Tape Reader	PTR	-	PC09 760(PDP-6)	INPUT, IN	A, AL, I IB, B	43

†Buffer sizes are subject to change and should be calculated rather than assumed by user programs. A DEVSIZ UUO may be employed.

††The buffer size for magnetic tape may be changed with the SET BLOCKSIZE monitor command (refer to the DECsystem-10 Operating System Commands).

Table 5-1 (Cont)
Nondirectory Devices

Device	Physical Name	Controller Number	Unit Number	Programmed Operators	Data Modes	Buffer Size (Octal)†
Plotter	PLT	XY10	XY10A XY10B	OUTPUT, OUT	A, AL, I IB, B	46
Pseudo-TTY	PTY	-	-	INPUT, IN OUTPUT, OUT	A, AL	23
Terminal	TTY0, TTY1, ..., TTY177	DC10 DC68A 630(PDP-6)	LT33A, LT33B LT35A, LT37AC VT06	INPUT, IN OUTPUT, OUT TTCALL	A, AL, I	23

†Buffer sizes are subject to change and should be calculated rather than assumed by user programs. A DEVSIZ UUO may be employed.

5.1 CARD PUNCH

The device mnemonic is CDP; the buffer size is dependent on the data mode.

Data Mode	Buffer Size
A, AL	23_8 (20_8 data) words - 80 7-bit ASCII characters
I, IB	36_8 (33_8 data) words - 80 12-bit bytes
B	35_8 (32_8 data, 33_8 punched) words - 26 data words, word count and checksum punched.

5.1.1 Concepts

The header card is the first card of an ASCII file and identifies the card code used (refer to Appendix C). This card is not punched for data modes other than ASCII. The header card has the same punches in all columns. This punch depends on the card code used; for example, in DEC026, the header card has 12-2-4-8 punched in columns 1-80.

The end-of-file (EOF) card is the last card of each output file. This card is punched for all data modes. The end-of-file card has a 12-11-0-1-6-7-8-9 punch in columns 1 through 80.

5.1.2 Data Modes

5.1.2.1 ASCII, Octal Code 0 - ASCII characters are converted to card codes and punched (up to 80 characters per card). Tabs are simulated by punching from 1 to 8 blank columns; form-feeds and carriage returns are ignored.

Line-feeds cause a card to be punched. All other nontranslatable ASCII characters cause a question mark to be punched. Cards can be split between buffers. Attempting to punch more than 80 columns per card causes the error bit IO.BKT (bit 21 of status word) to be set. The CLOSE will punch the last partial card and then punch an EOF card.

Cards are normally punched with DEC026 card codes. If bit 29 (octal 100) of the status word is on (from INIT, OPEN, or SETSTS), cards are punched with DEC029 codes (refer to Appendix C). The first card of any file (the header card) indicates the card code used (12-0-2-4-6-8 punched in column 1 for DEC029 card codes; 12-2-4-8 punched in column 1 for DEC026 card codes).

5.1.2.2 ASCII Line, Octal Code 1 - The same as ASCII mode.

5.1.2.3 Image, Octal Code 10 - In image mode, each buffer contains 27 words, each of which contains three 12-bit bytes. Each byte corresponds to one card column. Since there is room for 81 columns in the buffer (3 x 27) and there are only 80 columns on a card, the last word contains only 2 bytes of data; the third byte is thrown away. If the byte size is set by the program to be 12-bit bytes (the monitor normally sets 36-bit bytes), the program must skip the last byte in the buffer. Image binary causes exactly one card to be punched for each output. A program should not force an output every 80 columns since, if the program is in spooled mode, it will waste a large amount of disk space. The CLOSE punches the last partial card and then punches an EOF card.

5.1.2.4 Image Binary, Octal Code 13 - Same as Image.

5.1.2.5 Binary, Octal Code 14 - Column 1 contains the word count in rows 12-3. A 7-9 punch is in column 1. Column 2 contains a checksum as described for the paper-tape reader (refer to Paragraph 5.7.1.5); columns 3 through 80 contain up to 26 data words, 3 columns per word. Binary causes exactly one card to be punched for each output. The CLOSE punches the last partial card and then punches an EOF card.

5.1.3 Special Programmed Operator Service

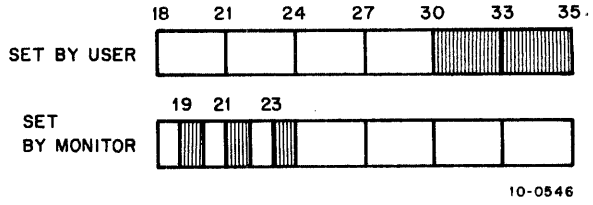
Following a CLOSE, an EOF card is punched. Columns 2 through 80 of the header card and the EOF card contain the same punches that appear in column 1 of the respective card for easy file identification. These laced punches are ignored by the card reader service routine.

After each interrupt, the card punch stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

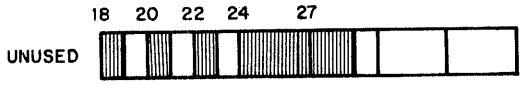
5.1.4 File Status (Refer to Appendix D)

The file status of the card punch is shown below.

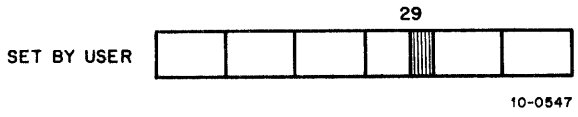
Standard Bits



Bit 19 - IO.DER	Punch error
Bit 21 - IO.BKT	Reached end-of-card with data remaining in buffer.
Bit 23 - IO.ACT	Device is active.



Device Dependent Bits



Bit 29 - IO.D29	If 1, punch DEC029 card codes in ASCII mode. If 0, punch DEC026 codes.

5.2 CARD READER

The card reader device mnemonic is CDR; the buffer size is 36_8 (33_8 data) words.

5.2.1 Concepts

For ASCII input, a header card can be the first card of the file and identifies the card code used (DEC026 or ANSI standard). The header card is used only when changing from or back to installation standard on ASCII input. The header card must not be present with any other data modes; if present, the header card is treated as incorrect format or read as data. Refer to Appendix C for the card codes.

An EOF card may have one of three forms: 12-11-0-1 punched in column 1, 6-7-8-9 punched in column 1, or a logical OR of the two punched in column 1. Columns 2 through 80 are ignored. The EOF card has the same effect as the EOF key on the card reader. This key must be depressed or the end-of-file card must be present at the end of each input file for all data modes.

To be compatible with PDP-11 operating systems, the DECsystem-10 card reader service accepts several other header card-code cards and EOF cards. Only column 1 is looked at; columns 2-80 are ignored.

	Punched by DECsystem-10	Also accepted
EOF	12-11-0-1-6-7-8-9†	12-11-0-1 6-7-8-9
DEC026	12-2-4-8	12-11-8-9†
ANSI	12-0-2-4-6-8	12-0-7-9

5.2.2 Data Modes

5.2.2.1 ASCII, Octal Code 0 - All 80 columns of each card are read and translated to 7-bit ASCII code. Blank columns are translated to spaces. At the end of each card a carriage return/line feed is appended. As many complete cards as can fit are placed in the input buffer, but cards are not split between two buffers. Using the standard-sized buffer, only one card is placed in each buffer.

Cards are normally translated as DEC026 card codes (refer to PDP-10 System Reference Manual). If a DEC029 header card is encountered, any following cards are translated as DEC029 codes (refer to Appendix C) until the 029 conversion mode is turned off. The 029 mode is turned off either by a RELEASE command or by a DEC026 header card. Columns 2 through 80 of both of these cards are ignored.

5.2.2.2 ASCII Line, Octal Code 1 - This mode is the same as ASCII mode.

5.2.2.3 Image, Octal Code 10 - All 12 punches in all 80 columns are packed into the buffer as 12-bit bytes. The first 12-bit byte is in column 1. The last word of the buffer contains columns 79 and 80 as the left and middle bytes, respectively. The EOF button is processed as in ASCII mode. Cards are not split between two buffers.

5.2.2.4 Image Binary, Octal Code 13 - This mode is the same as Image.

5.2.2.5 Binary, Octal Code 14 - Card column 1 must contain a 7-9 punch to verify that the card is in binary format. Column 1 also contains the word count in rows 12 through 3. The absence of the 7-9 punch results in setting the IO.IMP (bit 18 of status word) flag in the card reader status word. Card column 2 must contain a 12-bit checksum as described for the paper-tape binary format. Columns 3 through 80 contain binary data, 3 columns per word for up to 26 words. Cards are not split between two buffers. The EOF button is processed the same as in ASCII mode.

5.2.2.6 Super-Image, Octal Code 110†† - Super-image mode may be initialized by setting bit 29 of the card reader's IOS word. This mode causes the 36 bits read from the I/O bus to be BLKI'd directly to the user's buffer. For this mode, the default size of the input buffer is 81_{10} words (80_{10} data words).

† These cards are symmetric in the sense that the pattern of the punches is the same if the card is turned upside down.

†† This mode depends on FTCDRSI which is normally off in the DECsystem-1040.

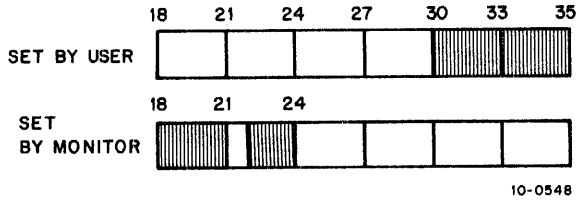
5.2.3 Special Programmed Operator Service

The card reader, after each interrupt, stores the results of a CONI in the DEVSTS word in the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

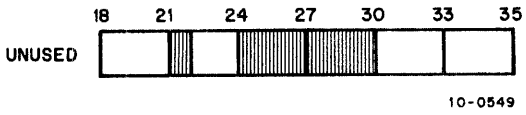
5.2.4 File Status (Refer to Appendix D)

The file status of the card reader is shown below.

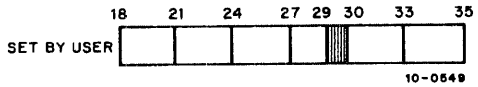
Standard Bits



Bit 18 - IO.IMP	7-9 punch absent in column 1 of a presumed binary card. The card reader is stopped.
Bit 19 - IO.DER	Photocell error, card motion error, data missed. The card reader is stopped.
Bit 20 - IO.DTE	Computed checksum is not equal to checksum read on binary card. The card reader is stopped.
Bit 22 - IO.EOF	EOF card read or EOF button pressed.
Bit 23 - IO.ACT	Device is active.



Device-Dependent Bits



Bit 29 - IO.SIM Super-Image mode.

5.3 DISPLAY WITH LIGHT PEN

The device mnemonic is DIS; there is no buffer because the display uses device-dependent dump mode only.

5.3.1 Data Modes

For IMAGE DUMP, Octal Code 15, an arbitrary length in the user area may be displayed on the scope. The command list format is as described in Chapter 4 with the addition for the Type 30, VR30 and VP10 display, that, if RH = 0, and LH $\neq$ 0, then LH specifies the intensity for the following data (4 to 13).

5.3.2 Background

During timesharing on a heavily-loaded system, the monitor service routine for the Type 30, VR30, and VP10 guarantees a flicker-free picture on the display if the job is locked in core. To maintain this picture, the picture data must be available for the display at least every two jiffies. If the system is lightly loaded, it is not necessary to keep the job in core. When the job is swapped, a minimum amount of flicker may occur, but the job has high priority to be swapped-in again.

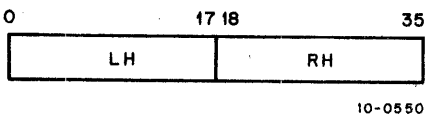
5.3.3 Display UUOs

The I/O UUOs for both displays operate as follows:

```
INIT D, 15              ;MODE 15 ONLY
SIXBIT /DIS/            ;DEVICE NAME
0                       ;NO BUFFERS USED
ERROR RETURN            ;DISPLAY NOT AVAILABLE
NORMAL RETURN
CLOSE D,                ;STOPS DISPLAY AND
    or                  ;RELEASES DEVICE AS
RELEAS D,               ;DESCRIBED IN CHAPTER 4
```

5.3.3.1 INPUT D, ADR - If a light pen hit has been detected since the last INPUT command, then C(ADR) is set to the location of last light pen hit. If no light pen hit has been detected since last INPUT command, then C(ADR) is set to -1.

5.3.3.2 OUTPUT D, ADR - ADR specifies the first address of a table of pointers. This table is composed of pointers with the following format:



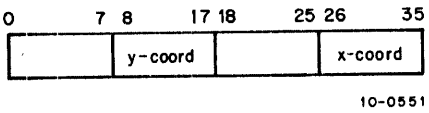
For the Type 30, VR30 and VP10 Display:

If LH = 0 and RH = 0, then this is the end of the command list.

If LH ≠ 0 and RH = 0, then LH is the desired intensity for the following data or commands. The intensity ranges from 4 to 13, where 4 is the dimmest and 13 is the brightest.

If LH = 0 and RH ≠ 0, then RH is the address of the next pointer. Successive pointers are interpreted beginning at RH.

If LH ≠ 0 and RH ≠ 0, then -LH words beginning at address RH + 1 are output as data to the display. The format of the data word is the following:



For the Type 340B Display:

If RH = 0, then this is the end of the command list.

If LH = 0 and RH ≠ 0, then RH is the address of the next pointer. Successive pointers are interpreted beginning at RH.

If LH ≠ 0 and RH ≠ 0, then -LH words beginning at address RH+1 are output as data to the display. The format of the data word is described in the Precision Incremental CRT Display Type 340 Maintenance Manual.

An example of a valid pointer list for the VR-30 display is:

```
OUTPUT   D, LIST                ;OUTPUT DATA
                                ;POINTED TO BY LIST
LIST;    XWD      5, 0          ;INTENSITY 5 (DIM)
         IOWD     1, A          ;PLOT A
         IOWD     5,SUBP1       ;PLOT SUBPICTURE 1
         XWD      13,0          ;INTENSITY 13 (BRIGHT)
         IOWD     1,C           ;PLOT C
         IOWD     2,SUBP2       ;PLOT SUBPICTURE 2
         XWD      0,LIST1       ;TRANSFER TO LIST 1

LIST1:   XWD      10,0          ;INTENSITY 10 (NORMAL)
         IOWD     1,B           ;PLOT B
         IOWD     1,D           ;PLOT D
         XWD      0,0           ;END OF COMMAND LIST
         OUTPUT   D, LIST       ;OUTPUT DATA
                                ;POINTED TO BY LIST
A:       XWD      6,6           ;Y=   6, X=6
B:       XWD     70,105         ;Y=  70, X=105
C:       XWD    105,70          ;Y= 105, X=70
D:       XWD   1000,200         ;Y=1000, X=200

SUBP1:   BLOCK    5             ;SUBPICTURE 1
SUB2:    BLOCK    2             ;SUBPICTURE 2
```

An example of a valid pointer list for the Type 340B Display is:

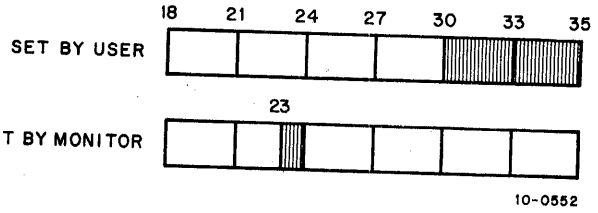
```
        OUTPUT  D, LIST         ;OUTPUT DATA POINTED
                                ;TO BY POINTER IN LIST
LIST:   IOWD    1,A             ;SET STARTING POINT TO (6,6)
        IOWD    5,SUBP1         ;DRAW A CIRCLE
        IOWD    1,C             ;SET STARTING POINT TO (70,105)
        IOWD    5,SUBP1         ;DRAW A CIRCLE
        IOWD    1,B             ;SET STARTING POINT TO (105,70)
        IOWD    2,SUBP2         ;DRAW A TRIANGLE
        IOWD    0,LIST1         ;TRANSFER TO LIST1
LIST1:  IOWD    1,D             ;SET STARTING POINT TO
                                ;(100,-200)
        IOWD    5,SUBP1         ;DRAW A CIRCLE
        IOWD    1,A             ;SET STARTING POINT TO (6,6)
        IOWD    2,SUBP2         ;DRAW A TRIANGLE
        XWD     0,0             ;STOP
A:      X=6     Y=6
B:      X=105   Y=70
C:      X=70    Y=105
D:      X=1000  Y=-200
SUBP1:  BLOCK   5               ;DRAW A CIRCLE
SUBP2:  BLOCK   2               ;DRAW A TRIANGLE
```

The example shows the flexibility of this format. The user can display a subpicture by setting up a pointer. He can also display the same subpicture in many different places by setting up pointers to the subpicture, each preceded by a pointer to commands for the display to reset its coordinates.

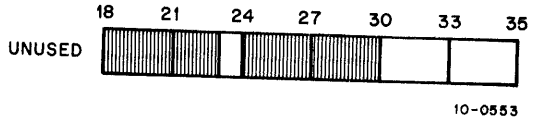
5.3.4 File Status (See Appendix D)

The file status of the display is shown below.

Standard Bits



Bit 23 - IO.ACT Device is active.



Device Dependent Bits - None.

5.4 LINE PRINTER

The device mnemonic is LPT; the buffer size is 37_8 (36_8 data) words.

5.4.1 Data Modes

5.4.1.1 ASCII, Octal Code 0 - ASCII characters are transmitted to the line printer exactly as they appear in the buffer. Refer to the PDP-10 System Reference Manual for a list of the vertical spacing characters.

5.4.1.2 ASCII Line, Octal Code 1 - This mode is exactly the same as ASCII and is included for programming convenience. All format control must be performed by the user's program; this includes placing a RETURN, LINE-FEED sequence at the end of each line.

5.4.1.3 Image, Octal Code 10 - This mode is the same as ASCII mode.

5.4.2 Special Programmed Operator Service

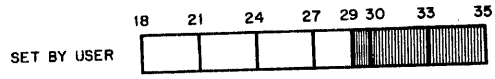
The first output programmed operator of a file and the CLOSE at the end of a file cause an extra form-feed to be printed to keep files separated.

After each interrupt, the line printer stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

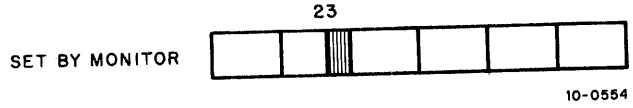
5.4.3 File Status (See Appendix D)

The file status of the line printer is shown below.

Standard Bits



Bit 29 - IO.SFF Suppress FORM FEEDS on an OPEN or RELEASE



Bit 23 - IO.ACT Device is active.

Device dependent bits - None.

5.5 MAGNETIC TAPE

Magnetic tape format is industry compatible, 7- or 9-channel 200, 556, and 800 bits/in. (see description below). The device mnemonic is MTA0, MTA1, ..., MTA7; the buffer size is 203_8 (200_8 data) words. The user may change the density and/or the blocksize of a magnetic tape by using the SET DENSITY and SET BLOCKSIZE monitor commands (refer to the DECsystem-10 Operating System Commands).

5.5.1 Data Modes

5.5.1.1 ASCII, Octal Code 0 - Data appears to be written on magnetic tape exactly as it appears in the buffer. No processing or checksumming of any kind is performed by the service routine. The parity checking of the magnetic tape system is sufficient assurance that the data is correct. Normally, all data, both binary and ASCII, is written with odd parity and at 800 bits per inch unless changed by the installation. A maximum of 200_8 words per record is allowed if the monitor has set up the buffer ring. If the user builds his own buffers, he may specify any number of words per record. The word count is not written on the tape. If an I/O error occurs or an end-of-tape is reached, reading ahead ceasing on input and implied output ceases on output.

5.5.1.2 ASCII Line, Octal Code 1 - The mode is the same as ASCII.

5.5.1.3 Image, Octal Code 10 - The mode is the same as ASCII, but data consists of 36-bit words.

5.5.1.4 Image Binary, Octal Code 13 - The mode is the same as Image.

5.5.1.5 Binary, Octal Code 14 - The mode is the same as Image.

5.5.1.6 DR (Dump Records), Octal Code 16 - Standard fixed length records (128 words is the standard unless installation standard is changed at MONGEN time) are read into or written from anywhere in the user's core area without regard to the standard buffering scheme. Control for read or write operations must be via a command list in core memory. The command list format is described in Chapter 4. For input operations a new record is read for each word in the command list (except GOTO words); if the record terminates before the command word is satisfied, the service routine reads the next records. If the command word runs out before the record terminates, the remainder of the record is ignored. For each output command word, enough standard length records are followed by one short record to exactly write all of the words on the tape. If an I/O error occurs or the end-of-tape is reached, no additional commands are retrieved from a dump mode command list, and I/O is terminated. When the end of file is read, the user receives the standard EOF return (the error return from the INPUT or IN UUO) and the IO.EOF bit is set in the file status word. This bit can be retrieved with the GETSTS

UUO. The EOF character (17_8 for 7-channel tapes or 23_8 for 9-channel tapes) is read into the user's buffer. The next INPUT or IN UUO will read the next record on the tape.

5.5.1.7 D (Dump), Octal Code 17 - Variable length records are read into or written from anywhere in the user's core area without regard to the standard buffering scheme. Control for read or write operations must be via a command list in core memory. The command list format is described in Chapter 4. For input operations a new record is read for each word in the command list (except GOTO words); if the record terminates before the command word is satisfied, the service routine skips to the next command word. If the command word runs out before the record terminates, the remainder of the record is ignored. For each output command word, exactly one record is written. Refer to Paragraph 5.5.1.6 for the description of EOF handling in dump mode.

5.5.2 Magnetic Tape Format

Magnetic tape format can generally be described as unlabelled, industry-compatible format. That is, as far as the user is concerned, the tape contains only data records and EOF marks, which signal the end of the data set or the end of the file.

An EOF mark consists of a record containing a 17_8 (for 7-channel tapes) or a 23_8 (for 9-channel tapes). EOF marks are used in the following manner:

a. No EOF mark precedes the first file on a magnetic tape.
b. An EOF mark follows every file.
c. Two EOF marks follow a file if that file is the last or only file on the tape.

Files are sequentially written on and read from a magnetic tape. A file consists of an integral number of physical records, separated from each other by interrecord gaps (area on tape in which no data is written). There may or may not be more than one logical record in each physical record.

5.5.3 Special Programmed Operator Service

CLOSE performs a special function for magnetic tape. When an output file is closed (both dump and nondump), the I/O service routine automatically writes two EOF marks and backspaces over one of them. If another file is opened, the second EOF mark is wiped out leaving one EOF mark between files. At the end of the in-use portion of the tape, however, a double EOF character, which is defined as the logical end of tape, appears.

After each interrupt, the magnetic tape service routine stores the results of a CONI in the DEVSTS word. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

5.5.3.1 MTAPE UUO - The MTAPE programmed operator provides for tape manipulation functions such as rewind, backspace record, backspace file, and 9-channel tape initialization. The format is

MTAPE D, FUNCTION

where D is the device channel on which the magnetic tape unit is initialized. FUNCTION is selected according to Table 5-2.

Table 5-2
MTAPE Functions

Symbol	Function	Action
MTWAT.	0	No operation; wait for spacing and I/O to finish.
MTREW.	1	Rewind to load point.
MTEOF.	3	Write EOF.
MTSKR.	6	Skip one record.
MTBSR.	7	Backspace record.
MTEOT.	10	Space to logical end of tape; terminates at either two consecutive EOF marks or at the end of first record beyond end of tape marker.
MTUNL.	11	Rewind and unload.
MTBLK.	13	Write 3 in. of blank tape.
MTSKF.	16	Skip one file; implemented by a series of skip record operations.
MTBSF.	17	Backspace files; implemented by a series of backspace record operations.
MTDEC.	100	Initialize for Digital-compatible 9-channel.†
MTIND.	101	Initialize for industry-compatible 9-channel tape.††

†Digital-compatible mode writes (or reads) 36 data bits in five frames of a 9-track magnetic tape. It can be any density, any parity, and is not industry compatible. This mode is in effect until a RELEAS D, or a MTIND.D, is executed.

††Industry-compatible 9-channel mode writes (or reads) 32 data bits per word in four frames of a 9-track magtape and ignores the low order four bits of a word. It must be 800 bits/in. density, odd parity.

MTAPE waits for the magnetic tape unit to complete the action in progress before performing the indicated function, including no operation (0). Bits 18 through 25 of the status word are then cleared, the indicated function is initiated, and control is immediately returned to the user's program. It is important to remember that when performing buffered input/output, the I/O service routine can be reading several blocks ahead of the user's program. MTAPE affects only the physical position of the tape and does not change the data that has already been read into the buffers. Therefore, an INPUT

or OUTPUT following a MTAPE may not retrieve the buffer containing the block requested. However, a single buffer ring retrieves the expected block since the device must stop after each INPUT or OUTPUT. Alternatively, if bit 30 (IO.SYN) of the file status word is set via the INIT or SETSTS UUO, the device stops after each buffer is filled on an INPUT or OUTPUT. Thus, the MTAPE will apply to the buffer supplied on the next INPUT or OUTPUT.

MTAPE functions must be followed by MTAPE 0 if subsequent operations depend on the completion of the MTAPE function. If this is not done, subsequent input and output UUOs are ignored until the magnetic tape control is freed. This problem occurs frequently in programs that issue a REWIND at the beginning of the program. The tape may actually be positioned at the beginning of the tape; however, the processing of the MTAPE function may cause the first input to be ignored.

Issuing a backspace file command to a magnetic tape unit moves the tape in the reverse direction until the tape has:

a. passed the end of file mark
b. reached the beginning of the tape.

The end of the backspace file operation positions the tape heads either immediately in front of a file mark or at the beginning of the tape.

In most cases it is desirable to skip forward over this file mark. This is decidedly not the case if the beginning of the tape is reached; in this case, giving a skip file command would skip the entire first file on the tape stopping at the beginning of the second file, rather than leaving the tape positioned at the beginning of the first file. Therefore, a typical (incorrect) sequence for backspace file would be:

```
MTBSF. MT,          ;Backspace file
WAIT MT,            ;Wait for completion
STATO MT,4000       ;Beginning of tape?
MTSKF. MT,          ;No, skip over file mark
```

It is necessary to wait after the backspace file instruction to ensure that the tape is moved to the EOF mark or the beginning of the tape before testing to see whether or not it is the beginning of the tape. The instruction WAIT MT, cannot be used for this purpose; it waits only for the completion of I/O transfer operation. (Backspace file is a spacing operation, not an I/O transfer operation.) Instead, use the following sequence for backspace file:

```
MTAPE MT,17         ;Backspace file
MTAPE MT,0          ;Wait for completion
STATO MT,4000       ;Beginning of tape?
MTAPE MT,16         ;No, skip over file mark
```

The device service routine must wait until the magnetic tape control is free before processing the MTAPE MT, 0 command, which tells the tape control to do nothing. Thus, the service routine achieves the waiting period necessary for the completion of the previous operation and the proper positioning of the tape.

5.5.3.2 MTCHR. AC, or CALLI AC, 112[1] - This UUO returns the characteristics (presently only the density is returned) of the specified magnetic tape drive. The density of the drive can be specified by setting bits 27 and 28 in the status word when the drive is INITed and can be changed with the SET DENSITY command. The call is:

```
MOVE AC, [SIXBIT/dev/]          ;or MOVEI AC, channel number
MTCHR. AC,                      ;or CALLI AC, 112
error return
normal return
```

In determining the density to return, the monitor examines the initial file status specified with the INIT UUO and returns the indicated density value. If this value is zero, the monitor then determines if the user specified a density with the SET DENSITY system command. If no density has been specified in this way, the monitor returns the system default density.

The MTCHR. UUO is used to obtain more complete information than that returned with the GETSTS UUO. The GETSTS UUO returns only the density specified in the INIT UUO and if the density is specified as zero (for the system default), zero is returned, not the actual system default. The density specified in the SET DENSITY command cannot be returned with the GETSTS UUO.

The error return is given if the UUO is not implemented (AC remains unchanged) or if there is no device on the specified channel or if the device is not a magnetic tape (AC contains -1).

On a normal return, bits 34 and 35 of AC contain the current density of the magnetic tape drive:

AC contains 1	200 bpi
AC contains 2	556 bpi
AC contains 3	800 bpi

5.5.4 9-Channel Magtape

Nine-channel magtape may be written and read in two ways: normal Digital-compatible format and industry-compatible format.

[1]This UUO depends on FT5UUO which is normally off in the DECsystem-1040.

5.5.4.1 Digital-Compatible Mode - Digital-compatible mode, the usual mode, allows old 7-channel user mode programs to read and write 9-channel tapes with no modification. Digital-compatible mode writes 36 data bits in five bytes of a nine track magtape. It can be any density, and parity, and is not industry compatible. The software mode is specified in the usual manner during initialization or with a SETSTS. User mode I/O is handled precisely as 7-track magtape. It is assumed that most DEC magtapes will be written and read in Digital-compatible mode.

For the data word in core there are 5 magnetic tape bytes per 36-bit word. Parity bits are unavailable to the user. Bits are written on tape as shown above; bits 30 and 31 are written twice and tracks 8 and 9 of byte 5 contain 0. On reading, parity bits and tracks 8 and 9 of byte 5 are ignored, the OR of bits (B30) is read into bit 30 of the data word, the OR of bits (B31) is read into bit 31.

Data Word on Tape

Tracks								
9	8	7	6	5	4	3	2	1
B0	B1	B2	B3	B4	B5	B6	B7	P
B8	B9	B10	B11	B12	B13	B14	B15	P
B16	B17	B18	B19	B20	B21	B22	B23	P
B24	B25	B26	B27	B28	B29	(B30)	(B31)	P
0	0	(B30)	(B31)	B32	B33	B34	B35	P

P = Parity
BN = Bit N in core.

5.5.4.2 Industry-Compatible Mode - For reading and writing industry-compatible 9-channel magtapes, an MTAPE D, 101 UUO must be executed to set the status. MTAPE D, 101 is meaningful for 9-channel magtape only and is ignored for all other devices. In the left half of the status word, bit 2 (which cannot be read by the user program) may be cleared, thus, the device is returned to 9-channel Digital-compatible status by a RELEAS, a call to EXIT, or an MTAPE D, 100 UUO. These MTAPE UUOs act only as a switch to and from industry-compatible mode and affect I/O status only by setting the density to 800 bits/in. and odd parity.

On INPUT, four 8-bit bytes are read into each word in the buffer, left justified, with the remaining four bits of the word containing character parity error indicators corresponding to the 8-bit bytes.

On OUTPUT, the leftmost four 8-bit bytes of each word in the buffer are written out in four frames, with the remaining four rightmost bits of the word being ignored.

Data Word on Tape

Tracks								
9	8	7	6	5	4	3	2	1
B0	B1	B2	B3	B4	B5	B6	B7	B32
B8	B9	B10	B11	B12	B13	B14	B15	B33
B16	B17	B18	B19	B20	B21	B22	B23	B34
B24	B25	B26	B27	B28	B29	B30	B31	B35

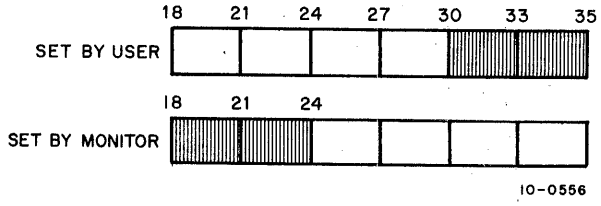
For data word in core, four magnetic tape bytes carry four 8-bit bytes from the data word. Parity bits are obtained as shown above when reading. The rightmost four bits (32-35) are ignored on writing.

5.5.4.3 Changing Modes - MTAPE CH, 101 automatically sets density at 800 bits (i.e., 800 eight-bit bytes) per inch and sets odd parity. Note that buffer headers are set up, when necessary by the monitor in the usual manner according to the I/O mode in which the device is initialized. In order to operate on eight-bit bytes, the user must insert the byte size in the byte pointer before the first IN or OUT.

5.5.5 File Status (refer to Appendix D)

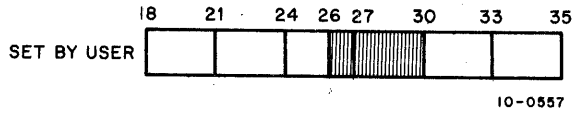
The file status of the magnetic tape is shown below.

Standard Bits

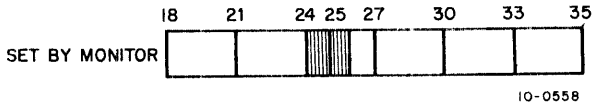


Bit 18 - IO.IMP	Unit was write-locked when output was attempted, or illegal operation was specified to the magnetic-tape control.
Bit 19 - IO.DER	Data was missed, tape is bad, or transport is hung.
Bit 20 - IO.DTE	Parity error.
Bit 21 - IO.BKT	Record read from tape exceeds buffer size.
Bit 22 - IO.EOF	EOF mark encountered. A 17_8 (for 7-channel tapes) or a 23_8 (for 9-channel tapes) appears in buffer.
Bit 23 - IO.ACT	Device is active.

Device Dependent Bits



Bit 26 - IO.PAR	I/O parity. 0 for odd parity, 1 for even parity. Odd parity is preferred. Even parity should be used only when creating a tape to be read in binary coded decimal (BCD) on another computer.
Bit 27-28 - IO.DEN	I/O density. 00 = System standard. Defined at MONGEN time and can be changed with the SET DENSITY command. 01 = 200 bits/in. 10 = 556 bits/in. 11 = 800 bits/in.
Bit 29 - IO.NRC	I/O no read check. Suppress automatic error correction if bit 29 is 1. Normal error correction repeats the desired operation 10 times before setting an error status bit.



Bit 24 - IO.BOT I/O beginning of tape. Unit is at beginning of tape mark.

Bit 25 - IO.EOT I/O tape end. Physical end of tape mark encountered.

5.6 PAPER-TAPE PUNCH

The device mnemonic is PTP; the buffer size is 43_8 (40_8 data) words.

5.6.1 Data Modes

5.6.1.1 ASCII, Octal Code 0 - The eighth hole is punched when necessary in order to make even parity. Tape-feed without the eighth hole (000) is inserted after form-feed. A rubout is inserted after each vertical or horizontal tab. Null characters (000) appearing in the buffer are not punched.

5.6.1.2 ASCII Line, Octal Code 1 - The mode is the same as ASCII mode. Format control must be performed by the user's program.

5.6.1.3 Image, Octal Code 10 - Eight-bit characters are punched exactly as they appear in the buffer with no additional processing.

5.6.1.4 Image Binary, Octal Code 13 - Binary words taken from the output buffer are split into six 6-bit bytes and punched with the eighth hole punched in each line. There is no format control or checksumming performed by the I/O routine. Data punched in this mode is read back by the paper-tape reader in the IB mode.

5.6.1.5 Binary, Octal Code 14 - Each bufferful of data is punched as one checksummed binary block as described for the paper-tape reader. Several blank lines are punched after each bufferful for visual clarity.

5.6.2 Special Programmed Operator Service

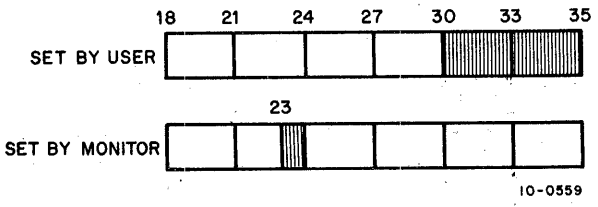
The first output programmed operator of a file causes approximately two fanfolds of blank tape to be punched as leader. Following a CLOSE, an additional fanfold of blank tape is punched as trailer. No EOF character is punched automatically.

After each interrupt, the paper-tape punch stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

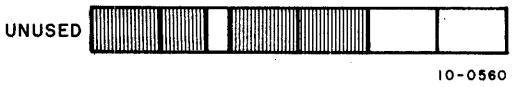
5.6.3 File Status (Refer to Appendix D)

The file status for the paper-tape punch is shown below.

Standard Bits



Bit 23 - IO.ACT Device is active.



Device Dependent Bits - None.

5.7 PAPER-TAPE READER

The device mnemonic is PTR; the buffer size is 43_8 (40_8 data) words.

5.7.1 Data Modes (Input Only)

NOTE

To initialize the paper-tape reader, the input tape must be threaded through the reading mechanism and the FEED button must be depressed.

5.7.1.1 ASCII, Octal Code 0 - Blank tape (000), RUBOUT (377), and null characters (200) are ignored. All other characters are truncated to seven bits and appear in the buffer. The physical end of the paper tape serves as an EOF, but does not cause a character to appear in the buffer.

5.7.1.2 ASCII Line, Octal Code 1 - Character processing is the same as for ASCII mode. The buffer is terminated by LINE FEED, FORM, or VT.

5.7.1.3 Image, Octal Code 10 - There is no character processing. The buffer is packed with 8-bit characters exactly as read from the input tape. Physical end of tape is the EOF indication but does not cause a character to appear in the buffer.

5.7.1.4 Image Binary, Octal Code 13 - Characters not having the eighth hole punched are ignored. Characters are truncated to six bits and packed six to the word without further processing. This mode is useful for reading binary tapes having arbitrary blocking format.

5.7.1.5 Binary, Octal Code 14 - Checksummed binary data is read in the following format. The right half of the first word of each physical block contains the number of data words that follow and the left contains half a folded checksum. The checksum is formed by adding the data words using 2's complement arithmetic, then splitting the sum into three 12-bit bytes and adding these using 1's complement arithmetic to form a 12-bit checksum. The data error status flag (refer to Table 4-3 in Paragraph 4.6.2) is raised if the checksum miscompares. Because the checksum and word count appear in the input buffer, the maximum block length is 40. The byte pointer, however, is initialized so as not to pick up the word count and checksum word.

Again, physical end of tape is the EOF indication, but does not result in putting a character in the buffer.

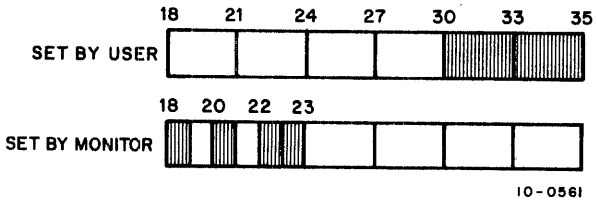
5.7.2 Special Programmed Operator Service

After each interrupt, the paper-tape reader stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

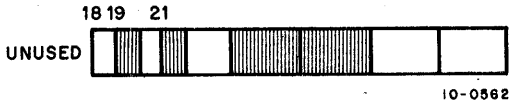
5.7.3 File Status (Refer to Appendix D)

The file status of the paper-tape reader is shown below.

Standard Bits



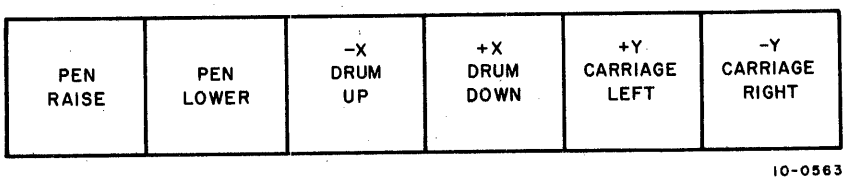
Bit 18 - IO.IMP	Binary block is incomplete.
Bit 20 - IO.DTE	Bad checksum in binary mode.
Bit 22 - IO.EOF	Physical end of tape is encountered. No character is stored in the buffer.
Bit 23 - IO.ACT	Device is active.



Device dependent bits - None

5.8 PLOTTER

The device mnemonic is PLT; the buffer size is 43_8 (40_8 data) words. The plotter takes 6-bit characters with the bits of each character decoded as follows:



Do not combine PEN RAISE or LOWER with any of the position functions. (For more details on the incremental plotter, refer to the PDP-10 System Reference Manual.)

5.8.1 Data Modes

5.8.1.1 ASCII, Octal Code 0 - Five 7-bit characters per word are transmitted to the plotter exactly as they appear in the buffer. The plotter is a 6-bit device; therefore, the leftmost bit of each character is ignored.

5.8.1.2 ASCII Line, Octal Code 1 - This mode is identical to ASCII mode.

5.8.1.3 IMAGE, Octal Code 10 - Six 6-bit characters per word are transmitted to the plotter exactly as they appear in the buffer.

5.8.1.4 IMAGE BINARY, Octal Code 13 - This mode is identical to Image mode.

5.8.1.5 BINARY, Octal Code 14 - This mode is identical to Image mode.

5.8.2 Special Programmed Operator Service

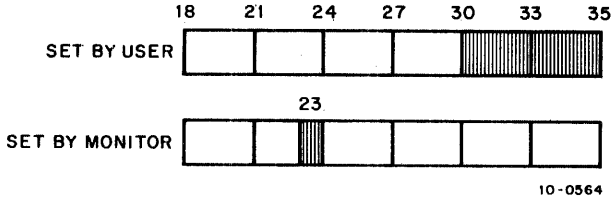
The first OUTPUT operator causes the plotter pen to be lifted from the paper before any user data is sent to the plotter. The CLOSE operator causes the plotter pen to be lifted after all user data is sent to the plotter. These two pen-up commands are the only modifications the monitor makes to the user output file.

After each interrupt, the plotter stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

5.8.3 File Status (Refer to Appendix D)

The file status of the plotter is shown below.

Standard bits



Bit 23 - IO.ACT Device is active.

Device dependent bits - None

5.9 PSEUDO-TTY

The device mnemonic is PTY0, PTY1, ..., PTYn. (The number of pseudo-TTYs is specified at MONGEN time.) The buffer size is 23_8 (20_8 data) words.

5.9.1 Concepts

Each job in the DECsystem-10 is usually initiated by a user at a physical terminal. Except in the case of a DETACH operation, the job remains under the control of the user's terminal until it is terminated by either the KJOB command or the LOGOUT UUO. For each physical terminal there is a block of core in the monitor, containing information about the physical terminal and including two buffers as the link between the physical terminal and the job. It is through these buffers that the terminal sends input to the job, and the job returns output to the terminal.

Sometimes it is desirable to allow a job in the DECsystem-10 to be initiated by a program instead of by a user. Since a program cannot use a physical terminal in the way a user can, some means must be provided in the monitor for the program to send input to and accept output from the job it is controlling. The monitor provides this capability via the pseudo-TTY (PTY). The PTY is a simulated terminal and is not defined by hardware. Like hardware-defined terminals, each PTY has a block of core associated with it. This block of core is used by the PTY in the same manner as a hardware-defined terminal uses its block of core. Figure 5-1 shows the parallel between a hardware-defined terminal and a software-defined PTY.

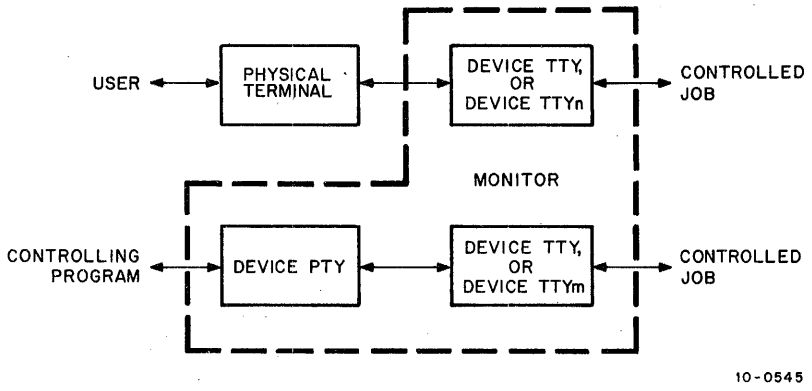


Figure 5-1 Pseudo-TTY

The controlling program, most commonly the batch processor, uses the PTY in the same way a user uses a physical device. It initiates the PTY, inputs characters to and waits for output from the PTY, and closes the PTY using the appropriate programmed operators. The job controlled by the program performs I/O to the PTY as though the PTY were a physical terminal.

A controlled job may go into a loop and not accept any input from its associated buffer; therefore, it is not possible for the controlling program to simply rely on waiting for activity in the controlled job. A controlling program may also wish to drive more than one controlled job, and be able to respond to any of these jobs; therefore, the controlling program cannot wait for any particular PTY. For these two reasons, the PTY differs from other devices in that it is never in a I/O wait state. Timing is accomplished by the HIBER UUO and the status bits of the PTY.

5.9.2 The HIBER UUO

The HIBER UUO (refer to Paragraph 3.1.4.2) allows the controlling program to temporarily suspend its operation until either there is activity in the controlled job or the specified amount of sleep time runs out, whichever occurs first. If bit 12 in the AC is set in the HIBER UUO call, any PTY activity since

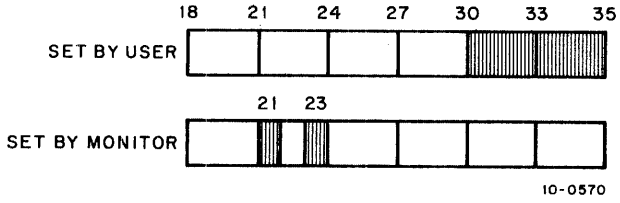
the last HIBER UUO causes the controlling program to be awakened. If no PTY activity occurs before the limit of sleep time is reached, the controlling program is activated, and it checks the controlled job's run time or other criteria to determine whether the job should be interrupted. If the job should be interrupted, the controlling program may output two control-C characters to stop the job. (A timesharing user stops a running job in the same way.) If the job should not be interrupted, the controlling program should repeat the HIBER UUO.

If bit 12 in AC is not set, unnecessary delays might result if activity occurred on a PTY while the controlling job was sleeping. To avoid these delays, a check is made when a PTY status bit changes to determine if the controlling program is in a sleep. If it is, the sleep time is cleared so the controlling program can service the PTY.

5.9.3 File Status (Refer to Appendix D)

The file status of the pseudo-TTY is shown below.

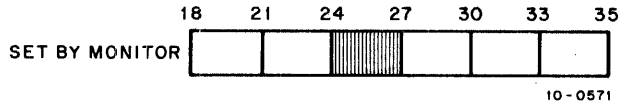
Standard Bits



Bit 21 - IO.BKT

Bit 23 - IO.ACT Device is active.

Device Dependent Bits



Bit 24 - IO.PTI Job is in a TTY input wait. The controlling job should perform an OUTPUT to the PTY.

Bit 25 - IO.PTO The TTY buffer has output to be read by an INPUT from the PTY.

Bit 26 - IO.PTM Any characters typed into the TTY buffer (by OUTPUT to the PTY) are read by the monitor command decoder instead of by the controlled job (i.e., the controlled job is in monitor mode).

5.9.4 Special Programmed Operator Service

5.9.4.1 OUT, OUTPUT UUOs - The first OUTPUT operation after an INIT or OPEN causes the special actions of the RELEASE UUO (refer to Paragraph 5.9.4.3) and then the following normal output operations.

a. Characters from the controlling program's buffer ring are placed in the input buffer of the TTY linked to the PTY.

b. The IO.PTI bit is cleared.

c. The IO.PTM bit is set or cleared as determined by the state of the TTY.

The following are exceptions to the normal output action:

a. NULLS (ASCII 000) are discarded.

b. If more OUTPUTS are performed than are accepted by the controlled job and if the limit on this excess is exceeded, the IO.BKT bit is set and the remainder of the controlling program's buffer is discarded.

c. Lower case characters sent to the controlled job are translated to upper case if the appropriate bit in the TTY is set.

5.9.4.2 IN, INPUT UUOs - Characters are read from the output buffer of the TTY and are placed in the buffer ring of the controlling program. If there are no characters to read, an empty buffer is returned. The INPUT UUO does not cause a WAIT.

All the available characters are passed to the controlling program. If there are more characters to read than can fit in the buffer of the controlling program, the IO.PTO bit remains set and another INPUT should be done. If the output buffer of the TTY is exhausted by the INPUT UUO, the IO.PTO bit is cleared.

5.9.4.3 RELEASE UUO - The RELEASE UUO causes the following special actions:

a. Any characters in the output buffer of TTY are discarded.

b. If the controlled job is still attached to TTY, it is detached.

c. The PTY is disassociated from the software channel.

CAUTION

Haphazard use of the PTY and subsequent RELEASE operations may leave detached jobs tying up core and other system resources.

5.9.4.4 JOBSTS UUO - This UUO provides status information about devices TTY and/or the controlled job in order to allow complete and accurate checking of a controlled job.

The call is:

```
MOVEI AC, user channel number     ;or MOVNI AC, job number
JOBSTS AC,                        ;or CALLI AC, 61
error return
normal return
```

When the UUO is called, AC contains a number n specifying the job and/or the TTY to be checked. If n is from 0 to 17, the specified TTY and job are those currently INITed on the user's channel n. If n is negative, the job to be checked is job number (-n).

The error return is given if one of the following is true:

a. the UUO is not implemented. If this is the case, check the I/O status word.

b. n is out of range.

c. there is no PTY INITed on channel n.

Otherwise the normal return is given and AC contains the following status information:

Name	Bit	Explanation
JB.UJA	Bit 0 = 1	Job number is assigned.
JB.ULI	Bit 1 = 1	Job is logged in.
JB.UML	Bit 2 = 1	TTY is at monitor level.
JB.UOA	Bit 3 = 1	TTY output is available.
JB.UDI	Bit 4 = 1	TTY is at user level and in input wait, or TTY is at monitor level and can accept a command. In other words, there is no command awaiting decoding or being delayed, the job is not running, and the job is not stopped waiting for operator device action.
JB.UJC	Bit 5 = 1	JACCT is set. In particular, ↑C↑C will not work.
	Bits 6-17	Reserved for the future.
JB.UJN	Bits 18-35	Job number being checked or 0 if no job number is assigned.

5.9.4.5 CTLJOB UUO - This UUO is used to determine the job number of the program (job) that is controlling the specified job, if any.

The call is:

```
MOVE AC, job number               ;-1 means user's job
CTLJOB AC,                        ;or CALLI AC, 65
error return
normal return
```

On a normal return, AC contains the job number of the program (job) that is controlling the controlled job. If AC = -1, the specified job is not being controlled via a PTY.

An error return is given if the UUO is not implemented or the job number is too large.

5.10 TERMINALS

The device mnemonic is TTY0, TTY1, ..., TTY176, TTY177, CTY; the buffer size is 23_8 (20_8 data) words.

Line number n of the Data Line Scanner DC10, PDP-8 680 System, or PDP-8/I DC68A System is referred to as TTYn. The console terminal is CTY. The DECsystem-10 monitor automatically gives the logical name TTY to the user's terminal when a job is initialized.

Terminal device names are assigned dynamically. For interconsole communication by program, one of the two users must type DEASSIGN TTY to make the terminal available to the other user's program as an I/O device. Typing ASSIGN TTYn is the only way to reassign a terminal that has been deassigned.

In a full-duplex terminal service, the two functions of a console, typein and typeout, are handled independently, and do not need to be handled in the strict sense of output first and then input. For example: if two operations are desired from PIP, the request for the second operation can be typed before receiving the asterisk after completion of the first. To stop unwanted output, a Control O is typed. Also, the command Control C does not stop a program instantly; the Control C will be delayed until the program requests input from the keyboard, and then the program will be stopped. When a program must be stopped instantly, as when it gets into a loop, Control C typed twice stops the program.

If, during output operations on a half-duplex terminal (not a local copy terminal), an echo-check failure occurs (i.e., the received character was not the same as the transmitted character), the I/O routine suspends output until the user types the next character. If that character is ↑C, the terminal is immediately placed in monitor mode. If it is ↑O, all TTY output buffers that are currently full are ignored, thus cutting the output short. All other characters cause the service routine to continue output. The user may cause a deliberate echo check by typing in while typeout is in progress. For example, to return to monitor control mode while typeout is in progress, the user must type any character ("X", for example) until an echo check occurs and output is suspended; then he types ↑C.

Programs waiting for TTY output are awakened ten characters before the output buffer is empty, causing them to be swapped in sooner and preventing pauses in typing. Programs waiting for TTY input will be awakened ten characters before the input buffer is filled, thus reducing the possibility of lost typein.

5.10.1 Data Modes

5.10.1.1 ASCII, Octal Code 0 and ASCII Line, Octal Code 1 - The input handling of all control characters is as follows. Characters with ASCII codes of 000 to 037 echo as ↑x and are passed to the program as a control character unless noted otherwise.

000	NULL		Ignored on input; suppressed on output.
001	↑A		No special action.
002	↑B		No special action.
003	↑C		Not passed to program. The user's terminal is switched to monitor mode the next time input is requested by the program. Two successive ↑Cs cause the terminal to be immediately switched to monitor mode. Performs a ↑U and a ↑O. For user program control of ↑C, refer to Paragraph 3.1.3.2.
004	↑D	(EOT)	Not echoed; therefore typing in a control-D (EOT) does not cause a full-duplex data phone to hang up.
005	↑E	(WRU)	No special action.
006	↑F		No special action.
007	↑G	(Bell)	Echoes as Bell and is a break character.
010	↑H	(Backspace)	Echoes as backspace.
011	↑I	(TAB)	Echoes as a TAB or an equivalent number of spaces. Refer to the SET TTY TAB command.
012	↑J	(Linefeed)	Echoes as a linefeed and is a break character.
013	↑K	(Vertical tab)	Echoes as a vertical tab or 4 linefeeds. Refer to the SET TTY FORM command.
014	↑L	(Form)	Echoes as a FORMFEED or 8 linefeeds. Refer to the SET TTY FORM command.
015	↑M	(Carriage return)	Passed to program if terminal is in a paper-tape input mode; otherwise, supplies a linefeed echo, is passed to program as a CR and LF, and is a break character due to the LF.
016	↑N		No special action.
017	↑O		Not passed to program. Complements output suppression bit allowing users to turn output on or off. INPUT, INIT, and OPEN clear the output suppression bit. This bit is also cleared by any other INPUT-class operation, such as DDTIN and TTCALLS 0, 2, 4, and 5, by input test TTCALLS 13 and 14, and by returning to monitor command level via ↑C or EXIT operations. Echoed as ↑O followed by carriage return/ linefeed.
020	↑P		No special action.
021	↑Q	(XON)	Starts paper-tape mode if .TTY TAPE command has been given; refer to Paragraphs 5.10.8 and 5.10.9.
022	↑R	(TAPE)	No special action.
023	↑S	(XOFF)	Ends paper-tape mode; refer to Paragraphs 5.10.8 and 5.10.9.

024	↑T	(NO TAPE)	No special action.
025	↑U		Deletes input line back to last break character. Echoed as ↑U followed by a carriage return/linefeed; is a break character. Passed to program if full character-set mode is true.
026	↑V		No special action.
027	↑W		No special action.
030	↑X		No special action.
031	↑Y		No special action.
032	↑Z		Acts as EOF on TTY input. Echoes as ↑Z followed by carriage return/linefeed. Is a break character.
033	↑[	(ESC)	The standard ASCII escape. Echoed as $; is a break character.
034	↑\		No special action.
035	↑]		No special action.
036	↑↑		No special action.
037	↑←		No special action.
040-137			Printing characters, no special action.
140-174			Lower case ASCII; translated to upper case, unless lower case mode is on. Echoes as upper case if translated to upper case.
175 and 176			Old versions of altmode; converted to the standard escape (033) unless in full character set mode INIT or TRMOP. UUO) or no ALTmode conversion is specified (TRMOP. UUO or SET TTY NO ALT command).
177			RUBOUT or DELETE: a. Completely ignored if in paper-tape mode (XON). b. Break character, passed to program if either DDT mode or full character-set mode is true. c. Otherwise (ordinary case) causes a character to be deleted for each rubout typed. All the characters deleted are echoed between a single pair of backslashes. If no characters remain to be deleted, echoes as a carriage return/linefeed.

On output, all characters are typed just as they appear in the output buffer with the exception of TAB, VT, and FORM, which are processed the same as on type-in. Programs should avoid sending ↑D, because it may have catastrophic effects (e.g., it may hang up certain data sets).

5.10.1.2 Image, Octal Code 10 - Image mode is legal for TTY input and output, except for terminals controlled by pseudo-TTYs (refer to Paragraph 5.9). Note that the terminal to be INITed in image mode must be ASSIGNed to a job. An attempt to do input to an unassigned terminal receives an error return with the IO.IMP bit set in the file status word. Image mode is available only in the 5.02 monitor and later.

Because, on input, any sequence of input characters must be allowed, ↑C and ↑Z may not cause their usual escape functions. This means that if the user program accepts all characters and does not release the terminal from image mode, no typein will release the user from this state; consequently, the terminal would effectively become dead to the system. The break character cannot be used to escape from this situation, because DC10 and the 630 do not detect the break character. To solve this design problem, an image input state is defined. If during the image input state, no characters are received for 10 seconds the end-of-file is forced. After another 10 seconds, the image input state is terminated by SCNSER (scanner service) and a ↑C is simulated. Therefore, if the user discovers that his program has failed because of this condition, he simply stops typing until a ↑C appears.

The image input state begins when the program goes into I/O wait because of an INPUT UUO in image mode. It ends when the program executes any non-image terminal output operation. If no output is desired, the TTCALL UUO can be executed to output a null string.

When using image mode input to read binary tapes, echoing should be suppressed by setting bit 28 in the TTY status word.

NOTE

Because there are no break characters in image mode, characters are transferred a character at a time instead of a line at a time. Therefore, an input buffer may only have one character in it when control is returned to the user program.

On output, the low-order eight bits of each word in the user's buffer are output. These characters are transmitted exactly as supplied by the user. Parity is neither checked nor added, and filler characters are not generated. Image mode affects buffered output (INIT, OUTPUT UUOs) only, except for one TTCALL function (refer to Paragraph 5.10.3).

5.10.2 DDT Submode [1]

To allow a user's program using buffered I/O and the DDT debugging program to use the same terminal without interferring with one another, the TTY service routine provides the DDT submode. This mode does not affect the TTY status if it is initialized with the INIT operator. It is not necessary to use INIT to perform I/O in the DDT submode. I/O in DDT mode is always to the user's terminal and not to any other device.

In the DDT submode, the user's program is responsible for its own buffering. Input is usually one character at a time, but if the typist types characters faster than they are processed, the TTY service routine supplies buffers full of characters at the same time.

[1] The usage described in this section is obsolete; new programs should use the TTCALL UUO (refer to Paragraph 5.10.3).

To input characters in DDT mode, use the sequence

```
MOVEI AC,BUF
CALL AC, [SIXBIT/DDTIN/]
```

BUF is the first address of a 21-word block in the user's area. The DDTIN operator delays, if necessary, until one character is typed in. Then all characters (in 7-bit packed format) typed in since the previous occurrence of DDTIN are moved to the user's area in locations BUF, BUF+1. The character string is always terminated by a null character (000). RUBOUTs are not processed by the service routine but are passed on to the user. The special control characters ↑O and ↑U have no effect. Other characters are processed as in ASCII mode.

To perform output in DDT mode, use the sequence

```
MOVEI AC,BUF
CALL AC,[SIXBIT /DDTOUT/]
```

BUF is the first address of a string of packed 7-bit characters terminated by a null (000) character. The TTY service routine delays until the previous DDTOUT operation is complete, then moves the entire character string into the monitor, begins outputting the string, and restarts the user's program. Character processing is the same as for ASCII mode output.

5.10.3 Special Programmed Operator Service

The TTCALL UUO is used to extend the capabilities of the terminal. The TTCALL operations are performed for a physical terminal (not a logical name TTY) and most operations reference the terminal controlling the job which executed the UUO. (There are exceptions, such as in the case of GETLCH.)

The general form of the TTCALL (operation code 051) programmed operator is as follows:

```
TTCALL AC, ADR
```

The AC field describes the particular function desired, and the argument (if any) is contained in ADR. ADR may be an AC or any address in the low segment above the job data area (137). It may be in high segment for AC fields 1 and 3. The functions are:

AC Field	Mnemonic†	Action
0	INCHRW	Input character and wait
1	OUTCHR	Output a character
2	INCHRS	Input character and skip
3	OUTSTR	Output a string
4	INCHWL	Input character, wait, line mode
5	INCHSL	Input character, skip, line mode
6	GETLCH	Get line characteristics
7	SETLCH	Set line characteristics
10	RESCAN	Reset input stream to command
11	CLRBFI	Clear type-in buffer
12	CLRBFO	Clear type-out buffer
13	SKPINC	Skip if a character can be input
14	SKPINL	Skip if a line can be input
15	IONEOU	Output as an image character
16-17		(Reserved for expansion)

†The TTCALL mnemonics are defined in a separate MACRO assembler table, which is scanned if an undefined OP CODE is found. If the symbol is found in the TTCALL table, it is defined as though it had appeared in an appropriate OPDEF statement, for example:

TYPE: OUTCHR CHARAC

If OUTCHR is undefined, it will be assembled as though the program contained the statement:

OPDEF OUTCHR TTCALL 1,

This facility is available in MACRO V.44 and later.

INPUT and INPUT TEST operations (TTCALLs 0, 2, 4, 5, 13 and 14) also clear the effect of the previous ↑O type in.

5.10.3.1 INCHRW ADR or TTCALL 0, ADR - This command inputs a character into the low-order seven bits of location ADR. If there is no character yet typed, the program waits.

5.10.3.2 OUTCHR ADR or TTCALL 1, ADR - This command outputs to the user's terminal the character in location ADR. Only the low order 7 bits of the contents of ADR are used. The remaining bits do not need to be zeroes.

If there is no room in the output buffer, the program waits until room is available. ADR may be in high segment.

5.10.3.3 INCHRS ADR or TTCALL 2, ADR - This command is similar to INCHRW, except that it skips on a successful return, and does not skip if there is no character in the input buffer; it never puts the job into a wait.

```
TTCALL 2,ADR
 JRST  NONE
JRST   DONE
```

5.10.3.4 OUTSTR ADR or TTCALL 3, ADR - This command outputs a string of characters in ASCIZ format:

```
TTCALL 3,MESSAGE
MESSAGE: ASCIZ /TYPE THIS OUT/
```

ADR may be in high segment.

5.10.3.5 INCHWL ADR or TTCALL 4, ADR - This command is the same as INCHRW, except that it decides whether or not to wait on the basis of lines rather than characters; as such, it is the preferred way of inputting characters, because INCHRW causes a swap to occur for each character rather than each line (compare DDT and PIP input). In other words, INCHWL returns the next character in the line if a break character has been typed.[1] If a break character has not been typed, INCHWL waits. Repeated uses of INCHWL return each of the successive characters of the line.

Note that a control-C character in the input buffer is sufficient to satisfy the condition of a pending line. Therefore, when the input is done, the control-C is interpreted and the job is stopped. This definition of a line also applies to TTCALL 5, and TTCALL 14,.

5.10.3.6 INCHSL ADR or TTCALL 5, ADR - This command is the same as INCHRS, except that its decision whether to skip is made on the basis of lines rather than characters.

5.10.3.7 GETLCH ADR or TTCALL 6, ADR - This command takes one argument, from location ADR, and returns one word, also in ADR. The argument is a number, representing a TTY line. Bits 18 and 19 of the line number are ignored since terminal numbers begin at 200000. If the argument is negative, the line number controlling the program is assumed. If the line number is greater than those defined in the system, a zero answer is returned.

---

[1] If the input buffer becomes nearly filled, the waiting-of-line condition is satisfied even though no break character appears. This is true of all line-mode input operations.

The normal answer format is as follows:

Name	Bit	Meaning
GL.ITY	0	Line is a pseudo TTY.
GL.CTY	1	Line is the CTY.
GL.DSP	2	Line is the display console.
GL.DSL	3	Line is the dataset data line.
	4	Obsolete.
GL.HDP	5	Line is half-duplex.
GL.REM	6	Line is a remote TTY.
GL.RBS	7	Line is at a remote batch station.
GL.LIN	11	A line has been typed in by the user.
	12	Obsolete.
GL.LCM	13	Lower case input mode is on.
GL.TAB	14	Terminal has tabs.
GL.LCP	15	Terminal input is not echoed, because device is local copy.
GL.PTM	16	Control Q (paper-tape) switch is on.
	17	Obsolete.
	18-35	200000 + line number.

5.10.3.8 SETLCH ADR or TTCALL 7, ADR - This command allows a program to set and clear some of the bits for GETLCH. They may be changed only for the job's controlling TTY. Bits 13, 14, 15, and 16 can be modified. Bits 18 and 19 of the line number argument are ignored.

Example:

```
SETO    AC,14
GETLCH  AC
TLZ     AC,BIT 13
TLO     AC,BIT 14
SETLCH  AC
```

5.10.3.9 RESCAN or TTCALL 10, 0 - This command is intended for use only by the COMPIL program. It causes the input buffer to be rescanned from the point where the last command began. If bit 35 of E is 1, the error return is given if there is a command in the input buffer. If the input buffer is empty, the skip return is given. Obviously, if the UUO is executed other than before the first input, that command may no longer be in the buffer. ADR is not used, but it is address checked.

5.10.3.10 CLRBFI or TTCALL 11, 0 - This command causes the input buffer to be cleared as if the user had typed a number of CONTROL Us. It is intended to be used when an error has been detected (e.g., if a user did not want any commands that he might have typed ahead to be executed).

5.10.3.11 CLRBFO or TTCALL 12, 0 - This command causes the output buffer to be cleared as if the user had typed CONTROL O. It should be used rarely, because usually one wants to see all output, up to the point of an error. This command is included primarily for completeness.

5.10.3.12 SKPINC or TTCALL 13, 0 - This command skips if the user has typed at least one character. It does not skip if no characters have been typed; however, it never inputs a character. It is useful for a computer-based program that wants to occasionally check for input and, if any, go off to another routine (such as FORTRAN operating system) to actually do the input.

5.10.3.13 SKPINL or TTCALL 14, 0 - This command is the same as SKPINC, except that a skip occurs if the user has typed at least one line.

5.10.3.14 IONEOU ADR or TTCALL 15, E - This command outputs the low-order eight bits of the contents of E as an image character to the terminal.

5.10.4 GETLIN AC, or CALLI AC, 34 - This UUO returns the SIXBIT physical name of the terminal that the job is attached to.

The call is:

```
GETLIN AC,                ;OR CALLI AC, 34
```

The name is returned left justified in the AC. If the job issuing the UUO is currently detached, the left half of AC contains a 0 on return. The right half of AC contains the right half of the physical name of the terminal to which the job was most recently attached. Therefore, by testing the left half of AC, jobs can determine if they are attached to a terminal.

Example:

```
CTY or TTY3 or TTY30
```

This UUO is used by the LOGIN program to print the TTY name.

5.10.5 TRMNO. AC, or CALLI AC, 115[1]

This UUO is used to obtain the number of the terminal currently controlling a particular job. This terminal number can then be used as the argument to the GETLCH (refer to Paragraph 5.10.3.7) and TRMOP. (refer to Paragraph 5.10.6) UUOs.

[1]This UUO depends on FT5UUO which is normally off in the DECsystem-1040.

The call is:

```
MOVE AC, job number
TRMNO. AC,                ;or CALLI AC, 115
error return
normal return
```

On a normal return, the right half of AC contains the universal I/O index (.UXxxx) for the terminal. The range of values is 200000 to 200777 octal. The symbol .UXTRM (octal value 200000) is the offset for the terminal indices.

On an error return, if the AC is unchanged, the UUO is not implemented. If the AC contains zero, one of three errors occurred:

1) The job is currently detached and therefore, no terminal is controlling it.
2) The job number is unassigned; i.e., there is no such job.
3) The job number is out of range and therefore illegal.

The particular error condition can be determined from the JOBSTS UUO (refer to Paragraph 5.9.4.4). For example,

```
MOVEI AC, number
TRMNO. AC,
   JRST .+2
JRST OK
JUMPN AC, not implemented
MOVNI AC, number
JOBSTS AC,
   JRST illegal number
JUMPL AC, detached
JRST no job assigned.
```

5.10.6 TRMOP. AC, or CALLI AC, 116[1]

This UUO allows the user to control, examine, and modify information about any terminal connected to the system. Many of the functions of this UUO are extensions to the TTCALL input and output functions (refer to Paragraph 5.10.3). Certain functions are privileged, or require that the user have the terminal ASSIGNed. Generally, any function is legal for the terminal on which the job issuing the UUO is running. In addition, any READ or SKIP function is legal for any terminal if the job issuing the UUO 1) has the privilege bit JP.SPM set, 2) is running with the JACCT bit set, or 3) is logged in as [1,2]. A SET or output function is legal for any terminal if the job 1) has the privilege bit JP.POK set, 2) is running with the JACCT bit set, or 3) is logged-in as [1,2].

---

[1] This UUO depends on FT5UUO which is normally off in the DECsystem-1040.

The call is:

```
        MOVE AC, [XWD N, ADR]
        TRMOP. AC,                      ;or CALLI AC, 116
        error return
        normal return

ADR:    function code
ADR+1:  universal I/O index
```

ADR is the address of the argument block and N is the length (N must be at least 2). The first word of the argument block contains the code for the requested function. The second word contains the universal I/O index of the terminal to be affected (.UXTRM + line number). This index is in the same format as returned by the TRMNO. UUO (refer to Paragraph 5.10.5). Remaining arguments in the argument block depend on the particular function used.

Function codes are defined within the following ranges:

0000-0777	Perform a specific action.
1000-1777	Read a parameter.
2000-2777	Set a parameter.
3000-3777	Reserved for DEC customers.

The functions within the range 0000-0777 are as follows:

.TOSIP	1	Skip if terminal input buffer is not empty.
.TOSOP	2	Skip if terminal output buffer is not empty.
.TOCIB	3	Clear terminal input buffer.
.TOCOB	4	Clear terminal output buffer.
.TOOUC	5	Output character to terminal from ADR+2 (not yet implemented).
.TOOIC	6	Output image mode (8-bit) character from ADR+2 (not yet implemented).
.TOOUS	7	Output ASCIZ string to terminal from address at ADR+2 (not yet implemented).
.TOINC	10	Input character from terminal to AC, normal mode (not yet implemented).
.TOIIC	11	Input character from terminal to AC, image mode (not yet implemented).
.TODSE	12	Enable modem for outgoing call.
.TODSC	13	Enable and place outgoing call on modem with dialer. Phone number of up to 17 digits is stored in 4-bit bytes in ADR+2 and ADR+3 and is terminated by a 17 byte. If caller must wait for a second dial tone (e.g., after dialing a 9), a 16 byte results in a 5 second wait.
.TODSF	14	Hang up modem (i.e., disconnect call).

The READ (1000-1777) and SET (2000-2777) functions are parallel; i.e., if function 1002 reads a particular parameter, then function 2002 sets the same parameter. Values for the READ functions are returned in AC; arguments to the SET functions are given in ADR+2. One-bit quantities are not range-checked; instead bit 35 of ADR+2 is stored. The following description of the READ function codes indicate if there is a corresponding SET function code.

Read Code	Range	Description	Corresponding SET
1000	1 bit	Output in progress (.TOOIP)	No
1001	1 bit	Terminal at monitor mode (.TOCOM)	No
1002	1 bit	Paper tape mode (.TOXON)	Yes
1003	1 bit	Lower case (if set, no lower case) (.TOLCT)	Yes
1004	1 bit	Slave switch (.TOSLV)	Yes
1005	1 bit	Tab switch (if 0 = spaces, if 1 = tab) (.TOTAB)	Yes
1006	1 bit	Form switch (if 0 = linefeeds, if 1 = formfeeds) (.TOFRM)	Yes
1007	1 bit	Local copy switch (if set, no echo) (.TOLCP)	Yes
1010	1 bit	Free CR-LF switch (if set, no CR-LF) (.TONFC)	Yes
1011	0 to 377	Horizontal position of carriage (.TOHPS)	No
1012	16. to 200.	Carriage width (.TOWID)	Yes
1013	1 bit	TTY GAG bit (if set, NO GAG) (.TOSND)	Yes
1014	1 bit	Half-duplex line (.TOHLF)	Yes, privileged
1015	1 bit	Remote line (.TORMT)	Yes, privileged
1016	1 bit	Display terminal (.TODIS)	Yes, privileged
1017	0 to 3	Filler class (.TOFLC)	Yes
1020	1 bit	Paper tape enabled (.TOTAP)	Yes
1021	1 bit	Paged display mode (also set and cleared by SET TTY PAGE)(.TOPAG)	Yes
1022	1 bit	Suspended output (need XON to resume) (also set by XOFF, formfeed, or page size exceeded, if paged display mode)(.TOSTP) Not implemented.	Yes
1023	0 to 63.	Page size (number of lines) (also set by SET TTY PAGE)(.TOPSZ) Not implemented.	Yes
1024	0 to 63.	Page counter (number of lines output this page)(.TOPCT)	Yes
1025	1 bit	Suppress blank lines on output (0 = normal output and 1 = suppress multiple linefeeds) and convert formfeeds and vertical tabs to linefeeds (also set and cleared by SET TTY BLANK)(.TOBLK)	Yes

Read Code	Range	Description	Corresponding SET
1026	1 bit	Suppress ALTmode conversion on input (0 = 175 and 176 converted to 033 and 1 = no conversion) (also set and cleared by SET TTY ALT)(.TOALT)	Yes

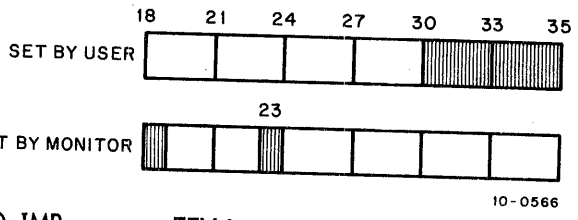
On an error return, AC is either unchanged or contains an error code.

AC	Name	Meaning
unchanged		UUO is not implemented.
0		The requested function is not implemented.
1	TOPRC%	User is not privileged to perform this function.
2	TORGB%	Argument is out of range.
3	TOADB%	Argument list length or address is illegal.
4	TOIMP%	Dataset activity to a non-dataset terminal.
5	TODIL%	Subfunction failed (e.g., call not properly completed from dialer).

5.10.7 File Status (Refer to Appendix D)

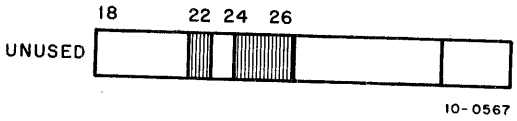
The file status of the terminal is shown below.

Standard Bits

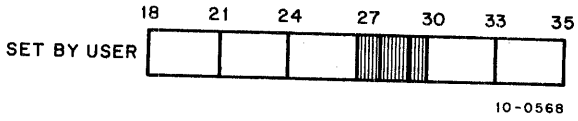


Bit 18 - IO.IMP — TTY is not assigned to a job (for image mode input processing).

Bit 23 - IO.ACT — Device is active.



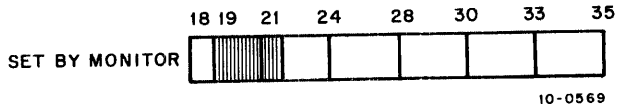
Device Dependent Bits



Bit 27 - IO.TEC — This bit causes 001 through 037, 175, and 176 (octal) to echo the character exactly as received by the monitor. There is no special echo (e.g., $ or ↑ x).

Bit 28 - IO.SUP — Suppresses echoing on the terminal.

Bit 29 - IO.FCS — Full character set. Pass all characters except lower case and ↑ C. Lower case is controlled by the SET TTY LC command and its corresponding TRMOP. UUO function.



Bit 19 - IO.DER Ignore interrupts for three-fourths of a second.

Bit 20 - IO.DTE Echo failure has occurred on output.

Bit 21 - IO.BKT Character was lost on typein.

5.10.8 Paper-Tape Input from the Terminal (Full-Duplex Software)

Paper-tape input is possible from a terminal equipped with a paper-tape reader that is controlled by the XON (↑Q) and XOFF (↑S) characters. When commanded by the XON character, the terminal service reads paper tapes, starting and stopping the paper tape as needed, and continuing until the XOFF character is read or typed in. While in this mode of operation, any RUBOUTS will be discarded and no free line feeds will be inserted after carriage returns. Also, TABS and FORMFEEDS will not be simulated on a Teletype Model 33 to ensure output of the reader control characters. To use paper tape processing, the terminal with a paper-tape reader must be connected by a full-duplex connection and only ASCII paper tapes should be used.

The correct operating sequence for reading a paper tape in this way is as follows:

```
.R PIP
*DSK:FILE←TTY:↑Q
THIS IS WHAT IS ON TAPE
MORE OF THE SAME
LAST LINE ↑Z
*↑C
```

5.10.9 Paper-Tape Output at the Terminal (Full-Duplex Software)

Paper-tape output is possible on any terminal-mounted paper-tape punch, which is controlled by the TAPE, AUX ON (↑R) and ~~TAPE,~~ AUX OFF (↑T) characters. The punch is connected in parallel with the keyboard printer, and therefore, when the punch is on, all characters on the keyboard are punched on tape.

LT33B or LT33H Teletypes can have the reader and punch turned off and on under program control. When commanded by the AUX ON character, the TTY service punches paper tapes until the AUX OFF character is read or typed in. The AUX OFF character is the last character punched on tape.

When writing programs to output to the terminal paper-tape punch, the user should punch several inches of blank tape before the AUX OFF character is transmitted. This last character may then be torn off and discarded.

CHAPTER 6
I/O PROGRAMMING FOR DIRECTORY DEVICES

This chapter explains the unique features of the standard directory devices. Each device accepts the programmed operators explained in Chapter 4, unless otherwise indicated. Table 6-1 is a summary of the characteristics of the directory devices. Buffer sizes are given in octal and include three bookkeeping words. The user may determine the physical characteristics associated with a logical device name by calling the DEVCHR UUO (refer to Paragraph 4.10.2).

Table 6-1
Directory Devices

Device	Physical Name	Controller Number	Unit Number	Programmed Operators	Data Modes	Buffer Sizes (Octal †)
DECtape	DTA0, DTA1, ...,DTA7 DTB0, DTB1, ...,DTB7††	TD10 551(PDP-6)	TU55 555(PDP-6)	INPUT, IN OUTPUT, OUT LOOKUP, ENTER MTAPE, USETF, USETO, USETI UTPCLR	A,AL,I B,IB DR,D	202
Fixed-Head Disk	DSK, FHA, FHA0, ..., FHA3	RC10	RD10 RM10B	INPUT, IN OUTPUT, OUT LOOKUP, ENTER RENAME, SEEK USETO, USETI	A,AL,I B,IB DR,D	203
Disk Pack	DSK, DPA, DPA0, ..., DPA7	RP10	RP01 RP02	INPUT, IN OUTPUT, OUT LOOKUP, ENTER RENAME, SEEK USETO, USETI	A,AL,I B,IB DR,D	203

†Buffer sizes are subject to change and should be calculated rather than assumed by user programs. A DEVSIZ UUO may be employed.

††Recognized if dual DECtape controller is supported.

6.1 DECTAPE

The device mnemonic is DTA0, DTA1, ..., DTA7; the buffer size is 202_8 words (177_8 user data, 200_8 transferred). On systems with dual DECtape controllers, the drives on the second controller have the mnemonic DTB0, DTB1, ..., DTB7.

6.1.1 Data Modes

Two hundred words are written. The first word is the link plus word count. The following 177_8 words are data supplied to and from user programs.

6.1.1.1 Buffered Data Modes - Data is written on DECtape exactly as it appears in the buffer and consists of 36-bit words. No processing or checksumming of any kind is performed by the service routine. The self-checking of the DECtape system is sufficient assurance that the data is correct. Refer to Paragraph 6.1.2 for further information concerning blocking of information.

6.1.1.2 Unbuffered Data Modes - Data is read into or written from anywhere in the user's core area without regard to the standard buffering scheme. Control for read or write operations must be via a command list in core memory. The command list format is described in Chapter 4. On the KI10, if the IOWD list is modified as the result of I/O performed (i.e., an INPUT UUO reads into the IOWD list) and the word count of any of the IOWDs read into the list is greater than the following value:

(maximum word count specified in original list-2)/512 + 2

then the job is stopped and the monitor types

ADDRESS CHECK AT USER adr

File-structured dump mode data is automatically blocked into standard-length DECtape blocks by the DECtape service routine. Each block read or written contains 1 link word plus 1 to 177_8 data words. Unless the number of data words is an exact multiple of the data portion of a DECtape block (177_8), the remainder of the last block written after each output programmed operator is wasted. The input programmed operator must specify the same number of words that the corresponding output programmed operator specified to skip over the wasted fractions of blocks.

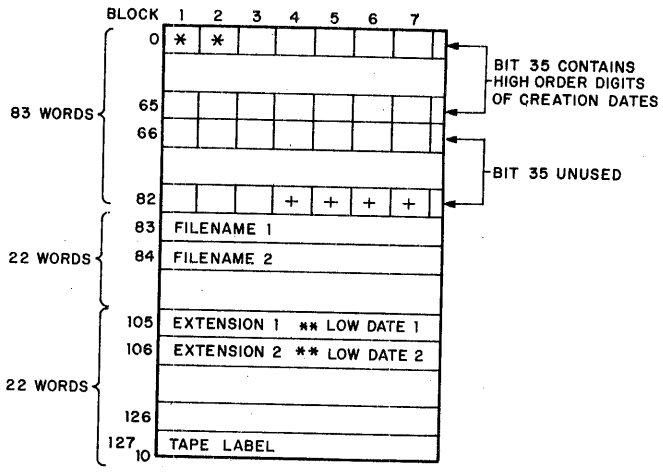
6.1.2 DECtape Format

A standard reel of DECtape consists of 578 (1102_8) prerecorded blocks each capable of storing 128 (200_8) 36-bit words of data. Block numbers that label the blocks for addressing purposes are recorded between blocks. These block numbers run from 0 to 1101_8. Blocks 0, 1, and 2 are normally not used during timesharing and are reserved for a bootstrap loader. Block 100_{10} (144_8) is the directory block, which contains the names of all files on the tape and information relating to each file. Blocks 3_{10} through 99_{10} ($1\text{-}143_8$) and 101_{10} through 577_{10} ($145\text{-}1101_8$) are usable for data.

If, in the process of DECtape I/O, the I/O service routine is requested to use a block number larger than 1101_8 or smaller than 0, the monitor sets the IO.BKT flag (bit 21) in the file status and returns.

6.1.3 DECtape Directory Format

The directory block (block 100_{10}) of a DECtape contains directory information for all files on that tape; a maximum of 22 files can be stored on any one DECtape (see Figure 6-1).



NOTES:

* Reserved for system, contains 36 as does block 144_8 for the directory.

** For zero-compressed files, this area holds the number of 1K blocks (-1) needed to load the file (up to 64K).

\+ Represents blocks 1102 through 1105, which are not available contains 37_8.

10-0572

Figure 6-1 DECtape Directory Format

The first 83 words (0 through 82_{10}) of the directory block contains slots for blocks 1 through 577 on a DECtape. Each slot occupies five bits (seven slots are stored per word) and represents a given block on the DECtape. Each slot contains the number of the file ($1\text{-}26_8$) occupying the given block. This allows for 581 slots (83 words x 7 slots per word). The four extra slots represent nonexistent blocks 1102 through 1105_8.

Bit 35 of the first 66 words (0 through 65_{10}) of the directory block contain the high order 3 bits of the 15-bit creation date of each file on the DECtape. (Note that the low order 12 bits of the creation date of each file are contained in words 105 through 126_{10}. This split format allows for compatibility among monitors and media as old as 1964.) The high order 3 bits of the 15-bit creation date for file 1 are contained in bit 35 of words 0, 22, and 44. Word 44 contains the first (most significant) digit; word

22 contains the second and word 0 contains the third. The high order digits for file 2 are contained in bit 35 of words 1, 23, and 45 with the digits in the same order as described for file 1. The high order digits for the remaining files are organized in the same fashion.

Words 83 through 104_{10} of the directory block contain the filenames of the 22 files that reside on the DECtape. Word 83 contains the filename for file 1, word 84 contains the filename for file 2. Filenames are stored in SIXBIT code.

The next 22 words of the directory block (words 105 through 126_{10}) primarily contain the filename extensions and the low order part of the creation dates of the 22 files that reside on the DECtape, in the same relative order as their filenames. The bits for each word are as follows:

Bits 0 - 17_{10}	The filename extension in SIXBIT code.
Bits 18 - 23_{10}	The number of 1K blocks minus 1 needed to load the file (maximum value is 63). This information is stored for zero-compressed files only.
Bits 24 - 35_{10}	The low order 12 bits of the date on which the file was created. (Note that the high order digits are encoded in bit 35 of words 0 through 65_{10}). The creation date is computed with the following formula: ((year-1964) * 12 + (month-1)) * 31 + day -1

Word 127_{10} of the directory block is the tape label.

The message

BAD DIRECTORY FOR DEVICE DTAn: EXEC CALLED FROM USER LOC n

occurs when any of the following conditions are detected:

a. A parity error occurred while reading the directory block.

b. No slots are assigned to the file number of the file.

c. The tape block, which may be the first block of the file (i.e., the first block for the file encountered while searching backwards from the directory block), cannot be read.

Ordinary user programs never manipulate DECtape directories explicitly since the LOOKUP and ENTER programmed operators (refer to Paragraphs 6.1.5.1 and 6.1.5.2) automatically record all necessary entries in the directory for the user. These programmed operators have all the capability needed to process the name and creation date of a file. However, a small number of special purpose programs do process directories by explicit action rather than using the LOOKUP and ENTER operators. For such programs, the following examples illustrate methods for 1) assembling the 15-bit creation date and 2) storing the 15-bit creation date. The number of the file (an integer from 1 to 22) is in register P1 and the directory block begins at location DIRECT.

Example 1 Special Purpose Assembly of the Creation Date

```
LDB     T1, [POINT 12, DIRECT+ ↑D104 (P1), 35]   ;GET LOW PART
MOVEI   T2, 1                                    ;SET UP TO TEST LOW BIT
TDNE    T2, DIRECT-1 (P1)                        ;IF SET IN DIRECTORY
TRO     T1, 1B23                                 ;THEN SET BIT IN DATE
TDNE    T2, DIRECT+ ↑D21 (P1)                    ;REPEAT FOR EACH BIT IN
TRO     T1, 1B22                                 ;HIGH PART OF DATE
TDNE    T2, DIRECT+↑D43 (P1)                     ;
TRO     T1, 1B21                                 ;
```

Example 2 Special Purpose Storage of the Creation Date

```
DPB     T1, [POINT 12, DIRECT+↑D104 (P1), 35]    ;SAVE LOW PART
MOVEI   T2, 1                                    ;SET UP TO MARK LOW BIT
ANDCAM  T2, DIRECT-1 (P1)                        ;CLEAR DIRECTORY BIT
TRNE    T1, 1B23                                 ;IF BIT IN DATE SET,
IORM    T2, DIRECT-1 (P1)                        ;SET DIRECTORY BIT
ANDCAM  T2, DIRECT+↑D21 (P1)                     ;
TRNE    T1, 1B22                                 ;REPEAT FOR EACH BIT IN
IORM    T2, DIRECT+↑D21 (P1)                     ;HIGH PART OF DATE
ANDCAM  T2, DIRECT+↑D43 (P1)                     ;
TRNE    T1, 1B21                                 ;
IORM    T2, DIRECT+↑D43 (P1)                     ;
```

6.1.4 DECtape File Format

A file consists of any number of DECtape blocks.

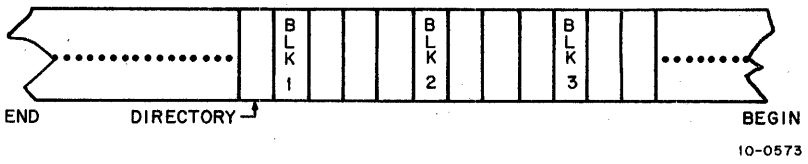


Figure 6-2 Format of a File on Tape

Each block contains the following:

Word 0	Left half	The link. The link is the block number of the next block in the file. If the link is zero, this block is the last in the file.
	Right half	Bits 18 through 27: the block number of the first block of the file. Bits 28 through 35: a count of the number of words in this block that are used (maximum 177_8).
Words 1 through 177_8		Data packed exactly as the user placed in his buffer or in dump mode files, the next 177 words of memory.

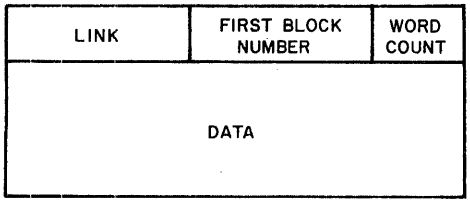


Figure 6-3 Format of a DECtape Block

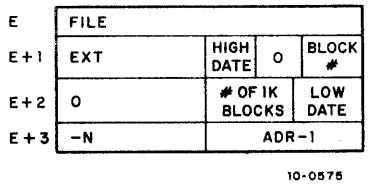
6.1.4.1 Block Allocation - Normally, blocks are allocated by starting with the first free block nearest the directory and going backwards to the front of the tape (block 0). When the end of the tape is reached, the direction of the scan is reversed. Blocks are not written contiguously; rather they are separated by a spacing factor. This allows the drive to stop and restart to read the next block of the file without having to back up the tape. The spacing factor is normally four, but for dump mode and UGETF followed by an ENTER, the spacing factor is two (refer to Paragraph 6.1.6.3).

6.1.5 I/O Programming

DECtape is a directory device; therefore, file selection must be performed by the user before data is transferred. File selection is accomplished with LOOKUP and ENTER UUOs. The UUO format is as follows:

UUO D, E

where D specifies the user channel associated with this device, and E points to a four-word parameter block. The parameter block has the following format:



(continued on next page)

where

FILE is the filename in SIXBIT ASCII.

EXT is the filename extension in SIXBIT ASCII.

HIGH DATE contains the high order 3 bits of the creation date.

BLOCK # is the number of the first block of the file.

# of 1K blocks is the number of blocks needed to load the file if the file is a zero-compressed file (bits 18-23).

LOW DATE contains the low order 12 bits of the date on which the file was originally created (bits 24-35). The format is the same as that used by the DATE UUO.

-N is the negative word length of the zero-compressed file.

ADR-1 is the core address of the first word of the file minus 1.

Location E + 3 is used for zero-compressed files.

6.1.5.1 LOOKUP D, E - The LOOKUP programmed operator sets up an input file on channel D. The contents of location E and E + 1 (left half) are matched against the filenames and filename extensions in the DECtape directory. If no match is found, the error return is taken (refer to Appendix E). If a match is found, locations E + 1 through E + 3 are filled by the monitor, and the normal return is taken (refer to Table 6-2). Refer to Section d. of Paragraph 6.2.8.1 for sample code of assembling the 15-bit creation date.

Table 6-2
LOOKUP Parameters

On Call			On Return		
Parameter	Use†	Contents	Parameter	Use†	Contents
E	A	SIXBIT /FILE/	E	V	SIXBIT /FILE/
E + 1	A	SIXBIT /EXT/	E + 1	V	LH = SIXBIT /EXT/ RH = high order 3 bits of 15-bit creation date (Bits 18-20) unused (Bits 21-25) first block # (Bits 26-35)
E + 2	I	-	E + 2	V	LH = 0 RH = # of 1K blocks (Bits 18-23) †† low order 12 bits of 15-bit creation date (Bits 24-35)††
E + 3	I	-	E + 3	V	IOWD LENGTH, ADR††

†A = argument from user program, V = value from monitor, I = ignored.

††For zero-compressed files only.

The first block of the file is then found as follows:

a. The first 83 words of the DECtape directory are searched backwards, beginning with the slot immediately prior to the directory block, until the slot containing the desired file number is found.

b. The block associated with this slot is read in and bits 18 through 27 of the first word of the block (these bits contain the block number of the first block of the file) are checked. If the bits are equal to the block number of this block, then this block is the first block; if not, then the block with that block number is read as the first block of the file.

6.1.5.2 ENTER D, E - The ENTER programmed operator sets up an output file on channel D. The DECtape directory is searched for a filename and filename extension that match the contents of location E and the left half of location E + 1. If no match is found and there is room in the directory, the monitor records the information in locations E through E + 2 in the DECtape directory (refer to Table 6-3). An error return is given if there is no room in the directory for the file (refer to Appendix E). Refer to Paragraph 6.2.8.3 for a special note on error recovery. If a match is found, the new entry replaces the old entry, the old file is reclaimed immediately, and the monitor records the file information. This process is called superseding and differs from the process on disk in that, because of the small size of DECtape, the space is reclaimed before the file is written rather than after. Refer to Section d. of Paragraph 6.2.8.1 for sample code for setting the 15-bit creation date.

Table 6-3
ENTER Parameters

On Call			On Return		
Parameter	Use †	Contents	Parameter	Use †	Contents
E	A	SIXBIT /FILE/	E	V	SIXBIT /FILE/
E + 1	A	LH = SIXBIT /EXT/ RH = high order 3 bits of 15-bit creation date (bits 18-20).	E + 1	V	LH = SIXBIT /EXT/ RH = high order 3 bits of 15-bit creation date (bits 18-20).
E + 2	A	RH = low order 12 bits of desired 15-bit creation date or 0. (0 implies current date)	E + 2	V	RH = low order 12 bits of 15-bit creation date (bits 24-35).
E + 3	I	-	E + 3	I	-

†A = argument from user program, V = value from monitor, I = ignored.

6.1.5.3 RENAME D, E - The RENAME programmed operator alters the filename or filename extension of an existing file, or deletes the file directory from the DECtape associated with channel D. If location E contains a 0, RENAME deletes the directory of the specified file; otherwise, RENAME searches for the file and enters the information specified in location E and E + 1 into the DECtape directory (refer to Table 6-4). RENAME must be preceded by a LOOKUP or an ENTER, to select the file that is to be RENAMED, and a CLOSE. The error return is given if a LOOKUP has not been done (refer to Appendix E). Refer to Paragraph 6.2.8.3 for a special note on error recovery.

Table 6-4
RENAME Parameters

On Call			On Return		
Parameter	Use †	Contents	Parameter	Use †	Contents
E	A	SIXBIT /FILE/ or 0	E	V	SIXBIT /FILE/
E + 1	A	LH = SIXBIT /EXT/ RH = high order 3 bits of 15-bit creation date (bits 18-20).	E + 1	V	LH = SIXBIT /EXT/ RH = high order 3 bits of 15-bit creation date (bits 18-20).
E + 2	A	RH = low order 12 bits of 15-bit creation date or 0 (0 implies current date).	E + 2	V	RH = low order 12 bits of 15-bit creation date (bits 24-35).
E + 3	I	-	E + 3	I	-

†A = argument from user program, V = value from monitor, I = ignored.

Unlike on disk, a DECtape RENAME works on the last file LOOKUPed and ENTERed for the device, not the last file for this channel. The UUO sequence required to successfully RENAME a file on DECtape is as follows:

```
LOOKUP    D,E
CLOSE D,
RENAME    D,E1

or

ENTER     D,E
CLOSE D,
RENAME    D,E1
```

6.1.5.4 INPUT, OUTPUT, CLOSE, RELEASE - When performing nondump input operations, the DECtape service routine reads the links in each block to determine what block to read next and when to raise the EOF flag.

When an OUTPUT is given, the DECtape service routine examines the left half of the third word in the output buffer (the word containing the word count in the right half). If this half contains -1, it is replaced with a 0 before being written out, and the file is thus terminated. If this half word is greater than 0, it is not changed and the service routine uses it as the block number for the next OUTPUT. If this half word is 0, the DECtape service routine assigns the block number of the next block for the next OUTPUT.

For both INPUT and OUTPUT, block 100 (the directory) is treated as an exception case. If the user's program gives

USETI D, 144_8

to read block 100, it is treated as a 1-block file.

The CLOSE operator places a -1 in the left half of the first word in the last output buffer, thus terminating the file.

The RELEASE operator writes the copy of the directory, which is normally kept in core onto block 100, but only if any changes have been made. Certain console commands, such as KJOB or CORE 0, perform an implicit RELEASE of all devices and, thus, write out a changed directory even though the user's program failed to give a RELEASE.

6.1.6 Special Programmed Operator Service

Several programmed operators are provided for manipulating DECtape. These UUOs allow the user to manipulate block numbers and to handle directories.

6.1.6.1 USETI D, E - The USETI programmed operator sets the DECtape on channel D to input block E next. Since the monitor reads as many buffers as it can on INPUT, it is difficult to determine which buffer the monitor is processing when the USETI is given. Therefore, the INPUT following the USETI may not obtain the buffer containing the block specified. However, if a single buffer ring is used, the desired block is retrieved since the device must stop after each INPUT. Alternatively, if bit 30 (IO.SYN) of the file status word is set via an INIT, OPEN, or SETSTS UUO, the device stops after each bufferful of data on an INPUT so that the USETI will apply to the buffer supplied by the next INPUT.

6.1.6.2 USETO D, E - The USETO programmed operator sets the DECtape on channel D to output block E next. With multiple-buffered I/O, the output following the USETO may not apply to the buffer containing the specified block, since the monitor transfers as many buffers as possible with each OUTPUT. Therefore, a single buffer ring should be used, or bit 30 (IO.SYN) of the file status word should be set. Refer to Paragraph 6.1.6.1.

6.1.6.3 UGETF D, E - The UGETF programmed operator places the number of the next free block of the file in the user's location E.

If UGETF is followed by an ENTER, the monitor modifies its algorithm in the following manner:

1) the first block is written nearest the front of the tape instead of nearest the directory.
2) the spacing factor is changed to 2 instead of 4 so that very large programs can fit almost entirely in a forward direction.

This feature allows user programs, such as PIP, to write SAV format files which can be read by the executive mode utility program TENDMP (see the DECsystem-10 Software Notebooks).

6.1.6.4 UTPCLR AC, or CALLI AC, 13 - The UTPCLR programmed operator clears the directory of the DECtape on the device channel specified in the AC field. A cleared directory has zeroes in the first 83 words except in the slots related to blocks 0, 1, 2, and 100_{10} and nonexistent blocks 1102 through 1105_8. Only the directory block is affected by UTPCLR. This programmed operator is a no-operation if the device on the channel is not a DECtape.

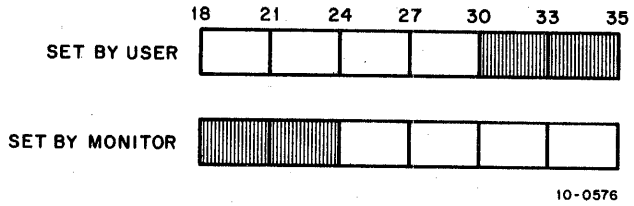
6.1.6.5 MTAPE D, 1 and MTAPE D, 11 - MTAPE D, 1 rewinds the DECtape and moves it into the end zone at the front of the tape. MTAPE D, 11 rewinds and unloads the tape, pulling the tape completely onto the left-hand reel, and clears the directory-in-core bit. These commands affect only the physical position of the tape, not the logical position. When either is used, the user's job can be swapped out while the DECtape is rewinding; however, the job cannot be swapped out if an INPUT or OUTPUT is done while the tape is rewinding.

6.1.6.6 DEVSTS UUO - After each interrupt, the DECtape service routine stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

6.1.7 File Status (Refer to Appendix D)

The file status of the DECtape is shown on the next page.

Standard Bits



Bit 18 - IO.IMP — An attempt was made to read block 0 in nonstandard dump mode.

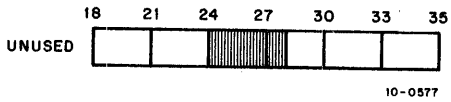
Bit 19 - IO.DER — Data was missed.

Bit 20 - IO.DTE — Parity error.

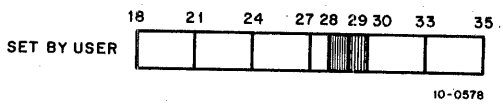
Bit 21 - IO.BKT — Block number is too large or tape is full on OUTPUT.

Bit 22 - IO.EOF — EOF mark encountered on input. No special character appears in buffer.

Bit 23 - IO.ACT — Device is active.



Device Dependent Bits



Bit 28 - IO.SSD — DECtape is in semi-standard I/O mode. The setting of this bit is recognized only if bit 29 (nonstandard I/O mode) is on. Semi-standard mode is similar to nonstandard mode except, 1) block numbers are checked for legality, and 2) the tape is started in the same direction as it was previously going.

Bit 29 - IO.NSD — DECtape is in a nonstandard-I/O mode format as opposed to standard-I/O mode. No file-structured operations are performed on the tape. Blocks are read or written sequentially; no links are generated (output) or recognized (input). The first block to be read or written must be set by a USETI or USETO. In nonstandard-I/O mode, up to 200_8 words per block are read or written as user data (as opposed to the standard mode of 1 link plus word count followed by 177_8 words). No dead reckoning is used on a search for a block number as the tape may be composed of blocks shorter than 200 words. The ENTER, LOOKUP, and UPTCLR UUOs are treated as no-ops. Block 0 of the tape may not be read or written in dump mode if bit 29 is on, because the data must be read in a forward direction and block 0 normally cannot be read forward.

6.1.8 Important Considerations

When positioning to a desired block on DECtape, the technique of dead reckoning is used. This means that the DECtape service routine starts the DECtape spinning and computes the time it should take to reach the desired block. Meanwhile, the service routine performs a service for another user, if any, and then returns just before the computed time has elapsed. If the desired block has not been reached, this process is repeated until it is successful. This technique is used to keep the controller free for other uses while the DECtape is spinning.

When an attempt is made to write on a write-locked tape or to access a drive that has no tape mounted, the message

DEVICE DTAn OPERATOR zz ACTION REQUESTED

is given to the user. When the situation has been rectified, CONT may be typed to proceed. However, if this message is output because of an attempt to write on a write-locked tape and any operation that causes a RESET to be performed (e.g., a GET or RUN command) is then executed, a RELEASE will be done on the DECtape. This RELEASE causes any attempt to write the directory to output the same message. To avoid the second output of the message, the user should ASSIGN the DECtape again thus causing the DECtape service routine not to write the directory on the RELEASE.

The DECtape service routine reads the directory from a tape the first time it is required to perform a LOOKUP, ENTER, or UGETF; the directory image remains in core until a new ASSIGN command is executed from the console. To inform the DECtape service routine that a new tape has been mounted on an assigned unit, the user uses an ASSIGN command. The directory from the old tape can be transferred to the new tape, thus destroying the information on that tape unless the user reassigns the DECtape transport every time he mounts a new reel.

Although DECtape is a file-structured blocked device, there is a limit to the number of files that may be opened simultaneously on a single DECtape. A given DECtape may be OPENed or INITed on two software channels (maximum) at the same time, once for INPUT and once for OUTPUT. An attempt to INIT on two channels for INPUT or two channels for OUTPUT generates no error indication, and only the most recent INIT is effective. This restriction explains why the following examples do not work.

Example 1:

```
.R FILCOM
*TTY:=DTA1:P1,DTA1:P2
```

FILCOM accepts the command string but the comparison does not work because the DECtape cannot be associated with the input side of two software channels at the same time.

Example 2:

```
.R MACRO
*DTA1:BIN,DTA1:LST←DTA2:PROG
```

MACRO accepts the command string but does not produce the desired results because a single DECtape cannot be associated with the output side of two software channels at the same time. However, the following example works, because only one file is opened for reading and one file for writing.

```
.R MACRO
*DTA1:BIN←DTA1:SOURCE
```

6.2 DISK

The device mnemonic is DSK, FHA, DPA; the buffer size is 203_8 (200_8 data) words.

6.2.1 Data Modes

6.2.1.1 Buffered Data Modes - Data is written on the disk exactly as it appears in the buffer. Data consists of 36-bit words.

CAUTION

All buffered mode operations utilize a 200 octal word data buffer. Attempts to set up non-standard buffer sizes are ignored. In particular, attempting to use buffer sizes smaller than 200 words for input result in data being read in past the end of the buffer destroying what information was there (e.g., the buffer header of the next buffer).

6.2.1.2 Unbuffered Data Modes - Data is read into or written from anywhere in the user's core area without regard to the normal buffering scheme. Control for read or write operations must be via a command list in core memory. The command list format is described in Chapter 4. The disk control automatically measures dump data into standard-length disk blocks of 200 octal words. Unless the number of data words is an exact multiple of the standard length of a disk block (200 words) after each command word in the command list, the remainder of that block is wasted.

6.2.2 Structure of Disk Files

The file structures of a disk system minimize the number of disk seeks for sequential or random access during either buffered or unbuffered I/O. The assignment of physical space for data is performed automatically by the monitor when logical files are written or deleted by user programs. Files may be any

length, and each user may have as many files as he wishes, as long as disk space is available and the user has not exceeded his logged-in quota. Users or their programs do not need to give initial estimates of file length or number of files. Files may be simultaneoulsy read by more than one user at a time, thus allowing data sharing. A new version of a file may be recreated by one user while other users continue to read the old version, thus allowing for smooth replacement of shared programs and data files. Finally, one user may selectively update portions of a file, rather than create a new one.

6.2.2.1 Addressing by Monitor - The file structure described in this section is generally transparent to the user, and a detailed knowledge of this material is not essential for effective user-mode use of the disk. One set of disk-independent file handling routines in the monitor services all disks and drums. This set of routines interprets and operates upon file structures, processed disk UUOs, queues disk requests, and makes optimization decisions. The monitor deals primarily with logical units within file structures and converts to physical units in the small device-dependent routines just before issuing I/O commands. All queues, statuses, and flags are organized by logical unit rather than by physical unit. The device-dependent routines perform the I/O for specific storage devices and translate logical block numbers to physical disk addresses.

All references made to disk addresses refer to the logical or relative addresses used by the system and not to any physical addressing scheme involving records, sectors, or tracks, that may pertain to a particular physical device. The basic unit that may be addressed is a logical disk block, which consists of 200_8 36-bit words.

6.2.2.2 Storage Allocation Table (SAT) Blocks - Unique to each file structure is a file named SAT.SYS. This file reflects the current status of every addressable block on the disk. Only the monitor can modify the contents of SAT.SYS as a result of file creation, deletion, or space allocation, although this file may be read by any user. The SAT file consists of bits indicating both the portion of file storage in use and the portion that is available. To reduce the size of SAT.SYS, each bit can be used to represent a contiguous set of blocks called a cluster. Monitor overhead is decreased by assigning and releasing file storage in clusters of blocks rather than single blocks.

If a particular bits is on, it indicates that the corresponding cluster is bad or nonexistent or has been allocated to a file. It may or may not contain data (i.e., files may contain allocated but unwritten clusters). If the bit is off, it indicates that the corresponding cluster is empty, or available to be written on.

It is recommended that cluster sizes should evenly divide blocks on a unit. In the 5.02 monitor, the refresher rounds up the number of clusters to the next highest full cluster. In the 5.03 and later monitors, the refresher truncates to the largest number of full clusters. With truncation, the last few

blocks are not included in the addressing space, but may be used for swapping; therefore, they are not part of SWAP.SYS even though they are in the swapping space. In addition, any bad blocks in the extra blocks are not included in SWAP.SYS.

6.2.2.3 File Directories - A directory is a file which contains as data pointers to other files on the disk. There are three levels of directories in each file structure:

a. The master file directory (MFD).
b. The user file directories (UFDs).
c. The sub-file directories (SFDs).[1]

The master file directory consists of two-word entries; the entries are the names of the user file directories on the file structure. The first word of each entry contains the project-programmer number of the user. The left half of the second word of each entry contains the mnemonic UFD in SIXBIT and the right half contains a pointer to the first cluster of the user file directory (see Figure 6-4). The main function of the master file directory is to serve as a directory of individual user file directories. A continued MFD is the MFDs on all file structures in the job's search list.

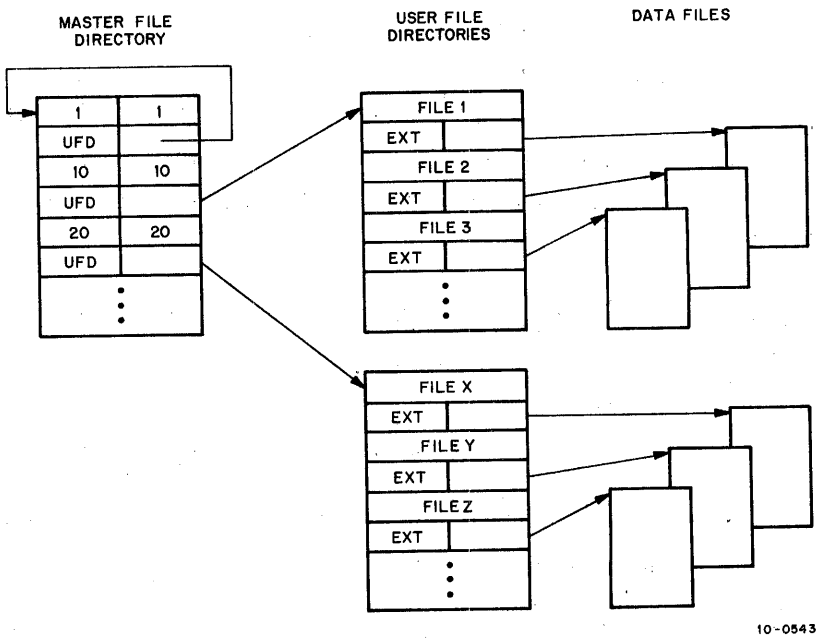


Figure 6-4 Basic Disk File Organization for Each File Structure

[1] Sub-file directories depend on FTSFD which is normally off in the DECsystem-1040.

The entries within a user file directory are the names of files existing in a given project-programmer number area within the file structure. The first word of each entry contains the filename in SIXBIT. The left half of the second word contains the filename extension in SIXBIT, and the right half contains a pointer to the first cluster of the file (see Figure 6-4). This pointer specifies both the unit and the super-cluster of the file structure in which the file appears. The right half of the directory entry is referred to as a compressed file pointer (CFP). A continued UFD is all the UFDs for the same project-programmer number on all file structures in the job's search list.

When the user is logged-in, each file structure for which he has a quota contains a UFD for his project-programmer number. Each UFD contains the names of all the user's files for that file structure only. UFDs are created only by privileged programs (i.e., LOGIN in response to a LOGIN command, and OMOUNT in response to a MOUNT command). A user is not prevented from attempting to read a file in another user's UFD on a file structure for which he does not have a UFD. Whether or not the user is successful depends on the protection specified for the file being referenced.

As an entry in the user file directory, the user can include a sub-file directory (SFD). The sub-file directory is similar to the other types of directories in that it contains as data all the names of files within the directory. This directory is pointed to by a UFD or a higher-level SFD nested in any arbitrary tree structure. The maximum number of nested SFDs allowed is defined via a MONGEN question and can be obtained from a GETTAB table (GETTAB table .GTLVD, item 17). Files can be written or read in SFDs nested deeper than the maximum but SFDs cannot be created. (There is an absolute maximum of 6, including the UFD.) Unlike UFDs, a sub-file directory can be created by any program. A continued SFD, or sub-directory, is all of the SFDs on all file structures in the job's search list with the same name and path.

This third level of directory allows groups of files belonging to the same user to be separate from each other. This is useful when organizing a large number of files according to function. In addition, simultaneous batch runs of the same program for a single user can use the same filenames without conflicting with each other. As long as the files are in different sub-file directories, they are unique.

A file is uniquely identified in the system by a file structure name, a directory path, a filename and an extension. The directory path is an ordered list of directory names, starting with a UFD, which uniquely specifies a directory without regard to a file structure. The PATH. UUO is used to set or read the default directory path for a job (refer to Paragraph 6.2.9.1). Default paths can be a job's UFD, an SFD in a job's UFD, a UFD different from the job's UFD, or an SFD in another UFD. If a default path is not specified, it is the job's UFD. The notation FILE.EXT [PPN,A,B,...,N] designates the file named FILE.EXT in the UFD [PPN] in the SFD N, which is in the SFD..., which is in the SFD A. The path to the file named FILE.EXT is [PPN,A,B,...,N].

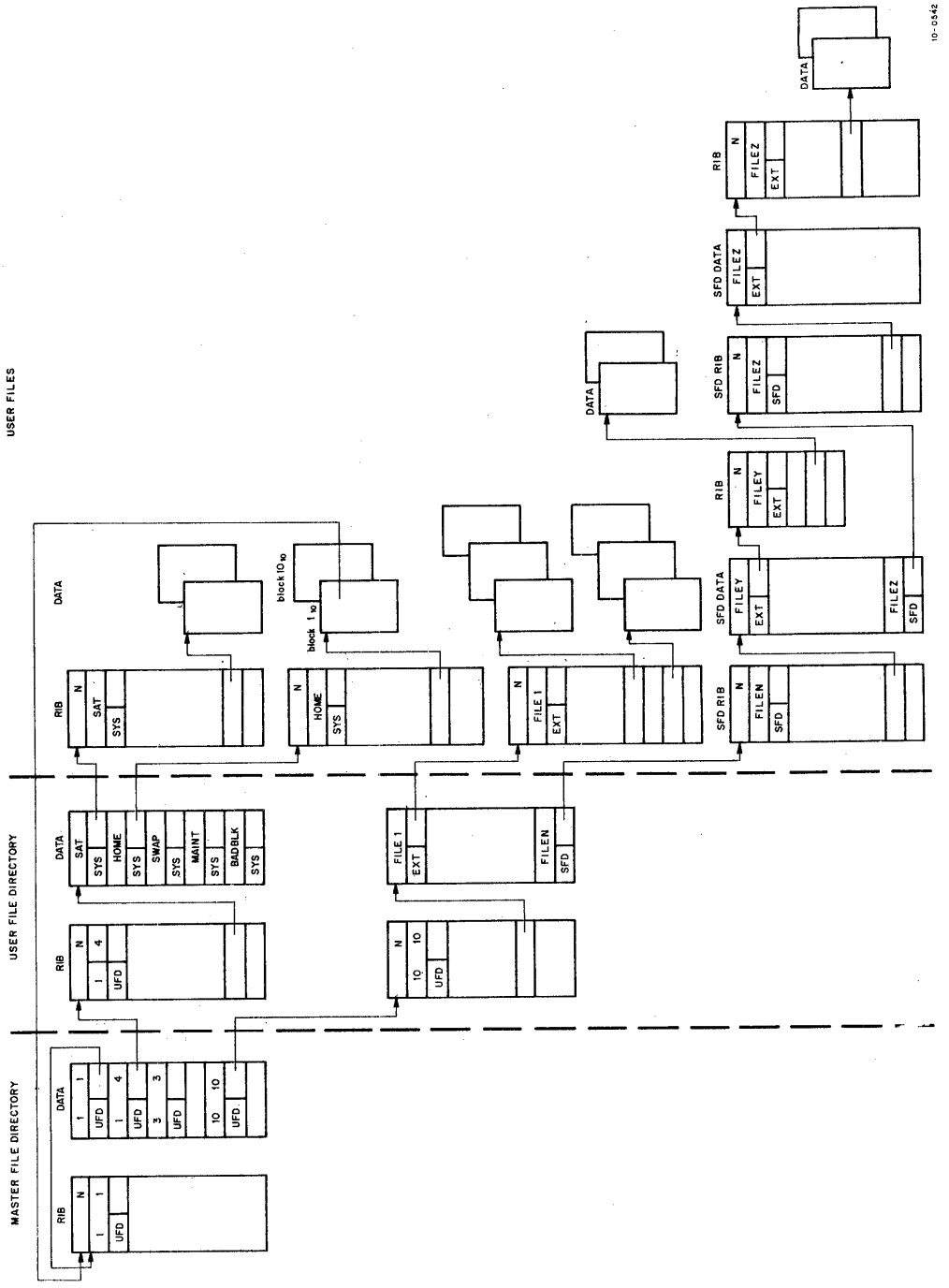


Figure 6-5 Disk File Organization

To improve disk access and core searching times, only UFD names are kept in the MFD (project-programmer number 1,1). All system programs and monitor file structure files are contained in another project-programmer number directory called the system library. For convenience both to users typing commands and to user programs, device name SYS is interpreted as the system library; therefore, no special programming is required to read as a specific file from device SYS. In command strings, the abbreviation SYSx: represents the system library on file structure DSKx; i.e., SYSA: represents the system library on DSKA.

6.2.2.4 File Format - All disk files (including directories) are composed of two parts:

a. pure data.
b. information needed by the system to retrieve this data.

Each data block contains exactly 200_8 words. If a partially filled buffer is output to the disk by a user, a full block is written with trailing zeros filling in to make 200_8 words. A partial block input later appears to have a full 200_8 data words. Word counts associated with individual blocks are not retained by the system except in the case of the last block of the file.

There are three links in the chain by which the system references data on the disk. This chain is transparent to the user, who might look on the directory as having four-word entries analogous to DEC-tapes. The first link is the two-word directory entry that points to the second link, the retrieval information block (RIB). The RIB, in turn, points to the third link, the individual data blocks of the file (see Figure 6-5).

The retrieval block contains all the pointers to the entire file. Retrieval information associated with each file is stored and accessed separately from the data; therefore, system reliability is increased because the probability of destroying the retrieval information is reduced. System performance is improved because the number of positionings necessary for random access is reduced.

For recovery purposes, a copy of the retrieval information block is written immediately after the last data block of the file when a CLOSE is completed. If the first RIB is lost or bad, the monitor can recover by allowing a recovery program to use the second RIB; therefore, a data file of n blocks has two additional overhead blocks: relative block 0, containing the primary RIB; and relative block n + 1, containing the redundant RIB (refer to Appendix H).

6.2.3 Access Protection

Nine bits of the retrieval information of a file are used to indicate the protection of that file. This protection is necessary because the disk is shared by many users, each of whom may desire to keep certain files from being written on, read, or deleted by other users. The nine bits are divided into

three classes because the users are divided into three categories: 1) the owner of the file, 2) the users with the same project number as the owner, and 3) all other users.

Ordinarily, the owner of a file is any user whose programmer number is the same as the programmer number of the UFD containing the file, regardless of whether the two project numbers match. Therefore, in order to maintain only one owner for each file, the installation should not assign the same programmer number to different users, no matter how many projects the installation has. A user working on more than one project, but having the same programmer number, can reference all his files as an owner under each of his project-programmer numbers.

However, some installations may decide that a user is the owner of a file only when both the project and programmer numbers under which the user is logged in match the pair identifying the UFD. If this is the case, the same programmer number can be assigned to different users in different projects. This allows the task of assigning programmer numbers to be delegated to project leaders without concern for duplication since the project numbers will be different from one project to another. However, a user working on more than one project cannot have the same owner access to all files that he has written.

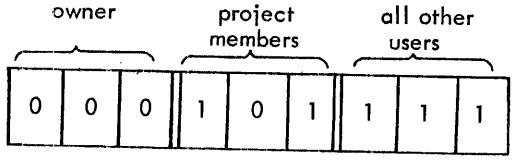
The definition of the owner of a file is specified at monitor generation time with MONGEN(INDPPN). No matter how the installation defines an owner, project numbers 0 to 7 are always independent of the project-programmer number (i.e., a user with a project number from 0 to 7 is considered the owner of all files with that project number).

A member of the owner's project is any user whose logged-in-project number is the same as the owner's, regardless of his programmer number.

The three bits associated with each category of users are encoded as follows:

Code	Access Protection
7	Greatest protection, which means no access privileges. However, the owner may LOOKUP the file so that he can change the protection to a less restrictive code via a RENAME. Thus for the owner, this code is equivalent to codes 6 and 5.
6	Execute-only. This disables user meddling and examining (DUMP, DCORE, D, E, SAVE, SSAVE, START n, CSTART n, DDT, COREn) with the error message ?ILLEGAL WHEN EXECUTE ONLY. An error return is given on a LOOKUP to an execute-only file to all users except the owner of the file.
5	Read, execute.
4	Append, read, execute.
3	Update, append, read, execute.
2	Write, update, append, read, execute.
1	Rename, write, update, append, read, execute.
0	Change protection, rename, write, update, append, read, execute.

The following example illustrates the nine-bit protection field of a file that has a protection of 057.



This code means:

1) The owner has complete privileges (code 0).
2) The project members have read and execute privileges (code 5).
3) All other users have no access privileges (code 7).

The greatest protection a file can have is 7, and the least is 0. Usually the owner's field is 0 or 1. However, it is always possible for the owner of a file to change the access protection associated with the file even if the owner-protection is not set to 0. Thus codes 0 and 1 are equivalent when they appear in the owner's field. Access protection can be changed by executing a RENAME UUO or by using the PROTECT monitor command as follows:

PROTECT file.ext <nnn >

When an ENTER UUO specifies a protection code of 000 and the file does not exist, the monitor substitutes the standard protection code as defined by the installation. The normal system standard is 057. This protection prevents users in different projects from accessing another user's files; however, a standard protection of 055 is recommended for in-house systems where privacy is not as important as the capability of sharing files among projects. No program should be coded to assume knowledge of the standard protection. If it is necessary to use this standard, it should be obtained through the GETTAB UUO.

To preserve files with LOGOUT, a protection code of 1 in the owner's field should be associated with the files. LOGOUT preserves all files in a UFD for which the protection code for the owner is greater than zero. The PRESERVE monitor command can be used to obtain a protection code of 1 in the owner's field.

6.2.3.1 UFD and SFD Privileges - The protection code associated with each file completely describes the access rights to that file independently of the protection code of the UFD. UFDs and SFDs may be read in the same manner as files but cannot be written explicitly, because they contain RIB pointers to particular disk blocks. For UFD and SFD privileges, users are divided into the same three categories as for files. Each category has three independent bits:

Bit	Access Privileges
4	Allow LOOKUPs in UFD or SFD.
2	Allow CREATEs in UFD or SFD.
1	Allow the UFD or SFD to be read as a file.

The owner is permitted to control access to his own UFD and SFD. It is always legal for the owner to issue a RENAME to change the protection of his directories. Any program can create or delete SFDs; however, only privileged programs are allowed to create, supersede, or delete a UFD. The monitor checks for the following types of privileged programs:

a. Jobs logged in under project-programmer number [1,2] (FAILSAFE).
b. Jobs running with the JACCT bit set in JBTSTS (LOGIN, LOGOUT).

Privileged programs are allowed to:

a. Create UFDs (and SFDs).
b. Delete UFDs (and SFDs).
c. Set privileged LOOKUP, ENTER, and RENAME arguments.
d. Ignore file protection codes.

UFD and SFD privileges are similar with the exception being that SFDs can be RENAMEd and deleted by both privileged programs and the owner of the SFD if his protection byte is 7.

6.2.4 Disk Quotas[1]

Each project-programmer number in each file structure is associated with two quotas that limit the number of blocks that can be stored under the UFD in the particular file structure. The quotas are:

a. Logged-in quota.
b. Logged-out quota.

When the user logs in, he automatically starts using his logged-in quota. Because this is not a guaranteed amount of space, the user competes with other users for it. The logged-out quota is the amount of space that the user must be within in order to log off the system. Normally, the logged-out quota is less than or equal to the logged-in quota, so that the user must delete temporary files.

If a user exceeds his logged-in quota, the monitor types the following message:

[EXCEEDING QUOTA ON fs]

where fs is the name of the file structure. The message appears in square brackets (like the TECO core expansion message) to suggest a warning rather than an error. Unlike most monitor messages, this message indicates that the user program may continue to run, and the console remains in user mode. The user program can no longer create or supersede files (ENTER gives an error return). Files already ENTERed are allowed to continue for a specified number of blocks. This amount is called the overdrawn amount and is a parameter of the file structure. The overdrawn amount specifies the number of blocks by which the logged-in UFD may exceed its logged-in quota. When the user exceeds the overdrawn amount, the IO.BKT bit is set, and further OUTPUTs are not allowed. A CLOSE operates successfully, including the writing of the last buffers and the RIBs.

When the user logs in, the LOGIN program reads the logged-in quota from the file AUXACC.SYS for all public file structures in which the user is allowed to have a UFD. This information is passed to the monitor where it is kept in core. If the quota has changed since the user logged in last, LOGIN updates (or creates) the RIB of each UFD with the new quotas.

[1] Quota checking depends on FTDQTA which is normally off in the DECsystem-1040.

6.2.5 Simultaneous Access

In its core area, the monitor maintains two four-word blocks called access blocks. These blocks control simultaneous access to a single file by a number of user channels. All active files have access blocks that contain file status information. The access blocks ensure that a maximum of one user channel supersedes or updates a given file at a given time.

6.2.6 File Structure Names

Each file structure has a SIXBIT name specified by the operator at system initialization time. This name can consist of four or less alphanumeric characters and must not duplicate any device, unit, or existing file structure name or its abbreviation. The recommended names for the file structures in the public pool are DSKA, DSKB, ..., DSKN (in order of decreasing speed).

When a specific file structure is INITed (e.g., DSKA), LOOKUP and ENTER searches are restricted to that file structure. Usually a channel is INITed with the generic name DSK, in which case all file structures in the active search list of the job are searched (refer to Paragraph 6.2.7).

6.2.6.1 Logical Unit Names - When a single file structure name is specified, the set of all the units in that file structure is implied; however, it is possible to specify a particular logical unit within a file structure (e.g., DSKA0, DSKA1, DSKA2 are three logical units in the file structure DSKA). The monitor deals with file structures rather than with individual units; therefore, when reading files, specifying a logical unit within a file structure is equivalent to specifying the file structure itself. The monitor locates the file regardless of which unit it is on within a file structure. However, in writing a file, the monitor uses the logical unit name as a guide in allocating space and will, if possible, write the file on the unit specified. In this way, a user can apportion files among different units for increased throughput.

6.2.6.2 Physical Controller Class Names - In addition to DSK, single file structure names (DSKA), and logical unit names (DSKA0), it is possible to specify a class of controllers. If the system has one controller of the type specified, the result is the same as if the user had specified the physical controller name. The controller classes supported by DEC are:

DR (future drum), FH, DP

6.2.6.3 Physical Controller Names - It is possible to specify any of the units on a particular controller. The monitor relates that name to the file structures, which contain at least one unit on the specified controller. More than one file structure may be specified when a physical controller name is used. The controllers that DEC supports are:

DRA, DRB (future drum), FHA, FHB, DPA, DPB

6.2.6.4 Physical Unit Names - When a physical controller name is specified, all units on that controller are implied. It is possible to specify a physical unit name on a particular controller. The physical unit names that DEC supports are:

DRA0, DRB0	Reserved for future drum (RX10).
FHA0, ..., FHA3	Mixture of Burroughs fixed-head disks (RD10) and Bryant drums (RM10B) on RC10 control.
FHB0, ..., FHB3	Mixture of Burroughs fixed-head disks (RD10) and Bryant drums (RM10B) on second RC10 control.
DPA0, ..., DPA7	Mixture of RP02 and RP03 disk packs on RP10 control.
DPB0, ..., DPB7	Mixture of RP02 and RP03 disk packs on second RP10 control.

6.2.6.5 Unit Selection on Output - If the user specifies a file structure name on an ENTER, the monitor chooses the emptiest unit on the file structure which does not currently have an open file (UFD's are not considered opened) for the job. This selection improves disk throughput by distributing files for a particular job on different units. For example, in a MACRO assembly with two output files and one input file, it is probable that the monitor would allocate the output files on units separate from each other and from the input file. If this were the only job running, there would be almost no seeks. Therefore, to take advantage of this, programs should LOOKUP input files before ENTERing output files.

6.2.6.6 Abbreviations - Abbreviations may be used as arguments to the ASSIGN command and the INIT and OPEN UUOs. The abbreviation is checked for a first match when the ASSIGN, INIT, or OPEN is executed. The file structure or device eventually represented by the particular abbreviation depends on whether a LOOKUP or ENTER follows. A LOOKUP applies to as wide a class of units as possible, whereas an ENTER applies to a restricted set to allow files to be written on particular units at the user's option. For example, consider the following configuration:

File Structure		Physical Unit
DSKA	=	FHA0, FHA1, FHA2
DKSB	=	FHB0, FHB1
DSKC	=	DPA0, DPA1, DPA2, DPA3
DSKD	=	DPB0, DPB1, DPB2
PRVA	=	DPB3

Table 6-5 shows the file structures and units implied by the various names and abbreviations.

Table 6-5
File Structure Names

Argument Supplied to ASSIGN, MOUNT, INIT, OPEN	File Structures or Units Implied	
	LOOKUP	ENTER
D, DS, DSK	Generic DSK according to job search list (refer to Paragraph 6.2.7)	
P, PR, PRV, PRVA	DPB3	DPB3
F, FH, FHA	DSKA, DSKB	FHA0
FHB	DSKB	FHB0
FHA0	DSKA	FHA0
FHB0	DKSB	FHB0
DP	DKSC, DSKD, PRVA†	DSKC
DPA	DSKC	DSKC
DPB	DSKD, PRVA†	DSKD
DPA0	DSKC	DPA0
DPB2	DSKD	DPB2
DPB3	PRVA	PRVA
†Only if user has done a MOUNT.		

6.2.7 Job Search List

To a user, a file structure is like a device; that is, a file structure or a set of file structures may be specified by an INIT or OPEN UUO or by the first argument of the ASSIGN or MOUNT command. A console user specifies a file structure by naming the file structure and following it with a colon.

There is a flexible naming scheme that applies to file structures; however, most user programs INIT device DSK, which selects the appropriate file structure, unless directed to do otherwise by the user. The appropriate file structure is determined by a job search list. A job search list is divided into two parts:

a. an active search list (usually referred to as the job search list), and
b. a passive search list.

The active search list is an ordered list of the file structures that are to be searched on a LOOKUP or ENTER when device DSK is used. The passive search list is an unordered list of file structures maintained by the monitor for LOGOUT time. At this time, LOGOUT requires that the total allocated

blocks on each UFD in both the active and passive search lists be below the logged-out quota. Each job has its own active search list (established by LOGIN) with file structures in the order that they appear in the administrative control file AUXACC.SYS. Thus, a user has a UFD for his project-programmer number in each file structure in which LOGIN allows him to have files. With the MOUNT command, mounted file structures may be added to the active search list. The following is an example of a search list:

DSKB, DSKA, FENCE, DSKC

DSKB and DSKA comprise the active search list. These file structures are represented by generic name DSK for this job. DSKC is the name of a file structure that was previously in the active search list. FENCE represents the boundary between the active and passive search list.

Each file structure in a job search list may be modified by setting one of two flags with the JOBSTR UUO:

a. Do not create in this structure if just generic DSK is specified.
b. Do not write in this structure.

Setting the "do not create" flag indicates that no new files are to be created on this file structure unless explicitly stated. For example: if the "don't create" flag is set

DSKA: FOO ←

allows FOO to be created on DSKA, but

DSK: FOO ←

does not. For LOOKUPs on device DSK, the monitor searches the structures in the order specified by the job search list. For ENTERs when the filename does not exist (creating, see below), the file is placed on the first file structure in the search list that has space and does not have the "do not create" flag set. For ENTERs when the filename already exists on any file structure in the search list (superseding, see below), the file is placed on the same structure that contains the older file. If the write-lock bit is set for the file structure, a write-lock error (ERWLK%) is given on the supersede. Because superseding is treated differently from creating, a user may explicitly place a file on a particular file structure, for example, a fast one with the do not create bit set, so that subsequent supersedes will remain on that file structure even though generic DSK is used.

6.2.8 User Programming

Three types of writing on the disk may be distinguished. If a user does an ENTER with a filename which did not previously exist in his UFD, he is said to be creating that file. If the filename previously existed in his UFD, he is said to be superseding that file; the old version of the file stays on the disk

(and is available to anyone who wants to read it) until the user does the output CLOSE. At the time of the CLOSE, the user's UFD is changed to point to the new version of the file and the old version is either deleted immediately or marked for deletion later if someone is currently reading it; the space occupied by deleted files is always reclaimed in the SAT tables (refer to Paragraph 6.2.2.2). Finally, if a user does a LOOKUP followed by an ENTER (the order is important) on the same filename on the same user channel, he will be able to modify selected blocks of that file, using USETO and USETI UUOs (refer to Paragraph 6.2.9.2) without creating an entirely new version; this third type of writing, called updating, eliminates the need to copy a file when making a small number of changes. A LOOKUP followed by an ENTER and OUTPUT (in that order) writes the output at the beginning of the file. To append information to the file, a USETI -1 is used before the OUTPUT.

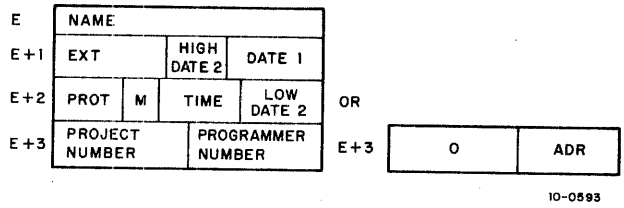
As a standard practice, user programs should read, create, and supersede (new file with same filename) files on different user channels. However, for compatibility with DECtapes, it is possible to read and create, or read and supersede, two files on the same user channel as long as all OUTPUTs and the CLOSE output are done before the LOOKUP and the first input, or vice versa. In other words, a CLOSE UUO is required between successive LOOKUPs and ENTERs unless updating is intended.

The actual file structure of the disk is generally transparent to the user. In programming for I/O on the disk, a format analogous to that of DECtapes is used; that is, the user assumes a four-word directory entry similar in form to the first four words of retrieval information. The UUO format is approximately the same as for DECtapes:

UUO D,E

where UUO is an I/O programmed operator, and D specifies the user channel associated with this device. E points either to a four-word directory entry or an extended argument block in the user's program.

6.2.8.1 Four-Word Arguments for LOOKUP, ENTER, RENAME UUOs - The four-word argument block has the following format:



where

NAME is the filename in SIXBIT, or, if a UFD, is the project number in the left half and the programmer number in the right half.

EXT is the filename extension in SIXBIT ASCII. (continued on next page)

HIGH DATE 2 contains the high order 3 bits of the date on which the file was originally created (bits 18-20).

DATE 1 is the date the file was last referenced (RENAME, ENTER, or INPUT) in the format of the DATE UUO (bits 21-35).

PROT is the protection code for the file (bits 0-8).

M is the data mode (ASCII, binary, dump) (bits 9-12).

TIME is the time that the file was originally created, represented as the number of minutes past midnight of the creation date (bits 13-23).

LOW DATE 2 is the low order 12 bits of the date (in the same format as the DATE UUO) on which the file was originally created (bits 24-35).

NOTE

The two-part format for DATE 2 (creation date) is used to maintain compatibility with monitors and media as old as 1964.

The programmed operators (UUOs) operate as follows:

a. ENTER UUO - ENTER D, E causes the monitor to store the four-word directory entry for later entry into the proper UFD or SFD when user channel D is closed or released.

NAME	The filename must be nonzero; otherwise, an error return results.
EXT	The filename extension may be zero; if so, the monitor leaves it as zero.
HIGH DATE 2	If a nonzero date is obtained by concatenating the high order 3 bits in this field with the low order 12 bits in LOW DATE 2, then the monitor uses that value as the creation date for the file. If the date is zero, the monitor supplies the high order digits of the current date from the overflow in E + 2.
DATE 1	The date may be zero, in which case the monitor substitutes the current date. The date must not be in the future; if this is so, the current date is used.
PROT	If the protection code is 0, the monitor substitutes the installation standard as specified at MONGEN time. If the protection code is 0 and this ENTER is superseding a file, the protection of the new file is copied from the old file. RENAME may be used to change the protection after a file has been completely written and when it is being closed.
M	The data mode is supplied by the monitor. It was set by the user in the last INIT or SETSTS UUO on channel D.

TIME, LOW DATE 2	If these are 0, and bits 18-20 of E + 1 are zero the monitor supplies the current date and time as the creation date and time for the file. The high order digits of the creation date overflow to bits 18-20 of E + 1 (HIGH DATE 2). If either is nonzero, the monitor uses the HIGH DATE 2 supplied by the user in E + 1 and the TIME and LOW DATE 2 supplied in E + 2. Thus, files may be copied without changing the original creation time and date.
PROJECT NUMBER PROGRAMMER NUMBER	If this word is 0, the file will be written in the default directory. (For example, if the default path is [10, 10, A], the file will be written in SFD A which is contained in [10, 10] .UFD.) The default path is determined by the PATH. UUO (refer to Paragraph 6.2.9.1). If a default path has not been specified via the PATH. UUO, it is the job's UFD (i.e., the project-programmer number under which the user is logged in). If this word is a project-programmer number, the file will be written in the UFD specified (i.e., sub-directories will not be scanned). This allows the program to write in the disk area under which the job is logged in although the default directory is different. Note that it is generally not possible to create (ENTER) files in another user's area of the disk, because UFDs are usually protected from all but the owner when creating files. If this word is XWD 0, ADR, the file will be written according to the path specified by ADR. The argument block beginning at ADR is the same as in the PATH. UUO (refer to Paragraph 6.2.9.1) except that the first two arguments (ADR and ADR + 1) are ignored. The scan switch (ADR + 1) is not needed since if the file is found in the specified directory, it will be superseded, and if not found, it will be created at the end of the path of the specified directory, even if a file with the same name appears in an upper-level directory. A path specification in the ENTER block overrides any default path specification given in the PATH. UUO.

With certain types of error returns peculiar to the disk, the right half of E + 1 is set to a specific number to indicate the error that caused the return. For example, if the extension UFD is specified and bit 18 (RP.DIR) of the file status status word is not set, the right of E + 1 is set to 2 (protection failure). Refer to Paragraph 6.2.8.3 for a special note on error recovery. Refer to Appendix E for the error codes returned on the ENTER UUO.

When issuing a supersede ENTER (an ENTER after a LOOKUP on the same channel), the user should check that locations E through E + 3 are as he desires.

When an ENTER is executed by the monitor on a file that exists, a new file by that name is written, and those bits in the SAT blocks that correspond to the blocks of the old file are zeroed when the CLOSE (or RELEAS) UUO is executed provided that bit 30 of the CLOSE is 0 (refer to Paragraph 4.7.7). Space is thereby retrieved and available to other users after the new file has been successfully written. If a file structure is INITed on channel D, the monitor maximizes the job's throughput by selecting the emptiest unit for which the job has no opened files (refer to Paragraphs 6.2.6.5 and 6.2.6.6).

b. LOOKUP UUO - LOOKUP D, E causes the monitor to read the appropriate UFD or SFD. If a later version of the file is being written, the old version pointed to by the UFD is read.

NAME	The same as on an ENTER.
EXT	The same as on an ENTER.
DATE 1, PROT, M, TIME, LOW and HIGH DATE 2	These arguments are ignored. The monitor returns these quantities to the user in E + 1 and E + 2.
PROJECT NUMBER PROGRAMMER NUMBER	If this word is 0, the file will be read from the user's default directory path. The entire path is searched only if the scan switch is set via the PATH. UUO (refer to Paragraph 6.2.9.1). If a default path has not been specified, it is the project-programmer number under which the user is logged in. If a project-programmer number is specified, the file will be read from the UFD specified (i.e., sub-directories will not be scanned). Thus, it is possible to read files in other user's directories, provided the file's protection mask permits reading and the UFD permits LOOKUPs. If this word is XWD 0, ADR, the file will be read according to the path specified by ADR. The argument block beginning at ADR is the same as in the PATH. UUO except that the first argument is ignored, and the second argument, if 0, uses the default value of the scanning switch (refer to the PATH. UUO). A path specification in the LOOKUP block overrides any default path specification given in the PATH. UUO.

The monitor returns the negative word count (or positive block count for files larger than 2^{17} words) in the LH of E + 3, 0 in RH of E + 3 when the four-word argument block is given. As a result, the monitor treats a negative project-programmer number as if it were 0, however, this will not always be true; therefore, programs must be written to either clear E + 3 before doing a LOOKUP, ENTER, or RENAME or set E + 3 to the desired project-programmer number. In the future, a negative project-programmer number may be used to indicate SIXBIT alphabetic characters for project and programmer initials.

The numbers placed in the RH of E + 1 on an error return have a significance analogous to that described for the ENTER UUO (refer to Appendix E).

If the file is currently being superseded, the old file is used.

c. RENAME UUO - RENAME D, E is used to alter the filename, the filename extension and/or protection of a file, or to delete a file from the disk. This UUO can be used to change the name of an SFD, but an attempt to change the extension or project-programmer number associated with an SFD, the name, extension, or project-programmer number associated with a UFD, or the project-programmer number of a device with an implied project-programmer number (e.g., SYS:, NEW:, OLD:) results in a protection error. To RENAME a file, a LOOKUP or ENTER must first be done to identify the file for the RENAME UUO. Locations E through E + 2 are as described for ENTER. If E + 3 = 0, there is no change in the directory of the file. If E + 3 is the default project-programmer number, the file is renamed in that UFD. If E + 3 has a different project-programmer number than the one in which the file is LOOKUPed or ENTERed (i.e., E + 3 is not the default project-programmer number), the monitor deletes the directory entry from the old directory (UFD or SFD) and inserts the directory entry into the specified UFD, provided the user has the privileges to delete files from the old directory, and to create files in the new UFD. (This is an efficient way to move a file from one UFD to another, since no I/O needs to be done on the data blocks of the file.) If E + 3 = XWD 0, ADR, the file is renamed into a new SFD or UFD according to the path specified by ADR. (Refer to the PATH.UUO.) Therefore, the only way to RENAME a file into a SFD different from the one which it is in to give an explicit path via an argument block.

A CLOSE is optional because RENAME performs a CLOSE. However, a CLOSE should not be issued between a LOOKUP and RENAME if the file is not in the default path or cannot be obtained from the default path by scanning because CLOSE erases all memory of the path of a file. If a CLOSE is performed and the file is not in the default path, the RENAME returns the FILE NOT FOUND error. In addition, disk accesses are minimized if a CLOSE does not precede a RENAME.

RENAME enters the information specified in E through E + 2 into the retrieval information and proper directory. If the contents of E is zero, RENAME has the effect of deleting the file. Although only a privileged job can delete a UFD, any job can delete an SFD. If the directory is not empty or if a job is currently using the directory, the RENAME returns the DIRECTORY NOT EMPTY error. (Refer to Appendix E for the error codes.) Refer to Paragraph 6.2.8.3 for a special note on error recovery.

When issuing a RENAME UUO, the user must ensure that the status at locations E through E + 3 are as he desires. An ENTER or LOOKUP must have preceded the RENAME; therefore, the contents of E through E + 3 will have been altered, or filled if the E is the same for all UUOs.

d. Examples - The sample code below can be used to assemble the 15-bit creation date of a disk (or DECtape) file in register T1 after a successful LOOKUP. The four-word argument block begins at location E.

```
LDB     T1, [POINT 12, E+2, 35]     ;GET LOW-ORDER PART
LDB     T2, [POINT 3, E+1, 20]      ;GET HIGH-ORDER PART
DPB     T2, [POINT 3, T1, 23]       ;MERGE THE TWO PARTS
```

The following sample code illustrates setting the 15-bit creation date in the four-word ENTER argument block from the value in register T1.

```
DPB     T1, [POINT 12, E+2, 35]     ;STORE LOW-ORDER PART
ROT     T1, -†D12                   ;POSITION HIGH PART
DPB     T1, [POINT 3, E+1, 20]      ;STORE HIGH-ORDER PART
```

6.2.8.2 Extended Arguments for LOOKUP, ENTER, RENAME UUOs - A number of quantities have been added to the existing four-word block. The user program may specify exactly the number of words in the argument block. If the left half of E is 0 and the right half of E is three or greater, the right half of E is interpreted as the count of the number of words which follow. If the right half of E is less than three, a file-not-found return is given because the user program is not supplying enough arguments. Allowed arguments supplied by the user program are returned by the monitor as values. If the user program supplies arguments that are not allowed, the monitor ignores these arguments and supplies values on return. Table 6-6 indicates the arguments that may be supplied by a user program.

Table 6-6
Extended LOOKUP, ENTER, and RENAME Arguments

Rel. Loc	Symbol	Lookup	Create Supers	Update Rename	Arguments and Value
0	.RBCNT	A	A	A	Count of arguments following
1	.RBPPN	A0	A0	A0	Directory name (project-programmer no.) or pointer
2	.RBNAM	A	A	A	Filename in SIXBIT
3	.RBEXT	A	A	A	File extension (LH)
		V	A0	A	High order 3 bits of 15-bit creation date (bits 18-20)
					Access date (bits 21-35)
4	.RBPRV	V	A0	A	Privilege (bits 0-8)
		V	V	A	Mode (bits 9-12)
		V	A0	A	Creation time (bits 13-23)
		V	A0	A	Low order 12 bits of 15-bit creation date (bits 24-35)
5	.RBSIZ	V	V	V	Length of file in data words written (+no. words)
6	.RBVER	V	A	A	Octal version number (36 bits)
7	.RBSPL	V	A	A	Filename to be used in output spooling.
10	.RBEST	V	A	A	Estimated length of file (+no. blocks)
11	.RBALC	V	A	A	Highest relative block number within the file allocated by user or monitor to file.

A = Argument (supplied by privileged or nonprivileged user program) and returned by monitor as a value.

A0 = Argument like A with the addition that a 0 argument causes the monitor to substitute a default value.

V = Value (returned by monitor) cannot be set even by privileged program, monitor ignores argument.

A1 = Argument if privileged program (ignored if nonprivileged).

(continued on next page)

Table 6-6 (Cont)
Extended LOOKUP, ENTER, and RENAME Arguments

Rel. Loc	Symbol	Lookup	Create Supers	Update Rename	Arguments and Value
12	.RBPOS	V	A	A	Logical block no. of first block to allocate within F.S.
13	.RBFT1	V	A	A	Future nonprivileged argument - reserved for DEC
14	.RBNCA	V	A	A	Nonprivileged argument reserved for customer to define
15	.RBMTA	V	A1	A1	Tape label if on backup tape
16	.RBDEV	V	V	V	Logical unit name on which the file is located
17	.RBSTS	V	A1	A1	1) LH=Combined status of all files in UFD 2) RH=Status of this file
20	.RBELB	V	V	V	Bad logical block within error unit
21	.RBEUN	V	V	V	1) LH=Logical unit no. within F.S. of bad unit (0,,,N).
		V	V	V	2) RH=No. of consecutive blocks in bad region
22	.RBQTF	V	A1	A1	(UFD-only) FCFS logged-in quota in blocks
23	.RBQTO	V	A1	A1	(UFD-only) logged-out quota in blocks
24	.RBQTR	V	A1	A1	(UFD-only) reserved logged-in quota
25	.RBUSD	V	A1	A1	(UFD-only) no. of blocks used at last logout
26	.RBAUT	V	A1	A1	Author project-programmer number (creator or superseder)
27	.RBNXT	V	A1	A1	Next file structure name if file continued
30	.RBPRD	V	A1	A1	Predecessor file structure name if file continued
31	.RBPCA	V	A1	A1	Privileged argument word reserved for each customer to define as he wishes
32	.RBUFD	V	V	V	Logical block number within F.S. (not cluster no.) of the RIB of the UFD in which the name of this file appears
33	.RBFLR	V	V	V	Relative block number in file of first block in RIB
34	.RBXRA	V	V	V	Extended RIB address
35	.RBTIM	V	V	V	Creation date in universal date-time standard (refer to Paragraph 3.6)

A = Argument (supplied by privileged or nonprivileged user program) and returned by monitor as a value.

A0 = Argument like A with the addition that a 0 argument causes the monitor to substitute a default value.

V = Value (returned by monitor) cannot be set even by privileged program, monitor ignores argument.

A1 = Argument if privileged program (ignored if nonprivileged).

The following explanation is a more complete description of the terms used in Table 6-6.

.RBPPN — LH=octal project number (right-justified).
RH=octal programmer number.
The project-programmer number is of the UFD in which the file is to be LOOKedUP, ENTERed, or RENAMEd. To LOOKUP the MFD, .RBPPN must contain a 1 in the left half and a 1 in the right half indicating that the filename (.RBNAM) is to be LOOKedUP in project 1, programmer 1's UFD (the MFD).

.RBNAM — SIXBIT filename, left justified with trailing nulls. If the MFD or UFD is being LOOKedUP, ENTERed, or RENAMEd, this location contains the project-programmer number. If a SFD is being LOOKedUP, ENTERed, or RENAMEd, this location contains the directory name. The argument can be 0 only on a RENAME, in which case the file is deleted. If the filename is not left justified on ENTER, most programs are unsuccessful on a subsequent LOOKUP. The monitor cannot left-justify the argument because it may be an octal project-programmer number.

.RBEXT — LH=SIXBIT filename extension, left justified with trailing nulls. Null extensions are discouraged because they convey no information. If the extension is not left justified on ENTER, most programs are unsuccessful on a subsequent LOOKUP. RH, bits 18-20 = high order 3 bits of 15-bit creation date, bits 21-35=access date in standard format. If an error return is given, bits 18-35 are set to an error code by the monitor before the error (no skip) return is taken. Refer to Paragraph 6.2.8.3 for a special note on error recovery.

.RBPRV — Bits 0-8=protection codes. (RB.PRV)
Bits 9-12=data mode in which file is created. (RB.MOD)
Bits 13-23=creation time in minutes since midnight. (RB.CRT)
Bits 24-35=low order 12 bits of 15-bit creation date in standard format. (RB.CRD)

.RBSIZ — Written length of file. The word is the positive number of words written in the file. For extended arguments, this word is never used for project-programmer numbers. (The four-word block remains compatible so that LH=-number of words in file, RH=0.) This argument is ignored, and a value is always returned.

.RBVER — Octal version number like the contents of location 137 in the job data area.

LH=patch level (A=1, B=2, etc.)
Set by monitor except in the case of privileged programs.

RH=octal version number, never converted to decimal. This argument is accepted, except on a LOOKUP. If a user program wishes to increase the version number by 1 on each UPDATE, it should add 1 to location E + 6 between the LOOKUP and the ENTER.

.RBSPL — Filename to be used to label the output on a device which is being spooled. The filename is taken from the ENTER to the device, or is 0 if an ENTER was not done.

.RBEST — Estimated length of file in positive number of blocks. On an ENTER, FILSER uses this value as the number of blocks to allocate for the file. If the estimated number of blocks is too low, incremental allocation is done.

.RBALC Number of 128-word blocks, N, to be allocated to the file after completion of ENTER or RENAME. This number includes the RIBs of the file. N is equivalent to last relative block of the file.

A 0 means do not change allocation rather than deallocate all the blocks of the file. All of the data blocks can be deallocated by superseding the file and doing no outputs before the CLOSE. This argument can be used to allocate additional space onto the end of the file, deallocate previously allocated but unwritten space, or truncate written data blocks.

The smallest unit of disk space that the monitor can allocate is a cluster of 128-word blocks. Typically small devices use a cluster size of 1 block. If N is not the last block of a cluster, the monitor rounds up, thereby adding a few more blocks than the user requested.

.RBPOS Logical block number, L, of the first block to be allocated for a new group of clusters appended to the file. A logical block number is specified with respect to the entire file structure. Logical block numbers begin with logical block number 0. This feature combined with DSKCHR UUO allows a user program to allocate a file with respect to tracks and cylinders for maximum efficiency when the program runs alone. Because SAT blocks, swapping space, and bad blocks are scattered throughout a file structure, programs using this feature must be prepared to handle such contingencies. It is discouraged for any programs to depend on blocks actually used for allocation to operate without errors.

.RBFT1 Future nonprivileged argument reserved for DEC.

.RBNCA Nonprivileged argument reserved for customer definition.

.RBMTA A 36-bit tape label if file has been put on magnetic tape. If allocated space is 0, then file was deleted from disk when it was copied on magnetic tape. Argument is accepted only from privileged programs; otherwise, it is ignored.

.RBDEV The logical name of the unit on which the file is located. Ignored as an argument, returned as a value.

.RBSTS File status word

LH=status of UFD. Bit 0=1 (RP.LOG) if the user is logged in and is set by LOGIN. LOGOUT clears this bit.

RH=status of file.

Bit 18=1 (RP.DIR) if file is a directory file; needed to protect the system from a user who might try to modify a directory file. The protection error is given if extension UFD is given on an ENTER or RENAME and this bit is not set.

Bit 19=1 (RP.NDL) if file cannot be deleted, renamed, or superseded, even by a privileged program or by a user logged in under [1,2].

Bit 21=1 (RP.NFS) if file should not be dumped by FAILSAFE because certain files are needed before FAILSAFE can run.

Bit 22=1 (RP.ABC) if file always has bad checksum (because the monitor never recomputes the checksum) e.g., SWAP.SYS, SAT.SYS.

Number of 128-word blocks, N, to be allocated to the file after completion of ENTER or RENAME. This number includes the RIBs of the file. N is equivalent to last relative block of the file.

(continued on next page)

.RBSTS (Cont) Bit 26=1 (RP.CMP) if UFD compressing.

Bit 32=1 (RP.BFA) if file is bad because of a FAILSAFE restore.

Bit 33=1 (RP.CRH) if file was closed after a crash.

Bit 35=1 (RP.BDA) if file is bad because of damage assessment.

The following bits appear in both the LH and RH of this location:

Bit 11 (RP.URE) and bit 29 (RP.FRE) = 1 if any file in this UFD (or this file) has had a hard data error while reading. (The IO.DTE bit has been set.) An entry is made in the BAT block so that the bad region is not reused.

Bit 10 (RP.UWE) and bit 28 (RP.FWE) = 1 if any file in this UFD (or this file) has had a hard data error while writing. (The IO.DTE bit has been set.) An entry is made in the BAT block so that the bad region is not reused.

Bit 9 (RP.UCE) and bit 27 (RP.FCE) = 1 if any file in this UFD (or this file) has had a software checksum error or redundancy check error. (The IO.IMP bit has been set.)

NOTE

Device errors (IO.DER) are not flagged in the file status word because they refer to a device and disappear when a device is fixed.

.RBELB Logical block number within the unit on which last date error (IO.DTE) occurred, as opposed to block within file structure. Set by the monitor in the RIB on a CLOSE when the hardware detects either a hard bad parity error or two search errors while reading or writing the file. Device errors, checksum, and redundancy errors are not stored here. This argument is ignored, and a value is returned.

.RBEUN LH=logical unit number within file structure on which last bad region was detected.

RH=number of bad blocks in the last-detected bad region. The bad region may extend beyond the file. This argument is ignored, and a value is returned.

.RBQTF Meaningful for UFD only. Contains first-come-first-served logged-in quota. This quota is the maximum number of data and RIB blocks that can be in this directory in this structure while the user is logged in. The UFD and its RIB are not counted. Argument is ignored unless it is from a privileged program.

.RBQTO Meaningful for UFD only. Contains logged-out quota. This quota is the maximum number of data and RIB blocks that can be left in this directory in this file structure after the user logs off. LOGOUT requires the user to be below this quota to log off. LOGIN stores these quotas in the RIB of the UFD, so that LOGOUT does not have to scan ACCT.SYS at LOGOUT time to find the quota. Argument is ignored unless it is from a privileged program.

.RBQTR	Meaningful for UFD only. (Reserved for the future.) Contains reserved logged-in quota. This quota is the guaranteed number of blocks the user has when he logs in. Argument is ignored unless it is from a privileged program.
.RBUSD	Meaningful for UFD only. Contains number of data and RIB blocks used in this directory in this file structure by the user when he last logged off. LOGIN reads this word so that it does not have to LOOKUP all files in order to set up the number of blocks the user has written. LOGIN sets bit 0 of the file status word (.RBSTS) and LOGOUT clears it in order to indicate whether LOGOUT has stored the quantity. Argument is ignored unless it is from a privileged program.
.RBAUT	Contains project-programmer number of the creator or superseder of the file, as opposed to owner of file. Usually the author and the owner are the same. Only when a file is created in a different directory are these different. This argument is used by Batch for validating queue entries in other directories. Argument is ignored unless it is from a privileged program.
.RBNXT	Reserved for future.
.RBPRD	Reserved for future.
.RBPCA	Privileged argument reserved for customer definition.
.RBUFD	The logical block number (not cluster number) in the file structure of the RIB of the UFD in which the name of this file appears.
.RBFLR	The relative block number of the file to which the first pointer of this RIB points. It is used for multiple RIBs (i.e., 0 for prime RIB).
.RBXRA	The extended RIB address (logical unit number and cluster address of next RIB in a multiple-RIB file).
.RBTIM	The date and time of creation of the file in the universal date-time standard (refer to Paragraph 3.6). That is, the LH contains the date and the RH contains the time as a fraction of a day.

6.2.8.3 Error Recovery for ENTER and RENAME UUOs - Error codes for the LOOKUP, ENTER, and RENAME UUOs are returned in the right half of location E + 1 of the four-word argument block and in the right half of location E + 3 (.RBEXT) in the extended argument block. This means that the error code overwrites the high order 3 bits of the creation date and the entire access date. Since the vast majority of programs recover from these errors either by aborting or by reinitializing the entire argument block, this overwriting of data does not cause any problems. However, a small number of programs may attempt recovery by fixing just the incorrect part of the argument block and then re-trying the UUO. These programs should always restore the right half of location E + 1 before retrying an ENTER or a RENAME UUO. (In order to eliminate problems for programs recovering from errors for files with zero creation dates, which is the most common case, error codes are restricted to a maximum of 15 bits even though the entire right half of E + 1 is used. In addition, the 5.06B and later monitors force access dates to be greater than or equal to the creation date, but never greater than the current date.)

6.2.9 Special Programmed Operator Service

The following are special programmed operator service UUOs.

6.2.9.1 PATH. AC, or CALLI AC, 110[1] - This UUO sets or reads the default directory path, or reads the current directory path on a channel. The call is:

```
        MOVE AC, [XWD n, ADR]
        PATH. AC,
        error return
        normal return
ADR:    arg
        scan switch
        ppn
        SFD1 name
        SFD2 name
          :
ADR+n-1: 0
```

The first word of the argument block contains one of the following:

C (ADR) = SIXBIT device name, or XWD 0, D[2]
Return the current path for the specified device or channel D.

C (ADR) = XWD JOB, -1
Return the default directory path.

C (ADR) = -2
Define the default directory path.

C (ADR) = -3
Define the additional path to be searched when a file is not found in the user's directory path.

C (ADR) = XWD JOB, -4
Return the additional path to be searched when a file is not found in the user's directory path.

[1] This UUO depends on FTSFD which is normally off in the DECsystem-1040.

[2] Note that this function of the PATH. UUO is available even if FTSFD is turned off.

If the left half of ADR is a job number N and the right half of ADR is -1 or -4, the returned values are for either

1) job N if 0<N≤ the highest legal job number, or
2) the current job if N is outside the above range (i.e., N≤0 or N > the highest legal job number).

When defining a path within a UFD (C(ADR) = -2), ADR+1 is the scan switch, ADR+2 is the default project-programmer number, and the remainder of the argument block up to the first zero word defines the default path. The scan switch determines whether or not the monitor scans for the file on a LOOKUP. If the switch is 1, the monitor examines the specified directory only; higher level directories are not searched. If the switch is 2, the following occurs:

1) The monitor searches the UFD or SFD specified by the path (either explicit or default path). If the file is found, the scan is terminated.
2) If the file is not found, the monitor backs up one directory along the path and continues the scan (i.e., it scans the directory in which the current SFD appears). The scan is terminated when the UFD is searched or when the file is found.

Scanning allows directories to be nested since any file not found in the current SFD is obtained automatically from a higher level directory. This is useful when a user has a default directory in use containing files he is currently working on and a higher level directory containing checked-out routines. Since SFDs are continued across file structures but the depth of the nesting of directories is not necessarily the same on each file structure, each scan searches the file structures that are 1) in the job's search list and 2) have SFDs to the depth specified in the path. The file structures are searched in the same order as they appear in the search list.

On an ENTER, the scan switch is ignored; if the file is found in the specified directory, it will be superseded. If the file is not found, it will be created at the end of the path in the specified directory whether or not a file with the same name appears in a higher level directory.

When defining the additional path to be used after the user's directory path is searched (C(ADR) = -3), ADR+1 indicates if SYS (bit 35 = 1) or experimental SYS (bit 34 = 1) is to be scanned, and ADR+2 is the project-programmer number to be used for a user library. These locations are used as follows. If the file is not found in the user's directory path on a LOOKUP DSK:, the directory specified in ADR+2 is searched for the file. This directory must be a UFD and allows users with different directory paths to share a common directory of files. If the file is not found in the library and if bit 35 of ADR+1 is set, the system library (SYS:[1,4]) is searched. In addition on a LOOKUP SYS:, if bit 34 of ADR+1 is set, the directory area [1,5] is searched before the system library area [1,4]. The [1,5] area is called the experimental SYS area (NEW:) and can be used to separate software that is near the end of the development and testing stages from the standard system software on the system library [1,4].

When returning a path, ADR+1 contains the following:

bits 34 and 35	the scan switch
bit 33=1	if experimental SYS (NEW:) is searched
bit 32=1	if SYS is searched
bit 31=1	if there is a user library

(continued on next page)

bit 30=1	if the user-supplied project-programmer number is to be ignored on a LOOKUP or ENTER UUO and the implied project-programmer number of the device is to be used (e.g., [1,4] if SYS; [1,5] if NEW). The implied project-programmer number is returned in ADR +2.
bits 27-29	the type of search list: 0 a non-standard search list (e.g., DSKA) 1 job search list 2 ALL search list 3 SYS search list

and ADR+2 through ADR+n-1 is the path. If the path is less than n-1 words, a zero word is stored at the end. If ADR contains a device name or channel number when the UUO is called, the file structure name or ersatz device name is returned in ADR depending on the name specified (e.g., SYS is returned only if C(ADR) = SYS and the job does not have a device with the logical name SYS). If a LOOKUP or ENTER has been done on the specified device or channel number, the following is returned in the argument block:

ADR:	the SIXBIT name of the file structure or ersatz device
ADR+1:	the scan switch.
ADR+2:	the actual project-programmer number associated with the file.
ADR+3:	the actual path of the file.
:	
ADR+m:	0 the end of the path if m <n-1.

If no LOOKUP or ENTER has been done, the following is returned:

ADR:	SIXBIT DSK or ersatz device name.
ADR+1:	the scan switch.
ADR+2:	the job's default project-programmer number (or the project-programmer number of the ersatz device).
ADR+3:	the default path to the file.
:	
ADR+m:	0 the end of the path if m <n-1

On an error return,

AC is unchanged if the UUO is not implemented. (SFD remains a reserved extension, but all SFD code disappears.) The GETTAB which returns the maximum number of SFDs allowed returns 0 or fails. The default path is the user's project-programmer number.

AC is 0 if the device or channel number does not represent a disk.

AC is -1 if a SFD in the path specification is not found.

(continued on next page)

Examples

1) This example sets the default path to [27,235,SUB] with no scanning in effect.

```
    MOVE 1, [XWD 5, A]
    PATH. 1,
    error
    normal
A:  -2
    1
    27,235
    SUB
    0
```

2) Refer to Figure 6-6. The path plus filename for file A is X.MAC [10,63]. The path plus filename for file B is Y.CBL [14,5]. The path plus filename for file C is Z.ALG [14,5,M].

3) Refer to Figure 6-7. The job's search list is DSKA/N, DSKB, DSKC and the default path is [PPN, A, B, C, D].

   A) LOOKUP DSK: with no matches scans in order: DSKA:D (.SFD), DSKA:C, DSKB:C, DSKA:B, DSKB:B, DSKA:A, DSKB:A, DSKA:PPN (.UFD), DSKB:PPN DSKC:PPN.

   B) LOOKUP DSK: FILE2 finds DSKA: FILE2 [PPN, A, B, C].

(continued on next page)

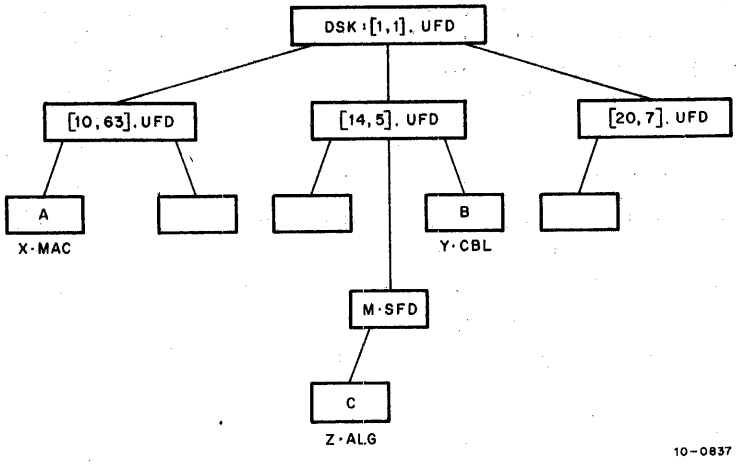


Figure 6-6 Directory Paths on a Single File Structure

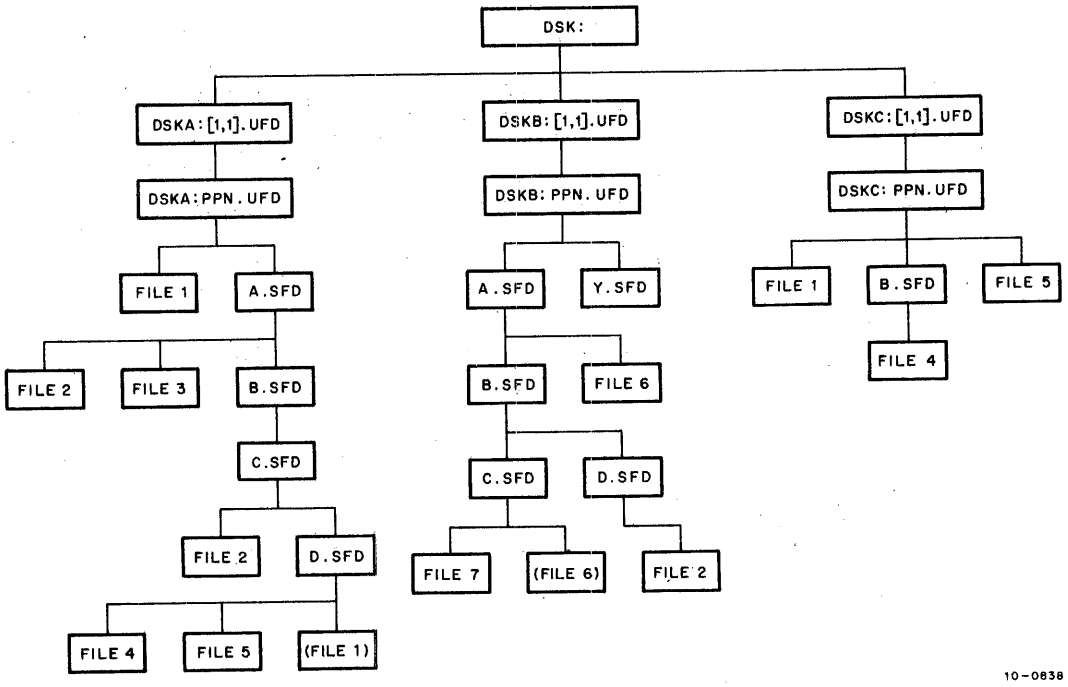


Figure 6-7 Directory Paths on Multiple File Structures

C) LOOKUP DSKB: FILE2, or LOOKUP DSKC: FILE2 fails.

D) ENTER DSK: FILE9 receives an error since no file structure has both the no-create bit off and the directory structure [PPN, A, B, C, D].

E) ENTER DSKA: FILE1 creates the file at the end of the path on DSKA (the file designated by (FILE1) in diagram).

If the default path is [PPN, A, B, C]:

A) ENTER DSK: FILE6 creates DSKB: FILE6 [PPN, A, B, C] (the file designated by (FILE6) in diagram).

B) ENTER DSK: FILE2 supersedes FILE2 in DSKA: [PPN, A, B, C]

C) LOOKUP DSK: FILE4 fails.

D) ENTER DSK: FILE7 supersedes FILE7 in DSKB: [PPN, A, B, C].

4) The user defines the following path.

```
MOVE 1, [XWD 5,A]
PATH. 1,
error
MOVE 1, [XWD 3,B]
PATH. 1,
error
```

```
A: -2          Define the default directory path
    2          Scanning is in effect
10,63          The UFD [10,63]
NAME           The SFD [NAME]
    0          The default path is [10,63,NAME]
B: -3          Define an additional path.
    3          Both experimental SYS and SYS are searched.
10,7           The user library is [10,7]
```

If the user is logged in as [10,10] and does a LOOKUP DSK: FILTST, the following directories are searched in order:

```
[NAME.SFD]
[10,63. UFD]   } job's search list.
[10,7. UFD]
[1,5. UFD]     } system's search list.
[1,4. UFD]
```

If the user is logged in as [10,10] and does a LOOKUP DSK: PRJFIL [10,155], the following directories are searched:

```
[10,155. UFD]  } job's search list
[10,7. UFD]
[1,5. UFD]     } system's search list
[1,4. UFD]
```

6.2.9.2 USETI and USETO UUOs - The function of these UUOs is to notify the disk service routines that a particular relative block (instead of the next block in sequence) is to be used on the following INPUT or OUTPUT on the specified channel. USETI and USETO do not perform I/O; they simply change the current position of the file. Note that each INPUT or OUTPUT also logically advances the file; therefore, to reread or rewrite the same block a USETI (or USETO) must be given before each INPUT (or OUTPUT).

Since the monitor reads (writes) as many buffers as it can on INPUT (OUTPUT), it is difficult to determine which buffer the monitor is processing when the USETI (USETO) is given. Thus, the INPUT (OUTPUT) following the USETI (USETO) may not read (write) the buffer containing the block specified with the USETI (USETO). However, a single buffer ring reads (writes) the desired block since the device must stop after each INPUT (OUTPUT). Alternatively, if bit 30 of the status word (IO.SYN) is set via an INIT, OPEN, or SETSTS UUO, the device stops after each bufferful of data on an INPUT (OUTPUT) so that the USETI (USETO) will apply to the buffer supplied on the next INPUT (OUTPUT). The calls are:

USETI D,N and USETO D,N

where D is the channel number, and N designates a block relative to the beginning of the file. N can be in the following ranges:

N	Block Represented
$1\text{-}777777_8$	Blocks of the file
0	Prime (1st) RIB
$-2,\ldots,-10_8$	Extended (2nd to the 8th) RIB[1]
-1	Last block accessed (USETO) or end of file (USETI).

Note that the 18-bit effective address used for N is interpreted as both an unsigned positive integer and a signed (2's complement) integer. This is required since, with extended RIBs, there can be more than 377777_8 (largest positive signed integer) blocks in a file. The exact interpretation of N depends upon the context of the USETI/USETO (i.e., reading, writing, updating).

When reading or writing a file, USETI precedes an INPUT and USETO precedes an OUTPUT (i.e., USETI is illegal for a non-privileged program unless a LOOKUP has been done, and USETO is illegal for a non-privileged program unless an ENTER has been done). However, there are special cases when updating a file (both a LOOKUP and an ENTER have been done) when USETI may be followed by an OUTPUT and USETO may be followed by an INPUT. The action performed on a USETI or USETO depends on the value of N.

When N is a block number less than or equal to the current size of the file in blocks (i.e., N is a block that has been previously written), USETI or USETO points to block N in order to read or write that block on the next INPUT or OUTPUT.

---

[1] The number of extended RIBs allowed on the system can be changed with MONGEN and can be obtained from a GETTAB table (.GTLVD, item 23). Extended RIBs depend on FTMRIB which is normally off in the DECsystem-1040.

When N is a block number greater than the current size of the file in blocks, USETI followed by an INPUT receives the end-of-file return (e.g., if the file is 5 blocks long, USETI with n=7 receives the end of file return). On a USETO followed by an OUTPUT, the monitor allocates the intervening blocks, writes zeroes in the first new block up to block N-1, and then writes block N. For example, if the file is 2 blocks long, USETO with n=4 writes zeroes in block 3 and the data on the OUTPUT in block 4. If the number of blocks requested cause the disk to be filled or the user's quota to be exceeded, as many blocks as allowed will be allocated and the IO.BKT bit will be set in the status word. In addition, in update mode, USETI followed by an OUTPUT appends the data to the end of the file (i.e., makes the file larger). USETO followed by an INPUT allocates and zeroes the first new block up to block N-1 and then receives the end-of-file return.

When N=0 on reading, writing, and updating, USETI and INPUT read the prime RIB, and USETO and OUTPUT receive the IO.BKT error. In addition, in update mode, USETO and INPUT read the prime RIB, and USETI and OUTPUT receive the IO.BKT error.

When N=-2 to -10_8, a USETI and INPUT read the indicated extended RIB (-2 is the 2nd RIB,..., -10_8 is the 8th RIB). USETO followed by an OUTPUT attempts to allocate a large number of blocks (since N is interpreted as an unsigned integer) and therefore is not recommended because the user's disk quota will probably be exceeded.

When N=-1, USETO and OUTPUT rewrite the last block in which I/O was performed. USETI and INPUT receive the end of file return. In addition, in update mode, USETI followed by OUTPUT appends the data to the end of the file, and USETO and INPUT read the last block in which I/O was performed.

The user can append data to the last block of an append-only file by specifying a USETO followed by an OUTPUT to the last block.[1] The monitor then reads the block (of N words) into a monitor buffer, copies words N+1 through 200 from the user's buffer into the monitor buffer, and rewrites the block. The current length of the block can be obtained from the LOOKUP/ENTER block. It is not necessary to read the last block of the file before appending to it because the data already existing in the block is not changed.

When appending data to the last block of a file, the IO.BKT bit is set and no output is done if

1) Any block before the last block is written.
2) The last block already contains 200 words.
3) Fewer blocks are written than the current size of the block.

If the last block is written with a buffer-mode OUTPUT, the size of the last block becomes 200 words, and therefore, cannot be appended to.

Append-only files can be read only if FTAIR is on. Note that BASIC stores data at the beginning of files that it must read and therefore, to run BASIC, FTAIR must be turned on.

---

[1] This feature depends on FTAPLB, which is normally off in the DECsystem-1040. Therefore a new block must be written in order to append to a file.

If no previous LOOKUP or ENTER has been done, these UUOs are considered to be super-USETI and super-USETO which are available only to privileged programs. If the program is non-privileged, super-USETI and super-USETO cause the IO.BKT bit to be set in the status word. These privileged UUOs are documented in UUOPRV.RNO in the DECsystem-10 Software Notebooks.

6.2.9.3 SEEK UUO[1] - This UUO, when used in conjunction with USETI and USETO, allows user programs control over the time at which positioning operations occur. Following a regular USETI or USETO, positioning is to the cylinder containing the requested relative block within a file. Following a super-USETI or super-USETO, positioning is to the cylinder containing the specified disk block.

The call is:

```
SEEK AC,          ; or CALLI D, 56
return
```

D specifies a software channel number. The SEEK UUOs are honored by the monitor only if the unit for which they are issued is idle. If the unit is in any other state, the SEEK UUO is a no-operation.

SEEK UUOs issued for public file structures are treated in the same way as private file structures. This allows users to debug programs using a public disk pack and later run the same programs using a private disk pack.

The following is proper UUO sequence for issuing a SEEK.

For output

a. USETO to select a block (relative or actual)
b. SEEK to request positioning
c. computations
d. OUTPUT to request actual output

For input

a. USETI to select a block (relative or actual)
b. SEEK to request positioning
c. computations
d. INPUT to request actual input.

6.2.9.4 RESET UUO - This UUO causes files that are in the process of being written, but have not been CLOSEd or RELEASed, to be deleted; the space is reclaimed. If a previous version of the file with the same name and extension existed, it remains unchanged on the disk (and in the UFD). If the programmer wishes to retain the newly created file and to delete the older version, he must CLOSE or RELEASe the file before doing a RESET UUO.

---

[1]This UUO depends on FTDSEK which is normally off in the DECsystem-1040.

6.2.9.5 DEVSTS UUO - After each interrupt, FILSER stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS UUO is used to return the contents of the DEVSTS word to the user (refer to Paragraph 4.10.1).

6.2.9.6 CHKACC UUO[1] - This UUO allows programs to check the user's access to a particular file. The call is:

```
MOVE AC, [EXP LOC]
CHKACC AC,                ;or CALLI AC, 100
error return
normal return
```

The LH of LOC contains the code for the type of access to be checked and the RH of LOC contains a 9-bit protection field. If the access code contained in the LH of LOC is greater or equal to 7, then the RH of LOC is interpreted as UFD privilege bits. LOC+1 contains the project-programmer number of the directory, and LOC+2 contains the project-programmer number of the user.

The type of access to be checked is represented by one of the following codes:

0	.ACCPR	Change protection.
1	.ACREN	Rename.
2	.ACWRI	Write.
3	.ACUPD	Update.
4.	.ACAPP	Append.
5	.ACRED	Read.
6	.ACEXO	Execute only.
7	.ACCRE	Create in UFD.
10	.ACSRC	Read directory as a file.

The error return is given if the UUO is not implemented. On a normal return, AC contains 0 if access is allowed or -1 if access if not allowed.

6.2.9.7 STRUUO AC, or CALLI AC, 50 - This UUO manipulates file structures and is intended primarily for monitor support programs.

The call is:

```
MOVE AC, [XWD N, LOC]
STRUUO AC,                ;or CALLI AC, 50
error return              ;AC contains an error code
normal return             ;AC contains status information
```

N is the number of words in the argument list starting at location LOC. For the functions with a fixed length argument list, N may be 0.

The first word of the argument list specifies the function to be performed. Function 0 (.FSSRC) is the only unprivileged function; the remaining functions are available only to jobs logged-in under [1,2]

---

[1] This UUO depends on FT5UUO which is normally off in the DECsystem-1040.

or to programs running with the JACCT bit set. Refer to the Specifications section of the DECsystem-10 Software Notebooks for a complete description of the privileged functions and their appropriate error codes.

The present functions are as follows:

Function	Name	
0	.FSSRC	Define a new search list for this job. This is the only unprivileged function.
1	.FSDSL	Define a new search list for any job or for the system. Privileged function.
2	.FSDEF	Define a new file structure. Privileged function.
3	.FSRDF	Redefine an existing file structure. Privileged function.
4	.FSLOK	Prevent any further new INITs, ENTERs, or LOOKUPs. Privileged function.
5	.FSREM	Remove file structure from system. Privileged function.
6	.FSULK	Test and set UFD interlock. Privileged function.
7	.FSUCL	Clear UFD interlock. Privileged function.
10	.FSETS	Simulate disk hardware errors. Privileged function.

6.2.9.7.1 Function 0 .FSSRC - This function allows a new file structure search list to be specified for the job issuing the UUO. The call is:

```
        MOVE AC, [XWD N, LOC]
        STRUUO AC,
        error return
        normal return

LOC:    0                              ;.FSSRC
LOC+1:  first file structure name
LOC+2:  0
LOC+3:  status bits
LOC+4:  second file structure name
LOC+5:  0
LOC+6:  status bits
          .
          .
          .
```

The argument list consists of word triplets, which specify the new search list order to replace the current search list. The current search list may be determined with the JOBSTR UUO. The first word contains a left-justified file structure name in SIXBIT. The second word is not used at present. The third word contains the following status bits:

Bit 0 = 1 if software write-protection is requested for this file structure.

Bit 1 = 1 if files are not to be created on this file structure unless the specific file structure is specified in an ASSIGN command or in an INIT or OPEN UUO.

The user may use the MOUNT command to add a new file structure name to his search list. The MOUNT program

a. Requests the file structure to be mounted (if it is not already mounted).

b. Creates a UFD for the user if he has a logged-in quota in file SYS: QUOTA.SYS on that file structure.

A user cannot create files on a file structure unless he or the project-programmer number specified has a UFD on that file structure. However, by using the .FSSRC function, the user may add a file structure name to his search list if the file structure is mounted and either the user has a UFD for that file structure or he does not want to write on that file structure. If the user attempts to delete a file structure name from his search list by the .FSSRC function, the monitor moves the file structure name from the active search list to the passive search list. The DISMOUNT command must be used to remove the file structure from the active or passive search list. The DISMOUNT command causes the mount count to be decremented, signifying that the user is finished with the file structure, and checks that the user has not exceeded his logged-out quota on the file structure.

Table 6-7
.FSSRC Error Codes

Symbol	Code	Explanation
FSILF%	0	Illegal function code.
FSSNF%	1	One or more file structures not found.
FSSSA%	2	One or more file structures single access only.
FSTME%	4	Too many entries in search list.
FSRSL%	17	File structure is repeated in a search list definition.

6.2.9.8 JOBSTR AC, or CALLI AC, 47[1] - This UUO returns the next file structure name in the job's search list along with other information about the file structure. Programs like DIRECT use this UUO to list a user's directory correctly and specify in which file structures the files occur, as well as the order in which they are scanned.

The call is:

```
MOVE AC, [XWD N,LOC]
JOBSTR AC,                    ;or CALLI AC, 47
error return
normal return
```

LOC is the address of the N-word argument block. When the UUO is called, the first word should be one of the following:

a. -1 to return the first file structure name in the search list.

b. a file structure name to return the next file structure following the specified name.

c. 0 to return the file structure name immediately following the FENCE. Refer to Paragraph 6.2.7.

[1] In the DECsystem-1040, FTSTR is normally off so that there is only one file structure on the system. However, this UUO is implemented and returns the file structure name or -1.

On return, the first word contains:

a. the first file structure name in the search list if -1 was specified.

b. the next file structure name appearing after the specified name or after the FENCE (if 0 was specified).

c. 0 if the item after the specified name is the FENCE.

d. -1 if there are no more file structure names in the search list, or the search list is empty.

The second word contains 0 (reserved for a future argument), and the third word contains status bits. Current status bits are:

Bit 0 = 1 if software write protection is in effect for this job.

Bit 1 = 1 if files are not to be created on this file structure, when a multiple file structure name is specified in an INIT or OPEN UUO. Files can be created if a specific file structure or physical unit is specified.

The following is an example of reading a job's search list.

```
        SETOM LOC               ;place -1 in LOC to get 1st
                                ;name in search list.
LOOP:MOVEI AC, LOC              ;set up AC.
        JOBSTR AC,              ;do the UUO.
        JRST ERROR              ;error return.
        MOVE AC, LOC            ;get file structure name returned.
        JUMPE AC, FENCE         ;jump if it is the FENCE.
        AOJE AC, END            ;jump if end of search list (-1).
        .                       ;LOC has next file structure name.
        .
        .
        JRST LOOP               ;repeat with next file structure name.
LOC: -1                         ;file structure name.
        0                       ;reserved for future use.
        0                       ;status bits.
```

6.2.9.9 GOBSTR AC, or CALLI AC, 66 - This UUO returns successive file structure names in the search list of either an arbitrary job or the system. The GOBSTR UUO is a generalization of the JOBSTR UUO (see Paragraph 6.2.9.8). It is a privileged UUO unless information being requested is either about the system search list or the jobs logged-in under the same project-programmer number as the calling job's number. For example, the KJOB program needs information about the search lists of jobs logged in under the same project-programmer number as the job logging out. The privilege bits required are either JP.SPA (bit 16) or JP.SPM (bit 17) of the privilege word (.GTPRV).

The call is:

```
        MOVE AC, [XWD N,LOC]
        GOBSTR AC,              ;or CALLI AC, 66
        error return            ;AC contains an error code
        normal return
```

When the UUO is called, AC specifies the length (N) and address (LOC) of an argument list. N may be 0, 3, 4, or 5 where N = 0 has the same effect as N = 3. Only the arguments included by N(LOC, LOC+1,...,LOC+N-1) are used or returned. The argument list is as follows:

```
LOC:  job number                    ;job whose search
                                    ;list is desired.
      XWD proj, prog                ;project-programmer
                                    ;number of above job.
      SIXBIT /file structure name/  ;or -1 or 0.
      0                             ;currently unused.
      Status                        ;status bits are the same
                                    ;as in JOBSTR UUO.
```

If the job number = -1, the number of the job issuing the UUO is used. If the job number = 0, the given project-programmer number is ignored and the system search list is used. When the given project-programmer number is -1, the project-programmer number of the job issuing the UUO is used.

On an error return, AC contains one of the following error codes:

Code	Meaning
3	If LOC+2 is not 0, -1, or a file structure name in jobs search list.
6	If job number (LOC) and project-programmer number (LOC +1) do not correspond.
10	If job issuing the UUO is not privileged.
12	If the length specified for the argument list is not valid.

6.2.9.10 SYSSTR AC, or CALLI AC, 46 - This UUO provides a simple mechanism to obtain all the file structure names in the system. The proper technique to access all files in all UFDs is to access the MFD on each file structure separately. Monitor support programs use this UUO to access all the files in the system.

The call is:

```
MOVEI AC, 0 or the last value returned by previous SYSSTR
SYSSTR AC,                          ;or CALLI AC, 46
error return
normal return
```

An error return is given if either

a. The UUO is not implemented

b. The argument is not a file structure name

On a normal return, the next public or private file structure name in the system is returned in AC. A return of 0 in AC on a normal return means that the list of file structure names has been exhausted. If

0 is specified as an argument, the first file structure name is returned in AC. The argument cannot be a physical disk unit name or a logical name.

6.2.9.11 SYSPHY AC, or CALLI AC, 51[1] - This UUO returns all physical disk units in the system. The SYSPHY UUO is similar to the SYSSTR UUO (see Paragraph 6.2.9.10).

The call is:

```
MOVEI AC, 0 or the last unit name returned by previous SYSPHY
SYSPHY AC,                  ; or CALLI AC, 51
error return                ;not implemented or not a physical disk
normal return               ;unit name
```

On the first call AC should be 0 to request the return of the first physical unit name. On subsequent calls, AC should contain the previously returned unit name.

An error return is given if AC does not contain a physical disk unit name as zero. On a normal return, the next physical unit name in the system is returned in AC. A return of 0 in AC indicates that the list of physical units has been exhausted.

6.2.9.12 DEVPPN AC, or CALLI AC, 55[1] - This UUO allows a user program to obtain the project-programmer number associated with a disk device. The device argument given can be a logical device name, a physical device name, or one of the special devices called ersatz devices. (Refer to DEC-system-10 Operating System Commands for the list of system devices.)

When the UUO is called, AC must contain either the device name as a left-justified SIXBIT quantity, or the channel number of the device as a right-justified quantity.

The call is:

```
MOVE AC, [SIXBIT /DEV/]     ;or MOVEI AC, channel number
DEVPPN AC,                  ;or CALLI AC, 55
error return
normal return
```

The error return is taken if:

[1]This UUO depends on FT5UUO which is normally off in the DECsystem-1040.

a. The UUO is not implemented; therefore, the contents of AC remain the same on return. In this case, obtain the appropriate project-programmer number as follows:

   1. For the user's area, use the GETPPN UUO (refer to Paragraph 3.6.2.3).

   2. For the special ersatz devices, use the default project-programmer numbers appearing in the following list.

Device	Project-programmer Number
ALL:	User's project-programmer number
BAS:	[5,1]
COB:	[5,2]
DSK:	User's project-programmer number
HLP:	[2,5]
LIB:	Set by each user
MXI:	[5,3]
NEW:	[1,5]
OLD:	[1,3]
SYS:	[1,4]

b. The device does not exist or the channel is not INITed; therefore, zero is returned in AC.

c. The device is not a disk; the user's project-programmer number is returned in AC.

If a legal device is specified, the normal return is given and the project-programmer number associated with the device is returned in AC. However, if the user has device NEW: enabled in his search list and device SYS: is given as the argument to the DEVPPN UUO, the project-programmer number returned is [1,5].

The following is an example for reading a UFD if either device SYS or the user's area is specified.

```
        MOVEI   A,16            ;GET MFD PROJECT-PROGRAMMER NUMBER
        GETTAB  A,              ;NO CHANGE IF NO GETTAB
          MOVE  A,[1,,1]        ;  IN CASE OF LEVEL C
        MOVEM   A,MFDPPN        ;STORE MFD DIRECTORY NUMBER

        MOVE    A,DEVNAM        ;GET DEVICE NAME TYPED BY USER
        MOVEM   A,MODE+1        ;STORE FOR OPEN
        DEVPPN  A,              ;GET PROJECT PROGRAMMER NUMBER
          JRST  GETPPX          ;  NOT IMPLEMENTED OR NO SUCH DEVICE

;BACK HERE WITH IMPLIED PPN IN A

GOTPPN: MOVEM   A,PPN           ;STORE PPN IMPLIED BY DEVICE NAME

        OPEN    I,MODE          ;TRY TO OPEN DEVICE
          JRST  ERROR           ;NOT AVAILABLE
        LOOKUP  I,PPN           ;TRY TO LOOKUP UFD
          JRST  ERROR           ;NOT THERE
        IN      I,              ;READ FIRST BLOCK
          JRST  USEIT           ;GO DO USEFUL WORK
        JRST    ERROR           ;ERROR OR END OF FILE

;HERE IF DEVPPN FAILS

GETPPX: CAMN    A,[SIXBIT /SYS/]  ;SEE IF DEVICE NAME SYS:
        JRST    GETPPS          ;YES--GO HANDLE SYS:
        GETPPN  A,              ;NO--GET OWN PPN
          JFCL                  ;(IN CASE OF JACCT)
        JRST    GOTPPN          ;OK--PROCEED ABOVE

GETPPS: MOVE    A,[1,,16]       ;FIND SYS: PPN
        GETTAB  A,              ;FROM MONITOR TABLES
          MOVE  A,[1,,1]        ;(IN CASE OF LEV. C)
        JRST    GOTPPN          ;OK--PROCEED ABOVE

MODE:   14              ;BINARY READ
        0               ;DEVICE NAME
        0,,INBUFH       ;BUFFER HEADER

PPN:    0               ;DIRECTORY NAME
        SIXBIT /UFD/    ;EXTENSION
        0
MFDPPN: 1,,1            ;LOOKUP UFD IN MFD
```

6.2.9.13 DSKCHR AC, or CALLI AC, 45 - The disk characteristics UUO provides necessary information for allocating storage efficiently on different types of disks. Most programs are able to use the generic device name DSK rather than special disk names; however, this UUO is needed by special monitor support programs.

This UUO accepts, as arguments, names of file structures (e.g., DSKA), types of controllers (e.g., DP), controllers (e.g.,DPA), logical units (e.g., DSKA3), physical disk units (e.g., DPA3), or logical device names (e.g., ALPHA). If the argument in LOC specifies more than one unit, the values returned in AC are for the first unit of the specified set. If the argument specifies more than one file structure (i.e., DSK or logical device name for disk), the first unit of the first file structure is returned.

The call is:

```
MOVE AC, [XWD+N,LOC]      ;N is the number of locations
                          ;of arguments and values starting
                          ;at location LOC
DSKCHR AC,                ;or CALLI AC, 45
error return              ;not a disk
normal return
```

On a normal return, AC contains status information in the left half and configuration information in the right half. The left half bits have been chosen so that the normal state is 0.

Name	Bit	Explanation
DC.RHB	Bit 0 = 1	The monitor must reread the home block before the next operation to ensure that the pack ID is correct. The monitor sets this bit when a disk pack goes off-line.
DC.OFL	Bit 1 = 1	The unit is off-line.
DC.HWP	Bit 2 = 1	The unit is hardware write-protected.
DC.SWP	Bit 3 = 1	The unit belongs to a file structure that is write-protected by software for this job.
DC.SAF	Bit 4 = 1	The unit belongs to a single-access file structure.
DC.ZMT	Bit 5 = 1	The unit belongs to a file structure with a mount count that has gone to zero (i.e., no one is using the file structure). Available in 5.02 monitors and later.
	Bit 6 = 1	Reserved for the future.
DC.STS	Bits 7 and 8	Unit status
.DCSTD	= 11	The unit is down.
.DCSTN	= 10	No pack is mounted.
	= 01	Reserved for the future.
.DCSTP	= 00	A pack is mounted.
DC.MSB	Bit 9 = 1	The unit has more than one SAT block.
DC.NNA	Bit 10 = 1	The unit belongs to a file structure for which the operator has requested no new INITs, LOOKUPs, or ENTERs; set by privileged STRUUO function.
DC.AWL	Bit 11 = 1	The file structure is write-locked for all jobs.
	Bits 12 - 14	Reserved for future expansion.

Name	Bit	Explanation
DC.TYP	Bits 15 - 17	The code identifies which type of argument was passed to the monitor in location LOC.
DC.DCN	Bits 18 - 20	Data channel number that software believes hardware is connected to; first data channel is 0.
DC.CNT .DCCFH .DCCDP	Bits 21 - 26 =1 =2	Controller type: FH (Burroughs disk, Bryant drum) controller RC 10 DP (Memorex disk packs) controller RP10
DC.CNN	Bits 27 - 29	Controller number; first controller of each type starts at 0 (e.g., DPA = 0, DPB = 1)
DC.UNT	Bits 30 - 32	Unit type; a controller-dependent field used to distinguish various options of a unit on its controller. If bits 21 - 26 and bits 30 - 32 then type is 1 0 RD10 Burroughs disk (.DCUFD) 1 1 RM10B Bryant drum (.DCUFM) 2 1 RP02 disk pack (.DCUD2) 2 2 RP03 disk pack (.DCUD3)
DC.UNN	Bits 33 - 35	Physical unit number within controller; first unit is 0

The user program supplies in location LOC a left-justified, SIXBIT disk name which may be one of the following:

.DCTDS	0	generic disk name
.DCTAB	1	subset of file structures because of file structure abbreviation
.DCTFS	2	file structure name
.DCTUF	3	unit within a file structure
.DCTCN	4	controller type
.DCTCC	5	controller
.DCTPU	6	physical disk unit name

or a logical name for one of the above assigned by the ASSIGN or MOUNT monitor command.

On a normal return, the monitor returns values in the following locations:

LOC+1 (.DCUFT)	The number of blocks left of the logged-in job quota before the UFD of the job is exhausted on the unit specified in LOC. If negative, the UFD is overdrawn. If the negative number is 400000 000000, the UFD has not been accessed since LOGIN; therefore, the monitor does not know the quota.
LOC+2 (.DCFCT)	The number of blocks on a first-come first-served basis left for all users in the file structure.
LOC+3 (.DCUNT)	The number of blocks left for all users on the specified unit.
LOC+4 (.DCSNM)	The file structure name to which this unit belongs.
LOC+5 (.DCUCH)	Characteristic sizes a. Bits 0-8 are the number of blocks/cluster (DC.UCC) b. Bits 9-17 are the number of blocks/track (DC.UCT) c. Bits 18-35 are the number of blocks/cylinder (DC.UCY) (see Appendix F).

LOC+6 (.DCUSZ)	The number of 128-word blocks on the specified unit.
LOC+7 (.DCSMT)	The mount count for the file structure. The mount count is the number of jobs that have done a MOUNT command for this file structure without executing a DISMOUNT command; it is a use count.
LOC+10 (.DCWPS)	The number of words containing data bits per SAT block on this unit.
LOC+11 (.DCSPU)	Number of SAT blocks per unit.
LOC+12 (.DCK4S)	Number of K allocated for swapping.
LOC+13 (.DCSAJ)	Zero if none or more than one job has this file structure mounted. XWD -1,,n if only job n has file structure mounted but it is not single access. XWD 0,,n if job has file structure mounted and it is single access.
LOC+14 (.DCULN)	The unit's logical name (e.g., DSKB0).
LOC+15 (.DCUPN)	The unit's physical name (e.g., DPA0).
LOC+16 (.DCUID)	The unit's ID (e.g., 2RP003).
LOC+17 (.DCUFS)	The first logical block used for swapping on this unit.

6.2.9.14 DISK. AC, or CALLI AC, 121 - The DISK. UUO is a general-purpose call designed for setting and examining parameters of the disk and file systems. Its present function allows the user to assign a priority for disk operations (data transfers and head positionings) either for a user I/O channel or for his job (all I/O channels). Therefore, when a disk operation is initiated, the request with the highest priority is selected. If there is more than one request with the same priority, the one most satisfying disk optimization is chosen (refer to Chapter 7).

The range of permissible disk priorities is -3 to +3 with 0 being the normal timesharing priority. Thus, a job can request a lower than normal priority (e.g., when the job is a background job). The maximum priority (greater than 0) that the user is allowed to assign is set by bits 1 and 2 (JP.PRI) of the privilege word .GTPRV.

The call is:

```
MOVE AC, [XWD function, ADR]
DISK. AC,                          ;or CALLI AC, 121
error return
normal return
```

The present function is

Function	Name	Description
0	.DUPRI	Set the disk priority

ADR contains, in the right half, the desired priority (-3 to +3) and in the left half, one of the following:

LH (ADR) = n	Sets the priority for channel n (n is from 0 to 17).
LH (ADR) = -1	Sets the priority for all channels OPENed by the job.
LH (ADR) = -2	Sets the priority for the entire job.

When a priority is set for a channel, it overrides any priority set for the job and remains in effect until the channel is RELEASed. When set for the job, the priority remains in effect until RESET by another DISK. UUO or the SET DSKPRI command (refer to DECsystem-10 Operating System Commands).

6.2.9.15 Simultaneous Supersede and Update - Files that may be simultaneously superseded or updated by several different users should be treated with care. The problem arises when one user has a copy of information to be superseded by another user. For example, file F contains a count of the number of occurrences of a certain event. The count is 10 at a given time. When two users observe separate instances of the event, each tries to increment the count.

Supersede - Incorrectly

```
Job 1                    Job 2
LOOKUP A,F
                         LOOKUP C, F
READ COUNT (=10)         READ COUNT (=10)
ADD 1 (=11)              ADD 1 (=11)
                              :
ENTER B, F               ENTER D, F          (Fail)
WRITE OUT (=11)               .
CLOSE B,                 ENTER D, F          (Succeed)
                         WRITE OUT (=11)
                         CLOSE D,
```

In this example, job 2 ignored job 1's increment.

Supersede - Correctly

```
Job 1                    Job 2
ENTER B, F
LOOKUP A, F
     .                   ENTER D, F          (Fail)
INPUT A, (=10)
ADD1 (=11)                    .
OUTPUT B, (=11)               .
CLOSE B,                      .
                         ENTER D, F          (Succeed)
                         LOOKUP C, F
                         INPUT C, (=11)
                         ADD1 (=12)
                         OUTPUT D, (=12)
                         CLOSE D,
```

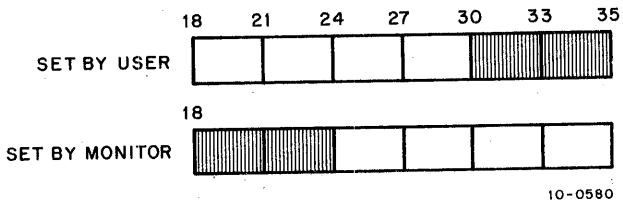
In this example, both jobs performed the ENTER FIRST; therefore, incorrect copies were not made and the increment of each job was recorded properly.

The similar problem with a update can be avoided by never using the information returned by the LOOKUP:

Job 1	Job 2	
LOOKUP A, F INPUT A,	LOOKUP B, F INPUT B,	
ENTER A, F OUTPUT CLOSE	ENTER B, F Here any information from the LOOKUP and INPUT must be discarded.	(Fail)

6.2.10 File Status (refer to Appendix D)

The file status of the disk is shown below.



Bit 18 - IO.IMP

a. INPUT UUO attempted on a read-protected file

b. INPUT UUO when no LOOKUP was done (or super-USETI/USETO previously attempted by nonprivileged user)

c. OUTPUT UUO when no ENTER was done (or super-USETI/USETO previously attempted by nonprivileged user)

d. Software-detected checksum error

e. Software-detected redundancy error in SAT block or RIB, or

f. Buffered mode I/O attempted after super-USETI/USETO.

g. OUTPUT UUO attempted on a write-locked unit.

Bit 19 - IO.DER — Search error, power supply failure

Bit 20 - IO.DTE — Disk or data channel parity error. Checksum failure on INPUT.

Bit 21 - IO.BKT

a. Quota is exhausted (past overdrawn)

b. File structure is exhausted

c. RIB is full

d. Super-USETI/USETO block is too large for the file structure

(continued on next page)

Bit 21 - IO.BKT (cont)

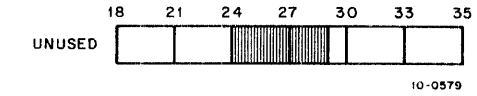
e. More than 777777 blocks were read with one super-USETI/USETO

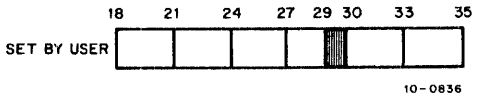
f. Block number specified is too low for writing in a file that has an append protection code (4). The block number must be greater than the current highest block number of the file. Not set on a USETI or USETO.

g. A super-USETI or USETO was issued by a non-privileged program.

Bit 22 - IO.EOF EOF encountered on INPUT. No special character appears in the buffer.

Bit 23 - IO.ACT Device is active





Bit 29 - IO.WHD Write disk pack headers

6.2.11 Disk Packs

A disk pack system combines disk and the DECtape features. Some packs (similar to individual DECtapes) are designed to be private, assignable, and removable. The other packs make up part or all of the public disk storage area where system programs and user files are stored. These disk packs belong to file structures in the storage pool and cannot be assigned to any single user. The system library and shared on-line storage is maintained and swapping storage is assigned within the public disk pack area.

The only distinction between public and private packs is that private packs are intended to be removed from the system during regular operation. Public packs usually stay on-line all the time. However, the file structure format for public and private disk packs is identical.

User programs can exercise much greater control over private packs. For example, a program may attempt to position the arms of disk packs in anticipation of future I/O (refer to Paragraph 6.2.9.3). This capability is useful to a program that is aware of the contents of a disk and is able to use this information to optimize positioning. The program may also specify the position of files on the disk by using the allocate arguments of the extended LOOKUP, ENTER, and RENAME UUOs.

Private packs may be accessed by more than one job (multi-access) or restricted to only one job (single access). To access a private file structure, the user must type the MOUNT monitor command. If the private file structure is already mounted, on-line, and multi-access, the user receives an immediate

response and may start using the private pack. When the user is finished using the private file structure, he should type the DISMOUNT monitor command. If no other job is using the file structure, a message is typed to the operator informing him that the drives belonging to the file structure are free.

6.2.11.1 Removable File Structures - All file structures are designed as if they could be removed from the system; therefore, disk packs are handled the same as other types of disks.

6.2.11.2 Identification - Disk packs have identifying information written on the home block, a block on every unit identifying the file structure to which the unit belongs and its position within the file structure. Part of this information is the pack ID, a one- to six-character SIXBIT name uniquely identifying the disk pack. The MOUNT and OMOUNT programs check that the operator has mounted the proper packs by comparing the pack ID in the home block with the information stored in the system administration file STRLST.SYS.

6.2.11.3 IBM Disk Pack Compatibility - The data format of IBM disk packs has variable-length sectors and no sector headers. DEC format has fixed-length sectors (128 words) and specially written sector headers. Latency optimization is employed to improve system throughput (refer to Paragraph 7.3). DEC's significantly simpler hardware controller is used without reducing user capabilities.

To transfer data from a IBM pack system to a DEC pack system, a simple program in a higher-level language should be written for both machines. The program then reads the IBM disk pack on the IBM computer and writes the files onto magnetic tape. The magnetic tape is then transferred to a DEC computer and read by another program, which writes the files onto the DEC RP01 or RP02 packs.

6.3 SPOOLING OF UNIT RECORD I/O ON DISK

Devices capable of spooling (card reader, line printer, card punch, paper-tape punch, and plotter) have an associated bit in the job's .GTSPL word. If this bit is on when the device is ASSIGNed or INITed, the device is said to be in spool mode. While in this mode, all I/O for this device is intercepted and written onto the disk rather than onto the device. System spooling programs later do the actual I/O transfer to the device.

Spooling allows more efficient use of the device because users cannot tie it up indefinitely. In addition, since the spooling devices are generally slow and the jobs that are to be spooled are usually large, the jobs do not spend unnecessary time in core.

6.3.1 Input Spooling

If a LOOKUP is given after the INIT of the card reader, it is ignored and an automatic LOOKUP is done, using the filename given in the last SET CDR command and the filename extension of .CDR. After every automatic LOOKUP, the name in the input-name counter .GTSPL is incremented by 1 so that the next automatic LOOKUP will use the correct filename.

6.3.2 Output Spooling

If an ENTER is done, the filename specified is stored in the RIB in location .RBSPL so that the output spooler can label the output. Therefore, programs should specify a filename if possible.

If an ENTER is not done, an automatic ENTER is given, using a filename in the general form

xxxyyy.zzz

where xxx is a three-character name manufactured by the monitor to make the 9-character name unique.

yyy is (1) an appropriate station number Snn if a generic device name is INITed or (2) a unit number if a specific unit is INITed.

zzz is the generic name of the device-type (LPT, CDP, PTP, or PLT).

Output spooling should not concern the user because all requests are queued when the user logs off the system. The files are moved to the output queues before the logged-out quota is computed.

CHAPTER 7
MONITOR ALGORITHMS

7.1 JOB SCHEDULING

The number of jobs that may be run simultaneously must be specified in creating the DECsystem-10 Monitor. Up to 127 jobs may be specified. Each user accessing the system is assigned a job number.

In a multiprogramming system all jobs reside in core, and the scheduler decides what jobs should run. In a swapping system, jobs exist on an external storage device (usually disk or drum) as well as in core. The scheduler decides not only what job is to run but also when a job is to be swapped out onto the disk (drum) or brought back into core.

In a swapping system, jobs are retained in queues of varying priorities that reflect the status of the jobs at any given moment. Each job number possible in the system resides in only one queue at any time. A job may be in one of the following queues:

a. Run queues - for runnable jobs waiting to execute. (There are three run queues of different levels of priorities.)

b. I/O wait queue - for jobs waiting while doing I/O.

c. I/O wait satisfied queue - for jobs waiting to run after finishing I/O.

d. Sharable device wait queue - for jobs waiting to use sharable devices.

e. TTY wait queue - for jobs waiting for input or output on the user's console.

f. TTY wait satisfied queue - for jobs that completed a TTY operation and are awaiting action.

g. Stop queue - for processes that have been completed or aborted by an error and are awaiting a new command for further action.

h. Null queue - for all job numbers that are inactive (unassigned).

Each queue is addressed through a table. The position of a queue address in a table represents the priority of the queue with respect to the other queues. Within certain queues, the position of a job determines its priority with respect to the other jobs in the same queue. For example, if a job is first in the queue for a sharable device, it has the highest priority for the device when it becomes available. However, if a job is in an I/O wait queue, it remains in the queue until the I/O is completed. Therefore, in an I/O wait queue, the job's position has no significance. The status of a job changes each time it is placed into a different queue.

Each job, when it is assigned to run, is given a quantum time. When the quantum time expires, the job ceases to run and moves to a lower priority run queue. The activities of the job currently running may cause it to move out of the run queue and enter one of the wait queues. For example: when a currently running job begins input from a DECtape, it is placed in the I/O wait queue, and the input is begun. A second job is set to run while the input of the first job proceeds. If the second job then decides to access a DECtape for an I/O operation, it is stopped because the DECtape control is busy, and it is put in the queue for jobs waiting to access the DECtape control. A third job is set to run. The input operation of the first job finishes, making the DECtape control available to the second job. The I/O operation of the second job is initiated, and the job is transferred from the device wait queue to the I/O wait queue. The first job is transferred from the I/O wait queue to the highest priority run queue. This permits the first job to preempt the running of the third job. When the quantum time of the first job becomes zero, it is moved into the second run queue, and the third job runs again until the second job completes its I/O operations.

Data transfers also use the scheduler to permit the user to overlap computation with data transmission. In unbuffered modes, the user supplies an address of a command list containing pointers to relative locations in the user area to and from which data is to be transferred. When the transfer is initiated, the job is scheduled into an I/O wait queue where it remains until the device signals the scheduler that the entire transfer has been completed.

In buffered modes, each buffer contains a use bit to prevent the user and the device from using the same buffer at the same time (refer to Paragraph 4.3). If the user overtakes the device and requires the buffer currently being used by the device as his next buffer, the user's job is scheduled into an I/O wait queue. When the device finishes using the buffer, the device calls the scheduler to reactivate the job. If the device overtakes the user, the device is stopped at the end of the buffer and is restarted when the user finishes with the buffer.

Scheduling occurs at each clock tick (1/60th or 1/50th of a second) or may be forced at monitor level between clock ticks if the current job becomes blocked (unrunnable). The asynchronous swapping algorithm is also called at each clock tick and has the task of bringing a job from disk into core. This function depends on

a. The core shuffling routine, which consolidates unused areas in core to make sufficient room for the incoming job,

b. The swapper, which creates additional room in core by transferring jobs from core to disk.

Therefore, when the scheduler is selecting the next job to be run, the swapper is bringing the next job to be run into core. The transfer from disk to core takes place while the central processor continues computation for the previous job.

7.2 PROGRAM SWAPPING

Program swapping is performed by the monitor on one or more units of the system independent of the file structures that may also use the units. Swapping space is allocated and deallocated in clusters of 1K words (exactly); this size is the increment size of the memory relocation and protection mechanism. Directories are not maintained, and retrieval information is retained in core. Most user segments are written onto the swapping units as contiguous units. Swapping time and retrieval information is, therefore, minimized. Segments are always read completely from the swapping unit into core with one I/O operation. The swapping space on all units appears as a single system file, SWAP.SYS, in directory SYS in each file structure. This file is protected from all but privileged programs by the standard file protection mechanism (refer to Paragraph 6.2.3).

The reentrant capability reduces the demands on core memory, swapping space, swapping channel, and storage channel; however, to reduce the use of the storage channel, copies of sharable segments are kept on the swapping device. This increases the demand for swapping space. To prevent the swapping space from being filled by user's files and to keep swapped segments from being fragmented, swapping space is preallocated when the file structure is refreshed. The monitor dynamically achieves the space-time balance by assuming that there is no shortage of swapping space. Swapping space is never used for anything except swapped segments, and the monitor keeps a single copy of as many segments as possible in this space. (The maximum number of segments that may be kept may be increased by individual installations but is always at least as great as the number of jobs plus one.) If a sharable segment on the swapping space is currently unused, it is called a dormant segment. An idle segment is a sharable segment that is not used by users in core; however, at least one swapped-out user must be using the segment or it would be a dormant segment.

Swapping disregards the grouping of similar units into file structures; therefore, swapping is done on a unit basis rather than a file structure basis. The units for swapping are grouped in a sorted order, referred to as the active swapping list. The total virtual core, which the system can allocate to users, is equal to the total swapping space preallocated on all units in the active swapping list. In computing virtual core, sharable segments count only once, and dormant segments do not count at all. The monitor does not allow more virtual core to be granted than the system has capacity to handle.

When the system is started, the monitor reads the home blocks on all the units that it was generated to handle. The monitor determines from the home blocks which units are members of the active swapping list. This list may be changed at once-only time. The change does not require refreshing of the file structures, as long as swapping space was preallocated on the units when they were refreshed. All of the units with swapping space allocated need not appear in the active swapping list. For example: a drum and disk pack system should have swapping space allocated on both drum and disk packs. Then, if the drum becomes inoperable, the disk packs may be used for swapping without refreshing.

Users cannot proceed when virtual core is exhausted; therefore, FILSER is designed to handle a variety of disks as swapping media. The system administrator allocates additional swapping space on slower disks and virtually eliminates the possibility of exhausting virtual core; therefore, in periods of heavy demand, swapping is slower for segments that must be swapped on the slower devices. It is also undesirable to allow dormant segments to take up space on high-speed units. This forces either fragmentation on fast units or swapping on slow units; therefore, the allocation of swapping space is important to overall system efficiency.

The swapping allocator is responsible for assigning space for the segment the swapper wants to swap out. It must decide

a. Onto which unit to swap the segment.
b. Whether to fragment the unit if not enough contiguous space is available.
c. Whether to make room by deleting a dormant segment.
d. Whether to use a slower unit.

The units in the active swapping list are divided into swapping classes, usually according to device speed. For simplicity, the monitor assumes that all the units of class 0 are first followed by all the units of class 1. Swapping classes are defined when the file structures are refreshed and may be changed at once-only time.

When attempting to allocate space to swap out a low or high segment, the monitor performs the following:

Step	Procedure
1	The monitor looks for contiguous space on one of the units of the first swapping class.
2	The monitor looks for noncontiguous space on one of the units in the same class.
3	The monitor checks whether deleting one or more dormant segments would yield enough contiguous or noncontiguous space.

If all of these measures fail, the monitor repeats the process on the next swapping class in the active swapping list. If none of the classes yield enough space, the swapper begins again and deletes enough dormant segments to fragment the segment across units and classes. When a deleted segment is needed again, it is retrived from the storage device.

7.3 DEVICE OPTIMIZATION

7.3.1 Concepts

Each I/O operation on a unit consists of two steps: positioning and data transferring. To perform I/O, the unit must be positioned, unless it is already on a cylinder or is a non-positioning device. To position a unit, the controller cannot be performing a data transfer. If the controller is engaged in a data transfer, the positioning operation of moving the arm to the desired cylinder cannot begin until the data transfer is complete.

The controller ensures that the arms have actually moved to the correct cylinder. This check is called verification, and the time required is fixed by hardware. If verification fails, the controller interrupts the processor, and the software recalibrates the positioner by moving it to a fixed place and beginning again. When verification is complete, the controller reads the sector headers to find the proper sector on which to perform the operation. This operation is called searching. Finally, the data is transferred to or from the desired sectors. To understand the optimization, the transfer operation includes verification, searching, and the actual transfer. The time from the initiation of the transfer operation to the actual beginning of the transfer is called the latency time. The channel is busy with the controller for the entire transfer time; therefore, it is important for the software to minimize the latency time

The FILSER code, the routines that queue disk requests and make optimization decisions, handles any number of channels and controllers and up to eight units for each controller.[1] Optimization is designed to keep:

a. As many channels as possible performing data transfers at the same time.
b. As many units positioning on all controllers, which are not already in position for a data transfer.

Several constraints are imposed by the hardware. A channel can handle only one data transfer on one control at a time. Furthermore, the control can handle a data transfer on only one of its units at a time. However, the other units on the control can be positioning while a data transfer is taking place provided the positioning commands were issued prior to the data transfer. Positioning requests for a unit on a controller that is busy doing a data transfer for another of its units must be queued until the data transfer is finished. When a positioning command is given to a unit through a controller, the controller is busy for only a few microseconds; therefore, the software can issue a number of positioning commands to different units as soon as a data transfer is complete. All units have only positioning mechanism that reaches each point; therefore, only one positioning operation can be performed on a unit at the same time. All other positioning requests for a unit must be queued.

[1] Disk latency optimization depends on FTDOPT which is normally off in the DECsystem-1040. If this switch is off, all requests are handled on a first-come first-served basis.

The software keeps a state code in memory for each active file, unit, controller, and channel, to remember the status of the hardware. Reliability is increased because the software does not depend on the status information of the hardware. The state of a unit is as follows:

I	Idle; No positions or transfers waiting or being performed.
SW	Seek Wait; Unit is waiting for control to become idle so that it can initiate positioning (refer to Paragraph 6.2).
S	Seek; Unit is positioning in response to a SEEK UUO; no transfer of data follows.
PW	Position Wait; Unit is waiting for control to become idle so that it can initiate positioning.
P	Position; Unit is positioning; transfer of data follows although not necessarily on this controller.
TW	Transfer Wait; Unit is in position and is waiting for the controller/channel to become idle so that it can transfer data.
T	Transfer; Unit is transferring; the controller and channel are busy performing the operation.

Table 7-1 lists the possible states for files, units, controllers, and channels.

Table 7-1
Software States

File†	Unit	Controller	Channel
I	I	I	I
	SW		
	S		
PW	PW		
P	P		
TW	TW		
T	T	T	T

†Cannot be in S or SW state because SEEKs are ignored if the unit is not idle.

7.3.2 Queuing Strategy

When an I/O request for a unit is made by a user program because of an INPUT or OUTPUT UUO, one of several things can happen at UUO level before control is returned to the buffer-strategy module in UUOCON, which may, in turn, pass control back to the user without rescheduling. If an I/O request requires positioning of the unit, either the request is added to the end of the position-wait queue for

the unit if the control or unit is busy, or the positioning is initiated immediately. If the request does not require positioning, the data is transferred immediately. If the channel is busy, the request is added to the end of the transfer-wait queue for the channel. The control gives the processor an interrupt after each phase is completed. Optimization occurs at interrupt level when a position-done or transfer-done interrupt occurs.

7.3.2.1 Position-Done Interrupt Optimization - The following action occurs only if a transfer-done interrupt does not occur first. Data transfer is started on the unit unless the channel is busy transferring data for some other unit or control. If the channel is busy, the request goes to the end of the transfer-wait queue for that channel.

7.3.2.2 Transfer-Done Interrupt Optimization - When a transfer-done interrupt occurs, all the position-done interrupts inhibited during the data transfer are processed for the controller, and the requests are placed at the end of the transfer-wait queue for the channel. All units on the controller are then scanned. The requests in the position-wait queues on each unit are scanned to see the request nearest the current cyclinder. Positioning is begun on the unit of the selected request. All requests in the transfer-wait queue for all units on the channel that caused the interrupt are then scanned and the latency time is measured. The request with the shortest latency time is selected, and the new transfer begins.

7.3.3 Fairness Considerations

When the system selects the best task to run, users making requests to distant parts of the disk may not be serviced for a long time. The disk software is designed to make a fair decision for a fixed percentage of time. Every n decisions the disk software selects the request at the front of the position-wait or transfer-wait queue and processes it, because that request has been waiting the longest. The value of n is set to 10 (decimal) and may be changed by redefining symbols with MONGEN.

7.3.4 Channel Command Chaining

7.3.4.1 Buffered Mode - Disk accesses are reduced by using the chaining feature of the data channel. Prior to reading a block in buffered mode, the device independent routine checks to see if there is another empty buffer, and if the next relative block within the file is a consecutive logical block within the unit. If both checks are true, FILSER creates a command list to read two or more consecutive blocks into scattered core buffers. Corresponding decisions are made when writing data in buffered mode, and, if possible, two or more separate buffers are written in one operation. The command chaining decision is not made when a request is put into a position-wait or transfer-wait queue;

instead, it is postponed until the operation is performed, thus increasing the chances that the user program will have more buffers available for input or output. The default size of the channel command list is 20 decimal words, and can be changed by redefining CCWMAX with MONGEN.

7.3.4.2 Unbuffered Mode - Unbuffered modes do not use channel chaining, and therefore, read or write one command word at a time. Each command word begins at the beginning of a 128-word block. If a command word does not contain an even multiple of 128 words, the remaining words of the last block are not read, if reading, and are written with zeroes, if writing.

7.4 MONITOR ERROR HANDLING

The monitor detects a number of errors. If a hardware error is detected, the monitor repeats the operation ten times. If the failure occurs eleven times in a row, it is classified as a hard error. If the operation succeeds after failing one to ten times, it is a soft error.

7.4.1 Hardware Detected Errors

Hardware detected errors are classified either as device errors or as data errors. A device error indicates a malfunction of the controller or channel. A data error indicates that the hardware parity did not check or a search for a sector header either did not succeed or had bad parity (the user's data is probably bad).

A device error sets the IO.DER bit in the channel status word, and a data error sets the IO.DTE bit.

Disk units may have imperfect surfaces; therefore, a special non-timesharing diagnostic program, MAP, is provided to initially find all the bad blocks on a specified unit. The logical disk addresses of any bad regions of one or more bad blocks are recorded in the bad allocation table (BAT) block on the unit. The monitor allocates all storage for files; therefore, it uses the BAT block to avoid allocating blocks that have previously proven bad. The MAP program writes two copies of the BAT block because the BAT block might be destroyed. If the MAP program is not used, the monitor discovers the bad regions when it tries to use them and adds this information to the BAT block. However, the first user of the bad region loses that part of his data.

A hard data error usually indicates a bad surface; therefore, the monitor never returns the bad region to free storage. This results in the bad region causing an error only once. The bad unit and the logical disk address are stored in the retrieval information block (RIB) of the file when the file is CLOSEd or RESET and the extent of the bad region is determined. The origin and length of the bad region is stored in the bad allocation table (BAT) block.

7.4.2 Software Detected Errors

The monitor makes a number of software checks on itself. It checks the folded checksum (refer to Appendix H) computed for the first word of every group and stored in the retrieval pointer. The monitor also checks for inconsistencies when comparing locations in the retrieval information block with expected values (filename, filename extension, project-programmer number, special code, logical block number). The monitor checks for inconsistencies in the storage allocation table block when comparing the number of free clusters expected with the number of zeroes. A checksum error or an inconsistency error in the SAT block or RIB normally indicates that the monitor is reading the wrong block. When these errors occur, the monitor sets the improper mode error bit (IO.IMP) in the user channel status word and returns control to the user program.

7.5 DIRECTORIES

7.5.1 Order of Filenames

In 5.02 and earlier monitors, the names of newly created files are appended to the directory if the directory does not contain more than 64 filenames. If the directory contains more than 64 filenames, a second block is used for the new filenames. When filenames are deleted from the first block, entries from the second block are not moved into the first. When additional new files are created, their names are added to the end of the first block of the directory instead of the end of the directory. Thus, the order of the filenames in the directory may not be according to the date of creation.

In 5.03 and later monitors, if FTDUFD = 1, files are always entered in the directory in the order in which they are created. In the DECsystem-1040, FTDUFD is normally off indicating that the order of filenames is the same as in the 5.02 and earlier monitors.

7.5.2 Directory Searches

Table space in core memory is used to reduce directory searching times. The JBTPPB table contains pointers to a list of four-word blocks for the user's project-programmer number, one block for each file structure on which the user has a UFD.

Four-word name and access blocks contain copies of LOOKUP information for recently-accessed files and may reduce disk accesses to one directory read for a LOOKUP on a recently-active file. Recent LOOKUP failures are also kept in core, but are deleted when space is needed.

7.6 PRIORITY INTERRUPT ROUTINES

7.6.1 Channel Interrupt Routines

Each of the seven PI channels has two abolute locations associated with it in memory: 40+2n and 41+2n, where n is a channel number (1-7). When an interrupt occurs on a channel, control is immediately transferred to the first of the two associated locations (unless an interrupt on a higher priority

channel is being processed). For fast service of a single device, the first location contains either a BLKI or BLKO instruction. For service of more than one device on the same channel, the first location contains a JSR to location CHn in the appropriate channel interrupt routine. The JSR ensures that the current state of the program counter is saved.

Each channel interrupt routine (mnemonic name, CHANn, where n is the channel number) consists of three separate routines:

CHn:	The contents of the program counter is saved in location CHn. CHn+1 contains a JRST to the first device service routine in the interrupt chain.
SAVCHn:	The routine to save the contents of a specified number of accumulators. It is called from the device service routines with a JSR.
XITCHn:	The routine to restore saved accumulators. Device service routines exit to XITCHn with a POPJ PDP, if SAVCHn was previously called.

7.6.2 Interrupt Chains

Each device routine contains a device interrupt routine DEVINT where DEV is the three-letter mnemonic for the device concerned. This routine checks to determine whether an interrupt was caused by device DEV. The interrupt chain of a given channel is a designation for the logical positioning of each device interrupt routine associated with that channel.

The monitor flow of control on the interrupt level through a chain is illustrated below. Channel 5 is used in the example.

```
Monitor Routine     Relevant Code                          Explanation

Absolute            52/JSR CH5                             ;control transferred here
Locations           53/                                    ;on interrupt
                       ↓
CHAN5               CH5: 0                                 ;contents of PC saved here
                       JRST PTPINT                         ;control transfers to first
                          ↓                                ;link in interrupt chain
                    PTPINT: CONSO PTP,PTPDON               ;if PDP done bit is
PTPSER                 JRST LPTINT                         ;on, PTP was cause
                          :                                ;of interrupt -
                          :                                ;otherwise, go to
                          ↓                                ;next device.
                    LPTINT: CONSO LPT,LPTLOV+LPTERR+LPTDON
LPTSER                 JEN @ CH5                           ;three possible bits
                          :                                ;may indicate that
                          :                                ;LPT caused interrupt
```

When a real-time device is added to the interrupt chain (CONSO skip chain) by a RTTRP UUO (refer to Paragraph 3.8.1), the device is added to the front of the chain. After putting a real-time device on Channel 5 in single mode (refer to Paragraph 3.8.1), the chain is as follows:

Monitor Routine	Relevant Code	Explanation
Absolute Locations	52/JSR CH5 53/	;control transferred here ;on interrupt
	↓	
CHAN5	CH5: 0	;contents of PC saved here
	JRST RDTINT	;control transfers to first
	↓	;link in interrupt chain
RTDEV	RTDINT: CONSO RTD,BITS	
	JRST PTPINT	
	JRST <context switcher and dispatch for real-time interrupts >	
	PTPINT: CONSO PTP,PTPDON	;if PTP done bit is
PTPSER	JRST LPTINT	;on, PTP was cause
	:	;of interrupt -
	:	;otherwise, go to
	↓	;next device.
	LPTINT:CONSO LPT, LPTLOV+LPTERR+LPTDON	
LPTSER	JEN @ CH5	;three possible bits
	:	;may indicate that
	:	;LPT caused interrupt

After putting a real-time device on channel 5 in normal block mode (refer to Paragraph 3.8.1), the chain is as follows:

Monitor Routine	Relevant Code	Explanation
Absolute Locations	52/JSR CH5 53/	;control transferred here ;on interrupt
	↓	
CHAN5	CH5: 0	;contents of PC saved here
	JRST RTDINT	;control transfers to first
	↓	;link in interrupt chain
RTDEV	RTDINT:CONSO RTD,BITS	
	JRST PTPINT	
	BLKI RTD,POINTR	
	JRST <context switcher >	
	JEN @ CH5	

(continued on next page)

Monitor Routine	Relevant Code	Explanation
PTPSER	PTPINT: CONSO PTP,PTPDON JRST LPTINT : : ↓	;if PTP done bit is ;on, PTP was cause ;of interrupt - ;otherwise, go to ;next device.
LPTSER	LPTINT:CONSO LPT, LPTLOV+LPTERR+LPTDON JEN @ CH5 : :	;three possible bits ;may indicate that ;LPT caused interrupt.

After putting a real-time device on channel 6 in fast block mode (refer to Paragraph 3.8.1), the chain is as follows:

Monitor Routine	Relevant Code	Explanation
Absolute Locations	54/BLKO RTD,POINTR 55/JSR CH6	;control transferred ;here on interrupt
CHAN6	CH6: 0 JRST <context switcher >	;contents of PC saved ;control transfers to ;context switcher.

The exec mode trapping feature can be used with any of the standard forms of the RTTRP UUO: single mode, normal block mode, and fast block mode. The following examples illustrate the chain when used with each of the three modes.

Single Mode (Exec Mode)

Monitor Routine	Relevant Code	Explanation
Absolute Locations	52/JSR CH5 53/ ↓	;control transferred here ;on interrupt
CHAN5	CH5: 0 JRST RDTINT ↓	;contents of PC saved here ;control transfers to first ;link in interrupt chain
RTDEV	RTDINT: CONSO RTD,BITS JRST PTPINT JSR TRPADR JEN @ CH5	

(continued on next page)

Single Mode (Exec Mode) (Cont)

Monitor Routine	Relevant Code	Explanation
PTPSER	PTPINT:CONSO PTP,PTPDON JRST LPTINT ↓	;if PTP done bit is ;on, PTP was cause ;of interrupt - ;otherwise, go to ;next device.
LPTSER	LPTINT:CONSO LPT, LPTLOV+LPTERR+LPTDON JEN @ CH5 ⋮	;three possible bits ;may indicate that ;LPT caused interrupt

Normal Block Mode (Exec Mode)

Monitor Routine	Relevant Code	Explanation
Absolute Locations	52/JSR CH5 53/ ↓	;control transferred here ;on interrupt
CHAN5	CH5: 0 JRST RTDINT ↓	;contents of PC saved here ;control transfers to first ;link in interrupt chain
RTDEV	RTDINT:CONSO RTD,BITS JRST PTPINT BLKI RTD,POINTR JSR TRPADR JEN @ CH5	
PTPSER	PDPINT: CONSO PTP,PTPDON JRST LPTINT ↓	;if PTP done bit is ;on, PDP was cause ;of interrupt - ;otherwise, go to ;next device.
LPTSER	LPTINT:CONSO LPT,LPTLOV+LPTERR+LPTDON JEN @ CH5 ⋮	;three possible bits ;may indicate that ;LPT caused interrupt.

Fast Block Mode (Exec Mode)

Monitor Routine	Relevant Code	Explanation
Absolute Locations	54/BLKO RTD,POINTR 55/JSR CH6 ↓	;control transferred here ;on interrupt

(continued on next page)

Fast Block Mode (Exec Mode) (Cont)

Monitor Routine	Relevent Code	Explanation
CHAN6	CH6: 0 JRST RDTINT ↓	;contents of PC saved here ;control transfers to first ;link in interrupt chain
RTDEV	RTDINT: JSR TRPADR JEN @ CH6	

7.7 MEMORY PARITY ERROR ANALYSIS, REPORTING, AND RECOVERY[1]

The memory parity error analysis and recovery software allows the machine to run with PARITY STOP off, thereby gaining increased CPU speed (10% more on the KA10 processor and 100% more on the KI10 processor), better error reporting, and improved failsoft recovery. The analysis software considers its goals to be

1) Never jeopardize the system or the user program by allowing it to continue with bad data from memory.
2) Always maintain the running of the system with the maximum number of users as possible as long as there is no possibility of violating the integrity of the system or the user program.

In either case, complete information is printed for the operator so that he can reconfigure the memories and reload the system when necessary. Additional information is recorded on the disk by DAEMON for field service in order that the cause of the error can be located and fixed.

7.7.1 Description of Analysis

The error analysis software differentiates between user mode and executive mode when a parity error occurs. If the processor is executing in user mode and the user is enabled for parity trapping (refer to Paragraph 3.1.3.1), control is transferred to the user's routine. Otherwise, the execution of the user's job is stopped and the user receives the error messages

```
?ERROR IN JOB n
?MEM PAR ERR AT USER PC nnnnnn
```

Simultaneously, a request is made for the lowest priority channel routine to sweep through core in order to locate all words with bad memory parity, in case there is more than one word. During the sweep, all locations with bad parity are rewritten, so that subsequent references usually will not receive a parity error. After the sweep of core is completed, all jobs (including the current job) with

---

[1]This feature depends on FTMEMPAR which is normally off in the DECsystem-1040.

parity errors in their low segments receive the above ERROR IN JOB message. All jobs with errors in their high segments are swapped out if the high segment has the hardware user-mode write protect bit set, since a copy exists on the swapping space. In this case recovery occurs for all jobs sharing the high segment except for the currently running job. If the high segment is not write protected for a job (so that there is no copy on the disk), if the high segment is locked, or if one of the sharing job's low segment is locked, all jobs sharing the high segment are stopped and receive an error message since no recovery is possible. In addition, the segment name is cleared so that new users will receive a new copy from the file system on a R, RUN, or GET command or a RUN or GETSEG UUO.

If the processor is in executive mode when the error occurs, the analysis procedure depends on the value of the PC. Two conditions are recognized as not being harmful:

1) a parity error during the PI 7 sweep of memory.
2) a parity error during the storing of data words around the location of a channel-detected memory parity error.

If the PC is at the BLT instruction which moves user core to facilitate core allocation, the bad word is determined from the BLT pointer. If the pointer is in the protected part of the job data area, this area is cleared so the monitor will not attempt to use the bad words, since they contain executive mode addresses. In either case, the user's job is stopped and an error message is output to the user. In addition, the memory sweep procedure is invoked to find additional words with bad parity.

If the PC is an executive mode location and there are no PIs in progress, the UUO is run to completion, the current user receives an error message, and the memory sweep procedure is invoked. If the sweep routine detects bad parity in an address within the monitor or detects no words with bad parity (because they have been rewritten on a read-pause-write instruction), the routine prints on the CTY (instead of OPR),

```
?EXEC PARITY HALT
?n MEM PAR ERRS FROM aaaaaa TO bbbbbb ON CPUn FOR JOBx [program]
```

and then HALTs. This message is printed without using the interrupt system in order to maximize the chances of the message being output. Although the operator can attempt to continue the system by pushing the CONT console switch, this is not a recommended operator procedure (e.g., the monitor may have incorrect data thereby causing more damage). (Refer to MEMPAR in Notebook 8 of the DECsystem-10 Software Notebooks for complete operator instructions on memory parity error recovery.)

If a PI is in progress when the parity error is detected, a sweep of core is made at the high priority APR PI level. If a word with bad parity is discovered in the monitor area or no parity errors are found, the monitor prints the above message to the operator and halts. The finding of words without bad parity is considered serious because the read-pause-write class instructions rewrite memory before the parity interrupt occurs so that the parity error is usually corrected. In this case, the operator receives the message

```
?0 MEM PAR ERRORS
```

On all recoverable or non-recoverable parity errors, the operator receives on either OPR or CTY a message similar to the following:

?n MEM PAR ERRS FROM aaaaaa TO bbbbbb ON CPUn for JOBx [program]

preceded by 5 bells. This alerts him to potential problems and gives him the necessary information for reconfiguring the memories. In addition, the operator is notified of the jobs that have been stopped in case they are crucial to the operation of the system. If they are, he can take appropriate action to restart them.

If the DF10 channel detects a memory parity error while reading for file I/O from memory, the user's job is not stopped and the user does not receive an error message. Instead the error is treated as a device error and the IO.DER error bit is set. However, the operator receives the message

?n MEM PAR ERRS FROM aaaaaa TO bbbbbb ON CHANNEL n

where n is the logical channel number starting with the fastest device as defined by MONGEN. For example, the fastest disk unit is on the first channel and the magnetic tape TM10B control is on the last channel.

If the DF10 channel detects a memory parity error while swapping a job out of core, the user's job is stopped and the user receives the following error message:

?ERROR IN JOB n
?SWAP OUT CHN MEM PAR ERR

The operator receives the message

?m MEM PAR ERRS FROM aaaaaa TO bbbbbb ON CHANNEL n FOR JOB x [prog]

If the error is detected in a high segment on the swap out, all jobs using the high segment receive the error message. The high segment name is cleared so that new users will receive a new copy of the segment from the file system.

On all parity errors detected by the processors or the channels, DAEMON is awakened to correct the information stored by the monitor's analysis routine. DAEMON writes this information in the hardware log file on the disk for the use of field service in diagnosing and solving the problem.

APPENDIX A
DECTAPE COMPATIBILITY BETWEEN DEC COMPUTERS

The following chart illustrates the ability to read the indicated tapes with a suitable program. In general, the standard software of machines of one family will not read tapes written by the standard software of machines of a different family.

The standard tapes of the PDP-1, PDP-4, PDP-6, PDP-7, PDP-9, PDP-10, PDP-11, and PDP-15 consist of 578 blocks of 128 36-bit words (256 18-bit words). The standard tapes of the PDP-15 and the PDP-8 family consist of 4096 blocks of 129 12-bit words (43 36-bit words).

Written by \ Read by	PDP-1 550,550-A And 555, TU55, TU56	PDP-4 550 And 555, TU55, TU56	PDP-5 552 And 555,TU55, TU56	PDP-6 551 And 555,TU55 TU56	PDP-7 550-A And 555,TU55, TU56	PDP-8 552,TC01 And 555,TU55, TU56	PDP-8/I,L TC01,TC08 And TU55,TU56	PDP-8/E TC08-P, TD8E And TU55, TU56	LINC-8 And Optional Converter And LINCtape Drive	PDP-9 TC02 And TU55, TU56	PDP-10 TD10 And TU55,TU56	PDP-11 TC11 And TU56	PDP-12 TC12 And TC12-F TU55, TU56	PDP-15 TC02,TC15 And TU55,TU56
PDP-1	A	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
PDP-4	Z	A	D	D	A	D	D	D	E	D	D	D	E	D
PDP-5	Z	D	A	B	C	A	A	A	E	A	A	A	E	A
PDP-6	Z	D	A	A	C	A	A	A	E	A	A	A	E	A
PDP-7	Z	A	C	C	A	C	C	C	E	C	C	C	E	C
PDP-8	Z	D	A	B	C	A	A	A	E	A	A	A	E	A
PDP-8/I,L	Z	D	A	B	C	A	A	A	E	A	A	A	E	A
PDP-8/E	Z	D	A	B	C	A	A	A	E	A	A	A	E	A
LINC-8 & Converter	Z	E	E	B	E	E	E	E	A	E	E	E	A	E
PDP-9	Z	D	A	A	C	A	A	A	E	A	A	A	E	A
PDP-10	Z	D	A	A	C	A	A	A	E	A	A	A	E	A
PDP-11	Z	D	A	B	C	A	A	A	E	A	A	A	E	A
PDP-12 & TC12-F	Z	E	E	B	E	E	E	E	A	E	E	E	A	E
PDP-15	Z	D	A	A	C	A	A	A	E	A	A	A	E	A

KEY: A = Can be done.

B = Can be done only by ignoring indicated checksum errors.

C = Can be done with programmed checksum.

D = Can probably be done as in (C) except that PDP-4 is too slow for calculating the exclusive-or checksum in line; calculations must be done before writing and after reading.

E = Program and optional hardware exist to convert to and from LINCtape format. Standard PDP-12 and LINC-8 tapes are in LINCtape format which is incompatible. DECtapes must be formatted on another machine before writing on PDP-12 or LINC-8.

Z = No information available.

NOTES:

1. PDP-8/S cannot use DECtape. Classic LINC only uses LINCtape which is incompatible with DECtape.
2. The PDP-6 and 10 (and probably other machines) cannot find the first or last block when searching from the end zone.
3. The PDP-9 and -15 software writes data in reverse order in blocks which are written while moving in the reverse direction.

APPENDIX B
WRITING REENTRANT USER PROGRAMS

B.1 DEFINING VARIABLES AND ARRAYS

The LOADER simplification makes it somewhat more difficult to define variables and arrays. The easiest way to define variables and arrays, so the resulting relocatable binary can be loaded on a one- or two-segment machine, is to put them all in a separate subprogram as internal global symbols using Block 1 and Block N pseudo-ops. All other subprograms refer to this data as external global locations. Most reentrant programs have at least two subprograms, one for the definition of low segment locations and one for instructions and constants for the high segment. (This last subprogram must have either a HISEG pseudo-op or a TWOSEG pseudo-op followed by RELOC 400000.) Programs are self-initializing; therefore, they clear the low segment when they are started although the monitor clears core when it assigns it to a user.

Block 1 and Block N pseudo-ops cause the LOADER to leave indications in the job data area (LH of JOBCOR) so a monitor SAVE command will not write the low segment. This is advantageous in sharable programs for two reasons. It reduces the number of files in small DECtape directories (the maximum is 22 files). Also, I/O is accomplished only on the first user's GET that initializes the high segment, but not on any subsequent user's GETs for either the high or low segment.

B.2 EXAMPLE OF TWO-SEGMENT REENTRANT PROGRAM

```
LOW SEGMENT SUBPROGRAM:

TITLE LOW - EXAMPLE OF LOW SEGMENT SUB-PROGRAM
        JOBVER=137
        LOC     JOBVER
        3                       ;VERSION3
        RELOC   0
        INTERNAL LOWBEG,DATA,DATA1,DATA2,TABLE,TABLE1

LOWBEG:
DATA:   BLOCK   1
DATA1:  BLOCK   1
DATA2:  BLOCK   1

TABLE:  BLOCK   10
TABLE1: BLOCK   10
LOWEND--1                       ;LAST LOCATION TO BE CLEARED
        END
```

```
HIGH SEGMENT SUBPROGRAM:

TITLE HIGH - EXAMPLE OF HIGH SEGMENT SUB-PROGRAM
        HISEG                           ;OR TWOSEG
                                        ;RELOC 400000
        EXTERN  LOWBEG,LOWEND
        T=1
BEGIN:  SETZM   LOWBEG                  ;CLEAR DATA AREA
        MOVEI   T,LOWBEG+1
        HRLI    T,LOWBEG
        BLT     T,LOWEND
        MOVE    T,DATA1                 ;COMPUTE
        ADDI    1,1
        MOVEM T, DATA2
           .
           .
           .
       END      BEGIN                   ;STARTING ADDRESS
```

B.3 CONSTANT DATA

Some reentrant programs require certain locations in the low segment to contain constant data, which does not change during execution. The initialization of this data happens only once after each GET, instead of after each START; therefore, programmers are tempted to place these constants in the sub-program that contains the definition of the variable data locations. This action requires the SAVE command to write the constants out and the GET command to load the constants again; therefore, the constant data should be moved by the programs from the high segment to the low segment when the rest of the low segment is being initialized. The exception is when the amount of code and constants in the high segment needed to initialize the low segment constants take up too much room in the high segment. In this case, it is best to have I/O in the low segment on each GET. A rule to follow in deciding between this high segment core space and the low segment GET I/O time is: put the code in the high segment.if it does not put the high segment over the next 1K boundary.

B.4 SINGLE SOURCE FILE

A second way of writing single save file reentrant programs is to have a single source file instead of two separate ones. This is more convenient, although it involves conditional assembly and, therefore, produces two different relocatable binaries. A number of system programs have been written this way. The idea is to have a conditional switch which is 1 if a reentrant assembly and 0 if a non-reentrant assembly.

```
DEMO - DEMO ONE SOURCE REENTRANT PROGRAM -V002  MACRO 44,0.4 14:30  2-APR-71 PAGE 1
DEMO

     1                                  TITLE DEMO - DEMO ONE SOURCE REENTRANT PROGRAM -V002
     2
     3  000137                                  LOC <JOBVER=137>
     4  000137  000000  000002                 EXP     002             ;VERSION NUMBER
     5  000000'                                 RELOC
     6
     7                                          INTERN  JOBVER,PURE
     8
     9                                  IFNDEF PURE,<PURE==1>           ;ASSUME REENTRANT IF PURE UNDEFINED
    10  400000'                                 IFN PURE,<TWOSEG>       ;TELL LOADER TO EXPECT TWO SEGMENTS
    11  400000'                                 IFN PURE,<RELOC 400000> ;START OF HIGH SEGMENT RELOCATION
    12
    13  400000' 047000  000000  BEG:    RESET                           ;RESET ALL I/O
    14  400001' 200000  400007'         MOVE    0,[DATAB,,DATAB+1]
    15  400002' 402000  000000'         SETZM   DATAB                   ;NOW CLEAR DATA REGION
    16  400003' 251000  000203'         BLT     0,DATAE-1               ;UP TO LAST LOCATION
    17  400004' 000000  400004'                 .
    18  400005' 000000  400005'                 .
    19  400006' 000000  400006'                 .
    20
    21  000000'                         IFN PURE,<RELOC>                ;SET RELOCATION COUNTER TO LOW SEGMENT
    22
    23  000000'                         DATAB:                          ;FIRST LOCATION CLEARED ON STARTUP
    24  000000'                         DATA:   BLOCK   1
    25  000001'                         TABLE:  BLOCK   200
    26  000201' 000000  000201'                 .
    27  000202' 000000  000202'                 .
    28  000203' 000000  000203'                 .
    29  000204'                         DATAE:                          ;END OF DATA AREA
    30
    31  400007'                         IFN PURE,<RELOC>                ;BACK TO HIGH SEGMENT
    32  400007'                                 LIT                     ;PUT LITERALS IN HIGH SEGMENT
    33  400007' 000000' 000001'
    34                  400000'                 END     BEG

NO ERRORS DETECTED

HI-SEG. BREAK IS 400010
PROGRAM BREAK IS 000204

2K CORE USED
```

APPENDIX C
CARD CODES

Table C-1
ASCII Card Codes

ASCII Character	Octal Code	Card Punches	ASCII Character	Octal Code	Card Punches
NULL	00	12-0-9-8-1	@	100	8-4
CTRL-A	01	12-9-1	A	101	12-1
CTRL-B	02	12-9-2	B	102	12-2
CTRL-C	03	12-9-3	C	103	12-3
CTRL-D	04	9-7	D	104	12-4
CTRL-E	05	0-9-8-5	E	105	12-5
CTRL-F	06	0-9-8-6	F	106	12-6
CTRL-G	07	0-9-8-7	G	107	12-7
CTRL-H	10	11-9-6	H	110	12-8
TAB	11	12-9-5	I	111	12-9
LF	12	0-9-5	J	112	11-1
VT	13	12-9-8-3	K	113	11-2
FF	14	12-9-8-4	L	114	11-3
CR	15	12-9-8-5	M	115	11-4
CTRL-N	16	12-9-8-6	N	116	11-5
CTRL-O	17	12-9-8-7	O	117	11-6
CTRL-P	20	12-11-9-8-1	P	120	11-7
CTRL-Q	21	11-9-1	Q	121	11-8
CTRL-R	22	11-9-2	R	122	11-9
CTRL-S	23	11-9-3	S	123	0-2
CTRL-T	24	9-8-4	T	124	0-3
CTRL-U	25	9-8-5	U	125	0-4
CTRL-V	26	9-2	V	126	0-5
CTRL-W	27	0-9-6	W	127	0-6
CTRL-X	30	11-9-8	X	130	0-7
CTRL-Y	31	11-9-8-1	Y	131	0-8
CTRL-Z	32	9-8-7	Z	132	0-9
ESCAPE	33	0-9-7	]	133	12-8-2
CTRL-\	34	11-9-8-4	\	134	0-8-2
CTRL-]	35	11-9-8-5	]	135	11-8-2
CTRL-↑	36	11-9-8-6	↑ ^	136	11-8-7
CTRL-←	37	11-9-8-7	←	137	0-8-5
SPACE	40		\ _	140	8-1

NOTE: The ASCII character ESCAPE (octal 33) is also CTRL-[on a terminal.

Table C-1 (Cont)
ASCII Card Codes

ASCII Character	Octal Code	Card Punches	ASCII Character	Octal Code	Card Punches
!	41	12-8-7	a	141	12-0-1
"	42	8-7	b	142	12-0-2
#	43	8-3	c	143	12-0-3
$	44	11-8-3	d	144	12-0-4
%	45	0-8-4	e	145	12-0-5
&	46	12	f	146	12-0-6
'	47	8-5	g	147	12-0-7
(	50	12-8-5	h	150	12-0-8
)	51	11-8-5	i	151	12-0-9
*	52	11-8-4	j	152	12-11-1
+	53	12-8-6	k	153	12-11-2
,	54	0-8-3	l	154	12-11-3
-	55	11	m	155	12-11-4
.	56	12-8-3	n	156	12-11-5
/	57	0-1	o	157	12-11-6
0	60	0	p	160	12-11-7
1	61	1	q	161	12-11-8
2	62	2	r	162	12-11-9
3	63	3	s	163	11-0-2
4	64	4	t	164	11-0-3
5	65	5	u	165	11-0-4
6	66	6	v	166	11-0-5
7	67	7	w	167	11-0-6
8	70	8	x	170	11-0-7
9	71	9	y	171	11-0-8
:	72	8-2	z	172	11-0-9
;	73	11-8-6	{	173	12-0
<	74	12-8-4	|	174	12-11
=	75	8-6	}	175	11-0
>	76	0-8-6	~	176	11-0-1
?	77	0-8-7	DEL	177	12-9-7

NOTE: The ASCII characters } and ~(octal 175 and 176) are treated by the monitor as ALT-MODE and are often considered the same as ESCAPE.

Table C-2
DEC-029 Card Codes

Character	Octal Code	Card Punches	Character	Octal Code	Card Punches
SPACE	40		@	100	8-4
!	41	11-8-2	A	101	12-1
"	42	8-7	B	102	12-2
#	43	8-3	C	103	12-3
$	44	11-8-3	D	104	12-4
%	45	0-8-4	E	105	12-5
&	46	12	F	106	12-6
'	47	8-5	G	107	12-7
(	50	12-8-5	H	110	12-8
)	51	11-8-5	I	111	12-9
*	52	11-8-4	J	112	11-1
+	53	12-8-6	K	113	11-2
,	54	0-8-3	L	114	11-3
-	55	11	M	115	11-4
.	56	12-8-3	N	116	11-5
/	57	0-1	O	117	11-6
0	60	0	P	120	11-7
1	61	1	Q	121	11-8
2	62	2	R	122	11-9
3	63	3	S	123	0-2
4	64	4	T	124	0-3
5	65	5	U	125	0-4
6	66	6	V	126	0-5
7	67	7	W	127	0-6
8	70	8	X	130	0-7
9	71	9	Y	131	0-8
:	72	8-2	Z	132	0-9
;	73	11-8-6	[	133	12-8-2
<	74	12-8-4	\	134	11-8-7
=	75	8-6	]	135	0-8-2
>	76	0-8-6	↑ ^	136	12-8-7
?	77	0-8-7	← _	137	0-8-5

NOTE: Octal codes 0-37 and 140-177 are the same punches as ASCII.

Table C-3
DEC-026 Card Codes

Character	Octal Code	Card Punches	Character	Octal Code	Card Punches
SPACE	40		@	100	8-4
!	41	12-8-7	A	101	12-1
"	42	0-8-5	B	102	12-2
#	43	0-8-6	C	103	12-3
$	44	11-8-3	D	104	12-4
%	45	0-8-7	E	105	12-5
&	46	11-8-7	F	106	12-6
'	47	8-6	G	107	12-7
(	50	0-8-4	H	110	12-8
)	51	12-8-4	I	111	12-9
*	52	11-8-4	J	112	11-1
+	53	12	K	113	11-2
,	54	0-8-3	L	114	11-3
-	55	11	M	115	11-4
.	56	12-8-3	N	116	11-5
/	57	0-1	O	117	11-6
0	60	0	P	120	11-7
1	61	1	Q	121	11-8
2	62	2	R	122	11-9
3	63	3	S	123	0-2
4	64	4	T	124	0-3
5	65	5	U	125	0-4
6	66	6	V	126	0-5
7	67	7	W	127	0-6
8	70	8	X	130	0-7
9	71	9	Y	131	0-8
:	72	11-8-2/11-0	Z	132	0-9
;	73	0-8-2	[	133	11-8-5
<	74	12-8-6	\	134	8-7
=	75	8-3	]	135	12-8-5
>	76	11-8-6	↑ ^	136	8-5
?	77	12-8-2/12-0	← _	137	8-2

NOTE: Octal codes 0-37 and 140-177 are the same punches as ASCII.

APPENDIX D
DEVICE STATUS BITS

Table D-1
Device Status Bits

Device	Function	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
CDP	SETSTS												DEC 029 Card Codes		User Word Count	Data Mode			
	GETSTS		Punch Error		Data when EOC reached		I/O Active												
CDR	SETSTS												Super Image Mode	Sync Input		Data Mode			
	GETSTS	No 7-9 Punch	Data Missed	Binary Checksum Error		EOF card EOF button	I/O Active												
DIS	SETSTS															Data Mode			
	GETSTS						I/O Active												
DSK	SETSTS												Write Headers	Sync Input	User Word Count	Data Mode			
	GETSTS	Write Lock	Search Error	Disk Parity Error	Block No. Too Large	End of File	I/O Active												

Table D-1 (cont)
Device Status Bits

Device	Function	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
DTA	SETSTS											Semi-Standard I/O Mode	Non-structured Dump Mode	Sync Input	User Word Count	Data Mode			
	GETSTS	Write Lock	Data Missed	Parity Error	Block No. Too Large	End of File	I/O Active												
LPT	SETSTS												Suppress Form Feeds		User Word Count	Data Mode			
	GETSTS						I/O Active												
MTA	SETSTS									Write Even Parity	Tape Density		No Retry	Sync Input	User Word Count	Data Mode			
	GETSTS	Write Lock Illegal Operation	Data Missed	Parity Error	Record Too Long	End of File	I/O Active	Load Point Rewinding	End Point										
PLT	SETSTS														User Word Count	Data Mode			
	GETSTS						I/O Active												
PTP	SETSTS														User Word Count	Data Mode			
	GETSTS						I/O Active												
PTR	SETSTS													Sync Input		Data Mode			
	GETSTS	Block Incomplete		Checksum Error		End of Tape	I/O Active												

Table D-1 (Cont)
Device Status Bits

Device Function		18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
PTY	SETSTS GETSTS				Block No. Too Large		I/O Active	PTY Wait	TTY Re-sponse	Monitor Mode						Data Mode			
TTY	SETSTS										Echo of $ Sup-press	Echo Sup-press	Full Char-acter Set	Sync Input	User Word Count	Data Mode			
	GETSTS	TTY Not Assig-ned for image mode input	Ignore Inter-rupt	Echo Fail-ure	Char-acter Lost		I/O Active												

Note 1: SETSTS UUO may set all bits except Bit 23 and GETSTS UUO may return all bits (18-35); however, the two are separated to show those bits normally set by the user program on INIT, OPEN, or SETSTS as distinct from those normally set by the monitor (GETSTS).

Note 2: Unused bits should always have the value 0.

Note 3: Refer to the appropriate device sections in Chapters 5 and 6 for the complete description of each status bit.

APPENDIX E
ERROR CODES

The error codes in Table E-1 are returned in AC on RUN and GETSEG UUOs, in location E + 1 on 4-word argument blocks of LOOKUP, ENTER, and RENAME UUOs, and in the right half of location E + 3 on extended LOOKUP, ENTER, and RENAME UUOs. The codes are defined in the S.MAC monitor file.

Table E-1
Error Codes

Symbol	Code	Explanation
ERFNF%	0	File not found, illegal filename (0,*), filenames do not match (UPDATE), or RENAME after a LOOKUP failed.
ERIPP%	1	UFD does not exist on specified file structures. (Incorrect project-programmer number.)
ERPRT%	2	Protection failure or directory full on DTA.
ERFBM%	3	File being modified (ENTER, RENAME).
ERAEF%	4	Already existing filename (RENAME) or different filename (ENTER after LOOKUP).
ERISU%	5	Illegal sequence of UUOs (RENAME with neither LOOKUP nor ENTER, or LOOKUP after ENTER).
ERTRN%	6	a. Transmission, device, or data error (RUN, GETSEG only). b. Hardware-detected device or data error detected while reading the UFD RIB or UFD data block. c. Software-detected data inconsistency error detected while reading the UFD RIB or file RIB.
ERNSF%	7	Not a saved file (RUN, GETSEG only).
ERNEC%	10	Not enough core (RUN, GETSEG only).
ERDNA%	11	Device not available (RUN, GETSEG only).
ERNSD%	12	No such device (RUN, GETSEG only).
ERILU%	13	Illegal UUO (GETSEG only). No two-register relocation capability.

(continued on next page)

Table E-1 (Cont)
Error Codes

Symbol	Code	Explanation
ERNRM%	14	No room on this file structure or quota exceeded (overdrawn quota not considered).
ERWLK%	15	Write-lock error. Cannot write on file structure.
ERNET%	16	Not enough table space in free core of monitor.
ERPOA%	17	Partial allocation only.
ERBNF%	20	Block not free on allocated position.
ERCSD%	21	Cannot supersede an existing directory (ENTER).
ERDNE%	22	Cannot delete a non-empty directory (RENAME).
ERSNF%	23	Sub-directory not found (some SFD in the specified path was not found).
ERSLE%	24	Search list empty (LOOKUP or ENTER was performed on generic device DSK and the search list is empty).
ERLVL%	25	Cannot create a SFD nested deeper than the maximum allowed level of nesting.
ERNCE%	26	No file structure in the job's search list has both the no-create bit and the write-lock bit equal to zero and has the UFD or SFD specified by the default or explicit path (ENTER on generic device DSK only).
ERSNS%	27	GETSEG from a locked low segment to a high segment which is not a dormant, active, or idle segment. (Segment not on the swapping space.)

APPENDIX F
COMPARISON OF DISK-LIKE DEVICES

Table F-1
Disk Devices

Device Name	Fixed-Head Disk	Drum	Removable Disk Pack(s)	
Manufacturer	Burroughs	Bryant	Memorex, ISS	
Device Type Controller	RD10 RC10	RM10B RC10	RP02 RP10	RP03 RP10
Maximum Disks per Controller	4	4	8	8
Maximum Controllers per System	2	2	3	3
Hardware Mnemonic	DSK	DSK	DPC	DPC
Software Mnemonic	FHA, FHB	FHA, FHB	DPA, DPB, DPC	DPA, DPB, DPC
Capacity Minimum (X10**6 words)	.5	.345	5.2	10.4
Maximum (1 control) (X10**6 words)	2	1.38	41.4	82.8
Blocks/Track	20	30	10	10
Blocks/Cylinder	4000	2700	200	400
Blocks/Unit	4000	2700	40000	80000
Rotational Speed (rpm)	1800	3600	2400	2400
Revolution Time (msec)	33	17	25	25
128-Word Blocks/ Revolution	20	30	10	10
Transfer Rate μs word	13	4.3	15	15

Table F-1 (Cont)
Disk Devices

Device Name	Fixed-Head Disk	Drum	Removable Disk Pack(s)	
Manufacturer	Burroughs	Bryant	Memorex, ISS	
Device Type Controller	RD10 RC10	RM10B RC10	RP02 RP10	RP03 RP10
Seek Time				
Average (msec)	0	0	50	50
Minimum (msec)	0	0	20	20
Maximum (msec)	0	0	80	80
Swapping Times (msec)			(includes 30 ms verify)	
1K	25	13	84	84
4K	73	27	144	144
10K	154	54	256	264
25K	358	120	589	589

NOTE

Although the Bryant drum is a drum in every sense, its software mnemonic is still FHA because it is connected to the system through the fixed head disk control.

APPENDIX G
MAGNETIC TAPE CODES

Table G-1
ASCII Codes and BCD Equivalents

ASCII	Character Symbol	BCD	ASCII	Character Symbol	BCD
040	blank	20	100	@	57
041	!	52	101	A	61
042	"	17	102	B	62
043	#	32	103	C	63
044	$	53	104	D	64
045	%	77	105	E	65
046	&	35	106	F	66
047	'	14	107	G	67
050	(	34	110	H	70
051	)	74	111	I	71
052	*	54	112	J	41
053	+	60	113	K	42
054	,	33	114	L	43
055	-	40	115	M	44
056	.	73	116	N	45
057	/	21	117	O	46
060	ø	12	120	P	47
061	1	01	121	Q	50
062	2	02	122	R	51
063	3	03	123	S	22
064	4	04	124	T	23
065	5	05	125	U	24
066	6	06	126	V	25
067	7	07	127	W	26
070	8	10	130	X	27
071	9	11	131	Y	30
072	:	15	132	Z	31
073	;	56	133	[	75
074	<	76	134	\	36†
075	5	13	135	]	55
076	>	16	136	↑	illegal
077	?	72	137	←	37

† Code used for all illegal codes.

When converting from ASCII to BCD, the following is done for ASCII codes 000-037 and 140-177:

000	ignored.
001-010	same as ASCII 134.
011	same as ASCII 040.
012-014	constitutes end of line.
015	ignored.
016-031	same as ASCII 134.
032	end of file.
033-037	same as ASCII 134.
140	same as ASCII 134.
141-172	same as ASCII 101-132.
173-176	same as ASCII 134.
177	ignored.

APPENDIX H
FILE RETRIEVAL POINTERS

Sequential and random file access are handled more efficiently by the monitor if all the information describing the file can be kept in core at once. To understand this effect, it is necessary to know how the monitor accesses files.

With each named file, UFD, and MFD, the monitor writes a special block containing necessary information needed to retrieve the data blocks that constitute the file. This block is called a retrieval information block, or RIB.

Retrieval pointers in the RIB describe contiguous blocks of file storage space called groups. Each pointer occupies one word and has one of three forms:

a. A group pointer
b. An EOF pointer
c. A change of unit pointer.

H.1 A GROUP POINTER

A group pointer has three fields:

a. A cluster count
b. A folded checksum
c. A cluster address within a unit. The width of each field may be specified at refresh time; therefore, the same code can handle a wider variety of sizes of devices.

The cluster count determines the number of consecutive clusters that can be described by one pointer. The folded checksum is computed for the first word of the first block of the group. Its main purpose is to catch hardware or software errors when the wrong block is read. The folded checksum is not a check on the hardware parity circuitry. The size of the cluster address field depends on the largest unit size in the file structure and on the cluster size. A cluster address is converted to a logical block address by multiplying by the number of blocks per cluster.

H.1.1 Folded Checksum Algorithm

This algorithm takes the low order n-bit byte, repeatedly adds it to the upper part of the word, and then shifts. The code is:

```
LOOP:   ADD     T1,T
        LDB     T,LOW ORDER N BITS OF T1
        LSH     T1,-N                       ;RIGHT SHIFT BY N BITS
        JUMPN   T1,LOOP
        DONE                                ;ANSWER IN T
```

This scheme elminated the usual overflow problem associated with folded checksums and terminates as soon as there are no more bits to add.

H.2 END-OF-FILE POINTER

The EOF is indicated by a zero word.

H.3 CHANGE OF UNIT POINTER

A file structure may comprise more than one unit; therefore, the retrieval information block must indicate which unit the logical block is on. Because a file can start on one device and move to another, a method of indicating a change from one unit to another in the middle of the file is necessary. To show this movement, a zero count field indicates that the right half of the word specifies a change in unit. A zero count field contains a unit number with respect to the file structure. The first retrieval pointer, with respect to the RIB, always specifies a unit number. Bit 18 is 1 to guarantee that the word is non-zero; otherwise it might be confused with an EOF pointer.

H.4 DEVICE DATA BLOCK

The monitor keeps a copy of up to 10 retrieval pointers in core at once. Therefore, if a file is allocated in 10 or less contiguous blocks (i.e., described in 10 or less pointers), all of the retrieval information can be kept in core and no additional accesses to the RIB are necessary.

H.5 ACCESS BLOCK

For each active file, the monitor keeps eight words of storage called an access block. These access blocks remain dormant in monitor core after a file is closed and are reclaimed only when the core space is necessary. Therefore, if a 4-word LOOKUP is done after a file has been active, access to the UFD and RIB blocks will not require I/O.