DEC-T9-MTZD-D

PDP-10
TIMESHARING MONITORS
PROGRAMMER'S REFERENCE MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Original Printing April 1967
Reprinted July 1967
Revised November 1967
Reprinted March 1968
Revised May 1968
Revised October 1968
Revised September 1969
Revised March 1970
Revised September 1970
Revised October 1970

CONTENTS

APPENDICES

APPENDICES (Cont)

APPENDICES (Cont)

ILLUSTRATIONS

TABLES

TABLES (Cont)

NEW AND CHANGED INFORMATION

Revision of March 1971

This manual has been extensively revised to increase technical accuracy, incorporate new material resulting from the development of the 5.03 release of the monitor, and improve overall presentation of technical information. The location of new or changed technical information is indicated by a black vertical line to the left of the text in which it appears. An example of a flagged passage (taken from page 2-2) is given below.

Each command is a line of ASCII characters in upper/lower case. Spaces and nonprinting characters preceding the command name are ignored. Comments may be typed on the same line as the command by preceding the comment with a semicolon. The Monitor Command Interpreter will not interpret or execute a line of comments.

Foreword

The Timesharing Monitors are described and the commands, program loading procedures, and user programming available under executive control are discussed in this manual. The Timesharing Monitors include the Multiprogramming Monitor (formerly known as 10/40) and the Swapping Monitor (formerly known as 10/50).

SYNOPSIS OF THE TIMESHARING MONITORS MANUAL

Chapter 1, an introduction, contains concepts important to the understanding of the system. Commands to the monitor that may be initiated by a user at a terminal are described in Chapter 2. Several Monitor Support CUSPs (Commonly Used Systems Programs) are also discussed in this chapter. Loading of user programs is explained in Chapter 3. The job data area and the loader are described briefly. The services the monitor performs for the user and how the user's program obtains such services are discussed in Chapter 4. Non-directory I/O devices (e.g., concepts, data modes, special programmed operator services, file status) are discussed in Chapter 5. The two directory devices, DECtape and DISK, are explained in Chapter 6 in the same manner as the devices in Chapter 5 are explained. Algorithms of the monitor, described in Chapter 7, give the user an insight into system operation. Appendices A to J contain supplementary reference material.

USE OF THE TIMESHARING MONITORS MANUAL

The Timesharing Monitors Manual is intended primarily as a reference manual for experienced programmers. The system manager and his programming and operations staffs may find additional information for operating and maintaining the PDP-10 timesharing system in the following publications:

PDP-10 System Manager's Guide (DEC-10-NWZA-D(L))
Five Series Monitor Installation Guide (DEC-10-MRZA-D)

The user interested in timesharing programming from a remote Teletype® should read the PDP-10 Timesharing Handbook, Books 2 and 7, for a detailed explanation of the monitor commands available.

® Teletype is a registered trademark of Teletype Corporation.

CONVENTIONS USED IN THE TIMESHARING MONITORS MANUAL

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 if the file is not on disk, and a project-programmer number, if the file is not in the user's disk area.
arg	A pair of file specifications or a string of pairs of file specifications.
job	A job number assigned by the monitor.
file. ext	Any legal filename and filename extension.
core	Decimal number of 1K blocks of core.
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.
(\$)	The symbol used to indicate an altmode.
↑x	A control character obtained by depressing the CTRL key and then the character key x.
←	A back arrow used in command strings to separate the input and output file specifications.
*	The CUSP response to a command string.
.	The monitor response to a command string.
⮠	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 Interpreter.
___	Underscoring used to indicate computer typeout.
n	A decimal number.

Chapter 1
Introduction

1.1 GENERAL

The PDP-10 Timesharing System allows many independent user programs to share the facilities of a single PDP-10 computer. Many users can access the computer at the same time from consoles at the computer site, at nearby offices or laboratories, or at remote points connected by telephone lines. Operating concurrently under monitor control, users may access available I/O devices and system software to compile, assemble, and execute their programs, or may have this sequence performed automatically for many programs by using the batch control processor (Batch). Real-time programs can operate either as independent user programs or as fully integrated monitor subroutines.

System facilities start with a minimum configuration of 32K of core and can accommodate DECtapes, magnetic tapes, disks, drums, disk packs, communication line controllers, card readers and punches, paper-tape readers and punches, line printers, displays, incremental plotters, and user consoles. Other special devices, including real-time digitizers and analog converters, easily interface with the system. Various peripheral devices and methods of programming are described in Chapters 5 and 6.

1.2 MONITOR FUNCTIONS

The timesharing operating system interfaces between the user and the computer so that all users are protected from one another and appear to have most system resources to themselves. The operating system schedules multiple-user timesharing of the system, relocates and protects user programs in core memory, directs data flow between I/O devices and user programs, and overlaps I/O operations concurrently with computation for high system efficiency.

The timesharing system is a multiprogramming system; that is, it allows several user programs to reside in core simultaneously and to operate sequentially. The timesharing operating system (TOPS) schedules each user program to run for a certain length of time (quantum time), using a scheduling algorithm that makes efficient use of system capabilities (refer to Paragraph 7.1). The switching between programs is initiated by a clock, which interrupts the central processor to signal that the quantum time for the program has elapsed. The interrupt function is provided by the priority interrupt system (refer to the PDP-10 Reference Manual).

To increase the number of users serviced, a secondary memory is employed. This memory, usually magnetic disk or drum, is slower than main memory but provides greatly increased capability. User programs can be located in secondary memory and moved into main memory or core for execution. Programs moved into main memory exchange places with programs that have just been serviced by the central processor. This process is called swapping (refer to Paragraph 7.2).

The asynchronous swapping algorithm is called in at every clock tick and has the task of bringing a user program from secondary memory into core, or vice versa. The central processor may be operating on one user program in one part of memory while another user program is being swapped to or from core. This independent overlapped operation greatly improves efficiency and increases the number of users that can be accommodated simultaneously.

The timesharing operating system is involved in keeping the actions of a user within his assigned memory space. A hardware device, a memory protection register (refer to Chapter 3) set by the monitor, limits the core area that a particular user can access. Any attempt to read or change information outside this area automatically stops the program and notifies the operating system.

1.2.1 Reentrant User Programming

Users of large timesharing systems have varying requirements; therefore, a good system provides a variety of software. Thus, many users may have compilers and other common system programs in core at the same time. To prevent excessive core usage, which results when a program is duplicated for several users, a reentrant user programming capability is employed. This means that a sequence of instructions may be entered by more than one user program at a time.

A reentrant program is written in two parts or segments. One segment contains pure code that is not modified during execution and can be used to simultaneously service any number of users (e.g., the FORTRAN compiler). The second segment belongs to each user and consists of code and data (impure code) that is developed during the compiling process. All versions of the timesharing operating system normally include this reentrant capability, but it may be deleted on systems lacking the dual relocation KT10A hardware option.

In a non-reentrant system, the one-relocation register hardware requires that a user area be a single continuous segment of logical and physical core. Each user has a separate copy of a program although a large part of it is the same as for other users. In a reentrant system, the two-relocation register hardware allows a user area to be divided into two logical segments, which may occupy non-contiguous areas in physical core. The operating system allows one of the segments of each user area to be the same as one or more other users; therefore, only one physical copy of a shared segment need exist no

matter how many users are using it. The operating system normally invokes hardware write-protection for shared segments to guarantee that they are not accidentally modified. User programs may also be written to make use of this protection (refer to Appendix C).

In the PDP-10 system, the reentrant capability causes the following system resources to be used more efficiently:

a. Core memory. Only one copy of a shared segment exists for the entire system. More programs can fit into a given amount of core. (Figure 1-1 illustrates this efficient use of core memory.)

b. Swapping storage. Many users share the single copy of the shared segment kept in swapping storage.

c. Swapping I/O channel. A shared segment is read into core only once and is not written back onto swapping storage unless modified.

d. File storage I/O channel. A shared segment exists on the faster swapping storage after it has been read into core the first time from the storage device, instead of being retrieved from file storage on each usage as necessary in a non-reentrant system.

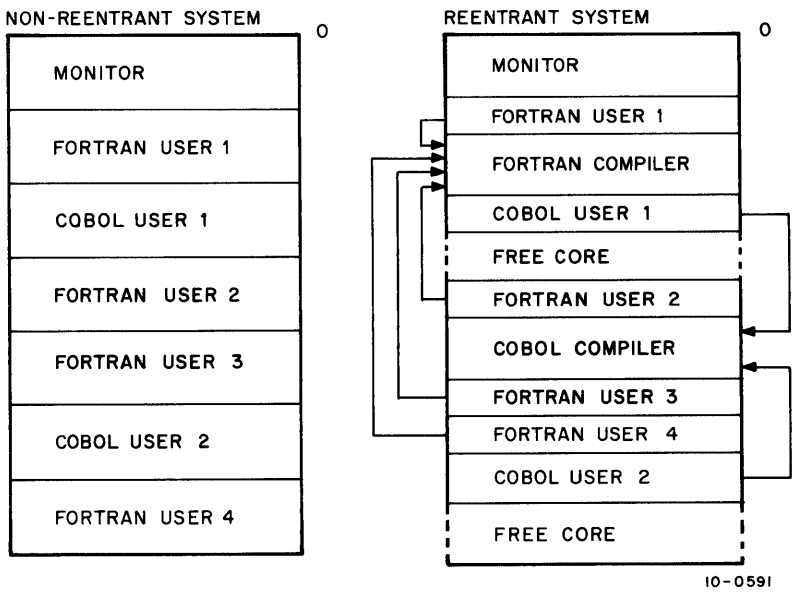


Figure 1-1 Core Management

1.3 USER FACILITIES

The basic function of the timesharing system is to allow a number of users simultaneous access to the central computer. To be fully useful, however, the system should also allow the users access to other system resources, such as storage devices for the user's programs and data; therefore, the operating system includes I/O control of all devices attached to the system, run-time selection of I/O devices, job-to-job transition, and job save and restore features.

Users gain access to the PDP-10 timesharing system from a terminal located either at the computer facility or at a remote site connected by telephone. Three levels of communication available at the console are:

a. monitor command level
b. CUSP command level
c. CUSP I/O level.

At monitor command level, the console communicates with the Monitor Command Interpreter. The Monitor Command Interpreter

a. provides the system with access protection
b. allocates and protects memory and peripherals requested by the user
c. provides communication with the operator for the mounting of special tapes and disk packs
d. provides run control for the user over programs stored in the system
e. allows the user to initiate background jobs
f. provides the user with job monitoring and debugging facilities
g. returns facilities to the system when the job is finished using them.

Various monitor commands providing each of these capabilities are described in Chapter 2.

Using monitor commands, the user at his console can call in programs from the file system. The file system contains programs for creating and editing program source files (TECO, LINED), for assembling or compiling program source files (MACRO, AID, FORTRAN, BASIC, COBOL), and for loading relocatable binary files (LOADER). The use of these and many other CUSPs are described in the PDP-10 Reference Handbook and the PDP-10 Timesharing Handbook.

The user's console provides both a control and data path to any CUSP or other user program that the user initiates via monitor commands. When a particular CUSP is called in, the user's console is at CUSP command level and the user can issue a command to the CUSP. In processing that command, the CUSP may access the user's console directly as an input or output device. This is illustrated by the following example.

.R PIP	Monitor command level. User calls CUSP named Peripheral Interchange Program (PIP).
*DSK:TEXT←TTY:	CUSP command level. User instructs PIP to create a file on the disk named TEST using Teletype console as input medium.
THIS IS FILE TEXT ↑Z	CUSP I/O level. User types input to PIP. ↑Z causes Teletype end of file. Return to CUSP command level.

(continued on next page)

*↑C	↑C is a special character that causes return to monitor command level.
.	The period (.) signifies return to monitor command level.

The console is switched back to the Monitor Command Interpreter by either the program or the user. The user can exercise another dimension of control over his program by loading it with the powerful Dynamic Debugging Technique (DDT) available in the system file. Entry to DDT is through the Monitor Command Interpreter or by breakpoints in the program. While DDT is in control of the program, the user can examine intermediate results on his console and then modify his program accordingly.

The user's program communicates with the monitor by the PDP-10 operation codes 040 through 077. These op-codes, called UUOs, are described in detail in Chapter 4. With these operation codes, the monitor provides the program with complete device-independent I/O services. The programmer is relieved of I/O programming and is freed from the dependence on the availability of particular devices at run time. In addition, the user's program may exercise control over central processor trapping, modify its memory allocation, and monitor its own running time. Provisions exist for inter-job communication and control, reentrant user programs, and, in selected cases, direct user I/O control.

1.4 SEGMENTS

A segment is a continuous region of the user's core area that the monitor maintains as a continuous unit in physical core/possibly fragmented unit on the swapping device. A program or user job is composed of one or two segments. A segment may contain instructions/data. The monitor determines the allocation and movement of segments in core and on the swapping device.

A sharable segment is the same segment for many users. The monitor keeps only one copy in core/on the swapping device, no matter how many users are using it. A non-sharable segment is different for each user in core/on the swapping device.

The two PDP-10 relocation and protection registers, which divide a user's core area into two parts, permit a user program to be composed of one or two segments at any time. The required low segment starts at user location 0. The optional high segment starts at user location 400000 or at the end of the low segment, whichever address is greater. The low segment contains the user's accumulators, job data area, instructions/data, I/O buffers, and DDT symbols. A user's core image is composed of a low segment, which may have from 1K to 256K words, in multiples of 1K ($1K=1024_{10}$ words), and a high segment, which may have from 0K to 128K words, also in multiples of 1K. A high segment may be sharable or nonsharable, whereas a low segment is always nonsharable. The high segment is usually write-protected, although the program can turn off write-protection and modify itself (refer to Paragraph 4.5.4).

A reentrant program is always composed of two segments: a low segment, which usually contains data, and a high (sharable) segment, which usually contains instructions and constants. The low segment is sometimes referred to as the impure segment. The sharable high segment, if write-protected, is referred to as the pure segment.

A one-segment non-reentrant program is composed of a single low-segment containing instructions and data. User programs written for machines with only a single relocation and protection register are always one-segment non-reentrant programs.

A two-segment non-reentrant program is composed of a low segment and a nonsharable high segment. This program is useful when there is a requirement for two fixed-origin data areas to increase and decrease independently during execution.

1.5 FILE STRUCTURES

A file structure is the logical arrangement of 128-word blocks on one or more units of the same type to form a two-level hierarchy of named files. File structures allow a user to specify which unit he wishes to use for his files. System reliability is increased because a file structure may be removed from the system without affecting other units.

A complete disk system is composed of one or more units of the same/different types of disks and, therefore, consists of one or more file structures. All information in the system (programs and data) is stored as named files in a uniform and consistent fashion. A file structure can exist on exactly one unit but is usually distributed over a number of physical units of the same type; however, two file structures cannot exist on the same unit. Each file structure is logically complete and is the smallest removable unit of file memory. All pointers to blocks within a file structure are by way of logical block numbers rather than physical disk addresses. There are no pointers to blocks in other file structures. This property allows a file structure to be removed from a disk system without disturbing any of the units in other file structures.

1.5.1 File Directories

Each file structure has two levels of directories: a Master File Directory (MFD) and User File Directories (UFD). The entries in the MFD are the names of the User File Directories. The entries within the UFDs are the names of files existing in a given project-programmer number area within the file structure (refer to Paragraph 6.2.2.3). Figure 1-2 shows the relationship of the directories.

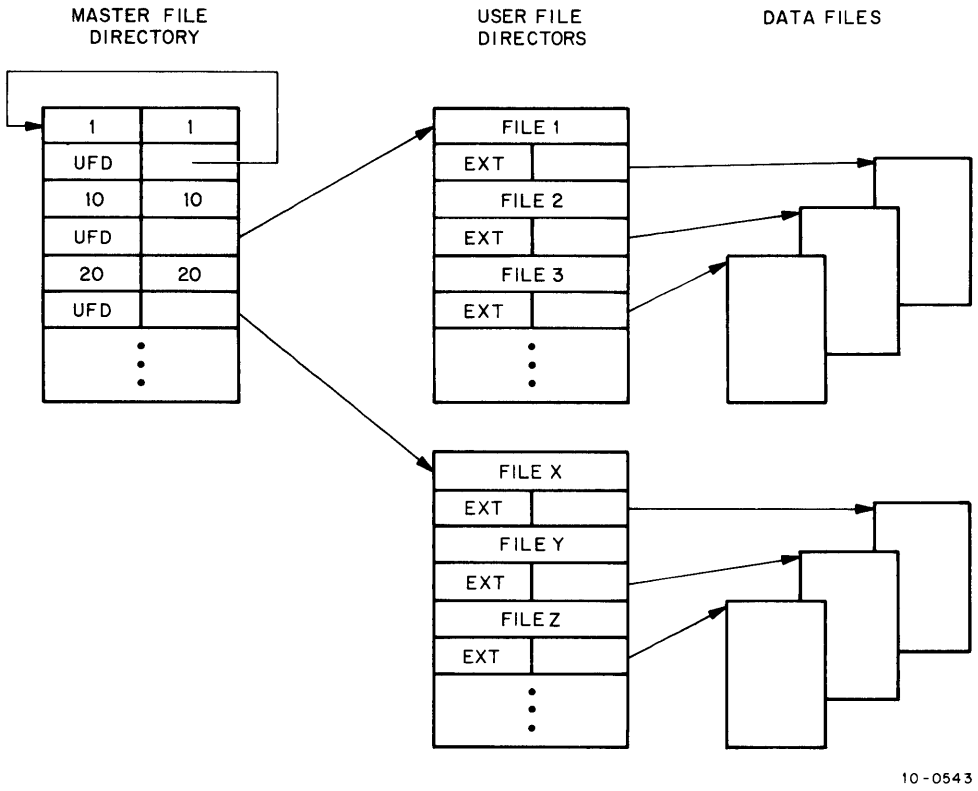


Figure 1-2 File Structure Directories

1.5.2 Quotas

Each project-programmer number in each file structure is associated with two quotas that limit the number of blocks that can be stored in the UFD in the particular file structure. The two quotas are: logged-in quota and logged-out quota. The logged-in quota is not a guaranteed amount of space, and the user competes with other users for space. The logged-out quota is the amount of space that the user must be within in order to log off the system. Quotas are used by system administrators to ration resources in a predetermined manner.

1.5.3 Files

A file is a collection of 36-bit words comprising computer instructions/data. A file can be of arbitrary length, limited only by the available space on the device and the user's maximum space allotment on that device. A disk file is limited by the smallest available space on a file structure and by the user's quota on that file structure.

A named file is uniquely identified in the system by its filename (up to six characters in length) and extension (up to three characters in length) and by its directory name (owner's project-programmer number) and file structure name for disk or physical device name for DECtape, in which the filename

and extension appear. The filename, being arbitrary, is specified by the owner, whereas the extension, usually one of a small number of standard names that identify the type of information in the file, is usually specified by the program (refer to Appendix G). A named file may be written by a user program in buffered or unbuffered mode, or in both. It may be read/modified sequentially or randomly with buffered or unbuffered mode I/O independently of how it was written. Named files are uniformly stored. Each named file has a certain access protection associated with it. These protections designate which users can read or write the file or change its access protections. For a given file, users are divided into three groups: the owner of the file, the users in his project, and the rest of the users (refer to Paragraph 6.2.3).

A file is created if no file by the same name existed in the file structure when the file was opened for writing. A file is superseded if another file by the same name exists. A file is updated when one or more blocks of the file are rewritten in place. Other users may read a disk file while a certain user is superseding it. The older version of the file is deleted only when all the readers have finished with it. Only one user may open a file for updating at a time; all other users attempting to open that file receive an error message.

1.5.3.1 Comparison of Files and Segments - Files and segments have certain similarities and differences. Both are named, one-dimensional arrays of 36-bit words. A named file can be as long as the size of DECtape or the sum of the available space on a file structure. A segment can be only as big as physical core. Both may be shared for reading, but only one user may supersede or update a file at a time, whereas many users may share a segment for writing. When many users share the same file, each user is given his own copy of the portion of the file that he is reading. It is read into his low segment by the INPUT UUO. When many users share the same segment, each user does not have his own copy of the segment. A file exists on the file structure and portions of it may exist in different parts of the low segment of one or more users. A segment never exists on the storage device; a segment exists as a continuous unit only in core or on the swapping device.

Chapter 2
Monitor Commands

2.1 CONSOLE AND JOB CONTROL

The PDP-10 timesharing system is a multiprogramming system. This means that control is transferred rapidly among a number of programs or processes in such a way that all the processes appear to be running simultaneously. Each process is called a job. The term job refers to the entire sequence of operations the user initiates from his console. In configuring and loading a timesharing monitor, the system administrator sets the maximum number of jobs that his system can handle simultaneously. This number may be up to 127 jobs if the system has enough core, disk storage, processor capacity, and timesharing consoles to handle this load.

Jobs are initiated by users typing on a timesharing console. A console is typically any of several models of Teletype machines but may also be a cathode ray tube (CRT) with a keyboard or any of the other interactive terminals available. The console may be directly connected to the computer, or may be remotely connected via a private wire or the public telephone system.

There is not necessarily a one-to-one relationship between jobs and consoles. A console must initiate a job, but the DETACH and ATTACH commands (refer to Paragraph 2.8) permit a job to float in a state where it is not associated with a particular console; therefore, a user may control several jobs from the same console. Each job is either in the ATTACHed or DETACHed mode depending on whether a console is currently associated with that job. At any time, each console is attached to at most one job. The console is often referred to as being in a detached mode, but this results from a semantic confusion. It is really meant that the job initiated from that console is in a detached mode. By typing an appropriate command, the job may be attached by the same console or by any other console in the system.

2.1.1 Monitor and User Mode

From the user's point of view, his console is either in monitor or user mode. In monitor mode, each line the user types in is sent to the Monitor Command Interpreter. The execution of certain commands (as noted in the examples below) places the console in user mode. When the program is in user mode,

the console becomes simply an I/O device for that user. In addition, user programs use the console for two purposes. The user program will accept command strings from the console or will use the console as a direct I/O device.

Example:

monitor mode	.R PIP *DSK:FILE1←TTY: THIS IS FILE 1 ↑Z	monitor command
user mode		user program command string
user mode		user program using console as an input device
monitor mode	*↑C .R MACRO	monitor command
user mode	*TTY:←DSK:PROG1	user program command string
user mode	assembly listing	user program using console as an output device

The special character ↑C (produced by typing C with the CONTROL key depressed) is used to stop a user program and return the console to monitor mode. If the user program is in a Teletype input wait state, one ↑C must be issued from the user's console; otherwise, two ↑C's must be issued. Because of this procedure, the user knows if his program is waiting for input if there is no response from the monitor after one ↑C. Certain commands cause the user program to start or continue running (as noted in the tables below) but leave the console in monitor mode.

When the system is started, each console is in monitor mode ready for users to begin typing in commands. However, if the system becomes fully loaded (i.e., the maximum number of jobs that the system has been set to handle has been initiated), then any unused consoles enter a special state where any command typed in will receive either the message JOB CAPACITY EXCEEDED or X.

2.2 COMMAND INTERPRETER AND COMMAND FORMAT

Each command is a line of ASCII characters in upper/lower case. Spaces and nonprinting characters preceding the command name are ignored. Comments may be typed on the same line as the command by preceding the comment with a semicolon. The Monitor Command Interpreter will not interpret or execute a line of comments. Every command to the Monitor Command Interpreter must be terminated by pressing the RETURN key on the console. If the command is not understood, the command up to the error is typed out by the monitor preceded and followed by a ?, and the mode is unchanged.

ANSWER TO SOFTWARE TROUBLE REPORT #10-1798

Diagnosis: An assigned teletype should not be allowed to type commands at all. There is a difference between ASSIGNED and CONTROLLING teletypes.

Solution: The fact that under SCNSRF in 4.72, an ASSIGNED teletype does partially succeed in starting the command processor is a bug that has been fixed in the completely re-written SCNSER for the 5.Ø2 release. After SCNSER reached the filed, SCNSRF will be obsolete, therefore we are no longer fixing bugs in it.

DOC. CORR:
Document Title: Timesharing Monitor
DEC Number - DEC-T9-MTZB-D
Chapter 2
Page 2-2

At end of Section 2.1: Add the following:

It is also possible for a user to ASSIGN additional consoles to his job, if they are not already in use. At any one time, however, only one TTY is actually ATTACHED to the job (i.e., controlling it). This controlling TTY is automatically ASSIGNED to the job at the time the job is initiated. The controlling TTY cannot be DEASSIGNED from the job.

Monitor commands may be typed from controlling TTY's, but not from ASSIGNED TTY's.

Also: At end of DEASSIGN description, Table 2-2, add:

If DEV is a controlling TTY, the logical device name is discarded, but the TTY is not actually DEASSIGNED from the job.

2.2.1 Command Names

Command names are strings from one to six letters. Characters after the sixth are ignored. Only enough characters to uniquely identify the command need be typed. In the commands which follow, the commonly used abbreviation of the command name is shown in parentheses. Installations choosing to implement additional commands should take care to preserve the uniqueness of the first three letters of existing commands.

2.2.2 Arguments

Arguments follow the command name, separated from it by a space or any printing character that is not a letter or a numeral. Argument formats are described under the associated commands.

If the Monitor Command Interpreter recognizes the command name, but a necessary argument is missing, the monitor responds with

```
TOO FEW ARGUMENTS
```

Extra arguments are ignored.

2.2.3 Log-In Check (Disk Monitor Systems)

If a user who has not logged in (refer to Paragraph 2.3) types a command requiring him to be logged in, the disk monitor systems will respond with

```
?LOGIN PLEASE
```

and the user's command will not be executed. Log-in is not required by a nondisk monitor system.

2.2.4 Job Number Check (Nondisk Monitor Systems)

If the nondisk monitor system recognizes a command name, which requires a job number, and no job number has been assigned, the monitor assigns a job number, n, and responds with

```
JOB n
```

and a line identifying the monitor version. The monitor then proceeds to execute the command.

2.2.5 Core Storage Check

If the Monitor Interpreter recognizes a command name, which requires core storage to be allocated to the job, and the job has no core, the monitor responds with

?NO CORE ASSIGNED

The user's command is not executed.

2.2.6 Delayed Command Execution

If the Monitor Command Interpreter recognizes a command that requires all devices to be inactive, and the job has devices actively transmitting data to or from its core area, the execution of the command is delayed until the devices are inactive. A command is also delayed if a job is swapped out to the disk and the command requires core residence. It will be executed when the job is swapped into core.

2.2.7 Completion-of-Command Signal

Most commands are processed without delay. The completion of each command is signaled by the output of a carriage return, line feed. If the console is left in monitor mode, a period follows the carriage return, line feed. If the console is left in user mode, any response other than the carriage return, line feed comes from the user's program. For example, all standard DEC CUSPs (except DDT) immediately send an asterisk (*) to the user's console to indicate their readiness to accept user-mode command strings. (DDT sends another carriage return, line feed.)

2.3 JOB INITIALIZATION COMMANDS

Access to the system is limited to authorized personnel. The system administrator provides each authorized user with a project number, a programmer number, and a password. The project numbers range from 1 to 377777 octal and the programmer numbers range from 1 to 377777 octal.

The project-programmer numbers identify the user and his file storage area on a file structure. The password is from one to six SIXBIT characters. To log-in successfully, the project-programmer numbers and the password typed in by the user must match the project-programmer numbers and password stored in the system accounting file (SYS:ACCT.SYS).

DOCUMENTATION CORRECTION

MANUAL: Timesharing Monitors Programmer's Reference Manual

DEC CODE: DEC-T9-MTZD-D

LOCATION OF ERROR: Chapter 2, Section 2.2.7, Page 2-4

CORRECTION OF ERROR:

1) Change (except DDT)

   to (except DDT and BASIC)

2) Change carriage return, linefeed.)

   to carriage return, linefeed, and BASIC sends the message READY.)

LOGIN (LOG)

Function

The LOGIN command is used to gain access to the system. This command loads a Monitor Support CUSP which accepts the user's LOGIN data. The user types in his project and programmer numbers followed by his password.

Command Format

LOGIN ppn

ppn = the user's project-programmer number. This argument may be typed on the same line as the LOGIN command, or on the following line after LOGIN types out the number sign.

Characteristics

The LOGIN command:

leaves the console in monitor mode or
starts a program running if specified in ACCT.SYS. entry for ppn.
runs the LOGIN CUSP,
is used with disk monitors only.

Associated Messages

?INVALID ENTRY - TRY AGAIN

An illegal project-programmer number was entered, or the password did not match.

?1+1/nk CORE
VIR. CORE LEFT =0

The swapping space or core allocated to timesharing is all in use (i.e., there is no available virtual core).

?JOB CAPACITY EXCEEDED

This message is received by the first user who attempts to LOGIN after the maximum number of jobs that the system has been set to handle have been initiated.

(continued on next page)

X

If the system is fully loaded, any user (after the first) who attempts to LOGIN receives this character in response to any character typed.

?SYSTEM NOT AVAILABLE

The operator has used the SET SCHEDULE command to prevent LOGINS from all consoles. The message of the day is still typed.

?NO REMOTE USERS. TRY AGAIN LATER

The operator has used the SET SCHEDULE command to prevent LOGINs from remote consoles. The message of the day is still printed.

?PROJECT 1 MAY NOT BE REMOTE OR PTY

Project 1 is never allowed to LOGIN at a remote teletype or over a pseudo-teletype, if using old scanner service.

?SOME OTHER TIME

User is not scheduled to LOGIN at this time. He should try again when he is allowed to log in.

WAIT PLS

FACT.SYS was busy for ten seconds. LOGIN retries for ten more seconds before trying FACT.X01. This message can appear if many people are logging in simultaneously.

?UFD OUTPUT FAILURE n

The output failed when trying to create UFD (level C); n is the software channel status.

?UFD ENTER FAILURE n

Failure in trying to create UFD; n is the ENTER error code.

?NO ENTRY IN AUXACC.SYS
NO SEARCH LIST OR UFDS CREATED

If user has no entry in AUXACC.SYS, LOGIN does not create UFDs or a search list. User is logged in and has UFDs if they existed previously. He may write on file structures which have UFDs or read file structures. He may also create a file structure search list with SETSRC (level D).

(continued on next page)

ACCOUNTING SYSTEM FAILURE . . .
CALL THE OPERATOR

LOGIN could not append an entry to the accounting file.

UFD LOOKUP FAILURE n

A failure occurred in setting up UFDs (level D); n is the LOOKUP error code.

UFD RENAME FAILURE n

A failure occurred in setting up UFDs (level D); n is the RENAME error code.

?UFD INTERLOCK BUSY

Could not get UFD interlock when trying to set up UFD. UFD is not currently set up (level D).

?CANT OPEN <file structure name >

The file structure is mounted but cannot be opened. No UFD is created, though one may already exist.

?CANT ADD TO YOUR FILE STRUCTURE SEARCH LIST n

n is the error code from STRUUO when trying to add a file structure to search list.

?MAY NOT LOGIN IN AS MFD PPN

If MFD PPN is not the same as SYS PPN, no one may log in as the MFD PPN, even though there may be an entry in ACCT.SYS.

?PPN HAS EXPIRED

The current date is greater than the expiration date of the PPN. User may not log in until expiration date is changed.

?ONLY BATCH USERS MAY LOGIN. TRY AGAIN LATER.

The operator has used the SET SCHEDULE command to prevent LOGINs, except for BATCH jobs. The message of the day is still printed.

(continued on next page)

?MAY NOT LOGIN { LOCAL / REMOTE / DATA SET / BATCH JOB SUBJOB }

ACCT.SYS entry does not permit the project-programmer number to login at the teletype that is being used.

?CANT ACCESS SYSTEM FILES

ACCT.SYS could not be read. No one may LOGIN until ACCT.SYS is ready. Consult the operator.

?WRONG FORMAT VERSION NUMBER IN SYSTEM FILES

Wrong version of ACCT.SYS or AUXACC.SYS is on SYS. Consult the operator.

<file structure name> FILE ERRORS EXIST

One of the files in a file structure has an error status, as flagged in the UFD of that file structure.

Example

The following is the procedure used to gain access to the system.

.LOGIN ↵ JOB 7 PDP-10 5S.01 TTY23	LOGIN types the job number assigned to user (job number 7), followed by monitor name, version number, and console line number.
#	LOGIN types out number sign to indicate user should type his project-programmer number. This happens only if the user did not type his project-programmer number on the same line as the LOGIN command.
34,570 ↵	User types in his project-programmer number. The user may separate the project-programmer numbers with a slash causing LOGIN not to type the message of the day. Normally, this should not be done since the message frequently contains important operational information.
PASSWORD:	System requests user to type his password. User types password followed by carriage return. To maintain password security, the monitor does dot echo the password. On half-duplex circuits (refer to Paragraph 5.9), a mask is typed to make the password unreadable.
1135 8-AUG-70 TYPE SYS:SCHED FOR NEXT WEEKS SCHEDULE .	If user entries are correct, responds with time, date, a message of the day (if any), and a period, indicating readiness to accept a command.

INITIAL (INI)

Function

The INITIAL command performs standard system initialization for the terminal issuing the command. This command is issued automatically at system startup and 143 restart at certain designated terminals, but may be re-issued at any time by the user. This command is used to initiate specific CUSPs, such as the line printer spooler CUSP, PRINTR, on a particular console.

The INITIAL command runs SYS:INITIA.SAV which, depending upon the system configuration and the TTY number from which it is typed, may cause any of a number of events to occur.

Command Format

INITIA

Characteristics

The INITIAL command:

leaves the console in monitor mode,
runs a CUSP.

Examples

```
.INITIAL
5S0111A SYS #2 22:12:17 TTY24
.
EXIT

.
```

2.4 FACILITY ALLOCATION COMMANDS

The monitor allocates peripheral devices, file structure storage, and core memory to users on request and protects these allocated facilities from interference by other users. The monitor maintains a pool of available facilities from which a user can draw.

A user should never abandon a timesharing console without returning his allocated facilities to the monitor pool. Until a user returns his allocated facilities to the pool no other users may utilize them except through operator intervention.

All devices controllable by the system are listed in Tables 5-1 and 6-1. Associated with each device is a physical device name, consisting of three letters and zero to three numerals to specify unit number. A logical device name may also be assigned to a physical device by the user. The logical name of one to six alphanumeric characters of the user's choice is used synonymously with a physical device name in all references to the device. In writing a program, the user may use arbitrarily selected device names which he assigns to the most convenient physical devices at runtime. All references to devices in the monitor pool are made either by physical names or by assigned logical names. However, the preferred method for user programs to obtain device names is through a command string.

When a nonsharable device is assigned to a job, it is removed from the pool of available facilities of the monitor. Any attempt by another job to reference the device fails. The device is returned to the pool when the user reassigns it or kills his job.

ASSIGN (AS)

Function

The ASSIGN command assigns an I/O device to the user's job for the duration of the job or until a DEASSIGN command is given. This command, applied to DECtapes, clears the copy of the directory currently in core, forcing any directory references to read a new copy from the tape. (Refer to Paragraph 6.1.7 for further details.)

Although DECtape is the only device that must be ASSIGNed before use, to ensure that the monitor has a copy of the proper DECtape directory in core, it is wise to ASSIGN all devices, such as magnetic tape, before use.

Command Format

ASSIGN phys-dev log-dev

phys-dev = any device listed in Tables 5-1 and 6-1, or any file structure. This argument is required.

(continued on next page)

log-dev = a logical name assigned by the user. This argument is optional. Except for disk devices, only one logical name can be assigned to a physical device. Subsequent ASSIGN commands to all devices except disk devices replace the old logical name with the new one. Logical names are disassociated from disk devices by the DEASSIGN command.

NOTE

If DTA, MTA, or LPT is used with no numeric argument, the monitor seraches for an available unit and then types DTAn, MTAn, or LPTn ASSIGNED.

Characteristics

The ASSIGN command:

leaves the console in monitor mode.

Restrictions

A comma may not be used to separate the logical and physical device names. If a comma is used, the monitor terminates its scan at the comma; therefore, the logical name is not assigned.

Associated Messages

dev: ASSIGNED

The device has been successfully assigned to the job.

?NO SUCH DEVICE

The device name does not exist, or all devices with this name are in use.

?ALREADY ASSIGNED TO JOB n

The device has already been assigned to another user's job.

?LOGICAL NAME ALREADY IN USE, dev: ASSIGNED

The user attempted to assign a previously-used logical name to this device.

Examples of logical and physical device names

.ASSIGN DTA ABC ↵	User requests a DECtape drive.
DEVICE DTA6 ASSIGNED	Monitor has given the user drive DTA6. The user mounts a DECtape on drive DTA6.
.ASSIGN PTP ABC ↵	User requests the paper tape punch.
LOGICAL NAME ALREADY IN USE, PTP ASSIGNED	Paper tape punch is reserved, but ABC still refers to DTA6 only.
.R PIP ↵	User requests the system program PIP.

(continued on next page)

*PTP:←ABC:FILEX↵	User issues a command string to PIP asking that the FILEX be transferred from device ABC (DTA6) to device PTP (assigned to the user).
*↑C	User returns to monitor mode.
.ASSIGN DTA DEF↵	User requests another DECtape drive.
NO SUCH UNIT	All drives are in use. No DECtape drive is assigned, and no logical assignment is made.
.ASSIGN DTA6 DEF↵	User requests drive DTA6 (which he already has). The logical device name ABC is no longer associated with DTA6. The copy of the directory currently in core is cleared.
DEVICE DTA6 ASSIGNED	User mounts a new DECtape on the previously assigned drive. The new DECtape directory is read into core.
.R PIP↵	User requests PIP.
*PTP:←ABC:FILEY↵	User requests that FILEY be transferred from device ABC to device PIP.
?DEVICE ABC DOES NOT EXIST	The logical device name ABC is no longer assigned.
*PTP:←DEF:FILEY↵	User reissues the command string to PIP asking that FILEY be transferred from device DEF to device PIP.
*↑C	User returns to monitor mode.
.ASSIGN DTA6 DEF↵	User requests drive DTA6 again. The old directory is cleared from core.
DEVICE DTA6 ASSIGNED	User mounts a new DECtape. The new directory is read into core. The same logical name is acceptable because the ASSIGN applies to the same unit (DTA6).

Examples of ASSIGN command

.ASSIGN DTA↵	The user assigns any available DECtape drive.
.ASSIGN MTA2↵	The user assigns a specific magnetic drive (drive 2).
.ASSIGN DTA3 INP↵	The user assigns DECtape drive 3 and gives it logical name INP.
.ASSIGN MTA FAILSA↵	The user assigns any available magnetic tape drive and gives it logical name FAILSA.
.ASSIGN DSK PTR↵	The user assigns generic name DSK and gives it logical name PTR.

DEASSIGN (DEA)

Function

The DEASSIGN command returns one or more devices currently assigned to the user's job back to the monitor pool of available devices. The command, applied to DECtapes, clears the copy of the directory currently in core, forcing any directory references to read a new copy from the tape. (Refer to Paragraph 6.1.7 for further details.)

Command Format

DEASSIGN dev

dev = either the logical or physical device name. This argument is optional. If it is not specified, all devices assigned to the user's job, except the controlling Teletype, are deassigned. The logical name of the controlling Teletype is cleared whether or not an argument is specified.

Characteristics

The DEASSIGN command:

leaves the console in monitor mode,
requires LOGIN and a job number.

Associated Messages

?NO SUCH DEVICE

The device name does not exist in this monitor configuration.

?dev WASN'T ASSIGNED

The device is not currently assigned to this job.

Examples

.DEASSIGN LPT↵	The line printer is returned to the monitor's pool of available resources.
.DEASSIGN↵	All devices assigned to the job are returned.

REASSIGN (REA)

Function

The REASSIGN command allows one job to pass a device to a second job without having the device go through the monitor device pool. This command clears the copy of the directory currently in core, but does not clear the logical name assignment.

Command Format

REASSIGN dev job

dev = the physical or logical name of the device to be reassigned. This argument is required.

job = the number of the job to which the device is to be reassigned. This argument is required.

Characteristics

The REASSIGN command:

leaves the console in monitor mode,
requires core,
does not allow an active device.

Restrictions

Console (controlling) Teletype cannot be reassigned.

Associated Messages

?dev WASN'T ASSIGNED

The device is not currently assigned to the user's job.

?NOT A JOB

The job number specified has not been initialized.

?NO SUCH DEVICE

The device does not exist in this monitor configuration.

(continued on next page)

?dev CAN'T BE REASSIGNED

(1) The console (controlling) Teletype cannot be reassigned, or (2) the logical name is duplicated, or (3) the logical name is a physical device name in the system.

Examples

.REASSIGN LPT 17↵	Reassign the line printer to job 17.
.REASSIGN CDP 4↵	Reassign the card punch to job 4.

MOUNT (MOU)

Function

The MOUNT command allows the user to gain access to a file structure. This command notifies the operator to mount packs (if necessary), allows the user to specify specific drives, places the file structure name at the end of the jobs search list, and waits for completion of operator action (if desired).

The MOUNT command runs the UMOUNT CUSP in the user's core area. UMOUNT scans the command string and does as much as it can without operator intervention. UMOUNT can always complete the action requested by the MOUNT command if the file structure is already mounted and ready. If operator intervention is required, UMOUNT queues a request to the OMOUNT CUSP by writing a command file on 3, 3 disk area. OMOUNT reads these command files and interacts with the operator. When the command file is deleted, the operator action has been completed.

Command Format

MOUNT dev: switches <drives>

dev: = the file structure name as recorded in STRLST.SYS. This argument is required.

switches =	/HELP	(type this list)
	/WENABL	(write enable, default condition, complement of /WLOCK)
	/WLOCK	(write locked)
	/RONLY	(read only, same as WLOCK)
	/MULTI	(multi-access, default condition, complement of /SINGLE)
	/SINGLE	(single access)
	/CHECK	(check and list pending requests)
	/LIST	(list physical drive names and file structure status)
	/SYSTEM	(add file structure to SYS search list)

The switches are optional and only enough characters to make the switch unique are required.

(continued on next page)

<drives> = the physical drives on which the units are to be mounted. The drives must be in the logical unit order within the file structure. Drive names are separated by commas. Leading and embedded drives that are not specified must be represented by null names (,,DPA3). Unspecified trailing drives may be omitted. Drive names are as follows:

Blank, null - unspecified. UMOUNT finds one of proper type.

Two letters - controller class. This may not be useful since file structure units are bound to one controller class.

Three letters - specific controller. UMOUNT finds a drive.

Three letters and one or two digits - specific drive.

The user, by specifying a drive list, may force the packs to be mounted on specific drives or controllers. If no drive (or incomplete) specification is given, an available drive of the proper type is found, even if a dormant file structure must be removed.

Characteristics

The MOUNT command:

places the console in user mode,
runs the UMOUNT CUSP,
is used with disk monitors only.

Associated Messages

If a special condition is encountered, a descriptive comment is typed to the user, and the command is continued. If the condition cannot allow the command to be continued, an error message preceded by a question mark is typed, and the command is aborted. (See Appendix F for these messages.)

WAITING...

A request has been queued to the operator, and the command is waiting for completion of the request. If the user does not want to wait for the completion of the operator's action, he may type control-C without aborting the command.

STRUCTURE ALREADY MOUNTED

The specified file structure is already mounted. However, it may not be in a readied condition.

OTHER USERS - CANNOT SINGLE ACCESS

The /S switch has been typed, and there are currently other users of the file structure. The switch is ignored.

UNIT id ALREADY MOUNTED ON DRIVE DPAn

The file structure is already mounted on a different drive than specified by the user.

(continued on next page)

MCO #D-604 (RLK) DOCUMENTATION CHANGE

COMMOD PP:61
94, 101

SYMPTOM: OMOUNT/UMOUNT CAN'T ALLOW "ONLY USER" OF F/S TO CHANGE STATUS (E,G, WRITE PROTECTION) BECAUSE STRUUO WON'T ALLOW IT UNLESS USER IS ALSO SINGLE-ACCESS.

DIAGNOSIS: STRUUO CAN'T TELL IF JOB IS "ONLY USER" UNLESS IT IS SINGLE-ACCESS (JOB # IN STRJOB).

CURE: CHANGE MEANING OF STRJOB AS FOLLOWS:
= XWD 0,0 IF 0 OR MORE THAN 1 JOB HAVE F,S MOUNTED (SINGLE ACCESS OR NOT).
= XWD -1, N IF JOB N IS ONLY JOB WITH F,S, MOUNTED (SINGLE ACCESS OR NOT).
= XWD 0,N IF JOB N HAS IT MOUNTED SINGLE-ACCESS.

CHANGE ACTION OF STRUUO AS FOLLOWS:
WRITE-PROTECTION (UNPAWL IN UNIDES) AND SINGLE-ACCESS (UNPSAF IN UNIDES) ARE ALLOWED TO BE CHANGED IF F, S, MOUNTED BY 0 OR ONLY 1 JOB (SINGLE-ACCESS OR NOT).

NOTE: THE MONITOR (STRUUO) DECIDES IF THESE BITS CAN BE CHANGED BUT LEAVES THE DECISION AS TO WHO MAY CHANGE THEM TO THE NON-RESIDENT MODULE (CUSP) OMOUNT.

MCO #D-606 DOCUMENTATION CHANGE

FILSER PP: 99, 103, 108

SYMPTOM: NO WAY TO ADD/REMOVE F,S,'S TO/FROM SYSTEM SEARCH LIST.

CURE: REMOVE: IF F,S, IS IN SYS,S,L,--DELETE FROM SYS,S.L, ALSO ADD WARNING MESSAGE IN OMOUNT.
ADD EXTRA ARGUMENT (SPECIAL ACTION BITS) AND DEFINE BIT 0 = 1 TO MEAN "ALSO ADD TO SYS.S,L." ADD SWITCH "/SYSTEM" TO MOUNT COMMAND (OMOUNT ONLY) TO CAUSE F,S, TO BE ADDED TO SYS,S,L.

MCO #D-596 (TH) DOCUMENTATION CHANGE LEVEL C BUG TOO

COMMOD 2

SYMPTOM: DISK PACK ERROR RECOVERY, DOES NOT FOLLOW MEMOREX SPECIFICATIONS.

CURE: CHANGE DSKTRY FROM 3 TO 10 SO DATA ERRORS WILL BE TRIED 10 TIMES INSTEAD OF 3 BEFORE BEING CALLED A HARD DATA ERROR.

MCO #D-598 (TH) DOCUMENTATION CHANGE

COMMOD P27 WRTBAT
COMMOD P2?

SYMPTOM: FIELD SERVICE NEEDS MORE HARDWARE INFORMATION ON DISK ERRORS,

CURE: ADD CONI BITS 12 THROUGH 29 FOR RC-10 AND RP-10 IN LH OF 2ND WORD OF BAT BLOCK, THE LH IS CURRENTLY UNUSED SINCE ONLY NEED 16 BITS TO STORE LOGICAL BLOCK OF THE BIGGEST UNIT DIGITAL MAKES SO FAR, DSKLST WILL TYPE THIS STATUS OUT ON /B SWITCH.

DPAn NOT READY

The specified drive is off-line or write-locked when write-enable is requested. The operator is notified.

NEW UFD CREATED ON STRUCTURE
RESERVED (n) F.C.F.S (n) LOGGED-OUT (n)

An initial UFD has been created for the user. The numbers n are the block quotas on this file structure as established by QUOTA.SYS.

UFD QUOTAS CHANGED
RESERVED (n) F.C.F.S (n) LOGGED-OUT (n)

The block quotas established by QUOTA.SYS on this file structure have changed since the user last used this file structure. The user's UFD are changed to specify the indicated quotas.

OPERATOR REQUESTED TO READY DRIVES

A request is queued to the operator to mount and ready the packs on the proper drives.

DPAn NOT AVAILABLE

The drive specified by the user is not currently available.

NOT ENOUGH DRIVES

There are not enough drives of the right type to mount the file structure.

MOUNT COMPLETE

The file structure is mounted and ready for use. The MOUNT command has completed.

Examples

```
.MOUNT MONITR: )
```

Asks the operator to mount the file sttructure named MONITR.

```
.MOUNT PAYROL:<DPA,DPB>/S )
```

Requests that the first unit of file structure PAYROL be mounted in Controller A, the second unit on Controller B, and any remaining units on any drives. All units are single access, (i.e., available only to this job).

DISMOUNT (DIS)

Function

The DISMOUNT command allows a user to withdraw his access to a file structure. This command enforces logged-out quotas (if necessary), allows physical removal of disk packs (if there are no other users of the pack), and removes the file structure name from the job's search list.

The UMOUNT CUSP runs privileged in the user's core area when the DISMOUNT command is typed. This CUSP scans the user's command string, checks its validity, and performs as much of the requested action as possible. The UMOUNT CUSP can complete all actions requested by the DISMOUNT command except for the action of physically removing a pack. When operator action is required, the UMOUNT CUSP writes a command file on 3,3 disk area for the OMOUNT CUSP. (By scanning the command files, the OMOUNT CUSP can request operator action.) When the command file is deleted, the UMOUNT CUSP knows that an operator action has been completed.

Command Format

DISMOUNT dev: switches

dev: = the file structure name as recorded in STRLST.SYS. This argument is required.

switches =	/HELP	(type this list)
	/CHECK	(check and list pending requests)
	/REMOVE	(notify operator to physically remove pack)

The switches are optional, and only enough characters to make the switch unique are required.

When /R is requested, the file structure is deleted from the system if no other users are using it, a request to remove the pack is queued to the operator, and the message WAITING... is typed to the user. If the user does not want to wait for confirmation of the operator action, he may type control-C.

Characteristics

The DISMOUNT command:

places the console in user mode,
runs the UMOUNT CUSP,
is used with disk monitors only.

Associated Messages

If the condition cannot allow the command to be continued, an error message preceded by a question mark is typed, and the command is aborted. (See Appendix F for these messages.)

WAITING...

A request for operator action has been queued and the command is waiting for completion of the action.

OTHER USERS - CAN'T REMOVE

A /R switch has been issued while there are other users of the file structure. The switch is ignored.

OPERATOR REQUESTED TO REMOVE PACKS

A request to physically remove the pack has been queued to the operator.

DISMOUNT COMPLETE

The DISMOUNT command has been completed.

FINISH (FIN)

Function

The FINISH command terminates any input or output currently in progress on the device specified and automatically performs the RELEASE UUO and DEASSIGN command, thus making the device available to another user.

Command Format

FINISH dev

dev = the logical or physical name of the device on which I/O is to be terminated. This argument is optional.

If dev is omitted, I/O is terminated on all devices, except the controlling Teletype, assigned to the job. The logical name of the controlling Teletype is cleared.

Characteristics

The FINISH command:
leaves the console in monitor mode,
requires core.

Associated Messages

?NO SUCH DEVICE

Either the device does not exist or it was not assigned to this job.

Examples

```
.FINISH CDR)
.
.FINISH DTA7)
.
.FINISH LPT)
.
```

CLOSE (CLO)

Function

The CLOSE command terminates any input or output currently in progress on the device specified, and automatically performs the CLOSE UUO. Files are CLOSEd, but not reset, and logical names and device assignments are preserved.

Command Format

CLOSE dev

dev = the logical or physical name of the device on which I/O is to be terminated. This argument is optional.

If dev is omitted, I/O is terminated on all devices, except for the controlling Teletype, assigned to the job, and all files are CLOSEd.

Characteristics

The CLOSE command:

leaves the console in monitor mode,
requires core.

Associated Messages

?NO SUCH DEVICE
Either the device does not exist or it was not assigned to this job.

(continued on next page)

Examples

```
.CLOSE PTR
.CLOSE DEVA
.CLOSE
```

SET CDR

Function

The SET CDR command sets the filename for the next card-reader spooling intercept.

Command Format

SET CDR filename

filename = one-to three-character filename to be used on next card-reader INIT.

Characteristics

The SET CDR command:

leaves the console in monitor mode,
is used with disk monitors only,
requires LOGIN.

Associated Messages

None

Examples

```
.SET CDR A
.SET CDR MAS
```

SET SPOOL

Function

The SET SPOOL command adds devices to or deletes devices from the current spool list.

Command Formats

1) SET SPOOL dev1, dev2, ...devn

   adds the specified devices to the job's spool list.

2) SET SPOOL ALL

   places all spooling devices into the spool list.

3) SET SPOOL NONE

   clears the entire spool list.

4) SET SPOOL NO dev1, dev2, ...devn

   removes the specified devices from the job's spool list.

   dev1, dev2,...devn = names of one or more devices to be added to or deleted from the current spool list.

Characteristics

The SET SPOOL command:

leaves the console in monitor mode,
is used with disk monitors only,
requires LOGIN.

Restrictions

To unspool devices, the job must have either the privilege bit set in JBTPRV, or the bit .UNSPL set in the STATES word.

Associated Messages

?NO PRIVS TO UNSPOOL

The job does not have privileges to unspool devices.

Examples

```
.SET SPOOL CDP
.SET SPOOL NO LPT
.SET SPOOL NONE
```

SEND (SEN)

Function

The SEND command provides a mechanism for one-way interconsole communication. (This command replaces the TALK command.) A line of information is transmitted from one terminal to another, with the identification of the terminal sending the information.

When the SEND command is sent from the operator's console (OPR) it allows a broadcast of a line of information to all terminals in the system. This allows important information to be dispersed, such as system shutdown or hardware problems.

A busy test is made before the message is sent unless the sender of the message or the receiver of the message is OPR. The receiver of the message is considered busy if his terminal is not at monitor command level. If the receiver is busy, the sender receives the message BUSY and the information is not sent. If the receiving console is turned off, the information appears to have been sent, since the hardware cannot detect this condition on hard-wired terminals.

Command Format

SEND dev: text

or

SEND JOB n text

dev = any physical Teletype name (CTY included) or OPR. If the Teletype sending the message is OPR, the argument may be ALL to provide the broadcast operation.

n = the job number to which the message is to be sent.

(continued on next page)

The message printed on the receiving terminal appears as follows:

;;TTY n: - text

where

n is the TTY sending the message, and text is the message.

Characteristics

The SEND command:

leaves the console in monitor mode.

Associated Messages

?BUSY

The receiving terminal is not at monitor command level.

?ILLEGAL JOB NUMBER

The job number is too large.

?NO SUCH TTY

The console number specified is not part of the system configuration.

?NOT A JOB

The job number specified does not exist.

Examples

```
.SEND OPR: PLEASE WRITE-ENABLE DTA3 )
.
```

PLEASE (PL)

Function

The PLEASE command allows the user non-conflicting two-way communication with the machine operator.

Command Format

PLEASE dev: text)

dev: = device (TTY) with which to communicate. If absent, TTY0: is assumed.
text = the user's message. The argument is required. Characters are not transmitted until the RETURN, vertical tab, or form feed key is depressed, at which point the entire line is transmitted.

When the user depresses the RETURN, vertical tab, or form feed key, a message informing the operator of the TTY number, proj-prog number of the user, and the time of day is printed on dev: An ALTMODE or control-C on either the user's console or dev: causes communication to terminate and the user's TTY to be left in monitor mode. Note that when the line terminates with an ALTMODE, the line is typed but the operator response is not waited for.

Characteristics

The PLEASE command:

places the console in user mode,
runs a CUSP.

Associated Messages

OPERATOR BUSY, HANG ON PLEASE

The user must wait for the operator to become available. The user does not need to issue another PLEASE command.

OPERATOR HAS BEEN NOTIFIED

The operator is available, and the user may continue with his message or wait for a response from the operator.

Example

```
.PLEASE TELL ME WHEN DTA3 WIL BE FREE)
OPERATOR HAS BEEN NOTIFIED
IN HALF AN HOUR
THANKS
↑C

.
```

CORE (COR)

Function

The CORE command types out or modifies the amount of core assigned to the user's job.

Command Format

CORE n

n = a decimal number. This argument is optional.

If n is omitted, monitor types out the amount of core used and does not change the core assignment.

If n = 0, the low and high segments disappear from the virtual addressing space of the job.

If n > 0, n represents the total number of 1K blocks of core to be assigned to the job from this point on.

If n is less than high plus minimum low segment size, n plus high segment size is assumed.

Characteristics

The CORE command:

leaves the console in monitor mode,
does not allow an active device.

requires a job number & LOGIN

Associated Messages

In Multiprogramming Systems:

m/p

Key: m = number of 1K blocks in low segment.
p = maximum K per job. Free plus dormant core.

(continued on next page)

In Swapping Reentrant Systems:

m+n/p CORE
VIR. CORE LEFT=v

Key: m = number of 1K blocks in low segment.
n = number of 1K blocks in high segment.
p = maximum K per job.

Maximum physical user core unless limited by operator or there are jobs locked in core (refer to Chapter 8).

v = number of K unassigned in core and on the swapping device.

?TRY LARGER ARG

n is too small for this program. This message is followed by the standard output.

Example

```
.CORE 5
.CORE
5+0/46K CORE
VIR. CORE LEFT = 274
```

R GRIPE

Function

This command runs the GRIPE CUSP which reads text from a user and records it in a disk file. The GRIPE CUSP enables users to record comments and complaints.

Command Format

R GRIPE

When the CUSP responds with a YES?, type the text, using as many lines as necessary, terminated with an altmode. The text is written as a file and includes a header with the date, time, and project programmer number of the user writing the comment. Therefore, the user does not need to identify himself.

Characteristics

The R GRIPE command:

places the console in used mode,
runs the GRIPE CUSP,
requires a job number and LOGIN.

Associated Messages

None

Example

```
.R GRIPE )

YES? (TYPE ALTMODE WHEN THROUGH) THIS CONSOLE IS
ALMOST OUT OF PAPER$
THANK YOU
EXIT

.
```

RESOURCES (RES)

Function

The RESOURCES command prints the names of all available devices (except TTY's and PTY's), all file structures, and all physical units not in file structures (unless they are down or nonexistent).

Command Format

RESOURCES

Characteristics

The RESOURCES command:

leaves the console in monitor mode,
does not require LOGIN.

Associated Messages

None

Example

```
.RES
DSKA,DSKB,DPA2,CDR,PTR,LPT,DTA0,1,3,7,MTA0,1,PTP
.
```

2.5 SOURCE FILE PREPARATION COMMANDS

The following commands call in the editing programs and cause these programs to open a specified text file for editing. Two of these commands call the TECO CUSP and two call the LINED CUSP (a disk-oriented version of EDITOR). For each editor, one command causes an existing file to be opened for changes and the other command causes a new file to be created. Each command requires a filename as its argument and may have an optional extension.

Filenames are one to six letters or digits. All letters or digits after the sixth are ignored. A filename is terminated by any character other than a letter or digit. If a filename is terminated by a period, a filename extension is assumed to follow. A filename extension is from zero to three letters or digits. It is generally used to indicate file format. The filename extension is terminated by any character other than a letter or digit. (See Appendix G).

Each time one of the commands listed below is executed, the command with its argument is remembered as a file on the disk; therefore, the filename edited last may be recalled for the next edit without specifying the arguments again. For example, if the command

```
.CREATE PROG1 .MAC
```

is executed, then the user may later type the command

```
.EDIT
```

instead of

```
.EDIT PROG1 .MAC
```

assuming no other source file preparation command was used in the interim.

pdp10

ANNOUNCEMENT

COMPILE Version 15A is available and will be distributed automatically as a SOUP update via paper tape. A list of corrected bugs is on the overleaf.

BUGS FIXED

1. STR 10-2339
COMPIL DID NOT ALLOW RENAME WITH PROTECTION.
REN FOO<155>=FOO

2. STR 10-2557
MULTIPLE ASSIGNMENTS DID NOT GIVE AN ERROR.
E.G. T1=T2=T3
COMPIL IGNORED AN EXTENSION SPECIFIED FOR THE OUTPUT FILE AND USED REL INSTEAD.
E.G. COM T1.RL1=T1

3. STR 10-2604
PIP CLASS COMMANDS COULD NOT HANDLE [PROJECT-PROG] NUMBERS.
E.G. DEL FOO[1,4]

4. DEC STANDARD VERSION OF COMPIL (WITH IMPCOR) WAS 3K PURE.
VERSION 15A IS 2K PURE.

5. VERSION 15A DOES A "CLOSE 20" TO RETAIN THE NAME BLOCKS IN THE LEVEL D DATA BASE.

6. /MAP NOW TAKES AN OPTIONAL FILE NAME.
E.G. /MAP:5S02
THIS IS USED IN THE MONITOR LOAD FILE.
THE MAP IS NOW OUTPUT JUST BEFORE THE LOADER TERMINATES.

7. THE TECO COMMAND NOW TAKES OPTIONAL DEVICE AND PROJ-PROG. SPECIFICATIONS.

CREATE (CREA)[1]

Function

The CREATE command runs LINED (Line Editor for disk) and opens a new file on disk for creation.

Command Format

CREATE file.ext

file.ext = any legal filename and filename extension. The filename is required; the filename extension is optional.

Characteristics

The CREATE command:

places the console in user mode,
runs the LINED CUSP,
is used with disk monitors only,
requires a job number and LOGIN.

Associated Messages

Refer to Table 2-1

Example

```
.CREATE TEST1.F4
*
```

[1]This command runs the COMPIL CUSP, which interprets the commands before running LINED.

EDIT (ED)[1]

Function

The EDIT command runs LINED (Line Editor for disk) and opens an already existing sequence-numbered file on disk for editing.

Command Format

EDIT file.ext

file.ext = a filename and filename extenstion of an existing file.

Characteristics

The EDIT command:

places the console in user mode,
runs the LINED CUSP,
is used with disk monitors only,
requires a job number and LOGIN.

Associated Messages

Refer to Table 2-1

Example

```
.EDIT TEST.F4
*
```

[1]This command runs the COMPIL CUSP, which interprets the commands before running LINED.

MAKE (MA)[1]

Function

The MAKE command runs TECO (Text Editor and Corrector) and opens a new file on the disk for creation.

Command Format

MAKE file.ext

file.ext = any legal filename and filename extension. The filename is required; the filename extension is optional.

Characteristics

The MAKE command:

places the console in user mode,
runs the TECO CUSP,
is used with disk monitors only,
requires a job number and LOGIN.

Associated Messages

Refer to Table 2-1

Example

```
.MAKE TEST3.MAC
*
```

[1]This command runs the COMPIL CUSP, which interprets the commands before running TECO.

TECO (TE)[1]

Function

The TECO command runs TECO and opens an already existing nonsequence-numbered file on disk for editing.

Command Format

TECO file.ext

file.ext = a filename and filename extension of an existing file.

Characteristics

The TECO command:

places the console in user mode,
runs the TECO CUSP,
is used with disk monitors only,
requires a job number and LOGIN.

Associated Messages

Refer to Table 2-1

[CREATING NEW FILE]

The specified file does not exist; therefore, a MAKE command is assumed by the COMPIL CUSP.

Example

```
.TECO TEST1.MAC ↵
*
```

[1]This command runs the COMPIL CUSP, which interprets the commands before running TECO.

Table 2-1
Monitor Command Diagnostic Messages
(For File Manipulation)

Message	Meaning
COMMAND ERROR	The COMPIL CUSP cannot decipher the command.
DEVICE NOT AVAILABLE	Specified device could not be initialized.
EXECUTION DELETED (typed by LOADER)	Errors detected during assembly, compilation, or loading prevent a program from being executed. Loading will be performed, but LOADER will EXIT to the monitor without starting execution.
FILE IN USE OR PROTECTED	A temporary command file cannot be entered in the user's UFD.
?FILENAME ALREADY IN USE	The specified file already exists.
INPUT ERROR	I/O error occurred while reading a temporary command file from the disk.
?INPUT FILE NOT FOUND	The specified file does not exist.
LINKAGE ERROR	I/O error occurred while reading a CUSP from device SYS:.
NESTING TOO DEEP	The @ construction exceeds a depth of nine (may be due to a loop of @ command files).
NO SUCH FILE - file.ext	Specified file cannot be found (may be a source file or a file required for operation of COMPIL CUSP).
NOT ENOUGH CORE	System cannot supply enough core for use as buffers or to read in a CUSP.
OUTPUT ERROR	I/O error occurred while writing a temporary command file on disk.
PROCESSOR CONFLICT	Use of + construction resulted in a mixture of source languages.
TOO MANY NAMES or TOO MANY SWITCHES	Command string complexity exceeds table space in COMPIL CUSP.
UNRECOGNIZABLE SWITCH	An ambiguous or undefined word followed a slash (/).

2.6 FILE MANIPULATION COMMANDS

Some of the following commands perform complex functions requiring a number of commands on a less sophisticated system. The commands below list the user's files and file directories, rename and delete files, provide remote control of DECtapes, allocate disk space, and manipulate job search lists.

TYPE (TY)

Function

The TYPE command directs PIP (Peripheral Interchange Program) to type the contents of the named source file(s) on the user's Teletype.

To stop the typing, type ↑C twice.

Command Format

TYPE list

list = a single file specification or a string of file specifications separated by commas. This argument is required.

A file specification may consist of a filename (with or without an extension), a device name if the source file is not on disk, a project-programmer number, if the source file is not in the user's disk area, and a protection.

Examples of file specifications:

```
PROG, PROG1.MAC,
DTA3:PROG2,PROG4[10,16]
```

In addition, the * construction may be used as follows:

filename.*	All files with this filename and any extension.
*.ext	All files with this extension and any filename.
.	All files

Characteristics

The TYPE command:

leaves the console in monitor mode,
runs the COMPIL CUSP,
is used with disk monitors only,
requires LOGIN.

Associated Messages

Refer to Table 2-1

Examples

```
.TYPE FILEA,DTA0:FILEB.MAC
.TYPE *.TMP,DTA4:C[15,107]
```

LIST (LI)

Function

The LIST command directs PIP to list contents of named source file(s) on the line printer (LPT).

Command Format

LIST list

list = a single file specification or a string of file specifications separated by commas. This argument is required.

Characteristics

The LIST command:

leaves the console in monitor mode,
runs the COMPIL CUSP,
is used with disk monitors only,
requires LOGIN.

Associated Messages

Refer to Table 2-1

Examples

```
.LIST TEST.*↵
.LIST *.MAC↵
.LIST DTA4:A,B,C↵
```

R PRINT

Function

The R PRINT command queues files upon the disk to be printed when the line printer is available.

Command Format

R PRINT

The user types in the names of the files to be printed, separated by commas. Only disk devices may be specified. If no device is specified, DSK is assumed. A particular file structure may be specified when there is more than one file with the same name in different file structures. Filenames may be continued on the next line by typing a hyphen followed by a carriage return. To delete a file after it is printed, insert the /D switch after the filename. To preserve a file that PRINTR would normally delete, insert the /P switch after the filename. To make more than one copy of the file, type a number from 2 to 9 as a switch.

Characteristics

The R PRINT command:

places the console in user mode,
runs the PRINTR CUSP,
requires a job number and LOGIN.

Associated Messages

?CAN'T FIND FILE file.ext

The specified file could not be found.

Example

```
.R PRINT
*TEST1.LST
*TEST2.LST, TEST3.LST, TEST1.MAC/2
*↑C
.
```

DIRECT (DIR)

Function

The DIRECT command lists the directory entries (filename, filename extension, size in blocks written, protection if file is on disk, and date created) specified by list.

Command Format

DIRECT list

list = a single file specification or a string of file specifications separated by commas. This argument is optional.

If list is omitted, DSK:*.* is assumed, and the directories in all file structures as defined by the job search list are listed, starting with the file structure name as a header.

Two switches may be used with the DIRECT command:

/F	Lists short form of directory (i.e., filename and filename extension only).
/L	Lists on the line printer (LPT) instead of Teletype.

Characteristics

The DIRECT command:

leaves the job in monitor mode,
runs the DIRECT CUSP,
is used with disk monitors only,
requires LOGIN.

Associated Messages

Refer to Table 2-1

Examples

```
.DIR DTA3: )                          Lists all files on DTA3
.DIR *.MAC )                          Lists all files with MAC filename extension in
                                      all file structures in the job search list.
.DIR TEST.F4[27,60] )                 Lists the directory entry for file TEST.F4 in
                                      user area 27,60.
.DIRECT )
DSKB: [10,63]

TEST    F4              01      <155>       11-SEP-70
TEST1   MAC             01      <055>       11-SEP-70

TOTAL BLOCKS            02

.
```

R LOOKFL

Function

The R LOOKFL command types all the characteristics of a single disk file on the user's console.

Command Format

R LOOKFL

The CUSP responds with the word FILE, and the user types the filename and filename extension of the file in which he is interested.

FILE: dev:file, ext [proj,prog]

The output is written on file TTY:LOOKFL.TXT.

Characteristics

The R LOOKFL command:

returns console to monitor mode via EXIT,
runs the LOOKFL CUSP,
is used with disk monitors only,
requires LOGIN.

Example

```
.R LOOKFL )

FILE: PIP.HGH )

DPA0:PIP.HGH[10,63]

ACCESS DATE: 12-SEP-70
CREATION TIME, DATE: 1800  12-SEP-70
ACCESS PRIVILEGS: 055
MODE: 16
WORDS WRITTEN: 3057.
VERSION NUMBER: 0,0
ESTIMATED LENGTH: 0.
BLOCKS ALLOCATED: 30.
POSITION OF LAST ALLOCATION: 0
NONPRIVILEGED CUSTOMER ARG: 000000000000
TAPE LABEL:
STATUS BITS: 400000000000
ERROR LOGICAL BLOCK: 0
ERROR LOGICAL UNIT: 0
NUMBER OF BAD BLOCKS: 0.
AUTHOR: 10,63
NEXT STR:
PREVIOUS STR:
PRIVILEGED CUSTOMER ARG: 000000000000
DATA BLOCK IN UFD: 567

EXIT

.
```

R DMPFIL

Function

The R DMPFIL command prepares an octal dump of part or all of a user file.

Command Format

.R DMPFIL
*dev:ofile.ext ← dev: ifile.ext/switch

(continued on next page)

dev:ofile.ext = the output file.

If dev: is omitted, LPT is assumed.
If .ext is omitted, .LST is assumed.
If ofile is omitted, ifile is assumed.
If entire output specification (including ←) is omitted, LPT: ifile .LST is assumed.

dev:ifile.ext/switch = the input file.

If dev: is omitted, DSK is assumed.

/switch =

/nnnnnnD	Dump DECtape beginning at block nnnnnn octal, includes listing of directory.
/nnnnnnK	Assume file is save file, dump as core dump beginning at location nnnnnn.
/nnnnnnH	Assume file is high segment file, dump as core dump.
/nnnnnnF	Dump disk as a standard file, beginning at logical block nnnnnn. If nnnnnn is 0, the RIB is printed.
/nnnnnnS	Dump disk, beginning at logical block nnnnnn with respect to file structure or unit depending on input device name. If nnnnnn is 0, block 0 is printed. User must be privileged since this operation uses a super USETI UUO, or the user must have mounted the file structure or pack as single access.
/nnnnnnT	Stop dump at nnnnnn octal. This switch may be combined with any of the above.

The range of nnnnnn is from 0 to 777777. To terminate the dump in the middle of the operation, type ↑C and REENTER. This action closes the files properly.

Characteristics

The R DMPFIL command:

places the console in user mode,
runs the DMPFIL CUSP.

Associated Messages

?SYNTAX ERROR

There is a syntax error in the command string.

Example

```
.ASSIGN TTY LPT
TTY24 ASSIGNED

.R DMPFIL

*PIP.HGH/400010H/400020T
DUMP OF FILE  PIP.HGH/400010H TO   09-12-70  18:32

  400010  001073400010  474300000000  402000000157  205000637163  047000
000055  200000404540  202000000257  200000405527
  400020  047000000041

*
```

FILE (FIL)

Function

The FILE command provides remote control of DECtape-to-disk and disk-to-DECtape transfers on operator-handled DECtapes.

Command Format

FILE option

option = F — Files information onto a DECtape. Requires Tape ID and list of filenames as arguments. The tape ID is a decimal number (user's tape), P (project tape), or A (general tape). Upon completion, an automatic FILE L is performed.

option = Z — Zeroes the directory of the DECtape before the files are copied and then performs the same operations as the F option. After the files are copied, an automatic FILE L is performed.

option = R — Recalls (transfers) information from the user's DECtape to the disk. After the files are transferred, an automatic FILE L is performed.

option = L — Reads the directory of a DECtape and places it in the user's disk area as an ASCII file with filename tapen.DIR. Tapen is the number of the user's DECtape and is the only argument. The user may then read the directory with a monitor command string. (See Examples).

option = C — Checks the queue of FILE commands to be read to determine if any of the user's requests are still pending. No argument is required. Pending request will be listed.

option = D — Deletes the specified files from DECtape. Requires Tape ID and list of filenames as arguments.

The C option is the only request that is performed immediately. The other requests are placed in a queue to be performed when possible. The user's console and job are free to proceed before the request is completed. The option argument is optional. If an argument is not specified, a brief dialogue is performed.

Characteristics

The FILE command:

leaves the console in monitor mode,
runs the UMOUNT CUSP,
is used with disk monitors only,
requires a job number and LOGIN.

Associated Messages

NONE PENDING

None of the user's requests is pending.

Examples

option = F

```
.FILE F,2, MAIN.F4,NAME.MAC
```

option = Z

```
.FILE Z,1, TEST.MAC,JOBS.CBL
```

option = R

```
.FILE R,2, *.MAC
```

option = L

```
.FILE L,1
```

The user may then read this directory with the monitor command string

```
.TYPE 1.DIR
```

option = C

```
.FILE C
```

option = D

```
.FILE D,2, FILE1,FILE2.*
```

R FILEX

Function

The R FILEX command runs the FILEX program. This program is a general file transfer program intended to convert between various core image formats, and to read and write various directory formats. Files are transferred as 36-bit data. The only processing on the data is that necessary to convert between various core image representations.

Command Format

.R FILEX↵
*dev:ofile.ext [proj,prog] <nnn>/switches←dev:ifile.ext [proj,prog] <nnn>/switches

If the project-programmer and/or the switches appear after the device name, they apply to all the following files. If they appear after the filename, the specifiers apply only to the preceding file. The input filename or extension may be * in which case the usual processing of the * construction occurs (refer to the TYPE command). The output filename and extension may be * in which case the filename and extension of the input file is copied. If the output filename or extension is missing, the same procedure occurs as with the * construction, except that all core image files are written with the default extension and format appropriate to the output device (unless overridden by switches).

If a protection <nnn> is not specified, files are written with the system standard protection unless the files are being written on SYS. On SYS., files are written with protection <155>, except for files with extension .SYS. These files have the default protection of <157>.

Meaning of Switches:

DECtape Format Specifiers

/M - MIT project MAC PDP-6/10 DECtape format
/O - Old DEC PDP-6 DECtape format
/T - normal PDP-10 directory format

File Format Specifiers

/B - binary processing; overrides default extension.
/C - compressed; save file format. This format is assumed for files with extensions .SAV, .LOW, .SVE. The default output extension is .SAV unless the input extension is .LOW or .SVE, in which case the extension remains unchanged.

(continued on next page)

File Format Specifiers (Cont)

/D - dump format. This format is assumed for files with extension .DMP.

/E - expanded core image files (used by FILDDT). This format is assumed for files with extension .XPN. The default output extension is .XPN.

/S - simple block (SBLK) format, project MAC's equivalent of .SAV format. The default output extension is .BIN.

DECtape Processing Specifiers

/G - (go on), ignores read errors on input device. FILEX checks the always-bad-checksum bit in the 5-series monitor, so this switch is not needed for files with .RPABC on (e.g. CRASH.SAV).

/L - (list), causes a directory on an input DECtape file to be typed on the Teletype, or causes a directory listing of the output DECtape at the end.

/P - (preserved), causes quick processing (/Q) and preserves the scratch file after processing for use by another command.

/Q - (quick), causes an input DECtape to be processed quickly via a scratch file.

/R - (reuse), reuses a scratch file preserved by a /P in a previous command.

/Z - (zero), causes the appropriate format of a zeroed directory to be written on a DECtape output file. If TAPEID appears in the output specifier, then TAPEID is written as the tape identifier in the directory. TAPEID may be 6 characters on a PDP-10 tape, 3 characters on a project MAC tape, and is not present on a PDP-6 tape.

Characteristics

The R FILEX command:

runs the FILEX CUSP.

Examples

```
.R FILEX
*DSK:←DTA1:TEST.DMP/C
```

The dump format file is compressed and written as TEST.SAV.

```
.R FILEX
*DSK:SER105.SAV[10,10]/E←DSKC:CRASH.SAV[1,4]
```

Copy CRASH.SAV to an expanded format file for FILDDT to examine.

R SETSRC

Function

The R SETSRC command runs the SETSRC CUSP to manipulate the job's file structure search list (refer to Paragraph 6.2.7). The file structure search list defines the order of search whenever device DSK is explicitly or implicitly specified by the user.

Command Format

R SETSRC

The CUSP responds with

TYPE H FOR HELP

The user may respond with

C) to create a new search list.
T) to type the current search list.
H) to get information about commands to be typed.

The current search list is typed in the form

fs1/s/s,fs2/s/s,...,FENCE,...,fs9/s/s

fs1 is the name of the first file structure.

/s is one or more of the following switches:

/C for create
/N for no create
/R for read only
/W for writeable

If no switches are specified, /C and /W are assumed.

/N indicates that the monitor is not to create files on this file structure when device DSK is specified. The user must specify the file structure name explicitly. This switch is useful when users have a small space on a fast file structure and a large space on a slow file structure. If /N is associated with the smaller file structure, all files are created on the larger file structure unless the smaller file structure is specified.

To create a new search list, type in the new search list (up to the FENCE) in the same form as it is typed out. The monitor moves the file structures that were in the old search list but that were not specified in the new search list to after the FENCE, thereby never decreasing the number of file structures that the user intends to use.

NOTE

Since the MOUNT command creates a UFD, it should be used to attach a new file structure to the search list.

Characteristics

The R SETSRC command:

places the console in the user mode,
runs the SETSRC CUSP,
requires LOGIN and a job number.

Associated Messages

None

Example

```
.R SETSRC
*T
DSKB, FENCE
*C DSKA,DSKB
*T
DSKA, DSKB, FENCE
*C DSKB
*T
DSKB, FENCE, DSKA
*
```

R ALCFIL

Function

The R ALCFIL command runs the ALCFIL CUSP to allocate space for a new file or re-allocate space for an existing file on the disk in one contiguous region. The size of the region is restricted by the size of the cluster count field (usually 512) times the cluster size of the file structure.

Command Format

R ALCFIL

The CUSP responds with

```
/H FOR HELP
FILE?
```

The user may respond with

```
dev:file.ext[proj.prog.]
or  /H (for help)
or  /X (to exit)
```

where dev: is a file structure or physical unit name. If dev: is omitted DSK is assumed. If one of the other arguments is omitted, 0 is assumed. If a filename is specified, the number of blocks presently allocated, if non-zero, is typed. ALCFIL responds

ALLOCATE?

User may type N or N,M (decimal numbers)

N = total number of blocks to be allocated for the file.
M = logical block within the file structure or unit (depending on dev:) where the allocation is to begin.

If the new blocks cannot be allocated, an error message is given and ALCFIL begins again. If the new blocks can be allocated, the message

ALLOCATED

is typed.

Since an extended ENTER (refer to Paragraph 6.2.8.2) is executed to allocate the new blocks, the file need not exist before allocating the blocks.

Characteristics

The R ALCFIL command:

places the console in user mode,
runs the ALCFIL CUSP,
requires LOGIN and a job number.

Associated Messages

BLOCK NOT FREE

M specifies a unit or file structure logical block that is not free.

n BLOCKS ALREADY ALLOCATED

The file already exists. The new specification replaces the old specification, rather than updating the old.

Associated Messages

Refer to Table 2-1

Example

```
.RENAME T11.MAC=T1.MAC )
FILES RENAMED:
T1.MAC

.RENAME *.BAK=*.MAC )
FILES RENAMED:
T11.MAC
T2.MAC
T3.MAC

.
```

CREF (CREF)

Function

The CREF command runs CREF and lists on the line printer any CREF listing files generated by previous COMPILE, LOAD, EXECUTE, and DEBUG commands using the /CREF switch since the last LOGIN. The file containing the names of these CREF-listing files is then deleted so that subsequent CREF commands will not list them again. If the logical device name LPT is assigned to DSK, the CREF files are converted to LST files with the same filenames.

Command Format

CREF

Characteristics

The CREF command:

leaves the console in monitor mode,
runs the COMPIL CUSP,
is used with disk monitors only,
requires LOGIN.

Associated Messages

Refer to Table 2-1

Example

```
.CREF )
.
```

2.7 COMPILATION COMMANDS

Each time a COMPILE, LOAD, EXECUTE, or DEBUG command is executed, the command with its arguments is remembered as a file on the disk or in core if the monitor has the TMPCOR feature implemented; therefore, the filename used last may be recalled for the next command without specifying the arguments again. (Refer to last paragraph in Section 2.5.)

COMPILE (COM)

Function

The COMPILE command produces relocatable binary file(s) for the specified program(s). The use of the MACRO assembler, COBOL compiler, and/or the FORTRAN IV compiler is determined as follows.

Condition	Action
If no .REL (binary) file	Translate source file
If source-file [date,time] is later than or equal to binary-file [date,time]	Translate source file
If other than above	Do not translate source file; use current .REL (binary) file.

Source File Extension	Translator Used
.MAC	MACRO assembler
.F4	FORTRAN IV compiler (F40)
.CBL	COBOL compiler
Other than above, or null	"Standard processor" is used (see 2.7.2)

Command Format

COMPILE list

list = a single file specification, or a string of file specifications separated by commas.

Characteristics

The COMPILE command:

leaves the console in monitor mode,
runs the COMPIL CUSP,
is used with disk monitors only,
requires LOGIN.

Restrictions

The * construction may not be used.

Associated Messages

Refer to Table 2-1

Example

```
.COMPILE PROGA
```

LOAD (LOA)

Function

The LOAD command performs the COMPILE function for the specified program(s), then runs LOADER and loads the .REL files.

Command Format

LOAD list

list = a single file specification, or a string of file specifications separated by commas.

Characteristics

The LOAD command:

leaves the console in monitor mode,
runs the COMPIL CUSP,
is used with disk monitors only,
requires LOGIN.

Associated Messages

Refer to Table 2-1

Example

```
.LOAD FILEA,FILEB,%60000FILEC )
```

Pass origin switch to Loader; refer to Paragraph 2.7.4.

```
.LOAD TEST )
MACRO: TEST
LOADING

LOADER 1K CORE

EXIT

.
```

EXECUTE (EX)

Function

The EXECUTE command performs the COMPILE and LOAD functions for the specified program(s) and begins execution of the loaded program.

Command Format

EXECUTE list

list = a single file specification or a string of file specifications separated by commas.

Characteristics

The EXECUTE command:

places the console in user mode,
runs the COMPIL CUSP,
is used with disk monitors only,
requires LOGIN.

Associated Messages

Refer to Table 2-1

Example

```
.EX TEST )
MACRO: TEST
LOADING

LOADER 1K CORE
EXECUTION
```

DEBUG (DEB)

Function

The DEBUG command performs the COMPILE and LOAD functions and, in addition, prepares for debugging. DDT (the Dynamic Debugging Technique program) is loaded first, followed by the user's programs with local symbols. DDT is entered on completion of loading.

Command Format

DEBUG list

list = a single file specification or a string of file specifications separated by commas.

Characteristics

The DEBUG command:

places the console in user mode,
runs the COMPIL CUSP,
is used with disk monitors only,
requires LOGIN.

Associated Messages

Refer to Table 2-1

Examples

```
.DEBUG/L FILEA,FILEB,FILEC/N,FILED
```

Generate listings for FILEA, FILEB, and FILED; refer to Paragraph 2.7.2.

```
.DEBUG TEST
MACRO: TEST
LOADING

LOADER 2K CORE
EXECUTION

./      BLT 15,0(16)
```

2.7.1 Extended Command Forms

The commands previously explained are adequate for the compilation and execution of a single program or a small group of programs at one time. However, the assembly of large groups of programs, such as the FORTRAN library or the Timesharing Monitor, is more easily accomplished by one or more of the extended command forms.

2.7.1.1 Indirect Commands (@ Construction) - When there are many program names and switches, they can be put into a file; therefore, they do not have to be typed in for each compilation. This is accomplished by the use of the @ file construction, which may be combined with any COMPIL-class commands.

The @ file may appear at any point after the first word in the command. In this construction, the word file must be a filename, which may have an extension and project-programmer numbers. If the extension is omitted, a search is made for the command file with a null extension and then for a command file with the extension .CMD. The information in the command file specified is then put into the command string to replace the characters @ file.

For example, if the file FLIST contains the string

```
FILEB,FILEC/LIST,FILED
```

then the command

```
.COMPILE FILEA,FILEB,FILEC/LIST,FILED,FILEZ
```

could be replaced by

```
.COMPILE FILEA,@FLIST,FILEZ
```

Command files may contain the @ file construction to a depth of nine levels. If this indirecting process results in files pointing in a loop, the maximum depth is rapidly exceeded and an error message is produced.

The following rules apply in the handling of format characters in a command file.

a. Spaces are used to delimit words but are otherwise ignored. Similarly, the characters TAB, VTAB, and FORM are treated like spaces.

b. To allow long command strings, command terminators (CARRIAGE RETURN, LINE FEED, ALTMODE) are ignored if the first nonblank character after a sequence of command terminators is a comma. Otherwise, they are treated either as commas by the COMPILE, LOAD, EXECUTE, and DEBUG commands or as command terminators by all other COMPIL-class commands.

c. Blank lines are completely ignored because strings of returns and line-feeds are considered together.

d. Comments may be included in command files by preceding the comment with a semicolon. All text from the semicolon to the line-feed is ignored.

e. If command files are sequenced, the sequence numbers are ignored.

2.7.1.2 The + Construction† - A single relocatable binary file may be produced from a collection of input source files by the "+" construction. For example: a user may wish to compile the parameter file, S.MAC, the switch file, FT50S.MAC, and the file that is the body of the program, COMCON.MAC. This is specified by the following command:

```
.COMPILE S+FT50S+COMCON
```

†Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

The name of the last input file in the string is given to any output (.REL, .CRF, and/or .LST) files (e.g., COMCON. in the preceding example). The source files in the "+" construction may each contain device and extension information and project-programmer numbers.

2.7.1.3 The = Construction† - Usually the filename of the relocatable binary file is the same as that of the source file, with the extension specifying the difference. This can be changed by the "=" construction, which allows a filename other than the source filename to be given to the associated output files. For example: if a binary file is desired with the name BINARY.REL from a source program with the name SOURCE.MAC, the following command is used.

```
.COMPILE BINARY=SOURCE
```

This technique may be used to specify an output name to a file produced by use of the "+" construction. To give the name WHOLE.REL to the binary file produced by PART1.MAC and PART2.MAC, the following is typed.

```
.COMPILE WHOLE=PART1+PART 2
```

Although the most common use of the "=" construction is to change the filename of the output files, this technique may be used to change any of the other default conditions. The default condition for processor output is DSK:source.REL[self]. For example: if the output is desired on DTA3 with the filename FILEX, the following command may be used:

```
EXECUTE DTA3:FILEX=FILE1.F4
```

2.7.1.4 The < > Construction† - The < > construction causes the programs within the angle brackets to be assembled with the same parameter file. If a + is used, it must appear before the < > construction. For example: to assemble the files LPTSER.MAC, PTPSER.MAC, and PTRSER.MAC, each with the parameter file S.MAC, the user may type

```
.COMPILE S+LPTSER, S+PTPSER, S+PTRSER
```

With the angle brackets, however, the command becomes

```
.COMPILE S+<LPTSER,PTPSER,PTRSER>
```

†Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

The user cannot type

```
.COMPILE <LPTSER,PTPSER,PTRSER>+S
```

2.7.2 Compile Switches†

The COMPILE, LOAD, EXECUTE, and DEBUG commands may be modified by a variety of switches. Each switch is preceded by a slash and is terminated by any non-alphanumeric character, usually a space or a comma. An abbreviation may be used if it uniquely identifies a particular switch.

These switches may be either temporary or permanent. A temporary switch is appended to the end of the filename, without an intervening space, and applies only to that file.

Example:

```
.COMPILE A,B/MACRO,C        (The MACRO assembler applies only to file B.)
```

A permanent switch is set off from filenames by spaces, commas or any combination of the two. It applies to all the following files unless modified by a subsequent switch.

Example:

```
.COMPILE /MACRO A,B,C
.COMPILE A /MACRO B,C
.COMPILE A,/MACRO,B,C
.COMPILE A,/MACRO B,C
```

2.7.2.1 Compilation Listings† - Listing files may be generated by switches. The listings may be of the ordinary or the cross-reference type. The operation of the switch produces a disk file with the extension .LST.

The compile-switches LIST and NOLIST cause listing and nonlisting of programs and may be used as temporary or permanent switches.

Listings of all three programs are generated by

```
.COMPILE /LIST A,B,C
```

A listing only of program A is generated by

```
.COMPILE A/LIST,B,C
```

---

†Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

Listings of programs A and C are generated by

```
.COMPILE /LIST A,B/NOLIST,C
```

The compile-switch CREF is like LIST, except that a cross-reference listing is generated, to be processed later by the CREF CUSP.

Unless the /LIST or /CREF is specified, no listing file is generated. When listing files are generated, the LIST and CREF commands can then be used to obtain printer output of the listing files.

Since the LIST, NOLIST, and CREF switches are commonly used, the switches L, N, and C are defined with the corresponding meanings, although there are (for instance) other switches beginning with the letter L. Thus, the command

```
.COMPILE /L A
```

produces a listing file A.LST (and A.REL).

2.7.2.2 Standard Processor - The standard processor is used to compile or assemble programs that do not have the extensions .MAC, .CBL, .F4, or .REL. A variety of switches set the standard processor. If all source files are kept with the appropriate extensions, this subject can be disregarded.

If the command

```
.COMPILE A
```

is executed and there is a file named A. (that is, with a blank extension), then A. will be translated to A.REL by the standard processor. Similarly, if the command

```
.COMPILE FILE.NEW
```

is executed, the extension .NEW, although meaningful to the user, does not specify a language; therefore, the standard processor is used. The user must be able to control the setting of the standard processor which is FORTRAN IV at the beginning of each command string.

The standard processor may be changed by the following compile-switches:

COBOL	change standard to COBOL
C	same as COBOL
MACRO	change standard to MACRO
M	same as MACRO
FORTRAN	change standard to FORTRAN IV
F	same as FORTRAN
REL	change standard to use relocatable binary; i.e., use existing .REL files, although a newer source file may be present (useful primarily in LOAD, EXECUTE, DEBUG commands).

These switches may be temporary or permanent. For example: assume that programs A, B, and C exist on the disk, with blank extensions. Then

```
.COMPILE A,B/M,C
```

will cause A and C to be translated by FORTRAN, B by MACRO. Also,

```
.COMPILE A,/M B,C
```

will cause A to be translated by FORTRAN, B and C by MACRO.

NOTE

Programs with .MAC, .CBL, and .F4 extensions are always translated by the extension implied, regardless of the standard processor unless forced by a temporary switch.

2.7.2.3 Forced Compilation - Compilation (or assembly) occurs if the source file is at least as recent as the relocatable binary file. The creation time for files is kept to the nearest minute. Therefore, it is possible for an unnecessary compilation to occur. If the binary is newer than the source, the translation does not usually have to be performed.

There are cases, however, where such extra translation may be desirable (e.g., when a listing of the assembly is desired). To force such an assembly, the switch COMPILE is provided, in temporary and permanent form. For example:

```
.COMPILE /CREF/COMPILE A,B,C
```

will create cross-reference listing files A.CRF, B.CRF, and C.CRF, although current .REL files may exist. The binary files will also be recreated.

The corresponding switch NOCOMPILE is also provided, to turn off the forced-compile mode. Note that this differs from the /REL switch, which turns off even the normal compilation caused by a source file that is newer than the .REL file.

2.7.2.4 Library Searches† - The LOADER normally performs a library search of the FORTRAN library. If it is necessary to search other files as libraries, the compile-switches LIBRARY and NOSEARCH (its complement) are provided. The switch /LIBRARY (equivalent to /LIB) signifies that all files to which it applies are searched in library search mode. The switch /NOSEARCH (equivalent to /N) signifies that all routines of the file to which it applies are loaded regardless of whether the routines are referenced or not. This is the normal loading mode, and the /NOSEARCH is used only to turn off the library search mode. Note that /NOSEARCH is not equivalent to /P of the Loader.

For example: if a special library file named SPCLIB.REL were kept on device SYS at a particular installation, to compile and load a user program, library search the special library, and search the normal FORTRAN library, the following command could be used:

```
.LOAD MAIN,SYS:SPCLIB/LIB
```

At this point, it should be noted that the program SPCLIB is not assembled simply because its source file is presumably not on device SYS. The COMPILE process will compile any program named in the command string, if its source is present and not older than the .REL file, unless prevented by the /REL switch.

2.7.2.5 Loader Maps - Loader maps are produced during the loading process by the compile-switch MAP. When the MAP switch is encountered, a loader map is requested from the loader. After a library search of LIB40, the map will be written with default filename MAP.MAP, in the user's disk area. An optional filename, preceded by a colon, may be specified after the MAP switch.

The MAP compile-switch is the one exception to the permanent compile-switch rule, in that it causes only one map to be output, although it may appear as a permanent switch.

```
.LOAD MAIN,SUB /MAP:MAIN
```

2.7.3 Processor Switches††

Occasionally it is necessary to pass switches to the assembler or compiler. For each translation (assembly or compilation), a command string is sent to the translator containing three parts: the source files, a binary output file, and a listing file. To add switches to those files, the user must:

---

†For more LOADER information, refer to the LOADER documentation in the PDP-10 Reference Handbook.

††Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

a. If the + construction is used, group the switches according to each related source filename.

b. Group the switches according to the three types of files (source, binary, and listing) for each source filename.

c. For each source filename, separate the groups of switches by commas.

d. Enclose all the switches for each source filename within one set of parentheses.

(SSSS)	Only source switches are present
(SSSS,BBBB)	Source and binary switches are present
(SSSS,BBBB,LLLL)	Source, binary, and listing switches are present.

e. Place each parenthesized string immediately after the source filename to which it refers.

Examples:

.DEBUG TEST(N)	Suppress typeout of errors during assembly.
.COMPILE OUTPUT=MTA0:(W,S,M)/L	Rewind the magtape (W), compile the first file, produce binary output for the PDP-6(S), and eliminate the MACRO coding from the output listing (M). Output files are given the names OUTPUT.REL and OUTPUT.LST.
.COMPILE/MACRO A=MTA0:(W,,Q)/L	Rewind the magtape (W), compile the first file, and suppress Q (questionable) error indications on the listing. Note that when a binary switch is not present, the delimiting comma must appear.
.COMPILE /MACRO A=MTA0:(,,Q)/L	Compile file at current position of the tape and suppress Q error indications on the listing. Note that when the source and binary switches are not present, the delimiting comma must appear.

2.7.4 Loader Switches

In usually complex loading processes, it may be necessary to pass loader-switches to the LOADER to direct its operation. This is accomplished by the % character. The % has the same meaning as that of the / in the Loader's command string. Also, like the /, the % takes one letter (or a sequence of digits and one letter) following it. Therefore, to set a program origin of 6000 for program C, the user types

```
.LOAD A,B,%600000C,D
```

COMPIL allows more than one loader switch to be specified. For example,

```
.LOAD PROG %F/MAP
```

The most commonly used switches are:

a. %S Load with symbols
b. %nO Set program origin to n
c. %F Cause early search of FORTRAN library
d. %P Prevent FORTRAN library search.

2.7.5 Temporary Files

The COMPIL CUSP deciphers the commands found in Tables 2-3 and 2-5 and constructs new commands for the referenced CUSPs. These new commands are written as temporary files in core or, if the TMPCOR area is full, on the disk, as are all of the monitor-level commands. COMPIL and the other CUSPs transfer control directly to one another without requiring additional typed-in commands from the user.

Temporary filenames have the following form:

nnnxxx.TMP

where nnn is the user's job number in decimal, with leading zeros to make three digits and xxx specifies the use of the file. In the filenames listed below, job number 1 will be assumed.

2.7.5.1 001SVC.TMP - This file contains the most recent COMPILE, LOAD, EXECUTE, or DEBUG command that included arguments. It is used to remember those arguments. (Refer to Paragraph 2.7.)

2.7.5.2 001EDS.TMP - This file contains the most recent EDIT, CREATE, TECO, or MAKE command that included an argument. It is used to remember that argument. (Refer to Paragraph 2.5.)

2.7.5.3 001MAC.TMP - This file contains commands to MACRO. It is written by COMPIL, and read by MACRO. It contains one line for each program to be assembled, and (if required) the command

NAME!

to cause MACRO to transfer control to the named CUSP ("name" may be F40, LOADER).

2.7.5.4 001FOR.TMP - This file corresponds to the one described in the preceding section, except that it is read by the FORTRAN IV compiler, F40.

2.7.5.5 001COB.TMP - This file corresponds to the one described in Paragraph 2.7.5.3, except that it is read by the COBOL compiler.

2.7.5.6 001PIP.TMP - This file is written by COMPIL and read by PIP. It contains ordinary PIP commands to implement the DIRECTORY, LIST, TYPE, RENAME, and DELETE commands.

2.7.5.7 001CRE.TMP - This file is written by COMPIL and read by CREF. It contains commands to CREF corresponding to each file which has produced a CREF listing on the disk.

COMPIL also reads this file, if it exists, each time a new CREF listing is generated, to prevent multiple requests for the same file, and to prevent discarding other requests that may not yet have been listed.

2.7.5.8 001EDT.TMP - This file is written by COMPIL for each EDIT, CREATE, TECO, or MAKE command, and is read by either the LINED or TECO CUSP.

For the commands MAKE or CREATE, it contains the command

EW file.ext (\$)(\$) or Sfile.ext (\$)

For the commands TECO or EDIT, it contains the command

EB file.ext (\$)(\$) or Sfile.ext ↵ ↓

2.7.5.9 001LOA.TMP - This file is written by COMPIL and contains commands to LOADER that are necessary for loading programs.

2.8 RUN CONTROL COMMANDS

By using a run control command, the user can load core image files from retrievable storage devices (i.e., disk, DECtape, magnetic tape). These files can be retrieved and controlled from the user's console. Files stored on disk and DECtape are addressable by name. Files on magnetic tape require the user to preposition the tape to the beginning of the file. (Refer to Table 3-1 for the description of job data area locations referenced by commands below.)

RUN (RU)

Function

The RUN command loads a core image from a retrievable storage device and starts it at the location specified within the file (JOBSA).

If the program has two segments, both the low and high segments are set up. If the high file has extension .SHR (as opposed to .HGH), the high segment will be shared. A two-segment program may have a low file extension (.LOW).

The RUN command 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.

Command Format

RUN dev:file.ext [proj,prog] core

dev: = the logical or physical name of the device containing the core image. The default device name is DSK: The colon following the device name is required.

file.ext = the name of the file containing the core image; .ext applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions. The default filename is the job's current name as set by the last R,RUN, GET, SAVE, or SSAVE command, the last SETNAM UUO, or the last command which ran a CUSP.

[proj,prog] = the project-programmer number; required only if core image file is located in a disk area other than the user's.

core = the amount of core to be assigned to the sum of the low and high segments if different from minimum core needed to load the program or from the core argument of the SAVE command which saved the file.

If core < the minimum low segment size, then an error message occurs.

If core ≥ the minimum low segment size and < the sum of the high segment and the minimum low segment size, then the core assignment is the low segment size.

If core ≥ the sum of the minimum low segment and the high segment size, then the core assignment is the size of both the low and high segments to be used.

Since previous core is returned, MTA must have the core argument because there is no directory telling how much core is for the low segment. Refer to Paragraph 2.8.1.

Characteristics

The RUN command:

places the console in user mode,
requires a job number and LOGIN.

Restrictions

On systems with a large amount of core memory, the user should not specify a core argument that forces the high segment to start higher than 400000 unless the programs high segment is self-relocating. If this is done, the ILLEGAL UUO error message is likely to occur.

Associated Messages

?dev: NOT AVAILABLE

The device has been assigned to another job.

?NO SUCH DEVICE

The device does not exist in this monitor configuration.

?nK OF CORE NEEDED

There is insufficient free core to load the file.

?NOT A SAVE FILE

The file is not a core image file.

?TRANSMISSION ERROR

A parity or device error occurred during loading.

?file.ext NOT FOUND

The program file requested cannot be found on the specified device.

?NO START ADR

Starting address was 0 because the user failed to specify a starting address in the END statement of the source program.

?ADDRESS CHECK FOR DEVICE dev

The save file is too large for the core assigned.

?LOOKUP FAILURE n

The LOOKUP to read the file failed, n is the disk error code (refer to Appendix E).

Examples

```
.RUN DSK:TEST )

.RUN DSK:HISTST [10,63] )
```

R (R)

Function

The R command is the same as RUN SYS:file.ext core. This command is the usual way to run a CUSP that does not have a direct monitor command to run it.

This command 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.

Command Format

R file.ext core

Arguments are the same as in the RUN command except that SYS: is used as the default device.

The extension applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions.

Characteristics

The R command:

places the console in user mode,
runs a CUSP,
requires a job number and LOGIN.

Associated Messages

?dev: NOT AVAILABLE

The device has been assigned to another job.

?NO SUCH DEVICE

The device does not exist in this monitor configuration.

?nK OF CORE NEEDED

There is insufficient free core to load the file.

?NOT A SAVE FILE

The file is not a core image file.

(continued on next page)

Associated Messages (Cont)

?LOOKUP FAILURE n

The LOOKUP to read the file failed, n is the disk error code (refer to Appendix E).

?TRANSMISSION ERROR

A parity or device error occurred during loading.

?file.ext NOT FOUND

The program file requested cannot be found on the specified device.

?NO START ADR

Starting address was 0 because the user failed to specify a starting address in the end statement of the source program.

?ADDRESS CHECK FOR DEVICE dev

The save file is too large for the core assigned.

Examples

```
.R PIP
*
.R PIP 5
*
```

GET (G)

Function

The GET command is the same as the RUN command except that the monitor types out

JOB SETUP

and does not start execution.

This command 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.

Command Format

GET dev:file.ext [proj.prog] core

The arguments and the defaults are the same as in the RUN command.

The extension applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions.

Characteristics

The GET command:

leaves the console in monitor mode,
does not allow an active device,
requires a job number and LOGIN.

Associated Messages

?dev: NOT AVAILABLE

The device has been assigned to another job.

?NO SUCH DEVICE

The device does not exist in this monitor configuration.

?nK OF CORE NEEDED

There is insufficient free core to load the file.

?NOT A SAVE FILE

The file is not a core image file.

?TRANSMISSION ERROR

A parity or device error occurred during loading.

?file.ext NOT FOUND

The program file requested cannot be found on the specified device.

?ADDRESS CHECK FOR DEVICE dev

The save file is too large for the core assigned.

?LOOKUP FAILURE n

The LOOKUP to read the file failed, n is the disk error code (refer to Appendix E).

Example

```
.GET SYS:PIP
JOB SETUP

.GET TEST
JOB SETUP
```

START (ST)

Function

The START command begins execution of a program previously loaded with the GET command. The old program counter is copied from JOBPC to JOBOPC.

Command Format

START adr

adr = the address at which execution is to begin if other than the location specified within the file (JOBSA). This argument is optional. If adr is not specified, the starting address comes from JOBSA.

Characteristics

The START command:

places the console in user mode,
does not allow an active device,
requires core,
requires a job number and LOGIN.

Associated Messages

?NO CORE ASSIGNED

No core was allocated to the user when the GET command was given, and no core argument was specified in the GET.

?NO START ADR

Starting address was 0 because the user failed to specify a starting address in the END statement of the source program.

Example

```
.START )
```

HALT (↑C)

Function

The HALT (↑C) command transmits a HALT command to the monitor command interpreter. It stops the job and stores the program counter in the job data area (JOBPC). Refer to Paragraph 2.1.1).

Command Format

HALT (↑C)

Characteristics

The HALT (↑C) command:

places the console in monitor mode.

Associated Messages

None

Example

```
↑C
.
```

CONT (CON)

Function

The CONT command starts the program at the saved program counter address stored in JOBPC by a HALT command (↑C) or a HALT instruction.

Command Format

CONT

Characteristics

The CONT command:

places the console in user mode,
requires core,
requires a job number and LOGIN.

Associated Messages

?CAN'T CONTINUE

The job was halted due to a monitor-detected error and cannot be continued.

Example

```
.CONT
```

JCONT

Function

The JOB CONTINUE command forces a continue of the specified job if the job was in a tC state because of a call to the device error message routine (HNGSTP).

Command Format

JCONT y

y = the number of the job to be continued. This argument is required.

Characteristics

The JCONT command:

places the console in monitor mode

Associated Messages

?NOT A JOB

The job specified does not exist.

?JOB NOT WAITING

The job specified is not waiting to be continued.

CONT BY OPR

The job has been continued by the operator.

Example

```
.JCONT
```

DDT (DD)

Function

The DDT command copies the saved program counter value from JOBPC into JOBOPC and starts the program at an alternate entry point specified in JOBDDT (beginning address of DDT as set by Linking Loader). DDT contains commands to allow the user to start or resume at any desired address.

Command Format

DDT

Characteristics

The DDT command:

places the console in user mode,
requires core,
requires a job number and LOGIN.

Associated Messages

?NO START ADR

DDT starting address was 0 (JOBDDT).

Example

.DDT ↵

REENTER (REE)

Function

The REENTER command is similar to the DDT command. It copies the saved program counter value from JOBPC into JOBOPC and starts the program at an alternate entry point specified in JOBREN (must be set by the user or his program).

Command Format

REENTER

Characteristics

The REENTER command:

places the console in user mode,
requires core,
requires a job number and LOGIN.

Associated Messages

?NO START ADR

REENTER starting address was 0 (JOBREN).

Example

```
.REE )
```

E (E)

Function

The E command examines a core location in the user's area (high or low segment).

Command Format

E adr

Adr is required the first time the E or D command is used. If adr is specified, the contents of the location are typed out in half-word octal mode.

If adr is not specified, the contents of the location following the previously specified E adr or the location of the previous D adr (whichever was last) are typed out.

Characteristics

The E command:

leaves the console in monitor mode,
requires core,
requires a job number and LOGIN.

Associated Messages

?OUT OF BOUNDS

The specified adr is not in the user's core area.

Example

```
.E 140
000140/ 264000 002616    .E
000141/ 000000 000000    .E
000142/ 000000 000000    .
.
```

D (D)

Function

The D command deposits information in the user's core area (high or low segment).

Command Format

D lh rh adr

lh = the octal value to be deposited in the left half of the location. This argument is required.

rh = the octal value to be deposited in the right half of the location. This argument is required.

adr = the address of the location into which the information is to be deposited. This argument is optional.

If adr is omitted, the data is deposited in the location following the last D adr or in the location of the last E adr (whichever was last).

Characteristics

The D command:

leaves the console in monitor mode,
requires core,
requires a job number and LOGIN.

Associated Messages

?OUT OF BOUNDS

The specified adr is not in the user's core area, or the high segment is write protected and the user does not have write privileges to the file that initialized the high segment.

Example

```
.D 266000 2616 140

.

.E 140
000140/ 047000 000000    .D 47000 1

.E
000140/ 047000 000001    .
```

SAVE (SA)

Function

The SAVE command writes out a core image of the user's core area on the specified device. It saves any user program (reentrant, one segment nonreentrant, or two segment nonreentrant) as one or two files. Later, when the program is loaded by a GET, R, or RUN command, it will be nonreentrant. If DDT was loaded with the program, the entire core area is written; if not, the area starting from zero up through the program break (as specified by JOBFF) is written.

Command Format

SAVE dev:file.ext core

dev: = the device on which the core image file is to be written. The default device name is DSK: The colon following the device name is required.

file.ext = the name to be assigned to the core image file. The default filename is the job's current name as set by the last R, RUN, GET, SAVE, or SSAVE command, or the last SETNAM UUO. Ext applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions. If ext is omitted and the program has only one segment, the ext is assumed to be .SAV. If ext is omitted and the program has two segments, the high segment will have extension .HGH, and the low segment will have extension .LOW.

core = the amount of core in which the program is to be run. This value is stored in JOBDAT as the job's core area (JOBCOR) and is used by the RUN and GET commands. Specified as number of 1K blocks. This argument is optional.

If core is omitted, only the number of blocks required by the core image area (as explained previously) is assumed.

Characteristics

The SAVE command:

leaves the console in monitor mode,
requires core,
does not allow an active device,
requires a job number and LOGIN.

Associated Messages

?nK CORE NEEDED

The user's current core allocation is less than the contents of JOBFF.

?DEVICE dev NOT AVAILABLE

Device dev: is assigned to another user.

?TRANSMISSION ERROR

An error was detected while reading or writing the core image file.

?ENTER FAILURE n

The ENTER to write the file failed; n is the disk error code (see Appendix E).

JOB SAVED

The output is completed.

?NO SUCH DEVICE

The device does not exist in this configuration.

Example

```
.SAVE DSK:TEST
JOB SAVED

.
```

SSAVE (SSA)

Function

The SSAVE command is the same as the SAVE command except that the high segment, if present, will be sharable when it is loaded with the GET command. To indicate this sharability, the high segment is written with extension .SHR instead of .HGH. A subsequent GET will cause the high segment to be sharable. Because an error message is not given if the program does not have a high segment, a user can use this command to save CUSPs without having to know which are sharable.

Command Format

SSAVE dev:file.ext core

Arguments are the same as in the SAVE command.

Characteristics

The SSAVE command:

leaves the console in monitor mode,
requires core,
does not allow an active device,
requires a job number and LOGIN.

Associated Messages

?nK CORE NEEDED

The user's current core allocation is less than the contents of JOBFF.

?DEVICE dev NOT AVAILABLE

Device dev: is assigned to another user.

?TRANSMISSION ERROR

An error was detected while reading or writing the core image file.

?ENTER FAILURE n

The ENTER to write the file failed; n is the disk error code (see Appendix E).

JOB SAVED

The output is completed.

?NO SUCH DEVICE

The device does not exist in this configuration.

Example

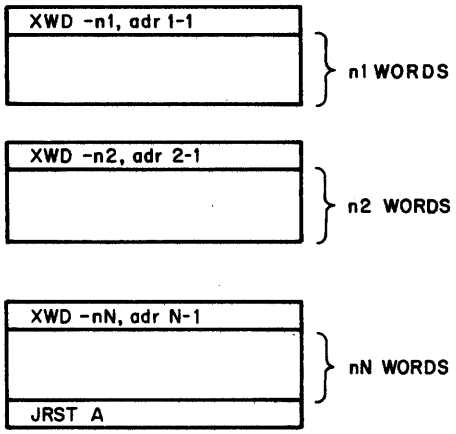
```
.SSAVE DSK:TEST)
JOB SAVED

.

.LOAD FILE1)
MACRO: FILE1
LOADING

LOADER 1K CORE
EXIT

.SSAVE)
JOB SAVED

.GET)
JOB SETUP
```

2.8.1 Additional Information on SAVE and SSAVE

Before writing SAVed or LOW files in response to SAVE and SSAVE commands, the monitor compresses the user's core image by eliminating consecutive blocks of zeroes. This technique is known as zero-compression and is used to save space on file media. Low segment files are zero-compressed on devices DTA, MTA, and DSK, but high segment files are not because the high segment may be shared at the time of the command.

Saved files are ordinary binary files and can be copied using the /B switch in PIP. Files with the LOW or SAV extension may be read in dump mode, but must be reexpanded before being run. The monitor expands the file after input on a RUN, R, or GET command.

The data format of a zero-compressed saved file consists of a series of IOWDs and data block pairs and is terminated by a JRST A where A is the contents of JOBSA. The format is as follows:



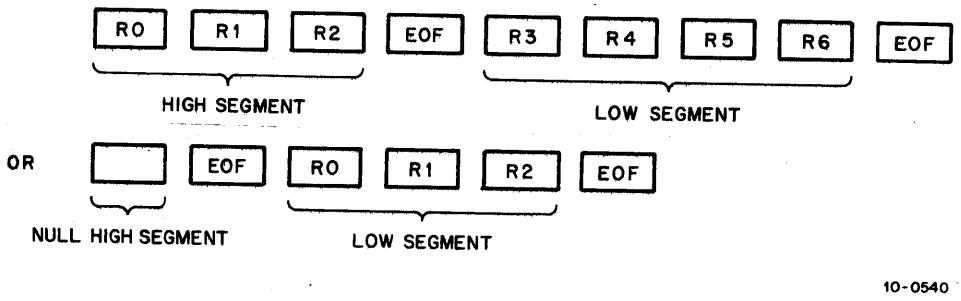
Each IOWD describes the length of the following data block and the original location of the data in core.

Saved files are read into the user's core area starting at location JOBSAV and then are expanded to occupy the original relative locations. If the first word read is not an IOWD and is positive, an old-format, non-compressed saved file is assumed and no expansion is performed.

A SAVE command issued to a magnetic tape writes

a. a high segment (possibly null)
b. an EOF
c. a low segment
d. an EOF.

(continued on next page)


R0
R1
R2
EOF
R3
R4
R5
R6
EOF
HIGH SEGMENT
LOW SEGMENT
OR
EOF
R0
R1
R2
EOF
NULL HIGH SEGMENT
LOW SEGMENT
10-0540

The monitor does not determine the file size of a low segment on a GET from magnetic tape; therefore, a user must always specify a core argument or have enough core assigned to his job for the file.

To save file space, only the high segment up through the highest location (relative to high segment origin) loaded, as specified in the LH of JOBHRL, will be written by the SAVE command. If LH is zero (high segment created by CORE or REMAP UUO) or DDT is present, the entire high segment will be written.

Most programs are written so that only the high segment contains non-zero data. This also saves file space and I/O time with the GET command. SAVE writes the high segment (.HGH) only. The LOADER indicates to the SAVE command that no data was loaded above the job data area in the low segment by setting the LH of JOBCOR to the highest location loaded in the low segment with non-zero data.

A number of locations in the job data area need to be initialized on a GET, although there is no other data in the low segment. The SAVE command copies these locations into the first 10_8 locations of the high segment, provided it is not sharable. The 10 locations are referred to as the vestigial job data area (refer to Paragraph 3.2.2.3). Therefore, the LOADER will load high segment programs starting at location 400010.

To prevent user confusion, SAVE and SSAVE delete a previous file with the extension .SHR or .HGH; therefore, SAVE deletes a file with the extension .SHR and SSAVE deletes a file with the extension .HGH. SAVE and SSAVE commands also delete files with the extension .LOW, if the high segment was the only segment written.

The regular access rights of the saved file indicate whether a user can perform a GET, R, or RUN command. These commands assume that the user wants to execute (but not modify) the high segment, independent of the access rights of the file used to initialize the segment. The monitor always enables the hardware user-mode write protect to prevent the user program from storing into the segment inadvertently.

To debug a reentrant CUSP in the system directory, the user should make a private, nonsharable copy, rather than modify the shared version and possibly cause harm to other users. To make a private, nonsharable copy, the following commands are used:

a.	GET SYS:cusp	
b.	SAVE dev:cusp	Writes a file in the user directory as nonsharable. The high segment in the user's addressing space remains sharable.
c.	GET dev:cusp	Overlays the sharable program with the nonsharable one from the user's directory. Now the user can make patches while other users share the version in the system directory.

A SAVE of a one-segment program and a SSAVE of a two-segment program of the same name can coexist in the same directory, and the monitor keeps the two versions separate. This allows for a common library of reentrant and non-reentrant versions of the same CUSPs to service both the PDP-6 and PDP-10. A sharable program may be superseded into the directory by the SSAVE command. The monitor clears the high segment in its table of storable segments in use but does not remove the segment from the addressing space of users currently using it. Only the users doing a GET, R, or RUN command or a RUN or GETSEG UUO have the new sharable version.

When the SAVE or SSAVE command is used to save a sharable program with only a high file, the monitor does not modify the vestigial job data area unless the user has write privileges to the file that initialized the shared segment. This prohibits unauthorized users from modifying the first 10 locations of a shared segment. This restriction does not exist if a low file is also written, because the GET command reads the low file after the high file. The real job data area locations are set from the low file.

2.9 DETACHED JOB CONTROL COMMANDS

A job is detached if it is not under control of a user console. Any console can initiate any number of detached jobs. Output to the console from a job running in a detached mode causes the job to stop. When the console is attached to the job, the job is continued and the output is done.

PJOB (PJ)

Function

The PJOB command causes the monitor to respond with the job number to which the user's console is attached.

Command Format

PJOB

Characteristics

The PJOB command:

leaves the console in monitor mode,
requires a job number and LOGIN.

Associated Messages

None

Example

```
.PJOB
1

.
```

CSTART (CS)
CCONT (CC)

Function

The CSTART and CCONT commands are identical to the START and CONT commands, respectively, except that the console is left in the monitor mode.

Command Format

CSTART
CCONT

To use:

1. Begin the program with the console in user mode.
2. Type control information to the program, then type ↑C to halt the job with console in monitor mode.
3. Type CCONT to allow job to continue running and leave console in monitor mode.
4. Additional monitor commands can now be entered from the console.

Characteristics

The CSTART and CCONT commands:

leave the console in monitor mode,
require core,
require a job number and LOGIN.

Restrictions

These commands should not be used when the user program (which is continuing to run) is also requesting input from the console.

Associated Messages

?NO CORE ASSIGNED

No core was allocated to the user when the GET command was given, and no core argument was specified in the GET.

?NO START ADR

Starting address was 0 because user failed to specify a starting address in the END statement of the source program.

?CAN'T CONTINUE

The job was halted due to a monitor-detected error and cannot be continued.

?PLEASE TYPE ↑C FIRST

A command which would start a job is issued after a CSTART or CCONT.

Example

```
.CSTART ↵
```

DETACH (DET)

Function

The DETACH command disconnects the console from the user's job without affecting the status of the job. The user console is now free to control another job, either by initiating a new job or attaching to a currently running detached job.

Command Format

DETACH

Characteristics

The DETACH command:

detaches the console,
requires LOGIN.

Associated Messages

FROM JOB n

This is an informative message telling the user the job number to which the console was attached.

Example

```
.DETACH
FROM JOB 1

.
```

ATTACH (AT)

Function

The ATTACH command detaches the current job, if any, and connects the console to a detached job.

Command Format

ATTACH job [proj,prog]

job = the job number of the job to which the console is to be attached. This argument is required.

[proj,prog] = the project-programmer number of the originator of the desired job. This argument may be omitted if it is the same as the job to which the console is currently attached. The operator (device OPR) may always attach to a job although another console is attached, provided he specifies the proper [proj,prog] .

Characteristics

The ATTACH command:

leaves the console in monitor mode.

Associated Messages

If an error message occurs, the console remains attached to its current job.

?TTYn ALREADY ATTACHED

The job number typed is erroneous and is attached to another console, or another user is attached to the job.

?ILLEGAL JOB NUMBER

The specified job number is impossible.

?NOT A JOB

The job number is not assigned to any currently running job.

?CAN'T ATT TO JOB

The project-programmer number entered is not that of the originator of the desired job.

FROM JOB n

An informative message telling the user the job number, if any, from which the console is detaching.

Example

```
.ATT 1[10,63]
FROM JOB 5

.
```

2.10 JOB TERMINATION COMMANDS

When a user leaves the system, all facilities allocated to his jobs must be returned to the monitor facility pool so that they are available to others.

KJOB (K)

Function

In multiprogramming systems, the KJOB command:

Stops all assigned I/O devices and returns them to the monitor pool.

Returns all allocated core to the monitor pool.

Returns the job number to the pool.

Leaves the console in the monitor mode.

Performs an automatic TIME command.

In swapping systems, the KJOB command performs all the above procedures. In addition, if the user has accessed any files, the command responds with

CONFIRM:

The user may type ↑C to abort logout, or type an optional file structure name (or list of file structure names) preceded by one of the following:

F) to logout immediately saving all files (including temporary files) as they are.

D) to delete all files on the specified file structures. Responds with ARE YOU SURE? TYPE Y OR CR.

K) to delete all unpreserved (unprotected) files on the specified file structures.

P) to save and preserve all but temporary files (TMP, CRF, LST) on the specified file structures.

S) to save without preserving all but temporary files on the specified file structures.

L) to list the directories of the specified file structures.

I) to individually determine what to do with all files on the specified file structure as follows:

After each filename is listed, type

P) to preserve the file.

S) to save the file.

K) to delete the file.

Q) to learn if over logged-out quota on this file structure. If not over quota, nothing is typed, and the same filename is repeated.

(continued on next page)

E) to skip to next file structure and save this file if below logged-out quota for this file structure. If not below logged-out quota, a message is typed and the same file-name is repeated.

H) to list responses and meanings.

U) to individually determine what to do with all but preserved files. Preserved files are always preserved.

B) to delete no files except when user is over quota, then delete enough files to be below quota.

Q) to learn if over logged-out quota on the specified file structures.

H) to list the KJOB options and their meanings.

If no file structure names are specified, the responses are for all file structure names in the job search list. If file structure names are specified, the responses apply to those file structures, and CONFIRM is retyped. The KJOB command ignores all logical assignments.

Command Formats

1) KJOB

CONFIRM:

When the CONFIRM: response is given, the user may type any of the above-described letters followed by an optional file structure name or list of file structure names separated by commas.

2) KJOB <file descriptor> ← or = /<letter><list of file structure names >/<letter> <list names > etc.

<file descriptor > = <dev:file.ext [ppn]>

/<letter > = any letter of the above-described letters. In addition, the /Z option is available to Batch jobs. /Z does not perform queuing operations which the Batch user asked to be deferred until LOGOUT time. The letters must appear after the ← or =. If no file descriptor is specified, the default is TTY.

Characteristics

The KJOB command:

detaches the console,
does not allow an active device,
runs the KJOB and LOGOUT CUSPs.

Associated Messages

NO SUCH STR

A nonexistent file structure was specified.

fs LOGGED OUT QUOTA n EXCEEDED BY m BLOCKS

The user's allocation on the file structure named is greater than his logged-out quota.

WAIT PLS

The accounting file FACT.SYS was busy for ten seconds. LOGOUT retries for ten more seconds before going on to FACT.X01.

?<file structure name>UFD READ ERROR, STATUS = n

A read error occurred while reading the user's UFD on the file structure. Status n tells which error occurred.

?SYSSTR FAILURE

SYSSTR UUO gave an error return. Notify the operator.

?STRUUO FAILURE

STRUUO UUO gave an error return. Notify the operator.

ACCOUNTING SYSTEM FAILURE ...

Notify the operator.

?DSKCHR FAILURE ON UNIT u

DSKCHR UUO gave an unexpected error return. Notify the operator.

JOB n USER [p,p] LOGGED OFF TTY n AT hhmm dd-mm-yy
DELETED <ALL>n FILES
SAVED <ALL>n FILES m TOTAL BLOCKS USED
ANOTHER JOB STILL LOGGED IN UNDER [p,p]
RUNTIME n MIN m SEC

This information is typed as user logs off successfully.
Note that m is total blocks allocated as opposed to blocks written.
Therefore, it is always greater than or equal to the number of blocks written.

TYPE H FOR HELP

An unintelligible response or command has been typed. Either the filename or the CONFIRM: message is repeated, depending on what was typed.

Example

```
.K
CONFIRM: I
DSKB:
TEST4  .TST   <055>   2000. BLKS    : K
TEST5  .TST   <055>   505. BLKS     : P
T11    .BAK   <055>   5. BLKS      : K
T2     .BAK   <055>   5. BLKS      : K
T3     .BAK   <055>   5. BLKS      : K
TEST   .BAK   <055>   5. BLKS      : K
TEST   .REL   <055>   5. BLKS      : S
TEST   .MAC   <055>   5. BLKS      : P
TEST   .SHR   <055>   30. BLKS     : S
JOB 5, USER [10,63] LOGGED OFF TTY24 AT 2309    11-SEP-70
DELETED 5 FILES
SAVED 4 FILES  2565 TOTAL BLOCKS USED
RUNTIME 0 MIN, 00.60 SEC
```

2.11 SYSTEM TIMING AND USAGE COMMANDS

All system times are kept in increments of one-sixtieth or one-fiftieth of a second, depending on the line frequency of the power connected to the PDP-10.

DAYTIME (DA)

Function

The DAYTIME command types the date followed by the time of day. The date and time are typed in the following format:

dd-mmm-yy hh:mm:ss.hh

where

dd = day
mmm = month
yy = year
hh = hours
mm = minutes
ss.hh = seconds to nearest hundredth.

Command Format

DAYTIME

Characteristics

The DAYTIME command:

leaves the console in monitor mode.

Associated Messages

None

Example

```
.DAY
11-SEP-70 22:36:34

.
```

SCHED

Function

The SCHED command types out the schedule bits as set by the last SET SCHED command.

Command Format

SCHED

Characteristics

The SCHED command:

leaves the console in monitor mode.

Associated Messages

None

Example

```
.SCHED
000000
```

TIME (TI)

Function

The TIME command causes typeout of the total running time since the last TIME command, followed by the total running time used by the job since it was initialized (logged in), followed by the integrated product of running time and core size (KILO-CORE-SEC=). Time is typed in the following format:

hh:mm:ss.hh

where

hh = hours
mm = minutes
ss.hh = seconds to nearest hundredth.

Interrupt level and job scheduling times are charged to the user who was running when the interrupt or rescheduling occurred.

NOTE

If automatic runtime is enabled using the WATCH command, the incremental runtime is usually 0.

Command Format

TIME job

job = the job number of the job whose timing is desired. If job is omitted, the job to which the console is attached is assumed. In this case, monitor types out the incremental running time (running time since last TIME command) as well as the total running time since the job was initialized.

Characteristics

The TIME command:

leaves the console in monitor mode.

Associated Messages

?ILLEGAL JOB NUMBER

The job number specified is impossible..

Example

```
.TIME )
6.32
6.32
KILO-CORE-SEC=26

.
```

R QUOLST

Function

The R QUOLST command runs the QUOLST CUSP and types the reserved, logged-in and logged-out quotas followed by the number of free blocks left for each file structure in the job search list (refer to Paragraph 6.2.7). In addition, the names of all the file structures in the system are typed followed by the number of free blocks in each file structure that are available to all users.

Command Format

R QUOLST

Characteristics

The R QUOLST command:

leaves the console in monitor mode,
runs the QUOLST CUSP,
requires a job number and LOGIN.

Associated Messages

None

Example

```
.R QUOLST )
YOUR STRUCTURES:
DSKB:  RSRVD = 0  FCFS = 20000  QUOTA OUT = 5000  FREE = 17580
SYSTEM STRUCTURES:
DSKA:  FREE = 831
DSKB:  FREE = 16525
EXIT
.
```

SET WATCH

Function

The SET WATCH command sets the system to print incremental job statistics automatically.

Command Formats

1) SET WATCH arg_1, arg_2,...,arg_5

   prints the specified WATCH statistics.

2) SET WATCH ALL

   prints all the WATCH statistics.

3) SET WATCH NONE

   eliminates the printing of all WATCH statistics.

4) SET WATCH NO arg_1, arg_2,...,arg_5

   eliminates the printing of the specified WATCH statistics.

(continued on next page)

The following argument enables printing whenever a monitor command switches the console from monitor mode to user mode.

arg = DAY prints the time of day, as [HH:MM.SS]

The following arguments enable printing whenever the console is returned to monitor mode via ↑C, EXIT, HALT, ERROR IN JOB n, DEVICE xxx OK ?

arg = RUN prints the incremental run time.

arg = WAIT prints the wait time (time since the user started or continued the program).

arg = READ prints the incremental number of disk blocks read modulo 4096.

arg = WRITE prints the incremental number of disk blocks written modulo 4096.

Any combination of the arguments may be specified in any order. Statistics are not printed for commands that do not run programs, such as ASSIGN or PJOB. When a user logs in, his job is set to WATCH the statistics that he has notified the system manager of. These statistics are kept in ACCT.SYS.

The order of the error message is the same as the order of output. Therefore, a user who forgets either the arguments or the significance of the statistics can find these out. A single space is always typed between each statistic, whether the statistic appears or not; therefore, it is possible to tell which statistics are being typed.

NOTE

Enabling WATCH output interacts with the incremental data typed by the TIME and DSK commands.

Characteristics

The SET WATCH command:

leaves the console in monitor mode.

Associated Messages

?ARGS ARE: DAY, RUN, WAIT, READ, WRITE

The user typed an illegal argument.

Example

```
.SET WATCH DAY RUN WAIT READ WRITE

.
.↑C

.R PIP
[22:38:19]
*↑C
[0.10 2.95 457 243]
```

```
.R PIP
[22:38:37]
*LPT:←SYS:PARIO.SCP
*↑C
[0.17 22.43 6 0]

.SET WATCH H
?ARGS ARE: DAY,RUN,WAIT,READ,WRITE
```

SYSTAT (SYS)

Function

The SYSTAT command runs a CUSP which prints status information about the system. This information allows a user to determine the load on the system before logging-in.

To write the output on the disk as a file with name SYSTAT.TXT, assign device DSK with logical name SYSTAT.

The SYSTAT command types the status of the system: system name, time of day, date, uptime, percent null time (idle plus lost time).

It types status of each job logged-in: job number (@ after job number indicates the high segment has been superseded; # after the job number indicates the high segment is from a directory or device other than the CUSP directory on device SYS; ! after the job number indicates the job is locked in core but is shufflable; & after the job number indicates the job is locked in core and not shufflable); project-programmer number (**,** if detached); Teletype number (CTY = console Teletype, DET = detached); program name being run; program size; job and swapped state (refer to Paragraph 4.9.3.3); and run time since logged in.

It types the status of high segments being used: name (PRIV = nonsharable, OBS = superseded); device or file structure name from which the segment came; directory name (**,** if detached); size (SW = swapped out, SWF = swapped out and fragmented, F = in core and fragmented on disk, SPY = user is executing the SPY UUO); number of users in core or on the disk.

It types the status of dormant segments: name, device name, directory name, size (SW = swapped out, SWF = swapped out and fragmented, F = in core and fragmented on disk).

The command types swapping space used, virtual core used, swapping ratio, virtual core saved by sharing.

It types status of busy devices: device name, job number, how device is assigned (AS = ASSIGN command, INIT = INIT or OPEN UUO, AS+INIT = both ways).

It types system file structures: free blocks, mount count, single-access job. It types disk performance, swapping, and error statistics: free blocks for each file structure followed by the following items, in columns, for each unit: number of free blocks (FREE), number of buffered-mode blocks read (BR) and written (BW), number of dump-mode blocks read (DR) and written (DW), number of blocks read (MR) and written (MW) for monitor I/O (UFD, RIB, MFD), and number of seeks for any I/O.

It types the status bits for each unit as follows: RHB, monitor must reread HOME block; OFL, unit is off-line; HWP, unit is hardware write-protected; SWP, unit is software write-protected for this job; SAF, unit is member of single-access file

structure; ZMT, unit is member of a file structure with zero mount count; PRF, unused by monitor; PNM, pack not mounted on this unit; DWN, unit is down; MSB, unit has more than 1 SAT block; NNA, no new accesses have been specified by operator.

It types the following error information: HDEV (number of hard device, channel and controller, errors on this unit); HDAT (number of hard data, parity, errors on this unit); SDEV (number of soft and hard device, channel and controller, errors on this unit); SDAT (number of soft and hard data errors, parity, on this unit); HPOS (number of hard positioning failures, recalibrating did not correct, for this unit); SPOS (number of soft positioning failures, recalibrating once corrected, for this unit); SER (number of SAT failures); RER (number of rib redundance failures); CER (number of software folded checksum failures); HERR STATUS (last device status on hard device or data errors, in octal); SERR STATUS (last device status on a soft device or data error, in octal); LBN (last logical block number in octal of region which had the latest hard data error).

It types the following data for each unit in the active swapping list: physical unit name, number of blocks (128 word) swapped in (R) and out (W), ratio and percent of number of K used and allocated for swapping.

Command Format

SYSTAT arg

arg = one or more single letters (in any order) used to type any subset of the SYSTAT output. This argument is optional. The letters are as follows:

- B = busy devices
- D = dormant segments
- F = file structures
- H = options available
- J = job information
- N = all but job information
- S = job information without state and run time
- L = output on device LPT. If LPT is unavailable, the message LPT BUSY - WAITING is typed, and every 5 seconds the LPT is tried again.
- P = disk performance

Characteristics

The SYSTAT command:

leaves the console in monitor mode,
runs the SYSTAT CUSP,
does not require LOGIN.

Associated Messages

None

Example

```
.SYSTAT)

STATUS OF 5@2@6A SYS#2 AT 16:@5:44 ON 17-DEC-7@

UPTIME 4@:35, 64% NULL TIME = 23% IDLE + 41% LOST
SHUFFLE TIME = 3@.42,  CORE ZEROING TIME = 2.15
22 JOBS IN USE OUT OF 37.  2@ LOGGED IN, 22 DETACHED

JOB     WHO     WHERE   WHAT    SIZE    STATE   RUN TIME

1       **,**   DET     PRINT   1K      TI SW       1:29
2       **,**   DET     OPSER   1+2K    SL SW          3
3       **,**   DET     PIP     1+4K    ↑C SW         43
4       **,**   DET     BATCON  6K      SL SW         15
5       **,**   DET     TECO    3+3K    TI SW         33
6       **,**   DET     OMOUNT  9K      SL SW          4
7       **,**   DET     SYSTAT  3K      RN             @
8       **,**   DET     OMOUNT  9K      TI SW          @
9       **,**   DET     LOGIN   1+2K    RN             @
1@      **,**   DET     SYSDPY  3+SPY   TI          4:4@
11      **,**   DET     PRINTR  2K      IO             7
12      **,**   DET     DBASIC  2@+7K   RN SW       1:2@
13      **,**   DET     DIRECT  1+2K    ↑C SW          @
14      **,**   DET     LPTSPL  2K      SL SW          4
15      **,**   DET     CDRSTK  4K      ↑C SW          9
16      **,**   DET     PIP     1+4K    ↑C SW          2
18      **,**   DET     KJOB    4+2K    TI SW          8
19      **,**   DET     PIP     1+4K    TI SW       1:12
2@      **,**   DET     DIRECT  1+2K    ↑C SW          1
21      **,**   DET     KJOB    4+2K    TI SW          4
22      **,**   DET     FAI31G  9K      ↑C SW       1:18
23      **,**   DET     MACRO   4+6K    RN SW         38

HIGH SEGMENTS:
PROGRAM DEVICE  OWNER   HIGH K  USERS

OPSER   DSKB    SYS     2K SW   1
LOGIN   DSKB    SYS     2K      1
PIP     DSKB    SYS     4K SW   3
DIRECT  DSKB    SYS     2K SW   2
(PRIV)          JOB 12  7K SW   1
TECO    DSKB    SYS     3K SW   1
MACRO   DSKB    SYS     6K SW   1
KJOB    DSKB    SYS     2K SW   2

DORMANT SEGMENTS:
PROGRAM DEVICE  OWNER   HIGH K

QMANGR  DSKB    SYS     1K SW
TECO    DSKB    12,141  3K SW
COMPIL  DSKB    SYS     2K SW
LPTSPL  DSKB    SYS     2K SW
LPTSPL  DSKB    1,2     2K SW
CDRSTK  DSKB    SYS     2K SW
LOADER  DSKB    SYS     3K SW
BINCOM  DSKB    11,131  1K SW
FED     DSKB    SYS     4K SW
F4@     DSKB    SYS     1@K SW
SRCCOM  DSKB    SYS     1K SW
RUNOFF  DSKB    SYS     2K SW
LOGOUT  DSKB    SYS     2K SW
```

(continued on next page)

```
SWAPPING SPACE USED = 144/375 = 38%
VIRT. CORE USED = 118/375 = 31%
SWAPPING RATIO = 118/27 = 4.4
VIRT. CORE SAVED BY SHARING = 12/(12+118) = 9%

BUSY DEVICES:
DEVICE  JOB     WHY

PTY0    2       INIT
PTY1    2       INIT
PTY3    2       INIT
PTY4    2       INIT
PTY5    2       INIT
PTY6    2       INIT
PTY10   2       INIT
DTA1    19      AS
DTA2    22      AS
DTA3    23      AS
DTA6    1       AS
DTA7    6       AS
MTA0    9       AS+INIT

SYSTEM FILE STRUCTURES:
NAME    FREE    MOUNT
DSKA    4631    9
DSKB    12269   19


DISK PERFORMANCE STATISTICS:
UNIT OR F/S
        FREE    BR      BW      DR      DW      MR      MW      SEEKS
DSKA    4628
FHA0(GLZX):
        1447    518     0       317     0       331     69      0
FHA1(GLZX2):
        3181    121     1       328     16      543     110     0
DSKB    12269
DPA0(2RP001):
        6114    2545    929     630     219     5155    1256    3634
 MSB  ERRORS: CER:13 HERR STATUS:15 SERR STATUS:400015
DPA1(2RP005):
        6125    2443    1474    1860    282     1116    547     2346
 MSB  ERRORS: SDAT:2 HERR STATUS:15 SERR STATUS:500400015

ACTIVE SWAPPING STATISTICS:
UNIT    R       W       USED(K)
FHA0    274656  194616  140/300 = 47%
DPA0    0       0       0/75 = 0%
```

DSK (DS)

Function

The DSK command types disk usage for the combined structures of the job, since the last DSK command, followed by the total disk usage since the job was initialized (logged in). Disk usage is typed in the following format:

RD,WT=I,J
RD,WT=M,N

where I and J are the incremental number of 128-word blocks read and written since the last DSK command, and M and N are the total number of 128-word blocks read and written since LOGIN.

NOTE

I and J are kept modulo 4096. If automatic READ or WRITE print outs have been enabled using the WATCH command, I and J are usually zero, since the WATCH output also resets these values.

Command Format

DSK job

job = the job number of the job for which the disk usage is desired. This argument is optional.

If job is omitted, the job to which the console is attached is assumed.

If job is supplied (whether the job of this user or another user) the incremental quantities are not reset to zero.

Characteristics

The DSK command:

leaves the console in monitor mode,
requires a job number and LOGIN.

Associated Messages

?NOT A JOB NUMBER

The job number specified is not assigned to any currently running job.

Example

```
.DSK )
RD,WT=12,0
RD,WT=475,243
```

2.12 TELETYPE CHARACTERISTICS COMMAND

The SET TTY command accepts text arguments and modifies the monitor table of characteristics for a Teletype line. This command also allows a Teletype to be assigned by a job which is not controlling it.

SET TTY

Function

The SET TTY command declares special properties of the Teletype line to the scanner service.

Command Format

SET TTY dev: NO word

dev: = the device argument that is used to control a line other than the one where the command is typed. This argument is optional and is legal only from the operator's console. It may be used to modify the characteristics of any Teletype lines in the system.

NO = the argument that determines whether a bit is to be set or cleared. This argument is optional.

word = the various words representing bits that may be modified by this command. The words are as follows:

SET TTY TAB	This terminal has hardware TAB stops set every eight columns.
SET TTY NO TAB	The monitor simulates TAB output from programs by sending the necessary number of SPACE characters.
SET TTY FORM	This terminal has hardware FORM (PAGE) and VT (vertical tab) characters.
SET TTY NO FORM	The monitor sends eight linefeeds for a FORM and four linefeeds for a VT.
SET TTY LC	The translation of lower-case characters input to upper case is suppressed.
SET TTY NO LC	The monitor translates lower-case characters to upper case as they are received. In either case, the echo sent back matches the case of the characters being sent.
SET TTY WIDTH n	The carriage width (the point at which a free carriage return is inserted) is set to n. The range of n is 17 (two TAB stops) to 200 decimal.
SET TTY NO CRLF	The carriage return normally outputted at the end of a line exceeding the carriage width is suppressed.
SET TTY CRLF	Restores the carriage return.

(continued on next page)

SET TTY SLAVE	The Teletype becomes slaved, i.e., no commands may be typed on the console, and the console may be ASSIGNed by another user.
SET TTY NO ECHO	The Teletype line has local copy and the computer should not echo characters typed in.
SET TTY ECHO	Restores the normal echoing of each character typed in.
SET TTY FILL n	The filler class n is assigned to this terminal. The filler character is always DEL (RUBOUT, 377 octal). No fillers are supplied for image mode output.
SET TTY NO FILL	Equivalent to TTY FILL 0. Fillers for output and echoing are determined from the following:

Character Name	Octal	Number of Fillers for Filler Class			
		0	1	2	3
BS	010	0	2	6	6
HT	011	0	1 or 2	1 or 2	1 or 2†
LF	012	0	1	6	6
VT	013	0	2	6	6
FF	014	0	12	21	21
XON	021	0	1	1	1
TAPE	022	0	1	1	1
XOFF	023	0	1	1	1
NTAP	024	0	1	1	1

Characteristics

The SET TTY command:

leaves the console in monitor mode.

Associated Messages

None

2.13 SYSTEM ADMINISTRATION COMMANDS

The commands in this section are restricted to system administrators only.

†1 if 0-3 spaces to TAB stop, 2 if 4-7 spaces to TAB stop.

SET DAYTIME

Function

The SET DAYTIME command when used with an argument changes the time of day.

Command Format

SET DAYTIME n

n = decimal number 0 through 2359, representing 24-hour time (i.e., hours * 100 + minutes). This argument is required.

Characteristics

The SET DAYTIME command:

leaves the console in monitor mode.

Restrictions

The user must be on device OPR or be logged in under [1,2].

Associated Messages

None

SET SCHEDULE

Function

The SET SCHEDULE command changes the scheduled use of the system, depending on n.

Command Format

SET SCHEDULE n

n is octal and is stored in RH of STATES word in COMMON.

n = 0	regular timesharing
n = 1	no further LOGINS allowed except from CTY.
n = 2	no further LOGINS from remote Teletype's, and do not answer data sets.

Characteristics

The SET SCHEDULE command:

leaves the console in monitor mode.

Restrictions

The user must be on device OPR or be logged in under [1,2].

Associated Messages

None

ASSIGN SYS:

Function

The ASSIGN SYS command changes the systems device to device dev.

Command Format

ASSIGN SYS:dev

dev = the device to which the system device is changed.

Characteristics

The ASSIGN SYS command:

leaves the console in monitor mode,
requires a job number and LOGIN.

Restrictions

The user must be on device OPR or logged in under [1,2].

Associated Messages

None

DETACH (DET)

Function

The DETACH command assigns the device dev to JOB 0, thus making it unavailable.

Command Format

DETACH dev

dev = the name of the device to be detached.

Characteristics

The DETACH command:

leaves the console in monitor mode,
requires a job number and LOGIN.

Restrictions

The user must be on device OPR or be logged in under [1,2]. DSK cannot be detached.

Associated Messages

?ALREADY ASSIGNED TO JOB n

The device specified is already in use.

?CAN'T DET DEV

The user is not logged-in under [1,2].

?NO SUCH DEVICE

The specified device does not exist in this monitor configuration.

ATTACH (AT)

Function

The ATTACH command returns a detached device to the user issuing the command, and then the user must DEASSIGN the device to return it to the monitor's pool of available resources.

Command Format

ATTACH dev

dev = the device to which the user is attaching.

Characteristics

The ATTACH command:

leaves the console in monitor mode,
requires a job number and LOGIN.

Restrictions

The user must be on device OPR or be logged in under [1,2].

Associated Messages

?CAN'T ATT DEV

The user is not logged-in under [1,2].

?NO SUCH DEVICE

The specified device does not exist in this monitor configuration.

?WASN'T DET

The specified device is not detached.

CTEST

Function

The CTEST command is used by system programmers to pass arguments to test extensions made to the COMPIL CUSP.

Command Format

CTEST

Characteristics

The CTEST command:

runs the COMPIL CUSP,
requires LOGIN.

Associated Messages

None

SET DATE

Function

The SET DATE command is used to change the date.

Command Format

SET DATE mm dd yy

mm = two-character number of month.
dd = two-digit day of month.
yy = two-digit year.

This command does not check the validity of the argument as does ONCE-only.

Characteristics

The SET DATE command:

leaves the console in monitor mode.

Restrictions

The user must be on device OPR or be logged in under [1,2].

Associated Messages

None

SET CORMAX

Function

The SET CORMAX command is used to change the system parameter CORMAX. CORMAX is the largest size that any job can be.

Command Format

SET CORMAX n

n = decimal number representing nK. This argument is required.

Characteristics

The SET CORMAX command:

leaves the console in monitor mode.

Restrictions

The user must be on device OPR or be logged in under [1,2].

Associated Messages

None

SET CORMIN

Function

The SET CORMIN command is used to change the system parameter CORMIN. CORMIN is the guaranteed amount of contiguous core that a single unlocked job can have. This command is used only with the real-time monitor.

Command Format

SET CORMIN n

n = decimal number representing nK. This argument is required.

Characteristics

The SET CORMIN command:

leaves the console in monitor mode.

Restrictions

The user must be on device OPR or be logged in under [1,2].

Associated Messages

None

SET TIME

Function

The SET TIME command sets a central processor time limit for a job. When the time limit is reached, the job is stopped and a message is typed. A timesharing job may be continued by typing CONT, but no time limit is in effect unless it is reset. A Batch job cannot be continued.

Command Format

SET TIME n

n = number of seconds of central processor time to which the job is limited.

Characteristics

The SET TIME command:

leaves the console in monitor mode,
requires LOGIN.

Associated Messages

?TIME LIMIT EXCEEDED

The time allowed for the job has been reached.

Examples

```
.SET TIME 10
.SET TIME 6
```

Chapter 3
Loading User Programs

3.1 MEMORY PROTECTION AND RELOCATION

Each user program is run with the processor in a special mode known as the user mode, in which the program must operate within an assigned area in core. In user mode, certain operations are illegal. Every user has an assigned area in core; therefore the rest of core is unavailable to him. He cannot gain access to the protected area for either storage or retrieval of information.

The assigned area of each user may be divided into two segments. If this is the case, the low segment is unique for a given user and can be used for any purpose. The high segment may be used by a single user or it may be shared by many users. If the high segment is shared by other users, the program is a reentrant program. The monitor can write-protect 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 may be used with any number of low impure segments. (Refer to Chapter 1 for the distinctions between pure and impure segments.) Any user program which 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 4.4.2).

The Timesharing monitor defines the size and position of a user's area by specifying protection and relocation addresses for the low and high segment. 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 PDP-10 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.

In user mode, the PDP-10 hardware automatically relocates user addresses by adding the contents of the memory relocation register in the central processor to the high-order eight bits of the user address before the address is sent to memory. The address before the addition is the relative address and after the addition is the absolute address. To determine whether a relative address is legal, its eight

high-order bits are compared with the contents of the memory protection register. If the eight high-order bits of the relative address are greater than the contents of the memory protection register, the memory protection flag is set in the central processor, and control traps to the monitor, which aborts the user program and prints an error message on the user's console, unless the user program has instructed the monitor to pass such interrupts to itself for error-handling. (Refer to APRENB UUO, Paragraph 4.3.3.1.)

Systems with the KT10 option have only the low pair of protection and relocation registers. The user program is always non-reentrant and the assigned area comprises only the low segment.

When the monitor schedules a user's program to run, the memory protection and relocation registers are set to the bounds of the user's allocated core area and the central processor is switched to user mode.

To take advantage of the fast accumulators, memory addresses $0\text{-}17_8$ are not relocated and all users have access to the accumulators. Therefore, relative locations $0\text{-}17_8$ 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 protection hardware.

3.1.1 Memory Parity Error Recovery

The memory parity error recovery code allows the machine to run with PARITY STOP up, thereby gaining 10% more CPU speed than with PARITY STOP enabled. This procedure differentiates between user mode and executive mode when a parity error occurs. If the machine was in user mode, the current job is stopped, and the word causing bad parity is rewritten with good parity. The following message is typed on the user's console:

```
? ERROR IN JOB n
? MEM PAR AT USER pppppp; BAD WORD ddddddddddd
    AT USER adr = ABS. xxxxxx
```

and the following message is typed simultaneously on device OPR, interrupting any current typeout:

```
? USER MODE PAR ERROR AT ABS LOC xxxxxx FOR JOB n
```

where

n is the job number of the current job
pppppp is the user PC when the parity error occurred
ddddddddddd is the bad data word read (expressed in octal)
adr is the user address of the bad word
xxxxxx is the absolute address of the bad word

If the machine was in executive mode when the parity error occurred, recovery is not attempted since a monitor routine has read bad data. The following message (with no carriage return, line feed following) is typed on CTY and the machine HALTS:

EXEC PARITY ERROR STOP

At this point, the operator must depress the PARITY STOP key and hit CONTINUE. The machine should stop almost immediately with a memory failure. If the parity error is not reproducible on a memory scan, the following message is typed on the CTY on the same line as the previous message and the machine HALTS with the PC at 777777:

----SPURIOUS

System reload is required after an executive mode memory parity failure.

The algorithm for determining the bad memory location in both executive and user mode is to scan core from location 20_8 through location C (MEMSIZ). In both modes, the parity error is detected at APR interrupt level. For executive mode the memory scan when CONTINUE is hit runs at APR level. For user mode a clock level interrupt is requested, and the memory scan and subsequent typeouts are processed at this level. The following two counters are kept for user mode parity analysis:

PARTOT - the total number of user mode parity errors since system was loaded.

PARSPR - the number of errors for which recovery failed (no parity error on memory scan) and the job was not stopped.

Also the counters PARPC, PARADR, and PARWRD contain the user PC, the absolute location, and the bad data word, respectively, of the most recent user mode memory parity error.

3.2 USER'S CORE STORAGE

A user's core storage consists of blocks of memory, the sizes of which are an integral multiple of 1024_{10} (2000_8) words. In a non-reentrant monitor, the user's core storage is a single contiguous block of memory. After relocation, the first address in a block is a multiple of 2000_8. The relative user and relocated address configurations are shown in Figure 3-1, where P_L, R_L, P_H, and R_H are the protection and relocation addresses, respectively, for the low and high segments as derived from the 8-bit registers loaded by the monitor. 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.

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, Chapter 2). The other method is to use the relocatable binary loader to link-load binary files. The user may then write the core image on a retrievable device for future use (refer to SAVE, Chapter 2).

3.2.1 Job Data Area

The first 140 octal locations of the user's core area are always allocated to the job data area (refer to Table 3-1). Location in this area are given mnemonic assignments where the first three characters are JOB. The job data area provides storage for specific information of interest to both the monitor and the user. Some locations, such as JOBSA and JOBDDT, are set by the user's program for use by the monitor. Other locations, such as JOBREL, are set by the monitor and are used by the user's program. In particular, the right half of JOBREL contains the highest legal address set by the monitor when the user's core allocation changes.

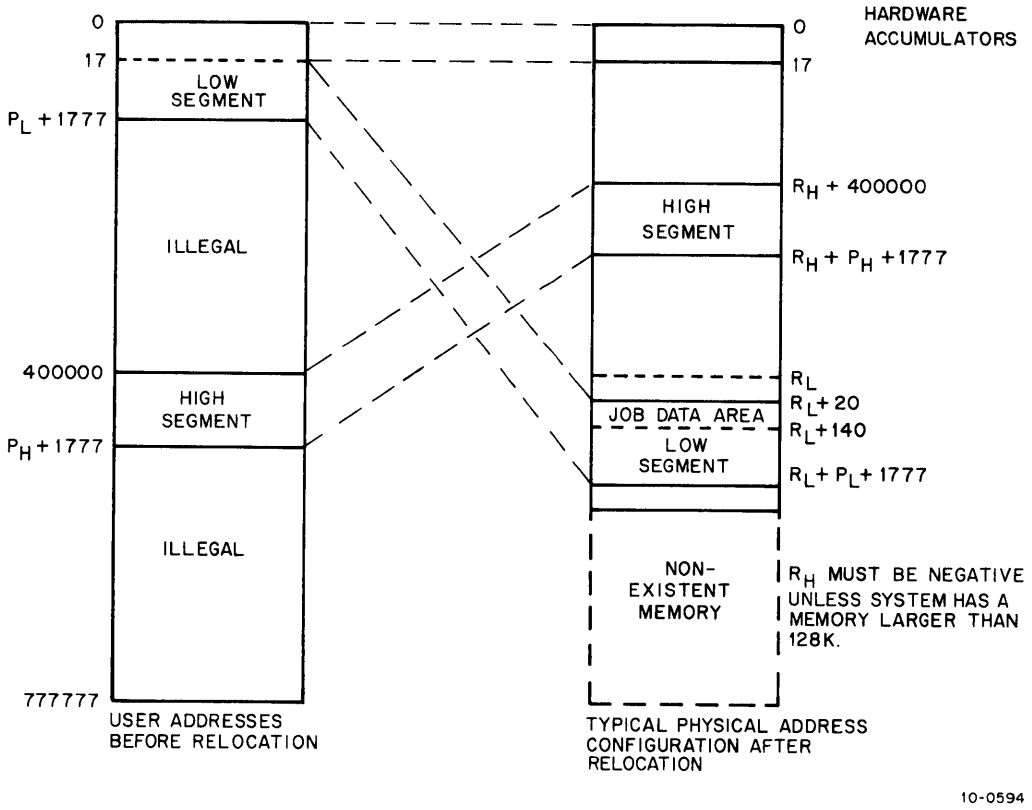


Figure 3-1 User's Core Area

Table 3-1
Job Data Area Locations
(for user-program reference)

Name	Octal Location	Description
JOBUUO	40	User's location 40_8. Used for processing user UUOs (001 through 037) and storing op code and effective address.
JOB41	41	User's location 41_8. Contains the beginning address of the user's programmed operator service routine (usually a JSR or PUSHJ).
JOBERR	42	Left half: Unused. Right half: Accumulated error count from one CUSP to the next. CUSPs should be written to look at the right half only.
JOBREL	44	Left half: 0 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).
JOBBLT	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.
JOBDDT	74	Left half: the last address of DDT. Right half: the starting address of DDT. If contents are 0, DDT has not been loaded.
JOBCN6	106	Six temporary locations used by CHAIN CUSP (refer to Timesharing Handbook) after it releases all I/O channels. JOBCN6 is defined to be in JOBJDA.
JOBPFI	114 (value)	All user I/O must be to locations greater than JOBPFI.
JOBHRL	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 $\geq$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 3-1 (Cont)
Job Data Area Locations
(for user-program reference)

Name	Octal Location	Description
JOBSYM	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.
JOBUSY	117	Contains a pointer to the undefined symbol table created by linking loader. Not used by DDT.
JOBSA	120	Left half: First free location in low segment (set by loader). Right half: Starting address of the user's program.
JOBFF	121	Left half: 0. Right half: Address of the first free location following the low segment. Set to $C(JOBSA)_{LH}$ by RESET UUO.
JOBREN	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.
JOBAPR	125	Left half: 0. 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 CALL APRENB UUO.
JOBCNI	126	Contains state of APR as stored by CONI APR when a user-enable APR trap occurs.
JOBTPC	127	Monitor stores PC of next instruction to be executed when a user-enabled APR trap occurs.
JOBOPC	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, JOBOPC 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, JOBOPC points to the address of the UUO. For example, in DDT to continue after a HALT, type JOBOPC/10000,,3010 JRST @ QX

Table 3-1 (Cont)
Job Data Area Locations
(for user-program reference)

Name	Octal Location	Description
JOBCHN	131	Left half: 0 or the address of first location after first FORTRAN IV loaded program. Right half: Address of first location after first FORTRAN IV Block Data.
JOBCOR	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.
JOBINT	134	Left half: Reserved for the future. Right half: 0 or the address of the error-intercepting block (refer to Paragraph 4.3.3.2).
JOBVER	137	Left half: 0 or the patch number of the installation that made the last modification to the program. Right half: Program version number in octal. The number is never converted to decimal. After a GET, R, or RUN command, an E command can be used to find the version number. (DEC always distributes CUSPs with the left half = 0, so customers making modifications to CUSPs should change only the left half. The right half will remain as a record of the DEC version.)
JOBDA	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 which are used by the monitor and those which are unused at present. User programs should not refer to any locations not listed above since 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 JOB prefix refer to locations in the job data area.

3.2.2 Loading Relocatable Binary Files

The relocatable binary loader (LOADER), which resides in the system file, is started by the command

R LOADER core

where core is an optional argument (see Figure 3-2). (Refer to the LOADER documentation in the PDP-10 Reference Handbook for a description of the loader command string.)

In writing reentrant user software, an effort is made to minimize the support required to run such software on a machine having only a single relocation register. Both the source and relocatable binary files are the same for a reentrant program that must run on a non-reentrant system.

The loader is reentrant; therefore, its instructions exist in the high segment. In loading two segments, both segments are data with respect to the loader and must exist in the low segment during load time. Therefore, the following loader variables must exist for each segment:

a. Offset (the number of locations a program must be moved toward zero before it can be executed)

b. Program origin (the location assigned by the loader to relocatable zero of a program)

c. Location counter (the register that indicates the location of the next instruction to be interpreted).

3.2.2.1 H Switch - A program written to be reentrant can be loaded into one segment instead of two by use of the H switch (/H). The H switch is used only when a two-segment program is to be loaded into one segment. It causes all following files to be loaded into the low segment. This switch is not required when a one-segment program is to be loaded into one segment.

To minimize the use of the H switch on single-register machines, the loader checks if the system (i.e., hardware plus software) has a two-segment capability. If the monitor has this capability but the machine does not, then the system does not have the two-segment capability. If the system does not have the two-segment capability, the loader automatically loads a two-segment program into one segment, just as if the user had typed the H switch.

To find out if the system has a two-segment capability, the loader uses the SETUWP UUO and attempts to set the user mode write-protect bit to 1. An error return indicates a single-register capability. The loader cannot produce a two-segment program, and the monitor cannot save a program as two segments.

If a user wants to load a program in which the low segment is longer than 400000 octal words, he uses the switch NNNNNNH, which changes the origin of the high segment from its initial setting of 400,000 to NNNNNN where NNNNNN is larger. If NNNNNN is missing, the loader loads everything into the low segment.

It is not known before load time whether a reentrant program is not going into the high segment; therefore, the core executed (including the monitor UUOs) is the same for either case.

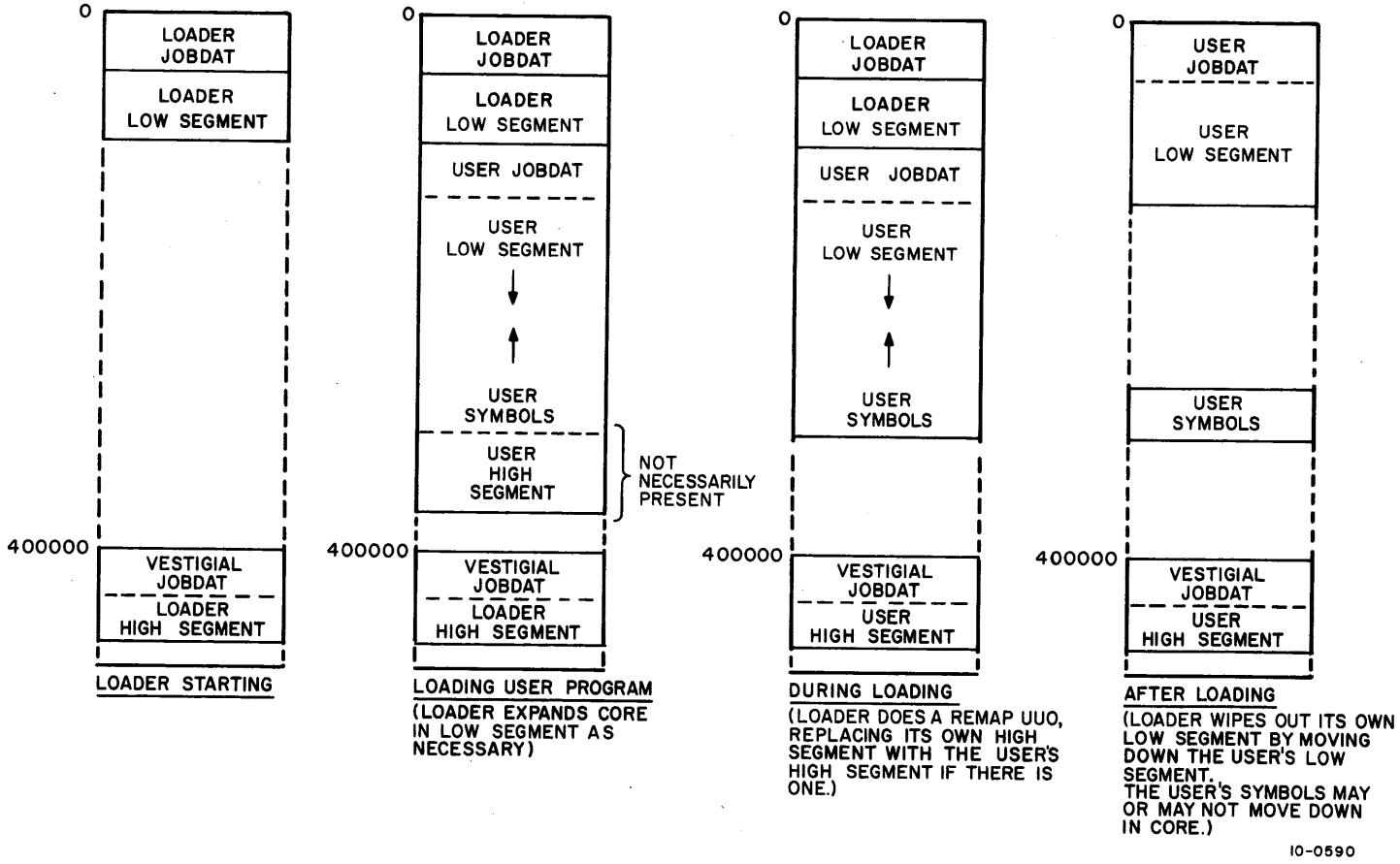


Figure 3-2 Loading User Core Area

Modifications to the H switch

a. Cause all following files to be loaded into the high-segment (/1H) and

b. Reset the loader to load high segment code in the high segment and low segment code into the low segment (/-H).

3.2.2.2 HISEG Pseudo-Op - After loading, a relocatable subprogram assembled by MACRO is put entirely in either the user low segment or the user high segment. To indicate that a subprogram is to be loaded into the high segment, the HISEG pseudo-op is used. Although the HISEG pseudo-op can appear anywhere in the program, the best position is at the beginning, because a reader wants to know that the program is destined for the high segment. Near the beginning of the binary output, MACRO generates code that tells the loader to load subprograms into the high segment. Loader Version 50 loads programs in any order. In earlier versions of the loader, all programs for the low segment must be loaded before any programs for the high segment.

3.2.2.3 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 (JOB41, JOBREN, JOBVER) and some locations are loaded by the monitor on a GET (JOBSA, JOBCOR JOBHRL). 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 3-2). 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 JOBCOR is 140_8 or greater. The vestigial job data area locations are referenced by the monitor only, not by user programs.

Table 3-2
Vestigial Job Data Area Locations

Symbol	Octal Location†	Description
JOBHSA	0	A copy of JOBSA
JOBH41	1	A copy of JOB41
JOBHCR	2	A copy of JOBCOR
JOBHRN	3	LH: restores the LH of JOBHRL RH: restores the RH of JOBREN

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

Table 3-2 (Cont)
Vestigial Job Data Area Locations

Symbol	Octal Location†	Description
JOBHVR	4	A copy of JOBVER
	5	
	6	Reserved for future use
	7	
JOBHDA	10	First location not used by vestigial job data area.

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

3.2.2.4 Completion of Loading - The new program code is loaded upward from an offset above the resident loader. The program origin (i.e., the first location loaded) is 140_8, unless the user changes the origin by the assembler LOC pseudo-instruction. After completion of the loading but before exiting, the loader does the following:

a. Sets the LH of JOBSA and the RH of JOBFF to the address of the first location above the new code area (i.e., the program break). The RH of JOBSA is set to the program starting address. This value is the last non-zero address of the assembler END pseudo-instruction to be loaded, or zero. It is used by the RUN and START commands. The LH of JOBFF is zero.

b. Sets the LH of JOBHRL to the new highest relative user address (relative to the high segment origin) in high segment, or zero if no high segment.

c. Sets the LH of JOBCOR to the highest location in the low segment that is loaded with non-zero data.

d. Uses REMAP UUO to take the top part of the low segment that contains the user's high segment, and replaces the loader high segment.

e. May move symbols and reduce core, if DDT was loaded.

f. Calls EXIT or starts up program.

If DDT was loaded by the D switch in the loader command string, the RH of JOBDDT is set by the loader to the starting address of DDT and the LH is the last address of DDT. A new switch, /K, implemented for use with DDT, moves core back to the absolute minimum needed. A /nK moves core back to nK, unless n is less than the minimum core, in which case the minimum core is assigned. The /D switch is used to imply /B/K.

Chapter 4
User Programming

4.1 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.

4.1.1 User Mode

User programs run with the processor in user mode 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 relocation (refer to Chapter 3)

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.

4.1.2 User I/O Mode

The user I/O mode (bits 5 and 6 of PC word = 11) of the central processor allows running privileged user programs with automatic protection and relocation 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 running infrequently used device service routines as a user job. Direct control by the user program of special devices is particularly important in real-time applications.

To utilize this mode, the job number must be 1. CALL [SIXBIT /RESET/] 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 4.10.10.1).

4.1.3 Executive Mode

The monitor operates with the processor in executive mode, which is characterized by the lack of memory protection and relocation (refer to Chapter 3) 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. This chapter describes the services the monitor makes available to user-mode programs and how a user program obtains these services.

4.2 PROGRAMMED OPERATORS (UUOs)

Operation codes 000 through 077 in the PDP-10 are programmed operators, sometimes referred to as UUOs (Unimplemented User Operators) 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 in user or monitor location 40 and the instruction in user or monitor location 41 is executed out of normal sequence. Location 41 must contain a JSR instruction to a routine to interpret the contents of location 40.

4.2.1 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:

ERROR IN JOB n
ILLEGAL UUO AT USER 41

This message is given because the user's relative location 41 contains zero (unless his program has overtly changed it) and 000 is an illegal monitor UUO.

4.2.2 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:

ERROR IN JOB n
ILLEGAL UUO AT USER addr

Table 4-1 lists the operation codes 040 through 077 and their mnemonics. Most of Chapter 4 is a detailed description of their operation.

Table 4-1
Monitor Programmed Operators

Op Code	Call	Function
040	CALL AC, [SIXBIT/NAME/]	Programmed operator extension (refer to Paragraph 4.2.2.1)
041	INIT D, MODE SIXBIT /DEV/ XWD OBUF, IBUF error return normal return	Select I/O device (refer to Paragraph 4.10.2.3)
042		No operation — Reserved for installation-dependent definition
043		No operation
044		No operation
045		No operation
046		No operation

Table 4-1 (Cont)
Monitor Programmed Operators

Op Code	Call	Function
047	CALLI AC, N	Programmed operator extension (refer to Paragraph 4.2.2.1)
050	OPEN, D, E error return normal return E: EXP STATUS SIXBIT /DEV/ XWD OBUF, IBUF	Select I/O device (refer to Paragraph 4.10.2.3)
051	TTCALL AC, ADR	Extended operations on job-controlling Teletype (refer to Paragraph 5.9.3)
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 4.10.4.3)
056	IN D, normal return error or EOF return	INPUT and skip on error or EOF. (See Section 4.10.5)
057	OUT D, normal return error return	OUTPUT and skip on error or EOT. (See Section 4.10.5)
060	SETSTS D, STATUS	Set file status. (See Section 4.10.6.2)
061	STATO D, BITS R0: NO SELECTED BITS = 1 R1: SOME SELECTED BITS = 1	Skip if file status bits = 1. (See Section 4.10.6.1)
062	GETSTS D, E	Copy file status to E. (See Section 4.10.6.1)
063	STATZ D, BITS R0: SOME SELECTED BITS = 1 R1: ALL SELECTED BITS = 0	Skip if file status bits = 0. (See Section 4.10.6.1)
064	INBUF D, N	Set up input buffer ring with N buffers (refer to Paragraph 4.10.3.2)
065	OUTBUF D, N	Set up output buffer ring with N buffers (refer to Paragraph 4.10.3.2)
066	INPUT D,	Request input or request next buffer (refer to Paragraph 4.10.5)

Table 4-1 (Cont)
Monitor Programmed Operators

Op Code	Call	Function
067	OUTPUT D,	Request output or request next buffer (refer to Paragraph 4.10.5)
070	CLOSE D,	Terminate file operation (refer to Paragraph 4.10.7)
071	RELEAS D,	Release device (refer to Paragraph 4.10.8.1)
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 6.2.8.3)
075	USETO D, E	Set next output block number (refer to Paragraphs 6.1.6.2 and 6.2.8.3)
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 4.10.4.1)
077	ENTER D, E error return normal return E: SIXBIT /FILE/ SIXBIT /EXT/ 0 XWD PROJ, PROG	Select a file for output (refer to a Paragraph 4.10.4.2)
100	UJEN	Dismiss real time interrupt (refer to Paragraph 4.10.10.2)

4.2.2.1 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 4-2 lists the monitor operations specified by the CALL and CALLI operations.

Table 4-2
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
CALLI AC, 0	RESET	CALL [SIXBIT /RESET/] return	Reset I/O device (refer to Paragraph 4.10.1.2)
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 4.3.1.1)
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 4.9.4.2)
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
CALLI AC, 10	WAIT	AC field is software channel number. CALL AC, [SIXBIT /WAIT/] only return	Wait until device is inactive (refer to Paragraph 4.10.5.3)
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 4.4.1)

Table 4-2 (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 4.3.2.3)
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 4.9.1.1)
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 available only to privileged programs (refer to Paragraph 4.7.1)
CALLI AC, 16	APRENB	MOVEI AC, BITS CALL AC, [SIXBIT /APRENB/] return	Enable central processor traps (refer to Paragraph 4.3.3.1)
CALLI AC, 17	LOGOUT	CALL AC, [SIXBIT /LOGOUT/] no return	Privileged UUO for use only by LOGOUT CUSP (refer to Paragraph 4.3.2.4)
CALLI AC, 20	SWITCH	CALL AC, [SIXBIT /SWITCH/] return AC: = contents of console data switches	Read console data switches (refer to Paragraph 4.9.4.1)
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 4.10.8.2)
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 4.9.1.2)

Table 4-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI Mnemonic	CALL	Function
CALLI AC, 23	MSTIME	CALL AC, [SIXBIT /MSTIME/] return AC: = time in milliseconds, right-justified	Read time of day in milliseconds (refer to Paragraph 4.9.1.3)
CALLI AC, 24	GETPPN	CALL AC, [SIXBIT /GETPPN/] normal return error return AC: = XWD proj. no., prog. no. of this job. Error 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 4.9.2.3)
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 4.10.10.1)
CALLI AC, 26	TRPJEN	CALL [SIXBIT /TRPJEN/]	Illegal UUO; replaced by UJEN (op code 100)
CALLI AC, 27	RUNTIM	MOVE AC, job number of 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 4.9.2.1)
CALLI AC, 30	PJOB	CALL AC, [SIXBIT /PJOB/] return AC: = job number, right-justified	Return job number (refer to Paragraph 4.9.2.2)
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 4.3.4.1)
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 4.9.3.1)
CALLI AC, 34	GETLIN	CALL AC, [SIXBIT /GETLIN/] return AC: = SIXBIT TTY name, left justified (e.g., CTY, TTY27)	Return SIXBIT name of attached Teletype console (refer to Paragraph 4.9.2.4)

Table 4-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI Mnemonic	CALL	Function
CALLI AC, 35	RUN	MOVSI AC, start adr increment HRRI AC, E RUN AC, error return normal return E: SIXBIT /DEVICE/ SIXBIT /FILE/ SIXBIT /EXT/ 0 XWD proj no, prog no XWD 0; optional core assignment	Transfer control to selected program (refer to Paragraph 4.5.1)
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 4.4.2)
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 4.5.3)
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 4.5.2)
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 4.9.3.3)
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 4.9.3.2)

Table 4-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI Mnemonic	CALL	Function
CALLI AC, 43	SETNAM	MOVE AC, [SIXBIT /NAME/] SETNAM AC, return	Set program name in monitor job table (refer to Paragraph 4.7.2)
CALLI AC, 44	TMPCOR	MOVE AC, [XWD CODE, BLOCK] TMPCOR, error return normal return BLOCK: XWD NAME, 0 IOWD BUFLEN, BUFFER AC: = value depending on CODE and whether error or normal return is taken.	Allow temporary in-core file storage for job (refer to Paragraph 4.8.1)
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 4.9.4.4)
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 4.9.4.5)
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 4.9.2.5)
CALLI AC, 50	STRUUO	MOVE AC, [XWD N, LOC] STRUUO AC, error return normal return AC: = status or error code	Manipulate file structures. Function 0 is the only unprivileged function (refer to Paragraph 4.6.1)

Table 4-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI Mnemonic	CALL	Function
CALLI AC, 50 (continued)	STRUUO	Contents Use LOC/function numbers arg LOC+1/ arg depending : on function : number	
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 4.9.4.8)
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 4.9.4.5).
CALLI AC,54	DEVSTS	MOVEI AC, channel no. of device DEVSTS AC, error return normal return	Return hardware device status word (refer to Paragraph 4.9.3.4)
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 4.9.4.3)
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 6.2.6.3)
CALLI AC,57	RTTRP	MOVEI AC, RTBLK RTTRP AC, error return normal return	Connect real-time devices to PI system (refer to Paragraph 8.3)
CALLI AC,60	LOCK	LOCK AC, error return normal return	Lock job in core (refer to Paragraph 8.2)
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 5.10.4.4)

Table 4-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI Mnemonic	CALL	Function
CALLI AC,62	LOCATE	MOVEI AC, location LOCATE AC, error return normal return	Change the job's logical location (refer to the Remote Batch Manual)
CALLI AC,63	WHERE	MOVEI AC, channel no. or MOVE AC, [SIXBIT/dev/] WHERE AC, error return normal return	Return the physical location of a device (refer to the Remote Batch Manual).
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 the Remote Batch Manual).
CALLI AC,65	CTLJOB	MOVE AC, job number CTLJOB AC, error return normal return	Return job number of controlling job (refer to Paragraph 5.10.4.5)
CALLI AC,66	GOBSTR	MOVE AC, [XWD N,LOC] GOBSTR AC, error return normal return LOC: SIXBIT/NAME/ or -1 LOC + 1: job number LOC + 2: XWD proj no, prog no LOC + 3: 0 LOC + 4: Status bits	Return next file structure name in an arbitrary job's search list (refer to Paragraph 4.9.2.6)
CALLI AC,67	ACTIVATE		Reserved for the future
CALLI AC,70	DEACTIVATE		
CALLI AC,71	HPQ	MOVE AC, high-priority queue no. HPQUUO AC, error return normal return	Place job in high priority scheduler's run queue (refer to Paragraph 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 4.3.4.2)
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 4.3.4.3)

Table 4-2 (Cont)
CALL and CALLI Monitor Operations

CALLI	CALLI Mnemonic	CALL	Function
CALLI AC,74	CHGPPN	MOVE AC, new proj. prog. no. CHGPPN AC, error return normal return	Change project-programmer number (refer to Paragraph 4.7.4)
CALLI AC,75	SETUUO	MOVE AC, [XWD function, argument] SETUUO AC, error return normal return	Set system and job parameters (refer to Paragraph 4.7.3)
CALLI AC,76	DEVGEN	MOVE AC, SIXBIT/dev/ DEVGEN AC, error return normal return	Return range and station of generic device (refer to the Remote Batch Manual)
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 4.9.2.7)
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 6.2.8.3)
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 4.9.4.6)

†The CALLI 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 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.

††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.

4.2.2.2 Suppression of Logical Device Names - Some CUSPs, e.g. LOGOUT, require I/O to specific physical devices regardless of the logical name assignments. Therefore, for any CALLI, if bit 19 (UPHNLY) is 1, only physical names will be used; logical device assignments 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.

4.2.2.3 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 (refer to Table 2-11).

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 (contains the SIXBIT monitor function name) and the effective address of the OPEN UUO (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.

4.2.3 Operation Codes 100-127 (Unimplemented Op Codes)

Op code 100-UJEN	Dismiss real-time interrupt from user mode (refer to Paragraph 8.4.2)
Op codes 101-127	Monitor prints ILL INST AT USER n and stops the job.

4.2.4 Illegal Operation Codes

The eight I/O instructions (e.g., DATAI) and JRST instructions attempting to enter executive or user I/O mode from the user mode are interpreted by the monitor as illegal instructions. The job is stopped and the following error message is printed on the user's console:

ERROR IN JOB n
ILL INST AT USER addr

4.3 EXECUTION CONTROL

4.3.1 Starting

A user program may start another program only by using the RUN or GETSEG UUOs (refer to Paragraph 4.5.1 and 4.5.2). A console user may start a program with the monitor commands RUN, START, CSTART, CONT, CCONT, DDT, and REENTER (refer to Chapter 2). The starting address is either an argument of the command or stored in the user's job data area (refer to Chapter 3).

4.3.1.1 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, JOBDDT. The starting address is used by the monitor command, DDT (refer to Paragraph 3.2.2.4).

4.3.2 Stopping

Any of the following procedures can stop a running program:

a. One ↑ C from user console if user program is in a Teletype input wait; otherwise, two ↑C's from user console (refer to Chapter 2);

b. A monitor detected error

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

4.3.2.1 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:

ERROR IN JOB
ILL INST AT USER adr

Note that the program cannot be continued by typing the CONT or CCONT commands.

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

ERROR IN JOB
HALT AT USER n

where n is the location of the HALT instruction.

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, JOBOPC 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 JOBOPC description, Table 3-1). HALT is not the instruction used to terminate a program (refer to Paragraph 4.3.2.3). HALT is useful for indicating impossible error conditions.

4.3.2.3 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 4.10.8.1), 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 8.4), it is resumed. In other

words, after releasing all I/O devices which close out all files, a RESET is done (refer to Paragraph 4.10.1.2). The carriage-return and line-feed is performed and

EXIT

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

When AC is 1, 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 console. 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 console to monitor mode. Programs using EXIT 1, as a substitute for EXIT (to eliminate the typing of EXIT) should RELEASE all devices first.

4.3.2.4 CALL [SIXBIT /LOGOUT/] or CALLI 17 - All I/O devices are RELEASed (refer to Paragraph 4.10.8.1), and returned with the allocated core and the job number to the monitor pool. The accumulated running time of the job is printed on the user's console, which is left in monitor mode. This UUO is not available to user programmers, but is only for use by the LOGOUT CUSP. If a user program executes a LOGOUT UUO, the monitor will treat it like EXIT (refer to Paragraph 4.3.2.3).

4.3.3 Trapping

4.3.3.1 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, nonexistent memory references, pushdown list overflow, arithmetic overflow, floating point overflow, and clock flag. To enable for trapping, a CALL AC, [SIXBIT /APRENB/] or CALLI AC, 16 is executed, where the AC contains the central processor flags to be tested on interrupts, as defined below:

AC Bit		Trap On
18	400000	Repetitive enable
19	200000	Pushdown overflow
22	20000	Memory protection violation
23	10000	Nonexistent memory flag
26	1000	Clock flag
29	100	Floating-point overflow
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 JOBCNI, and the PC is stored in location JOBTPC in the job data area (refer to Table 3-1). Then control is transferred to the user trap-answering routine specified by the contents of the right half of JOBAPR, after the arithmetic overflow and floating point overflow flags are cleared. The user program must set up location OBJAPR before executing the CALL AC, [SIXBIT /APRENB/] or CALLI AC, 16. To return control to his interrupted program, the user's trap-answering routine must execute a JRSTF @ JOBTPC to restore the state of the processor.

The APRENB UUO normally enables traps for only one occurrence of any selected condition and must be reissued 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

ERROR IN JOB n

followed by one of the following appropriate messages:

PC OUT OF BOUNDS AT USER addr
ILL MEM REF AT USER addr
NON-EX MEM AT USER addr
PDL OV AT USER addr

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

4.3.3.2 Error Intercepting - Device errors that can be corrected by human intervention are intercepted by the monitor, and control may be returned to the user program when the error is rectified. When these errors are detected, the monitor examines location JOBINT in the job data area. If this location is zero, the job is stopped and both the user and the operator are notified. The user receives the message

OPR zz ACTION REQUESTED FOR DEVICE xxx

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

OPERATE ON DEVICE xxx FOR JOB n

where xxx is the device name, and n is the number of 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 console signifying that the error has been corrected.

If location JOBINT 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 the following:

if bit 0 = 1, an error message is not to be typed on the user's Teletype.
if bit 0 = 0, an error message is to be typed on the user's Teletype.

CLASS is a set of bits interpreted as the following:

if bit 35 = 0, the job is to be stopped and a message is to be typed on the user's Teletype.

if bit 35 = 1, bit 0 (BITS) is examined to see if a message is to be typed. Bits 18-34 are reserved for future types of errors.

The monitor examines the CLASS bits first. If Bit 35 is zero, the user and the operator are given messages (see above), and the job is stopped. If Bit 35 is 1, the monitor examines LOC+2 in the block. If this location is non-zero, the messages are typed to the user and the operator, and the job is stopped. If the location LOC+2 is zero, the monitor examines bit 0 (BITS) to determine if a message should be typed. The following information is then stored in LOC+2 whether or not a message was typed:

LOC+2	the last user PC word
LOC+3	RH = the channel number LH = the error bit as defined in CLASS

The job is then started at location INTLOC.

4.3.3.3 Console-Initiated Traps - Program control can be regained from the user's console by use of the ↑C command (refer to Chapter 2).

4.3.4 Suspending

4.3.4.1 CALL AC, [SIXBIT /SLEEP/] or CALLI AC, 31 - This UUO stops the job, and continues automatically after an elapsed real time of [C(AC) x clock frequency] modulo 2^{12} jiffies. The contents

of the AC are thus interpreted as the number of seconds the job wishes to SLEEP; however, there is an implied maximum of approximately 68s (82s in 50-Hz countries) or 1 min.

4.3.4.2 HIBER AC, or CALLI AC,72

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) user's Teletype input activity (both line mode and character mode), 2) PTY activity for any PTY currently INITed by this job, 3) I/O activity for any I/O device INITed by this job, 4) the time-out of a specified amount of sleep time, or 5) 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-conditon enable bits, and in the right half the number of ms for which the job is to sleep before it is awakened.

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. 0 means no clock request (i.e., infinite sleep).
15-17	Wake protection code. Bit 17 = 1, project codes must match. Bit 16 = 1, programmer codes must match. Bit 15 = 1, only this job can wake itself.
13-14	TTY input activity. Bit 14 = 1, wake on character ready. Bit 13 = 1, wake on line of input ready.
12	PTY activity since last HIBERNATE.

An error return is given if the UUO is not implemented. Return is given on a normal return 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.

4.3.4.3 WAKE AC, or CALLI AC,73 - The WAKE UUO allows one job to activate a dormant job when some event occurs. This feature may 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 (refer to Paragraph 8.3), thereby allowing a real-time job to wake its background portion quickly in order to respond to some real-time condition.

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 this bit is set. The next time a HIBERNATE UUO is executed, this bit is cleared and the HIBERNATE UUO returns immediately. This bit eliminates the problem of a job going to sleep and missing any wake conditions.

On a normal return, the job has been awakened.

4.4 CORE CONTROL

4.4.1 CALL AC, [SIXBIT /CORE/] or CALLI, 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. 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.

The error return is given if LH is greater than or equal to 400000 and if either the system does not have a two-segment capability or the user has been meddling without write access privileges (refer to Paragraph 6.2.3). An RH of 0 leaves the low segment core assignment unaffected. The monitor clears new core before assigning it to the user; so therefore privacy of information is ensured.

In swapping systems, these programmed operators return 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 are returned; therefore, the CORE UUO and the CORE command return the same information.

The call is:

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

The CORE UUO reassigns the low segment (if RH is non-zero) and then reassigns 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 return 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 return 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 reestablishing 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.

4.4.2 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. An error 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.

4.4.3 LOCK AC, or CALLI AC, 60

This UUO locks jobs in core; refer to Paragraph 8.2.

4.5 SEGMENT CONTROL

4.5.1 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 starting
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.        ;if = 0, use current user's proj,prog
    XWD 0, optional core            ;RH = new highest user address to be
           assignment                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). 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 (CUSPs) 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 CUSP is reentrant or not.

The LH of AC is added to and stored in the starting address (JOBSA) 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 CUSP over again. The user is considered to be meddling with the program if the LH of AC is not 0 or 1. (Refer to Paragraph 4.5.4.)

Programs accept commands from a Teletype 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 CUSPs: 0 in LH of AC means type an asterisk and accept commands from the Teletype. A 1 means accept commands from a command file, if it exists; if not, type an asterisk and accept commands from the Teletype. The convention for naming CUSP 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 CUSP 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 are, therefore, deleted by the LOGOUT CUSP. (Refer to KJOB command in Chapter 2.)

The RUN UUO can give an error return with one of 20 error codes in AC if any errors are detected; thus, the user program may attempt to recover from the error/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 console 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.

To successfully program the RUN UUO for all size systems and for all CUSPs 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 low segment and new high segment and old high segment together may not exceed the capacity of core.)
   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 file name .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 capacity of core.)
    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 JOBCOR > 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. Otherwise, the monitor prints the error message

    ?filename.SAV NOT FOUND

    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 JOBCOR).

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.

4.5.2 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 operating 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. JOBDAT is not changed except for the setting of JOBHRL.

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, control is returned to two locations following the UUO, whether it is called from low or high segment. It should be called from 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. JOBSA and JOBREN are zeroed if they point to a high segment that is being removed. This produces the message

?NO START ADDRESS

Refer to steps 1 through 31 of the RUN UUO description (Paragraph 4.4.2) for details of GETSEG UUO operation.

4.5.3 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
```

To ensure that the amount remapped is a multiple of 1K decimal words, the monitor performs the inclusive OR function of 1777 and the user's request. 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 contents of AC are unchanged. The contents of JOBREL (refer to Paragraph 3.2.1) are set to the new highest legal user address in the low segment. The RH of JOBHRL will be 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.

4.5.4 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, 14           ;See if high segment is sharable
GETTAB  T,              ;look at monitor JBTSGN table
JRST    .+2             ;table or UUO not present
TLNN    T,200000        ;is sharable bit on?
JRST    NOTSHR          ;no, go ahead and modify here
                        ;if high segment is sharable.
```

4.5.5 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 UUOs. 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 DEASSIGN, FINISH, KJOB if 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. In a DECtape system, a new CUSP tape can be mounted followed by an ASSIGN SYS command, which clears segment names for the physical device, but does not assign the device because everyone needs to share it.

4.6 FILE STRUCTURE CONTROL

4.6.1 STRUUO AC, or CALLI AC, 50

This UUO manipulates file structures and is intended primarily for monitor support CUSPs.

The first word of the argument list specifies the function to be performed. Function 0 (.FSSRC) is the only unprivileged function; the other functions are used with the OMOUNT and UMOUNT CUSPs and are not discussed in this manual since they are not meant for general use.

The call is:

```
MOVE AC, [XWD N, LOC]     ;N is the number of words in the
                          ;argument list starting at location
                          ;LOC.
STRUUO AC,                ;or CALLI AC, 50
error return              ;AC contains an error code
normal return             ;AC contains status information
```

(Continued on next page)

```
LOC/ .FSSRC
LOC+1/ First SIXBIT filestructure name, left justified
LOC+2/ 0
LOC+3/ Status bits
LOC+4/ Second SIXBIT file structure name, left justified
LOC+5/ 0
LOC+6/ Status bits
  .
  .
  .
```

4.6.1.1 Function 0 .FSSRC - This function allows a new file structure search list to be specified for the job issuing the UUO. 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 a INIT or OPEN UUO.

The user may use the MOUNT command to add a new file structure name to his search list. The MOUNT CUSP

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 .FFSRC function, the monitor moves the file structure name from the active search list to the passive search list. However, because the mount count is not decremented, the user may still do I/O explicitly to the file structure. 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 that file structure.

Table 4-3
.FSSRC Error Codes

Symbol	Code	Explanation
.ERILF	0	Illegal function code
.ERSNF	1	One or more file structures not found.
.ERSSA	2	One or more file structures single access only.
.ERTME	4	Too many entries in search list.
.ERRSL	17	File structure is repeated in a search list definition.

4.7 PROGRAM AND PROFILE IDENTIFICATION

4.7.1 CALL AC, [SIXBIT /LOGIN/] or CALLI AC, 15

This UUO is not available to user programmers. It is for the exclusive use of the LOGIN CUSP, which uses this operator to exit to the monitor and to pass it certain crucial parameters (including project and programmer numbers) about the user who just successfully logged in. When the LOGIN CUSP calls this UUO, any devices the CUSP was using are released, and a period is printed on the user's console.

The console is left in monitor mode ready to accept the user's first command.

Any other user program that calls this UUO receives the error message

ILLEGAL UUO AT USER addr

The user's console is then put in monitor mode, and the CONT and CCONT commands are not permitted.

4.7.2 CALL AC, [SIXBIT /SETNAM/] or CALLI AC, 43

This UUO is used by the LOADER. The contents of AC contain a left-justified SIXBIT program name, which is stored in a monitor job table. The information in the table is used by the SYSTAT CUSP (refer to Table 4-4).

4.7.3 SETUUO AC, or CALLI AC, 75

The SETUUO is used to set various system or job parameters. Certain functions of this UUO are privileged. Privileges are granted if the job is either logged in under [1,2] or running with the JACCT bit on. The contents of AC contain a function code in the left half and an argument in the right half. The functions and arguments are as follows:

Function	Argument
0	CORMAX (actual number). Privileged function.
1	CORMIN (actual number). Privileged function.
2	DAYTIME (decimal number of minutes since midnight, hours * 100 + minutes). Privileged function.
3	SCHED (argument stored in RH of STATES word in COMMON). Privileged function.
4	CDR (input name counter for this job). If AC is non-zero, the contents is the same as the next input name. If 0, the current counter is returned in AC. Not a privileged function.
5	SPOOL for this job (bits are as stored in JBTSPL). Not a privileged function.
6	WATCH for this job (bits are as stored in JBTWCH). Not a privileged function.
7	DATE (decimal number of days since January 1, 1964, refer to Paragraph 4.9.1.1). Privileged function.

4.7.4 CHGPPN AC, or CALLI AC, 74 - This UUO is used by the LOGIN CUSP to change a user's project-programmer number. The call is:

```
MOVE AC, new project-programmer number
CHGPPN AC,                          ;or CALLI AC, 74
error return
normal return
```

The error return is given if the UUO is not implemented or if the job associated with the project-programmer is already logged in. The normal return is given if the job is not logged in, and the project-programmer number is changed.

4.8 INTER-PROGRAM COMMUNICATION

4.8.1 CALL AC, [SIXBIT /TMPCOR/] or CALLI AC, 44

This allows a job to leave several short files in core from the running of one user program or CUSP to the next. These files are referenced by a three-character filename and are unique to each job. All files are deleted when the job is killed. This system of temporary storage improves response times and reduces the number of disk operations.

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]
        CALL AC,  [SIXBIT /TEMPCOR/]      ;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.

4.8.1.1 CODE = 0, 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 free words available to the user is returned in AC.

4.8.1.2 CODE = 1, 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 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.

4.8.1.3 CODE = 2, 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.

4.8.1.4 CODE = 3, Write File - If a file exists with the specified name, it is deleted and its space reclaimed. The requested size of the file is specified by BUFLEN. 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.

4.8.1.5 CODE = 4, 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.

4.8.1.6 CODE = 5, 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 CUSP.

4.9 ENVIRONMENTAL INFORMATION

4.9.1 Timing Information

The central processor clock, which generates interrupts at the power-source frequency (60 Hz in North America, 50 Hz in most other countries), keeps time in the monitor. Each clock interrupt (tick) corresponds to 1/60th (or 1/50th) of a second of elapsed real time. The clock is set initially to the current time by console input when the system is started, as is the current date. When the clock reaches midnight, it is reset to zero, and the date is advanced.

4.9.1.1 CALL AC, [SIXBIT /DATE/] or CALLI AC, 14 - A 12-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 accumulator AC.

4.9.1.2 CALL AC, [SIXBIT /TIMER/] or CALLI AC, 22 - This UUO returns the time of day, in clock ticks (jiffies), right justified in accumulator AC.

4.9.1.3 CALL AC, [SIXBIT /MSTIME/] or CALLI AC, 23 - This UUO returns the time of day, in milliseconds, right justified in accumulator AC.

4.9.2 Job Status Information

4.9.2.1 CALL AC, [SIXBIT /RUNTIM/] or CALLI AC, 27 - The accumulated running time (in milliseconds) of the job number in accumulator AC is returned right justified in accumulator 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.

4.9.2.2 CALL AC, [SIXBIT /PJOB/] or CALLI AC, 30 - This UUO returns the job number right justified in accumulator AC.

4.9.2.3 CALL AC, [SIXBIT /GETPPN/] 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 is LOGIN or LOGOUT from the system device, a skip return is given if the old project-programmer number is also logged in on another job.

4.9.2.4 CALL AC, [SIXBIT /GETLIN/] or CALLI AC, 34 - This UUO returns the SIXBIT physical name of the Teletype console that the program is attached to.

The call is:

```
CALL AC, [SIXBIT/GETLIN/]          ;OR CALLI AC, 34
```

The name is returned left justified in the AC.

Example:

CTY or TTY3 or TTY30

This UUO is used by the LOGIN program to print the TTY name.

4.9.2.5 CALL AC, [SIXBIT /JOBSTR/] or CALLI AC, 47 - This UUO returns the next file structure name in the job's search list along with other information about the file structure. Programs like PIP 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]
CALL AC,  [SIXBIT /JOBSTR/]          ;or CALLI AC, 47
error return
normal return
```

LOC is the address of an N-word block. The first word of this block should contain either -1 or the last value returned by the previous JOBSTR. On return, the first word is either the next file structure name or -1 if all file structure names have been returned. The second word contains the project-programmer number requested in the file structure, and the third word contains status bits. Current status bits include the following:

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.

4.9.2.6 GOBSTR AC, or CALLI AC, 66 - This privileged 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 4.9.2.5).

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:  SIXBIT /file structure name/     ;or -1
      job number                       ;job whose search
                                       ;list is desired.
      XWD proj, prog                   ;project-programmer
                                       ;number of above job.
      0                                ;currently unused.
      Status                           ;status bits are the same
                                       ;as in JOBSTR UUO.
```

If the job number and project-programmer number are both zero, the system search list is searched.

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

Code	Meaning
0	If LOC is not -1 or a file structure name in jobs search list.
1	If job issuing the UUO is not privileged.
2	If job number (LOC + 1) and project-programmer number (LOC+2) do not correspond.

4.9.2.7 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.

4.9.3 Monitor Examination

4.9.3.1 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. Some customers may want to restrict the use of this UUO to project 1.

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.

4.9.3.2 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. Some installations may restrict use of the SPY UUO to certain privileged users (e.g., project 1 only).

4.9.3.3 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
```

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 inidcates 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 is non-reentrant. The right half of AC contains a table number from the list of monitor data tables and parameters in Table 4-4. 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 operations.

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 that 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 CUSP 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. Because the field is currently zero, there is no reason to believe that it will always be zero.

Table 4-4
GETTAB Tables

Table Numbers (RH of AC)	Table Names	Explanation
00	JBTSTS	Job status word; index by job or segment number.
01	JBTADR	Job relocation and protection; index by job or segment number.
02	PRJPRG	Project and programmer numbers; index by job or segment number.
03	JBTPRG	User program name; index by job or segment number.
04	TTIME	Total time used; index by job number.
05	JBTKCT	Kilo-Core ticks of job; index by job number.
06	JBTPRV	Privilege bits of job; index by job number.
07	JBTSWP	Swapping Parameters of job; index by job or segment number.
10	TTYTAB	Teletype-to-job translation; index by console line number.
11	CNFTBL	Configuration table; index by item number, see below.
12	NSWTBL	Nonswapping data; index by item number, see below.
13	SWPTBL	Swapping data; index by item number, see below.
14	JBTSGN	High segment table; index by job number. Bit 0 = 0, then bits 18-35 is index of high segment (if bits 18-35 = 0, then there is no high segment). Bit 0 = 1, then bits 18-35 is number of K to spy on. Bit 1 = 1 if job has a high segment that is sharable.
15	ODPTBL	Once-only disk parameters; index by item number, see below.
16	LVDTBL	Level D disk parameters; index by item number, see below.
17	JBTRCT	Disk blocks read by job; used by DSK command: a. Bits 0-11 = incremental blocks. b. Bits 12-35 = total blocks since LOGIN. Index by job number.
20	JBTWCT	Disk blocks written by job: a. Bits 0-11 = incremental blocks. b. Bits 12-35 = total blocks since LOGIN. Index by job number.
21	JBTDBS	Reserved for future.
22	JBTTDB	a. Bits 0-16 = time of day in seconds of last disk allocation. b. Bits 17-35 = number of disk blocks allocated on all file structures of this job; index by job number.

10-1406

```
 R LOADER
*/DDSK:BASLD.15R,BASHF.15R$

LOADER 7+6K CORE
14+3K MAX 263 WORDS FREE

EXIT
↑C

.DDT


$G

NEW OR OLD--OLD
OLD FILE NAME--NUMBR

READY
RUNNH

 1.00000 E+8
 1.00000 E-24    1.00000 E-36    2.00000 E-15    0.0004          0
 3000            4.00000 E+14    5.00000 E+23    6.00000 E+35
 2.34560 E-24
 4.78000 E-13    4.67890 E-5     5.64792 E-9
 3.9            -3.67000 E+12   -5.56453 E+22   -7.78788 E-22
-4567
-56785678        0               -2              -8.9999
 6.5432

READY
↑C

.
```

ANSWER TO SOFTWARE TROUBLE REPORT 10-1406

Diagnosis: The user asked user DDT to put break in a sharable high segment. This caused the sharable high segment to be modified with the JSR instruction. Subsequent GETS do not change the sharable high segment. This is the whole point of sharable segments, file IO is minimized.

Solution: To avoid this problem, it is advisable to debug with non-sharable high segments using the SAVE command instead of the SSAVE.

Since this is not an obvious pitfall, I am submitting a suggestive STR to DDT, that it not allow breakpoints in a high segment if it is sharable.

Also, an addition to the Reference Manual will describe how a program may determine if its high seg is sharable.

DOCUMENTATION CORRECTION:

Document: Time Sharing Monitors
DEC #: DEC-T9-MTZB-D
Location of Error: Chapter 4
Page 392
Line After Section 4.3.7.4

4.9.3.2 1/2

4.3.7.5 Testing for a sharable high segment.

Occasionally it is desirable for a program to find out whether its high segment is sharable or not. The program may decide not to modify itself, if it is sharable. The following code will work whether or not the system has a high segment capability or not, whether the job has a high segment or not:

```
HRROI  T,14    ;SEE IF OUR HI-SEG IS SHARABLE
GETTAB T,      ;LOOK AT MONITOR JBTSGN TABLE
  JRST  .+2    ;TABLE OR UUO NOT PRESENT
TLNN  T,200000    ;IS SHARABLE BIT ON?
JRST NOTSHR       ;NO, GO AHEAD AND MODIFY
                  ;YES, DO NOT MODIFY
```

Also: P. 381
After "14-JBTSGN" on next line, add:
Bit 1=1 if job has a high segment and it is sharable.

Table 4-4 (Cont)
GETTAB Tables

Table Numbers (RH of AC)	Table Names	Explanation
23	NUMTAB	Table of GETTAB addresses (GETTAB immediate); index by GETTAB table number, see below.
24	JBTDEV	Device or file structure name of sharable high segment. Index by high segment number.
25	STSTBL	Two-character SIXBIT names for job queues; index by item numbers, see below.
26	JBTLOC	Reserved for future.
27	CORTAB	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	COMTAB	Table of SIXBIT names of monitor commands.
31	JBTNM1	First half of name of user in SIXBIT; index by job number.
32	JBTNM2	Last half of name of user in SIXBIT; index by job number.
33	JBTCNO	Job's charge number; index by job number.
34	JBTTMP	Job's TMPCOR pointers; index by job number.
35	JBTWCH	Job's WATCH bits; index by job number.
36	JBTSPL	Job's spooling control bits; index by job number.
37	JBTRTD	Job's real-time status word; index by job number.
40	JBTLIM	Job's time limit in jiffies; index by job number.
41	QQQTAB	Timesharing scheduler's queue headers.
42	JBTQ	Timesharing scheduler's queue that job is in; index by job number.

Entries in Table 11 - CNFTBL (Configuration Table)

Item	Location	Use
0	CONFIG	Name of system in ASCIZ
-		
4	CONFIG+4	
5	SYSDAT	Date of system in ASCIZ
6	SYSDAT+1	
7	SYSTAP	Name of system device (SIXBIT)
10	TIME	Time of day in jiffies

Entries in Table 11 - CNFTBL (Configuration Table) (Cont)

Item	Location	Use
11	THSDAT	Today's date (12-bit format)
12	SYSSIZ	Highest location in monitor + 1
13	DEVOPR	Name of OPR TTY console (SIXBIT)
14	DEVLST	LH is start of DDB (device-data-block) chain
15	SEGPTR	LH=-# of high segments, RH=+# of JOBS (counting NULL job)
16	TWOREG	Non-zero if system has two-register hardware and software
17	STATES	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 (FTDISK) Bit 1=1 if swap system (FTSWAP) Bit 2=1 if LOGIN system (FTLOGIN) Bit 3=1 if full duplex software (FTTYSER) Bit 4=1 if privilege feature (FTPRV) Bit 5=1 if assembled for choice of reentrant or non-reentrant software at monitor load time (FT2REL) Bit 6=1 if clock is 50 cycle instead of 60 cycle. Bits 7-9 type of disk system if 0, 4-series disk system if 1, 5-series disk system Bit 10=1 if independent programmer numbers between project (INDPPN is non-zero) Bit 11=1 if image mode on Teletype (8-bit SCNSER) Set by the privileged operator command, SCHEDULE: Bit 34=1 means no remote LOGINs Bit 35=1 means no more LOGINs except from CTY
20	SERIAL	Serial number of PDP-10 processor Set by MONGEN dialog
21	MEMNSP	Number of nanoseconds per memory cycle for memory system. Used by SYSTAT to compute shuffling time.
22	PTYCNF	PTY parameters for Batch. LH = the number of the first invisible Teletype (PTYOFS). RH = the number of PTY's in the system configuration (PTYN).
23	FREPTR	AOBJN word to use bit map in monitor for allocating 4-word core blocks.
24	LOCORE	LH=0, RH=address in monitor for free 4-word core block areas. (This is never changed while monitor runs.)
25	STBSTR	Link to STB chain for remote Batch.

Entries in Table 12 - NSWTBL (Nonswapping Data)

Item	Location	Use
0	CORTAB	Obsolete,
-		unspecified data
7	CORTAB+7	
10	CORMAX	Size in words of largest legal user job (low seg+high seg)
11	CORLST	Byte pointer to last free block in CORTAB
12	CORTAL	Total free+dormant+idle K physical core left
13	SHFWAT	Job number shuffler has stopped
14	HOLEF	Absolute address of job above lowest hole, 0 if no job
15	UPTIME	Time system has been up in jiffies
16	SHFWRD	Total number of words shuffled by system
17	STUSER	Number of job using SYS if not a disk
20	HIGHJB	Highest job number currently assigned
21	CLRWRD	Total number of words cleared by CLRCOR
22	LSTWRD	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
23	MEMSIZ	Size of physical memory in words
24	PARTOT	Total number of user parity errors (memory) since system was loaded.
25	PARSPR	Total number of spurious (refer to Paragraph 3.1.1) parity errors (memory).
26	PARCON	Total number of multiple parity errors (memory).
27	PARADR	The absolute location of the last user mode memory parity error.
30	PARWRD	The contents of the last user mode memory parity error.
31	PARPC	The user PC of the last user mode memory parity error.
32	EPOCNT	Total number of PDL OVR's at UUO level in exec mode.
33	EPOREC	Number of PDL OVR's at UUO level which were recovered by assigning extended list.
34	MAXMAX	Highest legal value of CORMAX.
35	SYSKTM	Count-down timer for SET KSYS command
36	CORMIN	Amount of core guaranteed to be available after locking jobs in core.

Entries in Table 13 - SWPTBL (Swapping Data)

Item	Location	Use
0	BIGHOL	Number of K in biggest hole in core
1	FINISH	+Job number of job being swapped out -Job number of job being swapped in
2	FORCE	Job being forced to swap out
3	FIT	Job waiting to be fit into core
4	VIRTAL	Amount of virtual core left in system in K (initially set to number of K of swapping space)

Entries in Table 13 - SWPTBL (Swapping Data) (Cont)

Item	Location	Use
5	SWPERC	LH=number of swap read or write errors RH=error bits (bits 18-21 same as status bits) + number of K discarded

Entries in Table 15 - ODPTBL (Once-Only Disk Parameters)

Item	Location	Use
0	SWPHGH	Unused, contains zero in 5-series monitors.
1	K4SWAP	K of disk words set aside for swapping on all units in active swapping list.
2	PROT	In-core protect time multiplies size of job in K-1
3	PROT0	In-core protect time added to above result after multiply.

Entries in Table 16 - LVDTBL (Level D Disk Parameters)

Item	Location	Use
0	MFDPPN	Project-programmer number for UFDs only (1,1)
1	SYSPPN	Project-programmer number for device SYS (1,4)
2	FSFPPN	Project-programmer number for FAILSAFE (1,2)
3	HELPPP	Project-programmer number for SYSTAT and HELP (2,5)
4	PNTPPN	Project-programmer number for PRINTER spooling program (3,3)
5	SYSPPB	a. LH=address of first PPB block b. RH=address of next PPB block to be scanned
6	SYSSTR	a. LH=address of first file structure data block b. RH=unused
7	SYSUNI	a. LH=address of data block of first unit in system b. RH=unused
10	SWPUNI	a. LH=address of first unit for swapping in system b. RH=unused
11	CORNUM	Number of 4-word access blocks for disk systems allocated at ONCE - only time.
12	STNPRT	Standard file privilege code (057), can be changed by installation
13	UFDPRT	Standard UFD privilege code (775), can be changed by installation
14	MBFNUM	Number of monitor buffers allocated at once-only time (2)
15	QUESTR	SIXBIT name of file structure containing 3,3.UFD for PRINTR and OPFILE queues.
16	CRUPPN	UFD used for storing system crashes.

Entries in Table 23 - NUMTAB (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.

The format of each entry is as follows:

LH = Bits 0-8 = maximum item number in table
Bits 14-17 = a monitor AC.

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

Examples:

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

Entries in Table 25 - STSTBL (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 code for a disk system are as follows:

RN	-	one of the run queues
WS	-	I/O wait satisfied
TS	-	Teletype I/O wait satisfied
DS	-	disk I/O wait satisfied
ST	-	system tape wait
AU	-	disk alter UFD wait
MQ	-	disk monitor buffer wait
DA	-	disk storage allocation wait
CB	-	disk core block scan wait
DT	-	DECtape control wait
DC	-	data control wait
MT	-	magnetic control wait
CA	-	core allocation wait (to be locked)
I/O	-	I/O wait
TI	-	Teletype I/O wait
DI	-	disk I/O wait

SL	-	sleep wait
NU	-	null state
ST	-	stop (↑C) state

These state codes are printed by SYSTAT and MOVEI.

4.9.3.4 DEVSTS AC, or CALLI AC, 54 - This UUO is a diagnostic UUO used to retrieve 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. 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
DEVSTS AC,                          ;or CALLI AC, 54
error return                        ;UUO not implemented for any
                                    ;devices
normal return                       ;AC contains the DEVSTS
                                    ;word of the DDB.
```

Upon 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).

The DEVSTS UUO is not meaningful when used in asynchronous buffered I/O mode unless a WAIT UUO (see Paragraph 4.10.5.3) is issued first to ensure synchronization of the actual data transferred with the device status returned.

4.9.4 Configuration Information

4.9.4.1 CALL AC, [SIXBIT /SWITCH/] 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.

4.9.4.2 CALL AC, [SIXBIT /DEVCHR/] 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 1) the logical or physical device name as a left-justified SIXBIT quantity, or 2) 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
        CALL AC, [SIXBIT/DEVCHR/]       ;or CALLI AC,4
        return
```

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

Bit	Explanation
Bit 0 = 1	DECtape directory is in core. This bit is cleared by an ASSIGN or DEASSIGN to that unit.
Bit 1 = 1	Device is a file structure.
Bit 2 = 1	Device is a card reader or card punch.
Bit 3 = 1	Device is a line printer.
Bit 4 = 1	TTY is attached to a job.
Bit 5 = 1	TTY is in use as a user console (even if detached).
Bit 6 = 1	TTY is in use as an I/O device.
Bit 7 = 1	Device is a display.
Bit 8 = 1	Device has a long dispatch table (that is, UUOs other than INPUT, OUTPUT, CLOSE, and RELEASE perform real actions).
Bit 9 = 1	Device is a paper-tape punch.
Bit 10 = 1	Device is a paper-tape reader.
Bit 11 = 1	Device is a DECtape.
Bit 12 = 1	Device is available to this job or is already assigned to this job.
Bit 13 = 1	Device is a magnetic tape.
Bit 14 = 1	Device is a TTY.
Bit 15 = 1	Device has a directory (DTA or DSK).
Bit 16 = 1	Device can perform input.
Bit 17 = 1	Device can perform output.
Bit 18 = 1	Device is assigned by a console command.
Bit 19 = 1	Device is assigned by program (INIT).
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).

4.9.4.3 CALL AC, [SIXBIT/DEVPPN/] or CALLI AC, 55 - This UUO allows a user program to obtain the project-programmer number associated with a device name.

The call is:

```
MOVE AC, [SIXBIT /DEV/]
CALL AC, [SIXBIT / DEVPPN/]          ;or CALLI AC, 55
error return
normal return
```

DEV may be a logical or physical device name or SYS. The error return is taken if:

a. The UUO is not implemented; therefore, the contents of AC remain the same on return

b. The device does not exist, therefore, zero is returned in AC.

If a legal device is specified, the normal return is given and the project-programmer number of either the user's directory or device SYS is returned in AC.

The following is an example to read a UFD even if device SYS is specified.

```
        MOVEI A,16                          ;get MFD project-programmer number
        CALL A,[SIXBIT /GETTAB/]            ;no change if no GETTAB
          MOVE A, [1,,1]                    ;in case of level C
        MOVEM A,MFDPPN                      ;store MFD directory number
        MOVE A, DEVICE NAME TYPED BY USER   ;store device name for OPEN
        MOVEM A,MODE+1
        CALL A,[SIXBIT /DEVPPN/]            ;get project-programmer number
                                            ;implied by the device name
        MOVEI A,0                           ;not implemented or no such device
        MOVEM A,PPN                         ;store project-programmer number
                                            ;associated with this device
        OPEN A,MODE                         ;try to assign device
          JRST ERROR                        ;not available
        LOOKUP A,PPN                        ;try to lookup UFD
          JRST ERROR                        ;not there
        INPUT A,                            ;read first block UFD
              .
              .
              .

MODE:   14                                  ;mode is binary
        0                                   ;device name
        XWD 0,INBUF                         ;buffer headers
PPN:    0                                   ;directory names
        SIXBIT /UFD/                        ;extension
        0
MFDPPN: XWD 1,1                             ;lookup UFD in MFD
```

4.9.4.4 CALL AC, [SIXBIT /DSKCHR/] 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 CUSPs.

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
CALL AC, [SIXBIT /DSKCHR/]    ;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.

Symbol	Bit	Explanation
.UPRHB	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.
.UPOFL	Bit 1 = 1	The unit is off-line.
.UPHWP	Bit 2 = 1	The unit is write-protected.
.UPSWP	Bit 3 = 1	The unit belongs to a file structure that is write-protected by software for this job.
.UPSAF	Bit 4 = 1	The unit belongs to a single-access file structure.
.UPZMT	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 models.
.UPPRF	Bit 6 = 1	The unit belongs to a private file structure.
	Bits 7 and 8	
	= 11	The unit is down
	= 10	No pack is mounted
	= 01	A pack is being mounted (desired file structure and pack ID is already known by the monitor).
	= 00	A pack is mounted.

Symbol	Bit	Explanation
.UPMSB	Bit 9 = 1	The unit has more than one SAT block.
.UPNNA	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.
.UNIAWL	Bit 11 = 1	The file structure is write-protected for all jobs.
	Bits 12 - 14	Reserved for future expansion.
	Bits 15 - 17	The code identifies which type of argument was passed to the monitor in location LOC.
	Bits 18 - 20	Data channel number that software believes hardware is connected to; first data channel is 0.
	Bits 21 - 26 = 0 = 1 = 2 = 3	Controller type: DR (future drum) controller RK10 FH (Burroughs disk, Bryant drum) controller RC10 DP (~~Memorex~~ disk packs) controller RP10 MD (Bryant mass disk) controller RA10
	Bits 27 - 29	Controller number; first controller of each type starts at 0 (e.g., DPA = 0, DPB = 1)
	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 on RC10
		1 1 RM10B Bryant drum on RC10
		2 0 RP01 disk pack on RP10
		2 1 RP02 disk pack on RP10
		3 0 RB10B dual positioner on RA10
		3 1 RB10A single positioner on RA10
	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:

0	generic disk name
1	subset of file structures because of file structure abbreviation
2	file structure name
3	unit within a file structure
4	controller class name
5	controller class
6	physical disk unit name

or a logical name for one of the above assigned by the ASSIGN command.

On a normal return, the monitor returns values in the following locations:

LOC+1 (.UFTAL)	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 (.STTAL)	The number of blocks on a first-come first-served basis left for all users on the file structure.
LOC+3 (.UNTAL)	The number of blocks left for all users on the specified unit.
LOC+4 (.STNAM)	The file structure name to which this unit belongs.
LOC+5 (.UNCHR)	a. Bits 0-8 are the number of blocks/cluster. b. Bits 9-17 are the number of blocks/track. c. Bits 18-35 are the number of blocks/cylinder (see Appendix H).
LOC+6 (.UNBPU)	The number of 128-word blocks on the specified unit.
LOC+7 (.STMNT)	The mount count is the number of jobs that have done a MOUNT command for this file structure without executing a REMOVE command; it is a use count (available in 5.02 monitors and later monitors).
LOC+10 (.UNWPS)	The number of words containing data bits per SAT block on this unit.
LOC+11 (.UNSPU)	Number of SAT blocks per unit.
LOC+12 (.UNK4S)	Number of K allocated for swapping.
LOC+13 (.STJOB)	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 n has file structure mounted and it is single access.
LOC+14 (.UNLOG)	The unit's logical name (e.g., DSKB0).
LOC+15 (.UNNAM)	The unit's physical name (e.g., DPA0).
LOC+16 (.UNHID)	The unit's ID (e.g., 2PR003).

4.9.4.5 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. On a normal return, if AC=0, the specified device does not exist. If the device exists, the following information is returned in AC.

Symbol	Bit	Explanation
.TYMAN	Bit 0 = 1	LOOKUP/ENTER mandatory.
	Bits 1-11	Reserved for the future.
.TYAVL	Bit 12 = 1	Device is available to this job.
.TYSPL	Bit 13 = 1	Spooled on disk. (Other bits reflect properties of real device, except variable buffer size.)
.TYINT	Bit 14 = 1	Interactive device (output after each break character).
.TYVAR	Bit 15 = 1	Capable of variable buffer size (user can set his own buffer lengths).
.TYIN	Bit 16 = 1	Capable of input.
.TYOUT	Bit 17 = 1	Capable of output.
	Bits 18-26	Job number that currently has device INITed or ASSIGNed.
	Bits 27-29	Reserved for the future.
	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-Teletype Code 13 (.TYPLT) Plotter Code 14-57 Reserved for Digital Code 60-77 Reserved for customer

4.9.4.6 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.

4.9.4.7 SYSSTR AC, or CALLI AC, 44 - 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 CUSPs 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.

4.9.4.8 SYSPHY AC, or CALLI AC, 51 -This UUO returns all physical disk units in the system. This SYSPHY UUO is similar to the SYSSTR UUO (see Paragraph 4.9.4.5).

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 or 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.

4.10 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 since the monitor

a. Allows a logical device name to be associated with a physical device (ASSIGN 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.

4.10.1 I/O Organization

4.10.1.1 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 mganetic 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.

4.10.1.2 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 command (refer to Chapter 2), the devices are returned to the monitor facilities pool. The content of the left half of JOBSA (program break) is stored in the right half of JOBFF so that the user buffer area is reclaimed if the program is restarting. The left half of JOBFF 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.

4.10.2 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 monitor command. Some CUSPs, 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, 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 4.10.8.1) 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).

4.10.2.1 Nondirectory Devices - For nondirectory devices, (e.g., card reader and punch, line printer, paper-tape reader and punch, and user console) 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.

4.10.2.2 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.

4.10.2.3 Device Initialization - The OPEN (operation code 050) and INIT (operation code 041) programmed operators initialize a file by specifying a device, ldev, and initial file status, STATUS, and the location of the input and output buffer headers.

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

a. Data Channel - OPEN and INIT establish a correspondence between the device, ldev, and a 4-bit data 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 release (refer to Paragraph 4.10.8.1). The device name, dev, is either a logical or physical name, with logical names taking precedence over physical names (refer to ASSIGN command, Chapter 2). If the device, dev, is not the system device SYS and is allocated to another job or does not exist, the error return is taken. In nondisk systems, if the device is the system device SYS, the job is put into a system device wait queue, and continues running when SYS becomes available. In disk systems where the system device SYS is one or more file structures, control returns immediately.

b. 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 4-7 for the file status bits. If the data mode is not legal (refer to Chapter 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 console. The console is left in monitor mode.

c. 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 operators. Tables 4-5 and 4-6 summarize the data modes.

Table 4-5
Buffered Data Modes

Octal Code	Mnemonic	Meaning
0	A	ASCII. 7-bit characters packed left justified, five characters per word.
1	AL	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 and PRT only
2-7		Unused
10	I	Image. A device dependent mode. The buffer is filled with data exactly as supplied by the device.
11-12		Unused.
13	IB	Image binary. 36-bit bytes. This mode is similar to binary mode, except that no automatic formatting or checksumming is done by the monitor.
14	B	Binary. 36-bit byte. 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 4-6
Unbuffered Data Modes

Octal Code	Mnemonic	Meaning
15	ID	Image dump. A device dependent dump mode.
16	DR	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.
17	D	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.

d. 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 the buffer available to

the user's program, advances to the next buffer in the ring, and fills the buffer if it is free. The user's program follows the monitor, emptying the next buffer if it is full, or waiting for the next buffer to fill.

During output operations, the user's program and the monitor exchange roles; the user 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 console, leaving the console 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.

4.10.3 Ring Buffers

4.10.3.1 Buffer Structure - The ring buffer (see Figure 4-1) is comprised of a buffer ring header block and bufferings.

a. 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 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 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 the 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. User's cannot use the same buffer ring for simultaneous input and output; only one buffer ring is associated with each buffer ring header.

---

†Buffer headers may not be in the user's ACs; however, the buffer headers may be in location above JOBPFI (refer to Table 3-1).

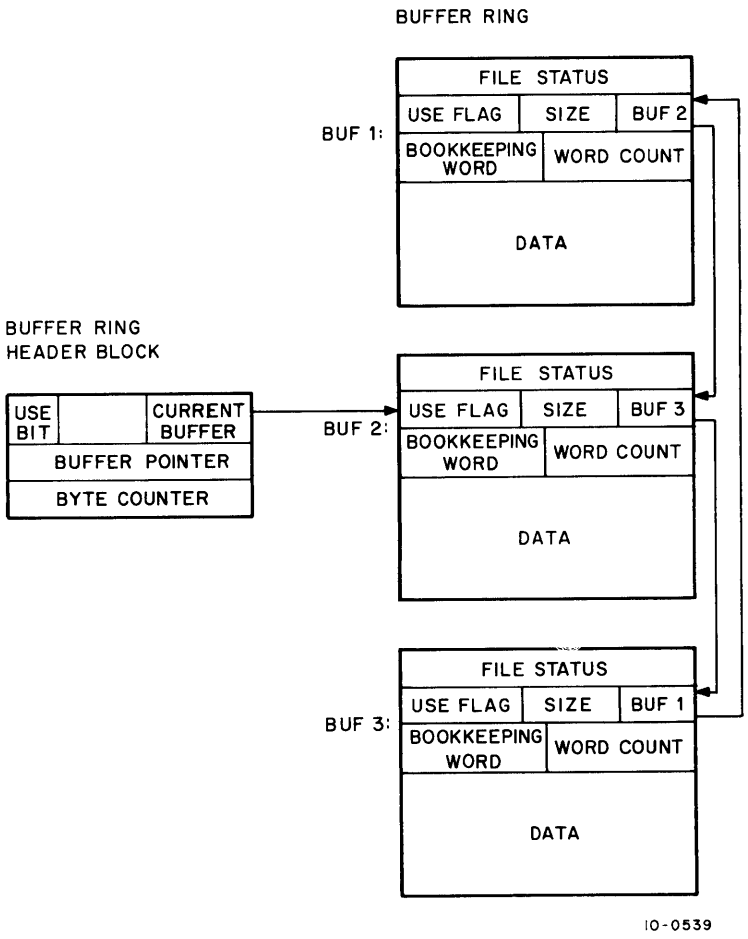


Figure 4-1 User's Ring of Buffers

b. 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 JOBFF in the user's job data area. The monitor updates JOBFF 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.

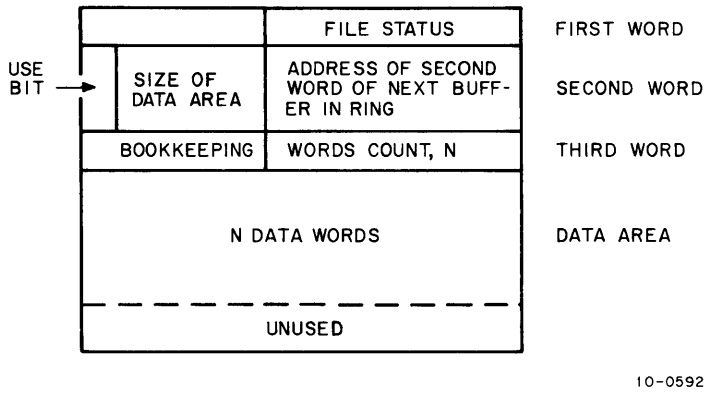


Figure 4-2 Detailed Diagram of Individual Buffer

4.10.3.2 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.

a. 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 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 console leaving the console 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 JOBFF. 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 console, leaving the console 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

b. 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.

```
CO:     INIT 1,0                        ;INITIALIZE ASCII MODE
        SIXBIT/MTA0/                    ;MAGNETIC TAPE UNIT 0
        XWD 0, MAGBUF                   ;INPUT ONLY
        JRST NOTAVL
        MOVE 0, [XWD 400000,BUF1+1]     ;THE 400000 IN THE LEFT HALF
                                        ;MEANS THE BUFFER WAS NEVER
                                        ;REFERENCED.
        MOVEM 0, MAGBUF
        MOVE 0, [POINT BYTSIZ,0,35]     ;SET UP NON-STANDARD BYTE
                                        ;SIZE
        MOVEM 0, MAGBUF+1
        JRST CONTIN                     ;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
        BLOCK SIZE+1                    ;SPACE FOR DATA, 1ST WORD
                                        ;RECEIVES WORD-COUNT. THUS
                                        ;ONE MORE WORD IS RESERVED
                                        ;THAN IS REQUIRED FOR DATA
                                        ;ALONE
BUF2:   0                               ;SECOND BUFFER
        XWD SIZE+1,BUF3+1
        BLOCK SIZE+1
BUF3:   0                               ;THIRD BUFFER
        XWD SIZE+1,BUF1+1               ;RIGHT HALF CLOSES THE RING
        BLOCK SIZE+1
```

4.10.4 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 the ASSIGN command, Chapter 2). The monitor gives the normal return for a LOOKUP or ENTER to a non-directory device; therefore, user programs can be coded in a device-independent fashion.

4.10.4.1 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
        0
        XWD project number, programmer number
```

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 console 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.

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 either the file is not found or, the file is found but the user does not have access to it.

4.10.4.2 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
        EXP<PROT>B8+EXP<TIME>B3+DATE
        XWD project number, programmer number.
```

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 console 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 upon 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 4.10.7.7). 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 file is not found (LOOKUP only)

b. The filename in location E is 0

c. The file is found but is being written or renamed (ENTER only)

d. 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.

4.10.4.3 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
    EXP<PROT>B8+<TIME>B23+DATE
    XWD project number, programmer number
```

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 console 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 or renamed.

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

e. 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 if found, the error return is taken.

Refer to Appendix E for the error codes returned in bits 33-35 of location E+1.

Examples

General Device Initialization

```
INIDEV: 0                       ;JSR HERE
        INIT 3, 14              ;BINARY MODE, CHANNEL 3
        SIXBIT/DTA5/            ;DEVICE DECTAPE UNIT 5
        XWD OBUF,IBUF           ;BOTH INPUT AND OUTPUT
        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
OBUF:   BLOCK 3                 ;SPACE FOR OUTPUT BUFFER HEADER
IBUF:   BLOCK 3                 ;SPACE FOR INPUT BUFFER HEADER
INNAM:  SIXBIT/NAME/            ;FILE NAME
        SIXBIT/EXT/             ;FILE NAME EXTENSION (OPTIONALLY 0),
                                ;RIGHT HALF WORD RECEIVES THE
                                ;FIRST BLOCK NUMBER
        0                       ;RECEIVES THE DATE
        0                       ;UNUSED FOR NONDUMP I/O
OUTNAM: SIXBIT/NAME/            ;SAME INFORMATION AS IN INNAM
        SIXBIT/EXT/
        0
        0
```

4.10.5 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 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 console and the console 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. 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.

4.10.5.1 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.

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 console and the console 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
        INIT 0, 16              ;CHANNEL 0, DUMP MODE
        SIXBIT/MTA2/            ;MAGNETIC TAPE UNIT 2
        0                       ;NO RING BUFFERS
        JRST NOTAVL             ;WHERE TO GO IF UNIT 2 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 OUTLIST
        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
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
```

4.10.5.2 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.

a. 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 4.10.3.2).

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

```
GETCHR: 0                       ;CALL IS JSR GETCHR
GETNXT: SOSG IBUF+2             ;DECREMENT THE BYTE COUNT
        IN 1,                   ;GET NEXT BUFFER FROM MONITOR
        JRST GETOK              ;RETURN WHEN BUFFER IS FULL
                                ;TEST ERROR BITS
        STATZ 1,740000          ;GO PROCESS ERROR
        JRST INERR              ;ASSUME END-OF-FILE
        JRST INEOF              ;GET CHARACTER FROM BUFFER
GETOK:  ILDB AC,IBUF+1          ;RETURN IF NOT NULL CHARACTER †
        JUMPN AC,@GETCHR        ;IGNORE NULL AND GET NEXT CHARACTER
        JRST GETNXT
```

b. 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 4.10.3.2). 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

†For some devices in ASCII mode, the item count provided is always a multiple of five characters. The last word of a buffer may be partially full; therefore, user programs that rely on the item count should always ignore null characters.

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 first data 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

```
PUTCHR: 0                   ;CALL IS JSR PUTCHR
        SOSG OBUF+2         ;DECREMENT BYTE COUNT
        OUT 2,              ;CALL MONITOR TO EMPTY THIS BUFFER
        JRST PUTOK          ;RETURN WHEN BUFFER AVAILABLE
        JRST PUTOK          ;PROCESS OUTPUT ERROR
        JRST OUTERR         ;STORE THIS CHARACTER
PUTOK:  IDPB AC,OBUF+1      ;RETURN TO CALLER
        JRST@PUTCHR         ;GET THE ERROR STATUS
OUTERR: GETSTS 2,AC
          .
          .
          .
          .
```

4.10.5.3 Synchronization of Buffered I/O (CALL D, [SIXBIT/WAIT/]) - In some instances, such as recovery from transmission errors, it is desirable to delay until a device completes its I/O activities. The programmed operators

CALL D, [SIXBIT /WAIT/] and CALLI D, 10

return 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 spacing and I/O activity to finish on magnetic tape. 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.

4.10.6 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 monitor programmed operators. Table 4-7 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 used by the user. However, the file status is stored with each buffer so that the user can determine which bufferful produced an error. A more thorough description of bits 18 through 29 for each device is given in Chapters 5 and 6.

4.10.6.1 File Status Checking - The file status (refer to Table 4-7) 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, STATO, or SETSTS programmed operator) on the user's console and the console 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.

4.10.6.2 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.

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 console and the console 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.

Table 4-7
File Status Bits

Bit	Meaning
18	Improper mode (IOIMPM). Attempt to write on a hardware or software write-locked tape or file structure, or a software detected redundancy failure occurred. Usually set by monitor.
19	Hard device detected error (IODERR), other than hardware checksum, parity, or search 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 (IODTER). The data read or written has incorrect parity or checksum as detected by hardware (or by software on CDR,PTR). Usually set by monitor.
21	Block too large (IOBKTL). 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 (IOEOF). The user program has requested data beyond the last record or block, or USET1 has specified a block beyond the last data block of the file. Usually set by monitor.
23	I/O active (IOACT). 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. Stops the device after each buffer is filled. Usually set by user.
31	User word count. 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. Refer to Tables 4-5 and 4-6. Usually set by user.

4.10.7 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.

4.10.7.1 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).

4.10.7.2 CLOSE D,1 (Bit 35=1) - The closing of the output side of channel D is suppressed. Other actions of CLOSE are unaffected.

4.10.7.3 CLOSE D,2 (Bit 34=1) - The closing of the input side of channel D is inhibited; other actions of CLOSE are unaffected.

4.10.7.4 CLOSE D,4 (Bit 33=1)† - 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.

4.10.7.5 CLOSE D,10 (Bit 32=1)† - 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.

4.10.7.6 CLOSE D, 20 (Bit 31=1)† - The deleting of the NAME block in monitor core on CLOSE input is inhibited if a LOOKUP was done without subsequent INPUT. This bit is used by the COMPIL CUSP to retain the core block in order to speed up the subsequent access by the CUSP called by COMPIL.

4.10.7.7 CLOSE D,40 (Bit 30=1)† - 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 to create a file or a unique name and not supersede the original file.

Any combinations of the above bit settings are legal.

Example: Terminating a File

```
DROPDV: 0
        CLOSE 3,                ;JSR HERE
                                ;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
```

4.10.8 Device Termination

4.10.8.1 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. If the device is neither associated with another data channel nor assigned by the ASSIGN command (refer to Chapter 2), it is returned to the monitor's pool of available facilities. Control is returned to the user's program.

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

4.10.8.2 REASSIGN - This UUO reassigns a device under program control from the current job to a specified job and clears the directory currently in core, but does not clear the logical name assignment.

The call is:

```
MOVE AC, job number
MOVE AC+1 [SIXBIT /DEVICE/]
CALL AC, [SIXBIT /REASSI/]          ;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 a console (controlling) Teletype, or the logical name is duplicated or is a physical name in the system. A REASSIGN UUO that specifies job 0 deassigns the device.

4.10.9 Examples

4.10.9.1 File Reading - The following UUO sequence is required to read a file:

INIT	Establishes a file structure channel correspondence (or set of file structures-channel correspondence).
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.

4.10.9.2 File Writing - The following UUO sequence is required to write a file:

INIT	Forms file structure-channel correspondence (or set of file structures-channel correspondence).
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.
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.

4.10.10 Real-Time Programming

Refer to Chapter 8 for real-time programming and the RTTRP and TRPSET UUOs.

Chapter 5
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 4.9.4.2).

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, IB, B	35
Card Reader	CDR	-	CR10A 461(PDP-6)	INPUT, IN	A, AL, I, B	36
Console Teletype	CTY	-	LT33A, LT33B LT35A, LT37AC 626(PDP-6)	INPUT, IN OUTPUT, OUT	A, AL	23
Display	DIS	-	VR30, VP10 340B, 30	INPUT, OUTPUT	ID	Dump only
Line Printer	LPT		LP10C	OUTPUT	A, AL, I	34
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
Plotter	PLT	XY10	XY10A XY10B	OUTPUT, OUT	A, AL, I IB, B	46

†Buffer sizes are subject to change and should be calculated rather than assumed by user programs. A dummy INBUF or OUTBUF may be employed.

Table 5-1 (cont)
Nondirectory Devices

Device	Physical Number	Controller Number	Unit Number	Programmed Operators	Data Modes	Buffer Size (Octal)†
Pseudo-Teletype	PTY	-	-	INPUT, IN OUTPUT, OUT	A, AL	23
Teletype	TTY0, TTY1, ..., TTY177	DC10 DC68A 630(PDP-6)	LT33A, LT33B LT35A, LT37AC VT06	INPUT, IN OUTPUT, OUT TTCALL	A, AL	23

†Buffer sizes are subject to change and should be calculated rather than assumed by user programs. A dummy INBUF or OUTBUF 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 Table 5-2). This card is not punched for data modes other than ASCII. The header card has the same punch in all columns.

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 the same punch in all columns (e.g., 12-11-0-1-6-7-8-9 in columns 1 through 80).

5.1.2 Data Modes

5.1.2.1 A (ASCII) - 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 IOBKTL (bit 21 of status word) to be set. The CLOSE will punch the last partial card and then punch an EOF card.

Table 5-2
PDP-10 Card Codes

CHAR	PDP-10 ASCII	DEC 029	DEC 026	CHAR	PDP-10 ASCII	DEC 029	DEC 026
SPACE	040			@	100	8-4	8-4
!	041	11-8-2	12-8-7	A	101	12-1	12-1
"	042	8-7	0-8-5	B	102	12-2	12-2
#	043	8-3	0-8-6	C	103	12-3	12-3
$	044	11-8-3	11-8-3	D	104	12-4	12-4
%	045	0-8-4	0-8-7	E	105	12-5	12-5
&	046	12	11-8-7	F	106	12-6	12-6
'	047	8-5	8-6	G	107	12-7	12-7
(	050	12-8-5	0-8-4	H	110	12-8	12-8
)	051	11-8-5	12-8-4	I	111	12-9	12-9
*	052	11-8-4	11-8-4	J	112	11-1	11-1
+	053	12-8-6	12	K	113	11-2	11-2
,	054	0-8-3	0-8-3	L	114	11-3	11-3
-	055	11	11	M	115	11-4	11-4
.	056	12-8-3	12-8-3	N	116	11-5	11-5
/	057	0-1	0-1	O	117	11-6	11-6
0	060	0	0	P	120	11-7	11-7
1	061	1	1	Q	121	11-8	11-8
2	062	2	2	R	122	11-9	11-9
3	063	3	3	S	123	0-2	0-2
4	064	4	4	T	124	0-3	0-3
5	065	5	5	U	125	0-4	0-4
6	066	6	6	V	126	0-5	0-5
7	067	7	7	W	127	0-6	0-6
8	070	8	8	X	130	0-7	0-7
9	071	9	9	Y	131	0-8	0-8

Table 5-2 (Cont)
PDP-10 Card Codes

CHAR	PDP-10 ASCII	DEC 029	DEC 026	CHAR	PDP-10 ASCII	DEC 029	DEC 026
:	072	8-2 or 11-0†	11-8-2 or 11-0†	Z	132	0-9	0-9
;	073	11-8-6	0-8-2	[	133	12-8-2	11-8-5
<	074	12-8-4	12-8-6	\	134	11-8-7	8-7
=	075	8-6	8-3	]	135	0-8-2	12-8-5
>	076	0-8-6	11-8-6	↑	136	12-8-7	8-5
?	077	0-8-7 or 12-0†	12-8-2 or 12-0††	←	137	0-8-5	8-2

†Either is accepted on input, but 11-8-2 or 8-2 is punched.
††Either is accepted on input, but 12-8-2 or 0-8-7 is punched.

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 (see Table 5-2, PDP-10 Card Codes). The first card of any file (the header card) indicates the card code used (12-0-2-4-6-8 punch in column 1 for DEC029 card codes; 12-2-4-8 punch in column 1 for DEC026 card codes).

5.1.2.2 AL (ASCII Line) - The same as A mode.

5.1.2.3 I (Image) - Up to 26 2/3 data words are punched in columns 1 through 80. The buffer set up by the monitor depends on the mode used. Image binary causes exactly one card to be punched for each output. The CLOSE punches the last partial card, and then punches an EOF card.

5.1.2.4 IB (Image Binary) - Same as I.

5.1.2.5 B (Binary) - 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

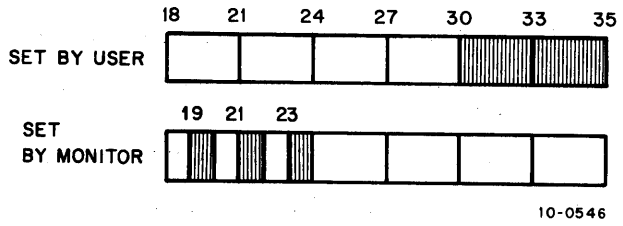
Following a CLOSE, an EOF card is punched. Both the header card of the file (identifies the card code used) and the EOF card are laced (i.e., all holes are punched) in columns 2 through 80 for easy file identification. These laced punches are ignored by the card reader service routine.

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 4.9.3.4).

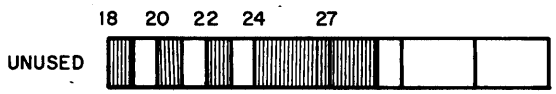
5.1.4 File Status (Refer to Appendix D)

The file status of the card punch is shown below.

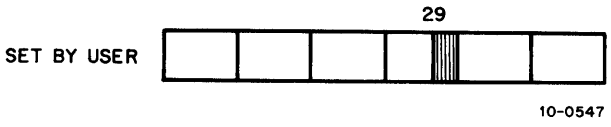
Standard Bits



Bit 19 - IODERR	Punch error
Bit 21 - IOBKTL	Reached end-of-card with data remaining in buffer.
Bit 23 - IDACT	Device is active.



Device Dependent Bits



Bit 29	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

A header card is the first card of the file and identifies the card code used. 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. The header card has a 12-2-4-8 in column 1.

An EOF card may have one of three forms: a 12-11-0-1 punched in column 1, a 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.

5.2.2 Data Modes

5.2.2.1 A (ASCII) - 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 card containing a 12-0-2-4-6-8 punch in column 1 is encountered, any following cards are translated as DEC029 codes (refer to Table 5-2) until the 029 conversion mode is turned off. The 029 mode is turned off either by a RELEASE command or by a card containing a 12-2-4-8 punch in column 1. Columns 2 through 80 of both of these cards are ignored.

5.2.2.2 AL (ASCII Line) - This mode is the same as the A mode.

5.2.2.3 I (Image) - All 12 punches in all 80 columns are packed into the buffer as 12-bit bytes. The first 12-bit byte is column 1. The last word of the buffer contains columns 79 and 80 as the left and middle bytes, respectively. The EOF card and the EOF button are processed as in the A mode. Cards are not split between two buffers.

5.2.2.4 IB (Image Binary) - This mode is the same as I.

5.2.2.5 B (Binary) - 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 IOIMPM (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 card and the EOF button are processed the same as in the A mode.

5.2.2.6 SI (Super-Image) - Super-image mode (data mode 110) 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).

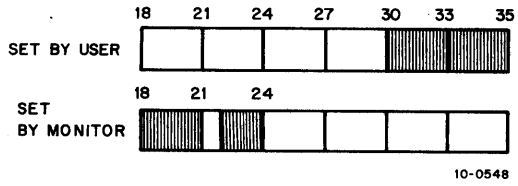
5.2.3 Special Programmed Operator Service

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 4.9.3.4).

5.2.4 File Status (Refer to Appendix D)

The file status of the card reader is shown below.

Standard Bits



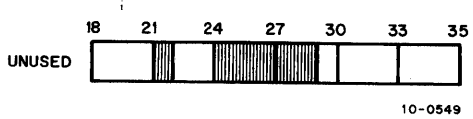
Bit 18 - IOIMPM 7-9 punch absent in column 1 of a presumed binary card. The card reader is stopped.

Bit 19 - IODERR Photocell error, card motion error, data missed. The card reader is stopped.

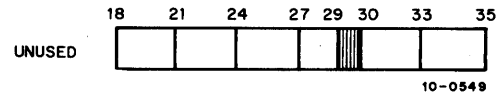
Bit 20 - IODTER Computed checksum is not equal to checksum read on binary card. The card reader is stopped.

Bit 22 - IOEOF EOF card read or EOF button pressed.

Bit 23 - IOACT Device is active.



Device-Dependent Bits



Bit 29 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 ID (IMAGE DUMP - 25), 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

The monitor service routine for the Type 30, VR30 and VP10 maintains a flicker-free picture on the display during time-sharing; therefore, the picture data must be available for display at least every two jiffies. This time requirement necessitates that the display data remain in core; also, the user program must remain in core. To minimize swapping of other programs and to make available a larger block of free core for other users, the user program is shuffled toward the top of core between pictures.

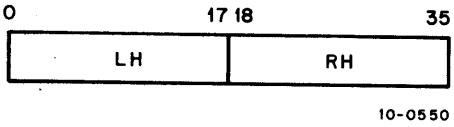
5.3.3 Display UUOs

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:



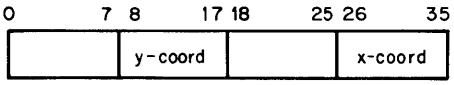
For the Type 30, VR30 and VP10 Display:

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:



10-0551

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:

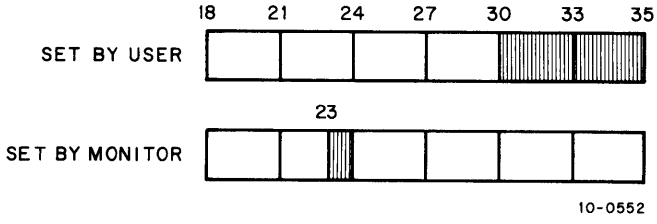
```
          OUTPUT   D, LIST          ;OUTPUT DATA POINTED
                                    ;TO BY POINTER IN LIST

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.

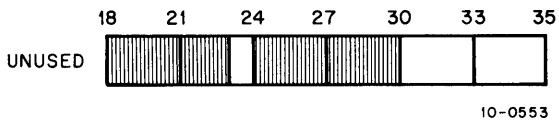
5.3.4 File Status (See Appendix D)

The file status of the display is shown below.

Standard Bits



Bit 23 - IOACT Device is active.



Device Dependent Bits - None.

5.4 LINE PRINTER

The device mnemonic is LPT; the buffer size is 34_8 (33_8 data) words.

5.4.1 Data Modes

5.4.1.1 A (ASCII) - 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 AL (ASCII Line) - This mode is exactly the same as A 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 I (Image) - This mode is the same as A (ASCII) mode.

5.4.2 Special Programmed Operator Service

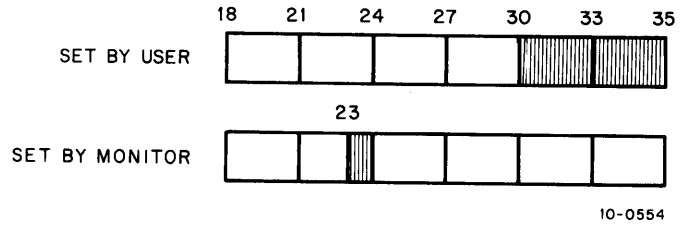
The first output programmed operator of a file and the CLOSE at the end of a file cause an extra form-feed to be printed to keep files separated.

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 4.9.3.4).

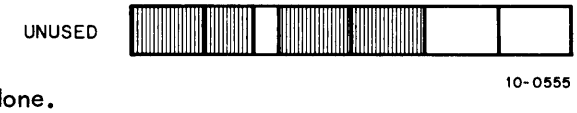
5.4.3 File Status (See Appendix D)

The file status of the line printer is shown below.

Standard Bits



Bit 23 - IOACT 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.

5.5.1 Data Modes

5.5.1.1 A (ASCII) - 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 AL (ASCII Line) - The mode is the same as A.

5.5.1.3 I (Image) - The mode is the same as A, but data consists of 36-bit words.

5.5.1.4 IB (Image Binary) - The mode is the same as I.

5.5.1.5 DR (Dump Records) - 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.

5.5.1.6 D (Dump) - 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 4.4.1.2 for command list format).

5.5.2 Magnetic Tape Format

Magnetic tape format can be generally 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. When an input dump file is closed, the I/O service routine automatically skips to the next EOF mark.

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 4.9.3.4).

A special programmed operator called MTAPE 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 5-3.

Table 5-3
MTAPE Functions

Function	Action
0	No operation; wait for spacing and I/O to finish.
1	Rewind to load point
11	Rewind and unload†
7	Backspace record
17	Backspace files; implemented by a series of backspace record operations.
3	Write EOF
6	Skip one record
13	Write 3 in. of blank tape
16	Skip one file; implemented by a series of skip record operations.
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.
100	Initialize for Digital-compatible 9-channel††
101	Initialize for industry-compatible 9-channel tape†††

†On the 516 Magnetic Tape Control, this function is implemented as such, but earlier types of transports consider it only as a REWIND function.

††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 an MTAPE D, 101 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 last 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.

5.5.3.1 Use of the MTAPE Operator - 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, indeed, 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:

```
MTAPE MT,17          ;Backspace file
CALLI MT,WAIT        ;Wait for completion
STATO MT,4000        ;Beginning of tape?
MTAPE MT,16          ;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 CALLI MT, WAIT 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.

5.5.4 9-Channel Magtape

Nine-channel magtape may be written and read in two ways: normal Digital-compatible format, and industry-compatible format.

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.

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.

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.

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 error checking information.

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

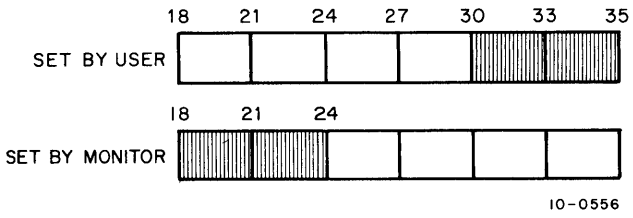
For data word in core, four magnetic tape bytes carry four 8-bit bytes from the data word. Parity bits are obtained as shown above when reading. The rightmost four bits (32-35) are ignored on writing.

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. Byte pointers and byte counts in buffer header have to be changed by the user to operate on eight-bit bytes.

5.5.5 File Status (refer to Appendix D)

The file status of the magnetic tape is shown below.

Standard Bits



Bit 18 - IOIMPM — Unit was write-locked when output was attempted, or illegal operation was specified to the magnetic-tape control.

Bit 19 - IODERR — Data was missed, tape is bad, or transport is hung.

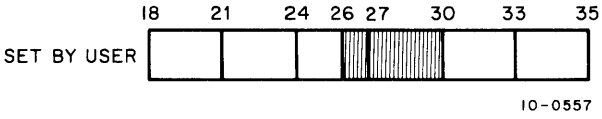
Bit 20 - IODTER — Parity error.

Bit 21 - IOBKTL — Record read from tape exceeds buffer size.

Bit 22 - IOEOF — EOF mark encountered. No special character appears in buffer.

Bit 23 - IOACT — Device is active.

Device Dependent Bits



Bit 26 - IOPAR — 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.

Bits 27-28 — I/O density. 00 = System standard. Defined at MONGEN time.
01 = 200 bits/in.
10 = 556 bits/in.
11 = 800 bits/in.

Bit 29 - IONRCK — 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 - IOBOT — I/O beginning of tape. Unit is at beginning of tape mark.

Bit 25 - IOTEND — 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 A (ASCII) - The eighth hole is punched for all characters. 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 AL (ASCII) Line - The mode is the same as A mode. Format control must be performed by the user's program.

5.6.1.3 I (Image) - Eight-bit characters are punched exactly as they appear in the buffer with no additional processing.

5.6.1.4 IB (Image Binary) - 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 B (Binary) - 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

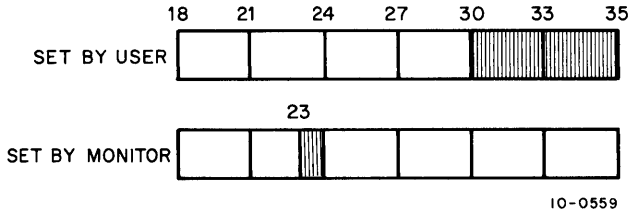
The first output programmed operator of a file causes approximately two fanfolds of blank tape to be punched as leader. Following a CLOSE, an additional fanfold of blank tape is punched as trailer. No EOF character is punched automatically.

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 4.9.3.4).

5.6.3 File Status (Refer to Appendix D)

The file status for the paper-tape punch is shown below.

Standard Bits



Bit 23 - IOACT Device is active.

Device Dependent Bits - None

5.7 PR-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 A (ASCII) - 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 AL (ASCII Line) - Character processing is the same as for the A mode. The buffer is terminated by LINE FEED, FORM, or VT.

5.7.1.3 I (Image) - 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 IB (Image Binary) - 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 B (Binary) - 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 4-7) 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.

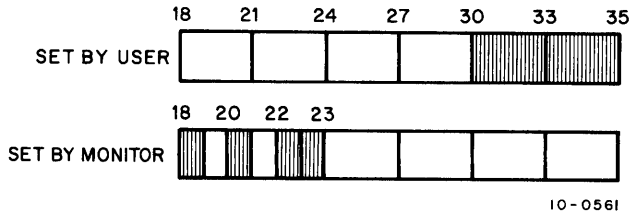
5.7.2 Special Programmed Operator Service

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 4.9.3.4).

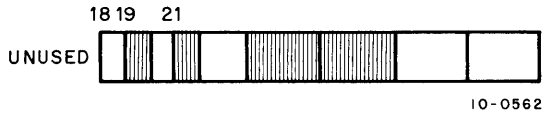
5.7.3 File Status (Refer to Appendix D)

The file status of the paper-tape reader is shown below.

Standard Bits



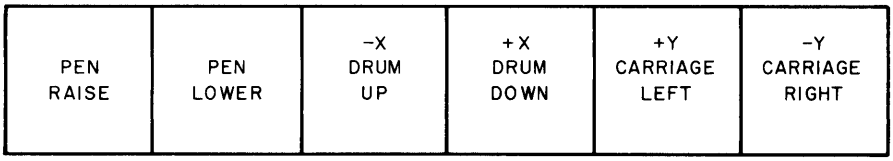
Bit 18	Binary block is incomplete.
Bit 20	Bad checksum in binary mode.
Bit 22 - IOEOF	Physical end of tape is encountered. No character is stored in the buffer.
Bit 23 - IOACT	Device is active.



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 A (ASCII) - 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 AL (ASCII Line) - This mode is identical to the A mode.

5.8.1.3 I (IMAGE) - Six 6-bit characters per word are transmitted to the plotter exactly as they appear in the buffer.

5.8.1.4 B (BINARY) - This mode is identical to the I mode.

5.8.1.5 IB (IMAGE BINARY) - This mode is identical to the I mode.

5.8.1.6 DR (DUMP RECORDS) - This mode is not available.

5.8.1.7 D (DUMP) - This mode is not available.

5.8.2 Special Programmed Operator Service

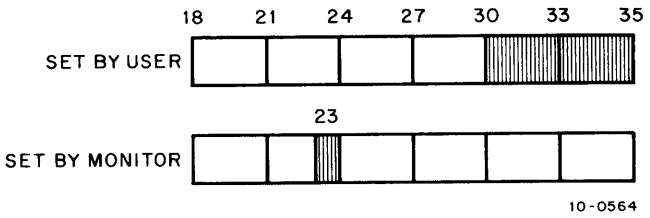
The first OUTPUT operator causes the plotter pen to be lifted from the paper before any user data is sent to the plotter. The CLOSE operator causes the plotter pen to be lifted after all user data is sent to the plotter. These two pen-up commands are the only modifications the monitor makes to the user 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 4.9.3.4).

5.8.3 File Status (Refer to Appendix D)

The file status of the plotter is shown below.

Standard bits



Bit 23 - IOACT Device is active.

Device dependent bits - None

5.9 TELETYPE

The device mnemonic is TTY0, TTY1, ..., TTY176, TTY177, CTY; the buffer size is 23_8 (20_8 data) words.

Line number n of the Type 630 Data Communications System, Data Line Scanner DC10, PDP-8 680 System, or PDP-8/I DC68A System is referred to as TTYn. The console Teletype is CTY. The Timesharing monitor automatically gives the logical name TTY to the user's console when a job is initialized.

Teletype device names are assigned dynamically. For interconsole communication by program, one of the two users must type DEASSIGN TTY to make the Teletype available to the other user's program as an I/O device. Typing ASSIGN TTYn is the only way to reassign a Teletype that has been deassigned (refer to TALK command, Table 2-2).

Two Teletype routines are provided: a newer, full-duplex software routine and an older, half-duplex software routine. The full-duplex software is recommended.

With a full-duplex Teletype 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 the 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. The echo-checking of typed-in characters disappears because the keyboard and the printing operations are independent. 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.

Programs waiting for Teletype output are awakened eight characters before the output buffer is empty, causing them to be swapped in sooner and preventing pauses in typing. Programs waiting for Teletype input will be awakened ten characters before the input buffer is filled, thus reducing the possibility of lost typein.

5.9.1 Data Modes

5.9.1.1 Full-Duplex Software A (ASCII) and AL (ASCII Line)

The input handling of all control characters is as follows. (All are passed to program except as noted below.)

000	NULL	Ignored on input, suppressed on output.
001	↑ A	Echoes as ↑A; passed to program.
002	↑ B	Complements switch controlling echoing, not passed to program. Used on local-copy dataphones and TWX's. (No special action with 5.02 monitors and later monitors.)
003	↑ C	The user's console is switched to monitor mode the next time input is requested by the program. Two successive ↑ C's cause the console to be immediately switched to monitor mode.
004	↑ D (EOT)	004 passed to program. 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		Complements switch controlling translation of lower case letters to upper case. Used when lower case input is desired to programs. Not sent to program, but program can sense the state of this switch by the TTCALL UUO. (No special action with 5.02 monitors and later monitors.
007	↑G	(Bell)	007 passed to program, and is a break character.
010	↑H	(Backspace)	Acts as a RUBOUT, unless either DDT mode or full character set mode is true, or the ↑F switch (or lower case mode in 5.02 monitors and later monitors) is on. In these cases, 010 is sent to the program.
011	↑I	(TAB)	011 passed to program. Echoed as spaces if ↑P switch is on. (In 5.02 monitors and later, this character is determined by the TTY command.) These spaces are not passed to program.
012	↑J	(Linefeed)	Break character; no other special action.
013	↑K	(Vertical tab)	013 passed to program. Echoes as four linefeeds if ↑P switch is on. (In 5.02 monitors and later monitors this character is determined by the TTY command.) Is a break character. These linefeeds are not passed to program.
014	↑L	(Form)	014 passed to program. Echoes as 8 linefeeds if the↑ P switch is on. (In 5.02 monitors and later models, this character is determined by the TTY command. Is a break character. These linefeeds are not passed to program.
015	↑M	(Carriage return)	If Teletype is in paper-tape input mode, 015 is simply passed to program; otherwise, 015 supplies a linefeed echo, and is passed to program as a CR and LF, and is a break character (due to LF).
016	↑N		No special action.
017	↑O		Complements output suppression bit allowing user to turn output on and off. INPUT, INIT, and OPEN clear the output suppression bit. Not passed to program. Echoed as ↑O followed by carriage return-linefeed.
020	↑P		Does not appear in the input buffer. When switch is off, TAB, VT, and FF are echoed normally. When switch is on, TAB is converted to spaces, and VT and FF to linefeeds to simulate the action indicated. The ↑P switch should be turned on if the user's terminal does not act on TAB, VT, and FF. (No special action with 5.02 monitors and later.)
021	↑Q	(XON)	Starts paper-tape mode, as described above. Passed to program.
022	↑R	(TAPE)	No special action.
023	↑S	(XOFF)	Ends paper-tape mode, as described above; is passed to program.
024	↑T	(NO TAPE)	No special action.
025	↑U		Deletes input line back to last break character. Typed back as ↑U followed by carriage return-linefeed.

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 Teletype input. Echoes as ↑ Z followed by carriage return-linefeed. Is a break character. Appears in buffer as 032.
033	↑[(ESC)	The current ASCII altmode, but is translated to 175 before being passed to the program, unless in full character set mode (bit 20 in INIT). 175 is the 1963 altmode; echoes as a dollar sign; is always 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 ↑ F switch (or lower case mode in 5.02 monitors and later monitors) is set. Echoes as upper case if translated to upper case.
175 and 176		Old versions of altmode; refer to description of ESC (033).
177		RUBOUT or DELETE: a. Completely ignored if in paper-tape mode (XON). b. Break character, passed to program if either DDTmode 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 back-slashes. 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).

5.9.1.2 Half-Duplex Software A(ASCII) - If, during output operations, an echo-check failure occurs (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 console is immediately placed in monitor mode. If it is ↑O, all Teletype output buffers that are currently full are ignored, thus cutting the output short. All other characters cause the service routines 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.

The buffer is terminated when it is full or when the user types ↑Z.

5.9.1.3 Half-Duplex Software AL(ASCII Line) - The mode is the same as ASCII mode (usually preferred) with the addition that the input is terminated by a CR/LF pair, FF, VT, or ALTMODE.

5.9.1.4 I (Image) - Image mode is legal for Teletype input and output, except for Teletypes controlled by pseudo-Teletypes (refer to Paragraph 5.10).

Since, 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 Teletype from image mode, no typein will release the user from this state; consequently, the Teletype 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 15 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 Teletype 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 Teletype status word.

NOTE

Since 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 outputted. 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

when allowing output to plotting devices by FORTRAN subroutines. For this case, an additional TTCALL function has been added (refer to Paragraph 5.9.3).

5.9.2 DDT Submode

To allow a user's program and the DDT debugging program to use the same Teletype without interferring with one another, the Teletype service routine provides the DDT submode. This mode does not affect the Teletype 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 Teletype 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 Teletype service routine supplies buffers full of characters at the same time.

To input characters in DDT mode, user 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 Teletype 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.

5.9.3 Special Programmed Operator Service

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 typein buffer
12	CLRBFO	Clear typeout 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.

5.9.3.1 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.

5.9.3.2 OUTCHR ADR or TTCALL 1, ADR - This command outputs a character to the Teletype from 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.

5.9.3.3 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
```

5.9.3.4 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

5.9.3.5 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).

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, .

5.9.3.6 INCHSL 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.

5.9.3.7 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 Teletype line. 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.

The normal answer format is as follows:

Right half of ADR:	The line number.
Left half of ADR:	Bits, as follows:

Bit	Meaning
0	Line is a pseudo-Teletype
1	Line is the CTY.
2	Line is the display console.
3	Line is the dataset data line.
4	Line is a dataset control line.
5	Line is half-duplex.
11	A line has been typed in by the user.
12	A rubout has been typed.
13	Lower case input mode is on.
14	Teletype has tabs.
15	Teletype input is not echoed.
16	Control Q (paper-tape) switch is on.
17	Line is in a talk ring.

5.9.3.8 SETLCH ADR or TTCALL 7,ADR - the bits described for GETLCH. They may be changed only for the controlling Teletype. Bits 14, 15, and 16 can be modified.

Example:

```
SETO    AC,0
TTCALL  6,AC
TLZ     AC,BIT 13
TLO     AC,BIT 14
TTCALL  7,AC
```

5.9.3.9 RESCAN ADR or TTCALL 10,0 - This command is intended for use only by the COMPIL CUSP. It causes the input buffer to be rescanned from the point where the last command began. Obviously, if it 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.

5.9.3.10 CLRBFI ADR 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, which he might have typed ahead, to be executed).

5.9.3.11 CLRBFO ADR 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.

5.9.3.12 SKPINC ADR 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, which wants to occasionally check for input and, if any, go off to another routine (such as FORTRAN Operating System) to actually do the input.

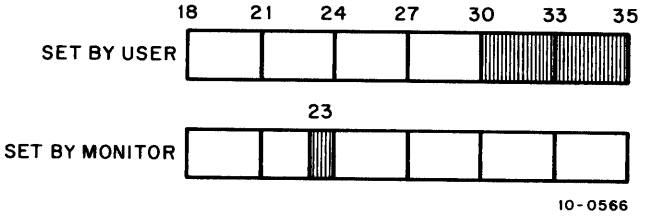
5.9.3.13 SKPINL ADR or TTCALL 14,0 - This command is the same as SKPINC, except that a skip occurs if the user has typed at least one line.

5.9.3.14 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. This function is available in 5.02 monitors and later monitors.

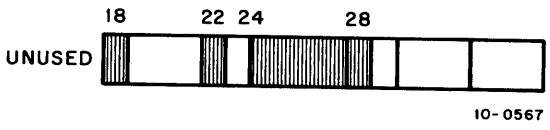
5.9.4 File Status (Refer to Appendix D)

The file status of the Teletype is shown below.

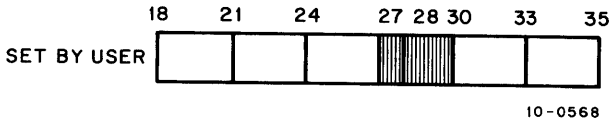
Standard Bits



Bit 23 - IOACT Device is active.



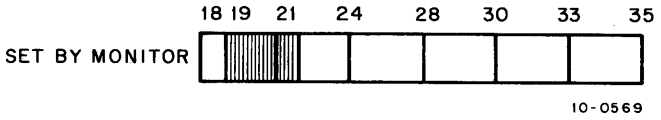
Device Dependent Bits



Bit 27 Suppresses echoing of ($) on the Teletype.

Bit 28 Suppresses echoing on the Teletype.

Bit 29 Full character set. Pass all characters except ↑C with no special character processing.



Bit 19 Ignore interrupts for three-fourths of a second.

Bit 20 Echo failure has occurred on output.

Bit 21 Character was lost on typein.

5.9.5 Paper-Tape Input from the Teletype (Full-Duplex Software)

Paper-tape input is possible from a Teletype equipped with a paper-tape reader which is controlled by the XON (↑Q) and XOFF (↑S) characters. When commanded by the XON character, the Teletype 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 Teletype 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
*↑S
```

5.9.6 Paper-Tape Output at the Teletype (Full-Duplex Software)

Paper-tape output is possible on any Teletype-mounted paper-tape punch, which is controlled by the TAPE (AUX ON) and TAPE (AUX OFF) characters. The punch is connected in parallel with the keyboard printer, and therefore, when the punch is on, all characters typed 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 Teletype 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 Teletype 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.

5.10 PSEUDO-TELETYPE

The device mnemonic is PTY0, PTY1, ..., PTYn. (The number of pseudo-Teletypes is specified at MONGEN time.) The buffer size is 23_8 (20_8 data)words.

5.10.1 Concepts

Each job in the PDP-10 timesharing system 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 Teletype there is a block of core in the monitor, containing information about the physical Teletype and including two buffers as the link between the physical Teletype and the job. It is through these buffers that the Teletype sends input to the job, and the job returns output to the Teletype.

Sometimes it is desirable to allow a job in the PDP-10 timesharing system 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-Teletype (PTY). The PTY is a simulated Teletype and is not defined by hardware. Like hardware-defined Teletypes, 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 Teletype uses its block of core. Figure 5-1 shows the parallel between a hardware-defined Teletype and a software-defined PTY.

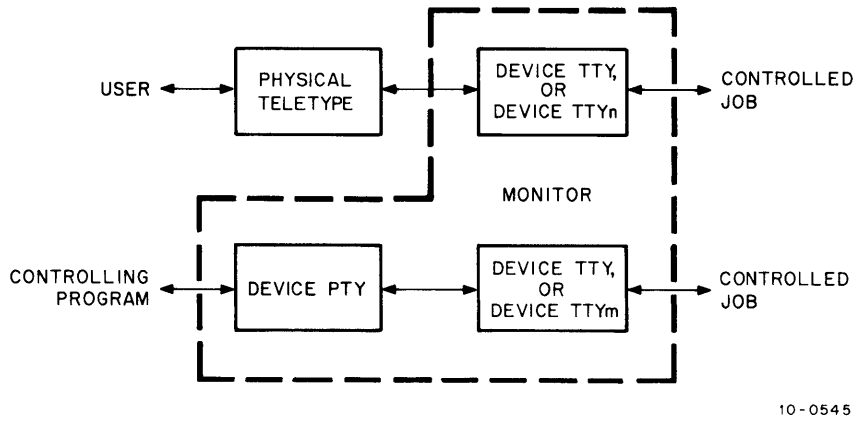


Figure 5-1 Pseudo-Teletype

The controlling program, most commonly the batch processor, uses the PTY in the same way as 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 SLEEP UUO and the status bits of the PTY.

5.10.2 The SLEEP UUO

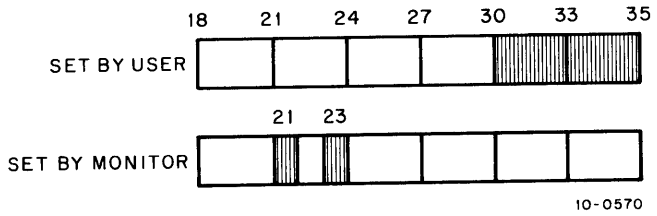
If the controlling program waits for activity in the controlled job, it delays the SLEEP UUO (refer to Paragraph 4.3.4.1). When the controlling program continues after a SLEEP UUO, it checks for activity on the PTY via the status bits. If there is no activity, it checks the job run time or other criteria to determine whether or not the job should be interrupted. If the job should be interrupted, the controlling program may output to the PTY two control-C characters to stop the job. (A user stops a running job in the same way.) If the job should not be interrupted, the controlling program should repeat the SLEEP UUO.

Unnecessary delays might result if activity occurred on a PTY while the controlling job was sleeping. These delays are avoided because 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.

5.10.3 File Status (Refer to Appendix D)

The file status of the pseudo-Teletype is shown below.

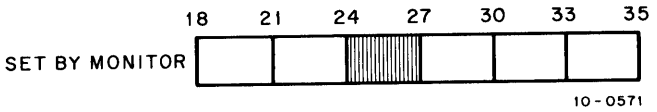
Standard Bits



Bit 21 - IOBKTL

Bit 23 - IOACT Device is active.

Device Dependent Bits



Bit 24 - IOPTW Job is in an input wait. The controller performs an OUTPUT to the PTY.

Bit 25 - IOPTRE The TTY buffer has output to be read by an INPUT from the PTY.

Bit 26 - MONMOD 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.

5.10.4 Special Programmed Operator Service

5.10.4.1 OUT, OUTPUT UUOs - The first OUTPUT operation after an INIT or OPEN causes the special actions of the RELEASE UUO (refer to Paragraph 5.10.4.3) 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 IOPTW bit is cleared.

c. The MONMOD 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 IOBKTL 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.

5.10.4.2 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 IOPTRE bit remains set and another INPUT should be done. If the output buffer of the TTY is exhausted by the INPUT UUO, the IOPTRE bit is cleared.

5.10.4.3 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.

5.10.4.4 JOBSTS UUO - This UUO provides status information about device 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.

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:

Bit	Explanation
Bit 0 = 1	Job number is assigned.
Bit 1 = 1	Job is logged in.
Bit 2 = 1	TTY is at monitor level.
Bit 3 = 1	TTY output is available.
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.
Bit 5 = 1	JACCT is set. In particular, ↑C ↑C will not work.
Bits 18-35	Job number being checked or 0 if no job number is assigned.

5.10.4.5 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.

Chapter 6
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 4.9.4.2).

Table 6-1
Directory Devices

Device	Physical Name	Controller Number	Unit Number	Programmed Operators	Data Modes	Buffer Sizes (Octal †)
DECtape	DTA0, DTA1, ...,DTA7	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 dummy INBUF or OUTBUF may be employed.

Table 6-1 (Cont)
Directory Devices

Device	Physical Name	Controller Number	Unit Number	Programmed Operators	Data Modes	Buffer Sizes (Octal †)
Mass-Disk File	MDA	RA10	RB10B	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 dummy INBUF or OUTBUF may be employed.

6.1 DECTAPE

The device mnemonic is DTA0, DTA1, ..., DTA7; the buffer size is 202_8 words (177_8 user data, 200_8 transferred).

6.1.1 Data Modes

Two hundred words are written. The first word is the link plus word count. The following 177 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. 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 IOBKTL 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).

The first 83 words (1 through 82_{10}) of the directory block contain slots for each of the 577 blocks 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. The next 22 words of the directory block (words 83 through 104_{10}) 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 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 is 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 date the file was last updated, according to the formula: ((year-1964) * 12 + (month-1)) * 31 + day -1

Word 127_{10} of the directory block is unused.

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.

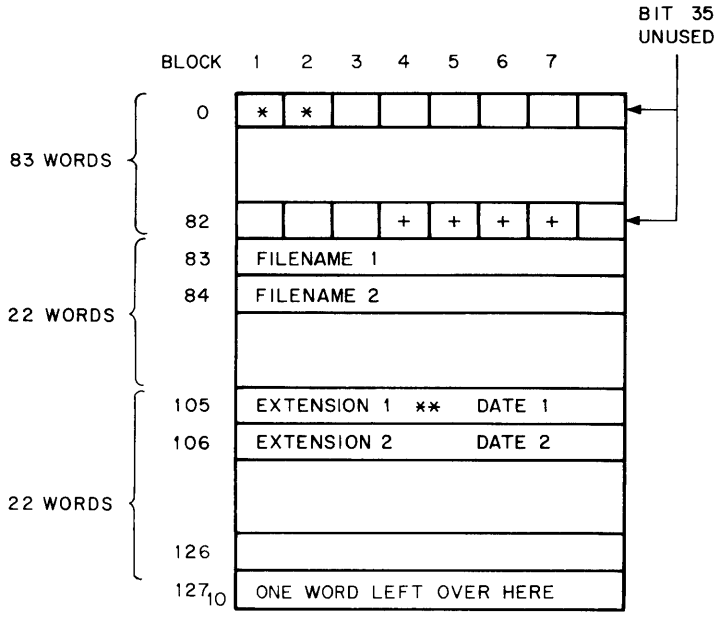


Figure 6-1 DECtape Directory Format

6.1.4 DECtape File Format

A file consists of any number of DECtape blocks.

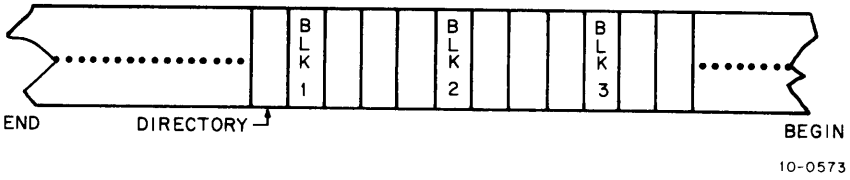


Figure 6-2 Format of a File on Tape

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 127 words of memory.

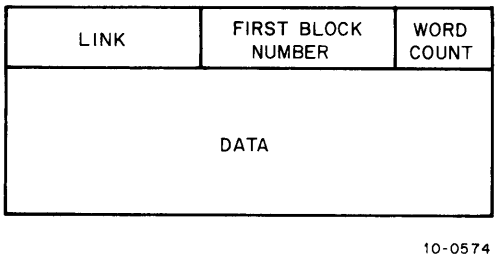


Figure 6-3 Format of a DECtape Block

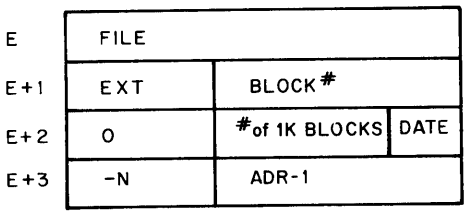
6.1.4.1 Block Allocation - Normally, blocks are allocated by starting with the first free block nearest the directory and going backwards to the front of the tape (block 0). When the end of the tape is reached, the direction of the scan is reversed. Blocks are not written contiguously; rather they are 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:



10-0575

where

FILE is the filename in SIXBIT ASCII.

EXT is the filename extension in SIXBIT ASCII.

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).

DATE is the date the file was originally created in the format of the DAYTIME programmed operator (bits 24-35).

-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 only 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. 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).

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 = first block #
E + 2	I	-	E + 2	V	LH = 0 RH = # of 1K blocks (Bits 18-23)†† 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, which 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. If a match is found, the new entry replaces the old entry, the old file space is reclaimed immediately, and the monitor records the file information. This process is called superseding and differs from disk in that, because of the small size of DECtape, the space is reclaimed before the file is written rather than after.

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	SIXBIT/EXT/	E + 1	V	LH = SIXBIT/EXT
E + 2	A	RH = desired creation date or 0. (0 implies current date)	E + 2	V	RH = creation date
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 to select the file that is to be RENAMED and a CLOSE. The error return is given if a LOOKUP has not been done.

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
RENAME    D,E1

or

ENTER     D,E
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.

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/	E + 1	V	LH = SIXBIT/EXT/ RH = error code on error return 0 = old name not found 4 = rename to existing name
E + 2	I	-	E + 2	I	-
E + 3	I	-	E + 3	I	-

†A = argument from user program, V = value from monitor, I = ignored

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 (refer to Chapter 2).

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. Input operations on the DECtape must not be active; otherwise, the user has no way of determining which buffer contains block E.

6.1.6.2 USETO D, E - The USETO programmed operator sets the DECtape on channel D to output block E next. USETO waits until the device is inactive before setting up the new output block number.

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 PDP-10 Software Notebook).

6.1.6.4 CALL AC, [SIXBIT/UTPCLRI] 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.

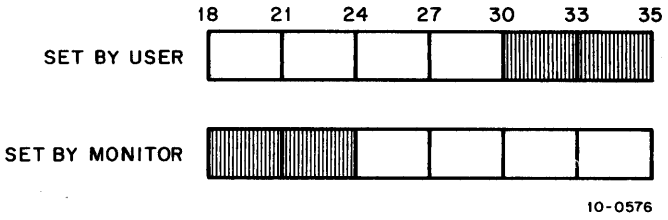
6.1.6.5 MTAPE D, 1 and MTAPE D, 11 - MTAPE D,1 rewinds the DECtape and moves it into the end zone at the front of the tape. MTAPE D,11 rewinds and unloads the tape, pulling the tape competely onto the left-hand reel. These commands affect only the physical position of the tape, not the logical position. 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 4.9.3.4).

6.1.7 File Status (Refer to Appendix D)

The file status of the DECtape is shown below.

Standard Bits



Bit 18 - IOIMPM — An attempt was made to read block 0 in nonstandard dump mode.

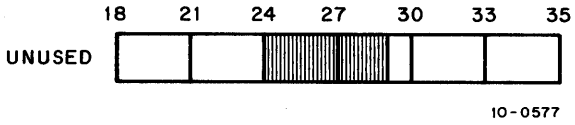
Bit 19 - IODERR — Data was missed.

Bit 20 - IODTER — Parity error.

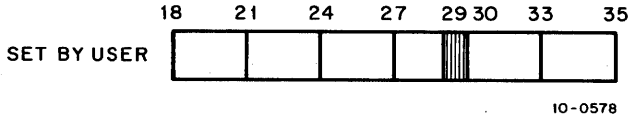
Bit 21 - IOBKTL — Block number is too large or tape is full on OUTPUT.

Bit 22 - IOEOF — EOF mark encountered on input. No special character appears in buffer.

Bit 23 - IOACT — Device is active.



Device Dependent Bits



Bit 29 — 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

Bit 29 (Cont)	number as the tape may be composed of blocks shorter than 200 words. The ENTER, LOOKUP, and UTPCLR 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

If 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 OK?

is given. When the situation has been rectified, CONT may be typed to proceed.

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 SRCCOM
*TTY:←DTA1:P1,DTA1:P2
```

SRCCOM 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 Paragraph 4.10.5.1. 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 accessing 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 simultaneously 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 creating 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 timesharing monitor services all disks and drums. This set of routines, FILSER, interprets and operates upon file structures, processes 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 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 bit is on, it indicates that the corresponding cluster is filled with data (all blocks on the disk are filled when any information is written on them) or is bad or nonexistent; if the bit is off, it indicates that the corresponding cluster is empty, or available to be written on.

6.2.2.3 File Directories - There are two levels of directories in each file structure:

a. The master file directory (MFD)
b. The user file directory (UFD).

The master file directory consists of two-word entries; the entries are the names of the user file directories. 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 for individual user file directories.

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).

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. The term disk directory refers to all the UFDs for a particular project-programmer number. 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.

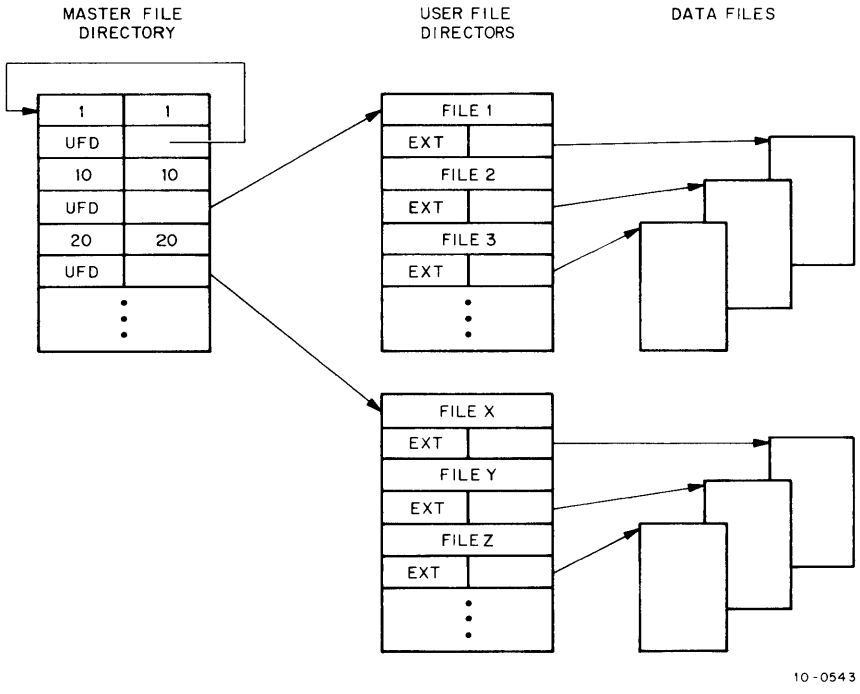


Figure 6-4 Basic Disk File Organization for Each File Structure

To improve disk access and core searching times, only UFD names are kept in the MFD (project-programmer number 1,1). All CUSPs and monitor file structure files are contained in another project-programmer number directory called the system library. For convenience 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.

6.2.2.4 File Format - All disk files (including MFD and UFDs) are composed of two parts:

a. pure data
b. information needed by the system to retrieve this data.

Each data block contains exactly 200 (octal) 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, which 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 and the number of positionings necessary for random access is reduced.

For recovery purposes, a copy of the retieval 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 I).

6.2.3 Access Protection

Nine bits of the retrieval information of a file are used to specify the protection of the file. This procedure is necessary because a 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.

Users are divided into three categories: the owner of the file, the members with the owner's project number, and all others. Each UFD in a file structure is associated with a distinct project-programmer number pair.

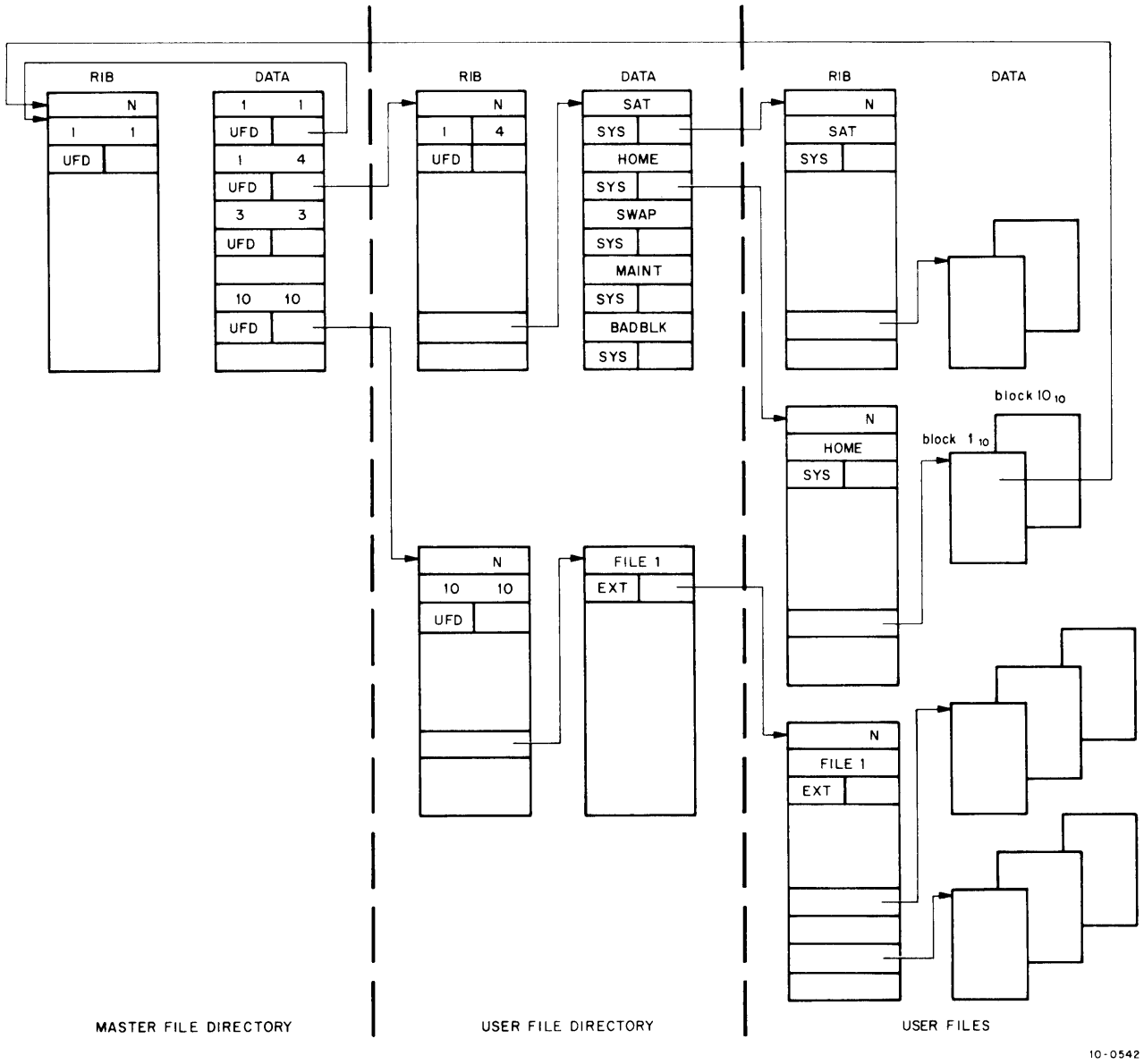


Figure 6-5 Disk File Organization

Any user who is logged in with the same programmer number is considered to be the owner of the files in that UFD; therefore, the same programmer numbers can be assigned to different users in different projects. However, a user who is working on more than one project cannot have the same access to all files he has written. Some installations may decide that a user is not an owner unless both the project and programmer number the user is logged in under match the pair associated with the UFD. This decision is made at monitor generation time with the MONGEN program.

A user is allowed project access if the programmer number he is logged in under is different from the one associated with the UFD, but his project number is the same as the one associated with the UFD.

The three bits associated with each category of users are encoded as follows:

Code	Access Protection
7	No access privileges. File may be looked up if the UFD permits.
6	Execute only.
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 greatest protection a file can have is 7, and the least protection is 0. It is always possible for the owner of a file to change the access protection associated with that file even if the owner-protection field is set to 0; thus, 0 and 1 are equivalent in the owner field. Access protection can be changed by executing a RENAME UUO or by using the RENAME switch in PIP as follows:

```
.R PIP
*FILE.EXT<NEW PROT>/R←FILE.EXT
```

When a file is created with a protection code of 000, 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 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.

6.2.3.1 UFD 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 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 privileges, users are divided into the same three categories as for files. Each category has three independent bits:

Code	Access Privileges
4	Allow LOOKUPs in UFD.
2	Allow CREATEs in UFD.
1	Allow the UFD to be read as a file.

The owner is permitted to control access to his own UFD. It is always legal for the owner to issue a RENAME to change the protection of his UFD. 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
b. Delete UFDs
c. Set privileged LOOKUP, ENTER, and RENAME arguments
d. Ignore file protection codes.

6.2.4 Disk Quotas

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. This is not a guaranteed amount of space, and 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 amount 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 IOBKTL 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 CUSP 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.

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. The recommended names for the file structures in the public pool are DSKA, DSKB, ..., DSKN (in order of decreasing speed). Names for private file structures may be any name starting with a letter. The system manager should ensure that private file structure names do not conflict with any device or file structure name or its abbreviation.

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 separate files on to 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, MD

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, MDA

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 RP01 and RP02 Memorex disk packs on RP10 control.
DPB0, ..., DPB7	Mixture of RP01 and RP02 Memorex disk packs on second RP10 control.
MDA0	Single-positioner Bryant mass disk (RB10B) on RA10 control.
MDB0	Single-positioner Bryant mass disk (RB10B) on second RA10 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 separate units 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; however, 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
DSKB	=	FHA2, 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, 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	DSKB	FHB0
DP	DSKC, 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 (available in 5.02 monitors and later monitors).

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 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 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:

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 state. 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 the file is placed on the first structure in the search list that has space and does not have the do not create flag set.

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 6.2.8.3) 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.

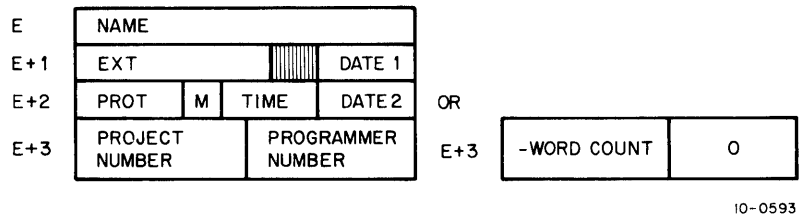
As a standard practice, user programs should read, create, and supersede (new file with same filename) files on different user channels. However, for compatibility with DECtapes, it is possible to read and create, or read and supersede, two files on the same user channel as long as all OUTPUTs and the 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 ASCII, if a UFD, or the project-programmer number in binary, if a MFD.

EXT is the filename extension in SIXBIT ASCII, if a UFD, or the word UFD, if a MFD.

DATE 1 is the date the file was last referenced (RENAME, ENTER, or INPUT) in the format of the DATE UUO (bits 24-35).

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 (bits 13-23).

DATE 2 is the date (in the same format as DATE 1) that the file was originally created (bits 24-35).

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 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.
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, DATE 2	If both of these are 0, the monitor supplies the current date and time as the creation date and time for the file. If either is nonzero, the monitor uses the TIME and DATE 2 supplied by the user E + 2; thus, files may be copied without changing the original creation time and date.
PROJECT NUMBER PROGRAMMER NUMBER	If both of these are 0, the project-number and programmer-number (binary) under which the user is logged in is supplied by the monitor. Otherwise, the monitor uses the project-number and programmer-number supplied by the user in E + 3. However, it is generally not possible to create (ENTER) files in another user's area of the disk, because UFDs are usually protected against creation with all but the owner.

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. Refer to Appendix E for the error codes returned on the ENTER UUO.

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 4.10.7.7). 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. If a later version of the file is being written, the old version pointed to by the UFD read.

NAME	The same as on an ENTER.
EXT	The same as on an ENTER.
DATE 1, PROT, M, TIME, 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 both of these are 0, the project-number and programmer-number (binary) under which the user is logged in is supplied by the monitor. Otherwise, the monitor uses the project-number, programmer-number supplied by the user in E+3. 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.

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. 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/protection of a file, or to delete a file from the disk. Locations E through E+3 are as described for ENTER. To RENAME a file, a LOOKUP or ENTER must first be done to identify the file for the RENAME UUO. CLOSE is optional because RENAME performs a CLOSE. In fact, to minimize disk accesses, a RENAME should not be preceded by a CLOSE.

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.

The error codes in the right half of E+1 are the same as for ENTER (refer to Appendix E).

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. If E+3 has a different project-programmer number than the one in which the file is LOOKUPed or ENTERed, the monitor

deletes the directory entry from the old UFD and inserts the directory entry in the new UFD, provided the user has the privileges to delete files from the old UFD, and to create files in the new UFD. This is an efficient way to move a file from one directory to another, since no I/O needs to be done on the data blocks of the file.

6.2.8.2 Extended Argument 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	-	A	A	A	Count of arguments following
1	.RBPPN	A0	A0	A0	Directory name (project-programmer no.)
2	.RBNAM	A	A	A	Filename in SIXBIT
3	.RBEXT	A	A	A	File extension (LH)
		V	A0	A	Access date (bits 24-35)
4	.RBPRV	V	A0	A	Privilege (bits 1-8)
		V	V	V	Mode (bits 9-12)
		V	A0	A	Creation time (bits 13-23)
		V	A0	A	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	.RBFUT	V	A	A	Reserved for future
10	.RBEST	V	A	A	Estimated length of file (+no. blocks)
11	.RBALC	V	A	A	Highest relative block number within F.S. allocated by user or monitor to file (not counting 2nd RIB)
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

Table 6-6 (Cont)
Extended LOOKUP, ENTER, and
RENAME Arguments

Rel. Loc	Symbol	Lookup	Create Supers	Update Rename	Arguments and Value
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 porject-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	A1	V	Logical block number within F.S. (not cluster no.) of UFD data block in which the name of this file appears

A = Argument (supplied by privileged or nonpriviledged 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. 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 24-35 =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.
.RBPRV	Bits 0-8 =protection codes. Bits 9-12 =data mode in which file is created. Bits 13-23 =creation time in minutes since midnight. Bits 24-35 =creation date in standard format.
.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.
.RBFUT	Reserved for the future.
.RBEST	Reserved for the future.
.RBALC	Number of 128-word blocks, N, to be allocated to the file, including both RIB blocks, after completion of ENTER or RENAME. 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.

.RBALC (Cont) 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 if the user is logged in and is set by LOGIN. LOGOUT clears this bit.

RH =status of file.

Bit 18=1 (.RPDIR) if file is a directory file; needed to protect the system from a user who might try to modify a directory file.

Bit 19=1 (.RPNDL) if file cannot be deleted, even by a privileged program.

Bit 20=1 (.RPNCN) if file cannot be renamed, even by a privileged program.

Bit 21=1 (.RPNFS) if file should not be dumped by FAILSAFE because certain files are needed before FAILSAFE can run.

The following bits appear in both the LH and RH of this location:

Bit 11 and bit 29=1 if any file in this UFD (or this file) has had a hard data error while reading. (The IODTER bit has been set.) An entry is made in the BAT block so that the bad region is not reused.

Bit 10 and bit 28=1 if any file in this UFD (or this file) has had a hard data error while writing. (The IODTER bit has been set.) An entry is made in the BAT block so that the bad region is not reused.

Bit 9 and bit 27=1 if any file in this UFD (or this file) has had a software checksum error or redundancy check error. (The IOTMPM bit has been set.)

NOTE

Device errors (IODERR) 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 data error (IODTER) 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 CUSP.
.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 CUSP.
.RBQTR	Meaningful for UFD only. (In 5.02 monitors and later monitors.) 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 CUSP.
.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 CUSP.
.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 UFD's data block in which the name of this file appears.

6.2.8.3 Special Programmed Operator Service - The following are special programmed operator service UUOs.

a. USETI and USETO UUOs - USETI D,A and USETO D,A are treated similarly by the disk service routines. Their function is to notify the service routine that a particular relative block is to be used on the next INPUT or OUTPUT on channel D (whichever occurs first). A designates a particular block relative to the beginning of the file. The only difference between USETI and USETO occurs when A is greater than the current size of the file (in blocks). On USETI, the monitor simply sets the IODEOF flag and returns, whereas on USETO, the monitor zeroes the intervening blocks and does not set the IODEOF flag. The next INPUT or IN causes the EOF flag to be set. The next OUTPUT or OUT writes the block at the appropriate place. On input, the RIB block is designated by A = 0. On output, A = 0 returns the error bit IOBKTL. The first data block of the file (i.e., the one following the RIB) is designated by A = 1. If no previous LOOKUP or ENTER has been done, this UUO will set the improper mode error bit (IOIMPM).

b. Super USETI and USETO UUOs - With disk packs, there is a need to read and write data without using a directory hierarchy (e.g., for testing a pack in a timesharing environment or for a privileged recovery on any file structure). There must be a way to specify individual blocks of a file structure and/or unit without reference to any file. These blocks are called logical blocks because they must be transformed for the particular hardware before doing I/O. Under certain conditions, USETI and USETO are used to specify these logical blocks. When the following conditions are true, USETI and USETO reference logical block numbers with respect to a file structure instead of relative blocks within a file:

(1) The channel is INITed with a file structure name.

(2) No file is opened on the channel specified in the AC field.

(3) The structure is a single-access structure assigned to the user, or the program is privileged.

When the following conditions are true, USETI and USETO reference logical block numbers with respect to a unit instead of logical block numbers with respect to a file structure or relative blocks within a file:

(1) The channel is INITed with a physical unit name.

(2) No file is opened on the channel specified in the AC field.

(3) The unit is a member of a single-access file structure mounted for the user, or the program is privileged.

The terms super-USETI and super-USETO distinguish these UUOs from regular USETI and USETO. Super-USETI and super-USETO provide their arguments in the contents of the effective address, rather than the effective address itself.

USETI and USETO and their counterparts do not perform I/O; they change either the current position of the file (regular) or the current position in the file structure (super). Both super-USETI and super-USETO set the IOBKTL flag if a specified logical block is too large for the file structure or unit.

If a program is nonprivileged, super-USETI and super-USETO is a no-operation.

c. SEEK UUO - 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:

```
CALL D, [SIXBIT/SEEK/] ; 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

(1) USETO to select a block (relative or actual)
(2) SEEK to request positioning
(3) computations
(4) OUTPUT to request actual output

For input

(1) USETI to select a block (relative or actual)
(2) SEEK to request positioning
(3) computations
(4) INPUT to request actual input.

d. CALL [SIXBIT/RESET/] - 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 extendsion 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.

e. 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 4.9.3.4).

f. CHKACC UUO - This UUO allows privileged programs to check the user's access to a particular file. The call is:

```
MOVE AC, [EXP LOC]
ACCESS 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 bits 27 through 35 are zero, then bits 18 through 26 are 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	Change protection, rename, write, update, append, read, execute.
1	Rename, write, update, append, read, execute.
2	Write, update, append, read, execute.
3	Update, append, read, execute.
4	Append, read, execute.
5	Read, execute.
6	Execute only.
7	Create in UFD.
10	Read UFD 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 is not allowed.

6.2.8.4 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	.	
WRITE OUT (=11)	ENTER D, F	(Fail)
CLOSE B, F	.	
	ENTER D, F	(Succeed)
	WRITE OUT (=11)	
	CLOSE D, F	

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)	LOOKUP C, F	
ADD1 (=11)	.	
OUTPUT B, (=11)	.	
CLOSE B, F	.	
	ENTER D, F	(Succeed)
	INPUT C, (=11)	
	ADD1 (=12)	
	OUTPUT D, (=12)	
	CLOSE D, F	

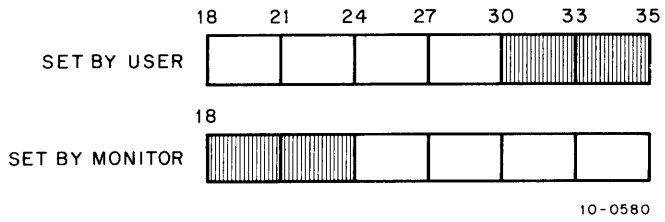
In this example, both jobs performed the ENTER FIRST; therefore, incorrect copies were not made and the increment of each job was recorded properly.

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	ENTER B, F	(Fail)
CLOSE	Here any information	
	from the LOOKUP and	
	INPUT must be discarded.	

6.2.9 File Status (refer to Appendix D)

The file status of the disk is shown below.



Bit 18 - IOIMPM

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 (6)buffered mode I/O attempted after super-USETI/USETO.

Bit 19 - IODERR — Search error, power supply failure.

Bit 20 - IODTER — Disk or data channel parity error. Checksum failure on INPUT.

Bit 21 - IOBKTL

a. Quota is exhausted (past overdrawn)

b. File structure is exhausted

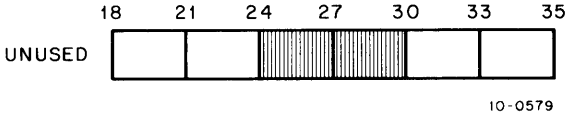
c. RIB is full

d. Super-USETI/USETO block is too large for the file structure

e. More than 777777 blocks were read with one super-USETI/USETO.

Bit 22 - IODEOF — EOF encountered on INPUT. No special character appears in the buffer.

Bit 23 - IOACT — Device is active



There are no device dependent bits.

6.2.10 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 CUSPs 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 most important distinction between public and private packs is that private packs are intended to be removed from the system during regular operation. Public packs 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 6.2.8.3c). 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 command (available in 5.02 monitors and later monitors). 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 REMOVE 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.

6.2.10.1 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.

6.2.10.2 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 CUSPs 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.

6.2.10.3 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 JBTSPL 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 on the disk rather than to 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 JBTSPL 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 so that the output spooler can label the output. Therefore, programs should give a filename if they can.

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 a PDP-10 Timesharing 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. Teletype wait queue - for jobs waiting for input or output on the user's console.

f. Teletype wait satisfied queue - for jobs that completed a Teletype 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 each queue, the position of a job determines its priority with respect to the other jobs in the same queue. The status of a job is changed when it is placed in 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 4.10.3). 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 re-started when the user finishes with the buffer.

Scheduling occurs at each clock tick (1/60th or 1/50th or 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 retrieved 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 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, a routine that queues disk requests and makes optimization decisions, handles any number of channels and controllers and up to eight units for each controller. 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, which 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.

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 cylinder. 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 (see MONITR.OPR).

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.

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 IODERR bit in the channel status word, and a data error sets the IODTER 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 timesharing 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 I) 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 (IOIMPM) in the user channel status word and returns control to the user program.

7.5 DIRECTORIES

7.5.1 Order of Filenames

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.

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 absolute 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 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
```

When a real-time device is added to the interrupt chain (CONSO skip chain) by a RTTRP UUO (refer to Paragraph 8.3), 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 8.3), the chain is as follows:

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 JRST <context switcher and dispatch for real-time interrupts>	
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 5 in normal block mode (refer to Paragraph 8.3), the chain is as follows:

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 JRST <context switcher> JEN @ CH5	
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.

Chapter 8

Real–Time Programming

Privileged jobs may for various reasons desire to be locked in core, that is, to never 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 control CUSP in core.
Performance analysis	Jobs monitoring the activities of the system need to be locked in core so that when they are entered to gather data, they are aware of their state and therefore, can record activities of the monitor independent of the monitor.

8.1 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 which 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 which 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 and 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.

8.2 LOCK AC, OR CALLI AC,60

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. When this UUO is executed by a privileged user program (privileges are granted by the system manager), it results in the job being locked in the optimal position in memory (at an extremity of user core).

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

a. The job is privileged (privileges 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.

The call is:

```
MOVSI  AC,1          ;if high segment is to be locked
MOVSI  AC,0          ;if no high segment or if high
                     ;segment is not to be locked
HRRI   AC,1          ;if low segment is to be locked
HRRI   AC,0          ;if low segment is not to be locked
LOCK AC,             ;or CALLI AC,60
error return         ;AC contains an error code
normal return
```

On a normal return, the job is locked in core. If there is a high segment, the LH of AC contains its absolute address, shifted right 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 contains an error code indicating the condition that prevented the job from being locked. The error codes are as follows:

Error Code	Explanation
1	The job does not have locking privileges.
2	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	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.

NOTE

The COR UUO may be given for the high or low segment of a locked job if the segment is not locked in core. When the segment is locked in core, the COR UUO and the CORE command with a non-zero argument cannot be satisfied and, therefore, always give an erroneous response. 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 8-1E).

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.

8.2.1 Non-Swapping Systems

a. Any number of jobs can be locked in core without fragmentation occurring if the jobs are initiated immediately after the monitor is loaded.

b. During normal timesharing, a job is locked at the top of user core if the hole at the top of core is large enough to contain the job. Otherwise, the job is locked as low in core as possible.

c. Locking a job in core never makes the system fail, but it is possible that all of available core will not be utilized in some mixes of jobs.

8.2.2 Swapping Systems

a. There is no memory fragmentation if no more than two jobs 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 content for on a first-come-first-serve basis (Total User Core - initial CORMAX + CORMIN).

8.2.3 Core Allocation Resource

Since the 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. Since a job may share a queue with other jobs and since 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 swapping systems, the ideal position is always achieved guaranteeing minimum fragmentation. In non-swapping systems, minimum fragmentation is achieved only if the ideal position does not contain an active segment (see Figure 8-1).

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 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.

8.2.4 Unlocking Jobs

A job relinquishes its locked status when either the user program executes a 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 CUSP-invoking command.

When the job is unlocked, it becomes a candidate for swapping and shuffling. 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.

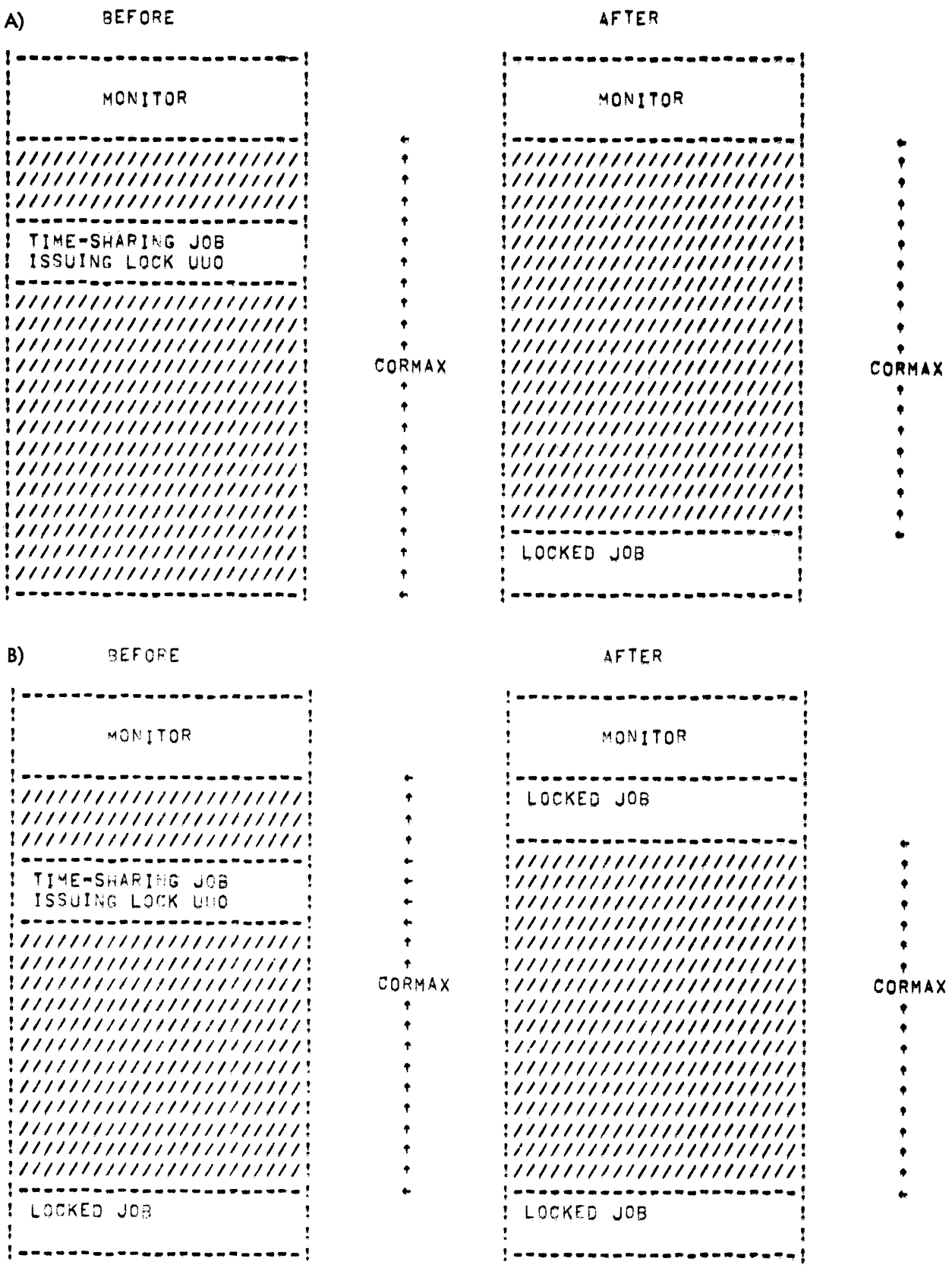


Figure 8-1 Locking Jobs In Core

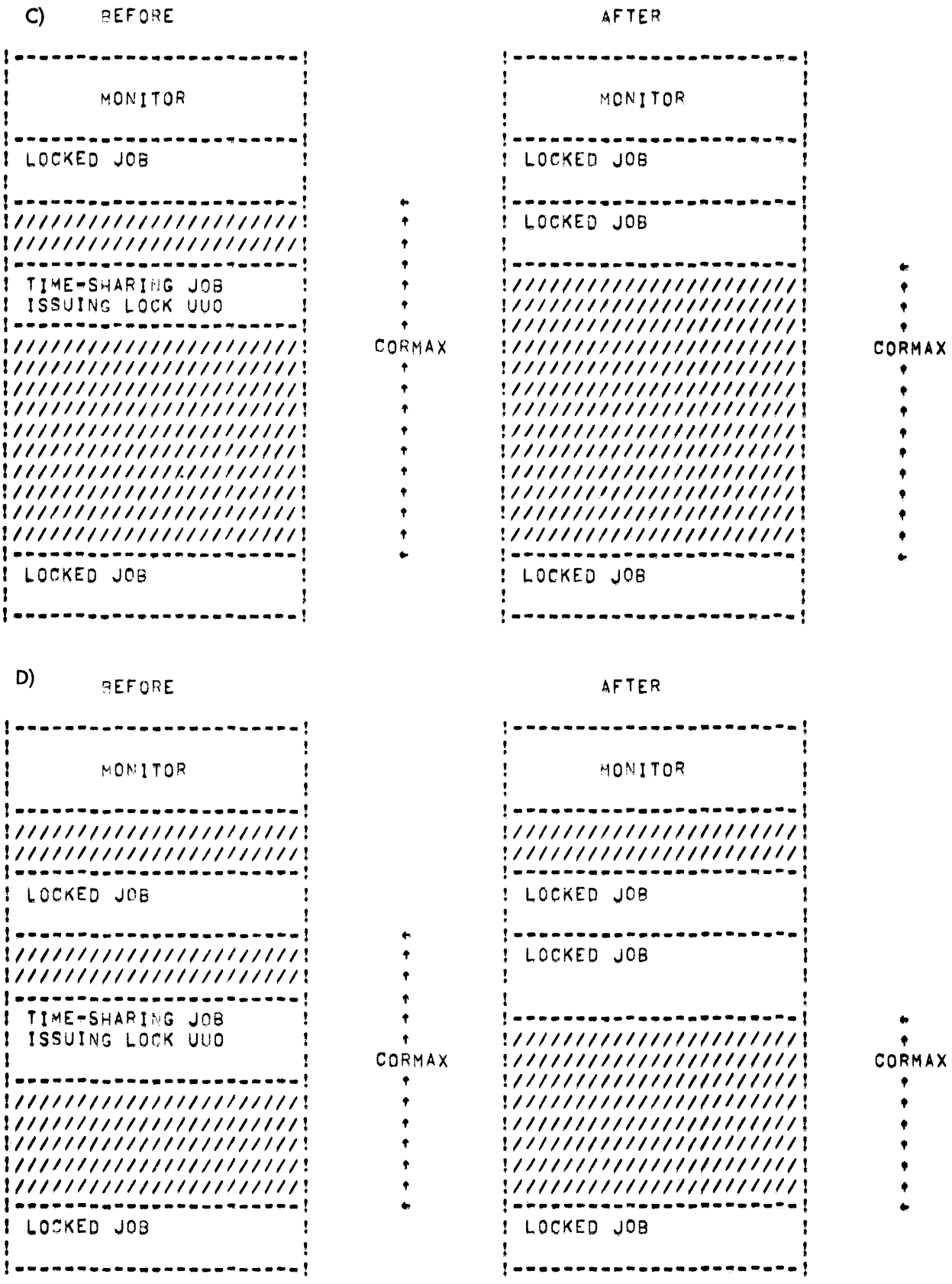


Figure 8-1 Locking Jobs In Core (Cont)

E) Unlikely Fragmentation Case

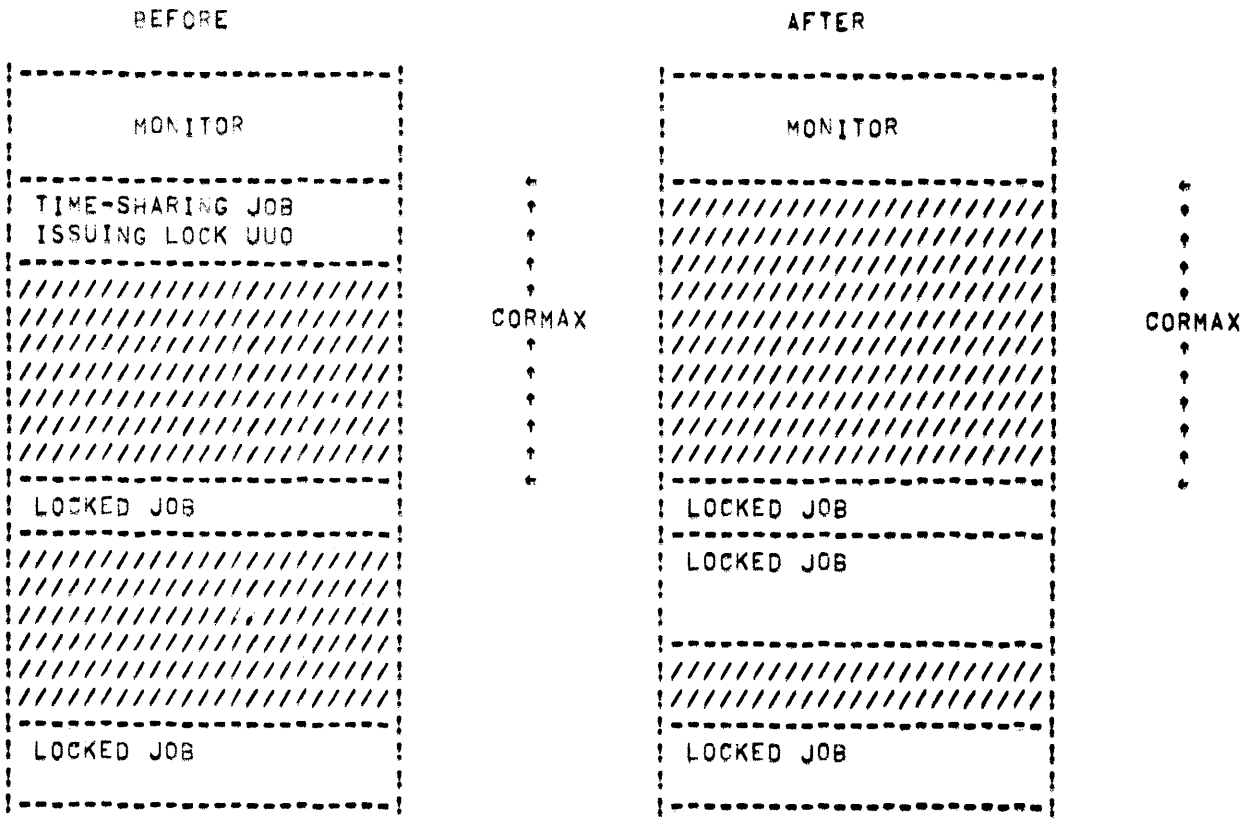


Figure 8-1 Locking Jobs In Core (Cont)

8.3 RTTRP AC, OR CALLI AC, 57

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 to 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) PVLOCK - allows the job to be locked in core.
2) PVRTT - allows the RTTRP UUO to be executed.

WARNING

> Improper use of features of the RTTRP UUO can cause the system to hang. 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.

Real-time jobs control devices in 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.

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). 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 (the ACs, APR flags, protection-relocation register, UUO trap addresses 40 and 41, and the reserved instruction trap addressed 60 and 61), sets the new protection-relocation register 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 as follows:

```
CONSO       DEV, BITS
JRST        NXT DEV
BLKI        DEV, POINTR
JRST        <CONTEXT SWITCHER>
JEN         @CH, PI
```

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 enable bits and 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 conditions
        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.
```

8.3.1 Data Block Mnemonics

The following mnemonics are used in describing the data block associated with the RTTRP UUO.

8.3.1.1 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.

8.3.1.2 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.

8.3.1.3 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 4.3.3.1). No APR errors are detected if the interrupt routine is on a PI level higher than or equal to the APR interrupt level.

8.3.1.4 DEV - DEV is the real-time device code.

8.3.1.5 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.

8.3.1.6 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 up as an absolute address

instead of a relative 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.

8.3.2 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 which 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.

8.3.3 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 since improper use of the I/O instructions could halt the system (i.e., CONO APR,0, CONO PI,0, or HALT). Note that a user can obtain just the user IOT privilege by issuing the RTTRP UUO with PICHL = 0 (see Paragraph 8.3.1.1).

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.

Code	Value	Meaning
Bit 26=1	1000	Device already in use by another job.
Bit 27=1	400	Illegal AC used during RTTRP UUO at interrupt level.

Code	Value	Meaning
Bit 28=1	200	Job not locked in core, or not privileged.
Bit 29=1	100	System limit for real-time devices exceeded.
Bit 30=1	40	Illegal format of CONSO, BLKO, or BLKI instruction.
Bit 31=1	20	BLKADR or pointer word illegal.
Bit 32=1	10	Error address out of bounds.
Bit 33=1	4	Trap address out of bounds.
Bit 34=1	2	PI channel not currently available for BLKI/BLKO's.
Bit 35=1	1	PI channel not available (restricted use by system).

8.3.4 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 (see Paragraph 8.3.1.5.). 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) since 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.

8.3.5 Removing Devices from a PI Channel

When PICHL=0 in the data block (see Paragraph 8.3.1.1), 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 8.2). This UUO causes a CONO DEV,0 to be executed before the device is removed. Monitor commands which issue implicit RESETS also remove real-time devices (e.g., R, RUN, GET, CORE 0, SAVE, SSAVE).

8.3.6 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 any instruction which 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 by trapping to location 60.

8.3.7 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
          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
          CALLI 31                   ;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
          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                ;RTTRP 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
          CALLI 31                   ;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
        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 WORD
        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
```

8.3.8 FORTRAN Usage of Real-Time Trapping

Real-time library subroutines allow FORTRAN programs to connect real-time devices to the priority interrupt system. These subroutines provide the FORTRAN-oriented user the flexibility needed to write real-time code without the requirement of learning assembly language coding and hardware peculiarities. At present these subroutines are not reentrant and cannot be called from two PI levels simultaneously. If this is a requirement, then a different routine must be called at each level.

To enter the same subroutine from two or more PI levels simultaneously, the following MACRO program can be used:

```
        TITLE       SUB1
        ENTRY       SUB1, SUB2
        EXTERN      SUB
SUB1:   Z
        JRST        SUB+1

SUB2:   Z
        JRST        SUB+1
        END
```

For each PI level which needs the subroutine SUB, a new ENTRY point must be made. In the example above, three different PI levels could be active simultaneously using the three different CALLS to the same routine (CALL SUB, CALL SUB1, and CALL SUB2). This technique can be used for any routine which is reentrant except for the FORTRAN CALL sequence. All of the real-time subroutines except for CONECT and DISCON meet this requirement.

The real-time subroutines and their functions are as follows.

8.3.8.1 LOCK - The LOCK subroutine locks the job in core. This routine must be called before any other routines can be executed. The call is

```
CALL LOCK
```

8.3.8.2 RTINIT - The RTINIT subroutine initializes all the internal tables controlling the real-time device specified. The call is

```
CALL RTINIT (unit, dev, PI, TRPADR, MASK)
```

where

unit is the real time device unit number (starting at 1).
dev is the device code for the real-time device.
PI is the PI level on which the device is to be placed.

TRPADR is the address in the FORTRAN program where real time interrupts are to trap. This address is loaded by ASSIGN 100 to TRPADR.

MASK is the mask of all interrupting flags for the real time device. This mask is set by RTSTRT (see Paragraph 8.3.8.5) and should be zero when the real-time device is inactive.

8.3.8.3 CONECT - The CONECT subroutine indicates to the system that the real-time device specified is to be connected to the indicated PI level and the trap address for trapping is to be set up. The call is

CALL CONECT (unit, mode)

where

unit is the real-time device unit number.

Mode is -1 for writing blocks of data.
0 for a trap on each device interrupt.
+1 for reading blocks of data.

8.3.8.4 DISCON - The DISCON subroutine disconnects the real-time device from the PI level. The call is

CALL DISCON (unit)

where

unit is the real-time device unit number.

8.3.8.5 RTSTRT - The RTSTRT subroutine starts the real-time device with a CONO DEV, START and can also be used to stop the device and zero the CONSO mask (i.e., CALL RTSTRT (unit, 0,0)). This is the preferred method for stopping the device. The call is

CALL RTSTRT (unit, start, intmsk)

where

unit is the real-time device unit number.
start is the flags necessary to start the device (CONO DEV, START).
intmsk is the mask of all interrupting bits. These bits are loaded into MASK (refer to Paragraph 8.3.8.2).

8.3.8.6 BLKRW - The BLKRW subroutine sets up the size and the starting address of the block of data. The call is

CALL BLKRW (unit, count, adr)

where

unit is the real-time device unit number.
count is the number of words to be read or written.
adr is the array from which the data is transferred depending on the MODE setting.

After the specified number of words are read or written, a trap occurs to the interrupt trap routine. A new count and starting address must be set up each time the present one is exhausted.

8.3.8.7 RTREAD - The RTREAD subroutine reads a single word of data from the real-time device (DATAI DEV, ADR). The call is

CALL RTREAD (unit, adr)

where

unit is the real-time device unit number.
adr is the address of the data to be read.

8.3.8.8 RTWRIT - The RTWRIT subroutine sends a single word of data to the real-time device (DATAO DEV, ADR). The call is

CALL RTWRIT (unit, adr)

where

unit is the real-time device unit number.
adr is the address of the data word to be sent to the real-time device.

8.3.8.9 STATO - The STATO subroutine sends an argument to the status register of the device. The call is

CALL STATO (unit, adr)

where

unit is the real-time device unit number.
adr is the location of the status bits to be sent to the real-time device (CONO DEV, @ ADR).

8.3.8.10 STATI - The STATI subroutine reads the current status bits into a specified location. The call is

CALL STATI (unit, adr)

where

unit is the real-time device unit number.
adr is the location in which the device status bits are to be read (CONO DEV, ADR).

8.3.8.11 RTSLP - The RTSLP subroutine causes the background portion of the FORTRAN program to sleep for a specified number of seconds (maximum of 60). The call is

CALL RTSLP (time)

where

time is the sleep interval in seconds.

RTSLP is called from the timesharing level causing the program to sleep for the specified amount of time. When the sleep interval reaches zero, the program checks to see if RTWAKE has been called at interrupt level. If RTWAKE has been called, RESLP returns to the calling program. If RTWAKE has not been called, the background portion of the job goes back to sleep again.

8.3.8.12 RTWAKE - The RTWAKE subroutine activates the background portion of the FORTRAN program. This subroutine is called at interrupt level. The call is

CALL RTWAKE

8.3.8.13 Example - The following is an example of a FORTRAN real-time program.

```
      IMPLICIT INTEGER (A-Z)
      DIMENSION A (100)

C     LOCK THE JOB IN CORE
      CALL LOCK

C     20 IS THE BEGINNING OF THE REAL-TIME SECTION
C     THE PI LEVEL OF THE REAL-TIME JOB IS 5
C     THE DEVICE CODE IS OCTAL 104
      ASSIGN 20 TO TRPADR
      UNIT = 1
      PI = 5
      DEV = "104
      MASK = 0

C     INITIALIZE REAL TIME DEVICE
      CALL RTINIT (UNIT, DEV, PI, TRPADR, MASK)
```

```
      MODE = 0

C     CONNECT DEVICE TO PI LEVEL
      CALL CONECT (UNIT, MODE)

      CONO = "25
      CONSO = "10

C     START REAL TIME DEVICE
      CALL RTSTRT (UNIT, CONO, CONSO)

C     SLEEP FOR FIVE SECONDS, THEN CHECK FOR A RTWAKE CALL
C     AT INTERRUPT LEVEL
10    CALL RTSLP (5)

      CALL EXIT

C     INTERRUPT LEVEL SECTION
C     HAS THE READER RUN OUT OF PAPER TAPE?
20    CALL STATI (UNIT, J)

      IF (J-"400.GE.0) GO TO 21

C     YES, THERE IS NO MORE TAPE, SO WAKE UP
C     BACKGROUND PORTION OF JOB

      CALL RTSTRT (UNIT,0,0)
      CALL DISCON (UNIT)
      CALL RTWAKE
      CALL DISMISS

C     THERE IS STILL PAPER TAPE IN THE
C     READER.  READ ANOTHER CHARACTER.
21    CALL RTREAD (UNIT,X)
      CALL DISMIS
      END
```

8.4 DIRECT USER I/O

In special cases, the RTTRP UUO does not offer a fast enough response to real-time interrupts. These special cases can always be solved by putting a device service routine into the monitor. However, some of the fast response requirements can be met with the TRPSET feature. In order to achieve fast response to interrupts, the TRPSET feature turns off timesharing 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, since 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.

8.4.1 TRPSET AC, or CALLI AC,25

This privileged UUO allows the user program to gain control of the interrupt locations. If the user does not have the TRPSET privileges (either PVTRPS = 1 (bit 15), or user is job 1), an error return to the next

location after the CALL 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, 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, adds the job relocation address to the address field, and stores it in the specified executive PI location.

Thus, the user can set up a priority interrupt trap into his relocated core area. Upon 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 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.
```

The monitor assumes that user location ADR contains either a JSR U or BLKI U, where U is a user address; consequently, the monitor adds the job relocation to the contents of location U to make it an absolute IOWD. 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.

```
INT 46:   BLKI RTD,INBLOK     ;relocation constant
                              ;for user is added
INT 47:   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
          AOBJN AC, .+1       ;normal return
          XCT .-3
          JRST EXITR          ;error return
          .                   ;normal return
          .
          .
XITINT:   0                   ;PC saved
          ...                 ;interrupt dismiss routine
```

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 executive mode, a JEN instruction issued in user mode does not work because of memory relocation. This is solved by a call to UJEN (opcode 100). This UUO causes the monitor to dismiss the interrupt from executive 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.

```
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
```

Upon entering the routine from absolute 47 with a JSR to XITINT + REL (where REL. is the relocation constant), the executive mode flip-flop is set. 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 coexist with other timesharing jobs.

8.4.2 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.

8.5 HPQ AC, OR CALLI AC,71

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 in one of these queues is executed before all other jobs in the system. Thus, real-time associated jobs can receive fast response times from the timesharing scheduler.

In addition to being scanned according to their priority before all other queues for job execution and swap-in, the high priority queues are scanned in reverse order (lowest priority first, highest priority last) for swap-out after all other queues have been scanned. If the highest priority job has been swapped onto the disk, then that job is the first job to be swapped in for execution.

8.5.1 HPQ UUO Format

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 bits 6 through 9 in the privilege word. 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 places the job back to the timesharing level.

Appendix A
DECtape Compatibility Between DEC Computers

Written By \ Read By	PDP 4 550 and 555 TU55	PDP 5 552 and 555 TU55	PDP 6 551 and 555 TU55	PDP 7 550 and 555 TU55	PDP 8 552 and 555 TU55	PDP 8 TC01 and TU55	PDP 8/I TC01 and TU55	PDP 9 TC01 and TU55	PDP 10 TC01 and TU55
PDP-4	A	D	D	A	D	D	D	D	D
PDP-5	D	A	B	C	A	A	A	A	A
PDP-6	D	A	A	C	A	A	A	A	A
PDP-7	A	C	C	A	C	C	C	C	C
PDP-8 552	D	A	B	C	A	A	A	A	A
PDP-8 TC01	D	A	B	C	A	A	A	A	A
PDP-8/I	D	A	B	C	A	A	A	A	A
PDP-9	D	A	A	C	A	A	A	A	A
PDP-10	D	A	A	C	A	A	A	A	A

A = Can be done

B = Cannot be done because of difference in writing checksum

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.

NOTE

The PDP-10 does not allow search to find first or last blocks when searching from the end-zone.

Appendix B
Monitor Sizes

B.1 MULTIPROGRAMMING NON-DISK MONITOR (JUNE 1970, REENTRANT 4 SERIES, VERSION 72)

Three resident components of the monitor are:

a. Required code (6.2K)

b. Optional device code (0-4.4K)

c. Tables and buffers per job (73 words per job)

B.1.1 Required Code

The required code, assuming all features, is:

Lower core	DLSINT
Common	ERRCON
CLSCSS	SCNSRF
CLOCK1	SEGCON
COMCON	SYSINT
CORE1	UUOCON

B.1.2 Optional Device Code

The optional devices are listed below with the number of devices used in figuring the optional device code.

Device	Number	Device	Number
DTA	8	DIS	1
MTA	2	LPT	1
PTY	2	PLT	1
CDR	1	PTP	1
CDP	1	PTR	1

B.1.3 Tables and Buffers

Tables and buffers allowed for each job are:

```
 18 words of tables
 55 words of TTY device data block space
 73 words per job
```

B.2 SWAPPING MONITOR (JUNE 1970, REENTRANT 5 SERIES, VERSION 01)

Three resident components of the monitor are:

a. Required code (14K)

b. Optional device code (5.2K)

c. Tables and buffers per job (1K for every 4 jobs)

B.2.1 Required Code

The required code, assuming all features, is:

Lower core	JOBDAT
COMMON	SCHED1
CCINT	SCNSRF
CLOCK1	SEGCON
COMCON	SWPSER
CORE1	SYSINI
ERRCON	TMPUUO
FILSER	UUOCON

B.2.2 Optional Device Code

The optional devices are listed below with the number of devices used in figuring the optional device code.

Device	Number	Device	Number
DTA	8	DIS	1
MTA	3	LPT	1
PTY	2	PLT	1
CDR	1	PTP	1
CDP	1	PTR	1
FHA	2	DPA	4

B.2.3 Tables and Buffers

Tables and buffers allowed for each job are:

```
 21 words of tables
 90 words of DSK device data blocks (approximately 3 files)
 40 words of DSK access information
 20 words of TMPCOR storage
 55 words of TTY device data block space
---
226 words per job
```

For a complete swapping system, the resident monitor is (assuming all devices):

8	JOBS	21K
16	JOBS	23K
24	JOBS	25K
32	JOBS	27K
40	JOBS	29K
48	JOBS	31K
56	JOBS	33K
64	JOBS	35K

Appendix C
Writing Reentrant User Programs

C.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 sub-program 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 a HISEG pseudo-op.) 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 (22 files in the maximum). 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.

C.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
        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
```

C.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 in 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 to put the code in the high segment if it does not put the high segment over the next 1K boundary.

C.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 CUSPs 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. The data is placed last in the source file following a LIT pseudo-op and consists only of Block 1 and Block N statements, along with data location tags. If a reentrant program is desired, a LOC 140 is assembled, which places the data at absolute 140 in the low segment. Because of the LOC, no other relocatable program can be loaded into the low segment. The program should be debugged

as a non-reentrant program with DDT because DDT is a low segment relocatable file. The LOADER switch /B is used to protect the symbols. The usual way of assembly is reentrant, therefore, unless already defined, the conditional switch is 1.

The program must have one location in the job data area when it is assembled to be reentrant so that the monitor starts assigning buffers at the end of the data area in the low segment instead of at location 140. This is accomplished by changing the LH of JOBSA before the CALLI 0 (RESET) or changing the contents of JOBFF after the CALLI 0, depending on how the program reinitializes itself on errors and on completion. The program should not change these locations if it is assembled as non-reentrant; thus, the symbol table can be protected using the LOADER /B switch, which places the symbols next to the last program loaded and sets the LH of JOBSA appropriately higher. Therefore, this code is under control of conditional assembly.

```
        TITLE DEMO - DEMO ONE SOURCE REENTRANT PROGRAM -V001

JOBVER=137
        LOC 137
        EXP 001                         ;VERSION NUMBER

        INTERN JOBVER,PURE
        EXTERN JOBSA,JOBFF

IFNDEF PURE,<PURE=1>                    ;ASSUME REENTRANT IF PURE UNDEFINED
        IFN PURE,<HISEG>                ;TELL LOADER TO LOAD IN HIGH SEGMENT
                                        ;IF REENTRANT

BEG:
IFN PURE,<                              ;ONLY NEED IF REENTRANT
                                        ;(NOT NEEDED IF TWO FILES)
        MOVSI   T,DATAE                 ;SET FIRST FREE LOCATION IN LOW SEG,
        HLLM    T,JOBSA                 ;RESET SETS JOBFF FROM LH OF JOBSA
>
        CALLI   0                       ;DO CALL RESET
        MOVE    T,JOBFF                 ;ASSIGN AT LEAST ENOUGH CORE FOR DATA
        CALLI   T,11                    ;CORE UUO
        JRST    ERROR
        MOVE    T,[XWD DATAB,DATAB+1]   ;NOW CLEAR DATA REGION
        SETZM   DATAB
        BLT     T,DATAE-1               ;LAST LOCATION CLEARED
        .
        .
        .
        .
        LIT                             ;PUT LITERALS IN HIGH SEG
                                        ;DATA AREA:
IFN PURE,<LOC 140>                      ;START DATA AREA AT 140 IN LOW SEG
                                        ;IF REENTRANT
DATAR:                                  ;FIRST LOCATION CLEARED EVERY STARTUP
DATA:   BLOCK 1
TABLE:  BLOCK 128
        .
        .
        .
DATAE:  END     BEG                     ;DEFINE FREE LOCATION
```

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													Sync Input		Data Mode			
	GETSTS	No 7-9 Punch	Data Missed	Binary Check-sum Error		EOF card EOF button	I/O Active												
DIS	SETSTS GETSTS						I/O Active									Data Mode			
DSK	SETSTS													Sync Input	User Word Count	Data Mode			
	GETSTS	Write Lock	Search Error	Parity Check-sum	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												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														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		Ignore Inter-rupt	Echo Fail-ure	Char-acter Lost														

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.

Appendix E
Error Codes

The following error codes (refer to 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,*), or filenames do not match (UPDATE).
.ERIPP	1	Incorrect project-programmer number. (UFD does not exist.)
.ERPRT	2	Protection failure or directory full on DTA.
.ERFBM	3	File being modified (ENTER).
.ERAEF	4	Already existing filename (RENAME) or different filename (ENTER after LOOKUP).
.ERISU	5	Illegal sequence of UUOs (RENAME with neither LOOKUP nor ENTER, 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).

Table E-1 (Cont)
Error Codes

Symbol	Code	Explanation
.ERILU	13	Illegal UUO (GETSEG only). No two-register relocation capability.
.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.

Appendix F
Monitor Diagnostic Messages

The following table contains a summary of the diagnostic messages that the system can issue. The conventions used in the summary are:

dev	represents any legal device name.
file.ext	represents any legal filename and filename extension.
adr	represents a user address.
n	represents a number, usually a job number or device unit number.
unit	represents any legal disk unit.

Programs and commands causing the error message are given in parentheses.

Table F-1
Monitor Diagnostic Messages

The typein is typed back preceded and followed by ?	The monitor encountered an incorrect character (e.g., a letter in a numeric argument). The incorrect character appears immediately before the second ?. For example: . CORE ABC ? CORE A?
ACTIVE SWAPPING LIST FULL	An attempt was made to specify more units to the active swapping list than the monitor tables can handle. The current limit is 8. If the operator needs more swapping space, he should increase the amounts on the eight units already specified. (ONCE ONLY).
?ADDRESS CHECK FOR DEVICE dev	The monitor checked a user address on a UUO and found it to be too large (>C(JOBREL)) or too small (<JOBPFI); in other words, the address lies outside the bounds of the user program.
?ALREADY ASSIGNED TO JOB n	The device is already assigned to another user's job (job n).

Table F-1 (Cont)
Monitor Diagnostic Messages

?ARGS ARE: DAY, RUN, WAIT, READ, WRITE	The user typed an illegal argument in the WATCH command string.
?BAD DIRECTORY FOR DEVICE DTAn	The system cannot read or write the DECtape directory without getting some kind of error. This error often occurs when the user tries to write on a write-locked tape or use a DECtape that has never been written on.
BLOCK NOT FREE	M specifies a unit or file structure logical block that is not free.
?BUSY	The console addressed is not communicating with the monitor. The operator's console is never busy. (SEND).
CANNOT EXCEED # BLOCKS IN FILE STRUCTURE = m	The number of disk blocks specified for reserved quotas or the number of disk blocks specified for the overdraw amount was too large. The operator should type in a number less than or equal to m. (ONCE ONLY).
CANNOT EXCEED # SAT BLOCKS ON UNIT = n	The number of SAT blocks in core cannot exceed the number of SAT blocks (n) on the unit. The operator should type a number less than or equal to n. (ONCE ONLY).
?CAN'T ATT TO JOB	The project-programmer number specified is not that of the owner of the desired job or is not [1,4].
?CAN'T CONTINUE	The job was terminated due to a monitor-detected error and cannot be continued.
CAN'T CREATE NEW FILE STRUCTURE SEARCH LIST	The monitor cannot create a new file structure search list.
?CAN'T DECIPHER THAT	There is a syntax error in the command string. (MOUNT, DISMOUNT, FILE).
?CAN'T FIND FILE file.ext	The specified file could not be found.
?CAN'T DET DEV	The user is not logged-in under [1.4].
COMMAND ERROR	General catch-all error response for the COMPIL commands. The syntax of the command is in error, and the command cannot be deciphered. (COMPIL)
[CREATING NEW FILE]	The specified file does not exist; therefore a MAKE command is assumed.
dev: ASSIGNED	The device has been successfully assigned to the user's job.
?DEVICE CAN'T BE REASSIGNED	A user's Teletype cannot be reassigned, or an attempt was made to reassign a device that a job is still using.

Table F-1 (Cont)
Monitor Diagnostic Messages

?DEVICE dev OK?	The device is temporarily disabled. The line printer may be turned off or out of paper. For magnetic tapes, no tape is mounted or the switch is in LOCAL. The user should correct the situation and then proceed (retry the operation) by typing CONTINUE.
?dev WASN'T ASSIGNED	The device is not currently assigned to the user's job and cannot be deassigned or reassigned by the job.
?DEVICE dev NOT AVAILABLE	Specified device cannot be initialized because another user is using it. (COMPIL)
DISMOUNT COMPLETE	The DISMOUNT command has completed.
DPAn NOT AVAILABLE	The drive indicated by the user is not currently available. (MOUNT).
DPAn NOT READY	The indicated drive is either off-line or physically write-locked when write-enabled was requested. The operator will be notified. (MOUNT).
DPA IS OFF-LINE DO YOU WANT IT TO BE 1) ON-LINE OR 2) DOWN? (TYPE #)	Controller DPA (RP10) is off-line. The operator should check settings of all switches in RP10 bay. All switches should be down. After changing switches, the operator should type 1. If the operator does not want the monitor to use the controller, he should type 2. Also applies to DPB. (ONCE ONLY).
DPA0 IS OFF-LINE DO YOU WANT IT TO BE 1) ON-LINE, 2) OFF-LINE, OR 3) DOWN? (TYPE #)	The operator should check the START/STOP rocker switch and the ENABLE/DISABLE switch on the individual disk pack unit. They should be in the normal position with the top of the switch in. After changing switches, the operator should type 1. If the operator does not want the monitor to use the unit, he should type 2. This message also applies with DPA1, DPA2...,DPA7, DPB0, DPB1,...,DPB7. (ONCE ONLY).
DPA0 IS WRITE PROTECTED. DO YOU WANT IT TO BE 1) WRITE-ENABLED, OR 2) WRITE-PROTECTED?	Disk pack unit DPA0 is on-line, but is write-protected. If the operator wishes it to remain this way, he should type 2. Otherwise he should set the READWRITE/READ ONLY rocker switch to normal (top of switch in) and then type 1. This message also applies with DPA1,...,DPA7, DPB0, DPB1,...,DPB7. (ONCE ONLY).
?ENTER FAILURE	The ENTER to write the file failed. The error code may be seen by examining location 1.

Table F-1 (Cont)
Monitor Diagnostic Messages

?ERROR IN JOB n	A fatal error occurred in the job or in the monitor while servicing the job. This typeout usually precedes a one-line description of the error.
?EXCEED LOG-OUT m QUOTA BY n BLOCKS	The total number of blocks for all the user's files exceeds the maximum permitted value (m) by the indicated amount n. The user may use PIP or the DELETE command to remove files. Until the user is under the limit, he cannot dismount the file structure. (DISMOUNT).
EXECUTION DELETED	A program is prevented from being executed because of errors detected during assembly, compilation, or loading. Loading is performed, but the loader exits to the monitor without starting execution. (LOADER).
FHA IS OFF-LINE DO YOU WANT IT TO BE 1) ON-LINE OR 2) DOWN? (TYPE #)	Controller FHA (RC-10) is off-line. The operator should check settings of all switches in RC-10 bay. All switches should be down. After changing switches, the operator should type 1. If the operator does not want the monitor to use the controller, he should type 2. Also applies to FHB. (ONCE ONLY).
FHA0 is OFF-LINE DO YOU WANT IT TO BE 1) ON-LINE, 2) OFF-LINE, OR 3) DOWN? (TYPE #)	The operator should check the unit dial selectors. One of them (DISK A, DISK B, DISK C, or DISK D) should be set to 0. The operator should set the switches for all the units he has to 0, 1, 2, 3. The other units should be OFF. The operator should not touch any dials which are dialed to numbers numerically less than the one just typed out, since the monitor has already read these units. After changing the switches and dials, the operator should type 1. If the unit is temporarily down and will be fixed while the system runs, he should type 2. In all other cases, he should type 3. Could apply to FHA1,...,FHA3, FHB0,...,FHB3. (ONCE ONLY).
file.ext FOUND BAD BY FAILSAFE READING MTA	The file in the file structure has an error status as flagged in the UFD of the file structure. (LOGIN).
file.ext HARDWARE DATA READ ERROR DETECTED	The file has a hardware data read error flagged in the UFD of the file structure. (LOGIN).
file.ext HARDWARE DATA WRITE ERROR DETECTED	The file has a hardware data write error flagged in the UFD of the file structure. (LOGIN).
?file.ext NOT FOUND	The program file requested cannot be found on the system device or the specified device.
file.ext SOFTWARE CHECKSUM OR REDUNDANCY ERROR	The file has an error as flagged in the UFD of the file structure. (LOGIN).

Table F-1 (Cont)
Monitor Diagnostic Messages

FILE IN USE OR PROTECTED	A temporary command file could not be entered in the UFD (user's file directory). (COMPIL).
?FILENAME ALREADY IN USE	The specified file already exists. (COMPIL).
FIRST BAT BLOCK CONSISTENCY ERROR	The ONCE ONLY dialog has discovered that the first of two redundant BAT blocks does not contain some of the data normally expected in a BAT block. This is not a fatal error since the other BAT block is probably all right. If both BAT blocks have this error, the operator should initialize the BAT blocks. This error may occur if some of the diagnostics are run. (ONCE ONLY).
FIRST BAT BLOCK HARDWARE ERROR	The ONCE ONLY dialog has had a hardware error while reading the first of two redundant BAT blocks. Since there is another BAT block, this error is usually not fatal. The controller status is put in the console lights. (ONCE ONLY).
FIRST HOM BLOCK CONSISTENCY ERROR	The ONCE ONLY dialog has discovered that the first of two redundant HOM blocks does not contain some of the data normally expected in a HOM block. Therefore, none of the data should be considered valid. This is not a serious error since the other HOM block is usually all right. If both HOM blocks have consistency errors, the operator has to dissolve the file structures, redefine, and refresh. (ONCE ONLY).
FIRST HOM BLOCK HARDWARE ERROR	The ONCE ONLY dialog has had a hardware error while reading or writing the first of two redundant HOM blocks. This is not fatal since there is another HOM block. The controller status is put in the console lights, and the controller is left in its error condition. (ONCE ONLY).
FROM JOB n	An informative message telling the user the job number to which the console was attached or from which the console is detaching. (ATTACH, DETACH).
fs MOUNT COMPLETE	The file structure (fs) is mounted and ready for use; the MOUNT command is complete. (MOUNT).
?HALT AT USER adr	The user's program executed a HALT instruction at adr. Typing CONTINUE resumes execution at the effective address of the HALT instruction.
?HUNG DEVICE dev	If a device does not respond within a certain period after it is referenced, the system decides that the device is not functioning and outputs this message.
?ILLEGAL DATA MODE FOR DEVICE dev AT USER adr	The data mode specified for a device in the user's program is illegal, such as dump mode for Teletype.

Table F-1 (Cont)
Monitor Diagnostic Messages

?ILLEGAL DRIVE DPAn	The drive specified by the user is in conflict with the unit or controller type required by the units of the file structure. (MOUNT).
?ILLEGAL JOB NUMBER	The job number is too large.
?ILLEGAL UUO AT USER adr	An illegal UUO was executed at user location adr.
?ILL INST. AT USER adr	An illegal operation code was encountered in the user's program.
?ILL MEM REF AT USER adr	An illegal memory reference was made by the user's program at adr or adr + 1.
?INPUT DEVICE dev CANNOT DO OUTPUT AT USER adr	Output was attempted on a device that can only do input (e.g., the card reader).
INPUT ERROR	I/O error occurred while reading a temporary command file from the disk. File should be rewritten. (COMPIL).
?INPUT FILE NOT FOUND	The specified file does not exist. (COMPIL).
?INVALID ENTRY - TRY AGAIN	An illegal project-programmer number or password was entered and did not match identification in system. (LOGIN).
?I/O TO UNASSIGNED CHANNEL AT USER adr	An attempt was made to do an OUTPUT, INPUT, OUT, or IN to a device that the user's program has not initialized.
?JOB CAPACITY EXCEEDED	This message is received by the first user who attempts to LOGIN after the maximum number of jobs that the system has been set to handle has been initiated. (LOGIN).
JOB SAVED	The output is completed.
LAST UNIT WASN'T FOUND IN STR DSKn	The last unit in the file structure is missing. The operator should check to see that all the proper packs are mounted and on-line. If not, he should remount them and restart the monitor at 140. Otherwise, the operator has to dissolve the file structure, redefine it, and then refresh it, thereby destroying any data already on the unit. (ONCE ONLY).
LINKAGE ERROR	An I/O error occurred while reading a CUSP from device SYS:. (COMPIL).
?LOCKED-OUT BY OPERATOR	The operator is preventing any new accesses to the file structure in order that it may be removed. (MOUNT).
?LOGICAL NAME ALREADY IN USE, DEVICE dev ASSIGNED	The user previously assigned this logical name to another device. The device is assigned but the logical name is not.

Table F-1 (Cont)
Monitor Diagnostic Messages

LOGICAL STR #n MISSING FROM "SYS" SEARCH LIST	A file structure is missing from the SYS search list. This condition need not be corrected, since the monitor will skip the missing file structure. To avoid the message in the future, the operator should change the SYS search list when asked. (ONCE ONLY).
LOGICAL UNIT n MISSING FROM ACTIVE SWAPPING LIST	A unit is missing from the active swapping list. This can happen if a unit is off-line or down. This error need not be corrected since the monitor will order the swapping list accordingly. (ONCE ONLY).
LOGICAL UNIT n MISSING FROM STR DSKn	A unit is missing from a file structure and must be remedied. The operator should check that all proper packs are mounted and on-line. If this is not so, the operator should add the proper packs and restart the monitor at 140. Otherwise he has to dissolve the file structure, redefine it, and refresh it, thereby destroying any data already on the unit. (ONCE ONLY).
?LOGIN PLEASE	A command that requires the user to be logged in has been typed to the monitor; it cannot be accepted until the user performs a LOGIN.
MORE THAN ONE LAST UNIT IN ACTIVE SWAPPING LIST	The active swapping list specified in the disk unit HOM blocks has more than one unit as the last one. The operator should redefine the units in the active swapping list to correct this situation. (ONCE ONLY).
MORE THAN ONE LAST UNIT IN STR DSKn	The file structure has more than one unit specified as the last unit as recorded in the disk home blocks. The operator should dissolve the file structure and redefine it. (ONCE ONLY).
MOUNT COMPLETE	The file structure is mounted and ready for use. (MOUNT).
?MOUNT UNSUCCESSFUL	The MOUNT command has not completed successfully. In most cases, the reasons for failure have already been listed by non-error messages.
?MUST BE IN OWNER'S PROJECT FOR SINGLE ACCESS	The user may not request single-access (/SINGLE switch) unless he has the same project number as the owner of the file structure. This requirement is enforced since a user with single access may execute super-USETI/USETO UUOs. (MOUNT).
n BLOCKS ALREADY ALLOCATED	The file already exists. The new specification replaces the old specification rather than updating the old one.

Table F-1 (Cont)
Monitor Diagnostic Messages

NESTING TOO DEEP	The @ construction exceeds a depth of nine and may be due to a loop of @ command files. (COMPIL).
NEW UFD CREATED ON STRUCTURE RE-SERVED (n) F.C.F.S. (n) LOGGED-OUT (n)	An initial UFD has been created on the file structure for the user. The numbers are block quotas as established by QUOTA.SYS for this file structure. (MOUNT).
?nK OF CORE NEEDED	There is insufficient free core to load the file.
?n1K BLOCKS OF CORE NEEDED	The user's current core allocation is less than the contents of JOBFF.
?NO CORE ASSIGNED	No core was allocated when the GET command was given and no core argument was specified in the GET.
NONE PENDING	None of the user's requests to the operator are pending.
?NON-EXISTENT DRIVE DPAn	The user has specified a drive that does not exist in the system. (MOUNT).
?NON-EX MEM AT USER adr	Usually due to an error in the monitor.
?NO START ADR	Starting address or reenter address is zero, because the user failed to specify the starting address.
?NO SUCH DEVICE	The device name does not exist or all devices of this type are in use.
NO SUCH FILE file.ext	Specified file could not be found. Could be a source file or a file required for operation of the COMPIL commands. (COMPIL).
?NO SUCH JOB	An attempt was made to attach to a job that has not been initialized.
NO SUCH UNIT	The unit does not exist or all units of this type are in use.
?NO SUCH TTY	The console number is not part of the system configuration.
?NOT A SAVE FILE	The file is not a core image file.
NOT A FILE STRUCTURE	The file structure specified is not recognized by the monitor.
?NOT A JOB	The job number is not assigned to any currently running job.
NOT ENOUGH CORE	System cannot supply enough core to use as buffers or to read in a CUSP. (COMPIL).
NOT ENOUGH DRIVES	There are currently not enough drives of the right type to mount the file structure. (MOUNT).

Table F-1 (Cont)
Monitor Diagnostic Messages

NO UNITS IN ACTIVE SWAPPING LIST	None of the on-line units are in the active swapping list. Since there must be swapping space, the operator must change the active swapping list to include a unit which has some swapping space. If there are no units with swapping space, the operator must define swapping space on a unit not in a file structure. If all units are in file structure, the operator must refresh a file structure, define the necessary swapping space, and redefine the active swapping list. (ONCE ONLY).
OPERATOR BUSY, HANG ON PLEASE.	The user must wait for the operator to become available.
OPERATOR HAS BEEN NOTIFIED.	The operator is available and the user may continue typing his message.
OPERATOR REQUESTED TO MOUNT UNITS	A request is queued to the operator to mount and ready the packs on the proper drives. (MOUNT).
OPERATOR REQUESTED TO READY DRIVES	One or more drives (as specified by previous messages) are not ready. A request is queued to the operator. (MOUNT).
OPERATOR REQUESTED TO REMOVE PACKS	A request to physically remove the packs has been queued to the operator. (DISMOUNT).
OTHER USERS - CANNOT SINGLE ACCESS	Other users are currently using the file structure that has been specified with the single-access switch (/SINGLE). The switch is ignored. (MOUNT).
OTHER USERS - CAN'T REMOVE	A DISMOUNT command requesting physical removal (/REMOV switch) of a pack has been issued and there are other users of the pack. The switch is ignored. (DISMOUNT).
?OUT OF BOUNDS	The specified adr is not in the user's core area, or the high segment is write-protected and the user does not have privileges to the file that initialized the high segment.
?OUTPUT DEVICE dev CANNOT DO INPUT AT USER adr	An attempt was made to input from an output device (e.g., the line printer).
OUTPUT ERROR	An I/O error occurred while writing a temporary command file on disk. (COMPIL).
PAUSE... (↑C TO QUIT, CR TO CONT)	The PAUSE switch has been specified, and an operator action is about to be requested. ↑C aborts the command before the request is queued to the operator. Carriage return-line feed allows the command to continue, and the request is queued to the operator.
?PC OUT OF BOUNDS AT USER adr	An illegal transfer has been made by the user program to user location adr.

Table F-1 (Cont)
Monitor Diagnostic Messages

?PLEASE TYPE ↑C FIRST	A command which would start a job has been issued after a CSTART or CCONT.
PROCESSOR CONFLICT	Use of + construction has resulted in a mixture of source languages. (COMPIL).
?PLEASE KJOB OR DETACH	Attempt was made to LOGIN a job when the user already has a job initialized at that Teletype.
SAT BLOCK HARDWARE ERROR	The ONCE ONLY dialog has had a hardware error while reading one of the SAT blocks. (ONCE ONLY).
SECOND BAT BLOCK CONSISTENCY ERROR	The ONCE ONLY dialog has discovered that the second of two redundant BAT blocks does not contain some of the data normally expected in a BAT block. This is not a fatal error since the other BAT block is probably all right. If both BAT blocks have this error, the operator should initialize the BAT blocks. This error may occur if some of the diagnostics are run. (ONCE ONLY).
SECOND HOM BLOCK CONSISTENCY ERROR	The ONCE ONLY dialog has discovered that the second of two redundant HOM blocks does not contain some of the data normally expected in a HOM block. Therefore, none of the data should be considered valid. This is not a serious error since the other HOM block is usually all right. If both HOM blocks have consistency errors, the operator has to dissolve the file structures, redefine and refresh. (ONCE ONLY).
SECOND HOM BLOCK HARDWARE ERROR	The ONCE ONLY dialog has had a hardware error while reading or writing the second of two redundant HOM blocks. This is not fatal since there is another HOM block. The controller status is put in the console lights, and the controller is left in its error condition. (ONCE ONLY).
?SINGLE-ACCESS BY JOB n	The file structure is already single access by the indicated user. (MOUNT).
STRUCTURE ALREADY MOUNTED	The requested file structure is already mounted, but may not be in a readied state. (MOUNT).
?STRUCTURE NOT IN STRLST. SYS	The file structure name does not exist in the system administrator's file SYS:STRLST.SYS, and therefore is not defined for the system. The operator or administrator may be requested to define the file structure by adding it to STRLST.SYS with the REACT CUSP. (MOUNT).
?SWAP READ ERROR	A consistent checksum error has been encountered when checksumming locations JOBDAC through JOBDAC+74 of the job data area during swapping.

Table F-1 (Cont)
Monitor Diagnostic Messages

?SYNTAX ERROR	There is a syntax error in the command string.
?SYSTEM ERROR - xxxxxx	System errors designate operator or system errors and are not a direct fault of the user. They are typed for possible diagnostic use.
?TOO FEW ARGUMENTS	A command has been typed, but necessary arguments are missing.
TOO MANY FILE STRUCTURES	The number of file structures exceeds the capacity of the monitor data base. The current limit is 14_{10}. (ONCE ONLY).
TOO MANY NAMES or TOO MANY SWITCHES	Command string complexity exceeds table space in the COMPIL CUSP. (COMPIL).
TOO SMALL - MIN. # = X	An answer to the ONCE ONLY dialog or a default value is too small. Type in an answer greater than or equal to X.
?TRANSMISSION ERROR	During a SAVE, GET, or RUN command, the system received parity errors from the device, or was unable to read the user's file in some other way. This can be as simple as trying to write on a write-locked tape.
?TRY LARGER ARG	The specified argument is too small for the program.
?TTYn ALREADY ATTACHED	Job number is erroneous and is attached to another console, or another user is attached to the job.
TWO LOGICAL UNIT n's FOUND IN ACTIVE SWAPPING LIST	The active swapping list has more than one unit in the same position. The operator must redefine the active swapping list. (ONCE ONLY).
TWO LOGICAL UNIT n's FOUND IN STR DSKn	Two units are marked to be in the same logical position in the file structure. This happens only if two different file structures have been given the same name. The operator should try to remove the pack that does not belong and then restart the monitor at 140. Otherwise, he has to dissolve DSKn, redefine it and refresh it. (ONCE ONLY).
TWO LOGICAL STR n's FOUND IN "SYS" SEARCH LIST	Two file structures are marked to be in the same position in the SYS search list. The operator should change the SYS search list when asked. Refreshing is not required. (ONCE ONLY).
UFD QUOTAS CHANGED, RESERVED (n) F.C.F.S (n) LOGGED-OUT (n)	The block quotas on this file structure as established by QUOTA.SYS have changed since the user's last use of the file structure. The user's UFD will be changed to specify the indicated quotas. (MOUNT).
?UNDEFINED SWITCH switch	The specified switch is either undefined or not unique. (MOUNT, DISMOUNT).

Table F-1 (Cont)
Monitor Diagnostic Messages

unit # BAD REGIONS = m # BAD BLOCKS = n DO YOU WANT TO INITIALIZE THE BAD BLOCKS ON THIS UNIT?	The operator should answer with N or a carriage return to leave the BAT blocks alone on this unit. The only time the operator should initialize is the first time the disk is written, since the blocks contain the accumulated information about bad sectors. If the operator answers Y, the ONCE ONLY dialog responds with NOT NORMALLY DONE, ARE YOU SURE?. Answer Y only if this important data is to be erased. (ONCE ONLY).
UNIT ALREADY IN ACTIVE SWAPPING LIST	An attempt was made to specify a unit to be in the active swapping list more than once. The operator should type a different unit name to be in the active swapping list. If the operator has included the unit name earlier by mistake, he will have another chance to change the active swapping list. (ONCE ONLY).
UNIT ALREADY IN FILE STRUCTURE	An attempt was made to specify a unit to be in more than one file structure. The operator should type a different unit name to be in this file structure. If the operator has included the unit in an earlier file structure by mistake, he will have to dissolve it. (ONCE ONLY).
UNIT id ALREADY MOUNTED ON DRIVE DPAn	The file structure is already mounted but is on different drives than the user specified. (MOUNT).
UNIT HAS NO SPACE ALLOCATED FOR SWAPPING	An attempt has been made to specify a unit which has no swapping space allocated to be part of the active swapping list. The unit is not added to the list. The operator should do one of the following: 1) specify another unit, 2) type an extra carriage return signifying completion, 3) define swapping space for a unit not in a file structure, 4) change the swapping space for a unit in a file structure and refresh it. (ONCE ONLY).
UNRECOGNIZABLE SWITCH	An ambiguous or undefined word has been preceded by a slash. (COMPIL).
?UUO AT USER adr	This message accompanies many error messages and indicates the location of the UUO that was the last instruction the user program executed before the error occurred.
WAITING...	A request has been queued to the operator and the command is waiting for completion of the necessary action. If the user does not want to wait for confirmation, he may type control-C. (MOUNT, DISMOUNT).

Table F-1 (Cont)
Monitor Diagnostic Messages

?WASN'T DET	The specified device is not detached.
X	If the system is fully loaded any user (after the first user) who attempts to LOGIN receives this character in response to any character typed. (LOGIN).
?1+1nK CORE VIR. CORE LEFT = 0	The swapping space or the core allocated to time-sharing is all in use (i.e., there is no available virtual core).

Appendix G
Filename Extensions

Table G-1
Filename Extensions

Filename Extension	Meaning
AID	Source file in AID language
ALG	Source file in ALGOL language
ALP	Printer forms alignment
BAK	Backup file from TECO or LINED
BAS	Source file in BASIC language
BIN	Binary file
BLI	Source file in BLISS language
CAL	CAL data and program files
CBL	Source file in COBOL language
CCL	Alternate convention for command file (@ construction for CUSPs other than COMPIL)
CKP	Checkpoint core image file created by COBOL operating system
CMD	Command file for indirect commands (@ construction for COMPIL)
COR	Correction file for SOUP
CRF	CRE (cross-reference) input file
CTL	MP batch control file
DAE	Default output for DAEMON-taken core dumps
DAT	Data (FORTRAN) file
DCR	Core image save (DCORE)
DDT	Input file to FILDDT
DIR	Latest directory from FILE command for this DECtape
DMP	PDP-6 format for a file created by a SAVE command

Table G-1 (Cont)
Filename Extensions

Filename Extension	Meaning
DOC	Listing of modifications to software
F4	Source file in FORTRAN language
FRM	Form
FUD	FUDGE2 listing output
HGH	Nonsharable high segment
HLP	Help files containing switch explanations
LOG	MP batch log file
LOW	Low segment of a two-segment program
LSD	Default output for DUMP CUSP
LSQ	Queue listing
LST	Listing data
MAC	Source file in MACRO language
MAN	Manual (documentation) file
MAP	Loader map file
MEM	Memorandum file
MSB	Music compiler binary output
MUS	Music compiler input
OPR	Installation instructions
OVR	COBOL overlay file
PAL	Source file in PAL 10 (PDP-8 assembler)
PBT	P-batch control file
PLG	P-batch log file
QUE	Queue (MPB) control or data file
REL	Relocatable binary file
RIM	RIM loader file
RMT	Read-in mode (RIM) format file (PIP)
RNO	RUNOFF input file
RSP	Script response time log file
RTB	Read-in mode (RIM10B) format file (PIP)
SAV	Low segment from a one-segment program
SCP	SCRIPT control file
SFD	Sub-file directory

Table G-1 (Cont)
Filename Extensions

Filename Extension	Meaning
SHR	Sharable high segment file
SYS	Special system files
TEC	TECO macro
TMP	Temporary files
TXT	Text file
UFD	User file directory
UPD	Updates flagged in margin (SRCCOM)
WCH	SCRIPT monitor (WATCH) file
XPN	Expanded save file (FILEX)

Appendix H

Comparison of Disk-Like Devices

Table H-1
Disk Devices

Device Name	Fixed-Head disk	Drum	Removable disk pack(s)		Mass disk file	Movable head disk
Manufacturer	Burroughs	Bryant	Memorex		Bryant	Data-Products
Device Type	RD10	RM10B	RP01	RP02	RB10B	270
Controller	RC10	RC10	RP10	RP10	RA10	-
Maximum Disks per Controller	4	4	8	8	1	4
Maximum Controller per System	2			2	2	0
Hardware Mnemonic	DSK	DSK	DPC	DPC	MDF	DF
Software Mnemonic	FHA,FHB	FHA,FHB	DPA,DPB	DPA,DPB	MDA,MDB	
Capacity Minimum (X10**6 words)	.5	.345	1.3	5.2	21	5.7
Maximum (1 control) (X1**06 words)	2	1.38	10.1	41.4	104.8	23
Blocks/Track	20	30	5	10	64	
Blocks/Cylinder	4000	2700	50	200	300,1600	
Blocks/Unit	4000	2700	10150	40600	163840,819200	
Rotational Speed (rpm)	1800	3600	2400	2400	1200	1200
Revolution Time (msec)	33	17	25	25	50	50
128-Word Blocks/ Revolution	20	30	5	10	16,11,5	7.4
Transfer Rate usec word	13	4.3	30	15	23,32,72	51.8,88

Table H-1 (Cont)
Disk Devices

Device Name	Fixed-Head disk	Drum	Removable disk pack(s)		Mass disk file	Movable head disk
Manufacturer	Burroughs	Bryant	Memorex		Bryant	Data-Products
Device Type	RD10	RM10B	RP01	RP02	RB10B	270
Controller	RC10	RC10	RP10	RP10	RA10	-
Seek Time						
Average (msec)	0	0	50	50	110	175
Minimum (msec)	0	0	20	20	50	80
Maximum (msec)	0	0	80	80	180	225
Swapping Times (msec)			(Includes 30 ms verify)			
1K	25	13	95	84	175	262
4K	73	27	225	144	284	454
10K	154	54	490	264	502	1062
25K	358	120	1165	589	1048	2462

NOTE

The dual-positioning Bryant disk is not supported by DEC software, only the single-positioning disk is supported. 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 I
Retrieval Pointers

Sequential and random file access is handled more efficiently by compacting the amount of information necessary to describe the location of a file. Retrieval information associated with each file is stored and accessed separately from the data. Retrieval pointers describe contiguous blocks of file storage space called groups. Each pointer has one of three forms:

a. A group pointer
b. An EOF pointer
c. A change of unit pointer.

I.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, which 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 the cluster size. A cluster address is converted to a logical block address by multiplying by the number of blocks per cluster.

I.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 eliminates the usual overflow problem associated with folded checksums and terminates as soon as there are no more bits to add.

I.2 END-OF-FILE POINTER

The EOF is indicated by a zero word.

I.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. A method of indicating a change from one unit to another in the middle of the file is necessary, because a file can start on one device and move to another. 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.

Appendix J
Once–Only Parameters

The operator must invoke the optional once-only dialog when the monitor is first loaded when file structure or disk unit parameters must be changed or file structures must be refreshed. When the operator changes one or more parameters, the new values are written back onto the home blocks of the affected disk units. Some parameter changes require refreshing the file structures.

J.1 FILE STRUCTURE PARAMETERS

The following file structure parameters may be changed without refreshing:

a. Number of consecutive blocks tried for on output.
b. Sum of the blocks guaranteed to users.
c. Number of blocks per user allowed for overdrawing.
d. Inclusion and position in SYS search list.

The following parameters may not be changed without refreshing:

a. Number of K for CRASH.SAV file.
b. Number of blocks per cluster.
c. Number of bits per cluster count.

J.2 PHYSICAL UNIT PARAMETERS

The following physical unit parameters may be changed without refreshing:

a. Number of SAT blocks in core.
b. Active swapping list.
c. Swapping classes.

The following parameters may not be changed without refreshing:

a. Physical unit ID
b. Logical blocks for both home blocks

c. K allowed for swapping on the unit

d. First logical block for swapping

e. Computed first logical block for swapping

f. Number of SAT blocks per unit.

J.3 SYSTEM PARAMETERS

These parameters are not rewritten on the disk:

a. Number of monitor buffers.

Glossary

Absolute Address	The address that is permanently assigned by the machine designer to a storage location.
Access List	The table in core that reflects the status of all files open for reading or writing in addition to the status of those files recently closed.
ACCT. SYS	The file that contains all project-programmer numbers, passwords, and time of day users are allowed on system. It does not contain file structure quotas.
Active Search List	An ordered list of file structures for each job, which specifies the order in which the directory is searched. Device DSK is defined by this list for each job.
Actual Transfer	The third step of the transfer operation. The operation passes data between the memory system and the control.
AUXACC. SYS	The file that contains the standard list of public file structures for each user and information, such as quotas, for those file structures.
Bad Allocation Table (BAT) block	A block written by the MAP program or the monitor on every unit. This block enumerates the bad regions of consecutive bad blocks on that unit. The BAT blocks appear in the HOME. SYS file.
BADBLK.SYS	The file that contains all bad blocks. It may be read but not deleted and is useful for testing error recovery.
Block	A 128-word unit of storage determined by hardware and software. At least 128 words are written, adding zeros as necessary, although less than 128 words may be read.
Cluster	A possibly multiblock unit of storage assignment.
Compressed File Pointer	An 18-bit pointer to the unit within the file structure and to the first super-cluster of the file.
Control	The device that controls the operation of up to eight connected units. It initiates simultaneous positioning commands to some of its units and then performs a data transfer for one of its units.
CORMAX	The largest contiguous size that an unlocked job can be. This value can range from CORMIN to total user core.
CORMIN	The guaranteed amount of contiguous core which a single unlocked job can have. This value can range from 0 to total user core.

CRASH.SAV	A file written on disk as a contiguous set of blocks by a short routine at the crash restart location in the monitor. Used for analysis by file salvage program and by FILDDT for system debugging.
Create	To open a file for writing, write the file, and close the file for the first time. Only one user at a time may create a file with a given name and extension in the same directory of a file structure.
Customer	A DEC customer who has a PDP-10 system as distinguished from a user at a console who may be purchasing time from a customer.
Cylinder	The hardware-defined region of consecutive logical disk blocks, which can be written with one DATAO instruction. This region does not require positioning.
Data Channel	The device that passes data between the memory system and the control.
Device Routines	Routines that perform I/O for specific storage devices and translate logical block numbers to physical disk addresses.
Directory Device	A storage retrieval device, such as disk or DECtape, which contains a file describing the layout of stored data (programs and other files).
Distributed UFD	A UFD that is distributed over the file structures on which the user can read or write.
Dormant File Structure	A file structure that is physically mounted but has no current users, i.e., the mount count is zero.
Dormant Segment	Description of a sharable high segment kept on swapping space, and possibly core, which is in no user's addressing space.
DSK	The generic device name for disk-like devices (e.g., drums, disk, disk packs). Actual file structure names are defined for each job by the file structure search list.
Dump	A listing of all variables and their values or a listing of the values of all locations in core.
File Structure	The logical arrangement of 128-word blocks on one or more units of the same type to form a collection of named files. Public file structures are given a name for use by all users and are always on the system. Private file structures are intended to be mounted and dismounted from the system.
File Structure Search List	A table, organized by jobs, that specifies the search order for the file structures the user can access.
File-Structured Device	A device on which data is given names and arranged into files; the device also contains directories of these names.
Filename	A one- to six-alphanumeric character name chosen by the user to identify a file.
Filename Extension	One to three alphanumeric characters usually chosen to describe the class of information in a file.
FILSER	The routine that interprets and operates on the file structure, processes disk UUOs, queues disk requests, and makes optimization decisions.

Fragmentation	The term applied to swapped segments, which cannot be allocated in one contiguous set of blocks on the swapping space.
Group	A contiguous set of clusters allocated as a single unit of storage, and described by a single retrieval pointer.
High Segment	The segment of the user's core that generally contains pure code, and that can be shared by other jobs; it is usually write-protected.
Home Block	The block written on every unit, which identifies the file structure the unit belongs to and its position on the file structure. This block specifies all the parameters of the file structure and the location of the MFD. The home block appears in the HOME.SYS file.
HOME. SYS	The file that contains a number of special blocks for system use. These blocks are the home blocks, the BAT blocks, the ISW blocks, and block zero.
Idle Segment	A sharable high segment that no users in core are using; however, at least one swapped-out user is using it; otherwise, it would be a dormant segment.
Idle Time	The percent of uptime in which no job wanted to run, i.e., all jobs were HALTed or were in a wait for some device.
Impure Code	The code that is modified during the course of a run (e.g., data tables).
ISW Block	A block written by the refresher, which contains the bit map for the initial storage allocation table for swapping. Any bad regions are marked as already in use. The ISW block appears in the HOME.SYS file.
Job	The succession of user programs run from log in to log out for a single user; a process.
Job Data Area	The first 140 octal locations of a user's core area. This area provides storage for items used by both the monitor and the user program.
Job Search List	See File Structure Search List.
Latency	a. The time for initiation of a transfer operation to the beginning of actual transfer (i.e., verification plus search time). b. The time delay while waiting for a rotating memory to reach a given location as desired by the user. The average latency is one half the revolution time.
Locked Job	A job in core that is never a candidate for swapping or shuffling.
Lost Time	The percent of uptime that the null job was running but at least one other job wanted to run (i.e., was not waiting for a device) but could not because one of the following was true: 1) the job was being swapped out. 2) the job was being swapped in. 3) the job was on disk waiting to be swapped in. 4) the job was momentarily stopped so devices could become inactive in order to shuffle job in core.
Low Segment	The segment of core containing the job data area and I/O buffers. This area is unique and accessible to the user and is often used to contain the users' program. If the user is working with a shared program, this area contains data tables.

MAINT. SYS	The area of the disk reserved for maintenance use.
Memory Protection	A scheme for preventing access to certain areas of storage for reading or writing.
MONGEN Time	The time at which the monitor is loaded because of new equipment being added. Therefore the hardware configuration must be changed.
Monitor	The specific program that schedules and controls the operation of several routines, performs overlapped I/O, provides context switching, and allocates resources so that the computer is efficiently used.
Multiprocessing	The simultaneous execution of two or more computer programs by a computer.
Multiprogramming	A technique that allows scheduling so more than one job is in an executable state at any one time.
Named File	A named collection of 36-bit words (instructions/data). Length is not restricted by size of core.
Nondirectory Device	A device (e.g., magnetic tape or paper tape) that does not contain a file describing the layout of stored data.
Nonsharable Segment	A segment for which each user has his own copy. Nonsharable segments never have names even if initialized from a file; they may be created by a core or REMAP UUO.
ONCE ONLY Time	The time at which the operator may change file structure and disk unit parameters.
Pack ID	A 6-character SIXBIT name or number used to uniquely identify a disk pack. This name is appended to the physical device list of the monitor.
Passive Search List	An unordered list of the file structures the job can access explicitly. Device DSK is not defined by this list.
Peripheral Equipment	In a data processing system, any unit of equipment distinct from the central processing unit, which may provide the system with outside communication.
Physical Unit Name	The SIXBIT name, consisting of three characters and a digit, which is associated with each unit. Examples: FHA0, FHA1, MDA0, MDA1, DPA0, DPA7.
Pointer	The location containing an address rather than data, which is used in indirect addressing.
Pool	One or more logically complete file structures that provide file storage for the users and that require no special action on the part of the user.
Position Operation	The operation of moving the read-write heads of a disk to the proper cylinder prior to a data transfer. This operation requires the control for several microseconds to initiate activity, but does not require the channel or memory system.

Private Disk Pack	a. A self-contained DEC file structure either single- or multi-job access. b. A disk pack (always single-job access) from another company, not certified, or certified but never refreshed.
Privileged Program	a. Any program running under project number 1, programmer number 2. b. A monitor support CUSP executed by a monitor command and, therefore, has the JACCT (job status bit) set (e.g., LOGOUT).
Priority Interrupt	The interrupt that usurps control of the computer program or system and jumps the sequencing to another device or program.
Program Break	The length of a program; the first location not used by a program (before relocation); the relocation constant for the program (after relocation).
Programmed Operators	Instructions, which, instead of doing computation, cause a jump into the monitor system at a predetermined point. The monitor interprets these entries as commands from the user to perform specified operations.
Public Disk Pack	A disk pack belonging to the storage pool with storage available to all users.
Pure Code	Code that is never modified in the process of execution; therefore, it is possible to let many users share the same copy of a program.
QUOTA. SYS	The file that contains a list of users and their quotas for the private file structure on which the file resides.
Random Access	A device in which the access time is effectively independent of the location of the data.
Read	To open a file for input.
RECOV. SYS	The file used only for disk crash recovery; temporary information is stored in this file when the disk is salvaged in place.
Reentrant Program	A two-segment program composed of a sharable and nonsharable segment.
Refresh	To remove files from a unit and write just a MFD, a few UFDs, a SAT.SYS, a HOME.SYS, and a few system files on the disk.
Relocate	To move a routine from one portion of storage to another and to adjust the necessary address references so that the routine can be executed in its new location.
Restore	To copy a file previously dumped on magnetic tape back onto disk.
Retrieval Information Block (RIB)	The block that contains pointers to all the groups in the file. Each file has two copies of the RIB, the first block of the first group and the block following the last data block in the last group of the file.
SAT. SYS	This file is the Storage Allocation Table file and contains a bit for each cluster in the file structure. Clusters which are free are indicated by zero and clusters which are bad, allocated, and nonexistent are indicated by one.
Search	The second step in the transfer operation, the controller reads sector headers to find the correct sector.

Search List	See File Structure Search List.
Sharable Segment	A segment which can be used by several users at a time.
Shared Code	Pure code residing in the high segment of user's core.
STRLST.SYS	The file that describes each file structure in the system. This file is used by the MOUNT command only.
Supersede	To open a file for writing, write the file, and close the file two or more times. Only one user at a time may supersede a given file at any one time. The older copy of the file is deleted when all users are finished reading it.
Super-Cluster	A contiguous set of one or more clusters introduced to compress the file pointer for large units into 18 bits (see Compressed File Pointer).
Swapping	The movement of program sections between core and secondary storage.
Swapping Class	The classes of swapping units divided according to speed. Class 0 contains the fastest swapping units.
SWAP.SYS	The file containing the swapping area on a file structure.
Total User Core	The amount of physical core which can be used for locked and unlocked jobs.
Transfer Operation	The operation of connecting a channel to a controller and a controller to a unit from passing data between the memory and the unit. The transfer operation involves verification, search, and actual transfer.
Trap	An unprogrammed conditional jump to a known location, automatically activated by hardware. The location from which the jump occurred is recorded.
Unit	The smallest portion of a device that can be positioned independently from all other units (e.g., a Burroughs disk, a side of the Bryant disk, a disk pack, and a drum).
Update	To open a file for reading and writing simultaneously on the same software channel, rewrite one or more blocks in place, and close the file. Only one user at a time may update a given file.
User	A person at a terminal.
User Mode	A hardware-defined state during which instructions are executed normally except for both I/O and HALT instructions, which cause immediate jumps to the monitor. This makes it possible to prevent the user from interferring with other users or with the operation of the monitor; memory protection and relocation are in effect so that the user can modify only his area of core.
User Program	All of the code running under control of the monitor in an addressing space of its own.
Verification	The first step in the transfer operation. The controller reads sector headers to see if the mechanical parts of the system have correctly positioned the arm.
Vestigial Job Data Area	The first ten octal locations of the high segment used to contain data for initializing certain locations in the job data area.

Virtual Core	The amount of core space that the user appears to be able to use. This area is usually handled by a program that allows the currently referenced parts of the program to be in core at one time, with additional information being retrieved from storage as needed.
1-4. UFD	The UFD of device SYS. The UFD number should be obtained via the DEVPPN UUO.
1-1. UFD	The master file directory, which contains all UFD files (including itself) as directory entries. The filename is in octal and, therefore, is right justified in each half word. This file is created at refresh time.

INDEX

G

H

I

J

P

Q

R

T

U

PDP-10
TIME-SHARING MONITORS
DEC-T9-MTZD-D

READER'S COMMENTS

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

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

_____

_____

_____

_____

Did you find errors in this manual? _____

_____

_____

_____

_____

How can this manual be improved? _____

_____

_____

_____

_____

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

☐ **Software Manual Update,** a quarterly collection of revisions to current software manuals.

☐ **User's Bookshelf,** a bibliography of current software manuals.

☐ **Program Library Price List,** a list of currently available software programs and manuals.

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

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

Postage will be paid by:

digital

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