Book 3

Communicating with the Monitor

Time-Sharing Monitors

FOREWORD

This manual covers the use of the Time Sharing Monitors, which include the Multiprogramming non-disk Monitor and the Multiprogramming disk Monitor (formerly known as 10/40) and the Swapping Monitor (formerly known as 10/50).

The Single-User Monitor (formerly known as 10/20, 10/30) is covered in the manual Single User Monitor Systems.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF TABLES (Cont)

CHAPTER 1
INTRODUCTION - MONITOR CAPABILITIES

This book discusses the commands, program loading procedures, and user programming facilities available under the PDP-10 Time-Sharing Monitors - three multiprogramming, time-sharing systems designed to allow 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 located 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 jobs by using the batch control processor (BATCH). Real-time jobs can operate either as independent user programs or as fully integrated Monitor subroutines.

The Multiprogramming non-disk Monitor (formerly called the 10/40 Monitor) is a multiprogramming, time-sharing system which includes I/O control of all devices attached to the system, run-time selection of I/O devices, job-to-job transition, job save and restore features, and dynamic debugging facilities. All of these features are incorporated with concurrent real-time processing, batch processing, and time sharing. The Multiprogramming disk Monitor adds a comprehensive file system with both sequential and random access of shared, named files to the Multiprogramming non-disk system. The Swapping Monitor (formerly called the 10/50 Monitor) has all the features of the Multiprogramming disk system and, in addition, swaps programs between

high-speed disk and core, thereby increasing the number of users that can be accommodated simultaneously.

1.1 Reentrant User-Programming Capability

The number of users that can be handled by a given size time-sharing configuration is further increased by adding a reentrant user-programming capability to the system. This means that a sequence of instructions may be entered by more than one user process at a time. A single copy of a reentrant program may be shared by a number of users at the same time, thereby increasing system economy. All the versions of the Time-sharing Monitor 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 even though 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 Monitor allows one of the segments of each user area to be the same as one or more other users, so that only one physical copy of a shared segment need exist no matter how many users are using it. The Monitor normally invokes hardware write-protection for shared segments to guarantee that they are not accidentally modified.

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

a) core memory, since only one copy of a shared segment exists for the entire system (Figure 1-1 illustrates this efficient

use of core memory),

b) swapping storage, since many users share the single copy of the shared segment kept in swapping storage,

c) swapping I/O channel, since a shared segment is read into core only once and is not written back onto swapping storage unless modified, and

d) file storage I/O channel, since 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 the non-reentrant system.

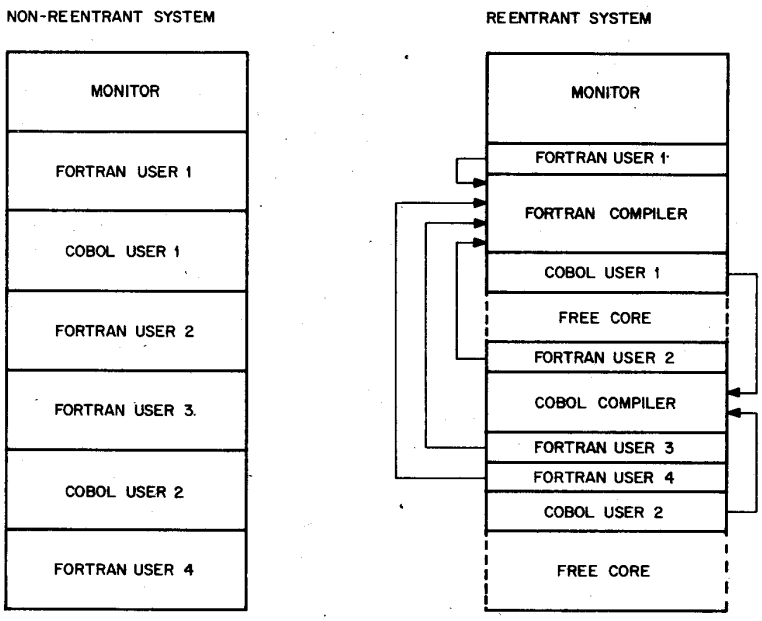


Figure 1-1
Core Management

1.2 MONITOR FUNCTIONS

The Time-Sharing Monitors act as the interface between the user and the computer so that all users are protected from one another and appear to have most resources available to themselves. The Monitors schedule multiple-user time sharing of the system, allocate available sharable resources to user programs, accept input from and direct output to all system I/O devices, and relocate and protect user programs in core memory.

The Monitors utilize the PDP-10 hardware features for memory protection, memory relocation, executive/user mode, and real-time clock to provide an advanced, third-generation, multiprogramming time-sharing environment. System facilities start with a minimum configuration of 16K core and two DECtapes and can accommodate magnetic tapes, disks, drums, communication line controllers, card readers and punches, paper tape readers and punches, line printers, displays, incremental plotters, and user Teletype consoles. Other special devices, including real-time digitizers and analog converters, easily interface with the system.

Several user programs are loaded into core at once and the Time-Sharing Monitors schedule each program to run for a certain length of time, utilizing a scheduling algorithm that makes efficient use of system capabilities. The Monitors direct data flow between I/O devices and the user programs, making them device independent, and overlap I/O operations concurrently with computation for high system efficiency.

1.2.1 Job Scheduling

One of the parameters which must be specified in

creating a PDP Time-Sharing Monitor is the number of jobs which may be run simultaneously. Up to 127 jobs may be specified. Each user who accesses the system is assigned a job number. The term job is used to refer to the entire sequence of operations the user initiates from his console.

In a multiprogramming system all jobs reside in core, and the scheduler decides which of these jobs should run. In a swapping system jobs can exist on an external storage device (usually disk) as well as in core. The scheduler decides not only which job is to run but also when a job is to be swapped out onto the disk or brought back into core.

In the Swapping Monitor, 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 point in time. The possible queues a job may be in include the following.

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 of these queues is addressed through tables. The position of a queue's 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 this 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 first job's input 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. Now the input operation of the first job finishes, making the DECtape control available to the second job. The second job's I/O operation 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 third job's running. 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.

Scheduling occurs at each clock tick (1/60th or 1/50th of a second) or may be forced at Monitor level between clock ticks

if the current job becomes 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 is dependent upon (1) the core shuffling routine, which consolidates unused areas in core so as to make sufficient room for the incoming job, and upon (2) 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 job to be run after that into core. The transfer from disk to core takes place while the central processor continues computation for the previous job.

1.2.2 Use of Swapping Space and Physical Core

The reentrant capability reduces the demands on core memory, swapping storage, swapping channel, and storage channel. However, to reduce the use of the storage channel, copies of the sharable segments are kept on the swapping device. This increases the demand for swapping storage. The Monitor achieves this space-time balance dynamically by assuming that there is no shortage of swapping space. The amount of swapping space is fixed by the operator at system initialization. Thereafter, the Monitor keeps a single copy of as many sharable segments as possible in the swapping 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 is currently unused, it is called a dormant segment. If the Monitor cannot find contiguous free space on the swapping device, it will fragment the high and low segments of the user whose job is being swapped out. If swapping space runs out, the Monitor deletes a dormant segment and continues to fragment

the user's segments. If and when a deleted segment is needed again, it is retrieved from the storage device.

The Monitor keeps track of the total amount of "virtual core" assigned to all users. 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 Monitor is started the amount of unused virtual core is set equal to the amount of swapping space pre-allocated on the disk. Thus, there is always room to swap out the largest possible job in core and swap in another job.

The same techniques used in allocating swapping space are used to allocate core in both swapping and non-swapping systems. A dormant segment will stay in core until core is needed. In the swapping system, an active write-protected segment remains in core even though no one in core is using it. Some swapped-out user must be using it or else it would be dormant rather than idle.

1.3 USER FACILITIES

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

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 mounting of special tapes,

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, and

g) returns facilities to the system when the job is finished using them.

Chapter 2 describes the various Monitor commands which provide each of these capabilities.

Using Monitor commands, the user at his console can call in programs from the system file. The system file contains programs for creating and editing program source files (TECO,EDITOR), for assembling or compiling program source files (MACRO,FORTRAN, BASIC, COBOL), and for loading relocatable binary files (LOADER). The usage of these and many other CUSPs (Commonly Used Systems Programs) are currently described in the System User's Guide (DEC-10-NGCC-D).

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. Once a particular CUSP has been 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 PIP, Peripheral Interchange Program.
*DSK:TEXT←TTY:	CUSP command level. User instructs PIP to create a file on the disk named TEXT using Teletype console as input medium.
THIS IS FILE TEXT	CUSP I/O level. User types input to PIP.
↑Z	↑Z causes Teletype end of file. Return to CUSP command level.
*↑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 means of PDP-10 operation codes 040 through 077. These op-codes are called UUO's and 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 the job 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 and/or as a possibly fragmented unit on the swapping device. A program or user job is composed of one or two segments. A segment may contain instructions and/or data. The Monitor determines the allocation and movement of segments in core and on the swapping device.

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

The PDP-10's two 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 point in 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 and/or 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 non-sharable, whereas a low segment is always non-sharable. The high segment may be write-protected.

A *reentrant program* is always composed of two segments - a low segment which usually contains just 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 non-sharable high segment. This kind of program is useful when there is a requirement for two fixed-origin data areas to increase and decrease independently during execution.

1.5 FILES

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

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 numbers for disk, 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 which identify the type of information in the file, is usually specified by the program. A named file may be written by a user program in buffered or unbuffered mode, or in both. It may be read and/or modified sequentially or randomly with buffered or unbuffered mode I/O independently of how it was written. Named files are stored on the storage device. Each named file has certain access

privileges associated with it. These privileges designate which users can read or write the file or change its access privileges. In regard to 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.

A file is said to be *created* if no file by the same name existed when the file was opened for writing. A file is said to be *superseded* if another file by the same name already exists. A file is said to be *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.

1.6 COMPARISON OF SEGMENTS AND FILES

Files and segments have certain similarities and differences. Both are named, one-dimensional arrays of 36-bit words. A file can be as long as the size of disk or DECtape. 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 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 storage device 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; it 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 time-sharing 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. In configuring and loading a time-sharing Monitor, the system administrator sets the maximum number of jobs which his system will handle simultaneously. This number may be up to 127 jobs if the system has enough core, disk storage, processor capacity, and time-sharing consoles to handle this load.

Jobs are initiated by users typing on a time-sharing console. A console is typically any of several models of Teletype machines but may also be a CRT (cathode ray tube) with a keyboard. 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 (see Table 2.7) 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 point in 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 to the same console or to any other console in the system.

2.1.1 Monitor Mode and User Mode

From the user's point of view, his console is in one of two states - monitor mode 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 tables below) places the console in user mode. Once the program is in user mode, the console becomes simply an input/output device for that user. In addition, user programs will use the console for two purposes. The user program will accept command strings from the console or will use the console as a direct input/output device. Example:

monitor mode	.R PIP	monitor command
user mode	*DSK:FOO←TTY:	user program command string
•		
user mode	THIS IS FILE FOO↑Z	user program using console as an input device
monitor mode	.R MACRO	monitor command
user mode	*TTY:,←DSK:PROG1	user program command string
user mode		user program using console as an output device
	•	
	•	
	code	
	•	
	•	

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. There are certain commands which

cause the user program to start or continue running (as noted in the tables below) but which 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., all the jobs that the system can accommodate have 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 and/or lower case. Spaces and non-printing characters preceding the command name are ignored. The Monitor Command Interpreter will not interpret or execute a line of comments preceded by a semicolon. Every command to the Monitor Command Interpreter must be terminated by pressing the RETURN key on the console. If the command is not understood, an error message is typed out by the Monitor and the mode is unchanged.

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 tables which follow, the commonly used abbreviation of the command name is shown. Installations which choose to implement additional commands should take care to preserve the uniqueness of the first few 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 Login Check (Disk Monitor Systems)

If a user who has not logged in (see Table 2.1) 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. Login is not required by a non-disk Monitor system.

2.2.4 Job Number Check (Non-disk Monitor Systems)

If the non-disk 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 will then proceed to execute the command.

2.2.5 Core Storage Check

If the Monitor Command Interpreter recognizes a command name which requires core storage to have been 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 will be 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 returned to 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 immediately send an asterisk (*) to the user's console to indicate their readiness to accept user-mode command strings.

2.3 SYSTEM ACCESS CONTROL 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 and programmer numbers are octal numbers up to six digits each. The project-programmer numbers will identify not only the user but also his file storage area on the disk. The password is from one to five ASCII characters. To LOGIN 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 (ACCT.SYS [1,1]).

Table 2-1

Monitor Command to Gain Access to the System

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
LOGIN	LOG	LOGIN initializes a Monitor routine to accept the user's LOGIN data. The following is the procedure used to gain access to the system. .LOGIN JOB n PDP-10 4S.50F Job number assigned to user, followed by Monitor name and version number. # System types out number sign to indicate user should type his project-programmer number. proj,prog User types in his project-programmer number PASSWORD: System requests user to type his password. User types password followed by carriage return. To maintain password security, the Monitor will not echo the password. 1135 8-AUG-69 TTY23 ↑C . If user entries are correct, Monitor responds with time, date, TTY number, ↑C and a period, indicating readiness to accept a command.	u R D	LOGIN PLEASE ? The user has typed a command that the Monitor cannot accept unless the user logs in. ?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 System core and swapping space exceeded.

*Characteristics:

d = places job in detached mode
m = places job in Monitor mode
u = places job in user mode

L = LOGIN required (Disk Monitor)
A = no active device
C = core required

J = requires a job number.
R = runs a CUSP thereby replacing previous program in user's addressing space.
D = available only in Multiprogramming Disk and in swapping systems, not in Multiprogramming non-disk systems.

2.4 FACILITY ALLOCATION COMMANDS

The Monitor allocates peripheral devices and core memory to users upon 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 time-sharing 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.

All devices controllable by the system are listed in Table 5-1. Associated with each device is a physical name, consisting of three letters and zero to three numerals to specify unit number. A logical device name may also be assigned by the user. This 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 by physical names or by assigned logical names.

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

Table 2-2

Monitor Commands to Allocate Facilities

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
ASSIGN[1] phys-dev log-dev	AS	To assign an I/O device to the user's job for the duration of the job or until a DEASSIGN command is given. phys-dev — Any device listed in Table 5-1[2]. This argument is required. log-dev — A logical name assigned by the user	m L J	dev: ASSIGNED The device has been successfully assigned to the job. NO SUCH DEVICE Device name does not exist. ALREADY ASSIGNED TO JOB n The device has already been assigned to another user's job. LOGICAL NAME ALREADY IN USE DEVICE dev: ASSIGNED The user has previously assigned this logical name to another device.
DEASSIGN[1] dev	DEA	Returns one or more devices currently assigned to the user's job to the Monitor's pool of available devices. dev — If this argument is not specified, all devices assigned to the user's job are deassigned. If this argument is specified, it can be either the logical or physical device name.	m L J	NO SUCH DEVICE Device name does not exist. DEVICE WASN'T ASSIGNED The device isn't currently assigned to this job.
REASSIGN dev job	REA	Allows one job to pass a device to a second job without going through the Monitor device pool. dev — The physical or logical name of the device to be reassigned. Cannot be a user console. job — The number of the job to which the device is to be reassigned.	m L J C A	DEVICE dev WASN'T ASSIGNED The device isn't currently assigned to this job. JOB NEVER WAS INITIATED The job number specified has not been initialized. NO SUCH DEVICE The device does not exist. DEVICE CAN'T BE REASSIGNED A user's console Teletype cannot be reassigned.

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
FINISH dev	F	Terminates any input or output currently in progress on the device. dev — The logical or physical name of the device on which I/O is to be terminated. If no name is specified, I/O is terminated on all devices assigned to the job.	m L J C A	NO SUCH DEVICE Either the device does not exist or it was not assigned to this job.
TALK dev	TA	To allow the user to type directly to another user's console. dev — Must be one of the following: CTY - Console Teletype TTYn - Where n can be in the range of 0 through 77. OPR - Operator's console (the Teletype designated as such when the Monitor was initialized).	m	BUSY The console addressed is either (1) not in the Monitor mode or (2) is not positioned at the left margin. (OPR is never busy.)
CORE n	COR	To modify the amount of core assigned to the user's job. n = o — The low and high segments disappear from the job's virtual addressing space. n > o — Total number of 1K blocks of core to be assigned to the job from this point on. If n is omitted, Monitor types out the same response as when an error occurs, but does not change core assignment.	m J A C	10/40 Systems: m/p 10/50 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 swapping systems-max. physical user core non-swapping systems - free + dormant core v = number of K unassigned in core and swapping device

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
RESOURCES	RES	To print out all the available devices (except TTY's) and the number of free blocks on the disk.	m	

*Refer to footnote in Table 2-1

[1]The ASSIGN 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. The DEASSIGN command applied to DECtapes performs the same function. (See 5.7.7 for further details.)

[2]If DTA or MTA is used, the Monitor performs a search for an available drive and then types out DTAn (or MTAn) ASSIGNED.

Examples showing use of logical and physical names:

User types	.ASSIGN DTA,ABC	
Monitor responds	DEVICE DTA6 ASSIGNED	(successful)
User then types	.ASSIGN DTA,DEF	(find another unit)
Monitor responds	NO SUCH DEVICE	(all in use)
User then types	.ASSIGN PTP,ABC	(reserve paper tape punch)
Monitor responds	LOGICAL NAME ALREADY IN USE DEVICE PTP ASSIGNED	(paper tape punch is reserved, but ABC still refers to DTA6 only)
User then types	.ASSIGN DTA1,DEF	
Monitor responds	ALREADY ASSIGNED TO JOB 2	(another user has it)

User then types	.R PIP	(request for system program PIP)
User then types	*PTP:←ABC:FOO	(command string to PIP asking that file FOO be transferred from device ABC (which is now assigned as DTA6) to device PTP (which is assigned to user).

NOTE: The user does not type the period or the asterisk. The period is the Monitor response to the user and the asterisk is the CUSP response. The user must terminate every command to the Monitor Command Interpreter by pressing the RETURN key on the Teletype.

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 from 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 one 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.

The following are the standard meanings for file extensions:

.TMP	Temporary file
.MAC	Source file in MACRO language
.F4	Source file in FORTRAN IV language
.CBL	Source file in COBOL language (available in 1970)
.LST	Listing or CREF data
.REL	Relocatable binary file
.CMD	Command file, for @ construction
.SAV	Core dump, from SAVE command
blank	Unspecified ASCII text file

Each time one of these commands is executed the command with its arguments is "remembered" as a file on the disk. Because of this, the filename last edited 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.

Table 2-3

Monitor Commands to Prepare Source Files

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
EDIT file.ext	ED	Runs LINED (Line Editor for Disk) and opens an already existing sequence-numbered file on disk for editing.	u L R D J	See Table 2-4
CREATE file.ext	CREA	Runs LINED and opens a new file on disk for creation.	u L R D J	See Table 2-4
TECO file.ext	TE	Runs TECO (Text Editor and Corrector) and opens an already existing non-sequence-numbered file on disk for editing.	u L R D J	See Table 2-4
MAKE file.ext	M	Runs TECO and opens a new file on disk for creation.	u L R D J	See Table 2-4

*Refer to footnote in Table 2-1

Table 2-4

Monitor Command Diagnostic Messages

(For File Manipulation Commands)

Message	Meaning
COMMAND ERROR	The COMPIL CUSP cannot decipher the command.
DEVICE NOT AVAILABLE	Specified device could not be initialized.
DISK NOT AVAILABLE	Device DSK: could not be initialized.

Table 2-4 (Cont)

Monitor Command Diagnostic Messages

(For File Manipulation Commands)

Message	Meaning
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 could not be entered in the user's UFD.
INPUT ERROR	I/O error occurred while reading a temporary command file from the disk.
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 could not 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 has 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

Each of the following commands performs complex functions which would require a number of commands on a less sophisticated system. The commands in Table 2-5 list the user's files and file directories and cause his source files to be compiled, loaded, and executed.

Table 2-5

Monitor Commands to Manipulate Files

Commands	Abbreviation	Explanation	Characteristics*	Monitor Messages
TYPE list	TY	Directs PIP (Peripheral Interchange Program) to type contents of named source file(s) on user's Teletype. list — A single file specification, or a string of file specifications separated by commas. A file specification is the same as that described for COMPILE, LOAD, EXECUTE, and DEBUG commands. In addition, the * construction can be used as follows: filename.* — All files with this filename and any extension *.ext — All files with this extension and any filename *.* — All files Examples: TYPE FILEA, DTAO:FILEB.MAC,*.TMP TYPE A,DTA4:B,C[15,107]	m L R D	See Table 2-4
LIST list	LI	Directs PIP to list contents of named source file(s) on the line printer (LPT). Examples: LIST TEST.* LIST *.MAC LIST DTA4:A,B,C	m L R D	See Table 2-4
DIRECT dev	DI	If dev: is omitted or DSK:, directory listing of user's disk files is typed on the user's Teletype. If DTAn: is specified, directory of that DECtape is typed. Two switches can be used with the DIRECT command: /F List short form of directory (i.e., omit dates) /L List on line printer (LPT) instead of Teletype. The : may be omitted in dev.	m L R D	See Table 2-4

Commands	Abbreviation	Explanation	Characteristics*	Monitor Messages
DELETE list	DEL	Deletes one or more files from disk or DECtape. If a device name is specified, it remains in effect until changed or end of command string is reached.	m L R D	See Table 2-4
RENAME arg	REN	Changes the name of one or more files on disk or DEC tape. The arg is a pair of file specifications separated by an = sign, or a string of such pairs separated by commas: RENAME new1 = old1,new2 = old2,... Device names can be specified only with the new filename and remain in effect until changed or end of command string is reached.	m L R D	See Table 2-4
CREF	CREF	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. The file containing the names of these CREF-listing files is then deleted so that subsequent CREF commands will not list them again.	m L R D	See Table 2-4
COMPILE list	COM	Produces relocatable binary file(s) for the specified program(s). The use of the MACRO assembler 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 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) Other than above, or null — "Standard processor" is used (see 2.6.2). The list of files which may be a single file specification, or a string of file specifications separated by commas. A file specification consists of a filename (with or without an extension) and may include a device name (if the source file is not disk) or a project-programmer number (if the source file is not in the user's disk area). Examples: PROG1,PROG1.MAC,PROG1.F4,PROG1.XYZ,DTA0:PROG1 PROG1[10,16],PROGA,DTA0:PROGB PROGC.MAC (See 2.6.1, 2.6.2)	m L R D	See Table 2-4

Commands	Abbreviation	Explanation	Characteristics*	Monitor Messages
LOAD list	LOA	Performs the COMPILE function for the specified program(s), then runs LOADER and loads the .REL files.	m L R D	See Table 2-4
EXECUTE list	EX	Performs the COMPILE and LOAD functions for the specified program(s) and begins execution of the loaded program.	u L R D	See Table 2-4
DEBUG list	DEB	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. Examples: COMPILE PROGA EXECUTE DTA1:TEST.MAC DEBUG/L FILEA,FILEB,FILEC/N,FILED (Generate listings for FILEA,FILEB,and FILED; see 2.6.2) LOAD FILEA,FILEB,%60000FILEC (Pass origin switch to LOADER; see "Loader-Switches" 2.6.4)	u L R D	See Table 2-4

*Refer to footnote in Table 2-1

Each time a COMPILE, LOAD, EXECUTE, or DEBUG COMMAND is executed, the command with its arguments is "remembered" as a file on the disk. Because of this, the filename last used may be recalled for the next command without specifying the arguments again. (See last paragraph in Section 2.5)

2.6.1 Extended Command Forms

The commands shown in Table 2-5 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 Time-Sharing

Monitor, is more easily accomplished by means of one or more of the extended command forms.

2.6.1.1 The @ File

When there are many program names and switches, they can be put into a file so that 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 of the commands in Tables 2-3 and 2-5.

The "@ file" must appear at any point after the first word in the command. In this construction "file" must be a file-name, 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 themselves may contain the "@ file" construction to a depth of nine levels. If this indirecting process should result in files pointing in a loop, the maximum depth will rapidly be exceeded and an error message will be produced.

The following rules are used 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) The characters CARRIAGE RETURN,LINE-FEED, AND ALTMODE are ignored if the first non-blank character after a sequence of returns, line-feeds, and altmodes is a comma. Otherwise, they are treated either as commas by the COMPILE, LOAD, EXECUTE, and DEBUG commands or as command terminators by all the other commands appearing in Tables 2-3 and 2-5.

c) Blank lines are completely ignored since 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 through the line-feed is ignored.

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

2.6.1.2 The "+" Construction[1]

A single relocatable binary file may be produced from a collection of input source files by means of the "+" construction. For example, a user may wish to compile the parameter file, S.MAC, the switch file, FT50SB.MAC, and the file that is the body of the program, APRSER.MAC. This is specified by the following command:

```
COMPILE S + FT50SB + APRSER
```

The name of the last input file in the string is given to any output (.REL and/or .LST) files (e.g., APRSER. in the foregoing example). The source files in the "+" construction may each con-

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

tain device and extension information and project-programmer numbers.

2.6.1.3 The "=" Construction[1]

Usually the filename of the binary file is the same as that of the source file, with the extension specifying the difference. This can be changed by use of the "=" construction, which allows a filename other than the source filename to be given to the output file. 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 same 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 + PART2
```

2.6.1.4 The "<>" Construction[1]

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
```

But by using the angle brackets, the command becomes

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

The user cannot type

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

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

2.6.2 Compile Switches[1]

The COMPILE, LOAD, EXECUTE, and DEBUG commands may be modified by a number of switches. Each switch is preceded by a slash (/SWITCH) and is terminated by any non-alphanumeric character, usually a space or a comma. An abbreviation may be used as long as 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 A,/MACRO,B,C
COMPILE A /MACRO B,C
COMPILE A,/MACRO B,C
```

2.6.2.1 Compilation Listings

Listing files may be generated by the use of 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 non-listing of programs. These switches may be used as either temporary or permanent switches.

---

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

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

will generate listings of all three programs.

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

will generate a listing only of program A.

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

will generate listings of programs A and C.

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

Unless the /LIST or /CREF is specified, no listing file is generated. The LIST command is used to obtain printer output of regular listing files and the CREF command to obtain printer output of CREF listing files.

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

```
COMPILE /L A
```

produces a listing file "A.LST" (as well as, of course, "A.REL").

2.6.2.2 The "Standard Processor"

The "standard processor" is used to compile or assemble programs which do not have the extensions .MAC, .F4, or .REL. There are a number of switches for setting 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, so the "standard processor" will be used. For these cases the user must be able to control the setting of the "standard processor".

The "standard processor" is FORTRAN IV at the beginning of each command string.

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

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, even though a newer source file may be present. (Useful primarily in LOAD, EXECUTE, DEBUG commands).

These switches may be used as "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.

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

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

NOTE

Programs with .MAC and .F4 extensions are always translated by the extension implied, regardless of the "standard processor."

2.6.2.3 Forced Compilation

The compilation (or assembly) occurs if the source file is at least as recent as the relocatable binary file. If the binary is newer than the source, there is not normally any need to perform the translation.

There are cases, however, where such extra translation may be desirable, as for instance, when one desires a listing of the assembly. To force such an assembly, the switch "COMPILE" is provided, again in both temporary and permanent form. For example:

COMPILE /CREF / COMPILE A, B, C

will create cross-reference listing files A.LST, B.LST, and C.LST, even though current .REL files may exist. In fact, the binary files will also be recreated.

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

2.6.2.4 Library Searches

The LOADER normally performs a library search of the FORTRAN library. Sometimes it is necessary to search other files as libraries. To do this, the compile-switches "LIBRARY" and (its complement) "NOSEARCH" are provided.

These switches may be used as either "permanent" or "temporary".

For example, suppose a special library file named SPCLIB.REL were kept on device SYS at a particular installation. Then to compile and load a user program, library search the

special library, and then 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.

2.6.2.5 Loader Maps

Loader maps are produced during the loading process by the compile-switch "MAP". When this switch is encountered, a loader map is requested from the Loader. The map will be written with filename MAP.MAP, in the user's disk area.

This compile-switch is the one exception to the "permanent compile-switch" rule, in that it causes only one map to be output, even though it may appear as a permanent switch.

2.6.3 Processor Switches [1]

Occasionally it is necessary to pass switches to the assembler or compiler. Recall that 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. If the user wishes to add switches to those files, he must do so as follows:

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.

---

[1] Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

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 eleminate 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.

2.6.4 Loader Switches[1]

In unusually complex loading processes, it may be necessary to pass loader-switches to the LOADER to direct its

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

operation. These are passed via the COMPILE, LOAD, EXECUTE, and DEBUG commands. These switches must be passed to the LOADER (not to the compiler or assembler). This is accomplished by the % character. The % has the same meaning as that of the / in the Loader's command string. Also, like the /, it 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, %60000C,D
```

The most commonly used switches are:

%S Load with symbols

%nO Set program origin to n

%F Cause early search of FORTRAN library

%P Prevent FORTRAN library search

2.6.5 Temporary Files

The COMPIL CUSP deciphers the commands found in Tables 2-3 and 2-5 and constructs new commands for the CUSPS that were referenced. These new commands are written as temporary files 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.

2.6.5.1 001SVC.TMP

This file contains the most recent COMPILE, LOAD, EXECUTE, or DEBUG command which included arguments. It is used to remember those arguments. See section 2.6.

2.6.5.2 001EDS.TMP

This file contains the most recent EDIT, CREATE, TECO, or MAKE command which included an argument. It is used to remember that argument. See section 2.5

2.6.5.3 001MAC.TMP

This file contains commands to MACRO. It is written by COMPIL, and ready 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, etc.).

2.6.5.4 001FOR.TMP

This file corresponds exactly to the one described in the preceding section, except that it is read by the FORTRAN IV compiler, F40.

2.6.5.5 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.

2.6.5.6 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 which may not yet have been listed.

2.6.5.7 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

Sfile.ext (ALTMODE)

For the commands TECO or EDIT, it contains the command

Sfile.ext (RETURN) (LINEFEED)

2.7 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.

Table 2-6

Monitor Commands to Call, Load, and Control Programs

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
RUN dev file.ext [proj,prog] core	RU	To load a core image from a retrievable storage device and start it at the location specified within the file (JOBSA). If the program has two segments, both the low and high segments will be 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). dev — The logical or physical name of the device containing the core image. file.ext — The name of the file containing the core image; if .ext is omitted, it is assumed to be SHR + LOW, HGH + LOW, or SAV. See SAVE, SSAVE. [proj.prog] — Project-programmer number; required only if core image file is located in a disk area other than the user's. core — Amount of core to be assigned if different from minimum core needed to load the the program or from the core argument of the SAVE command which saved the file. Since previous core is returned, MTA must have this argument because there is no directory to tell how much core for low segment.	u L J	dev: NOT AVAILABLE The device has been assigned to another job. NO SUCH DEVICE The device does not exist. nK OF CORE NEEDED There is insufficient free core to load the file. NOT A DUMP FILE The file is not a core image file. TRANSMISSION ERROR A parity or device error occurred during loading.
R file.ext core	R	Same as RUN SYS: file.ext core. The R command is the usual way to run a CUSP that does not have a direct Monitor command to run it.	u L J R	Same as RUN
GET dev file.ext [proj,prog] core	G	Same as RUN command except that Monitor types out JOB SETUP and does not start execution.	m L J A	Same as RUN

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
START adr	ST	Begins execution of a program previously loaded with the GET command. adr — The address at which execution is to begin if other than the location specified within the file (JOBSA). If adr is not specified, the starting address comes from JOBSA.	u L J C A	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 END statement of source program.
HALT (↑C)	↑C	Places the console in Monitor mode and transmits a HALT command to the Monitor Command Interpreter. Stops the job and stores the program counter in the job data area (JOBPC).	m	
CONT	CON	Starts the program at the saved program counter address stored in JOBPC by a HALT command (↑C) or a HALT instruction.	u L J C	CAN'T CONTINUE The job was halted due to a Monitor-detected error and can't be continued.
DDT	DD	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	u L J C	NO START ADR DDT starting address was 0 (JOBDDT).
REENTER	REE	Similar to the DDT command. Copies saved program counter value from JOBPC into JOBOPC and starts program at an alternate entry point specified in JOBREN (must be set by the user or his program).	u L J C	NO START ADR REENTER starting address was 0 (JOBREN).
E adr	E	Examines a core location in the user's area (high or low segment). adr — If this argument is specified, the contents of the location are typed out in halfword octal mode. Adr is required the first time the E or D command is used. If adr is not specified, the contents of the location following the previously specified E adr or the location of the previous D adr are typed out.	m L J C	OUT OF BOUNDS The specified adr is not in the user's core area, or the user does not have read privileges to file which initialized the high segment.

Command	Abbreviation	Explanation	Characteristics	
D lh rh adr	D	Deposits information in the user's core area (high or low segment). lh — The octal value to be deposited in the left half of the location. rh — The octal value to be deposited in the right half of the location. adr — The address of the location into which the information is to be deposited. 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.	m L J C	OUT OF BOUNDS The specified adr is not in the user's core area, or high segment is write protected and user does not have write privileges to file which initialized the high segment.
SAVE dev file.ext core	SA	Writes out a core image of the user's core area on the specified device. Saves any user program (reentrant, one segment non-reentrant, or two segment non-reentrant) as one or two files. Later when the program is loaded by a GET, R, or RUN command, it will be non-reentrant. 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. dev — The device on which the core image file is to be written. file.ext — The name to be assigned to the core image file. 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 — Amount of core in which the program is to be run. This value is stored in the job's core area (JOBCOR) and is used by the RUN and GET commands. Specified as number of 1K blocks.	m L J C A	n 1K BLOCKS OF CORE NEEDED The user's current core allocation is less than the contents of JOBFF. DEVICE NOT AVAILABLE Device dev is assigned to another user. TRANSMISSION ERROR An error was detected while reading or writing the core image file. DIRECTORY FULL The directory of device dev is full; no more files can be added. JOB SAVED The output is completed.

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
		If core is omitted, only the number of blocks required by the core image area (as explained above) is assumed.		
SSAVE dev file.ext core	SSA	Same as SAVE except that the high segment 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 CUSP's without having to know which are sharable.	m L J A C	

*Refer to footnote in Table 2-1

2.7.1 Additional Information on SAVE and SSAVE

Low segment files will be zero compressed on all devices (DTA,MTA,DSK), but high segment files will not since the high segment may be shared at the time of the command. Saved files are ordinary binary files and can be copies using the /B switch in PIP.

In order 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.

It is possible for most programs to be written so that only the high segment contains non-zero data. This will also save file space and I/O time with the GET command. SAVE will write the high segment (.HGH) only. The LOADER will indicate 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.

There are a number of locations in the Job Data area which need to be initialized on a GET, even though 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. These 10 locations are referred to as the Vestigial Job Data area. 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. Both commands also delete a file 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 do a GET, R, or RUN command. These commands will 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 will always enable the hardware user-mode write protect to prevent the user program from storing into the segment inadvertently.

To debug a reentrant CUSP which is in the system directory, the user should make a private, non-sharable copy, rather than modifying the shared version and possibly causing

harm to other users. To make a private, non-sharable copy, the following commands are used.

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

The Monitor will keep the shared and the non-shared versions separate from each other. A sharable program may be superseded into the directory by the SSAVE command. The Monitor will clear the high segment in its table of storable segments in use but will 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 will 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 will not modify the Vestigial Job Data area unless the user has write privileges to the file which 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, since the GET command reads the low file after the high file. The real Job Data area locations are set from the low file.

2.8 BACKGROUND JOB CONTROL COMMANDS

A job is a background, or detached, job if it is not under control of a user console. Any console can initiate any number of background jobs. I/O to the console while a job is running in a background mode causes the job to stop until a console is attached.

Table 2-7

Monitor Commands to Control Background Jobs

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
PJOB	PJ	Monitor responds by typing the job number to which the user's console is attached. 10/40 System - If the console is not attached to a job, Monitor assigns a job number and types the job number and a line identifying the Monitor version. 10/50 System - If the console is not attached to a job, Monitor responds with LOGIN PLEASE.	m L J	
CSTART CCONT	CS CC	Identical to the START and CONT commands, respectively, except that the console is left in the Monitor mode. To Use: 1. Begin the program with the console in user mode. 2. Type control information to the program, then type ↑C to halt job with console in Monitor mode. 3. Type CCONT to allow job to continue running and leave console in Monitor mode. 4. Further Monitor commands can now be entered from the console. Caution: These commands should not be used when the user program (which is continuing to run) is also requesting input from the console.	m L J C	Same as START and CONT.
DETACH	DET	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 background job.	d L	

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
ATTACH job	AT	Connects a console to a background job. job — The job number of the job to which the console is to be attached. [proj,prog] — The project-programmer number of the originator of the desired job. May be omitted if same as job to which console is currently attached. The operator (device OPR) may always attach to a job even though another console is attached, provided he specifies the proper [proj,prog].	m	If an error message occurs, the console remains attached to its current job. TTYn ALREADY ATTACHED Job number typed is erroneous and is attached to another console, or another user is attached to the job. NOT A JOB The job number is not assigned to any currently running job. CAN'T ATTACH TO JOB The project-programmer number entered is not that of the originator of the desired job.

*Refer to footnote in Table 2-1

2.9 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 other users.

Table 2-8

Monitor Command to Terminate Jobs

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
KJOB	K	In Multiprogramming Systems: Stops all allocated 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: All of the above procedures. In addition, if user has any files, responds with: CONFIRM:	m A	

Command	Abbreviation	Explanation	Characteristics* Monitor Messages
		To which the user may type ↑C to abort log-out; or type one of the following: K) to kill job and delete all unprotected files; L) to list his disk directory; I) to individually save and delete files as follows: After each file name is listed, type: P to save and protect, S to save without protecting, or) to delete. Files with extensions.LST and .TMP will be deleted automatically.	

*Refer to footnote in Table 2-1

2.10 SYSTEM TIMING COMMANDS

All system times are kept in increments of one-sixtieth or one-fiftieth of a second, depending on the power frequency of the country in which the PDP-10 is installed.

Table 2-9

Monitor Commands for System Timing

Command	Abbreviation	Explanation	Characteristics* Monitor Messages
DAYTIME	DA	Types the date followed by the time of day. Time is typed in the format. hh:mm where hh = hours mm = minutes	m
TIME job	TI	Types out 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 format hh:mm:ss.hh where hh = hours mm = minutes ss.hh = seconds to nearest hundredth.	m

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
		Interrupt level and job scheduling times are charged to the user who was running when the interrupt or rescheduling occurred. 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. If job = 0, an approximation of the time spent core shuffling (SHFL) is printed, followed by the amount of time spent clearing core (ZCOR), the running time of the null job (NULL), the time during which one or more jobs wanted to run but were swapped out or in the process of being swapped out (LOST), and the total time system has been up (UP).		

*Refer to footnote in Table 2-1

2.11 SYSTEM ADMINISTRATION COMMANDS

The SYSTAT command permits a user to learn how heavily the system is loaded and the status of devices in the sharable device pool. The other commands in this section are restricted to system administrators only.

Table 2-10

Monitor Commands for System Administration

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
SCHEDULE n	SCH	Changes the scheduled use of the system, depending on n. This command is legal only from the operator's console. n is stored in RH of STATES word in COMMON: 0 = regular time sharing. 1 = no further LOGINS allowed. 2 = no further LOGINS from remote TTY's. If n is omitted, the current value of n is printed.	m	
SYSTAT	SYS	Types out status of the system: system name, time of day, date, uptime, percent null time. Status of each job: job number, project-programmer number (**** if detached), TTY number, program name being run, size of low segment, state (RN = runable, TT = TTY input wait, C = Monitor command mode) and run time. Status of high segments being used: name, directory name, size, number of users in core or on disk. Status of each assigned device: name, job number, how assigned (AS = ASSIGN command, INIT = INIT UUO).		
ASSIGN SYS:dev		To change the systems device to device "dev." The user must be logged in under either [1,1] or [1,2].	m L J	
DETACH dev	DET	To assign the device "dev" to JOB 0, thus making it unavailable. The user must be logged in under [1,1].	m L J	
ATTACH dev	AT	To return a detached device to the Monitor pool of available devices. The user must be logged in under [1,1].	m L J	
CTEST		This command is used by system programmers to test extensions made to the COMPIL CUSP.	L R	

*Refer to footnote in Table 2-1

2.12 MONITOR DIAGNOSTIC MESSAGES

Once a user program has been started, a number of error conditions may arise which cause the job to revert to monitor mode. The error messages typed, and the meanings for each are summarized in the following table.

Table 2-11

Time-Sharing Monitor Diagnostic Messages

Message	Meaning
The typein is typed back followed by ?)	The Monitor command decoder has encountered an incorrect character, such as a letter in a numeric argument. The incorrect character appears immediately before the ?. Example: User types in: CORE ABC Monitor responds: CORE A ?)
ADDRESS CHECK FOR DEVICE dev AT USER adr	Monitor has checked a user address and has found it to be too large (>C(JOBREL)) or too small (<JOBPFI). Some user addresses can be the user's accumulators while others cannot. One of the following addresses may be wrong: buffer buffer header dump mode command list data specified by dump mode command list insufficient core available for setting up Monitor-generated buffers.
BAD DIRECTORY FOR DEVICE DTAn; UUO AT USER adr	The DECtape directory is not in proper format or had a parity error when read. Many times this error occurs when an attempt is made to use a virgin tape.
DEVICE dev OK?	Device dev is temporarily in an inoperable state, such as LPT off-line. The user should correct the obvious condition and then type a CONT command.

Table 2-11 (Cont)

Time-Sharing Monitor Diagnostic Messages

Message	Meaning
ERROR IN JOB n	A fatal error has occurred in the user's job (or in Monitor while servicing the job). This typeout is normally followed by a 1-line description of the error.
HALT AT USER adr	The user program has executed a halt instruction at loc. adr. Typing CONT will resume execution at the effective address of the halt.
HUNG DEVICE dev; UUO AT USER adr	A device has not generated an interrupt for a timed period and, therefore, is in need of attention.
ILLEGAL DATA MODE FOR DEVICE dev AT USER adr	The data mode specified for a device in the user's program is illegal.
ILLEGAL UUO AT USER adr	An illegal UUO has been executed at user location adr.
ILL INST. AT USER adr	An illegal operation code has been encountered in the user's program.
ILL MEM REF AT USER adr	An illegal memory reference has been made by the user program at adr or adr+1.
INCORRECT RETRIEVAL INFORMATION: UUO AT USER adr	The retrieval pointers for a file are not in the correct format; the file is unreadable. If this typeout occurs, the user should report it on a Software Trouble Report.
INPUT DEVICE dev CANNOT DO OUTPUT; UUO AT USER adr	An illegal OUTPUT UUO has been executed at user location adr.
I/O TO UNASSIGNED CHANNEL AT USER adr	No OPEN or INIT was performed on the channel.
LOOKUP AND ENTER HAVE DIFFERENT NAMES: UUO AT USER adr	An attempt has been made to read and write a file on the disk. However, the LOOKUP and ENTER UUO's have specified different names on the same user channel. This message does not indicate a DECtape error.

Table 2-11 (Cont)

Time-Sharing Monitor Diagnostic Messages

Message	Meaning
MASS STORAGE DEVICE FULL; UUO AT USER adr	The storage disk is full. Users must delete unneeded files before the system can proceed.
NON-RECOVERABLE DISC READ ERROR; UUO AT USER adr NON-RECOVERABLE DISC WRITE ERROR; UUO AT USER adr	Monitor has encountered an error while reading or writing a critical block in the disk file structure (e.g., the MFD or the SAT table). If this condition persists, the disk must be reloaded using Fail-safe after the standard location for the MFD and SAT table has been changed using the Monitor once-only dialogue.
NOT ENOUGH FREE CORE IN MONITOR: UUO AT USER adr	The Monitor has run out of free core for assigning disk data blocks and Monitor buffers. If this type-out occurs, the user should report it on a Software Trouble Report.
NOT FOUND	The program file requested cannot be found on the systems device (or on the specified device).
OUTPUT DEVICE dev CANNOT DO INPUT; UUO AT USER adr	An illegal INPUT UUO has been executed at user location adr.
PC EXCEEDS MEMORY BOUND AT USER adr	An illegal transfer has been made by the user program to user location adr.
SWAP READ ERROR	A consistent checksum error has been encountered when checksumming locations JOBDAC through JOBDAC+74 of the Job Data area during swapping.

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 and certain operations are illegal. Since every user has an assigned area in core, 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. See 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 (see SETUWP UUO).

The Time-Sharing Monitor defines the size and position of a user's area by specifying protection and relocation addresses for the low and high segments. The protection address is the maximum relative address the user can reference. The relocation address is the absolute core address of the first location in the segment, as

seen by the Monitor and the hardware. The Monitor defines these addresses by loading four 8-bit registers (two 8-bit registers in PDP-10's without the KT10A option), each of which corresponds to the left eight bits of an 18-bit PDP-10 address. Thus, segments always contain an even 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 relative address is 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). See APRENB UUO, 4.3.3.1.

Some systems have only the low pair of protection and relocation registers. In this case, 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 0-17 are not relocated, all users having access to the accumulators. Therefore, relative locations 0-17 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. See Book 1 for a more complete description of the relocation and protection hardware.

3.2 USER'S CORE STORAGE

A user's core storage consists of blocks of memory whose sizes 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 illustrated below, 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> 400000), the high segment starts at the first location after the low segment (at P_L + 2000). The high segment is limited to 128 K.

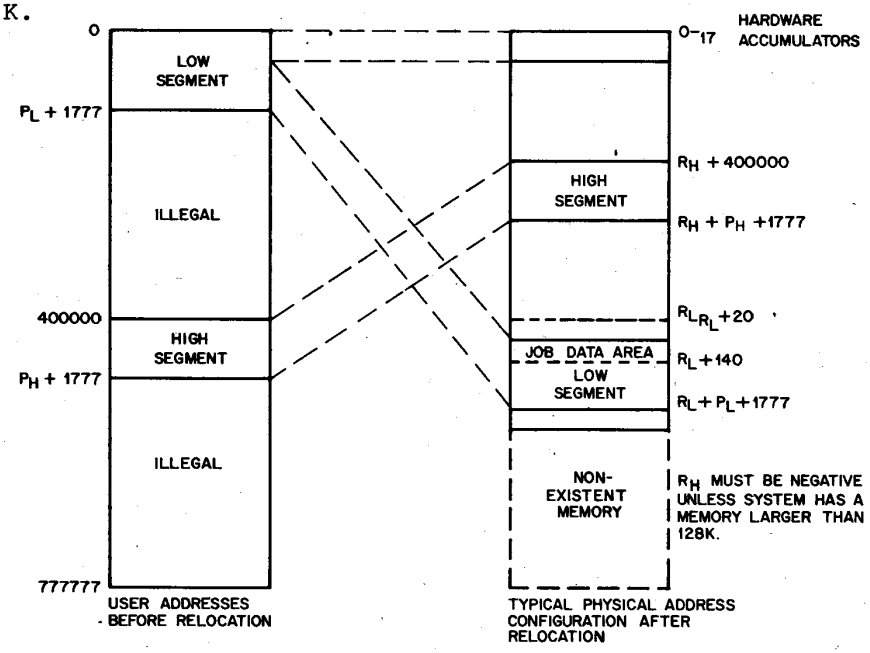


Figure 3-1

User's Core Area

There are two methods available to the user for loading his core area. The simplest way is to load a core image stored on a retrievable device (see 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 (see SAVE, Chapter 2).

3.2.1 Job Data Area

The Job Data area provides storage for specific information of interest to both the Monitor and the user. The first 140 (octal) locations of the user's core area always are allocated to the Job Data area. Locations in this area have been given mnemonic assignments whose first three characters are JOB. Therefore, all mnemonics in this manual with a JOB prefix refer to locations in the Job Data 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 UUO's (001 through 037). Op code and effective address are stored here.
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 at the present. 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).

Table 3-1 (Cont)

Job Data Area Locations

(for user-program reference)

Name	Octal Location	Description
JOBBLT	45	Three consecutive locations where 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	Contains the starting address of DDT. If contents are 0, DDT has not been loaded.
JOBCN6	106	Six temporary locations used by CHAIN (FORTRAN Runtime Routine) after it releases all I/O channels. JOBCN6 is defined to be in JOBJDA.
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.
JOBSYM	116	Contains a pointer to the symbol table created by Linking Loader. Left half: Negative count of the length of the symbol table. Right half: Lowest register used.
JOBUSY	117	Contains a pointer to the undefined symbol table created by Linking Loader. Not yet 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.

Table 3-1 (Cont)

Job Data Area Locations

(for user-program reference)

Name	Octal Location	Description
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 at present. Right half: REENTER starting address. Set by user or by Linking 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-enabled 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 user's program counter are stored here by Monitor upon execution of a DDT, REENTER, START, or CSTART command.
JOBCHN	131	Left half: 0 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.
JOBVER	137	Left half: Zero or the programmer number of the programmer who made last identification to the program. Right half: Program version number in octal. The number is never converted to decimal. After a GET, R, or RUN command, a E command can be used to find the version number. (Digital always distributes CUSPs with the left half = 0, so customers making modifications to CUSPs should change only the left

Table 3-1 (Cont)

Job Data Area Locations

(for user-program reference)

Name	Octal Location	Description
JOBVER (Cont)	137	half. The right half will remain as a record of the Digital 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 the present time. User programs should not refer to any locations not listed above since such locations are subject to change without notice.

Some locations in the Job Data area, such as JOBSA and JOBDDT, are set by the user's program for use by the Monitor. Others, such as JOBREL, are set by the Monitor for use by the user's program. In particular, the right half of JOBREL contains the highest legal address set by the Monitor whenever the user's core allocation changes.

JOBDAT exists in binary form in the Systems Library for loading with user programs that refer to Job Data area locations symbolically. User programs must reference locations by means of the assigned mnemonics, which are declared as EXTERNAL references to the assembler. JOBDAT is loaded automatically, if needed, during the Loader's library search for undefined global references, and the values are assigned to the mnemonics.

3.2.2 Loading Relocatable Binary Files

The relocatable binary loader (LOADER, V.47) which resides in the system file is started by the command

R LOADER core

where core is an optional argument. (See Book 5 for a description of the Loader command string.)

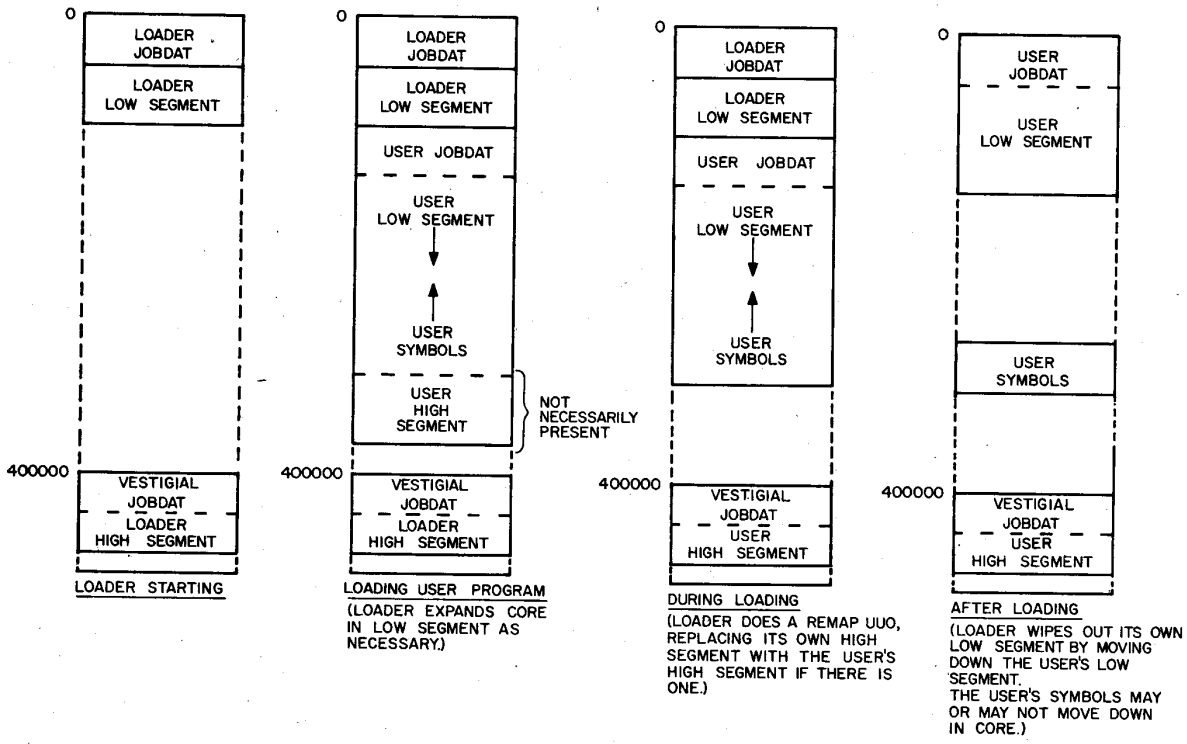


Figure 3-2

Loading User Core Area

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.

Since the Loader is reentrant, 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 be duplicated 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), and

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

3.2.2.1 The H Switch

A program written to be reentrant can be loaded into one segment instead of two by use of the H switch (/H). This switch is used only when a two-segment program is to be loaded into one 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 will check to see 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 one. 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 can use the switch NNNNNNH. This switch changes the origin of the high segment from its initial setting of 400000 to NNNNNN where NNNNNN is larger. If NNNNNN is missing, the Loader loads everything into the low segment.

Since it is not known before load time whether a reentrant program is not going into the high segment, the code executed (including the Monitor UUO's) is the same for either case.

3.2.2.2 The HISEG Pseudo-Op

After loading, a relocatable subprogram assembled by MACRO is either put entirely in the user low segment or entirely in the user high segment. To indicate that a subprogram is to be loaded into the high segment, the HISEG pseudo-op is used. It can appear anywhere in the program although it is best to place it at the beginning since a reader of the program 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 47 loads programs in any order. In earlier versions of the Loader, programs for the low segment must be loaded before any programs for the high segment.

3.2.2.3 The Vestigial Job Data Area

There are a few "constant" data in the Job Data area

which may be loaded by a two-segment, one-file program without using instructions on a GET command (JOB41, JOBREN, JOBVER) and there are a number of locations which the Monitor loads 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. 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.

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 it by means of 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 date.

d) Uses REMAP UUO to take the top part of the low segment

which 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 means of 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 zero. A new switch, /K, has been implemented for use with DDT. This switch moves core back to the absolute maximum needed. A /nK moves core back to nK.[1]

---

[1] In the latest version of the Loader, V.50, the /D is used to imply /B/K.

CHAPTER 4
USER PROGRAMMING

The PDP-10 central processor operates in one of three modes: executive mode, user I/O mode, or user mode. The Monitor operates in executive mode, which is characterized by the lack of memory protection and relocation (see Chapter 3) and by the normal execution of all defined operation codes. The user I/O mode is a special mode, wherein memory protection and relocation are in effect, as well as the normal execution of all defined operation codes. (This mode is not used by the Monitor, and is not normally available (see TRPSET) to the time-sharing user.) User programs are run in user mode in order to guarantee the integrity of both the Monitor and each user program.

4.1 USER MODE

The user mode of the central processor is characterized by the following features:

a) Automatic memory protection and relocation (see Chapter 3).

b) Trap to absolute location 40 on any of the following:

1. Operation codes 040 through 077 and operation code 000.
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 on execution of operation codes 001 through 037.

Since user programs run in user mode, the Monitor must

perform all input/output operations for the user, as well as any other user-requested operations that are not available in user mode. The purpose of this chapter is to describe the services the Monitor makes available to user mode programs and how a user program obtains such services.

4.2 PROGRAMMED OPERATORS (UUO's)

Operation codes 000 through 077 in the PDP-10 are programmed operators (sometimes referred to as UUO's- Unimplemented User Operators since 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 UUO's)

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 UUO's)

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 input/output 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 thru 077 and their mnemonics. Most of this chapter is a detailed description of their operation.

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 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 perform the lookup himself 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.

The customer is allowed to add his own CALL and CALLI calls to the Monitor. A negative CALLI effective address

(starting with -2) should be used to specify such customer added operations.

4.2.2.2 Restriction on Monitor UUO's in Re-Entrant User Programs

There are a number of restrictions on UUO's which involve a high segment. These restrictions are to prevent naive or malicious users from clobbering other users while sharing segments and to minimize Monitor overhead in handling two-segment programs. The basic rules are as follows.

a) All UUO's 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 UUO's cannot be in the high segment. If any one of these rules is violated, an address check error message is given (see 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, even though 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.

4.2.3 Operation Codes 100-127 (Unimplemented Op Codes)

Op code 100-UJEN	Dismisses realtime interrupt from user mode (see 4.3.6.2).
Op codes 101-127	Monitor prints ILL INST AT USER n and stops job.

4.2.4 Illegal Operation Codes

The eight input/output instructions (DATAI, etc.) 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 PROGRAM CONTROL

4.3.1 Starting

All program starting is accomplished by the Monitor commands RUN, START, CSTART, CONT, CCONT, DDT, and REENTER (see Chapter 2). The starting address is either an argument of the command or stored in the user's job data area (see Chapter 3).

4.3.1.1 CALL AC, [SIXBIT/SETDDT/] or CALLI AC,2 - These cause the contents of the AC to replace the DDT starting address, which is stored in the protected job data area location, JOBDDT. This starting address is used by the Monitor command, DDT (see 3.2.2.4).

4.3.2 Stopping

Any one 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 (see Chapter 2);

b) A Monitor detected error; or

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 n

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

However, the CONT and CCONT commands are still valid and, if typed, will continue the program at the effective address of the HALT instruction. HALT is not the instruction used to terminate a program (see EXIT, section 4.3.2.3). HALT is useful for catching "impossible" error conditions.

Table 4-1

Monitor Operation Codes

Op Code	Mnemonic	Function
040	CALL	Operation code extension (See 4.2.2.1)
041	INIT	Initialize I/O device (See 4.4.2.2)
042		No operation — Reserved for installation-dependent calls
043		No operation
044		No operation
045		No operation
046		No operation
047	CALLI	Operation code extension (See 4.2.2.1)
050	OPEN	Open file (See 4.4.2.2)
051	TTCALL	Special Teletype Operations (See 5.1.3)
052		No operation — Reserved for future expansion by Digital
053		No operation
054		No operation
055	RENAME	Rename or delete a file (See 4.4.2.5)
056	IN	Input and Skip on error of EOF (See 4.4.3)
057	OUT	Output and skip on error of EOF (See 4.4.3)
060	SETSTS	Set file status (See 4.4.4)
061	STATO	Skip on file status one (See 4.4.4)
062	STATUS or GETSTS	Read file status (See 4.4.4)
063	STATZ	Skip on file status zero (See 4.4.4)
064	INBUF	Set up input buffer ring (See 4.4.2.3)
065	OUTBUF	Set up output buffer ring (See 4.4.2.3)
066	INPUT	Read (See 4.4.3)
067	OUTPUT	Write (See 4.4.3)
070	CLOSE	Close file (See 4.4.5)

Table 4-1 (Cont)

Monitor Operation Codes

Op Code	Mnemonic	Function
071	RELEAS	Release device (See 4.4.7)
072	MTAPE	Position tape (See 5.8.2 and 5.7.5)
073	UGETF	Get next free block number (See 5.7.5)
074	USETI	Set next input block number (See 5.7.5)
075	USETO	Set next output block number (See 5.7.5)
076	LOOKUP	Select file (See 4.4.2.4)
077	ENTER	Create file (See 4.4.2.4)
100	UJEN	Dismiss real-time interrupt (See 4.3.6.2)

Table 4-2

CALL and CALLI Monitor Operations

CALLI AC, x	CALL AC, [SIXBIT/y/]	Function
x = -2,.. ..,-n	y = Customer defined	Reserved for definition by each customer installation.
-1	LIGHTS	Displays AC in console lights
0	RESET	Reset I/O devices (See 4.4.2.1)
1	DDTIN	DDT mode console input (See 5.1.2)
2	SETDDT	Set protected DDT starting address (See 4.3.1.1)
3	DDTOUT	DDT mode console output (See 5.1.2)
4	DEVCHR	Get device characteristics (See 5.12)
5	(DDTGT)	No operation
6	(GETCHR)	Same as DEVCHR(4)

Table 4-2 (Cont)

CALL and CALLI Monitor Operations

CALLI AC, x	CALL AC, [SIXBIT/y/]	Function
x = 7	y = (DDTRL)	No operation
10	WAIT	Wait until device inactive (See 4.4.6)
11	CORE	Allocate core (See 4.5)
12	EXIT	Stop job, may release devices (See 4.3.2.3)
13	UTPCLR	Clear directory (See Table 5-3)
14	DATE	Return date (See 4.3.4.1)
15	LOGIN	Special operation for LOGIN (See 4.3.5.3)
16	APRENB	Enable central processor traps (See 4.3.3.1)
17	LOGOUT	Kill job (See 4.3.2.4)
20	SWITCH	Read processor console switches (See 4.3.6.3)
21	REASSI	Reassign device (See 2.4.4)
22	TIMER	Read clock in ticks (See 4.3.4.2)
23	MSTIME	Read clock in milliseconds (See 4.3.4.3)
24	GETPPN	Read project-programmer pair (See 4.3.5.2)
25	TRPSET	Set trap for user I/O mode (See 4.3.6.1)
26	TRPJEN	Illegal UUO
27	RUNTIM	Return job running time (See 4.3.4.4)
30	PJOB	Return job number (See 4.3.5.1)
31	SLEEP	Stop job for specified time (See 4.3.4.5)
32	(SETPOV)	Set pushdown overflow trap (this command has been superseded by APRENB (16)

Table 4-2 (Cont)

CALL and CALLI Monitor Operations

CALLI AC, x	CALL AC, [SIXBIT/y/]	Function
x = 33	y = PEEK	Return specified Monitor location (See 4.3.5.4)
34	GETLIN	Return physical name of attached Teletype console. (See 4.3.5.5)
35	RUN	Call new program (both high and low segments) (See 4.3.7.2)
36	SETUWP	Set user's write protect for high segment (See 4.5.1)
37	REMAP	Remap top of low segment into high segment (See 4.3.7.1)
40	GETSEG	Replace high segment only (See 4.3.7.3)
41	GETTAB	Examine contents of specified Monitor location (See 4.3.5.6)
42	SPY	Make physical core be high segment for efficient looking at Monitor (See 4.3.7.4)
43	SETNAM	Set program name (See 4.3.6.4)

NOTE

1) Other CALLI UUO's will be implemented from time to time and will be documented in Software Manual Updates and in revised editions of this manual

2) Execution of a CALLI UUO with an address higher than the last implemented operator will return control to the next location in the user program. New implemented operators will cause a skip return. In this way, the user program will know if the UUO has been implemented.

4.3.2.3 CALL AC, [SIXBIT/EXIT/] or CALLI AC, 12 - When AC is zero, all input/output devices are RELEASed (see Section 4.4.7) and the job is stopped. The CR-LF operation is performed and

```
EXIT
↑C
.
```

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 non-zero, the job is stopped but devices are not released. Instead of printing EXIT and ↑C, only the CR-LF operation is performed and a period is printed on the user's console. The CONT and CCONT commands may be used to continue the program.

4.3.2.4 CALL [SIXBIT/LOGOUT/] or CALLI 17 - All input/output devices are RELEASed (see Section 4.4.7), 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. It is only for use by the LOGOUT CUSP. If a user program executes a LOGOUT UUO, the Monitor will treat it like EXIT (See 4.3.2.3).

4.3.3 Trapping

4.3.3.1 CALL AC, [SIXBIT/APRENB/] or CALLI AC, 16 - APR trapping allows a user to handle any and all traps that occur while his job is running on the central processor, including illegal memory references, non-existent memory references, pushdown list overflow, arithmetic overflow, floating point overflow, and clock flag. To enable for trapping a 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
19	200000	pushdown overflow
22	20000	memory protection violation
23	10000	non-existent 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 (see 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 have been cleared. The user program must set up location JOBAPR 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 JRST 2, @ JOBTPC to restore the state of the processor.

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 table above. If the user program produces such an error condition, the Monitor will cause the user job to be stopped and print

```
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 Console-Initiated Traps - Program control can be changed from the user's console by use of the ↑C, START, DDT, and REENTER commands (see Chapter 2).

4.3.4 Timing Control

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 of day 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.

4.3.4.1 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.

4.3.4.2 CALL AC, [SIXBIT/TIMER/] or CALLI AC, 22 - These return the time of day, in clock ticks (jiffies), right-justified in accumulator AC.

4.3.4.3 CALL AC, [SIXBIT/MSTIME/] or CALLI AC, 23 - These return the time of day, in milliseconds right-justified in accumulator AC.

4.3.4.4 CALL AC, [SIXBIT/RUNTIM/] or CALLI AC, 27 - The accumulated running time, in milliseconds, of the job whose number is 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 whose number is in AC does not exist, zero is returned.

4.3.4.5 CALL AC, [SIXBIT/SLEEP/] or CALLI AC, 31 - These stop the job, and continue automatically after an elapsed real time of [c(AC)xclock 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 68 seconds (82 seconds in 50 Hz countries) or one minute.

4.3.5 Identification

4.3.5.1 CALL AC, [SIXBIT/PJOB/] or CALLI AC, 30 - These return the job number right-justified in accumulator AC.

4.3.5.2 CALL AC, [SIXBIT/GETPPN/] or CALLI AC, 24 - These return 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 being run is LOGIN or LOGOUT from the system device, the current project-programmer number is changed to 1,2 so that all files are accessible for reading and writing, and a skip return is given if the old project-programmer number is also logged in on another job.

4.3.5.3 CALL AC, [SIXBIT/LOGIN/] or CALLI AC, 15 - These are not available to user programmers. They are for the exclusive use of the LOGIN CUSP, which uses these operators 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 these UUO's, any devices the UUO's were using are released, and the following is printed

on the user's console

↑C

The console is left in Monitor mode ready to accept the user's first command.

Any other user program that calls these UUO's 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.

4.3.5.4 CALL AC, [SIXBIT/PEEK/] or CALLI AC, 33 - These allow a user program to examine any location in the Monitor. 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
CALL AC, [SIXBIT/PEEK/]         ;OR CALLI AC, 33
```

This call returns with the contents of the Monitor location in AC. It is used by SYSTAT and could be used for on-line Monitor debugging.

4.3.5.5 CALL AC, [SIXBIT/GETLIN/] or CALLI AC, 34 - These return 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.

4.3.5.6 CALL AC, [SIXBIT/GETTAB/] or CALLI AC, 41 - These provide a mechanism for user programs to examine the contents of certain Monitor locations in a way which will not vary from Monitor to Monitor.

The call is:

```
CALL AC, [SIXBIT/GETTAB/]        ;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 negative LH means the current job number. The right half of AC contains a table number from the list of Monitor data tables and parameters set forth below. The entries in these tables are all 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 descriptions of these globals are found in the listing of COMMON.

An error return leaves the AC unchanged. This means that the job number or index number in the left half of AC was too high, or the table number in the right half of AC was too high, or that the user does not have the privilege of accessing that table. A skip 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 these UUO's.

The list of tables and their entries is as follows, with a brief description of each.

Table Numbers (RH of AC)

```
00 - JBTSTS (job status word)
             Index by job or segment number
01 - JBTADR (job relocation and protection)
             Index by job or segment number
```

Table Numbers (RH of AC) (Cont)

```
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)
             Index by job number
06 - JBTPRV (privilege bits)
             Index by job number
07 - JBTSWP (job's swapping parameters)
             Index by job or segment number
10 - TTYTAB (Teletype to job translation)
             Index by line number
11 - CNFTBL (configuration table)
             Index by item number, see below
12 - NSWTBL (non-swapping 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
15 - ODPTBL (once-only disk parameters)
             Index by item number, see below
```

Entries in 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 the system device (SIXBIT)
10	TIME	Time of day in jiffies
11	THSDAT	Today's date (12-bit format)
12	SYSSIZ	Highest location in the Monitor + 1
13	DEVOPR	Name of the 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 (FTTTYSER)
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 Set by the privileged operator command, SCHEDULE: Bit 34=1 Means no remote LOGINs Bit 35=1 Means no more LOGINs
20	SERIAL	Serial number of PDP-10 processor Set by MONGEN dialog

Entries in ODPTBL (once only disk parameters)

Item	Location	Use
0	SWPHGH	Highest logical block # in the swapping space
1	K4SWAP	K of disk words set aside for swapping
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 NSWTBL (non-swapping data)

Item	Location	Use
0 - 7	CORTAB CORTAB+7	Map of physical core 1 bit for each K of core
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 no. shuffler has stopped
14	HOLEF	Abs. adr. of job above lowest hole, 0 if no job
15	UPTIME	Time system has been up in jiffies
16	SHFWRD	Tot. no. 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 no. of words cleared by CLRCOR
22	LSTWRD	Total no. of clock ticks when null job ran and other jobs wanted to but couldn't because: 1. Swapped out or on way in or out 2. Monitor waiting for I/O to stop so can shuffle or swap 3. Job being swapped out because expanding core

Entries is SWPTBL (swapping data)

Item	Location	Use
0	BIGHOL	No. of K in biggest hole in core
1	FINISH	+Job no. of job being swapped out -Job no. 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 no. of K of swapping space)
5	SWPERC	LH=no. of swap read or write errors RH=error bits (bits 18-21 same as status bits) +no. of K discarded

4.3.6 Direct User I/O

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. This 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.

4.3.6.1 CALL AC, [SIXBIT/TRPSET/] or CALLI AC, 25 - These are privileged UUO's which may or may not stop time-sharing (stop jobs from being scheduled) and allow the user program to gain control of the interrupt locations. If the user is not job 1, an error return to the next location after the CALL will always be given and the user will remain in user mode. Time-Sharing will be turned back on. If the user is job 1, the central processor is placed in user I/O mode. Under job 1, if AC contains zero, time-sharing is turned back 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 these UUO's. If the LH of AC is not within the range 40 through 57, an error return will be given just as if the user was not job 1.

The call is:

```
        MOVE AC, XWD N, ADR
        CALL AC, [SIXBIT/TRPSET/]
        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 will add the job's 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.

```
        MOVEI AC, PNTR
        HRRM  AC, ADR
        MOVE  AC, XWD N, ADR
        CALL  AC, [SIXBIT/TRPSET/]
        error return
        normal return
        .
        .
        .
ADR:    BLKI DEV,PNTR       ;Block in PNTR to be stored
                            ;in interrupt location
PNTR:   IOWD LEN,BUFFER
```

This UUO is a temporary expedient until some real-time UUO's are implemented which will not stop time sharing and which cannot crash the system.

4.3.6.2 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 can dismiss the interrupt. If the interrupt came from executive mode, however, this operator must be used to dismiss the interrupt. The program must restore all accumulators,and execute UJEN U where user location U contains the program counter as stored by a JSR instruction when the interrupt occurred.

4.3.6.3 CALL AC, [SIXBIT/SWITCH/] or CALLI AC, 20 - These return the contents of the central processor data switches in AC. Caution must be exercised in using the data switches since they are not an allocated resource and are always available to all users.

4.3.6.4 CALL AC, [SIXBIT/SETNAM/] or CALLI AC, 43 - These are 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 (See JBTPRG table under GETTAB UUO 4.3.5.6).

4.3.7 Segment Handling

4.3.7.1 CALL AC, [SIXBIT/REMAP/] or CALLI AC, 37 - These take the top part of a low segment and remap 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
CALL AC, [SIXBIT/REMAP/]     ;or CALLI AC, 37
error return
normal return
```

The amount remapped must be a multiple of 1K decimal words. To insure this, the Monitor will perform 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 (see Job Data area, Chapter 3) 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 more core and moving the data from the low to the high segment with a BLT instruction. The GET command also uses this UUO to do I/O into the low segment instead of the high segment.

4.3.7.2 CALL AC, [SIXBIT/RUN/] or CALLI AC, 35 - These have 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 arg. block
CALL  AC, [SIXBIT/RUN/] 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 as-
                              sumed 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
       assignment             address to be assigned
                              to low segment.
                              LH is ignored rather
                              than setting high
                              segment.
```

Usually a user program will specify only the first two words and set the others to zero. 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.

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 1,1. 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. ↑C followed by the START command will restart the program at the same location as 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. (See Section 4.6)

Programs which accept commands from a Teletype or a file, depending on how they were started, do so as controlled by the program calling the RUN UUO. The following convention is used with all of Digital's standard CUSPs: 0 in LH of AC means type an asterisk and accept commands from the Teletype. 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. (See LOGOUT command in Chapter 2.)

The RUN UUO can give an error return with one of 13 error codes in AC if any errors are detected. Thus, the user program may attempt to recover from the error and/or give the user a more informative message on how to proceed. Some user programs do not go to the bother of including error recovery code. The Monitor detects this and does not give an error return if the LH of the error return location is a HALT instruction. If this is the case, the Monitor simply prints its standard error message for that type of error and returns the user's 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.

LOOKUP, ENTER, RENAME, RUN, GETSEG UUO Error Codes

FNFEER	0	File not found
IPPEER	1	Incorrect proj-prog no.
PRTERR	2	Protection failure or directory full on DTA
FBMEER	3	File being modified
AEFEER	4[1]	Already existing file
NLEEER	5[1]	Neither LOOKUP or ENTER
TRNEER	6	Transmission error
NSFEER	7	Not a saved file
NECEER	10	Not enough core
DNAEER	11	Device not available
NSDEER	12	No such device
ILUEER	13[1]	Illegal UUO (GETSEG UUO on a one-register machine)

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.

In order to successfully program the RUN UUO for all size systems and for all CUSPs whose size is not known at the time the RUN UUO is coded, it is necessary to understand the sequence of operations it 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.

Does a high segment already exist with desired name? If yes, go to 30.
INIT and LOOKUP file name .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.

[1]Not possible on RUN UUO

Read high file into top of low segment by extending it. (Here again 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

    ERROR IN JOB n
    filename NOT FOUND, UUO AT USER addr

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 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

In order to always be guaranteed of handling the most 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. An even better idea would be to have the error handling code be written once and put in a seldom used (probably non-sharable) 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.

4.3.7.3 CALL AC, [SIXBIT/GETSEG/] or CALLI AC, 40 - These have 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 and shared program overlays. It is also used for run-time routines such as FORTRAN or COBOL operating systems. These programmed operators work exactly like the RUN UUO with the following exceptions.

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

b) The only change that is made to the low segment of the Job Data area is to both halves 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.

See steps 1 through 31 of the RUN UUO description for details of the operation of the GETSEG UUO.

4.3.7.4 CALL AC, [SIXBIT/SPY/] or CALLI AC, 42 - These are used for efficient examination of the Monitor during time sharing. Any number of K of physical core is placed into the user's logical high segment. This amount cannot be saved (no error return if tried), cannot be increased or decreased by the CORE UUO (error return taken), or cannot have the user-mode write protect bit set (error return taken).

The call is:

```
MOVEI AC, Highest physical core location
          desired
CALL  AC, [SIXBIT/SPY/]      ;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).

4.4 INPUT/OUTPUT PROGRAMMING

All user input/output operations are controlled by the use of Monitor programmed operators. These are device independent, in the sense that if an operator is not pertinent to a given device, the operator is treated as a no-operation code. For example, a rewind directed to a line printer does nothing. Devices are referenced by logical names or physical names (see Chapter 2), and the characteristics of a device can be obtained from the Monitor. Properly used, these systems characteristics permit the programmer to delay the device specification for his program from program-generation until program-run time. I/O is accomplished by associating a device, a file, and a ring buffer or command list with one of a user's I/O channels.

4.4.1 File

A file is an ordered set of data on a peripheral device. Its extent on input is determined by an end-of-file condition dependent on the device. For instance, a file is terminated by reading an end-of-file gap from magnetic tape, by an end-of-file card from a card reader, or by depressing the end-of-file switch on a card reader (see 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.

4.4.1.1 Device - To specify a file, it is necessary to specify the device from which the file is to be read or onto which the file is to be written. This specification is made by an argument of the INIT or OPEN programmed operators. Devices are separated into two categories--those with no filename directory, and those with one or more filename directories.

a) Non-directory Devices - For non-directory devices, e.g., card reader, line printer, paper tape reader and punch, and user console, the only file specification required is the device name. All other file specifiers, if given, are ignored by the Monitor. Magnetic tape, which is also a non-directory device, requires, in addition to the name, that the tape be properly positioned. Even though LOOKUP is not required to read and ENTER is not required to write, it is always advisable to use them so that a directory device may be substituted for a non-directory device at run time (using the Monitor command, ASSIGN). Only in this way can user programs be truly device independent.

b) Directory Devices - 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.

4.4.1.2 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. Table 4-3 and Table 4-4 summarize the data modes.

Table 4-3
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.
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 4-4
Unbuffered Data Modes

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.

a) 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 of the IN, INPUT, OUT, and OUTPUT programmed operators points to the first word of the command list. Three types of entries may occur in the command list.

1) 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.

2) XWD 0, y - Causes the next command to be taken from location y. Referred to as a GOTO word.

3) 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, leaving the console in Monitor mode.

b) Buffered Data Modes - Data modes 0, 1, 10, 13, and 14 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. A ring of buffers consists of a 3-word 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 it if it is free. The user's program follows along behind, 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 filling the buffers and the Monitor emptying them.

1) Buffer Structure - A ring of buffers consists of a 3-word header block and a data storage area subdivided into one or more individual buffers linked together to form a ring. The ring buffer layout is shown in Figure 4-1, and explained in the paragraphs which follow.

   (a) Buffer Header Block - The location of the 3-word buffer 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 in use 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 header be used for more than one I/O function at a time.

   (b) Buffer Data Storage Area - The buffer data storage area 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 storage area 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. As Figure 4-2 shows, the right half of the first word contains the file status at the time that the Monitor advanced to the next buffer in the ring. Bit 0 of the second word of a buffer, called 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. 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 using the UUO's 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 UUO's. Bits 1 through 17 of the second word of the buffer contain the size of the data area of the buffer which immediately follows the second word. The size of this data area depends on the device. The right half of the first word of the data area of the buffer, i.e.,

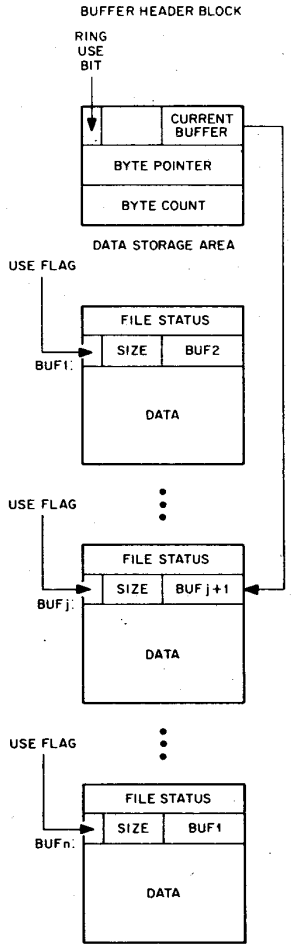


Figure 4-1
User's Ring of Buffers

the third word of the buffer, is reserved for a count of the number of words (excluding itself) 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.

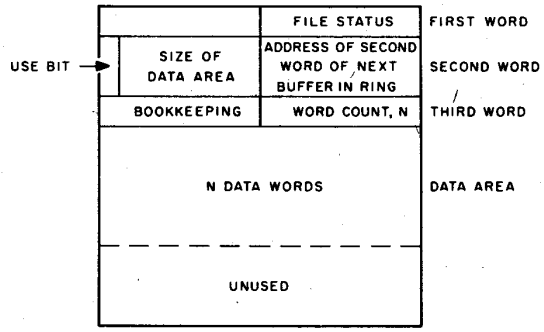


Figure 4-2

Detailed Diagram of Individual Buffer

4.4.1.3 File Status - 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 4-5 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 that the user is currently working on. 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 is given in Chapter 5.

Table 4-5

File Status

Bit	Meaning
18	Improper mode, e.g., attempt to write on a write-locked tape.
19	Device detected error, other than hardware chechsum or parity. Checksum, and/or parity error detected by hardware and/or software.
20	Data error, e.g., a computed checksum failed or invalid data was received.
21	Block too large. A block of data from a device is too large to fit in a buffer, or a block number is too large.
22	End of file.
23	Device is actively transmitting or receiving data.
24-29	Device dependent parameters. (See Chapter 5.)
30	Synchronous input. Stop the device after each buffer is filled.
31	Forces the Monitor to use the word count in the first data word of the buffer (output only). The Monitor normally computes the word count from the byte pointer in the buffer header.
32-35	Data mode. See Table 4-3 and Table 4-4.

4.4.2 Initialization

4.4.2.1 Job 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 input/output 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

(see 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 starting over. The left half of JOBFF is cleared. Any files which have not been closed are deleted on disk. Any older version having the same filename remains. The user-mode write-protect bit is automatically set if a high segment exists, whether it is sharable or not, so that a program cannot inadvertently store into the high segment.

4.4.2.2 Device Initialization

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

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.

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 released. (See RELEAS in this chapter.) The device name, ldev, is either a logical or physical name, with logical names taking precedence over physical names. (See ASSIGN command, Chapter 2.) If the device, ldev, is not the system device, SYS, and is allocated to another job or does not exist, the error return is taken. If the device is the system device, SYS, the job is

put into a system device wait queue, and will continue running when SYS becomes available.

b) Initial File Status - The file status, including the data mode, is set to the value of the symbol STATUS. If the data mode is not legal (see Chapter 5) 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 and leaves the console in Monitor mode.

c) Buffer Header - Symbols OBUF and IBUF, if non-zero, specify the location of the first word of the 3-word buffer header for output and input, respectively. Only those headers which are to 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,[1] 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.

---

[1]Buffer headers may not be in the user's AC's. However, they may be in locations above JOBPFI. (See Table 3.1)

4.4.2.3 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 associated with it a standard buffer size (see Chapter 5). 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 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 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 will automatically attempt to expand the user's core allocation by 1K. If this fails, the Monitor stops the job and prints

ADDRESS CHECK FOR DEVICE ldev AT USER addr

(ldev 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.

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.

```
GO:       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 HEADER
BUF1:     0                                  ;BUFFER 1, 1ST WORD UNUSED
          XWD SIZE+2,BUF2+1                  ;LEFT HALF CONTAINS BUFFER
                                             ;SIZE, 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+2,BUF3+1
          BLOCK SIZE+1
BUF3:     0                                  ;THIRD BUFFER
          XWD SIZE+2,BUF1+1                  ;RIGHT HALF CLOSES THE RING
          BLOCK SIZE+1
```

4.4.2.4 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. Although these operators are not necessary for non-directory devices, it is good programming practice to always use them so that directory devices may be substituted at run time. (See ASSIGN, Chapter 2.)

```
a)  LOOKUP D,E
    error return
    normal return
    .
    .
    .
 E: SIXBIT/file/            ;filename, 1 to 6 characters.
    SIXBIT/ext/             ;filename extension, 0 to 3
                            ;characters.
    0
    XWD project number, programmer number,
```

LOOKUP selects a file for input on channel D. If no device has been associated with channel D by an INIT or OPEN UUO, the Monitor prints

I/O TO UNASSIGNED CHANNEL AT USER addr

and returns the user's console to Monitor mode. If the input side of channel D is not closed (see CLOSE, in this chapter), it is now closed. The output side of channel D is not affected. If the device associated with channel D does not have a directory, the normal return is now taken. If the device has multiple directories, e.g., disk, the Monitor searches the master file directory of the device for the user's file directory whose number is in location E+3 and whose extension is UFD. If E+3 contains zero, the project-programmer pair of the current job is used as the name of the user's file directory. If this file is not found in the master file directory, 1 is stored in bits 33 through 35 of location E+1 and the error return is taken.

The user's file directory or the device directory in the case of a single-directory device (e.g., DECtape) is searched for the file whose name is in location E and whose extension is in the left half of location E+1. If the file is not found, 0 is stored in the right half of E+1 and the error return is taken. If the device is a multiple-directory device (e.g., disk) and the file is found, but is read protected (see File Protection in this chapter),

2 is stored in the right half of location E+1 and the error return is taken. Otherwise, location E+1 through E+3 are filled by the Monitor with the following data concerning the file, and the normal return is taken.

1) The left half of location E+1 remains set to the filename extension.

2) If the device is a multiple-directory device, bits 24 through 35 of location E+1 are set to the date (in the format of DAYTIME programmed operator) that the file was last referenced.

   If the device is a single-directory device, the right half of location E+1 is set to the device block number of the first block of the file.

3) If the device is a multiple-directory device, bits 0 through 8 of location E+2 are set to the file protection. (See "File Protection," this chapter.)

4) Bits 9 through 12 of location E+2 are set to the data mode in which the file was written.

5) Bits 13 through 23 of location E+2 are set to the time, in minutes, and bits 24 through 35 of location E+2 are set to the date (in the format of the DAYTIME programmed operator) of the file's creation, i.e., of the last ENTER or RENAME programmed operator.

6) If the device is a multiple-directory device, the left half of location E+3 is set to the negative of the number of words in the file, and the right half is unchanged. If the file contains more than 2^{17} words, then the left half contains the positive number of 128-word blocks in the file.

   If the device is a single-directory device, location E+3 is used only for SAVed files (see Chapter 3), and contains the IOWD of the core image, i.e., the left half is the negative word length of the file and the right half is the core address of the first word minus 1.

```
b)  ENTER D,E
    error return
    normal return
    .
    .
    .
E:  SIXBIT/file/                ;filename, 1 through 6
                                ;characters.
    SIXBIT/ext/                 ;filename, extension, 0
                                ;through 3 characters.
    EXP<TIME>B23+DATE
    XWD project number, programmer number.
```

ENTER selects a file for output on channel D. If no device has been associated with channel D by an INIT or OPEN UUO, the Monitor prints

I/O TO UNASSIGNED CHANNEL AT USER addr

and returns the user's console to Monitor mode. If the output side of channel D is not closed (see CLOSE in this chapter), it is now closed. The input side of channel D is not affected. If the device does not have a directory, the normal return is now taken.

If the device has multiple directories, e.g., disk, the Monitor searches the master file directory of the device for the user's file directory whose name is in location E+3 and whose extension if UFD. If E+3 contains 0, the project-programmer pair of the current job is used as the name of the user's file directory. If this file is not found in the master file directory, 1 is stored in bits 33 through 35 of location E+1, and the error return is taken. Since a null filename is illegal, if the filename in location E is 0, 0 is stored in bits 33 through 35 of location E+1, and the error return is taken. The user's file directory, or the device file directory in the case of a single-directory device, such as DECtape, is searched for the file whose name is in location E and whose extension is in the left half of location E+1.

If the device is a multiple-directory device and the file is found but is being written or renamed, 3 is stored in bits 33

through 35 of location E+1, and the error return is taken. If the file is write protected (See "File Protection", this chapter), 2 is stored in bits 33 through 35 of location E+1, and the error return is taken.

If the file is found, and is not being written or renamed and is not write protected, then the file is deleted, and the storage space on the device is recovered.

On disk, this deletion of the previous version does not occur until output CLOSE time. Consequently, if the new file is aborted when partially written, the old version remains. On DECtape, the deletion must occur immediately upon ENTER to insure that space is available for writing the new version of the file.

The Monitor then makes the file entry by recording the following information concerning the file and takes the normal return.

a) The filename is taken from location E.

b) The filename extension is taken from the left half of location E+1.

c) If the device is a multiple-directory device, then

1) the current date is taken as the date of last reference,

2) the file protection key is set to 055 (see "File Protection," this chapter),

3) the current data mode is taken as the mode in which the file is to be written,

4) the project number of the current job is taken as the file owner's project number, and

5) if bits 13 through 35 of location E+2 are non-zero, bits 13 through 23 are taken as the time of creation, in minutes, and bits 24 through 35 are taken as the date of creation (in the format of the DAYTIME programmed operator) of the file. Otherwise, the current time and date are used.

If the device is a single-directory device, and if bits 24 through 35 of location E+2 are non-zero, they are taken as

the date of creation; otherwise, the current date is used.

4.4.2.5 File Protection and the RENAME Operator - File protection on non-directory and single-directory devices is obtained by use of the ASSIGN command (see Chapter 2). Multiple-directory devices have a master file directory for the device which contains entries for each user's file directory. File selection (see LOOKUP and ENTER in this chapter) requires specification of the name of a user's file directory and a filename within that directory. Since this permits each user to access all files on the device, a file protection scheme to prevent unauthorized references is necessary. For file protection purposes users are divided into three categories:

a) The file owner is the user whose programmer number is the same as that in the NAME field of the user's file directory in which the file is entered. (Some installations may modify the Monitor to require both project and programmer numbers to match.)

b) Project members are users whose project number is the same as that of the file owner.

c) All other users.

There are three types of protection against each of the three categories of users.

a) Protection-protection - the protection cannot be altered.

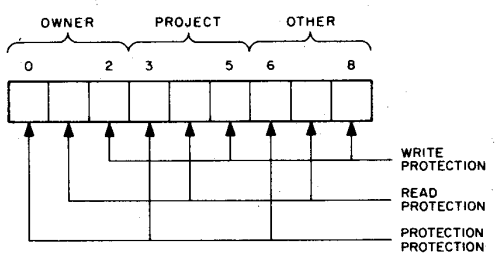


Figure 4-3 File Protection Key

b) Read protection - the file may not be read.

c) Write protection - the file may not be modified.

The file protection key, shown in the foregoing figure, is a set of nine bits which specify the three types of protection for each category of user. (See 5.8.2.4) When a file is created by an ENTER programmed operator, the file protection key is set to 055, indicating that the file is protection-protected and write-protected against all users except the owner. The protection key is returned by the LOOKUP D, E programmed operator in bits 0 through 8 of location E+2. It can be changed by the RENAME programmed operator. The owner's protection-protection and read-protection bits are ignored by the Monitor, thereby preventing a file from becoming inaccessible to everyone. Moreover, the owner protection-protection bit has been taken over to specify that a user wishes to protect his file from deletion when he logs off the system. This feature is handled completely by the LOGOUT CUSP.

```
        RENAME D,E
        error return
        normal return

    E:  SIXBIT/file/     ;filename, 1 through 6 characters.
        SIXBIT/ext/      ;filename extension, 0 through 3 characters.
        EXP<PROT>B8+<TIME>B23+DATE
        XWD project number, programmer number.
```

The RENAME programmed operator (operation code 055) is used to alter the filename, filename extension, and file protection key or delete a file associated with channel D on a directory device.

If no device is associated with channel D, the Monitor prints I/O TO UNASSIGNED CHANNEL AT USER addr and returns the user's console to Monitor mode. If the device is a nondirectory device, the normal return is taken. If no file is selected on channel D, 5 is stored in bits 33 through 35 of location E+1, and the error return is taken.

If the device has multiple directories, e.g., disk, the Monitor searches the master file directory of the device for the user's file directory whose name is in location E+3 and whose extension is UFD. If E+3 contains 0, the project-programmer pair of the current job is used as the name of the user's file directory If this file is not found in the master file directory, 1 is stored in bits 33 through 35 of location E+1, and the error return is taken. The user's file directory, or the device file directory in the case of a single-directory device, is searched for the file currently selected on channel D. If the file is not found, 0 is stored in bits 33 through 35 of location E+1, and the error return is taken.

If the device is a multiple-directory device and the file is found, but is being written or renamed, 3 is stored in bits 33 through 35 of location E+1, and the error return is taken. If the file is owner write-protected or if the protection key is being modified, i.e., bits 0 through 8 of location E+2 differ from the current protection key, and the file is owner protection-protected, 2 is stored in bits 33 through 35 of location E+1, and the error return is taken.

If the new filename in location E is 0, the file is deleted, or marked for deletion, after all read references are completed, and the normal return is taken. If the filename in location E and the filename extension in the left half of location E+1 are the same as the current filename and filename extension, respectively, the protection key is set to the contents of bits 0 through 8 of location E+2, and the normal return is taken.

If the new filename in location E and/or the filename extension in the left half of location E+1 differ from the current filename and/or filename extension, the user's file directory (or the device directory) is searched for the new filename and extension, as

in LOOKUP. If a match is found, 4 is stored in bits 33 through 35 of location E+1, and the error return is taken. If no match is found, the file is changed to the new name in location E, the filename extension is changed to the new filename extension in the left half of location E+1, the protection key is set to the contents of bits 0 through 8 of location E+2, the access date is set to the current date, and the normal return is taken.

4.4.2.6 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, SV JBFF    ;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 FILE NAME 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 FILE NAME
            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 INNAME
            SIXBIT/EXT/
            0
            0
```

4.4.3 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 leaving the console 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 retruns 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) and an OUT (operation code 057), if no end-of-file or error condition exists, i.e., if bits 18 through 22 of the file status are all 0. Otherwise, control returns to the second location following the IN or OUT. Note that IN and OUT UUO's are the only ones in which the error return is a skip and the normal return is not a skip.

4.4.3.1 Unbuffered (Dump) Modes - In data modes 15, 16, and 17, the effective address E of the INPUT, IN, OUTPUT, and OUT programmed operators is the address of the first word of a command list (see Section 4.4.1). Control does not return to the program until transmission is terminated or an error is detected.

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
                               ;AT LOCATION BUFFER
          0                    ;SPECIFIES THE END OF THE COMMAND
                               ;LIST
BUFFER    BLOCK BUFSIZ         ;OUTPUT BUFFER, MUST BE CLEARED
                               ;AND FILLED BY THE MAIN PROGRAM
```

4.4.3.2 Buffered 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 nonzero, 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. (See INBUF, Section 4.4.2.3)

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 does the following.

1. Clears the use bit in the second word of the buffer whose address is 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, such as 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                   ;JSR HERE AND STORE PC
GETCNT:     SOSG  IBUF+2        ;DECREMENT THE BYTE COUNT
            JRST  GETBUF        ;BUFFER IS EMPTY (OR FIRST CALL AFTER
                                ;INIT)

GETNXT:     ILDB AC, IBUF+1     ;GET NEXT CHAR FROM BUFFER
            JUMPN AC @GETCHR    ;RETURN TO CALLER IF NOT NULL CHAR[1]
            JRST  GETCNT        ;IGNORE NULL AND GET NEXT CHAR

GETBUF:     IN    3             ;CALL MONITOR TO REFILL THIS BUFFER
            JRST  GETNXT        ;RETURN HERE WHEN NEXT BUFFER IS
                                ;FULL (PROBABLY IMMEDIATELY)
            JRST  ENDTST        ;RETURN HERE ONLY IF ERROR OR EOF

ENDTST:     STATZ 3, 740000     ;CHECK FOUR ERROR BITS FIRST
            JRST  INERR         ;WHERE TO GO ON AN ERROR
            JRST  ENDFIL        ;WHERE TO GO ON AN END OF FILE
```

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 (see OUTBUF, this chapter). If the ring use bit (bit 0 of the first word of the buffer header) is 1, it is set to 0, the current buffer is cleared to all 0s, and the position and address fields of the buffer byte pointer (the second word of the buffer header) are set so that the first byte is properly stored by an IDPB instruction. The byte count (the third word of the buffer header) is set to the maximum of bytes that may be stored in the buffer, and control is returned to the user's program. Thus, the first OUT or OUTPUT initializes the buffer header and the first buffer, but does not result in data transmission.

If the ring use bit is 0 and bit 31 of the file status is

---

[1] For some devices in ASCII mode, the item count provided will always be a multiple of five characters. Since the last word of a buffer may be partially full, user programs which rely upon the item count should always ignore null characters.

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 first data 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                ;JSR HERE AND STORE PC
                SOSG     OBUF+2  ;INCREMENT BYTE COUNT
                JRST     PUTBUF  ;NO MORE ROOM (OR FIRST CALL AFTER INIT)

PUTNXT:         IDPB AC, OBUF+1  ;STORE THIS CHARACTER
                JRST    @PUTCHR  ;AND RETURN TO CALLER

PUTBUF:         OUT      3       ;CALL MONITOR TO EMPTY THIS BUFFER
                JRST     PUTNXT  ;RETURN HERE WHEN NEXT BUFFER IS
                                 ;EMPTY (PROBABLY IMMEDIATELY)
                JRST     OUTERR  ;RETURN HERE ONLY IF OUTPUT ERROR

OUTERR:         GETSTS   3,AC    ;GET THE ERROR STATUS TO LOOK AT
                  .
                  .
                  .
```

4.4.4 Status Checking and Setting

The file status (see Table 4-5) is manipulated by the GETSTS (operation code 062), STATZ (operation code 063), STATO (operation code 061) and SETSTS (op code 060) 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 leaving the console 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.

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 leaving the console 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). Changing the output item count should be done using the count already placed there by the Monitor and dividing or multipling by the appropriate conversion factor, rather than assuming the length of a buffer.

4.4.5 Terminating A File (CLOSE)

File transmission is terminated by the CLOSE D,N (Operation code 070) programmed operator. If no device is associated with channel D or if bits 34 and 35 of the instruction are both 1, control returns to the user's program immediately.

If bit 34 is 0 and the input side of data channel D is open, it is now closed. In data modes 15, 16, and 17, the effect is to execute a device dependent function and clear the end-of-file flag, bit 22 of the file status. Data modes 0, 1, 10, 13, and 14 have the additional effect, if an input buffer ring exists, of setting the ring use bit (bit 0 of the first word of the buffer header) to 1, setting the buffer byte count (the third word of the buffer header) to 0 and setting the buffer use bit (bit 0 of the second word of the buffer) of each buffer to 0.

If bit 35 of the instruction is 0 and the output side of channel D is open, it is now closed. In data modes 15, 16, and 17, the effect is to execute a device dependent function. In data modes 0, 1, 10, 13, and 14, if a buffer ring exists, the following operations are performed.

a) All data in the buffers that has not yet been transmitted to the device is now written.

b) Device dependent functions are performed.

c) The ring use bit is set to 1.

d) The buffer byte count is set to 0.

e) Control returns to the user after transmission is complete.

Example:

Terminating A File

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

4.4.6 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 input/output 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 Magtape spacing operation, since no data transfer is in progress. An MTAPE D, 0 (see Section 5.7.2) should be used to wait for spacing and I/O activity to finish on Magtape. 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.

4.4.7 Relinquishing A Device (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 (see Chapter 2), it is returned to the Monitor's pool of available facilities. Control is returned to the user's program.

4.5 CORE CONTROL

4.5.1 CALL AC,[SIXBIT/CORE/] or CALLI, 11 - These provide a user program with the ability to expand and contract its core size as its memory requirements change. In order 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 memory.

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 (see section 4.6). A RH of 0 leaves the low segment core assignment unaffected. The Monitor clears new core before assigning it to the user, so that privacy of information is insured.

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 non-swapping 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 non-swapping 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 UUO's should be used in order 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, non-sharable 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, and d) initialized from the storage device by GET,R, or RUN commands or RUN or GETSEG UUO's. This is the same sequence which the Loader and the SAVE and GET commands use to create and initialize new sharable segments.

4.5.2 CALL AC, [SIXBIT/SETUWP/]or CALLI AC,36 - These allow 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:

```
CALL AC, [ SIXBIT/SETUWP/ ]    ; 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. This happens 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 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 each set or clear the bit can be written, provided the subroutines save the previous value of the bit and restore it upon returning to its caller.

4.6 Modifying Shared Segments, and Meddling

Usually a high segment is write-protected, but it is possible for a user program to turn off the user write-protect bit or to increase or decrease a shared segment's core assignment by using the SETUWP or CORE UUO's. These are legal from the high or low segment, provided 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 do any of the foregoing with a non-sharable program. It is never considered meddling to type ↑C followed by START (withoug 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 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 so as to protect the high segment in case it is being shared. The command and the two UUO's 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. Whenever 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 segment's name will be set to 0. New users will not share the older version, but will 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 which 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 (2) which the shared program is prepared to handle, then the shared program can do anything it has the privileges to do, even though 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 from 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 time-sharing, if the programmer has the access privileges. In a DECtape system, a new CUSP tape can be mounted followed by an ASSIGN SYS command which clears segment namesfor the physical device but does not assign the device because everyone needs to share it.

CHAPTER 5

Device Dependent Functions

This chapter explains the unique features of each standard I/O device. All devices accept the programmed operators explained in Chapter 4 unless otherwise indicated. Buffer sizes are given in octal and include two bookkeeping words. The user may determine the physical characteristics associated with a logical device name by executing a DEVCHR UUO. (See 5.12.) Table 5-1 is a summary of the characteristics of all devices.

Table 5-1

Device Summary

Physical Name	Name	Hardware Type Number	Programmed Operator	Data Modes	Buffer[1] Size (Octal)
CTY	Console Teletype	626 Models 33, 35, 37	INPUT, IN OUTPUT, OUT	A, AL	23
TTY0, TTY1, ..., TTY77	Teletype	630, 680, or DC10	INPUT, IN OUTPUT, OUT, TTCALL	A, AL	23
PTY	Pseudo-Teletype	None	INPUT, IN OUTPUT, OUT	A, AL	23
PTR	Paper Tape Reader	760	INPUT, IN	A, AL, I, B, IB	43
PTP	Paper Tape Punch	761	OUTPUT, OUT	A, AL, I, B, IB	43
PLT	Plotter	XY 10	OUTPUT, OUT	A, AL, I, B, IB	46
LPT or LPT0, ..., LPT7	Line Printer	646, LP10	OUTPUT, OUT	A, AL, I,	34
CDR	Card Reader	461, CR10	INPUT, IN	A, AL, I, B	36

Table 5-1 (Cont.)

Device Summary

Physical Name	Name	Hardware Type Number	Programmed Operator	Data Modes	Buffer[1] Size (Octal)
CDP	Card Punch	CP10	OUTPUT, OUT	A, AL, B, IB	35
DTA0, DTA1, ..., DTA7	DECtape	551/555, TD10/TD55	INPUT, IN OUTPUT, OUT LOOKUP ENTER MTAPE USETO USETI UGETF CALL [SIXBIT/UTPCLR/]	A, AL, I, B, IB, DR D	202
MTA0, MTA1, ..., MTA7	Magnetic Tape	516, TM10 TU20, TU79	INPUT, IN OUTPUT, OUT MTAPE	A, AL, I, B, IB, DR, D	203
DSK	Disk	RC10	INPUT, IN OUTPUT, OUT LOOKUP ENTER RENAME USETO	A, AL, I, B, IB, DR, D	203
DIS	Display	30, 340	INPUT OUTPUT	ID	Dump only

[1]Buffer sizes are subject to change and should be calculated rather than assumed by user programs. A dummy INBUF or OUTBUF may be employed for this purpose.

5.1 TELETYPE

Device Name - TTY0, TTY1, ..., TTY76, TTY77, CTY

Line number n of the Type 630 Data Communications System, Data Line Scanner DC10, PDP-8 680 System, or PDP-8/I 680I System is referred to as TTYn. The console Teletype is CTY. The Time-Sharing Monitor automatically gives the logical name, TTY, to the user's

console whenever a job is initialized.

Teletype device names are assigned dynamically. For inter-console communication by program, it is necessary for one of the two users to type DEASSIGN TTY in order to make his Teletype available to the other user's program as an output or input device. Typing ASSIGN TTYn is the only way to reassign a Teletype that has been de-assigned. Also see TALK command, Chapter 2.

Buffer Size - 23_8 words.

Two choices of Teletype routines are provided: a newer, full duplex software routine and an older, half duplex software routine. Use of the full duplex software is encouraged.

With a full duplex Teletype service, the two functions of a console, typein and typeout, are handled independently and need not be handled in the strict sense of output first and then input. For example, if two operations are desired from PIP, the request for the second operation can be typed before receiving the asterisk after completion of the first. The echo of characters typed in will disappear since the keyboard and the printing operations are independent. To stop output that is not wanted, a "Control O" is typed. Also, the command "Control C" will not stop a program instantly. Rather, 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 will stop the program.

Programs waiting for Teletype output will be 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 probability of lost typein.

5.1.1 Data Modes

5.1.1.1 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.
003	↑C	The Teletype mode is switched to Monitor mode the next time input is requested by the program. Two successive ↑C's cause the mode to be switched to Monitor mode immediately.
004	↑D (EOT)	004 passed to program. Not echoed, so typing in a "Control D" (EOT) will not cause a full duplex dataphone 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.
007	↑G (Bell)	007 passed to program, and is a break character.
010	↑H (Back-space)	Acts as a RUBOUT, unless either DDT mode or full character set mode is true, or the ↑F switch is on. In these cases, 010 is sent to the program.
011	↑I (TAB)	011 passed to program. Echoed as spaces if Teletype is a model 33 (determined by ↑P switch). Spaces are not passed to program.
012	↑J (Line-feed)	Is a break character. No other special action.
013	↑K (Vertical Tab)	013 passed to program. Echoes as four linefeeds, if a model 33. Is a break character. Linefeeds are not passed to program.
014	↑L (Form)	014 passed to program. Echoes as 8 linefeeds on a 33. Is a break character. 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 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	Suppresses output until an INPUT, or an INIT, or OPEN UUO occurs. Not passed to program. Typed as ↑O followed by carriage return-linefeed.

020	↑P	Does not appear in the input buffer. Some Teletype units (usually Models 35 and 37) have horizontal tab, vertical tab, and form feed mechanisms while other units (usually Model 33s) do not. If the user finds that his particular Teletype unit does not have these mechanisms, he should type ↑P. Otherwise, tabs will not be printed at all or spaces will be substituted for a tab depending upon the Monitor's assumption.
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. 023 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 end-of-file on Teletype input. Echoes as ↑Z followed by carriage return-linefeed. Is a break character. Appears in buffer as 032.
033	↑[(ESC)	This is the ASCII altmode these days, but is translated to 175 before being passed to the program, unless in full character set mode (bit 29 in INIT). 175 is the 1963 altmode. Echoes as a dollar sign. Always, is a break character.
034	↑\	No special action.
035	↑]	No special action.
036	↑↑	No special action
037	↑←	No special action.
040-137		Printing characters, no special action.
140-174		"Lower case" ASCII. Translated to upper case, unless ↑F switch is set. Echoes as upper case if translated to upper case.
175 and 176		Old versions of altmode. See description of "ESC" (033).
177		RUBOUT or DELETE: A) Completely ignored if in papertape mode (XON) B) Is a break character, passed to program if either DDTmode or fullcharacter-set mode is true. C) Otherwise (ordinary case) causes a character to be deleted for each rubout typed. All the characters deleted are echoed between a single pair of backslashes. If no characters remain to be deleted, echoes as a carriage return-linefeed.

On output, all characters are typed just as they appear in the output buffer with the exception of TAB, VT, and FORM, which are processed the same as on type in.

5.1.1.2 Half-Duplex Software A(ASCII) - If, during output operations, an echo-check failure occurs (the transmitted character was not the same as the intended character), the I/O routine suspends output until the user types the next character. If that character is ↑C, the console is placed in Monitor mode immediately. 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 and only then he types ↑C.

The buffer is terminated when it fills up or when the user types ↑Z.

5.1.1.3 Half-Duplex Software AL(ASCII Line) - Same as ASCII mode (usually preferred) with the addition that the input buffer is terminated by a CR/LF pair, FF, VT, or ALTMODE.

5.1.2 DDT Submode

To allow a user's program and the DDT debugging program to use the same Teletype without interfering 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 in order to do 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 bufferfuls of characters at a time.

To input characters in DDT mode, use the sequence

```
MOVEI AC,BUF
CALL   AC, [SIXBIT/DDTIN/]
```

BUF is the first address of a 21-word block in the user's area. The DDTIN operator delays, if necessary, until one character is typed in. Then all characters (in 7-bit packed format) typed in since the previous occurrence of DDTIN are moved to the user's area in locations BUF, BUF+1, etc. 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.

5.1.3 Special Programmed Operator Service

TTCALL UUO is (and will always be) implemented only in the "full duplex scanner service", SCNSRF. The general form of this UUO is as follows:

```
OPDEF   TTCALL    [51B8]
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 low segment above JOB 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	GETLIN	Get line characteristics
7	SETLIN	Set line characteristics
10	RESCAN	Reset input stream to command
11	CLRBFI	Clear typein buffer
12	CLRBFO	Clear typeout buffer
13	SKPINC	Skips if a character can be input
14	SKPINL	Skips if a line can be input
15-17	(Reserved for Expansion)	

INCHRW TTCALL 0,ADR

This command inputs a character into location ADR. ADR may be an AC or any other location in the user's low segment. If there is no character yet typed, the program waits for it.

OUTCHR 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 rest need not be zeroes.

If there is no room in the output buffer, the program waits until room is available. ADR may be in high segment.

INCHRS 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          ;NO TYPEIN
        JRST     DONE          ;CHARACTER IN ADR
```

OUTSTR 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

INCHWL 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, since INCHRW causes a swap to occur for each character rather than each line (compare DDT and PIP input, for instance).

INCHSL 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.

GETLIN 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 a display console.
3	Line is a dataset data line.
4	Line is a dataset control line.
5	Line is half-duplex.

Bit	Meaning
11	A line has been typed in by the user.
12	A rubout has been typed.
13	"Control F" switch is on.
14	"Control P" switch is on.
15	"Control B" switch is on.
16	"Control Q" (paper tape) switch is on.
17	Line is in a "talk" ring.

SETLIN TTCALL 7,ADR

This command allows a program to set and clear some of the bits described for GETLIN. They may be changed only for the controlling Teletype. The bits which may be modified are bits 13, 14, 15 and 16. Example:

```
SETO   AC,0
TTCALL 6,AC
TLZ    AC,BIT 13
TLO    AC,BIT 14
TTCALL 7,AC
```

RESCAN TTCALL 10,0

This command is intended for use only by the CCL CUSP. It causes the Input Buffer to be re-scanned 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 is address checked).

CLRBFI TTCALL 11,0

This command causes the Input Buffer to be cleared (as if the user had typed a number of "Control U's"). It is intended to be used when an error has been detected, such that a user probably would not want any commands to be executed which he might have typed ahead.

CLRBFO TTCALL 12,0

This command causes the output buffer to be cleared, as if the user had typed "CONTROL O". It should be used only rarely, since usually one wants to see all output, up to the point of an error. It is included primarily for completeness.

SKPINC 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 compute 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.

SKPINL TTCALL 14,0

This command is the same as SKPINC except that a skip occurs if a line has been typed.

5.1.4 Special Status Bits (Full Duplex Software Only)

An INIT or OPEN, with bit 28 a one, suppresses echoing on the Teletype. This is useful for LOGIN to eliminate the mask for the password.

5.1.5 Paper Tape Input from the Teletype (Full Duplex Software Only

Paper tape input is possible from a Teletype equipped with a paper tape reader, controlled by the XON and XOFF characters. When commanded by the XON character, the Teletype service will read 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 Model 33's, to insure output of the reader control characters. In order to use paper tape processing, the Teletype with paper tape reader must be connected by a full duplex connection and only ASCII paper tapes are intended to be used.

The correct operating sequence for reading a paper tape in this way is as follows:

```
.R PIP <RETURN>
*DSK: FILE←TTY: <XON><RETURN><LINEFEED>
THIS IS WHAT IS ON TAPE
MORE OF SAME
LAST LINE
↑Z
*<XOFF>
```

5.2 PAPER TAPE READER

Device Mnemonic - PTR

Buffer Size - 43_8 words

5.2.1 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 depressed.

5.2.1.1 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 end-of-file.

5.2.1.2 AL (ASCII Line) - Character processing is the same as for the A mode. The buffer is terminated by LINE FEED, FORM, or VT.

5.2.1.3 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 end-of-file indication but does not cause a character to appear in the buffer.

5.2.1.4 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.

5.2.1.5 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 2s complement arithmetic, then splitting the sum into three 12-bit bytes and adding these using 1s complement arithmetic to form a 12-bit checksum. The data error status flag (see Table 4.5) 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 end-of-file indication but does not result in putting a character in the buffer.

5.3 PAPER TAPE PUNCH

Device Mnemonic - PTP

Buffer Size - 43_8 words

5.3.1 Data Modes

5.3.1.1 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.

5.3.1.2 AL (ASCII Line) - The same as A mode. Format control must be performed by the user's program.

5.3.1.3 I (Image) - Eight-bit characters are punched exactly as they appear in the buffer with no additional processing.

5.3.1.4 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.

5.3.1.5 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.

5.3.2 Special Programmed Operator Service

The first output programmed operator of a file causes about two fanfolds of blank tape to be punched as leader. Following a CLOSE, an additional fanfold of blank tape is punched as trailer. No end-of-file character is punched automatically.

5.4 LINE PRINTER

Device Mnemonic - LPT

Buffer Size - 34_8 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. See 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) - 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.

5.5 CARD READER

Device Mnemonic - CDR

Buffer Size - 36_8 words

5.5.1 Data Modes

5.5.1.1 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. A card with the character 12-11-0-1 punched in column 1 is an end-of-file card. Columns 2 through 80 are ignored. The end-of-file button on the card reader has the same effect as the end-of-file card. 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 IBM 026 card codes. If a card containing a 12-0-2-4-6-8 punch in column 1 is encountered, any following cards are translated as 029 codes (see Table 5-2

PDP-10 Card Codes) 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.

5.5.1.2 AL (ASCII Line) - Exactly the same as the A mode.

5.5.1.3 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 end-of-file card and the end-of-file button are processed the same as in the A mode. Cards are not split between two buffers.

5.5.1.4 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 12-2. The absence of the 7-9 punch results in raising the IOIMPM (improper mode) flag in the card reader status word. Card column 2 must contain a 12-bit checksum as described for the paper tape reader 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 end-of-file card and the end-of-file button are processed the same as in the A mode with a word containing 003200000000 appearing as the last word in the file.

5.6 CARD PUNCH

Device Mnemonic - DCP

Buffer Size - 35_8 words

5.6.1 Data Modes

Table 5-2

PDP-10 Card Codes

CHAR	PDP-10 ASCII	DEC 029	DEC 026
SPACE	040		
!	041	1182	1287
"	042	87	085
#	043	83	086
$	044	1183	1183
%	045	084	087
&	046	12	1187
'	047	85	86
(	050	1285	084
)	051	1185	1284
*	052	1184	1184
+	053	1286	12
,	054	083	083
-	055	11	11
.	056	1283	1283
/	057	01	01
0	060	0	0
1	061	1	1
2	062	2	2
3	063	3	3
4	064	4	4
5	065	5	5
6	066	6	6
7	067	7	7
8	070	8	8
9	071	9	9
:	072	82	1182 or 110
;	073	1186	082
<	074	1284	1286
=	075	86	83
>	076	086	1186
?	077	087	1282 or 120

CHAR	PDP-10 ASCII	DEC 029	DEC 026
@	100	84	84
A	101	121	121
B	102	122	122
C	103	123	123
D	104	124	124
E	105	125	125
F	106	126	126
G	107	127	127
H	110	128	128
I	111	129	129
J	112	111	111
K	113	112	112
L	114	113	113
M	115	114	114
N	116	115	115
O	117	116	116
P	120	117	117
Q	121	118	118
R	122	119	119
S	123	02	02
T	124	03	03
U	125	04	04
V	126	05	05
W	127	06	06
X	130	07	07
Y	131	08	08
Z	132	09	09
[	133	1282	1185
\	134	1187	87
]	135	082	1285
↑	136	1287	85
←	137	085	82

5.6.1.1 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 to be raised. The CLOSE will punch the last partial card and then punch an EOF Card

(12-11-0-1 in column 1).

Cards are normally punched with DEC026 card codes. If bit 26 (octal 1000) of the status word is on (from INIT, OPEN, or SETSTS), cards are punched with DEC029 codes. The first card of any file 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).

5.6.1.2 AL (ASCII Line) - The same as A mode.

5.6.1.3 IB (Image Binary) - Up to 26 2/3 data words will be punched in columns 1-80. The buffer set up by the Monitor will only contain room for 26 data words. To punch a full 80-column card, the user has to set up his own buffers. Image binary will cause exactly one card to be punched for each output. The CLOSE will punch the last partial card, and then punch an EOF card (12-11-0-1 in column 1).

5.6.1.4 B (Binary) - Column 1 will contain the word count in rows 12-2. A 7-9 punch will also be in column 1. Column 2 will contain a checksum; columns 3-80 will contain up to 26 data words, 3 columns per word. Binary will cause exactly one card to be punched for each output. The CLOSE will punch the last partial card, and then punch an EOF card (12-11-0-1 in column 1).

5.6.2 Special Programmed Operator Service

Following a CLOSE, an end-of-file card is punched.

Both the first card of the file (the one that identifies the card code used) and the end-of-file card are laced in columns 2 through 80 for easy identification of files. These laced punches

are ignored by the card reader service routine.

5.7 DECTAPE

Device Mnemonic - DTA0, DTA1, ..., DTA7

Buffer Size - 202_8 words

5.7.1 Data Modes

5.7.1.1 A (ASCII) - Data is written on DECtape exactly as it appears in the buffer. 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. See the description of DECtape format below for further information concerning blocking of information.

5.7.1.2 AL (ASCII Line) - Same as A.

5.7.1.3 I (Image) - Same as A. Data consists of 36-bit words.

5.7.1.4 IB (Image Binary) - Same as I.

5.7.1.5 B (Binary) - Same as I.

5.7.1.6 DR (Dump Records) - This mode is accepted but actually functions as dump mode 17.

5.7.1.7 D (Dump) - 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 as described in

Chapter 4, "Unbuffered (Dump) Modes;" any positive number appearing in a command list terminates the list. Dump data is automatically blocked into standard-length DECtape blocks by the DECtape control. Unless the number of data words is an exact multiple of the standard length of a DECtape block (128_{10}), after each output programmed operator, the remainder of the last block written is wasted. The input programmed operator must specify the same number of words that the corresponding output programmed operator specified in order to skip over the wasted fractions of blocks.

5.7.2 DECtape Block Format

A standard reel of DECtape consists of 578 (1102_8) pre-recorded blocks each capable of storing 128 (200_8) 36-bit words of data. Block numbers which 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 time-sharing 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 1_{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 Block Too Large flag (bit 21) in the file status and returns.

5.7.3 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.

Words 0 through 82_{10}

The first 83 words of the directory contain "slots," each "slot" representing one of the 577 (blocks 1 through 1101_8 are represented in these 83 words) blocks on the DECtape. Each slot occupies five bits (seven slots are stored per word) and contains the number of the file ($1-26_8$) to which the block the slot represents is assigned.

Words 83 through 104_{10}

The next 22 words contain the filenames of the 22 files residing on the DECtape. Word 83 contains the filename for file #1, word 84 the filename #2, etc. Filenames are stored in 6-bit code.

Words 105 through 126_{10}

The next 22 words contain the extension names and dates of the 22 files, in the same relative order as their filenames above.

Bits 0 through 17_{10}	The extension name of the file (in 6-bit code).
Bits 18 through 23_{10}	Number of 1K blocks minus 1 needed to load the file (maximum value = 63) This information is stored for SAVEd files only.
Bits 24 through 35_{10}	The date the file was last updated, according to the formula: ((year-1964)*12+(month-1))*31+day-1
Word 127_{10}	Unused.

The message

BAD DIRECTORY FOR DEVICE DTAn: EXEC CALLED FROM USER LOC n

is produced whenever any of the following conditions are detected.

a. A parity error while reading the directory block.

b. No "slots" are assigned to the file number of the file.

c. The tape block which may possibly 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.

5.7.4 DECtape File Format

A file consists of any number of DECtape blocks. Each block contains:

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 which 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.[1]

5.7.5 Special Programmed Operator Service

Several programmed operators are provided for manipulating DECtape. These allow the user to manipulate block numbers and to handle directories.

[1]The Monitor compresses the user's core image by squeezing out blocks of two or more consecutive zeroes before creating the SAVed files; files with extension .SAV may be read in Dump Mode, but must be reexpanded before being run. The Monitor takes this action after input on a RUN or GET.

In addition to the operators above, INPUT, OUTPUT, CLOSE, and RELEAS have special effects. When performing nondump input operations, the DECtape service routine reads the links in each block to determine the next block to read and when to raise the end-of-file flag.

When an OUTPUT is given, the DECtape service routine examines the left half of the first data 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.

Table 5-3

DECtape Programmed Operators

Programmed Operator	Effect
USETI D, E	Sets the DECtape on device channel D to input block E next. Input operations on this DECtape must not be active because otherwise the user has no way of determining which buffer contains block E.
USETO D, E	Similar to USETI but sets the output block number. USETO waits until the device is inactive before setting up the new output block number.
UGETF D, E	Places the number of the first free block of the file in user's location E.
ENTER D, E	User's location E, E+1, E+2, and E+3, must be reserved for a directory entry. The DECtape service routine searches the directory for a filename and extension that match the contents of E and the left half of E+1. If no match is found and there is room in the directory, the service routine places the first free block number into the right half of E+1, places the date in E+2 (unless already non-zero), and places the necessary

Table 5-3 (Cont)

DECtape Programmed Operators

Programmed Operator	Effect
	information into the directory. If a match is found, similar actions occur, but the new entry replaces the old. If there is no room in the directory, ENTER returns to the next location. Otherwise, ENTER skips one location.
LOOKUP D, E error return	Similar to ENTER but sets up an input file. The contents of E and E+1 are matched against the filenames and extension names in the DECtape directory. If a match is found, information about the file is read from the directory into the appropriate portions of the 4-word block beginning at E. The first block of the file is then found as follows. 1. The first 83 words of the DECtape directory are searched in a backwards manner, beginning with the slot immediately prior to the directory block, until the first slot containing the desired file number is found. 2. The block associated with this slot is then 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 they are equal to the block number of this block, then this block is the first block of the file; if not, then the block with that block number is read as the first block of the file. LOOKUP then skips one location. If no match is found, LOOKUP returns to the user's program at the next location.
CALL D, [SIXBIT/UTPCLR/]	UTPCLR clears the directory of the DECtape on device channel D. A cleared directory has zeroes in the first 83 words except in those slots related to blocks 0, 1, 2, and 100_{10} and nonexistent blocks 1102 through 1105_8. Only the directory block (block 100) is affected by UTPCLR; the other blocks are unaffected. This programmed operator does nothing if the device on channel D is not DECtape.
RENAME D, E	This programmed operator is used to alter the name and extension of a file or to delete it from the DECtape. Locations E to E+3 are as in LOOKUP and ENTER. To be RENAMEd a file must first be CLOSEd on channel D, in order to identify for the for the RENAME UUO. RENAME then seeks

Table 5-3 (Cont)

DECtape Programmed Operators

Programmed Operator	Effect
	out this file and 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 return is given if the new file name and extension already exist or if neither a LOOKUP nor an ENTER has been done to identify the file to be renamed.

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 RELEAS 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 RELEAS of all devices and, thus, write out a changed directory even though the user's program failed to give a RELEAS.

Two other special programmed operators are available: MTAPE D, 1 and MTAPE D, 11. MTAPE D, 1 rewinds the DECtape and moves it into the end zone at the front of the tape. MTAPE D, 11 rewinds and unloads the tape, pulling the tape completely onto the lefthand 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.

5.7.6 Special Status Bits

If an attempt is made to write on a unit with the WRITE-LOCK switch on, the message

```
DEVICE DTAn OK?
↑C
.
```

is typed on the user's Teletype. When the situation has been rectified, CONT may be typed to proceed as normal.

5.7.6.1 Special DECtape Status Bits - An INIT or SETSTS to a DECtape with bit 29 ON informs the DECtape service routine that the DECtape is in nonstandard format. This implies that no file-structured operations will be performed on that tape. Blocks will be read or written sequentially; no links will be generated (output) or recognized (input). The first block to be read or written must be set by a USETI or USETO. In Dump Mode, 200_8 data words per block will be read or written (as opposed to the normal 177_8 words). No "dead reckoning" will be used on a search for a block 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, as the data must be read in a forward direction and block 0 normally cannot be read forward.

5.7.7 Important Considerations

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 must use an ASSIGN command. The directory from the old tape

could 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.

5.8 MAGNETIC TAPE

Magnetic tape format is industry compatible, 7- or 9-channel 200, 556, and 800 bpi (see description below).

Device Mnemonic - MTA0, MTA1, ..., MTA7

Buffer Size - 203_8 words

5.8.1 Data Modes

5.8.1.1 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 556 bits per inch. A maximum of 200_8 words per record is standard. The word count is not written on the tape.

5.8.1.2 AL (ASCII Line) - Same as A.

5.8.1.3 I (Image) - Same as A but data consists of 36-bit words.

5.8.1.4 IB (Image Binary) - Same as I.

5.8.1.5 B (Binary) - Same as I.

5.8.1.6 DR (Dump Records) - Standard fixed length records (128

words is the standard unless installation standard is changed with MONGEN) 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 as described in Chapter 4, "Unbuffered (Dump) Modes." 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, as many standard length records are written followed by one short record to exactly write all of the words on the tape.

5.8.1.7 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 as described in Chapter 4, "Unbuffered (Dump) Modes." 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. See Section 4.4.1.2 for command list format.

5.8.2 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 end-of-file marks which signal the end of the data set or the end of the file. Files are read from and written on the tape in a sequential manner.

An end-of-file mark consists of a record containing a 17_8 (for 7-channel tapes) or a 23_8 (for 9-channel tapes). End-of-file marks are used in the following manner.

a. No end-of-file mark precedes the first file on a magtape.

b. An end-of-file mark follows every file.

c. Two end-of-file marks follow a file if that file is the last or only file on the tape.

Files are written on and read from a magtape in a sequential manner. 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.

5.8.3 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 end-of-file marks and backspaces over one of them. If another file is now opened, the second end-of-file is wiped out leaving one end-of-file between files. At the end of the in-use portion of the tape, however, there appears a double end-of-file character which is defined as the logical end of tape. When an input dump file is closed, the I/O service routine automatically skips to the next end-of-file.

A special programmed operator called MTAPE pfovides for such tape manipulation functions as rewind, backspace record, backspace file, 9-channel tape initialization, etc. 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 the following table:

Table 5-4

MTAPE Functions

Function	Action
0	No operation; wait for spacing and I/O to finish
1	Rewind to load point
11	Rewind and unload[1]
7	Backspace record
17	Backspace file
3	Write end of file
6	Skip one record
13	Write 3 inches of blank tape
16	Skip one file
10	Space to logical end of tape
100	Digital Compatible; 9-channel[2]
101	Initialize for 9-channel tape[3]

MTAPE waits for the magnetic tape unit to complete whatever action is in progress before performing the indicated function, including

---

[1]On the 516 Control, this function is not currently implemented as such, but is treated as a Rewind function only.

[2]Digital Compatible mode writes (or reads) 36 data bits in five frames of a 9-track magtape. 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.

[3]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 bpi density, odd parity.

no operation (0). Bits 18 through 25 of the status word are then cleared, the indicated function is initiated, and control is returned to the user's program immediately. 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.

5.8.3.1 Backspace File on Magtape - Issuing a backspace file command to a magtape unit will move the tape in the reverse direction until the tape has A) passed the end of file mark or B) reached the beginning of the tape. This means that the end of the backspace file operation will position 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 you've reached the beginning of the tape; 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 WAIT           ;*Wait for completion*
STATO   MT, 4000     ;Beginning of tape?
MTAPE   MT, 16       ;No, skip over file mark
```

Note that it is necessary to wait after the backspace file instruction in order to insure that the tape is moved to the end-of-file mark or the beginning of the tape before testing to see whether or not it is the beginning of the tape. The instruction CALLI 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
```

In this case the device service routine must wait until the magtape 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.

5.8.4 9-Channel Magtape

Nine-channel magtape may be written and read in two ways: normal Digital Compatible format, and industry compatible format.

5.8.4.1 Digital Compatible Mode - Digital Compatible mode is the usual mode and will allow 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 in the case of 7-track magtape. It is assumed that most DEC magtapes will be written and read this way.

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

Data Word in Core - 5 magtape bytes/36-bit word. Parity bits are unavailable to the user. Bits are written on tape as shown in diagram, note that bits 30 and 31 get written twice and that 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.

5.8.4.2 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 (which returns the device to 9-channel Digital Compatible status) by a RELEAS, a call to EXIT, or an MTAPE D, 100 UUO. These MTAPE UUO's act only as a switch to and from industry compatible mode and in no other way affect I/O status, except to set the density to 800 bpi 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

Data Word in Core - four magtape bytes carry 4 8-bit bytes from data word, parity bits are obtained as shown when reading. Rightmost four bits are ignored on writing (bits 32-35).

5.8.4.3 Changing Modes - MTAPE CH, 101 automatically sets density at 800 bits (or 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 the device is initialized in. Byte pointers and byte counts in buffer header will have to be changed by the user in order to operate on eight-bit bytes.

5.8.5 Special Status Bits

Special bits of the status word are reserved for selecting the density and parity mode of the magnetic tape. Table 5-4 lists the bits that are set and cleared by INIT or SETSTS.

Table 5-5

Magnetic Tape Special Status Bits.

Bit	Action
18[1]	Improper mode. When set to one during an output operation means that the write enable ring is out.
24[1]	I/O Beginning of Tape. The tape is at the load point.
25[1]	I/O Tape END. The tape is at or past the end point.
26	I/O Parity. 0 for odd parity, 1 for even parity[2]
27-28	I/O Density. 00 = System Standard (defined at MONGEN time) 01 = 200 bpi 10 = 556 bpi 11 = 800 bpi
29	I/O No Read Check. Suppress automatic error correction if bit 29 is a 1. Normal error correction is to repeat the desired operation 10 times before setting an error status bit.

[1]These bits indicate special magnetic tape conditions and are set by the magnetic tape service routine when the conditions occur.

[2]Odd parity is preferred. Even parity should be used only when creating a tape to be read in BCD (Binary Coded Decimal) on another computer.

5.9 DISK

Device Mnemonic - DSK

Buffer Size - 203_8 words (of which 200_8 words are data)

5.9.1 Data Modes

5.9.1.1 A (ASCII) - Data is written on the disk exactly as it appears in the buffer. Data consists of 36-bit words.

5.9.1.2 AL (ASCII Line) - Same as A.

5.9.1.3 I (Image) - Same as A.

5.9.1.4 IB (Image Binary) - Same as I.

5.9.1.5 B (Binary) - Same as I.

5.9.1.6 DR (Dump Records) - Functions exactly the same as D.

5.9.1.7 D (Dump)- 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 as described in Chapter 4, "Unbuffered (Dump) Modes." 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.

5.9.2 Structure of Files on Disk

The file structure of the disk system has been designed to minimize the number of disk seeks for sequential or random accessing using either buffered or dump mode I/O. The assignment of physical space for data is performed automatically by the Monitor as logical files are written or deleted by user programs. Files may be of any length, and each user may have as many files as he wishes, as long as disk space is available. No initial estimate of file length or number of files need be given by users or their programs. 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 (see "General Notes," 5.9.3.3).

5.9.2.1 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. There are two programs in the Time-Sharing Monitor that service the disk, DSKSER and DSKINT. DSKSER is the device service routine for a disk and references a disk by symbolic addressing only. This routine is essentially independent of what physical disk is attached to the system. DSKINT serves only two functions: 1) that of translating the logical addressing used elsewhere in the system to the physical addressing of the particular disk being utilized, and 2) controlling the physical disk. The Monitor can be thought of as seeing all disks in the same manner; a change of disks requires only a change in DSKINT to provide the proper software interface between the physical device and the rest of the system.

All references made herein to addresses on the disk refer to the logical or relative addresses used by the system and not to any physical addressing scheme involving records, sectors, tracks, etc., that may pertain to a particular physical device. The basic unit which may be addressed is a logical disk block which consists of 200_8 36-bit words.

5.9.2.2 Storage Allocation Table (SAT) Blocks - There is a

storage allocation table on the disk, which reflects the current status of every addressable block on the disk. These SAT blocks are contained in a file with the name "*SAT* .SYS". This file may be used by any user, but can only be modified by the Monitor. Each addressable block on the disk is represented by one particular bit within the SAT blocks.

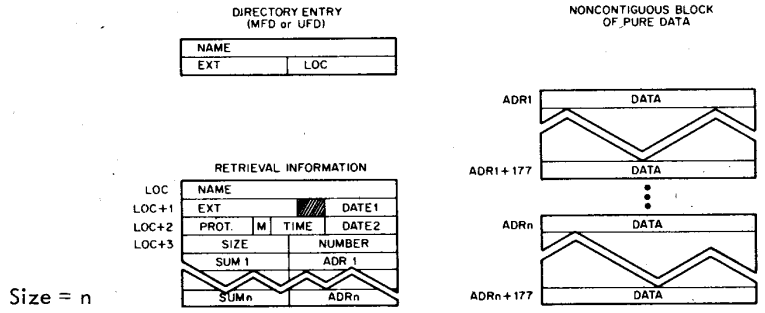
If a particular bit is on, it indicates that the corresponding block is filled with data (all blocks on the disk are filled when any information is written on them); if the bit is off, it indicates that the corresponding block is empty or available to be written on. The disk can be wiped out by zeroing the SAT blocks (which is exactly what is done when the disk is refreshed). The disk may optionally be "refreshed" whenever the Monitor is reloaded.

5.9.2.3 File Directories - There are two levels of directories on the disk; one is referenced mainly by the system and the other is referenced by individual users. There is only one higher level directory, known as the Master File Directory (MFD). One of the functions of the MFD is to serve as a directory for individual User's File Directories (UFD's). A UFD is a particular user's own directory and will contain the names of files he has written on the disk. The UFD itself is a file like any other file except that its filename is a binary number combination (project-programmer) rather than a 6-bit code and its extension is always UFD in SIXBIT. The binary combination consists of a left half, which is the project number, and a right half, which is called the programmer number. When a user is logged in under a specific project-programmer number and references the disk, he is actually referencing his own area through the UFD having his project-programmer number as its name. He may, of course, specifically code his

routine to reference files listed in the UFD's of other users or the MFD; whether he is successful or not will then depend upon the type of protection that has been specified for the file he is trying to reference.

5.9.2.4 File Format - All disk files (including MFD and UFDs) are composed of two parts: 1) pure data, and 2) 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. Word counts associated with individual blocks are not retained by the system. If such a partial block is input later, it will appear to have a full 200_8 data words.

There are three links in the chain by which the system references data on the disk. The first link is the 2-word directory entry in the UFD, which points to the Retrieval Information block(s), which in turn points to the individual pure data blocks. This chain is transparent to the user, who may look upon the directory as having 4-word entries analogous to DECtapes.



Directory Entry

NAME - Filename in 6-bit ASCII, unless the directory is the MFD and the file is a UFD; in that case, NAME is a project-programmer number in binary.

EXT - Filename extension in 6-bit ASCII; if NAME is a project-programmer number, EXT is UFD.

LOC - Address of the first block on the disk that contains Retrieval Information for this file.

Retrieval Information

NAME and EXT as above; used to check hardware for possible read error, and to check against software malfunctions. (A failure to match NAME and EXT results in the message "INCORRECT RETRIEVAL INFORMATION".)

DATE1 - In format of DATE UUO; date file last referenced (RENAME, or ENTER, or INPUT done) (bits 24-35).

DATE2 - Same format as DATE1; date file originally created (ENTER) (bits 24-35).

PROT. - Protection; see below (bits 0-8).

M - Data Mode (ASCII, Binary, Dump, etc.) (bits 9-12).

TIME - 24-hour time (in minutes) that file was originally created (bits 13-23).

SIZE - If negative, this portion indicates the number of words in the file, where all blocks with the possible exception of the last are assumed to contain a full 200_8 words. If positive, this is a count of the number of 200_8-word blocks contained in the file. For files of less than 2^{17} words, the negative word count is used; for larger files, the positive block count is used instead.

NUMBER - Programmer Number.

SUM1, ...SUMn - Checksum; two's complement, end-around-carry, sum of data in data-block whose disk address is ADR1.

ADR1, ...ADRn - Address of data block (logical block number on disk).

Protection

The first nine bits of the third word of a file's retrieval information are used to specify the protection of the file. This is a necessary procedure since the disk is shared by many users, who may each desire to keep certain files from being written over, read, or deleted by other users.

The total number of users falls into three categories:

a. Owner of file (person whose programmer number is the same as that in the right half of the NAME field of the UFD in which the file is entered).

b. Project members (users whose project number is the same as that in the left half of the NAME field of the UFD in which the file is entered).

c. All other users.

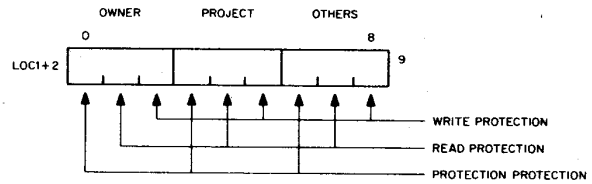
There are three types of protection against each of the three categories of users:

(1) Protection - The protection itself cannot be altered.

(2) Read protection - The file may not be read.

(3) Write Protection - The file may not be rewritten, renamed, or deleted.

The protection mask (see above) consists of the first nine bits of the third word of retrieval information; each bit (when on) represents a particular type of protection against a specific category of user, according to the following scheme. However, owner protection-protection and owner read-protection are ignored lest the file become totally inaccessible.



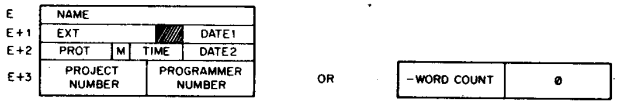
All files created with an ENTER are given the protection, 055_8 by the Monitor; if some other protection mask is desired, the the RENAME UUO must be employed by the user. (Also see Section 4.4.2.5, "File Protection".)

5.9.3 User Programming for the Disk

5.9.3.1 Format - The actual file structure of the disk is generally transparent to the user. In programming for input/output on the disk, a format analogous to that of DECtapes is used; that is, the user assumes a 4-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 input/output programmed operator and D specifies the user channel associated with this device. E points to a 4-word directory entry in the user's program which has the following format:



(Note that E+3 differs from the fourth word of retrieval information. See Retrieval Information, 5.8.2.4 for description.)

5.9.3.2 Special Functions of Programmed Operators (UUO's) -

ENTER D,E
error return

Causes the Monitor to store away the 4-word directory entry for later entry into the proper UFD when user channel D is CLOSEd or RELEASed.

NAME - The filename must be non-zero; if not, an error return results.

EXT - The file extension may be zero; if so, the Monitor will leave it zero.

DATE1 - The correct date is always filled in by the Monitor.

PROT - The protection is always supplied by the Monitor as 055. The RENAME may be used to change protection after file has been completely written and a CLOSE done.

M - The data mode is supplied by the Monitor as set by the user in the last INIT, or SETSTS UUO on channel D.

TIME, DATE2 - 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 non-zero, the Monitor will use the TIME and DATE2 supplied by the user in 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 will use 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, since UFDs are usually write-protected against 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 which type of error caused the return. These numbers have the following significance:

0 - E contained a zero file name

1 - E+3 contained an incorrect (or nonexistent) project-programmer number.

2 - File already exists, but is write-protected.

3 - File was being created, re-created, updated, or renamed.

No user, except an administrator with project number 1, may create a UFD, since the MFD is universally write-protected. The LOGIN CUSP (running under the administrator project number) creates a UFD for any user the first time he logs into the system.

When an ENTER is executed by the Monitor on a file that already 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, thereby retrieving space and making it available to any other user.

LOOKUP D, E
error return

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 will be read.

NAME - The filename in SIXBIT.

EXT - The file extension in SIXBIT. A zero extension is not treated in any special manner.

DATE1, PROT, M, TIME, DATE2 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 will use 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 that the file's protection mask permits reading.

The Monitor returns the negative word count (or positive block count for large files) in the LH of E+3, 0 in RH or E+3.

The numbers placed by the Monitor in the right half of E+1 upon an error return have a significance analogous to that described for the ENTER UUO:

0 - File was not found

1 - Incorrect project-programmer number in E+3

2 - Protection failure

3 - File was being created (no earlier version existed).

If the file is currently being re-created, the old file is used.

RENAME D, E
error return

This programmed operator is used to alter the name, extension, and/or protection of a file or to delete a file from the disk. Locations E through E+3 are as described above. RENAME is the only UUO that

can set the protection of a file to that specified in E+2. To be RENAMEd a file must first be CLOSEd on channel D, in order to identify for the RENAME UUO. RENAME then seeks out this file and 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 return numbers in the right half of E+1 are the same as for ENTER, with the added possibilities:

4 - Tried to RENAME file to already-existing name.

5 - Neither LOOKUP nor ENTER has been done to identify the file to be renamed.

USETO D, A

USETI D, A

These programmed operators are treated identically by the disk service routines. Their function is to notify the service routine that a particular block is to be used on the next INPUT or OUTPUT on channel D. A is a number that designates a particular block relative to the beginning of the file. If A is greater than the current size of the file (in blocks), the next OUTPUT will write a block immediately after the file; the next INPUT will cause the end-of-file flag to be set. A=1 refers to the first block of the file (i.e., block 0).

If A = 0 or if no previous LOOKUP or ENTER has been done, this UUO will set the improper mode error bit (see bit 18, Table 4-4, and Section 4.4.4).

5.9.3.3 General Notes - 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 did previously exist 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 this point, his 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 - see Section 5.8.2.2). 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, without creating an entirely new version of it; this third type of writing is called "updating" and eliminates the need to copy a file when making only 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.

When issuing a RENAME UUO, the user must insure that the status at locations E through E+3 are as he desires them to be. Since an ENTER or LOOKUP, as well as CLOSE, must have preceded the RENAME; the contents of E through E+3 will have been altered, or filled if the E is the same for all UUO's.

CALL [SIXBIT/RESET/] - Any files which are in the process of being written, but have not been CLOSEd or RELEASed, will be deleted and the space reclaimed. If a previous version of the file with the same name and extension existed, it will remain on the disk (and in the UFD) unchanged.

If the programmer wants to retain the newly created file and have the older version deleted, he must CLOSE or RELEAS the file before doing a RESET UUO.

5.10 INCREMENTAL PLOTTER

Device Mnemonic - PLT

Buffer Size - 43 (octal) words

The plotter takes 6-bit characters with the bits of each character decoded as follows:

Pen Raise	Pen Lower	-X Drum Up	+X Drum Down	+Y Carriage Left	-Y Carriage Right

Do not combine pen raise or lower with any of the position functions. (For more details on the incremental plotter, see the PDP-10 System Reference Manual, DEC-10-HGAA-D.)

5.10.1 Data Modes

5.10.1.1 A (ASCII)

Five, 7-bit characters per word are transmitted to the plotter exactly as they appear in the buffer. Since the plotter is a 6-bit device, the leftmost bit of each character is ignored.

5.10.1.2 AL (ASCII LINE) This mode is identical to the A mode.

5.10.1.3 I (IMAGE) Six, 6-bit characters per word are transmitted to the plotter exactly as they appear in the buffer.

5.10.1.4 B (BINARY) This mode is identical to the I mode.

5.10.1.5 IB (IMAGE BINARY) This mode is identical to the I mode.

5.10.1.6 DR (DUMP RECORDS) Not available.

5.10.1.7 D (DUMP) Not available.

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.

5.11 DISPLAY WITH LIGHT PEN (TYPE 30 and TYPE 340)

Device Mnemonic - DIS

Buffer Size - None (uses device-dependent dump mode only - 15)

5.11.1 Data Modes

5.11.1.1 ID (IMAGE DUMP - 15)

An arbitrary length area in the user area may be displayed on the scope. The command list format is as described in Chapter 4, "Unbuffered (Dump) Modes," with the addition for the Type 30 display, that, if RH = 0, and LH $\neq$ 0, then LH specifies the intensity for the following data (4 to 13).

5.11.2 Background

The purpose of the Monitor service routine for the VR-30 is to maintain a flicker-free picture on the display during time-sharing. To do this, the picture data must be available for display at least every two jiffies. This necessitates that the display data remain in core. At present, this means that the user program must also 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.

5.11.3 Display UUO's

The input/output UUO's 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 MANUAL
```

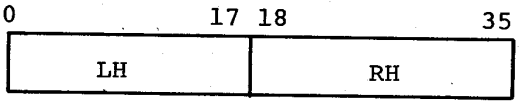
5.11.3.1 INPUT D, ADR

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.

5.11.3.2 OUTPUT D, ADR

ADR specifies the first address of a table of pointers. This table is composed of pointers with the following format:



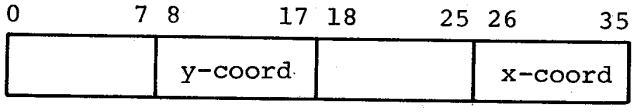
For the VR-30 Display:

If LH = 0 and RH = 0, then this is the end of the command list.

If LH ≠ 0 and RH = 0, then LH is the desired intensity for the following data or commands. The intensity ranges from 4 to 13, where 4 is the dimmest and 13 is the brightest.

If LH = 0 and RH ≠ 0, then RH is the address of the next pointer. Successive pointers are interpreted beginning at RH.

If LH ≠ 0 and RH ≠ 0, then -LH words beginning at address RH+1 are output as data to the display. The format of the data word is the following:



For the 340 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 340 programming 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 340 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
                                     ;(1000,-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 merely setting up a pointer to it. 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.

5.12 CALL AC, [SIXBIT/DEVCHR/] or CALLI AC, 4

The user may determine the physical characteristics associated with a logical device name by executing a DEVCHR UUO. The DEVCHR UUO returns the following information in the AC

referred.

$(AC)_L$:	1	Device can do output
	2	Device can do input
	4	Device has a directory (DTA or DSK)
	10	Device is a TTY
	20	Device is a magnetic tape
	40	Device is available to this job or is already assigned to this job
	100	Device is a DECtape
	200	Device is a paper tape reader
	400	Device is a paper tape punch
	1000	Device has a long dispatch table (that is, UUO's other than INPUT, OUTPUT, CLOSE, and RELEAS perform real actions)
	2000	Device is a display
	4000	TTY in use as an I/O device
	10000	TTY in use as a user console (even if detached)
	20000	TTY attached to a job
	40000	Device is a line printer
	100000	Device is a card reader
	200000	Device is a disk
	400000	DECtape directory is in core (this bit is cleared by an ASSIGN or DEASSIGN command to that unit)
$(AC)_R$:	400000	Device assigned by a console command
	200000	Device assigned by program (INIT UUO)
	Remaining Bits:	If bit 35-n contains a 1, then mode n is legal for the device.

NOTE

The mode number (0 through 17) must be converted to decimal; for example, mode 17_8 is represented by bit $35\text{-}15_{10}$ or bit 20.

APPENDIX 1

DECtape Compatibility Between DEC Computers

	Read By	PDP 4 550& 555 TU55	PDP 5 552& 555 TU55	PDP 6 551& 555& TU55	PDP 7 550& 555 TU55	PDP 8 552& 555 TU55	PDP 8 TC01 & TU55	PDP 8/I TC01 & TU55	PDP 9, TC01 & TU55	PDP 10 TC01 & 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
Written By	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 = Can not 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 - this must be done before writing.

NOTE: PDP-10 will not allow search to find first or last blocks when searching from the end-zone.

APPENDIX 2

Size of Multiprogramming non-disk Monitor (Reentrant 4 series, Version 50) June, 1969

There are three components to the Monitor:

1) Required code (4.7K)
2) Optional device code (0-4.4K)
3) Tables and buffers per job (73 words per job)

A. Required code (Assuming all features)

Lower core	96.
COMMON	409.
CLKCSS	82.
CLOCK1	367.
COMCON	1322.
CORE1	182.
DLSINT	48.
ERRCON	214.
SCNSRF	1260.
SEGCON	602.
SYSINT	78.
UUOCON	1144.
	4692. words (Decimal)

B. Optional devices

			Complete system
DTA	1284.	+N(1)*146. N(1) = 8	2612.
MTA	452.	+N(2)*9. N(2) = 2	470.
PTY	176.	+N(3)*10. N(3) = 2	196.
CDR	220.		220.
CDP	308.		308.
DIS	190.		190.
LPT	100.		100.
PLT	65.		65.

Optional devices		Complete system
PTP	167.	167.
PTR	105.	105.

3067. +N(1)=146.+N(2)*9.+N(3)*10. 4433.

C. Tables and buffers

18. words of tables per job

55. word of TTY device data block space per job

73. words per job

Total for complete 8 user system = 4692. + 443. + 8.*73. = 9709.

WARNING: The Monitor will continue to grow despite our best efforts to prevent it. Most new features are put in with conditional assembly so that a customer can reduce this size of the Monitor by giving up some of the new features.

These sizes are subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

APPENDIX 3

Size of Swapping Monitor (Reentrant 4 series, Version 50) June, 1969

There are three components to the Monitor:

1) Required code (10K)
2) Optional device code (0-4K)
3) Tables and buffers per job (1K for every 8 jobs)

A. Required code (Assuming all features)

Lower core	96.
COMMON	475.
CLOCK1	376.
COMCON	1592.
CORE1	214.
DLSINT	48.
DSKINT	130.
DSKSRB	2448.
ERRCON	211.
SCHEDB	741.
SCNSRF	1264.
SEGCON	709.
SYSINI	81.
UUOCON	1190.
	10375. words (Decimal)

B.

Optional devices				Complete system
DTA	1286.	+N(1)*146.	N(1) = 8	2454.
MTA	452.	+N(2)*9.	N(2) = 2	470.
PTY	166.	+N(3)*10.	N(3) = 2	196.
CDR	220.			220.
CDP	308.			308.
DIS	191.			191.
LPT	104.			104.

Optional Devices		Complete system
PLT	80.	80.
PTP	167.	167.
PTR	105.	105.
	3089. +N(1)=146.+N(2)*9.+N(3)*10.	4295.

C. Tables and buffers

21. words of tables per job

54. words of DSK device data block space per job (1.5 files/job)

55. word of TTY device data block space per job

130. words per job

Total for complete 16 user system = 10375. + 3987. + 16.*130. = 16442.

WARNING: The Monitor will continue to grow despite our best efforts to prevent it. Most new features are put in with conditional assembly so that a customer can reduce this size of the Monitor by giving up some of the new features.

For a complete Swapping System (all devices):

8	JOBS	15.7K
16	JOBS	16.7K
24	JOBS	17.7K
32	JOBS	18.7K
40	JOBS	19.7K
48	JOBS	20.7K
56	JOBS	21.7K
64	JOBS	22.7K

These sizes are subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

APPENDIX 4

Writing Reentrant User Programs

The LOADER simplification makes it somewhat more difficult to define variables and arrays. The easiest way to define them so that the resulting relocatable binary can be loaded on a one- or two-segment machine is to put them all in a separate subprogram as internal global symbols using Block 1 and Block N pseudo-ops. All other subprograms must refer to this data as external global locations. Most reentrant programs will 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.) Since programs are self-initializing, they clear the low segment when they are started even though the Monitor clears core whenever it assigns it to a user.

Using Block 1 and Block N pseudo-ops causes the LOADER to leave indications in the Job Data area (LH of JOBCOR) so that 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 done 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.

An Example of a Reentrant Program:

low segment subprogram:

```
TITLE LOW - EXAMPLE OF LOW SEGMENT SUB-PROGRAM

        JOBVER=137
        LOC     JOBVER
        3                   ;version3
        RELOC   Ø
        INTERNAL LOWBEG,DATA,DATA1,DATA2,TABLE,TABLE1

LOWBEG:
DATA:   BLOCK   1
DATA1:  BLOCK   1
DATA2:  BLOCK   1

TABLE:  BLOCK   1Ø
TABLE1: BLOCK   1Ø
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
```

Some reentrant programs require certain locations in the low segment to contain "constant" data which does not change during execution. Since the initialization of this data happens only once after each GET instead of after each START, programmers are tempted to place these "constants" in the same subprogram that contains the definition of the variable data locations. This action requires the SAVE command to write them out and the GET command to load them in again. Therefore the "constant" data should be moved by the programs

from the high segment to the low segment at the same time that 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.

A second way of writing single save file reentrant programs has been developed in which the source file can be a single 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 Ø 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 area 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 since DDT is a low segment relocatable file. The LOADER switch /B is used to protect the symbols. The usual way of assembly is reentrant so, 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 will know to start assigning buffers at the end of the data area in the low segment instead of at location 140. This is accomplished by chang-

ing the LH of JOBSA before the CALLI Ø (RESET) or changing the contents of JOBFF after the CALLI Ø, depending on how the program reinitializes itself on errors and upon completion. The program should not change these locations if it is assembled as non-reentrant. This is so that 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 -VØØ1
SUBTTL                T. HASTINGS      25 JUN 69
JOBVER=137
    LOC 137
    EXP ØØ1                         ;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  Ø                       ;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 14Ø>                  ;start data area at 14Ø in low seg
                                    ;if reentrant
DATAB:                              ;first location cleared every startup
DATA:       BLOCK 1
TABLE:      BLOCK 128
     .
     .
     .
DATAE:      END     BEG             ;define free location
```